arXiv:2304.00181v1 [math.NT] 1 Apr 2023

有限域上广义分圆映射的函数图

Alexander Bors[1] Daniel Panario[1] Qiang Wang[1]

2023年4月4日

摘要

The functional graph of a function $g : X \to X$ is the directed graph with vertex set X the edges of which are of the form $x \to g(x)$ for $x \in X$. Functional graphs are heavily studied because they allow one to understand the behavior of g under iteration (i.e., to understand the discrete dynamical system (X, g)), which has various applications, especially when X is a finite field $\mathbb{F}_q$. This paper is an extensive study of the functional graphs of so-called index d generalized cyclotomic mappings of $\mathbb{F}_q$, which are a natural and manageable generalization of monomial functions. We provide both theoretical results on the structure of their functional graphs and Las Vegas algorithms for solving fundamental problems, such as parametrizing the connected components of the functional graph by representative vertices, or describing the structure of a connected component given by a representative vertex. The complexity of these algorithms is analyzed in detail, and we make the point that for fixed index d and most prime powers q (in the sense of asymptotic density), suitable implementations of these algorithms have an expected runtime that is polynomial in $\log q$ on quantum computers, whereas their expected runtime is subexponential in $\log q$ on a classical computer. We also discuss four special cases in which one can devise Las Vegas algorithms with this kind of complexity behavior over most finite fields that solve the graph isomorphism problem for functional graphs of generalized cyclotomic mappings.

[1]数学与统计学院，卡尔顿大学，1125 Colonel By Drive，渥太华，安大略省 K1S 5B6，加拿大。第一作者邮箱：alexanderbors@cunet.carleton.ca；第二作者邮箱：daniel@math.carleton.ca；第三作者邮箱：wang@math.carleton.ca。作者获得加拿大自然科学与工程研究理事会资助，项目编号为 RGPIN-2018-05328（A. Bors 和 D. Panario）及 RGPIN-2017-06410（Q. Wang）。2020年数学主题分类：主要分类：11T22。次要分类：05C05、05C25、05C60、11A07、37P25。关键词和短语：分圆映射；有限动力系统；有限域；函数图；广义分圆映射。

1 引言

A *discrete dynamical system* is a pair (X, g) where X is a set and g is a function $X \to X$. The motivation behind this definition is to think of a complicated system that evolves in discrete time steps (such as a neural network), with X being the set of all states which the system can assume, and $g(x)$ being the successor state of $x \in X$. For this reason, one calls X the *state space* and g the *(state) transition function* of (X, g). When studying a discrete dynamical system (X, g), one is naturally interested in the behavior of g under iteration (i.e., in the function iterates g^n for $n \in \mathbb{N}_0 = \{n \in \mathbb{Z} : n \geq 0\}$). See the monograph [45] for a general introduction to discrete dynamical systems, and [45, Chapter 7] in particular for some examples of practical applications of them.

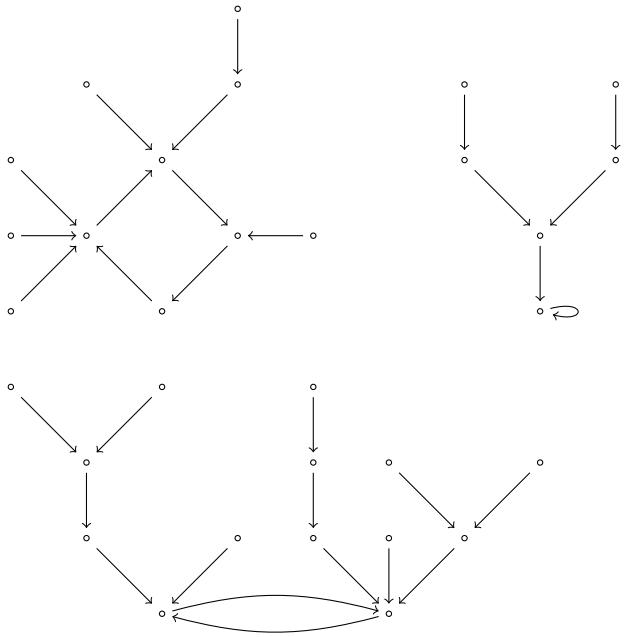
当 X 是有限的，也称 (X, g) 为有限动力系统。与应用相关的一些重要特殊情形包括：$X = \mathbb{Z}/m\mathbb{Z}$ 和 g 是模 m 的多项式（这用于 Pollard rho 算法 [55]）；或者 $X = \mathbb{F}_q^n$（有限域 $\mathbb{F}_q$ 的笛卡尔幂），其中文献特别关注 $q = 2$（见 [34, 41, 42, 50, 67]）。需要指出的是，通过在后者中固定一个 $\mathbb{F}_q$-基，可以将 $\mathbb{F}_q^n$ 与 $\mathbb{F}_{q^n}$ 等同，因此假设 $n = 1$（即仅将有限域本身视为状态空间）实际上并无一般性损失。

当 X 有限时，一个简单却重要的事实是，所有点 $x \in X$ 在 g 下都是预周期的，也就是说，存在唯一的最小整数 $\mathrm{pperl}_g(x) \geq 0$ 和 $\mathrm{perl}_g(x) \geq 1$，分别称为 x 在 g 下的预周期长度和周期长度，使得 $g^{\mathrm{pperl}_g(x)+\mathrm{perl}_g(x)}(x) = g^{\mathrm{pperl}_g(x)}(x)$；若 $\mathrm{pperl}_g(x) = 0$，则称 x 在 g 下是周期的（或 g-周期的）。由所有 g-周期点组成的 X 子集记作 $\mathrm{per}(g)$。X 中不是 g-周期的点称为在 g 下暂态的（或 g-暂态的）。文献 [21] 确定了随机函数 $X \to X$ 的若干随机参数，这些参数对于研究有限动力系统很有意义，例如一个点的期望预周期长度和周期长度。

可视化离散动力系统 (X, g) 的一种重要方式，尤其当 X 是有限集时，是使用所谓的 g 的函数图，记作 Γ_g。这是一个以 X 为顶点集的有向图，其中每个 $x \in X$ 都有一条弧（有向边） $x \to g(x)$，除此之外没有其他弧。很容易证明，以 X 为顶点集的有向图 Γ 是函数图（即，对于某个 $g : X \to X$，它具有 Γ_g 的形式）当且仅当 Γ 中的每个 $x \in X$ 的出度都为 1。

Particularly for *finite* functional graphs Γ_g, one can give the following precise characterization of their shape: A *connected component of* Γ_g is the induced subgraph of Γ_g on a subset of X that is the vertex set of a connected component of the underlying undirected graph of Γ_g. Each such connected component contains a single cycle of periodic points of g. Apart from those periodic points, the connected component consists precisely of those points which eventually map to the cycle after sufficiently many iterations of g – the *iterated pre-images (under g)* of points on the cycle. For each x on the cycle, the iterated pre-images of x form a directed rooted tree, with root x, that has all of its arcs oriented toward the root. Henceforth, for simplicity, whenever we say "(directed) rooted tree", it means "directed rooted tree in which all arcs are oriented toward the root". Here is a picture to illustrate the

情形:



Conversely, each finite digraph of the shape described above is a functional graph, as it is readily verified that all vertices in it have out-degree 1. The study of finite dynamical systems may be understood as the study of finite functional graphs. In this context, it is also noteworthy that in case ψ_1 and ψ_2 are permutations of a finite set, we have $\Gamma_{\psi_1} \cong \Gamma_{\psi_2}$ if and only if ψ_1 and ψ_2 are of the same *cycle type*, i.e., they have the same number of cycles of each given length. Formally, the cycle type of a permutation ψ of X, denoted by $\mathrm{CT}(\psi)$, is defined as the unique monomial in $\mathbb{Q}[x_n : n \in \mathbb{N}^+]$, where $\mathbb{N}^+ = \{n \in \mathbb{Z} : n \geq 1\}$, in which the degree of each variable x_n is the number of ψ-cycles of length equal to n. For example, if $X = \{1, 2, \ldots, 9\}$ and $\psi = (1, 2, 3)(4, 5)(6, 7)(8)(9)$, then $\mathrm{CT}(\psi) = x_1^2 x_2^2 x_3$. Cycle types (and the related notion of cycle indices) are well-studied in combinatorics, and studying isomorphism types of functional graphs may be seen as a natural generalization of this to arbitrary functions on finite sets.

近年来，有限域上某些函数类的函数图受到广泛关注，参见论文 [16, 46, 53, 56, 57, 65, 66] 及其中的参考文献。此外，论文 [16, 54] 并未明确讨论函数图，而是研究有限域上函数的迭代，其结果可以用函数图的术语重新表述。本文沿着这一研究方向，对下述意义上的所谓广义分圆映射的函数图进行研究。

Definition 1.1. *Let q be a prime power, and let $d \mid q-1$. A* generalized cyclotomic mapping of $\mathbb{F}_q$ of index d *is a function $f : \mathbb{F}_q \to \mathbb{F}_q$ with $f(0) = 0$ such that the restriction of f to each coset C_i of the unique index d subgroup of $\mathbb{F}_q^*$ agrees with a monomial function $x \mapsto a_i x^{r_i}$.*

More specifically, let ω be a primitive element of $\mathbb{F}_q$ (i.e., a generator of the cyclic multiplicative group $\mathbb{F}_q^*$), and let C be the index d subgroup of $\mathbb{F}_q^*$. The d cosets of C in $\mathbb{F}_q^*$ are of the form $C_i = \omega^i C$ for $i = 0, 1, \ldots, d-1$. The general form of an index d generalized cyclotomic mapping f of $\mathbb{F}_q$ is

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ a_0 x^{r_0}, & \text{if } x \in C = C_0, \\ a_1 x^{r_1}, & \text{if } x \in C_1, \\ \vdots & \vdots \\ a_{d-1} x^{r_{d-1}}, & \text{if } x \in C_{d-1}, \end{cases} \tag{1}$$

其中，对于 $i = 0$， $a_i \in \mathbb{F}_q$和 $r_i \in \{0, 1, \ldots, q-2\}$，1，.....， $d-1$。这些函数之所以有趣，是因为它们推广了单项式映射（其构成特殊情形 $d = 1$），同时仍然相对易于控制。从抽象代数的角度来看，值得注意的是，单项式函数 $\mathbb{F}_q^* \to \mathbb{F}_q^*$, $x \mapsto ax^r$ ，其中 $a \neq 0_{\mathbb{F}_q}$ 必然成立，是乘法群 $\mathbb{F}_q^*$ 上的仿射映射；也就是说，它们是 $\mathbb{F}_q^*$ 的群自同态（即幂函数$x \mapsto x^r$）与由固定群元素 a 所确定的一个（乘法）平移 $x \mapsto ax$ 的复合（另见定义 2.1.15）。因此，至少所有来自(1)的系数 a_i 均非零的广义分圆映射，都可以视为“逐陪集仿射”函数；在第6.4小节中，我们将探讨把本文的方法和结果推广到其他（可能是非阿贝尔）群的想法。在此背景下，我们还注意到，著名的考拉兹函数 $g:\ \mathbb{Z} \to \mathbb{Z}$，由公式

$$g(x) = \begin{cases} x/2, & \text{if } x \in 2\mathbb{Z}, \\ 3x+1, & \text{if } x \in 2\mathbb{Z}+1, \end{cases}$$

也是其各自定义域群 $\mathbb{Z}$上的陪集仿射函数。我们之所以能够在本文中建立一套理论来理解广义分圆映射在迭代下的行为（而与之类似的考拉兹函数任务仍然悬而未决），是因为广义分圆映射保持了将 $\mathbb{F}_q$ 划分为陪集 C_i 和单元素集合 $\{0_{\mathbb{F}_q}\}$ 的相关划分（并且，与此相关的是，它们在函数复合下构成一个半群）——另请参见定义 6.4.1(2,3) 中引入的两个概念之间的区别。

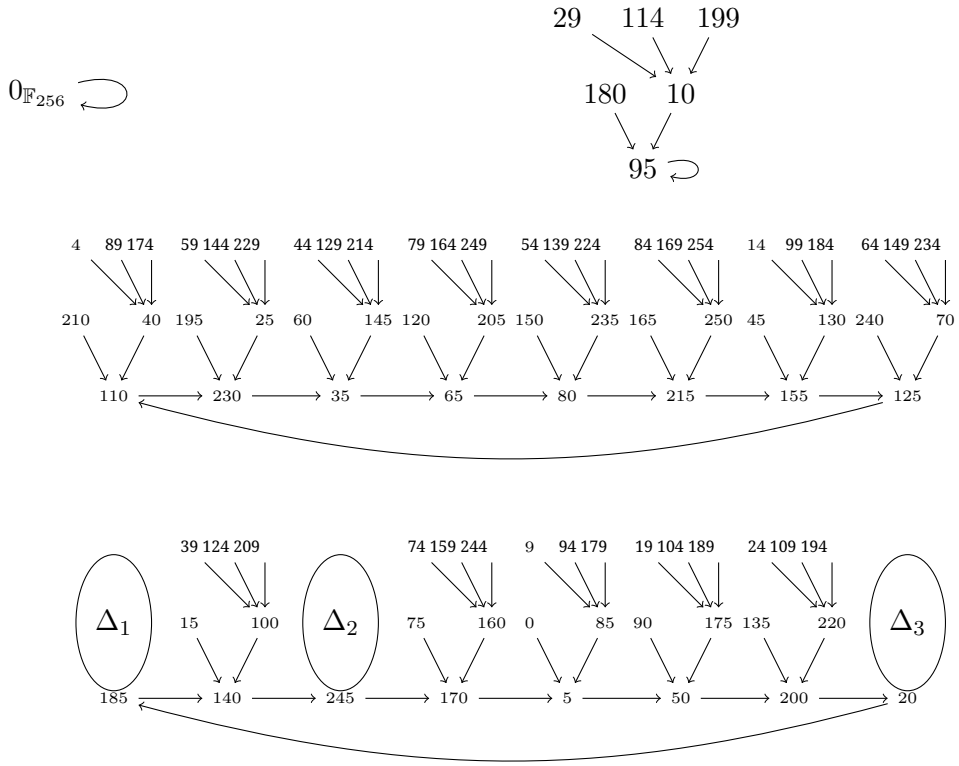
我们注意到，给定的 $\mathbb{F}_q$ 广义分圆映射可能有多个不同的指标，并且每个满足 $f(0) = 0$ 的函数 $f: \mathbb{F}_q \to \mathbb{F}_q$ 都是一个阶为 $q-1$ 的 $\mathbb{F}_q$ 广义分圆映射，尽管对广义分圆映射的研究大多集中在 d 的较小取值上。对于 $d = q-1$，处理广义分圆映射的已知方法，例如 [13, 算法 1], ，本质上都是简单的蛮力方法。除了 [13], 之外，广义分圆映射也在 [71, 72, 78] 中得到研究。一个重要的特殊情形是，对于所有 i，都有 $r_i = r$ ；此时称其为指标为 d 的 $\mathbb{F}_q$ 的 r 阶分圆映射，这些函数例如在 [52, 69, 70, 73] 中得到过研究。

Our goal in this paper is to develop algorithms that answer fundamental questions concerning the structure of the functional graph Γ_f of a given index d generalized

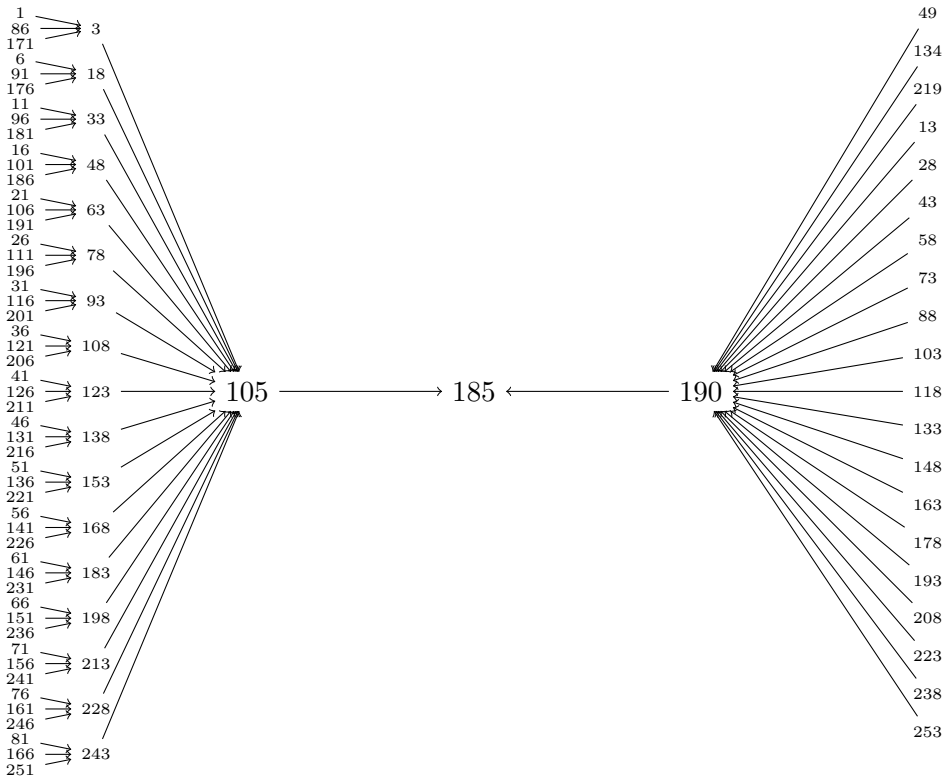
cyclotomic mapping f of $\mathbb{F}_q$, specified in the form (1). For example, let ω be any fixed primitive element of $\mathbb{F}_{256}$, and consider the following index 5 generalized cyclotomic mapping f of $\mathbb{F}_{256}$.

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ \omega^5 x^9, & \text{if } x \in C_0, \\ x^3, & \text{if } x \in C_1, \\ x^{17}, & \text{if } x \in C_2, \\ \omega^3 x^{34}, & \text{if } x \in C_3, \\ \omega^4 x^9, & \text{if } x \in C_4. \end{cases} \tag{2}$$

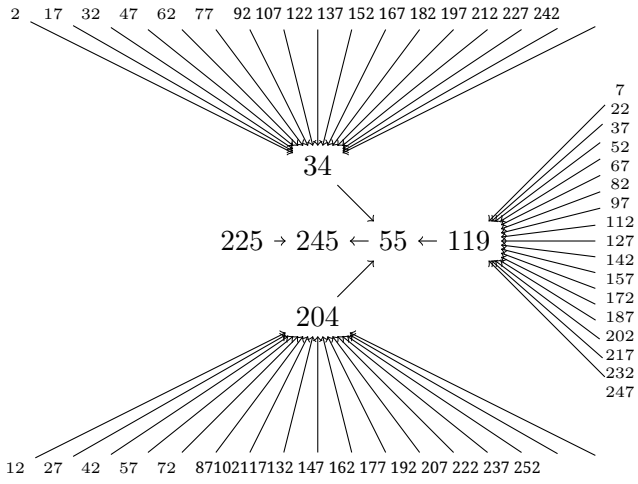
函数图 Γ_f 有 256 个顶点，其结构可以通过绘图来了解，下面我们就这样做。在这幅图中，标记为 $n \in \{0, 1, \ldots, 254\}$的顶点对应域元素 $\omega^n \in \mathbb{F}_q^*$（特别是，标签 0 对应域元素 $\omega^0 = 1_{\mathbb{F}_{256}}$），而表示域元素 0 的顶点则标记为 $0_{\mathbb{F}_{256}}$。结果发现，Γ_f 有四个连通分量，其中一个（按我们的绘图顺序为第四个）中，属于该连通分量的三个特定 f-周期点的 f-暂态迭代原像所形成的有根树相当大，因此单独绘出；在相应连通分量的示意图中，我们用 Δ_k 标记这些有根树，表示 $k \in \{1, 2, 3\}$。



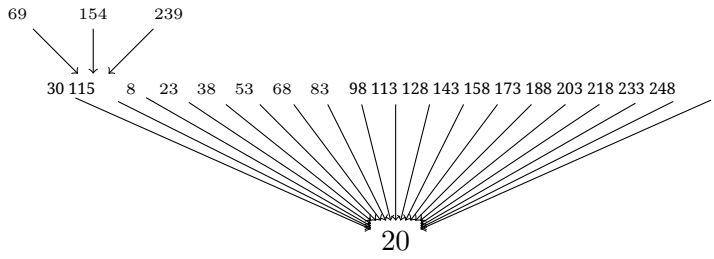
The rooted tree Δ_1 looks as follows.



The rooted tree Δ_2 looks as follows.



The rooted tree Δ_3 looks as follows.



Of course, this approach of understanding Γ_f by drawing it becomes intractable for large values of q, as its complexity is at least linear in q (i.e., exponential in $\log q$). The aim of our algorithms is to obtain an understanding of the structure of Γ_f without needing to draw it vertex by vertex. A detailed complexity analysis of those algorithms, which we carry out in Section 5, shows that for asymptotically almost every finite field and fixed index d, our algorithms have implementations with polynomial complexity (in $\log q$) on quantum computers, and implementations with subexponential complexity on classical computers. In the remainder of this introduction, we discuss the main ideas underlying our algorithms. We also note that we revisit the example (2) in Subsection 4.2, where we derive the structure of its functional graph with our methods.

理解任意函数 g：$X \to X$ 的函数图 Γ_g，首先要对 Γ_g 的连通分支进行适当的参数化。以下概念对此有所帮助。

定义 1.2. 设 X 为有限集，且 g： $X \to X$。 g 的循环代表元与长度列表（简称 CRL 列表）是一个（有限）集 $\mathcal{L} \subseteq X \times \mathbb{N}^+$，满足以下性质：

(1) 有序对的首项以 $\mathcal{L}$ 形式构成 g 在其周期点上的循环的一个代表元系。(2)若(r, l) $\in \mathcal{L}$，则 l 是 r 在 g 作用下的周期长度。

注 1.3。当 g 是有限集上的函数时，很容易根据 g的任意 CRL 列表 $\mathcal{L}$ 确定限制 $g_{|\mathrm{per}(g)}$ 的循环类型。也就是说，

$$\mathrm{CT}(g_{|\mathrm{per}(g)}) = \prod_{(r,l)\in\mathcal{L}} x_l.$$

因此， g 的 CRL 列表可以被视为比 CT($g_{|\mathrm{per}(g)}$) 更精细的信息。

我们从上文回顾，每个 Γ_g 的连通分支恰好包含 g 在其周期点上的一个循环。这意味着， g 的 CRL 列表也通过代表顶点给出了 g 的连通分量的参数化，同时还提供了每个代表元的循环长度这一基本信息。下面我们将进一步说明，当 $X = \mathbb{F}_q$ 和 g 是一个指标为 d 的广义分圆映射 f 时，如何获得 $\mathcal{L}$。

We already introduced the notation $C_i = \omega^i C$ for $i \in \{0, 1, \ldots, d-1\}$ to denote the cosets of C in $\mathbb{F}_q^*$. Let us additionally set $C_d := \{0_{\mathbb{F}_q}\}$. Then the sets C_i for

$i = 0, 1, \ldots, d$ form a partition of $\mathbb{F}_q$ that is preserved by f in the sense that f maps blocks of this partition to other such blocks (not necessarily surjectively). In other words, there is a unique function $\overline{f} : \{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$, which we call *induced by* f, such that $f(C_i) \subseteq C_{\overline{f}(i)}$ for each $i = 0, 1, \ldots, d$. We note in particular that $\overline{f}(d) = d$, and that $\overline{f}^{-1}(\{d\}) = \{d\}$ unless at least one of the coefficients a_i in (1) is 0.

令 $s := (q-1)/d = |C|$，我们可以通过双射 $\iota_i : \mathbb{Z}/s\mathbb{Z} \to C_i, x \mapsto \omega^{i+dx}$，将每个陪集 $C_i = \omega^i C$ （其中 $i = 0$ 为 $1, \ldots, d-1$ ）视为循环群 $\mathbb{Z}/s\mathbb{Z}$ 的一个副本（其底层集合为 $\{0, 1, \ldots, s-1\}$，群运算为模加法）。因此，f 可视为一个在 $\mathbb{Z}/s\mathbb{Z}$ 的各个副本（以及一个唯一的单元素块）之间映射的函数。更具体地说，若 $i \in \{0, 1, \ldots, d-1\}$ 和 $a_i \neq 0$，且记作$a_i = \omega^{e_i}$，则必有 $d \mid e_i + r_i i - \overline{f}(i)$，而 f 将 $\omega^{i+dx} \in C_i$ 映射到

$$a_i(\omega^{i+dx})^{r_i} = \omega^{e_i + r_i i + r_i dx} = \omega^{\overline{f}(i) + d\cdot\left(\frac{e_i + r_i i - \overline{f}(i)}{d} + r_i x\right)} \in C_{\overline{f}(i)}.$$

这意味着，在将 C_i 和 $C_{\overline{f}(i)}$ 识别为 $\mathbb{Z}/s\mathbb{Z}$ 后，f 在 C_i 上的限制对应于 $\mathbb{Z}/s\mathbb{Z}$ 的仿射函数 $A_i : x \mapsto r_i x + {}^{e_i + r_i i - \overline{f}}{}_{(i)\ d}$。

总之，f 本质上由在$\mathbb{Z}/s\mathbb{Z}$的多个副本之间映射的仿射函数组成，不过在对应系数 a_i 为 0 的情况下，$\mathbb{Z}/s\mathbb{Z}$ 的某些副本也可能被f恒定地映射到 $C_d = \{0\}$。我们注意到，如果所有 a_i 均非零，那么$f_{|\mathbb{F}_q^*}$将$\mathbb{F}_q^*$划分为陪集 C_i（其中 $i \in \{0, 1, \ldots, d-1\}$）的方式，与非本原置换 wreath 积 $\mathrm{Sym}(C) \wr \mathrm{Sym}(d)$ 的元素保持此划分的方式类似（其中 $\mathrm{Sym}(X)$ 和 $\mathrm{Sym}(n)$ 分别表示集合 X和 $\{0, 1, \ldots, n-1\}$上的对称群）。事实上，非本原置换 wreath 积的定义可以自然地推广为变换半群的非本原 wreath 积；在此推广下，$f_{|\mathbb{F}_q^*}$ 是 C^C（所有函数 $C \to C$构成的变换半群）与 $\{0, 1, \ldots, d-1\}^{\{0,1,\ldots,d-1\}}$的非本原 wreath 积中的一个元素，而 $\overline{f}_{|\{0,1,\ldots,d-1\}}$是$f$到 $\{0, 1, \ldots, d-1\}^{\{0,1,\ldots,d-1\}}$的投影。变换半群的 wreath 积此前已有研究，并且在代数自动机理论中发挥着核心作用，不过该理论采用的概念是本原置换 wreath 积 [32, 第 55 页及后续页]的自然推广。

无论如何，这些思想使我们能够轻松地将确定 f 上的 CRL 列表 $\mathcal{L}$ 归结为确定 $\mathbb{Z}/s\mathbb{Z}$ 上仿射函数的 CRL 列表 —— 详见第 3.1 小节。有限循环群上的仿射函数的 CRL 列表已在第 2.3 小节中确定。

我们算法方法的其余部分旨在理解：对于每个给定的 $(r, l) \in \mathcal{L}$，包含 r 的 Γ_f 连通分支的同构类型。我们回顾一下上文所述：这个连通分支实质上是通过将某些有向根树粘接到循环上的顶点而得到的。下面引入以下精确定义的记号。

定义 1.4. 设 Γ 是一个顶点集为 X 的有限函数图，并设 g 是满足 $\Gamma = \Gamma_g$ 的唯一函数 $X \to X$。对于每个 $x \in X$，我们将 $\mathrm{Tree}_\Gamma(x)$ 定义为 Γ 中位于 x 上方的树，如下所示。

(1) 如果 x 是 g-暂态，我们将 $\mathrm{Tree}_\Gamma(x)$ 定义为 Γ 在该集合上的诱导子图。

$$\{x\} \cup \{y \in \mathrm{V}(\Gamma) : y \neq x,\ and\ g^k(y) = x\ for\ some\ k = k(y) \geq 1\}.$$

(2) 如果 x是 g-周期的，我们将树$_\Gamma(x)$定义为 Γ 在该集合上的诱导子图

$$\{x\} \cup \bigcup \{\mathrm{V}(\mathrm{Tree}_\Gamma(y)) : y\ is\ g\text{-}transient\ and\ g(y) = x\},$$

并约定，在 $g(x) = x$的情况下，将 x 处的循环从树$_\Gamma(x)$中删除。

根据此定义，对于所有 $x \in X = \mathrm{V}(\Gamma_g)$，$\mathrm{Tree}^{\Gamma_g}(x)$ 都有定义；对于周期顶点 x，这些树需要与 g 中的循环粘合，以获得 Γ_g 的完整连通分量。

项链是组合学中研究十分广泛的概念。让我们来考虑由单个有向循环构成的顶点标记有向图（我们称此类图为项链图）。直观地说，可以把顶点想象成项链（通常意义上的项链）上的珠子，而顶点标签则表示珠子的颜色。顶点标记有向图的同构是保持顶点标签的有向图同构，而项链则是项链图在顶点标记有向图同构下的一个同构类。若 $\vec{\mathfrak{x}} = (\mathfrak{x}_0, \mathfrak{x}_1, \ldots, \mathfrak{x}_{L-1})$ 是一个长度为 L、元素取自集合 $\mathfrak{X}$ 的序列，则我们用 $[\vec{\mathfrak{x}}] = [\mathfrak{x}_0, \mathfrak{x}_1, \ldots, \mathfrak{x}_{L-1}]$ 表示 $\vec{\mathfrak{x}}$ 在循环群 $\mathbb{Z}/L\mathbb{Z}$ 对 $\mathfrak{X}^L$ 的自然作用下的轨道。因此，$[\vec{\mathfrak{x}}]$ 由那些能够通过对 $\vec{\mathfrak{x}}$ 进行循环移位而得到的、取值于 $\mathfrak{X}$ 的长度为 L 的序列构成。我们也称 $[\vec{\mathfrak{x}}]$ 为与 $\vec{\mathfrak{x}}$ 相关联的循环序列。我们注意到，两个项链图同构，当且仅当它们的顶点标签序列循环等价。因此，在组合学中，项链通常直接定义为一个循环序列（字符串的循环等价类）。

函数 $g : X \to X$的函数图 Γ_g 的连通分量，其中 X 是一个有限集，可以视为项链图。事实上，我们将给定连通分量中包含的唯一有向循环作为相应项链图的底层有向图。循环上某个（代表元）顶点 r 的标签定义为树$^{\Gamma_g}(r)$的有根树同构类型。例如，如果我们记作

- $\mathfrak{I}_0$ 由不含弧的单个顶点构成的平凡有根树的有向图同构类型；
- $\mathfrak{I}_1$ 在上述示例广义分圆映射 f 的函数图中周期顶点上方最常见的有根树同构类型，该映射定义于上文（2）中（即，一个高度为 2 的有根树，其中根的入度为 2，且根的两个邻接点中一个的入度为 0，另一个的入度为 3）；
- $\mathfrak{I}_2$ 上述示例中 Δ_3 的有向图同构类型（为 $\mathfrak{I}_j$ 设置的看似混乱的编号是特意选择的，使其与第4.2小节中的表6相匹配）；
- $\mathfrak{I}_3$ the digraph isomorphism type of Δ_1;

- $\mathfrak{I}_4$ Δ_2的有向图同构类型；

then the four connected components of the example above may be identified with necklace graphs corresponding to the following cyclic sequences of rooted tree isomorphism types (in order of drawing).

- $[\mathfrak{I}_0]$;
- $[\mathfrak{I}_1]$;
- $[\mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1]$;
- $[\mathfrak{I}_3, \mathfrak{I}_1, \mathfrak{I}_4, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_2]$。

按照上述将函数图的连通分量与某些项链图相对应的约定，有限函数图的两个连通分量作为有向图同构，当且仅当它们作为项链图同构（即它们表示相同的有根树同构类型项链）。这意味着，要理解 Γ_g 的连通分量，我们需要理解相关的有根树同构类型循环序列。

我们注意到，如果目标是理解函数图的一个连通分量的底层无向图的（无向图）同构类型，也可以采用类似的方法。只需将项链图替换为手镯图（无向的顶点标记循环图），将循环序列替换为二面体序列（即二面体群在 $\mathfrak{X}^L$ 上的自然作用所产生的轨道，其中生成反射通过倒序书写序列来作用；该二面体群的阶为 $2L$），并将项链替换为手镯（手镯图的同构类）。

接下来，我们将说明如何通过有根树同构类型的项链，来理解 Γ_g 的连通分量的有向图同构类型；这里假设 $X = \mathbb{F}_q$ 和 g 是 $\mathbb{F}_q$ 的指标为 d 的广义分圆映射 f。为此，我们首先需要理解给定（周期）点上方的有根树。基本思路是为每个陪集 C_i 构造一个特定的划分 $\mathcal{P}_i$，以“控制”其每个块中顶点上方的有根树同构类型。一次处理整块顶点至关重要，这样才能确保我们的算法复杂度不会至少与顶点数 q 成线性关系；许多用于处理图同构的通用算法都是如此，包括 Babai 的突破性准多项式算法（发表于 [8]）。

为了概述如何构造所述的 C_i 的划分 $\mathcal{P}_i$，我们需要介绍一些更多的概念。对于给定的正整数 m，我们将 m-(不)同余式定义为形如 $\nu(x \equiv \mathfrak{b} \text{ (模 } \mathfrak{a}))$的(不)同余式，其中$\nu \in \{\emptyset, \neg\}$是一个“逻辑符号”，而 $\mathfrak{a}$、$\mathfrak{b}$是满足 $\mathfrak{a} \geq 1$和 $\mathfrak{a} \mid m$的整数。我们将这两个概念统称为 m-同余条件，简称 m-CC。接下来，我们考虑 $\mathbb{Z}/m\mathbb{Z}$的算术划分这一概念，见下述定义的第(2)点。

定义 1.5。令 m 为正整数。我们将 $\mathbb{Z}/m\mathbb{Z}$ 的元素与其在 $\{0, 1, \ldots, m-1\}$ 中的标准代表元等同起来。

(1) 令 $x \equiv \mathfrak{b}_j (\text{模 } \mathfrak{a}_j)$（对于 $j = 1$、2、.....、K）为 m-同余式。存在 $\mathbb{Z}/m\mathbb{Z}$ 的唯一划分，我们将其记为 $\mathfrak{P}(x \equiv \mathfrak{b}_j (\text{模 } \mathfrak{a}_j)\colon\ j = 1、2、\ldots..、K)$,

使得该划分的每个块都是模 m的一个形如.....的 m-CC系统的解集

$$\begin{aligned}&\nu_1(x \equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1})\\&\nu_2(x \equiv \mathfrak{b}_2 \pmod{\mathfrak{a}_2})\\&\quad\vdots\\&\nu_K(x \equiv \mathfrak{b}_K \pmod{\mathfrak{a}_K})\end{aligned} \tag{3}$$

其中 ν_1、ν_2、.....、$\nu_K \in \{\emptyset, \neg\}$ 是逻辑符号。

(2) 划分 $\mathcal{P}$ （针对 $\mathbb{Z}/m\mathbb{Z}$ ）被称为算术运算，如果它具有以下形式

$$\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K)$$

对于适当的非负整数 K 以及适当的 m 个同余式 $x \equiv \mathfrak{b}_j$（模 $\mathfrak{a}_j$），其中 $j = 1$ 为 2,. . . , K。如果 $\mathcal{P} = \mathfrak{P}$ （ $x \equiv \mathfrak{b}_j$ （模 $\mathfrak{a}_j$）： $j = 1, 2, \ldots, K$ ），那么我们也称 $\mathcal{P}$ 为由同余式 $x \equiv \mathfrak{b}_j$（模 $\mathfrak{a}_j$）所张成的 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分，其中 $j = 1$ 为 $2, \ldots, K$。

(3) 当 $\mathcal{P}$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分时，使得 $\mathcal{P}$ 能由适当选取的 K 个 m-同余式张成的 $K \in \mathbb{N}_0$ 的最小值称为 $\mathcal{P}$ 的（算术）复杂度，记作 AC($\mathcal{P}$)。

(4) 当 $\mathcal{P}$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分，且已为 $\mathcal{P}$ 固定了一个生成同余式序列 $(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j})_{j=1,\ 2,\ \ldots,\ K}$ 时，我们还将对于每个 $\vec{\nu} = (\nu_1, \nu_2, \ldots, \nu_K) \in \{\emptyset, \neg\}^K$ 所对应的 $\mathbb{Z}/m\mathbb{Z}$ 的唯一子集记作 $\mathcal{B}(\mathcal{P}, \vec{\nu})$；该子集是方程组 (3) 的解集（只要非空，该解集就是 $\mathcal{P}$ 的一个块）。

注 1.6。 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分数量远少于其划分总数。事实上， $\mathbb{Z}/m\mathbb{Z}$ 的（集合）划分总数是贝尔数 B_m，其满足

$$B_m \sim \frac{1}{\sqrt{m}} \left(\frac{m}{W(m)} \right)^{m+\frac{1}{2}} \exp\left(\frac{m}{W(m)} - m - 1 \right)$$

当 $m \to \infty$ 时，其中 $W(m) \sim \log\ m$ 是朗伯 W 函数（见 [44, 第 1.14 节，问题 9]）。特别地，当 $m \to \infty$ 时，

$$\log B_m \sim -\frac{1}{2} \log m + \left(m + \frac{1}{2} \right) (\log m - \log W(m)) + \frac{m}{W(m)} - m - 1 \sim m \log m.$$

另一方面， $\mathbb{Z}/m\mathbb{Z}$ 的每个算术划分都由一组选取的同余式 $x \equiv \mathfrak{b} \pmod{\mathfrak{a}}$ 张成，其中 $\mathfrak{a}$ 遍历 m 的正因数，且 $\mathfrak{b} \in \{0, 1, \ldots, \mathfrak{a} - 1\}$。由于此类同余式的总数为$\sigma(m)$（即 m 的所有正因数之和），因此 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分数至多为 $2^{\sigma(m)}$，所以其自然对数至多为

$$\log 2 \cdot \sigma(m) \leq \log 2 (\mathrm{e}^{\gamma} + \epsilon) m \log\log m,$$

其中 γ 表示欧拉–马歇罗尼常数，第二个界由 Robin 的一项结果 [58] 得出。

让我们回到 d f $\mathbb{F}_q$ 的广义分圆映射。回想一下， $s=(q-1)/d$ 表示 $\mathbb{F}_q^*$ 的指标为 d 的子群 C 的阶（大小）。上述陪集 C_i 的划分 $\mathcal{P}_i$ 是作为 $\mathbb{Z}/s\mathbb{Z}$ 的算术划分构造的，并通过上述引入的双射 ι_i 将其与 C_i 对应起来。它们具有如下性质：对于从 $\mathcal{P}_i$ 的同一块 $\mathcal{B}$（$\mathcal{P}_i$， $\vec{\nu}$）中选取的顶点 x、$y \in C_i$，有 Tree^{Γ_f}（x） $\cong$ Tree^{Γ_f}（y），而这个共同的同构类型记作 Tree_i（$\mathcal{P}_i$， $\vec{\nu}$）。这些划分 $\mathcal{P}_i$ 以及相关有根树同构类型 Tree_i（$\mathcal{P}_i$， $\vec{\nu}$）的构造详见第 3.3 小节，其依据是两个关键工具：

- 第 2.2 小节中的基本结果引理 2.2.2；以及
- 对在所有陪集的并集上 $\Gamma_{\text{每个}}$ Γ_f 的诱导子图中的有根树结构有明确的理解；这一理解在第 3.2 小节中建立，同时也基于第2.1小节中建立的早期理论，其中 C_i 是 $\overline{f}$-周期的 i。

一旦显式构造出 $\mathcal{P}_i$ 和树$_i(\mathcal{P}_i,\vec{\nu})$，为了理解以周期顶点 r为代表元的 Γ_f 的连通分支的同构类型，就需要了解循环如何经过相应陪集划分中的各个块。当然，如果 r 在f 作用下的循环长度 l 较小，就可以直接通过穷举列出循环上的点，检查它们分别位于哪些块中，并写出相应的有根树循环序列；我们在第4.2小节示例的末尾就是这样做的。不过，如果 l 较大，则可以通过某个算术划分元组，对有根树循环序列作更简洁的描述；其中的块表示 r 的循环与所涉及算术划分 $\mathcal{P}_i$中的块的交集。详情请参见第 3.4 小节，该小节建立在第2.4小节的基础上。

以下是我们理解 Γ_f 的方法概述。

(1) 确定诱导函数 $\overline{f}$: $\{0,1,\ldots,d\} \to \{0,1,\ldots,d\}$，并将 f 重写为一组在 $d+1$ 集合 C_i 之间映射的仿射函数，每个函数的形式为 $\mathbb{Z}/s\mathbb{Z}$ 或 $\{0\}$。 (2) 按照第 3.1 小节的规定，计算 f 的 CRL 列表 $\mathcal{L}$；其依据是第 2.3 小节中关于有限循环群仿射映射的结果。 (3) 对每个 $i \in \{0,1,\ldots,d-1\}$，按照第 3.3 小节的规定，计算算术划分 $\mathcal{P}_i$ 及其对应的有根树同构类型 $\mathsf{Tree}_i(\mathcal{P}_i,\vec{\nu})$，以及 $\mathsf{Tree}^{\Gamma_f}(0_{\mathbb{F}_q})$ 的同构类型。这需要运用第 2.1 小节和第 3.2 小节中建立的理论。(4) 对每个 $(r,l) \in \mathcal{L}$，理解 r 在 f 作用下沿循环所对应的有根树同构类型循环序列，或者说

- 通过穷举列出 r 循环上的元素，然后查找它们属于相关算术划分的哪些块，或者
- 遵循第 3.4 小节中的方法，该方法依赖于第2.4小节。

在第 5 节中，我们对算法复杂度进行详细分析，并将步骤 (2)–(4) 中描述的每个过程都视为一个单独的算法来分析。

We note that in general, f has too many cycles in order for it to be possible to spell out a CRL-list of f element-wise if the procedure is to be efficient (i.e., subexponential in $\log q$); one can, however, obtain a concise parametrization of a CRL-list of f efficiently. Likewise, the approach described in point (4) can be carried out for each given pair (r, l) individually in an efficient manner for asymptotically almost all finite fields $\mathbb{F}_q$, but it is not clear in general how to obtain a "global" understanding of Γ_f efficiently. In fact, the number of distinct isomorphism types of connected components of Γ_f might be super-polynomial in $\log q$ even for fixed d (cf. Problem 6.3.3), so one would first need to come up with a compact way of parametrizing those isomorphism types. Still, as we will see in Subsection 5.3, for some special cases of generalized cyclotomic mappings f of $\mathbb{F}_q$, there are algorithms for describing Γ_f as a whole which are efficient for all or at least for "most" q (in an asymptotic density sense). In particular, in those cases, it can be efficiently decided whether the functional graphs of two given generalized cyclotomic mappings of $\mathbb{F}_q$ are isomorphic.

第 6 节通过列出有待进一步研究的开放问题来总结本文。为方便读者，附录（位于论文末尾）中的表 10 和表 11 列出了本文中出现的记号和术语，并提供了详尽的索引。

2 预备知识

本节证明了一些辅助结果，这些结果将在第 3 节讨论算法细节时用到。

2.1 有限群上的仿射映射的函数图

本小节中，我们推导有限循环群 $\mathbb{Z}/m\mathbb{Z}$ 上仿射映射 $A : x \mapsto ax + b$ 的函数图的一些结果。我们注意到，Deng [19]，早先研究过这些图，因此在适当之处，我们会使用 Deng 的若干结果和思路。不过，为方便读者，我们力求使本文的论述自成一体。我们还将在本小节末尾指出，这些结果实际上可以推广到任意有限群。首先，我们考虑以下概念。

定义 2.1.1. 设 $(\Gamma_j)_{j\in I}$ 是一个有向图族。它们的张量积记作 $\bigotimes_{j\in I}\Gamma_j$，是一个顶点集为 $\prod_{j\in I} V(\Gamma_j)$ 的有向图；当且仅当对每个 $j \in I$，Γ_j 中都存在一条弧 $y_j \to z_j$ 时，顶点 $(y_j)_{j\in I} \to (z_j)_{j\in I}$ 之间存在一条弧。

这一概念对应于第2]节中 Deng 的 [19, 公式（1）所定义的乘积图。

注 2.1.2. 若 $(g_j)_{j\in I}$ 是一族函数 $g_j\colon X_j \to X_j$，且若 $\bigotimes_{j\in I} g_j$ 表示该函数

$$\prod_{j\in I} X_j \to \prod_{j\in I} X_j, \quad (y_j)_{j\in I} \mapsto (g_j(y_j))_{j\in I},$$

then

$$\Gamma_{\bigotimes_{j\in I} g_j} = \bigotimes_{j\in I} \Gamma_{g_j}.$$

Due to Remark 2.1.2, the tensor product of digraphs is a useful tool when studying functional graphs of affine maps (in particular of endomorphisms) of finite cyclic groups $\mathbb{Z}/m\mathbb{Z}$. Indeed, if we factor $m = p_1^{v_1} \cdots p_K^{v_K}$, then for each given affine map $A : x \mapsto ax+b$ of $\mathbb{Z}/m\mathbb{Z}$ and each $j \in \{1, 2, \ldots, K\}$, we may consider the reduction A_j of A modulo $p_j^{v_j}$, which is the affine map $A_j : x \mapsto ax + b$ of $\mathbb{Z}/p_j^{v_j}\mathbb{Z}$. By the Chinese Remainder Theorem, $\mathbb{Z}/m\mathbb{Z}$ is in a natural isomorphism with $\prod_{j=1}^{K} \mathbb{Z}/p_j^{v_j}\mathbb{Z}$, and under this isomorphism, A corresponds to $\bigotimes_{j=1}^{K} A_j$. Hence we obtain the following, which is [19, Theorem 2].

引理 2.1.3. 设 $m = p_1^{v_1} \cdots p_K^{v_K}$ 为一个正整数，其显示的因式分解由两两互素的素数幂组成。设 $A: x \mapsto ax + b$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的一个仿射映射，并记 A_j 为 A 模 $p_j^{v_j}$ 的约化，其中 $j = 1$、2、.....、K。那么 $\Gamma_A \cong \bigotimes_{j=1}^{K} \Gamma_{A_j}$。

引理 2.1.3 使我们能够将关于 $\mathbb{Z}/m\mathbb{Z}$ 上仿射映射的函数图的许多论证，化归到 $m = p^v$ 为素数幂的情形。我们注意到以下有趣的二分现象（另见 [19, 引理 3]）。

命题 2.1.4. 设 $m = p^v$ 为一个素数幂，并设 $A: x \mapsto ax + b$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的一个仿射映射。

(1) 如果 $p \mid a$，那么 A 恰有一个周期点 x （必定是不动点），并且Γ_A 由树$\Gamma_A(x)$在根 x处添加一个循环得到。(2) 如果 $p \nmid a$，那么 A 是 $\mathbb{Z}/m\mathbb{Z}$的一个置换，因此 Γ_A 是若干互不相交的有向圈的并。

我们希望利用这些约化思路来证明以下定理。

定理 2.1.5。设 $A: x \mapsto ax + b$ 是有限循环群$\mathbb{Z}/m\mathbb{Z}$的仿射映射。那么， Γ_A 中周期顶点上的所有树彼此同构。事实上，它们都同构于 Γ_{μ_a}中任意周期顶点上的树，其中 μ_a 是 $\mathbb{Z}/m\mathbb{Z}$的端同态 $x \mapsto ax$ 。

我们注意到，Sha [61, 推论 3.4] 证明了 Γ_{μ_a} 中周期顶点上的所有树彼此同构。定理 2.1.5 的陈述本身隐含在 Deng 对 [19, 定理 11] 的证明中。

利用命题 2.1.4，我们可以迅速推出以下部分结果；在其证明中，我们使用记号。

$$\nu_p^{(v)}(n) := \min\{v, \nu_p(n)\}$$

其中，$\nu_p(n)$ 是 n 的 p-进赋值，即整数 n 的素数幂因式分解中 p 的指数；若 $n = 0$, 则定义为 ∞ 。

引理 2.1.6。定理 2.1.5 在 $m = p^v$ 是素数幂时成立。

证明。根据 [19, 引理 4], 若 $\nu_p^{(v)}(a-1) \le \nu_p^{(v)}(b)$，则 A 有一个不动点，这将导致 Γ_A 与 Γ_{μ_a} 之间的有向图同构（另见我们的引理2.3.3）。因此，根据 [61, 推论 3.4]，结果显然成立。

另一方面，如果 $\nu_p^{(v)}(b) < \nu_p^{(v)}(a-1)$，那么 $a \equiv 1$ (模 p)，这意味着 $p \nmid a$。因此，根据命题 2.1.4(2)， Γ_A 中周期顶点上的所有有根树都是平凡的（即，它们同构于没有弧的单个顶点），Γ_{μ_a} 中的那些树也如此，证毕。

□

当然，我们现在可以通过观察到“周期顶点上的所有有根树在对函数图取张量积时保持同构”这一性质，推导出完整强度的定理2.1.5。不过，经过一些额外工作，我们还可以得到一个更为详细的结果，下面将对此进行阐述。这需要用到第一作者论文中的一些概念和结果 [11], ，该论文刻画了有限群自同态的函数图中周期顶点上的树的结构（从而推广了 Sha 的结果 [61, 推论 3.4]）。

定义 2.1.7. 设 $\Gamma = (V, E)$ 是一个有限有向图，且 $x \in V$。

(1) 对偶有向图 Γ^* 由 Γ 通过反转每条弧得到；形式上， $\Gamma^* = (V, E^{-1})$，其中 $E^{-1} = \{(y, z) \in V^2 : (z, y) \in E\}$ 是 E的逆关系。(2) 若顶点 $y \in V$ 满足 Γ 有一条边 $x \to y$，则称 $y \in V$ 为 x的后继或子节点。(3) 对每个正整数 k， x的第 k 个繁衍数记作 $\mathrm{proc}_k(x) = \mathrm{proc}_k^{(\Gamma)}(x)$，它表示 x 的子节点 y 的数量，其中在 Γ 中存在一条长度为 $k-1$ 的有向路径 $(w_1, w_2, \ldots, w_k)$，满足 $w_1 = y$（在这种情况下，我们也说 y 具有（至少） $k-1$ 代后继）。(4) 若对于所有 y、$z \in V$ 以及满足 $\mathrm{proc}_k(y)$、$\mathrm{proc}_k(z) > 0$的所有正整数 k，都有 $\mathrm{proc}_k(y) = \mathrm{proc}_k(z)$，则称 Γ 具有刚性繁衍。

使用定义 2.1.7中的记号，我们注意到 $\mathrm{proc}_1(x)$ 就是 x 的所有子节点的数量（即 x 的出度），而$\mathrm{proc}_2(x)$ 是具有自身子节点的 x的子节点数量，等等。刚性繁衍意味着所有具有子节点的顶点必须具有相同数量的子节点（不过允许存在没有子节点的顶点），所有至少有一个“孙节点”的顶点必须具有相同数量的有子节点的子节点（特别是相同数量的孙节点），等等。以下事实曾在 [11, 定理 3], 之前的备注中被提及但未加以证明；为方便读者，我们在此处（引理 2.1.10 的证明之后）给出证明。

命题 2.1.8. 设 Γ 是一个有限有向图，且为函数图（即 Γ 的所有顶点的出度均为 1），记作 $\Gamma = \Gamma_g$。若对偶有向图 Γ^* 具有刚性繁衍，则对于任意两个 g-周期顶点 x 和 y，有 $\mathrm{Tree}_\Gamma(x) \cong \mathrm{Tree}_\Gamma(y)$。此外， Γ 中周期顶点 x 上方的共同有根树同构类型，仅由繁衍数序列 $(\mathrm{proc}_k(x))_{k \ge 1}$ 决定（即，它在任何具有刚性繁衍且周期顶点的繁衍数序列相同的有限函数图中都相同）。

读者若想了解如何从$(\mathrm{proc}_k(x))_{k\geq 1}$ 推导出树$_\Gamma(x)$的同构类型，对于具有刚性繁衍的周期顶点 x ，若 Γ^* ，可以在第4.1小节中找到详细内容。在证明命题2.1.8之前，我们将树$_\Gamma(x)$这一记号扩展到 Γ 为有限有向有根树 Δ （所有弧均朝向根）的情形；该记号已在第 1 节中为有限函数图 Γ 定义，并证明一个引理。

定义 2.1.9。设 Δ 是一棵有限有向有根树，其根为 $\mathrm{rt}(\Delta)$。我们注意到，除 $\mathrm{rt}(\Delta)$ 外，所有顶点的出度均为 1，并令 g 为唯一的函数 $\mathrm{V}(\Delta)\setminus\{\mathrm{rt}(\Delta)\}\to\mathrm{V}(\Delta)$，使得对于每个顶点 $x\neq\mathrm{rt}(\Delta)$， Δ 都有一条弧 $x\to g(x)$。对于每个 $x\in\mathrm{V}(\Delta)$，我们定义 $\mathrm{Tree}_\Delta(x)$，即 Δ 中位于 x 上方的树，作为 Δ 在集合上的诱导子图。

$$\{x\}\cup\{y\in\mathrm{V}(\Delta): y\neq x,\ and\ g^k(y)=x\ for\ some\ k=k(y)\geq 1\}.$$

在以下引理及后续内容的陈述中，我们将有限有向根树 Δ 的高度记为 $\mathrm{ht}(\Delta)$。

引理2.1.10。设 Δ_1 和 Δ_2 为有限有向根树。此外，我们假设 Δ_1 和 Δ_2 具有相同的高度，对偶有向图 Δ_1^* 和 Δ_2^* 都具有刚性繁衍，并且 $\mathrm{proc}^{(\Delta_1^*)}{}_h(\mathrm{rt}(\Delta_1)) = \mathrm{proc}^{(\Delta_2^*)}{}_h(\mathrm{rt}(\Delta_2))$ 对于 $1\leq h\leq \mathrm{ht}(\Delta_1)=\mathrm{ht}(\Delta_2)$成立。那么 Δ_1 和 Δ_2 同构。

证明。我们对 Δ_1 和 Δ_2 的公共高度进行归纳。若 $\mathrm{ht}(\Delta_1)=0$，则 Δ_1 和 Δ_2 均由一个没有任何弧的单个顶点组成，因此同构。现在假设 $\mathrm{ht}(\Delta_1)\geq 1$，且该结论对所有更小的高度均成立。注意， Δ_1 和 Δ_2 同构，当且仅当以下多重集相等，其中 $[\Gamma]_\cong$ 表示有限有向图 Γ 的同构类型:

$$\begin{aligned}&\{[\mathrm{Tree}_{\Delta_1}(y_1)]_\cong : y_1 \text{ is a child of } \mathrm{rt}(\Delta_1) \text{ in } \Delta_1^*\} =\\ &\{[\mathrm{Tree}_{\Delta_2}(y_2)]_\cong : y_2 \text{ is a child of } \mathrm{rt}(\Delta_2) \text{ in } \Delta_2^*\}.\end{aligned}\tag{4}$$

因此，我们的目标是证明等式 (4)。设定 $h\in\{0,1,2,\ldots,\mathrm{ht}(\Delta_1)-1\}$。在 Δ_1^* 中，$\mathrm{rt}(\Delta_1)$ 的子节点 y_1 中，使得 $\mathrm{Tree}_{\Delta_1}(y_1)$ 的高度恰好为 h 的数量是

$$\mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1))-\mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)).$$

至于子节点 y_2 ，$\mathrm{rt}(\Delta_2)$中的 Δ_2^* ，使得树$_{\Delta_2}(y_2)$ 的高度恰好为 h，同样可得其数量为

$$\mathrm{proc}_{h+1}^{(\Delta_2^*)}(\mathrm{rt}(\Delta_2))-\mathrm{proc}_{h+2}^{(\Delta_2^*)}(\mathrm{rt}(\Delta_2))=\mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1))-\mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)).$$

此外，对于 Δ_1^* 中 $\mathrm{rt}(\Delta_1)$ 的每个子节点 y_1 ，若 $\mathrm{Tree}_{\Delta_1}(y_1)$ 的高度恰好为 h，则在 $(\mathrm{Tree}_{\Delta_1}(y_1))^*$ 中， y_1 的前 h 个繁殖数与 Δ_1^* 中 $\mathrm{rt}(\Delta_1)$ 的相同，因为 Δ_1^* 具有刚性繁衍。特别地，根据归纳法假设，对于固定的 h，所有满足以下条件的有向图 $\mathrm{Tree}_{\Delta_1}(y_1)$ 都彼此同构： y_1 是 Δ_1^* 中 $\mathrm{rt}(\Delta_1)$ 的子节点，且在 Δ_1^* 中恰有 h 个后继代；它们的同构

type $\mathfrak{I}_h^{(1)}$ being determined by the first h procreation numbers of $\mathrm{rt}(\Delta_1)$ in Δ_1^*. An analogous statement holds with Δ_2 and $\mathrm{rt}(\Delta_2)$ in place of Δ_1 and $\mathrm{rt}(\Delta_1)$, say with isomorphism type $\mathfrak{I}_h^{(2)}$. But by assumption, the procreation numbers of $\mathrm{rt}(\Delta_1)$ in Δ_1^* and of $\mathrm{rt}(\Delta_2)$ in Δ_2^* are the same, whence $\mathfrak{I}_h^{(1)} = \mathfrak{I}_h^{(2)}$ for each h. This shows that the two multisets in formula (4) are the same, each consisting of exactly $\mathrm{proc}_{h+1}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1)) - \mathrm{proc}_{h+2}^{(\Delta_1^*)}(\mathrm{rt}(\Delta_1))$ copies of $\mathfrak{I}_h^{(1)}$ for each $h = 0, 1, \ldots, \mathrm{ht}(\Delta_1) - 1$. □

命题 2.1.8 的证明。首先，为了使断言有意义，我们注意到，序列 $(\mathrm{proc}^{(\Gamma^*)}{}_k(x))_{k\geq 1}$ 不依赖于周期顶点 x 的选择。事实上，若 y 是另一个周期顶点，那么沿着各自的循环，可以从 Γ^* 中的 x 或 y 出发，构造任意长的有向路径。这意味着，对每个 k，$\mathrm{proc}^{(\Gamma^*)}{}_k(x), \mathrm{proc}^{(\Gamma^*)}{}_k(y) > 0$，因此 $\mathrm{proc}^{(\Gamma^*)}{}_k(x) = \mathrm{proc}^{(\Gamma^*)}{}_k(y)$，因为 Γ^* 具有刚性繁衍。

令 $\Gamma = \Gamma_g$ 表示一个适当选取的函数 $g : \mathrm{V}(\Gamma) \to \mathrm{V}(\Gamma)$，并固定一个 g-周期顶点 x。注意，对于每个 $y \in \mathrm{V}(\mathrm{Tree}_\Gamma(x))$，都有 $\mathrm{Tree}_\Gamma(y) = \mathrm{Tree}_{\mathrm{Tree}_\Gamma(x)}(y)$。我们注意到

$$\mathrm{proc}_k^{(\mathrm{Tree}_\Gamma(x)^*)}(x) = \mathrm{proc}_k^{(\Gamma^*)}(x) - 1$$

对于每个 $k \geq 1$；这是因为由 $\mathrm{proc}^{(\Gamma^*)}{}_k(x)$ 计数的 x 的子节点中，恰好有一个是周期的，因此在树 $\Gamma(x)^*$ 的繁衍数中必须忽略。特别地，对于每个 $h \geq 0$，树 $\Gamma(x)^*$ 中满足树 $\Gamma(y)$ 的高度恰好为 h 的 x 的子节点 y 的数量为

$$\mathrm{proc}_{h+1}^{(\mathrm{Tree}_\Gamma(x)^*)}(x) - \mathrm{proc}_{h+2}^{(\mathrm{Tree}_\Gamma(x)^*)}(x) = \mathrm{proc}_{h+1}^{(\Gamma^*)}(x) - \mathrm{proc}_{h+2}^{(\Gamma^*)}(x).$$

此外，对于树 $\Gamma(x)^*$ 中 x 的每个子节点 y，Γ^* 具有刚性繁衍这一事实意味着树 $\Gamma(y)^*$ 也具有刚性繁衍。更具体地说，每当对于某个 $z \in \mathrm{V}(\mathrm{Tree}_\Gamma(y))$，$\mathrm{proc}^{(\mathrm{Tree}_\Gamma(y)^*)}{}_k(z) > 0$ 时，就有 $\mathrm{proc}^{(\mathrm{Tree}_\Gamma(y)^*)}{}_k(z) = \mathrm{proc}_k^{\Gamma^*}(z) = \mathrm{proc}_k^{\Gamma^*}(x)$。因此，引理 2.1.10 蕴含这些同构类型构成的多重集

$$\{[\mathrm{Tree}_\Gamma(y)]_\cong : g(y) = x, y \text{ is } g\text{-transient}\},$$

它决定了树 $\Gamma(x)$ 的同构类型，而这又完全由繁衍数序列 $(\mathrm{proc}^{(\Gamma^*)}{}_k(x))_{k\geq 1}$ 决定，这正是我们需要证明的。

□

鉴于命题2.1.8，以下结果，即 [11, 定理 2]，，既表明 Γ_{μ_a}（$x \mapsto ax$ 在 $\mathbb{Z}/m\mathbb{Z}$ 上的端同态的函数图）中的周期顶点上的有根树两两同构，又刻画了相应的有根树同构类型。

定理 2.1.11. 设 m 为正整数，且 $\mu_a : x \mapsto ax$ 是循环群 $\mathbb{Z}/m\mathbb{Z}$ 的一个自同态。对偶函数图 $(\Gamma_{\mu_a})^*$ 具有刚性的繁衍性质，并且对于每个 $k \in \mathbb{N}^+$ 以及 Γ_{μ_a} 的每个周期顶点 x，都有 $\mathrm{proc}_k(x) = |\ker^{(k)}(\mu_a) : \ker^{(k-1)}(\mu_a)|$，其中 $\ker^{(j)}(\mu_a) := \{y \in \mathbb{Z}/m\mathbb{Z} : (\mu_a)^j(y) = a^j y = 0\}$。

鉴于此，只要证明以下结果，定理 2.1.5 就显而易见。该结果使我们能够更一般地根据 A 模 m 的素数幂因子 $p_j^{v_j}$ 的约化所对应的函数图 Γ_{A_j} 的同构类型，推导出 Γ_A 的同构类型。

命题 2.1.12. 设 $g_j:\ X_j \to X_j$ for $j = 1, 2$ 是定义在有限集上的函数。

(1) 我们有 $\mathrm{per}(g_1 \otimes g_2) = \mathrm{per}(g_1) \times \mathrm{per}(g_2)$。特别地，如果 g_1 和 g_2 各自恰有一个周期点，那么 $g_1 \otimes g_2$ 也如此。(2) 假设对偶函数图 $\Gamma_{g_1}^*$ 和 $\Gamma_{g_2}^*$ 具有刚性繁衍。那么，对偶函数图

$$\Gamma_{g_1 \otimes g_2}^* \cong (\Gamma_{g_1} \otimes \Gamma_{g_2})^* = (\Gamma_{g_1})^* \otimes (\Gamma_{g_2})^*$$

具有刚性繁衍；并且若 y_j 是 g_j for $j = 1, 2$ 的周期点，则 $\vec{y} = (y_1, y_2)$ 是 $g_1 \otimes g_2$ 的周期点，且 $\mathrm{proc}_k(\vec{y}) = \mathrm{proc}_k(y_1) \cdot \mathrm{proc}_k(y_2)$。

(3) 记 $*$ 为 $\mathbb{Q}[x_n$ 中多项式的唯一 $\mathbb{Q}$-双线性乘积：$n \in \mathbb{N}^+]$，使得

$$(x_1^{e_1} \cdots x_N^{e_N}) * (x_1^{e'_1} \cdots x_{N'}^{e'_{N'}}) = \prod_{1 \le n \le N, 1 \le n' \le N'} x_n^{e_n} * x_{n'}^{e_{n'}}$$

and

$$x_n^e * x_{n'}^{e'} = x_{\mathrm{lcm}(n,n')}^{ee' \gcd(n,n')}.$$

我们假设每个 g_j 都是相应集合 X_j 上的一个置换。那么，$g_1 \otimes g_2$ 是 $X_1 \times X_2$ 的一个置换，其循环类型可以计算为 g_1 和 g_2 的循环类型的 $*$-乘积。

(4) 我们假设 g_1 恰好有一个周期点 y_1（必然是一个不动点），并且 g_2 是 X_2的一个置换。于是，$g_1 \otimes g_2$ 在 $\mathrm{per}(g_1 \otimes g_2)$上的诱导子图与 Γ_{g_2}同构，并且对于每个 $\vec{y} \in \mathrm{per}(g_1 \otimes g_2)$，都有 $\text{树}^{\Gamma_{g_1 \otimes g_2}}(\vec{y}) \cong \text{树}^{\Gamma_{g_1}}(y_1)$。

证明。由于 $g_1 \otimes g_2$ 是在 $X_1 \times X_2$ 上逐分量应用 g_1 和 g_2，显然，点 $(y_1, y_2) \in X_1 \times X_2$ 在 $g_1 \otimes g_2$ 下是周期的，当且仅当对于 $j = 1, 2$，y_j 在 g_j 下是周期的。这就证明了陈述（1），以及陈述（2）中关于 $\vec{y}$ 的第一个断言。

对于命题（2）的其余部分，我们进行如下处理。为了证明$(\Gamma_{g_1})^* \otimes (\Gamma_{g_2})^*$ 具有刚性繁衍，令 k 为一个正整数，并令 $\vec{y} = (y_1, y_2)$和$\vec{z} = (z_1, z_2)$为 $X_1 \times X_2$ 中的点，它们在该图中各自至少具有 k 个后继代。这等价于 y_1、y_2、z_1、z_2 中的每一个在相应的图 $\Gamma_{g_1}^*$ 或 $\Gamma_{g_2}^*$中至少具有 k 个后继代。由此可得，$\mathrm{proc}_k(y_1) = \mathrm{proc}_k(z_1)$ 且 $\mathrm{proc}_k(y_2) = \mathrm{proc}_k(z_2)$。现在，对于 $\vec{w} = (w_1, w_2) \in X_1 \times X_2$，繁衍数$\mathrm{proc}_k(\vec{w})$ 统计的是 $(\Gamma_{g_1})^* \otimes \vec{(\Gamma_{g_2})^*}$中 $\vec{w}$ 的子节点 $\vec{w'} = (w'_1, w'_2)$ 的数量，其中这些子节点至少具有 $k-1$ 个后继代。但是，w' 至少具有 $k-1$ 个后继代，当且仅当每个 w'_j 都是w_j 在 $\Gamma_{g_j}^*$ 中的子节点，并且至少具有 $k-1$个后继代。因此，$\mathrm{proc}_k(\vec{w}) = \mathrm{proc}_k(w_1) \cdot \mathrm{proc}_k(w_2)$。特别地，

$$\mathrm{proc}_k(\vec{y}) = \mathrm{proc}_k(y_1)\,\mathrm{proc}_k(y_2) = \mathrm{proc}_k(z_1)\,\mathrm{proc}_k(z_2) = \mathrm{proc}_k(\vec{z}),$$

如要求。

对于陈述（3），请参见 [77, 定理 2.4 及其证明]。

对于陈述（4），我们注意到这隐含在 [11, 定理 1(4)], 中，但由于那里未作详细证明，我们在此给出证明。根据陈述（1），我们知道 $\vec{z} = (z_1, z_2) \in X_1 \times X_2$ 是 $g_1 \otimes g_2$ 的周期点，当且仅当 $z_j \in \mathrm{per}(g_j)$ 对于 $j = 1$、2。因此， $g_1 \otimes g_2$ 的周期点通过 $y_2 \mapsto (y_1, y_2)$ 与 g_2 的周期点构成双射，并且此双射保持周期长度。因此，所断言的同构 $\Gamma_{g_2} \cong \Gamma_{(g_1 \otimes g_2)|\mathrm{per}(g_1 \otimes g_2)}$ 显然成立。最后，不难检查，对于每个 $y_2 \in \mathrm{per}(g_2)$，函数 $X_1 \to X_1 \times X_2$, $z_1 \mapsto (z_1, ((g_2)|_{\mathrm{per}(g_2)})^{-\mathrm{pperl}_{g_2}(z_1)}(y_2))$ 是树 $\Gamma_{g_1}(y_1)$ 与树 $\Gamma_{g_1 \otimes g_2}((y_1, y_2))$ 之间的有向图同构。 □

现在我们可以证明定理 2.1.5 了。事实上，我们证明它的以下加强版。

定理 2.1.13。设 m 为正整数，且令 $A:\ x \mapsto ax + b$ 为循环群 $\mathbb{Z}/m\mathbb{Z}$ 上的仿射映射。对偶函数图 $(\Gamma_A)^*$ 具有刚性繁殖，并且对于每个 $k \in \mathbb{N}^+$ 以及 Γ_A 的每个周期顶点 x，有 $\mathrm{proc}_k(x) = |\ker^{(k)}(\mu_a) \colon \ker^{(k-1)}(\mu_a)|$。特别地，$\Gamma_A$ 中周期顶点上方的每棵有根树都与 Γ_{μ_a} 中周期顶点上方的每棵有根树同构。

我们注意到，在定理 2.1.13的情形下，对于所有 $j \in \mathbb{N}_0$，都有 $|\ker^{(j)}(\mu_a)| =$ 最大公约数(a^j, m)，这使得繁衍数的计算（从而也使得理解周期顶点上的树的同构类型）变得容易。

定理 2.1.13 的证明。“特别地”这一陈述由命题 2.1.8 和定理 2.1.11 立即可得，因此我们专注于主要陈述的证明。

如上所述，我们将 $m = p_1^{v_1} \cdots p_K^{v_K}$ 因数分解，并将 A 模 $p_j^{v_j}$ 的约化记为 A_j。我们声称，对每个 $j = 1$、2、.....、K，对偶函数图 $(\Gamma_{A_j})^*$ 都具有刚性繁衍；并且对每个 A_j-周期顶点 $x \in \mathbb{Z}/p_j^{v_j}\mathbb{Z}$，$(\Gamma_{A_j})^*$ 中 x 的繁衍数序列与 μ_a 模 $p_j^{v_j}$ 的约化 $(\mu_a)_j$ 的对偶函数图中某个周期顶点的繁衍数序列一致。事实上，沿用引理 2.1.6 的证明可知，当 $\nu^{(v_j)}{}_{p_j}(a-1) \le \nu^{(v_j)}{}_{p_j}(b)$ 时（此时有向图 $(\Gamma_{A_j})^*$ 和 $(\Gamma_{(\mu_a)_j})^*$ 实际上整体同构），以及当 $\nu^{(v_j)}{}_{p_j}(a-1) > \nu^{(v_j)}{}_{p_j}(b)$ 时（此时 $(\Gamma_{A_j})^*$ 和 $(\Gamma_{(\mu_a)_j})^*$ 均为有向圈的不交并），结论都显然成立。

现在，我们回顾一下 $\Gamma_A = \bigotimes_{j=1}^K \Gamma_{A_j}$ 和 $\Gamma_{\mu_a} = \bigotimes_{j=1}^K \Gamma_{(\mu_a)_j}$。在两个张量积中，由 j 索引的因数分解的对偶均具有刚性繁衍，其周期顶点的繁殖数序列相同。因此，根据命题 2.1.12(2)，$(\Gamma_A)^*$ 与 $(\Gamma_{\mu_a})^*$ 也同样如此；再结合定理 2.1.11 中的繁殖数公式，主要结论便得以证明。 □

我们还注意到命题 2.1.12(1) 的以下推论，这一点稍后将变得很重要。

引理 2.1.14. 设 m 为正整数，设 a、$b \in \mathbb{Z}/m\mathbb{Z}$，并设 L 为一个足够大的正整数，使得 $\gcd(a^L, m) = \prod_{p|\gcd(a,m)} p^{\nu_p(m)}$（例如，$\mathrm{mpe}(m) :=$

$\max_p \nu_p(m) \le \lfloor \log_2 m \rfloor$ 是 L 的一个有效取值）。那么，当且仅当满足以下条件时，$y \in \mathbb{Z}/m\mathbb{Z}$ 是 $\mathbb{Z}/m\mathbb{Z}$ 上仿射映射 $A: z \mapsto az+b$ 的周期点

$$y \equiv \sum_{t=0}^{L-1} a^t \cdot b \pmod{\gcd(a^L, m)}.$$

证明。根据命题2.1.12(1)和中国剩余定理，点 $y \in \mathbb{Z}/m\mathbb{Z}$ 在 A 下是周期点，当且仅当对于每个素数 $p \mid m$，y 模 $p^{\nu_p(m)}$的约化y_p 在 A 模 $p^{\nu_p(m)}$的约化 A_p 下是周期点。但是，如果 $p \nmid a$，那么根据命题 2.1.4(2)，A_p 是 $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$的一个置换，因此 y_p 在A_p下是周期点。因此，实际上只有素数 $p \mid$ 最大公约数(a, m)会对 y施加限制。对于这些素数 p，根据命题2.1.4(1)，我们知道 A_p 恰好有一个周期点。由此可知，如果 $y_0 \in \mathbb{Z}/m\mathbb{Z}$ 是 A的一个周期点，那么 A 的周期点y 由同余式 $y \equiv y_0 \pmod{\gcd(a^L, m)}$刻画。因此，只剩下证明 $\sum_{t=0}^{L-1} a^t \cdot b$ 是 A的一个周期点。

现在，使用相同的论证并令 $b = 0$，使其适用于 $\mu_a: z \mapsto az$，我们看到，μ_a 的周期点 y 由同余式 $y \equiv 0 \pmod{\gcd(a^L, m)}$刻画。特别地，对于每个 $z \in \mathbb{Z}/m\mathbb{Z}$，$(\mu_a)^L(z) = a^L z$ 在 μ_a 下是周期的。由于 μ_a 和A 的函数图中周期顶点上的树（根据定理 2.1.5）同构，因而具有相同的高度，所以对于每个 $z \in \mathbb{Z}/m\mathbb{Z}$，$A^L(z)$ 在 A 下是周期的。特别地，$A^L(0) = \sum_{t=0}^{L-1} a^t \cdot b$ 在 A下是周期的，正如我们需要证明的那样。 □

这就总结了我们关于循环群的结果。为了结束本小节，我们再做一些工作，将定理 2.1.13 推广到任意有限群。首先，我们需要澄清一般情况下“仿射映射”的含义。下文中，对于固定的群 G，我们记 $\rho_{\mathrm{r}}: G \to \mathrm{Sym}(G)$，$x \mapsto (y \mapsto yx)$ 为 G 在自身上的所谓右正则表示（对于每个 $x \in G$，函数值 $\rho_{\mathrm{r}}(x) \in \mathrm{Sym}(G)$ 是在 G 上右乘 x）。类似地，ρ_{l} 表示 G 在自身上的左正则表示，其函数值是在 G 上左乘 G 的固定元素。按照群论中的惯例，我们将函数 $g: X \to Y$ 与函数 $g': Y \to Z$ 的复合写作乘积 $gg': X \to Z$（另一种等价的记号是 $g' \circ g$）。使用这种记号表示复合时，函数作用于自变量通常用指数表示：写作 x^g，而不是 $g(x)$，因此 $x^{gg'} = (x^g)^{g'}$。

定义 2.1.15. 设 G 是一个群。G 的仿射映射是一个形如 $\varphi\rho_{\mathrm{r}}(b): x \mapsto x^{\varphi} b$ 的函数 $G \to G$，其中 $b \in G$ 是某个固定元素，φ 是 G 的群自同态。

注 2.1.16. 关于仿射映射的概念，我们注意到以下几点。

(1) 由于 $\varphi\rho_{\mathrm{r}}(1_G) = \varphi$，仿射映射是群自同态的推广。(2) 给定群的仿射映射 G 构成其上的函数幺半群，因为该

y

由公式构成

$$\varphi\rho_{\mathrm{r}}(b) \cdot \varphi'\rho_{\mathrm{r}}(b) = \varphi\varphi'\rho_{\mathrm{r}}(b^{\varphi'} b').$$

(3) 或者，也可以将仿射映射定义为先应用群自同态 φ，再左乘一个常数，即定义为复合$\varphi\rho_{\mathrm{l}}(b) = \rho_{\mathrm{l}}(b) \circ \varphi$。这会得到相同的函数类，因为

$$(\rho_{\mathrm{l}}(b) \circ \varphi)(x) = b\varphi(x) = bx^{\varphi} = bx^{\varphi}b^{-1}b = x^{\varphi\,\mathrm{conj}(b^{-1})}b = x^{\varphi\,\mathrm{conj}(b^{-1})\cdot\rho_{\mathrm{r}}(b)},$$

其中，conj(g) 是内自同构（共轭） $x \mapsto g^{-1}xg$ 的 G。

我们简要回顾 [11] 中的一些其他概念和结果。

定义 2.1.17.设 G 是一个有限群，且 φ 是 G 的端同态。 φ 的超核记作nil(φ)，由满足以下条件的那些 $x \in G$ 构成：对于某个 $n = n(x) \in \mathbb{N}_0$， $\varphi^n(x) = 1_G$ 。

注2.1.18。记号nil(φ)源于以下事实：它是 G 的最大 φ-不变子群，并且相应的 φ 限制在其上是幂零端同态（即，如果自身复合足够多次便会变为平凡的端同态）。

在以下定理中，我们使用记号 $G = H \ltimes N$ 来表示群 G 是 H 与 N的（内部）半直积，这意味着 H 是 G的子群， N是 G的正规子群，并且有 $H \cap N = \{1_G\}$ 和 $HN = \{hn : h \in H, N \in N\} = G$。在这种情况下， G 中的每个元素都可以唯一地写成乘积 hn ，其中 $h \in H$ 和 $n \in N$ ，并且可以通过公式 $(hn) \cdot (h'n') = hh'n^{h'}n'$ 来计算 G 中的元素，其中 $n^{h'} = n^{\mathrm{conj}(h')} = (h')^{-1}nh'$。

定理2.1.19。设 G 是一个有限群，且 φ 是 G 的一个端同态。那么 $G = \mathrm{per}(\varphi) \ltimes \mathrm{nil}(\varphi)$ 。

证明。参见 [11, 定理1(1–3)的证明]。我们注意到，在观察到per(φ) 是一个子群且 nil(φ) 是 G 的正规子群之后，其余陈述便可由Caranti证明的群论版 Fitting 引理轻易推出，参见 [14, 定理 4。2] □

定理2.1.19对函数图Γ_{φ}有一个有趣的推论，该推论最初表述为 [11, 定理 1(4)] ，使用命题 2.1.12 以及 $X_1 = \mathrm{per}(\varphi)$ 和 $X_2 = \mathrm{nil}(\varphi)$ 很容易证明。

推论2.1.20。设 G 是一个有限群，且 φ 是 G 的一个端同态。则

$$\Gamma_{\varphi} = \Gamma_{\varphi_{|\mathrm{per}(\varphi)}} \otimes \Gamma_{\varphi_{|\mathrm{nil}(\varphi)}},$$

并且因此，对于每个 $x \in$每(φ)，有树$\Gamma_{\varphi}(x) \cong$树$^{\Gamma_{\varphi_{|\mathrm{nil}(\varphi)}}}(1_G)$，这是一个有根树，可以通过在 1_G处删除后者的唯一循环，从 $\Gamma_{\varphi_{|\mathrm{nil}(\varphi)}}$ 中得到。

以下结果将定理 2.1.13 推广到了任意有限群。

定理 2.1.21. 设 G 为有限群， $b \in G$，且 φ 是 G 的一个端同态。那么， $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 具有刚性繁衍。此外， $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 中任意周期顶点 x 的繁衍数序列 $(\mathrm{proc}_k(x))_{k\geq 1}$ ，与 Γ^*_{φ} 中某个周期顶点的繁衍数序列相同。

证明。我们对 $k \geq 1$ 使用归纳法，证明如果 x、$y \in G$ 在 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 中各自至少有 k 个后继代，那么 $\mathrm{proc}_k(x) = \mathrm{proc}_k(y)$ 在 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 中。对于归纳基础情形 $k = 1$，我们注意到，对于 $z \in \{x, y\}$，有

$$\begin{aligned}\mathrm{proc}_1(z) &= \#\text{ children of } z \text{ in } \Gamma^*_{\varphi\rho_{\mathrm{r}}(b)} \\ &= |\{w \in G : w^{\varphi\rho_{\mathrm{r}}(b)} = z\}| = |\{w \in G : w^{\varphi} = zb^{-1}\}|.\end{aligned}$$

由于 $\{w \in G : w^{\varphi} = zb^{-1}\}$ 要么为空，要么是$\ker\varphi$的陪集，因此只要$\mathrm{proc}_1(x)$，$\mathrm{proc}_1(y)$ 就是 $|\ker\varphi|$，其中 $\mathrm{proc}_1(x)$，$\mathrm{proc}_1(y) > 0$，如所要求。

现在我们假设 $k \geq 2$ 并且该陈述对 $k-1$以内的情形成立。两个顶点 x 和 y 在 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 中，对于每个 $n \in \{1, 2, \ldots, k-1\}$，都至少有 n 个后继代，因此根据归纳假设，对于 $n = 1$、2、.....、$k-1$，有 $\mathrm{proc}_n(x) = \mathrm{proc}_n(y)$。现在，对于每个 $z \in G$，从 z 出发的、长度为 k 的有向路径在 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 中的端点数量为

$$\begin{aligned}|\{w \in G : w^{((\varphi\rho_{\mathrm{r}}(b))^k)} = z\}| &= |\{w \in G : \varphi^k(w) \cdot \varphi^{k-1}(b)\varphi^{k-2}(b) \cdots \varphi(b)b = z\}| \\ &= |\{w \in G : \varphi^k(w) = zb^{-1}\varphi(b)^{-1} \cdots \varphi^{k-1}(b)^{-1}\}|,\end{aligned}$$

且集合 $\{w \in G : \varphi^k(w) = zb^{-1}\varphi(b)^{-1} \cdots \varphi^{k-1}(b)^{-1}\}$ 要么为空，要么是.....的一个陪集

$$\ker^{(k)}(\varphi) := \{w \in G : \varphi^k(w) = 1_G\}.$$

因此，x 和 y 从它们出发的长度为 k 的有向路径的端点数量相同，即为 $|\ker^{(k)}(\varphi)|$。利用归纳假设的一般形式（其基本表述是，$\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$具有“ $k-1$代的”刚性繁衍），对 $n = 1$、2、.....、$k-1$ 进行简单归纳可知，对于$\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$中至少具有 n 个后继代的每个顶点$w \in G$，从 w 出发的$\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$中长度为 n 的有向路径的端点数量为 $\prod_{j=1}^{n} \mathrm{proc}_j(w)$。特别地，如果 w 是 $z \in \{x, y\}$ 的一个具有至少 $k-1$ 个后继代的子节点，那么从 w 出发的长度为 $k-1$ 的路径的端点数量为 $\prod_{j=1}^{k-1} \mathrm{proc}_j(w)$，根据归纳假设，这等于 $\prod_{j=1}^{k-1} \mathrm{proc}_j(z)$。由于 z 的这类子节点 w 的数量为$\mathrm{proc}_k(z)$，我们得出

$$\begin{aligned}\prod_{j=1}^{k} \mathrm{proc}_j(z) &= \mathrm{proc}_k(z) \cdot \prod_{j=1}^{k-1} \mathrm{proc}_j(z) \\ &= \#\text{ endpoints of length } k \text{ paths starting at } z = |\ker^{(k)}(\varphi)|.\end{aligned}$$

由于 $z \in \{x, y\}$ 是任意的，因此可得

$$\prod_{j=1}^{k} \mathrm{proc}_j(x) = \prod_{j=1}^{k} \mathrm{proc}_j(y),$$

且由于 $\mathrm{proc}_j(x) = \mathrm{proc}_j(y) > 0$ 对于 $j = 1, 2, \ldots, k-1$，这使我们能够得出结论：$\mathrm{proc}_k(x) = \mathrm{proc}_k(y)$，如所要求。

关于 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 和 Γ^*_{φ} 的繁殖数相同这一断言，上述论证表明，对于任意正整数 k 以及$\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$的任意周期顶点 x，我们有

$$\prod_{j=1}^{k} \mathrm{proc}_j(x) = |\ker^{(k)}(\varphi)|.$$

由此可得

$$\mathrm{proc}_k(x) = |\ker^{(k)}(\varphi) : \ker^{(k-1)}(\varphi)|$$

对每个 $k \in \mathbb{N}^+$（注意，$\ker^{(0)}(\varphi) = \{1_G\}$）。因此，$\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)}$ 的周期顶点的繁衍数序列仅取决于 φ，而非 b；并且对于 $b := 1_G$，有 $\Gamma^*_{\varphi\rho_{\mathrm{r}}(b)} = \Gamma^*_{\varphi}$。

□

2.2 主引理

根据引言，我们回顾 m-CC（即“m-同余条件”）这一概念。我们将其定义为形如 $\nu(x \equiv \mathfrak{b} \pmod{\mathfrak{a}})$的条件，其中 $\mathfrak{a} \mid m$且 $\nu \in \{\emptyset, \neg\}$。本小节考虑由一个公共变量中的 m-CC 构成的系统。若此类系统有整数解，则称其为相容；若两个此类系统在 $\mathbb{Z}$（或等价地，在 $\mathbb{Z}/m\mathbb{Z}$）中具有相同的解集，则称它们等价。相容的 m-CC 系统在 $\mathbb{Z}/m\mathbb{Z}$ 中的解集，是相关算术划分 $\mathbb{Z}/m\mathbb{Z}$ 的一个块（见定义 1.5）。我们注意到，关于 m-同余式系统的以下基本结果，是中国剩余定理的一个广为人知的推广。

命题 2.2.1. 设 m 是一个正整数。我们考虑一个由 m 个同余式组成的系统，其形式为

$$\begin{aligned} x &\equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1} \\ x &\equiv \mathfrak{b}_2 \pmod{\mathfrak{a}_2} \\ &\vdots \\ x &\equiv \mathfrak{b}_K \pmod{\mathfrak{a}_K} \end{aligned} \tag{5}$$

下列陈述等价。

(1) 系统 (5) 是相容的。(2) 对所有 $1 \le j < k \le K$，都有 $\gcd(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$。(3) 系统 (5) 中任意一对 m 个同余式都构成一个相容系统。(4) 系统 (5) 等价于一个单独的 m 个同余式，其形式为

$$x \equiv \mathfrak{b} \pmod{\operatorname{lcm}(\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_K)}.$$

特别地，如果方程组 (5) 有解，那么它模 m 的解的个数等于 $m/\operatorname{lcm}(\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_K)$，因此与 $\mathfrak{b}_j$ 无关。

证明。关于等价关系“(1)⇔(2)”和蕴含关系“(1)⇒(4)”，例如参见 [39, 第 3.3.4 定理（第 78], 页）。此外，蕴含关系“(4)⇒(1)”显然成立。至于等价关系“(1)⇔(3)”，注意到将已证明的等价关系“(1)⇔(2)”应用于方程组

$$\begin{aligned} x &\equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} \\ x &\equiv \mathfrak{b}_k \pmod{\mathfrak{a}_k}, \end{aligned} \tag{6}$$

当且仅当单个整除关系最大公约数$(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$成立时，该系统是一致的。但陈述（3）只要求系统（6）对所有 $1 \le j < k \le K$ 都是一致的，因此这等价于陈述（2），进而等价于陈述（1）。 □

命题 2.2.1 是证明以下引理的基础，该引理对于采用递归方法理解第 3.3 小节中的有根树至关重要。

引理 2.2.2。（主引理）设 m 是一个正整数，且 $\mathcal{P}$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的一个算术划分。此外，设 a、$b \in \mathbb{Z}/m\mathbb{Z}$，并考虑 $\mathbb{Z}/m\mathbb{Z}$ 上的仿射映射 $A:\ x \mapsto ax + b$。存在 $\mathbb{Z}/m\mathbb{Z}$ 的一个算术划分 $\mathcal{P}'$，使得 $\mathrm{AC}(\mathcal{P}') \le \mathrm{AC}(\mathcal{P}) + 1$，并且若 x、$y \in \mathbb{Z}/m\mathbb{Z}$ 属于 $\mathcal{P}'$ 的同一个块，且若 B 是 $\mathcal{P}$ 的一个块，则 $|A^{-1}(\{x\}) \cap B| = |A^{-1}(\{y\}) \cap B|$。

更具体地说，如果 $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j (\text{模 } \mathfrak{a}_j):\ j = 1, 2, \ldots, K)$，那么

$$\mathfrak{P}'(\mathcal{P}, A) := \mathfrak{P}\left(\begin{array}{l} x \equiv a\mathfrak{b}_1 + b \pmod{\gcd(a\mathfrak{a}_1, m)} \\ x \equiv a\mathfrak{b}_2 + b \pmod{\gcd(a\mathfrak{a}_2, m)} \\ \quad \vdots \\ x \equiv a\mathfrak{b}_K + b \pmod{\gcd(a\mathfrak{a}_K, m)} \\ x \equiv b \pmod{\gcd(a, m)} \end{array}\right)$$

是 $\mathcal{P}'$ 的一个有效选择，并且可以如下得到 $|A^{-1}(\{x\}) \cap B|$ 关于 $\mathcal{P}$-块B 和包含 x 的唯一 $\mathfrak{P}'(\mathcal{P}, A)$- 块 B' 的公式：将 $B = \mathcal{B}(\mathcal{P}, \vec{\nu})$ 和 $B' = \mathcal{B}(\mathfrak{P}'(\mathcal{P}, A)\ \vec{\nu'})$ 写为 $\vec{\nu} = (\nu_1, \ldots, \nu_K) \in \{\emptyset, \neg\}^K$ 和 $\vec{\nu'} = (\nu'_1, \ldots, \nu'_{K+1}) \in \{\emptyset, \neg\}^{K+1}$。我们引入以下记号。

- $J_-(\vec{\nu}) := \{j \in \{1, 2, \ldots, K\}:\ \nu_j = \neg\}$ 以及（类似地） $J_-(\vec{\nu'}) := \{j' \in \{1, 2, \ldots, K+1\}:\ \nu'_{j'} = \neg\}$;
- $J_+(\vec{\nu}) := \{1, 2, \ldots, K\} \setminus J_-(\vec{\nu})$;

对于 $J \subseteq J_-(\vec{\nu})$，我们用 $E(\vec{\nu}, J)$表示条件“对于所有 j_1， $j_2 \in J_+(\vec{\nu}) \cup J$：最大公约数$(\mathfrak{a}_{j_1}, \mathfrak{a}_{j_2}) \mid \mathfrak{b}_{j_1} - \mathfrak{b}_{j_2}$。”

然后，对于每个 $x \in \mathcal{B}(\mathfrak{P}'(\mathcal{P}, A)\ \vec{\nu'})$，交集大小 $|A^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}, \vec{\nu})|$ 等于

$$\sum_{J \subseteq J_-(\vec{\nu})} (-1)^{|J|} \kappa(\vec{\nu}, \vec{\nu'}, J) =: \sigma_{\mathcal{P}, A}(\vec{\nu}, \vec{\nu'})$$

其中

$$\kappa_{\mathcal{P}, A}(\vec{\nu}, \vec{\nu'}, J) :=$$

$$\delta_{\nu'_{K+1}=\emptyset}\cdot\delta_{E(\vec{\nu},J)}\cdot\delta_{(J_+(\vec{\nu})\cup J)\cap J_-(\vec{\nu'})=\emptyset}\cdot\frac{m}{\operatorname{lcm}(\frac{m}{\gcd(a,m)},\mathfrak{a}_j : j\in J_+(\vec{\nu})\cup J)},$$

这三个δ均为克罗内克δ。

我们将形如 $\sigma_{\mathcal{P},A}(\vec{\nu},\vec{\nu'})$ 的数称为 $\mathcal{P}$ （在 A 下）的分布数。

引理 2.2.2 的证明。为了验证 $\mathcal{P}' := \mathfrak{P}'(\mathcal{P},A)$ 具有所需性质，我们令

- $M_j := \{x\in\mathbb{Z}/m\mathbb{Z}\colon\ x\equiv\mathfrak{b}_j(\text{模 }\mathfrak{a}_j)\}$ 对于 $j=1,2,\ldots,K$;
- $M_+(\vec{\nu}) := \bigcap_{j\in J_+(\vec{\nu})} M_j$;

并注意到 $B = M_+(\vec{\nu})\cap\bigcap_{j\in J_-(\vec{\nu})} M_j^c$（上标 c 表示在 $\mathbb{Z}/m\mathbb{Z}$ 中取集合补集）。因此，根据容斥原理，我们所求的交集大小 $|A^{-1}(\{x\})\cap B|$ 等于

$$\sum_{J\subseteq J_-(\vec{\nu})}(-1)^{|J|}|(M_+(\vec{\nu})\cap A^{-1}(\{x\}))\cap\bigcap_{j\in J}M_j|.$$

因此，我们的目标是论证该和的值等于 $\sigma_{\mathcal{P},\ A}(\vec{\nu},\vec{\nu'})$（特别地，它与选择 $x\in B' = \mathcal{B}(\mathcal{P}',\vec{\nu'})$ 无关）。我们通过论证事实上，交集大小

$$|(M_+(\vec{\nu})\cap A^{-1}(\{x\}))\cap\bigcap_{j\in J}M_j|$$

等于 $\kappa_{\mathcal{P},A}(\vec{\nu},\vec{\nu'},J)$，对于每个 $J\subseteq J_-(\vec{\nu})$。现在，记 $J_+(\vec{\nu})\cup J=\{j_1,j_2,\ldots,j_N\}$，交集 $(M_+(\vec{\nu})\cap A^{-1}(\{x\}))\cap\bigcap_{j\in J}M_j$ 是 $\mathbb{Z}/m\mathbb{Z}$ 中以下关于变量 y 的方程组的解集：

$$\begin{aligned}
y &\equiv \mathfrak{b}_{j_1} \pmod{\mathfrak{a}_{j_1}}\\
y &\equiv \mathfrak{b}_{j_2} \pmod{\mathfrak{a}_{j_2}}\\
&\vdots\\
y &\equiv \mathfrak{b}_{j_N} \pmod{\mathfrak{a}_{j_N}}\\
ay+b &\equiv x \pmod{m}.
\end{aligned}\tag{7}$$

现在，单个同余式 $ay+b\equiv x\ (\text{mod }\ m)$ 在 y 中有解，当且仅当 $\gcd(a,m)\mid x-b$，也就是说，当且仅当 $x\equiv b\,(\text{mod }\gcd(a,m))$。这是关于 $\mathcal{P}'$ 的生成同余式之一，因此对所有 $x\in B'$，其真值都相同。如果该同余式对所有 $x\in B'$ 都不成立（等价地，如果 $\nu'_{K+1}=\neg$），那么方程组 (7) 恒为假，因此对所有 $J\subseteq J_-(\vec{\nu})$，$|(M_+(\vec{\nu})\cap A^{-1}(\{x\}))\cap\bigcap_{j\in J}M_j|=0$，且与 x 无关。这解释了 $\kappa_{\mathcal{P},\ A}(\vec{\nu},\nu',J)$ 定义中的第一个克罗内克δ。

因此，今后我们可以假设 $\nu'_{K+1}=\emptyset$，即对于所有 $x\in B'$，$x\equiv b$（模 最大公约数(a,m)）。然后，同余式 $ay+b\equiv x$（模 m）可以等价地改写为

$$y\equiv \operatorname{inv}_{m/\gcd(a,m)}\left(\frac{a}{\gcd(a,m)}\right)\cdot\frac{x-b}{\gcd(a,m)}\ \left(\text{mod }\frac{m}{\gcd(a,m)}\right),\tag{8}$$

其中，$\mathrm{inv}_n(x)$ 表示单位元 x 模 n 的乘法逆元。如果我们将系统(7) 中的同余式 $ay + b \equiv x \pmod{m}$替换为等价的同余式(8)，那么所得系统将完全由 m-同余式组成。根据命题 2.2.1，该系统模 m 的解的数量只有两种可能：要么系统不相容，因此有0个解；要么有 m/L 个解，其中 L 是所出现模数的最小公倍数。这两个解数量表达式都不依赖于 x，因此我们旨在证明其不依赖于选择 $x \in B'$ 两种情况中的哪一种。

现在，命题 2.2.1 还表明，系统 (7) 相容，当且仅当其中任意一对条件相容。因此，只需证明，对于系统 (7) 中任意一对条件，由这两个条件构成的系统是否相容都不取决于 $x \in B'$ 的取值。若这两个条件都不同于同余式 $ay+b \equiv x \pmod{m}$，则它们分别具有 $y \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j}$ 和 $y \equiv \mathfrak{b}_k \pmod{\mathfrak{a}_k}$ 的形式，其中 $j, k \in J_+(\vec{\nu}) \cup J$ 取适当值；根据命题 2.2.1，这两个条件构成的系统相容，当且仅当 $\gcd(\mathfrak{a}_j, \mathfrak{a}_k) \mid \mathfrak{b}_j - \mathfrak{b}_k$。等价地，从 (7) 中删去唯一的同余式 $ay + b \equiv x \pmod{m}$ 后得到的系统相容，当且仅当条件 $E(\vec{\nu}, J)$ 成立；这解释了 $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$ 定义中的第二个克罗内克δ。

接下来考虑（7）中具有如下形式的二条件子系统

$$\begin{aligned} ay + b &\equiv x \pmod{m} \\ y &\equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} \end{aligned} \tag{9}$$

对某个 $j \in J_+(\vec{\nu}) \cup J$。我们声称，系统（9）有解当且仅当$x \equiv a\mathfrak{b}_j + b$（模 $\gcd(a\mathfrak{a}_j, m)$）。事实上，若系统（9）有解，则存在一个$k \in \mathbb{Z}$，使得某个 $y = \mathfrak{b}_j + k\mathfrak{a}_j$ 满足系统的第一个条件。也就是说，有

$$x \equiv ay + b = a\mathfrak{b}_j + ka\mathfrak{a}_j + b \pmod{m}.$$

特别地，

$$x \equiv a\mathfrak{b}_j + ka\mathfrak{a}_j + b \equiv a\mathfrak{b}_j + b \pmod{\gcd(a\mathfrak{a}_j, m)},$$

这正是所需的。另一方面，假设 $x \equiv a\mathfrak{b}_j + b$（模 $\gcd(a\mathfrak{a}_j, m)$）。那么，对某个 $k' \in \mathbb{Z}$, 可写成 $x = a\mathfrak{b}_j + b + k'\gcd(a\mathfrak{a}_j, m)$。我们需要验证，存在一个整数 k，使得对于 $y = \mathfrak{b}_j + k\mathfrak{a}_j$，有

$$ay + b = a\mathfrak{b}_j + ka\mathfrak{a}_j + b \equiv x = a\mathfrak{b}_j + b + k'\gcd(a\mathfrak{a}_j, m) \pmod{m},$$

这等价于 $ka\mathfrak{a}_j \equiv k'\gcd(a\mathfrak{a}_j, m)$（模 m）。确实，该同余式关于 k可解，因为 $\gcd(a\mathfrak{a}_j, m) \mid k'\gcd(a\mathfrak{a}_j, m)$。

现在，由于 $x \equiv a\mathfrak{b}_j + b$(模 $a\mathfrak{a}_j$、m的最大公约数) 是 $\mathcal{P}'$ 的第 j 个生成同余式，因此可知，若 $\nu'_j = \neg$（等价地，若 $j \in J_-(\vec{\nu'})$），那么只要 $j \in J_+(\vec{\nu}) \cup K$，交集 $(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j \in J} M_j$ 就为空；这也解释了 $\kappa_{\mathcal{P},\ A}(\vec{\nu}, \vec{\nu'}, J)$ 定义中的第三个克罗内克δ。

如果 $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$ 的定义中三个克罗内克δ全都为 1，那么我们的论证表明，系统 (7) 是相容的，因此等价于一个单独的

根据命题 2.2.1 应用m-同余式（我们回顾一下，系统 (7) 中的最后一个同余式可以替换为等价的 m-同余式(8)），其模数为系统 (7)中条件的 m-同余式形式所涉及模数的最小公倍数 L。由此可知，系统 (7) 的解集大小为

$$\frac{m}{L} = \frac{m}{\operatorname{lcm}\left(\frac{m}{\gcd(a,m)}, \mathfrak{a}_j : j \in J_+(\vec{\nu}) \cup J\right)}.$$

因此，我们的技术参数 $\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$确实始终与交集大小 $|(M_+(\vec{\nu}) \cap A^{-1}(\{x\})) \cap \bigcap_{j\in J} M_j|$一致，从而证明完毕。 □

2.3 有限循环群的仿射映射 CRL 列表

我们将 $\mathbb{Z}/m\mathbb{Z}$（其中 $m \in \mathbb{N}^+$）视为一个环，其底集为 $\{0, 1, \ldots, m-1\}$，环运算为模加法和模乘法。特别地，若 $m_1 \le m_2$ 是正整数，则有集合包含关系 $\mathbb{Z}/m_1\mathbb{Z} \subseteq \mathbb{Z}/m_2\mathbb{Z}$。我们也可以通过模 m 约化，将范围 $\{0, 1, \ldots, m-1\}$ 之外的整数视为 $\mathbb{Z}/m\mathbb{Z}$ 的元素（将 $x \in \mathbb{Z}$ 与模 m 的 x 视为相同）。提醒读者注意记号 $\nu_p^{(v)}(x) := \min\{v, \nu_p(x)\}$，其中 p 为素数，$v \in \mathbb{N}_0$ 且 $x \in \mathbb{Z}$；该记号最初于定理 2.1.5 之后引入。

正如引言中所提到的，为 $\mathbb{F}_q$ 的广义分圆映射构造 CRL 列表（按照定义 1.2 的含义）可以归结为有限循环群仿射映射的相应问题，而我们将在本小节中解决这一问题。我们还在引言中指出，对于有限的 X，确定函数 $g\colon X \to X$ 的 CRL 列表通常比确定 $g_{|\mathrm{per}(g)}$ 的循环类型更困难；对于 g 是有限循环群上的仿射置换这一情形，我们希望概述后一个问题的发展历史。

Ahmad [4] 确定了有限循环群自同构的循环结构。有限循环群 $\mathbb{Z}/m\mathbb{Z}$ 的仿射置换群的循环指标（这是一个多项式，用于编码每种给定循环类型的仿射置换数量）由 Wei 和 Xu [77] 描述。在他们的论文中，他们给出了 m 为素数幂时的公式，但未提供证明，而是引用了 Wei、Gao 和 Yang 撰写的两页研究简报 [76]。遗憾的是，Bors 和 Wang 在研究 [13], 时，未能通过在线搜索找到 [76]，因此他们独立推导出了这些公式，并将其作为 [13, 定理 4.8], ，其依据是对给定仿射置换 $A\colon x \mapsto ax + b$ 的 $\mathbb{Z}/p^v\mathbb{Z}$ 的循环类型进行精确描述，该描述以 a 和 b 表示，并陈述为 [13, 命题 4.7]。截至目前，据作者所知，后一个结果是唯一一份以表格形式明确列出这些循环类型的可获取参考文献。在撰写当前论文期间，作者意识到，[13, 命题 4.7] 也可以很容易地由 Deng 的结果 [19, 引理 4 和 7] 以及 Ahmad 的结果 [4, 定理 1] 推导出来。

现在我们转向 CRL 列表的确定。设 m 为正整数，并令 a、$b \in \mathbb{Z}$。我们考虑 $\mathbb{Z}/m\mathbb{Z}$ 的仿射映射 $A\colon x \mapsto ax + b$。由

命题 2.1.4，我们知道以下内容。

- A 模 m' 的约化：$= \prod_{p|m,\ p\nmid a} p^{\nu_p}\ (m)$ 是$\mathbb{Z}/m'\mathbb{Z}$上的仿射置换。

- A 模 m'' 的约化：$= \prod_{p|\gcd(a,\ m)} p^{\nu_p}\ (m)$ 在 $\mathbb{Z}/m''\mathbb{Z}$中恰有一个周期点 $\mathfrak{f}''$，借助引理 2.1.14，我们可以明确得知该点。

现在，投影在

$$\mathbb{Z}/m\mathbb{Z} \cong \mathbb{Z}/m'\mathbb{Z} \times \mathbb{Z}/m''\mathbb{Z} \to \mathbb{Z}/m'\mathbb{Z}$$

到 $\mathbb{Z}/m'\mathbb{Z} \times \{\mathfrak{f}''\}$ 的限制是双射；我们将其逆函数记为 $\Lambda\ \ \mathbb{Z}/m'\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$。那么，如果 $\mathcal{L}'$ 是 A 模 m'的约化的 CRL 列表，则集合 $\{(\Lambda(r)\ l) : (r,l)\ \in\mathcal{L}'\}$ 是 A的 CRL 列表。这将问题归结为特殊情形：A 是有限循环群上的仿射置换，以下我们均假设如此。

为了理解有限循环群的仿射置换的 CRL 列表，它
is 分几个步骤进行会很有帮助：

(1) 首先，我们为每个有限主循环群（即 $\mathbb{Z}/p^v\mathbb{Z}$）上的每个群自同构确定一个 CRL 列表。(2) 接下来，我们将其推广到有限主循环群上的任意仿射置换。(3) 最后，我们运用中国剩余定理和一些额外的技巧，根据 A 对不同素数幂 $p^{\nu_p(m)}$ 的约化的 CRL 列表，构造每个有限循环群上的任意仿射置换 A 的 CRL 列表。

在正式处理步骤（1）之前，我们先证明下面这个有用的引理。

引理 2.3.1。设 X 为一个有限集，$\psi \in$对称群(X)，$\mathcal{L}$ 为 ψ的 CRL 列表，且 $n \in \mathbb{Z}$。则以下是 ψ^n的 CRL 列表：

$$\left\{ \left(\psi^j(r), \frac{l}{\gcd(n,l)} \right) : (r,l) \in \mathcal{L}, j = 0, 1, \ldots, \gcd(n,l) - 1 \right\}.$$

特别地，如果最大公约数$(n, \mathrm{ord}(\psi))\ = 1$，那么 $\mathcal{L}$ 也是 ψ^n的 CRL 列表。

证明。ψ^n 的每个循环都包含在 ψ的一个循环中。对于 ψ的任意给定的长度为 l的循环 ζ，ζ 分解成的ψ^n-循环对应于由 $n + l\mathbb{Z}$生成的 $\mathbb{Z}/l\mathbb{Z}$ 子群的陪集。由于 n 模 l 的加法阶为$l/\gcd(n,l)$，因此 ζ分解为$\gcd(n,l)$个长度各为$l/\gcd(n,l)$的 ψ^n循环。若 r 是 ζ在 $\mathcal{L}$中的代表元，则对于包含在 ζ中的每个 ψ^n-循环 ζ'，ζ' 上的元素恰为如下形式：对于某个 $k = k(\zeta')\ \in \mathbb{Z}$，取 $t \in k + \gcd(n,l)\mathbb{Z}$时的 $\psi^t(r)$。由此可知，对于 $j = 0$，取1，.....，$\gcd(n,l)\ - 1$时，(支撑集中的) ζ的$\gcd(n,l)$个元素 $\psi^j(r)$分别位于两两不同的 ψ^n循环上，因此构成包含在 ζ中的 ψ^n循环的一个代表元系。这就证明了引理的主要结论。“特别地”部分则由此得出，因为等式$\gcd(n, \mathrm{ord}(\psi))\ = 1$等价于“对于 ψ的每个循环长度 l，都有$\gcd(n,l)\ = 1$”。 □

现在我们已经准备好为每个有限主循环群的每个自同构指定一个 CRL列表。在以下引理的证明及后文中，我们使用记号 $H \leq G$ 表示“H是 G的子群”，并用 $\langle g_1$、g_2、……、$g_n\rangle$ 表示由元素 g_1、g_2、……、$g_n \in G$生成的群 G 的子群。

引理 2.3.2。设 p 是一个素数，$v \in \mathbb{N}^+$，并且 $a \in \mathbb{Z}$ 满足 $p \nmid a$。若 p 为奇数，令 $\mathfrak{r}$ 为模 p^v 的一个固定原根，并令 ϕ 表示欧拉 φ 函数。表1给出了 $\mathbb{Z}/p^v\mathbb{Z}$ 的自同构 μ_a：$x \mapsto ax$ 的 CRL 列表 $\mathcal{L}(p^v, a)$。

关于 p 和 a的情况	$\mathcal{L}(p^v, a)$ 的元素
$p > 2$	$\left(\mathfrak{r}^j p^t, \frac{\phi(p^{v-t})\text{最大公约数}\left(\phi(p^v)\right.}{\left.\operatorname{ord}(a)\ \phi(p^{v-t})\right)}\right)$ for $t = 0$、1、……、v和 $j = 0$、1、……、最大公约数 $\left(\phi(p^v)\operatorname{ord}(a)\ \phi(p^{v-t})\right) - 1$。
$p = 2,\ a \equiv 1$ （模 4）	$(0, 1)$, $(2^{v-1}, 1)$; $\left(5^j 2^t, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\operatorname{ord}(a)}, 2^{v-t-2}\right)}\right)$, $\left(-5^j 2^t, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{\operatorname{ord}(a)}, 2^{v-t-2}\right)}\right)$ 为 $t = 0, 1, \ldots, v-2$和 $j = 0, 1, \ldots,$ 最大公约数 $\left(\frac{2^{v-2}}{a}\operatorname{ord}()\ 2^{v-t-2}\right) - 1$.
$p = 2,\ a \equiv 3$ (模 4)	$(0, 1)$, $(2^{v-1}, 1)$; $(j \cdot \operatorname{ord}(-a), 2)$ 对于 $j = 1, 2, \ldots, \frac{2^{v-1}}{\operatorname{ord}(-a)} - 1$; $\left(5^j 2^t, \frac{\operatorname{ord}(-a)}{2^t}\right)$, $\left(-5^j 2^t, \frac{\operatorname{ord}(-a)}{2^t}\right)$ 用于 $t = 0, 1, \ldots, \log_2(\operatorname{ord}(-a)) - 1$ 并且 $j = 0, 1, \ldots, \frac{2^{v-2}}{\operatorname{ord}(-a)} - 1$.

表1：有限主循环群的自同构的 CRL 列表。

引理2.3.2 的证明。首先，我们假设 $p > 2$。如果 a 是模p^v的一个原根，则循环群 $\langle a\rangle = (\mathbb{Z}/p^v\mathbb{Z})^* \cong \operatorname{Aut}(\mathbb{Z}/p^v\mathbb{Z})$在 $\mathbb{Z}/p^v\mathbb{Z}$ 的每个子集上传递作用，其中每个子集由所有具有给定加法阶的元素组成。事实上，一方面，$\mathbb{Z}/p^v\mathbb{Z}$ 的自同构必须保持元素的加法阶；反之，如果 x、$y \in \mathbb{Z}/p^v\mathbb{Z}$ 具有相同的阶，那么它们互为倍数。因此，对于某个 $z \in (\mathbb{Z}/p^v\mathbb{Z})^*$，有 $y = z \cdot x$。由于 z是 a的幂，传递性断言得证。我们得出，如果 a 是模 p^v 的一个原根，则 $\mathcal{L}(p^v, a)$可以取为 $\{(p^t, \phi(p^{v-t}))：t = 0$、1、…、$v\}$，这与表1相符。

对于一般的 a，我们注意到 a 和 $\mathfrak{r}^{\phi(p^v)/\operatorname{ord}(a)}$ 互为幂，因此根据引理 2.3.1 的“特别是”部分，我们不妨假设 $a = \mathfrak{r}^{\phi(p^v)/\operatorname{ord}(a)}$。现在将引理 2.3.1 的主结论应用于 $n := \phi(p^v)/\operatorname{ord}(a)$ 和 $\mathcal{L} := \{(p^t, \phi(p^{v-t}))：t = 0, 1, \ldots, v\}$，即可得证。

现在我们假设 $p = 2$。首先，讨论 $a = 5$ 的情形。自同构 μ_5：$x \mapsto 5x$ 与 $\mathbb{Z}/2^v\mathbb{Z}$ 的任意自同构一样，固定唯一的元素 0 和 2^{v-1}，它们的加法阶分别为 1 和 2。它还固定阶为 4 的元素 2^{v-2} 和 $3 \cdot 2^{v-2} = -2^{v-2}$。此外，我们声称，对于每个 $t' \in \{3, 4, \ldots, v\}$，自同构 μ_5 在 $\mathbb{Z}/2^v\mathbb{Z}$ 中加法阶为 $2^{t'}$ 的元素上恰有两个循环，它们的长度均为 $2^{t'-2}$，分别由 $2^{v-t'}$ 和 $-2^{v-t'}$ 张成。事实上，这

对于 $t' = 3$ 和 $t' = v$，这一点显然成立；对于后者，我们利用以下事实：$5 = 1 + 2^2$ 模 2^v 的乘法阶为 2^{v-2}，$\langle 5\rangle \leq (\mathbb{Z}/2^v\mathbb{Z})^*$ 在生成元（即 $\mathbb{Z}/2^v\mathbb{Z}$ 中加法阶为 2^v 的元素）集合上半正则地作用（也就是说，在该作用下，该群中除单位元 1 以外的任何元素都没有不动点），且 $1 \not\equiv -1$ （模 4）。对于 t' 的其他每个取值，记 $\mathrm{aord}(x)$ 为 x 模 2^v 的加法阶，由该交换图可得

$$\begin{array}{ccc} \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^v\} & \xrightarrow{x \mapsto 5x} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^v\} \\ \Big\downarrow {\scriptstyle x \mapsto 2^{v-t'}x} & & \Big\downarrow {\scriptstyle x \mapsto 2^{v-t'}x} \\ \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^{t'}\} & \xrightarrow[x \mapsto 5x]{} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^{t'}\} \end{array}$$

即，$x \mapsto 5x$ 在阶为 $2^{t'}$ 的元素集合上至多有两个循环，即由 $2^{v-t'} \cdot 1 = 2^{v-t'}$ 和 $2^{v-t'} \cdot (-1) = -2^{v-t'}$张成的循环。同样地，该图表的交换性

$$\begin{array}{ccc} \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^{t'}\} & \xrightarrow{x \mapsto 5x} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 2^{t'}\} \\ \Big\downarrow {\scriptstyle x \mapsto 2^{t'-3}x} & & \Big\downarrow {\scriptstyle x \mapsto 2^{t'-3}x} \\ \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 8\} & \xrightarrow[x \mapsto 5x]{} & \{x \in \mathbb{Z}/2^v\mathbb{Z} : \mathrm{aord}(x) = 8\} \end{array}$$

这意味着 $2^{v-t'}$ 和 $-2^{v-t'}$ 位于 $x \mapsto 5x$ 的不同循环上。这表明可以按照表1中的指示选择 $\mathcal{L}(2^v, 5)$。

至于 a的其他取值，若 $a \equiv 1$ （模4），则 a 与 5的某个幂模 2^v 同余，而表1中为 $\mathcal{L}(2^v, a)$指定的选择，可以利用引理2.3.1 从 $\mathcal{L}(2^v,5)$ 的选择推导出来（与上文关于 $p > 2$ 的论证结尾类似）。

接下来要处理 $a \equiv 3 \pmod 4$ 的情形。此时 $-a \equiv 1 \pmod 4$。我们将自同构 μ_a of $\mathbb{Z}/2^v\mathbb{Z}$ 视为自同构 μ_{-a} 和μ_{-1}的复合。对于每个 $t' \in \{0, 1, \ldots, v\}$，我们希望了解 μ_a 在加法阶为 $2^{t'}$ 的元素集合 x 上的循环，并通过区分 t' 的几种情形来分析。

- 若 $t' \in \{0, 1\}$ （即 $x \in \{0, 2^{v-1}\}$），那么 x 是 $\mathbb{Z}/2^v\mathbb{Z}$ 中唯一具有该加法阶的元素，因此在 μ_a 下保持不变。
- 若 $t' \in \{2, 3, \ldots, v - \log_2(\mathrm{ord}(-a))\}$，则 μ_{-a} 固定每个阶为 $2^{t'}$ 的元素；使用表1中“$a \equiv 1 \pmod 4$”的公式，并注意到若 $x \in \{a^j 2^t, -a^j 2^t\}$，则 $2^{t'} = \mathrm{aord}(x) = 2^{v-t}$，即可看出这一点。由此可知，$\mu_a$ 在阶为 $2^{t'}$ 的元素集合上的限制与 μ_{-1} 相同。因此，

$$\left\{(j \cdot \mathrm{ord}(-a), 2) : j = 1, 2, \ldots, \frac{2^{v-1}}{\mathrm{ord}(-a)} - 1\right\},$$

这是 μ_{-1} 在 $\mathbb{Z}/2^v\mathbb{Z}$ 中阶属于 $\{2^2, 2^3, \ldots, 2^{v-\log_2(\mathrm{ord}(-a))}\}$ 的元素集合上的限制所对应的 CRL 列表，也同样是 μ_a 对应限制的 CRL 列表。

• 最后，如果 $t' > v - \log_2(\mathrm{ord}(-a))$，那么 μ_{-a} 在 $\mathbb{Z}/2^v\mathbb{Z}$ 中阶为 $2^{t'}$ 的元素集合上的所有循环长度均为偶数（事实上，其长度是非平凡的 2 的幂）。对于固定的 $j \in \{0, 1, \ldots, \gcd(2^{v-2}\mathrm{ord}(-a)2^{t'-2}) - 1\} = \{0, 1, \ldots, 2^{v-2}\mathrm{ord}(-a) - 1\}$，我们分别考察由 $5^j 2^{v-t'}$ 和 $-5^j 2^{v-t'}$ 生成的 μ_{-a} 的两个循环。根据表1中应用于 $-a$ 的“$a \equiv 1 \pmod 4$”情形，这两个循环互不相同，且长度均为

$$\frac{2^{t'-2}}{\gcd(\frac{2^{v-2}}{\mathrm{ord}(-a)}, 2^{t'-2})} = \frac{2^{t'-2}}{2^{v-2-\log_2(\mathrm{ord}(-a))}} = 2^{t'-v+\log_2(\mathrm{ord}(-a))}$$

并且在 μ_{-1}作用下互为像。事实上，由于 μ_{-a} 和 μ_{-1} 可交换，对于每个 $n \in \mathbb{Z}$，都有 $\mu_{-1}(\mu_{-a}^n(\pm 5^j 2^{v-t'})) = \mu_{-a}^n(\mu_{-1}(\pm 5^j 2^{v-t'}))$，由此可得到如下示意图，说明在 μ_{-1}作用下这些循环如何匹配。令 $k := 2^{t'-v+\log_2(\mathrm{ord}(-a))-1}$，则 k 是 $\pm 5^j 2^{v-t'}$ 在 μ_{-a}作用下共同周期长度的一半：

$$\begin{array}{ccccccccc}
5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & (-a)5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & \cdots & \xrightarrow{\mu_{-a}} & (-a)^{2k-1}5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & 5^j 2^{v-t'} \\
\downarrow{\scriptstyle \mu_{-1}} & & \downarrow{\scriptstyle \mu_{-1}} & & \downarrow{\scriptstyle \mu_{-1}} & & & & \downarrow{\scriptstyle \mu_{-1}} \\
-5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & -(-a)5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & \cdots & \xrightarrow{\mu_{-a}} & -(-a)^{2k-1}5^j 2^{v-t'} & \xrightarrow{\mu_{-a}} & -5^j 2^{v-t'}
\end{array}$$

因此，$\mu_a = \mu_{-1} \circ \mu_{-a}$ 在上述 μ_{-a}两个循环的支撑集之并上分解为两个循环：一个由 $5^j 2^{v-t'}$生成，包含上方循环中的“偶元素” $5^j 2^{v-t'}$、$(-a)^2 5^j 2^{v-t'}$、$(-a)^4 5^j 2^{v-t'}$、……，以及下方循环中的“奇元素”；另一个由 $-5^j 2^{v-t'}$ 生成，包含上方循环中的“奇元素”和下方循环中的“偶元素”。若令 t' 和 j 分别遍历其取值范围，则相应的循环对将加法阶大于$2^v/\mathrm{ord}(-a)$的所有元素划分为若干部分。结合上文对 t' 较小取值的观察，我们得到 μ_a的一个 CRL 列表；容易验证，它与表1中给出的列表一致（注意，在该表的记号中，$t' = v - t$）。

□

现在我们处理本小节计划中的步骤 (2)，即为每个有限主循环群的每个仿射置换 A 制定一份 CRL- 列表。一个有用的观察（我们将对此作更详细的解释）是，A 具有不动点的情形可以约化为自同构情形（即引理 2.3.2）。这一思想出现在 [19, 引理 4], 的证明中，该引理涉及循环群，但它很容易推广到一般群。我们提醒读者，我们用 ρ_r表示一个群（这一点必须从上下文中明确）在其自身上的右正则表示。

引理 2.3.3. 设 G 是一个群， $b \in G$, α 是 G 的自同构， A 是 G 的仿射置换，且 $\mathfrak{f} \in G$ 是 A 的不动点。那么 $A = \rho_{\mathrm{r}}(\mathfrak{f})^{-1}\alpha\rho_{\mathrm{r}}(\mathfrak{f})$。特别地， $\rho_{\mathrm{r}}(\mathfrak{f})$ 是从 Γ_α 到 Γ_A 的有向图同构。

证明。等式 $x^A = x$ 等价于 $b = (\mathfrak{f}^{-1})^\alpha\mathfrak{f}$。对于每个 $x \in G$，我们有

$$x^{\rho_{\mathrm{r}}(\mathfrak{f})^{-1}\alpha\rho_{\mathrm{r}}(\mathfrak{f})} = (x\mathfrak{f}^{-1})^{\alpha\rho_{\mathrm{r}}(\mathfrak{f})} = x^\alpha(\mathfrak{f}^{-1})^\alpha\mathfrak{f} = x^\alpha b = x^A,$$

如所需。“特别是”这一陈述源于一个众所周知（且容易验证）的事实：对于每个集合 X，以及所有 ψ、$\psi' \in \in$对称群(X)，置换 ψ 将 $x \in X$ 映射到 $y \in X$，当且仅当其 ψ'-共轭 $(\psi')^{-1}\psi\psi'$ 将 $x^{\psi'}$ 映射到 $y^{\psi'}$。 □

引理 2.3.3 对我们很有意义，因为有如下初等观察。

引理2.3.4.设 X 是一个有限集，且设 ψ_1、$\psi_2 \in$ 对称群(X) 是共轭置换，即 $\psi_2 = (\psi')^{-1}\psi_1\psi'$。如果 $\mathcal{L}$ 是 ψ_1的 CRL 列表，那么 $\{(r^{\psi'}, l) : (r, l) \in \mathcal{L}\}$ 是 ψ_2的 CRL 列表。

证明。显然，因为（y_0， y_1，…， y_{l-1}）是 ψ_1 的一个循环，当且仅当

$$(y_0^{\psi'}, y_1^{\psi'}, \ldots, y_{l-1}^{\psi'})$$

是 ψ_2 的一个循环（见引理 2.3.3 证明的最后一句）。 □

结合引理 2.3.3 和引理 2.3.4，可得如下结果。

引理2.3.5。设 G 是一个有限群， $b \in G$、 α 是 G 的一个自同构， A 是 G 的仿射置换 $x \mapsto x^\alpha b$，且 $\mathfrak{f} \in G$ 是 A 的一个不动点。若 $\mathcal{L}$ 是 α 的一个 CRL 列表，则 $\{$（$r\mathfrak{f}$， l）：（r， l）$\in \mathcal{L}\}$ 是 A 的一个 CRL 列表。

现在我们可以为每个有限素数循环群上的每个仿射置换构造一个 CRL 列表。

命题 2.3.6。设 p 是一个素数， $v \in \mathbb{N}^+$，且 a， $b \in \mathbb{Z}$ 满足 $p \nmid a$。表 2 给出了仿射置换 A： $x \mapsto ax + b$ 在 $\mathbb{Z}/p^v\mathbb{Z}$ 上的一个 CRL 列表 $\mathcal{L}$（p^v， a， b），使用以下条件记号。

- 如果 p 为奇数，我们用 $\mathfrak{r}$表示模 p^v的一个固定原根。
- 若 $\nu_{(v)p}(b) \geq \nu_{(v)p}(a-1)$，令

$$\mathfrak{f} := -\frac{b}{p^{\nu_p^{(v)}(a-1)}} \cdot \mathrm{inv}_{p^{v-\nu_p^{(v)}(a-1)}}\left(\frac{a-1}{p^{\nu_p^{(v)}(a-1)}}\right).$$

No.	p^v、 a、 b 的情形	$\mathcal{L}(p^v, a, b)$ 的元素
1	$p > 2$, $\nu_p^{(v)}(b) \geq \nu_p^{(v)}(a-1)$	$\left(\mathfrak{r}^j p^t \mathfrak{f},\ \frac{\phi(p^{v-t})\text{最大公约数}}{\text{数}\left(\frac{\phi(p^v)\,\mathrm{ord}(a)}{\phi(p^{v-t})}\right)}\right)$ 对于 $t = 0, 1, \ldots, v$ 和 $j = 0, 1, \ldots, \left(\frac{\phi\text{的最大公约数}}{(p^v)\ \mathrm{ord}\ (a)}\ \frac{\phi\ (}{p^{v-t})}\right) - 1$。

2	$p>2$, $\nu_p^{(v)}(b)<\nu_p^{(v)}(a-1)$	$\left(j, p^{v-\nu_p^{(v)}(b)}\right)$ 对于 $j=0,1,\ldots,p^{\nu_p^{(v)}(b)}-1$.
3	$p=2$, $v\le 2$, $a=1$	$(j, \text{aord}(b))$ 对于 $j=0,1,\ldots,\frac{2^v}{\text{aord}(b)}-1$.
4	$p^v=4$, $a=3$, $b=0$	(0, 1), (1, 2), (2, 1).
5	$p^v=4$, $a=3$, $b=2$	(0, 2), (1, 1), (3, 1).
6	$p^v=4$, $a=3$, $2\nmid b$	(0, 2), (2, 2).
7	$p=2$, $v\ge 3$, $\nu_2^{(v)}(b)\ge\nu_2^{(v)}(a-1)$, $a\equiv 1$ (模 4)	$(\mathfrak{f}, 1)$, $(2^{v-1}+\mathfrak{f}, 1)$; $\left(5^j2^t+\mathfrak{f}, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{a\text{的阶}}, 2^{v-t-2}\right)}\right)$, $\left(-5^j2^t+\mathfrak{f}, \frac{2^{v-t-2}}{\gcd\left(\frac{2^{v-2}}{a\text{的阶}}, 2^{v-t-2}\right)}\right)$ 对于 $t=0,1,\ldots,$ 和 $j=0,1,\ldots,$ 最大公约数$\left(\frac{2^{v-2}}{a^{\text{ord}()}}-2^{v-t-2}\right)-1$。
8	$p=2$, $v\ge 3$, $\nu_2^{(v)}(b)\ge\nu_2^{(v)}(a-1)$, $a\equiv 3$ (模 4)	$(\mathfrak{f}, 1)$, $(2^{v-1}+\mathfrak{f}, 1)$; $(j\cdot\text{阶}(-a)+\mathfrak{f}, 2)$ 对于 $j=1,2,\ldots,\frac{2^{v-1}}{\text{阶}(-a)}-1$; $\left(5^j2^t+\mathfrak{f}, \frac{\text{阶}(-a)}{2^t}\right)$, $\left(-5^j2^t+\mathfrak{f}, \frac{\text{阶}(-a}{2^t}\right)$ 对于 $t=0$、1、.....、$\log_2(\text{ord}(-a))-1$ 以及 $j=0$、1、…、$\frac{2^{v-2}}{\text{ord}(-a)}-1$.
9	$p=2$, $v\ge 3$, $\nu_2^{(v)}(b)<\nu_2^{(v)}(a-1)$, $a\equiv 1$ (模 4)	$\left(j, 2^{v-\nu_2^{(v)}(b)}\right)$ 对于 $j=0,1,\ldots,2^{\nu_2^{(v)}(b)}-1$.
10	$p=2$, $v\ge 3$, $\nu_2^{(v)}(b)<\nu_2^{(v)}(a-1)$, $a\equiv 3$ (模 4)	$(b\cdot j, 2\ \text{ord}(-a))$ 对于 $j=0,2,3,4,\ldots,\frac{2^{v-1}}{\text{ord}(-a)}$.

表 2：有限主循环群的仿射置换 CRL 列表。

证明。首先，我们注意到，如果 $\nu_p^{(v)}(b)\ge\nu_p^{(v)}(a-1)$，那么 $\mathfrak{f}$ 是不动点of A。事实上，$x\in\mathbb{Z}/p^v\mathbb{Z}$ 是不动点 of A 当且仅当

$$\begin{aligned}
&ax+b\equiv x \pmod{p^v} \Leftrightarrow (a-1)x\equiv -b \pmod{p^v}\\
&\Leftrightarrow \frac{a-1}{p^{\nu_p^{(v)}(a-1)}}x\equiv -\frac{b}{p^{\nu_p^{(v)}(a-1)}} \pmod{p^{v-\nu_p^{(v)}(a-1)}}\\
&\Leftrightarrow x\equiv -\frac{b}{p^{\nu_p^{(v)}(a-1)}}\cdot \text{inv}_{p^{v-\nu_p^{(v)}(a-1)}}\left(\frac{a-1}{p^{\nu_p^{(v)}(a-1)}}\right)=\mathfrak{f} \pmod{p^{v-\nu_p^{(v)}(a-1)}}.
\end{aligned}$$

因此，表 2 中情形 1、7 和 8 所指定的 A 的 CRL 列表，可由引理2.3.5以及从表1中读出的对应 μ_a 的 CRL 列表推出。此外，表 2 中的情形 3–6 很容易分别验证。剩下需要论证的是表 2 中情形 2、9 和 10 所指定的 CRL 列表，我们现在就来说明。

- 情形2：我们注意到，在此情形下，$a\equiv 1$ （模 p）必然成立。模p^v 中与 1 模 p 同余的单位构成 $(\mathbb{Z}/p^v\mathbb{Z})^*$ 的唯一循环的 Sylow p-子群，其阶为 p^{v-1}。对于每个 $t\in\{0,1,\ldots,v-1\}$，单位元 $1+p^{v-t}$ 的阶为

p^t，因此，模 p^v 的所有阶为 p^t 的单位都是 $1+p^{v-t}$ 的幂，且其指数与 p 互素，反之亦然。因此，利用引理 2.3.1 的“特别地”结论，以及 ord(A) 是 p 的幂这一事实，不失一般性地，我们可以假设 $a = 1 + p^{v-t}$，其中 $t \in \{0, 1 \ldots, v-1\}$。我们注意到$v - t = \nu_p^{(v)}(a-1)$，因此根据各情形的假设，$v - t > \nu_p^{(v)}(b)$。对于每个 $x \in \mathbb{Z}/p^v\mathbb{Z}$，都有 $A(x) = ax + b = (1 + p^{v-t})x + b \equiv x \left(\text{模 } p^{\nu_p^{(v)}(b)} \right)$。因此，元素 0、1、.....、$p^{\nu_p^{(v)}(b)} - 1$ 分别位于 A 的互不相同的循环上。另一方面，根据 [13, 表 3]，A 恰有 p^v 个 aord(b) $= p^{\nu_p^{(v)}(b)}$ 循环，每个循环的长度为 aord(b) $= p^{v-\nu_p^{(v)}(b)}$，所以这些元素是 A 所有循环的代表元，并且 $\{(j, p^{v-\nu_p^{(v)}(b)}) : j = 0, 1, ..., p^{\nu_p^{(v)}(b)} - 1\}$ 是 A 的一个 CRL 列表，证毕。

- 情况9：这可以用与情况2类似的方法处理。我们注意到，单位元$5 = 1+2^2$ 的乘法阶为 2^{v-2}，并生成$(\mathbb{Z}/2^v\mathbb{Z})^*$的一个指标为 2 的循环子群，该子群恰好由那些模4 同余于1的单位元组成。对于每个 $t \in \{0, 1, \ldots, v-2\}$，单位元 $1 + 2^{v-t}$ 属于该子群且其阶为 2^t，因此，任何阶为 2^t 且模4 同余于1的单位元，都是 $1 + 2^{v-t}$ 的奇数次幂；反之亦然。利用引理 2.3.1 的“特别地”表述，并注意到 ord(A) 是 2 的幂次，我们不妨设 $a = 1 + 2^{v-t}$，其中 $t \in \{0, 1, \ldots, v-2\}$，而论证的其余部分与情况 2 中的论证类似，从而得到$\{(j, 2^{v-\nu_{(v)\,2}(b)}) : j = 0, 1, \ldots, 2^{\nu_{(v)\,2}(b)} - 1\}$ 是 A 的 CRL 列表。

- Case 10: Due to $-a \equiv 1 \pmod 4$, we may assume without loss of generality that $-a = 1 + 2^{v-t}$ for some $t \in \{0, 1, \ldots, v-2\}$ (see the argument for case 9). Let A' be the affine function $x \mapsto -x + b$ of $\mathbb{Z}/2^{v-t}\mathbb{Z}$. For each $x \in \mathbb{Z}/2^v\mathbb{Z}$, we have $A(x) = -(1 + 2^{v-t})x + b \equiv -x + b \pmod{2^{v-t}}$. This means that elements of $\mathbb{Z}/2^{v-t}\mathbb{Z}$ that lie on different cycles of A' also lie on different cycles of A (we remind the reader that $\mathbb{Z}/2^{v-t}\mathbb{Z} \subseteq \mathbb{Z}/2^v\mathbb{Z}$ by our convention on the underlying set of $\mathbb{Z}/m\mathbb{Z}$ stated at the beginning of this subsection). Now, A' is an involution without fixed points (because $2 \nmid b$) and thus consists of 2^{v-t-1} transpositions. But by [13, Table 4], A has exactly 2^{v-t-1} cycles. Indeed, if $a = -5^n$, then $2^t = \mathrm{ord}(-a) = 2^{v-2-\nu_2^{(v-2)}(n)}$, and therefore $t = v-2-\nu_2^{(v-2)}(n)$, whence the cycle number $2^{1+\nu_2^{(v-2)}(n)}$ specified in [13, Table 4] equals 2^{v-t-1}. Therefore, any set of of representatives for the cycles of A' on $\mathbb{Z}/2^{v-t}\mathbb{Z}$ is also a set of representatives for the cycles of A on $\mathbb{Z}/2^v\mathbb{Z}$, all of which are of length 2^{t+1} by [13, Table 4]. Thus, in order to find a CRL-list for A, it suffices to find cycle representatives for A'. To that end, we first assume that $b = 1$. Then every cycle (i.e., transposition) of A' on $\mathbb{Z}/2^{v-t}\mathbb{Z}$ contains exactly one element from the "left half" $\{1, 2, \ldots, 2^{v-t-1}\}$ and one from the "right half" $\{2^{v-t-1} + 1, 2^{v-t-1} + 2, \ldots, 2^{v-t} - 1, 2^{v-t} \equiv 0\}$. It follows that $\{0, 2, 3, 4, \ldots, 2^{v-t-1}\}$ is a set of representatives for the cycles of A', and this matches with the CRL-list for A specified in Table 2. For general b, we observe that

$$A' = (x \mapsto -x + b) = (x \mapsto b^{-1}x) \cdot (x \mapsto -x + 1) \cdot (x \mapsto bx),$$

由此，引理 2.3.4 使我们得以得出结论：

$$b \cdot \{0, 2, 3, 4, \ldots, 2^{v-t-1}\} = \{0, 2b, 3b, 4b, \ldots, 2^{v-t-1}b\}$$

是 A' 的循环的代表元集合，符合要求。

□

既然我们已经知道了每个有限主循环群的每个仿射置换所对应的CRL列表，下面讨论如何处理一般的有限循环群。通过将群 $\mathbb{Z}/m\mathbb{Z}$ 与直积 $\prod_{p|m} \mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$ 按照中国剩余定理进行识别，我们可以将任意仿射置换 $A : x \mapsto ax + b$ 上的 $\mathbb{Z}/m\mathbb{Z}$视为“函数张量积” $\bigotimes_{p|m} A_p$，其中 A_p 是 A 模$p^{\nu_p(m)}$的约化，即由备注 2.1.2 引入的 $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$上的仿射置换 $x \mapsto ax+b$。也就是说，在这种识别下， A 变为其约化 A_p 的逐分量应用。由此得到下面这个更一般的问题，我们接下来将解决该问题。

问题 2.3.7. 给定有限集 $X_1, X_2, \ldots, X_n$，置换 $\psi_j \in$对称群(X_j)，其中$j = 1,2, \ldots, n$，以及 ψ_j的 CRL 列表 $\mathcal{L}_j$，其中 $j = 1, 2, \ldots, n$，构造 $\psi := \bigotimes_{j=1}^{n} \psi_j \in$对称群$(\prod_{j=1}^{n} X_j)$ 的 CRL 列表 $\mathcal{L}$。

在本小节余下部分，我们使用问题2.3.7 中确定的记号。我们将 $\mathcal{L}_j^{(1)} \subseteq X_j$ 记为 $\mathcal{L}_j$ 中各对的第一项集合（即由 $\mathcal{L}_j$ 展示的 ψ_j 的循环代表元集合），对于 $r \in \mathcal{L}_j^{(1)}$，我们将 $r^{\langle \psi_j \rangle}$ 记为 r 在置换群 $\langle \psi_j \rangle$ 的作用下的轨道（即 r 的ψ_j-循环上的点集）。

对于每个 $\vec{r} = (r_1, r_2, \ldots, r_n) \in \prod_{j=1}^{n} \mathcal{L}_j^{(1)}$，我们定义 $B_{\vec{r}} := \prod_{j=1}^{n} r_j^{\langle \psi_j \rangle}$。这些集合 $B_{\vec{r}}$ 构成了 $\prod_{j=1}^{n} X_j$的一个划分，并且每个集合 $B_{\vec{r}}$都是 ψ的若干循环的并。因此，只需对每个 $\vec{r}$求出限制 $\psi_{|B_{\vec{r}}}$的 CRL 列表 $\mathcal{L}_{\vec{r}}$，然后定义 $\mathcal{L} := \bigcup_{\vec{r}} \mathcal{L}_{\vec{r}}$。

因此，假设 $\vec{r}$是固定的。对于 $j = 1, 2, \ldots, n$，我们用 $l_j = l_j(\vec{r})$ 表示 r_j的 ψ_j-循环周期长度。 ψ在 $B_{\vec{r}}$上的每个循环的长度都是 $l_{\vec{r}}$:=lcm$(l_1, l_2, \ldots, l_n)$，且恰有 $\left(\prod_{j=1}^{n} l_j\right) / l_{\vec{r}}$个这样的循环（另见 [77, 引理 2.1]）。剩下的工作是为它们求出代表元。

我们考虑这个双射

$$\iota_{\vec{r}} : \prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z} \to B_{\vec{r}}, (k_1, k_2, \ldots, k_n) \mapsto (\psi_1^{k_1}(r_1), \psi_2^{k_2}(r_2), \ldots, \psi_n^{k_n}(r_n)).$$

如果通过这个双射将 $B_{\vec{r}}$ 与 $\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$ 视为相同，那么 ψ 在 $B_{\vec{r}}$ 上的作用就转化为函数的作用

$$\mathfrak{s}_{\vec{r}} : \prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z} \to \prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}, (k_1, k_2, \ldots, k_n) \mapsto (k_1 + 1, k_2 + 1, \ldots, k_n + 1),$$

当然，每个显示的加法都是模相应的 l_j进行的。因此，只需找出 $\mathfrak{s}_{\vec{r}}$在 $\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$上的循环的一组代表元，再将该集合通过 $\iota_{\vec{r}}$映射即可。为了简洁地描述 $\mathfrak{s}_{\vec{r}}$ 的一组特定周期代表元，我们引入以下辅助概念。

定义 2.3.8。我们用 $\pi(l_{\vec{r}})$表示 $l_{\vec{r}}$的所有素因子的集合。

(1) 若函数 $\mathcal{I}$：$\pi(l_{\vec{r}}) \to \{1,2,\ldots,n\}$ 满足：对于每个 $p \in \pi(l_{\vec{r}})$，都有 $\nu_p(l_{\mathcal{I}(p)}) = \max\{\nu_p(l_j) : j = 1,2,\ldots,n\}$，则称其为 $\vec{r}$-可容许索引函数。(2) 若 $\mathcal{I}$ 是一个 $\vec{r}$-可容许索引函数，则元组

$$(k_1,k_2,\ldots,k_n) \in \prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$$

若对于每个 $p \in \pi(l_{\vec{r}})$，都有 $k_{\mathcal{I}(p)} \equiv 0 \left(\mathrm{mod}\ p^{\nu_p(l_{\mathcal{I}(p)})}\right)$，

则称其为 $\mathcal{I}$-良好元组。

(3) 对于每个 $\vec{r}$-可容许索引函数 $\mathcal{I}$，我们用 $\mathrm{Good}_{\vec{r}}(\mathcal{I})$ 表示 $\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$ 中所有 $\mathcal{I}$-良好元组构成的集合。

下面的结果解决了问题 2.3.7。

命题 2.3.9. 设 $\mathcal{I}$ 是一个 $\vec{r}$-可容许索引函数。那么，$\mathrm{Good}_{\vec{r}}(\mathcal{I})$ 是 $\mathfrak{s}_{\vec{r}}$ 在 $\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$ 上各循环的代表元集合。等价地，

$$\iota_{\vec{r}}(\mathrm{Good}_{\vec{r}}(\mathcal{I})) \times \{l_{\vec{r}}\}$$

是 $\psi_{|B_{\vec{r}}}$ 的一个 CRL 列表，因此

$$\bigcup_{\vec{r} \in \prod_{j=1}^{n} \mathcal{L}_j^{(1)}} (\iota_{\vec{r}}(\mathrm{Good}_{\vec{r}}(\mathcal{I})) \times \{l_{\vec{r}}\})$$

是 ψ 的一个 CRL 列表。

证明。根据定义和中国剩余定理，$\mathcal{I}$ 中良好元组的数量为

$$\frac{\prod_{j=1}^{n} l_j}{\prod_{p \in \pi(l_{\vec{r}})} p^{\nu_p(l_{\mathcal{I}(p)})}} = \frac{|\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}|}{l_{\vec{r}}},$$

这也等于 $\mathfrak{s}_{\vec{r}}$ 在 $\prod_{j=1}^{n} \mathbb{Z}/l_j\mathbb{Z}$ 上的循环数。因此，只需证明不同的 $\mathcal{I}$-良好元组属于 $\mathfrak{s}_{\vec{r}}$ 的不同循环。设 $\vec{k} = (k_1,k_2,\ldots,k_n)$ 和 $\vec{k'} = (k'_1,k'_2,\ldots,k'_n)$ 是属于 $\mathfrak{s}_{\vec{r}}$ 同一循环的 $\mathcal{I}$-良好元组。这意味着存在一个 $t \in \mathbb{Z}$，使得对每个 $j = 1, 2, \ldots, n$，都有 $k_j + t \equiv k'_j$(模 l_j)。现在，令 $p \in \pi(l_{\vec{r}})$。由于 $t \equiv k'_{\mathcal{I}(p)} - k_{\mathcal{I}(p)}$ $\left(\text{模 } l_{\mathcal{I}(p)}\right)$，且 $\vec{k}$, $\vec{k'}$ 是 $\mathcal{I}$-良好元组，因此可得$t \equiv 0 \left(\text{模 } p^{\nu_p(l_{\mathcal{I}(p)})}\right)$。因为这对每个 $p \in \pi(l_{\vec{r}})$ 都成立，所以我们得出

$$\mathrm{lcm}(l_1,l_2,\ldots,l_n) = l_{\vec{r}} = \prod_{p \in \pi(l_{\vec{r}})} p^{\nu_p(l_{\mathcal{I}(p)})}$$

整除 t，从而对于每个 $j = 1$、2、.....、n，有 $k_j \equiv k'_j$（模 l_j）。这意味着 $\vec{k} = \vec{k'}$，正如所要求的。 □

2.4 仿射离散对数与周期长度

设 $m \geq 1$ 为一个整数，并设 a、$b \in \mathbb{Z}/m\mathbb{Z}$ 满足a与m的最大公约数为 $= 1$。我们考虑仿射置换 $A\colon\ x \mapsto ax + b$，其作用于 $\mathbb{Z}/m\mathbb{Z}$。给定 x、$y \in \mathbb{Z}/m\mathbb{Z}$，我们令

$$\log_A^{(m)}(x,y) := \begin{cases} \infty, & \text{if there is no } k \in \mathbb{Z} \text{ with } A^k(x) = y, \\ \min\{k \in \mathbb{N}_0 : A^k(x) = y\}, & \text{otherwise.} \end{cases}$$

在这一小节中，我们讨论如何计算$\log_A^{(m)}(x,y)$ 以及 x 在 A下的循环长度；后者与前者密切相关，因为它是满足 $A^k(x) = x$ 的最小正整数 k （而$\log_A^{(m)}(x,x) = 0$）。模离散对数在其中发挥重要作用并不令人惊讶，因为它们是$\log_A^{(m)}(x,y)$ 这一概念的特殊情形。也就是说，以 $a \in (\mathbb{Z}/m\mathbb{Z})^*$ 为底、模 m 的 $x \in (\mathbb{Z}/m\mathbb{Z})^*$ 的离散对数，记作$\log_a^{(m)}(x)$，等于 $\log_{\mu_a}^{(m)}(1,x)$。

为了讨论计算细节，我们进行分类讨论。

首先，我们假设 $a \equiv 1 \pmod m$，这是一个无需计算离散对数的简单情形。事实上，此时有 $A^k(x) = x + kb \equiv y \pmod m$ 当且仅当 $kb \equiv y - x \pmod m$。最后一个同余式在 k 中有解，当且仅当最大公约数$(b,m) \mid y - x$，在这种情况下，该同余式等价于

$$\frac{b}{\gcd(b,m)}k \equiv \frac{y-x}{\gcd(b,m)} \pmod{\frac{m}{\gcd(b,m)}} \tag{10}$$

且具有最小解

$$\left(\frac{y-x}{\gcd(b,m)} \cdot \mathrm{inv}_{m/\gcd(b,m)}\left(\frac{b}{\gcd(b,m)}\right)\right) \bmod \frac{m}{\gcd(b,m)} = \log_A^{(m)}(x,y).$$

我们注意到，在 $x = y$ 的情况下，同余式 (10) 的最小正解，因此 x 在模 m 下受 A 作用时的周期长度，为 $m/\gcd(b,m)$。

- 现在假设 $a \not\equiv 1 \pmod m$。那么 $m > 1$，且由于 $\gcd(a,m) = 1$，有 $a \not\equiv 0 \pmod m$，因此我们可以假设作为一个整数，$a > 1$。当且仅当 $A^k(x) = y$

$$\begin{aligned} & a^k x + \frac{a^k - 1}{a-1} b \equiv y \pmod m \\ \Leftrightarrow\ & a^k(a-1)x + (a^k-1)b \equiv (a-1)y \pmod{(a-1)m} \\ \Leftrightarrow\ & a^k((a-1)x+b) \equiv (a-1)y + b \pmod{(a-1)m}. \end{aligned} \tag{11}$$

为了使同余式 (11) 在 k 中有解，必要条件是 $(a-1)x+b$ 与 $(a-1)y+b$ 模 $(a-1)m$ 的加法阶相同，即

$$\gcd((a-1)x+b,(a-1)m) = \gcd((a-1)y+b,(a-1)m) =: \mathfrak{d}. \tag{12}$$

如果条件（12）成立，则同余式（11）等价于

$$a^k \cdot \frac{(a-1)x+b}{\mathfrak{d}} \equiv \frac{(a-1)y+b}{\mathfrak{d}} \pmod{\frac{(a-1)m}{\mathfrak{d}}},$$

即，也就
是 to

$$a^k \equiv \frac{(a-1)y+b}{\mathfrak{d}} \cdot \mathrm{inv}_{(a-1)m/\mathfrak{d}} \left(\frac{(a-1)x+b}{\mathfrak{d}} \right) \left(\mathrm{mod}\ \frac{(a-1)m}{\mathfrak{d}} \right), \qquad (13)$$

这表明

$$\log_A^{(m)}(x,y) = \log_a^{((a-1)m/\mathfrak{d})} \left(\frac{(a-1)y+b}{\mathfrak{d}} \cdot \mathrm{inv}_{(a-1)m/\mathfrak{d}} \left(\frac{(a-1)x+b}{\mathfrak{d}} \right) \right),$$

约定 $\log_a^{(m)}(x) = \infty$：若 x 不是 a 模 m 的幂。如果$x = y$，则同余式 (13) 中的右端化简为 1，由此 x 在 A 作用下的周期长度等于 a 模 $(a-1)m/\mathfrak{d}$ 的乘法阶。

讨论的结论是，如果存在高效算法，能够计算 $(\mathbb{Z}/m\mathbb{Z})^*$ 中单位的离散对数和乘法阶，那么就能高效计算 $\log_A^{(m)}(x,y)$ 以及 x 在模 m 下经 A 作用后的周期长度。因此，可以在量子计算机上高效计算 $\log_A^{(m)}(x,y)$。事实上，Shor 证明了这类计算机能够高效地计算离散对数并进行整数分解 [63]，后者足以高效计算 $(\mathbb{Z}/m\mathbb{Z})^*$ 中的元素阶；实际上，只需对欧拉函数的函数值 $\phi(m)$ 进行显式因式分解，另见引理 5.1.6(2) 的证明。

3 广义分圆映射的函数图

设 f 是 $\mathbb{F}_q$ 上指标为 d 的广义分圆映射。根据引言，我们回顾记号 C_i 表示 $i \in \{0,1,\ldots,d\}$，其中 $C_d = \{0_{\mathbb{F}_q}\}$；$C_i = \omega^i C$ 表示 $i < d$，它是 $\mathbb{F}_q^* = \langle\omega\rangle$ 的指标为 d 的子群 C 的一个陪集。此外，我们回顾，对于每个 $i \in \{0,1,\ldots,d-1\}$，都有一个自然双射 ι_i：$\mathbb{Z}/s\mathbb{Z} \to C_i = \omega^i C$，$x \mapsto \omega^{i+dx}$。借助该双射，我们将 C_i 视为 $\mathbb{Z}/s\mathbb{Z}$ 的一个副本。只要 f 不将 C_i 恒定地映射到 $C_d = \{0\}$，我们就可以将限制 $f_{|C_i}$ 视为一个仿射函数 A_i：$\mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$。最后，我们回顾诱导函数 $\overline{f}$：$\{0,1,\ldots,d\} \to \{0,1,\ldots,d\}$（满足对每个 i 都有 $f(C_i) \subseteq C_{\overline{f}(i)}$ 的唯一函数）。

本节的目标是介绍一些方法，按照引言中概述的思路，借此理解函数图 Γ_f 的同构类型。

3.1 周期点和 CRL 列表

理解 f 的周期点并找到其 CRL 列表，可以归结为有限循环群仿射映射的相应问题。我们观察到，f 的周期点必然包含在“周期块”中（即满足 i 在 $\overline{f}$ 下为周期的块 C_i）。我们假设，由于 d 足够小，找到 $\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$ 是一项容易完成的任务。我们根据 f 所在的“块循环” $\overline{f}$ 来确定其周期点，因此令 $(i,\ell) \in \overline{\mathcal{L}}$。

如果 $i = d$，那么 $\ell = 1$，需要考虑的唯一点是域元素 0；根据定义，它在 f 下的周期长度为 1。我们将贡献 $\mathcal{L}_d := \{(0,1)\}$ 记入正在构建的 f 的 CRL 列表 $\mathcal{L}$。

现在假设 $i < d$，且 i 在 $\overline{f}$ 下的循环为 $(i_0, i_1, \ldots, i_{\ell-1})$，其中 $i_0 = i$。点 $x \in C_i$ 在 f 下是周期的，当且仅当它在迭代 f^ℓ 下是周期的；该迭代稳定了 C_i，并通过仿射映射 $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ 作用于对应的 $\mathbb{Z}/s\mathbb{Z}$ 副本。换言之，C_i 中 f 的周期点与 $\mathbb{Z}/s\mathbb{Z}$ 中 $\mathcal{A}_i$ 的周期点（经由 ι_i）一一对应，因此可由引理 2.1.14 刻画。注意，在同一 $\overline{f}$-循环的另一个陪集 C_{i_t} 中，f 的周期点集合恰好是迭代集合像 $f^t(\mathrm{per}(f_{|C_i}))$。此外，周期点 $x \in C_i$ 在 f 下的周期长度，是其在 $\mathcal{A}_i$ 下周期长度的 ℓ 倍。由此可知，若 $\mathcal{L}'_i \subseteq \mathbb{Z}/s\mathbb{Z} \times \mathbb{N}^+$ 是 $\mathcal{A}_i$ 的 CRL 列表（可按第 2.3 小节所述确定），那么 $\mathcal{L}_i := \{(\iota_i(r)\ \ell \cdot l) : (r, l) \in \mathcal{L}'_i\}$ 是将 f 限制在整个“由 C_i 张成的陪集循环”（即集合 $\bigcup_{t=0}^{\ell-1} C_{i_t}$）上的 CRL 列表。总之，我们得到以下命题。

命题 3.1.1. 设 $\overline{\mathcal{L}}$ 是 $\overline{f}$ 的 CRL 列表。我们令 $\mathcal{L}_d := \{(0_{\mathbb{F}_q}, 1)\}$。此外，对于满足 $(i, \ell) \in \overline{\mathcal{L}}$ 的 $i < d$，我们如下定义 $\mathcal{L}_i$。设 $(i_0, i_1, \ldots, i_{\ell-1})$ 是 i 的 $\overline{f}$-循环，其中 $i_0 = i$，并设 $\mathcal{L}'_i$ 是仿射映射 $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ of$\mathbb{Z}/s\mathbb{Z}$ 的 CRL 列表。于是我们令 $\mathcal{L}_i := \{(\iota_i(r)\ \ell \cdot l) : (r, l) \in \mathcal{L}'_i\}$。对于每个满足 $(i, \ell) \in \overline{\mathcal{L}}$ 的 $\mathcal{L}_i$，按此定义，我们有 that.

$$\mathcal{L} := \bigcup_{(i,\ell) \in \overline{\mathcal{L}}} \mathcal{L}_i.$$

是 f 的 CRL 列表。

3.2 周期陪集上的诱导子图

我们的下一个目标是理解 Γ_f 上方 x 的f周期点所对应的树$\Gamma_f(x)$，其含义遵循定义 1.4。我们提醒读者，树$\Gamma_f(x)$是为 x 的任意顶点 Γ_f定义的，而不仅仅是周期顶点。一般来说，采用递归方法是有利的，即根据顶点 x 在 Γ_f中的深度来理解树$\Gamma_f(x)$，从叶节点开始，逐步处理到周期顶点，而周期顶点位于递归的末端。不过，在进行这一过程之前，我们必须先理解所有周期块 C_i 的并集上的 Γ_{per} Γ_f（即，在 f作用下i 为周期的块），以此作为过渡。

我们观察到，Γ_{per} 是 f 在所有周期块的并集上的限制 f_{per} 的函数图。正如 f 在指标集$\{0, 1, \ldots, d\}$上具有诱导函数 $\overline{f}$ 一样，限制 f_{per} 具有诱导函数 $\overline{f_{\mathrm{per}}}$，它是 $\overline{f}$ 在其周期点集上的限制。因此，$\overline{f_{\mathrm{per}}}$ 是其定义域上的一个置换，这一点对我们的论证很重要。

与命题 2.1.8中所述情形类似，如果对于给定的周期顶点 x（属于 $\Gamma_{\mathrm{per}} = \Gamma_{f_{\mathrm{per}}}$），我们知道每个树$\Gamma_{\mathrm{per}}(y)$（其中 y 是 Γ^*_{per}中的一个子节点）都具有刚性繁衍，并且知道每个子节点 y 的前 $h = h(y)$ 个繁殖数，其中 h 是树$\Gamma_{\mathrm{per}}(y)$的高度，那么这就唯一刻画了每个树$\Gamma_{\mathrm{per}}(y)$的同构类型，因而也唯一刻画了树$\Gamma_{\mathrm{per}}(x)$的同构类型。事实上，尽管 Γ^*_{per}

在我们此处所考虑的更一般情形中，它本身不必具有刚性繁衍，但我们刚才提到的树确实具有刚性繁衍。更具体地说，我们有如下结果（排除 $i = d$ 不失一般性，因为树$\Gamma_{\mathrm{per}}(0_{\mathbb{F}_q})$ 无论如何都是平凡的）。

定理 3.2.1. 设 f_{per} 和 $\overline{f_{\mathrm{per}}}$ 如上。此外，设 $i \in \mathrm{dom}(\overline{f_{\mathrm{per}}}) = \mathrm{per}(\overline{f})$，其中 $i < d$，并设 $(i_0, i_1, \ldots, i_{\ell-1})$ 为 $i = i_0$ 在 $\overline{f_{\mathrm{per}}}$ 下的循环。我们将记号 i_t 扩展到任意 $t \in \mathbb{Z}$，方法是对 ℓ 取模来约化 t（例如，$i_\ell = i_0$）。对于 $t = 0, 1, \ldots, \ell-1$，记 A_{i_t}: $z \mapsto \alpha_{i_t} z + \beta_{i_t}$ 为描述 f_{per}（或等价地，f）如何从 C_{i_t} 映射到 $C_{i_{t+1}}$ 的 $\mathbb{Z}/s\mathbb{Z}$ 的仿射映射，并令 $\varphi_{i_t} := \mu_{\alpha_{i_t}} : z \mapsto \alpha_{i_t} z$ 为 $\mathbb{Z}/s\mathbb{Z}$ 的相应群自同态。那么，对于任意正整数 k，以下结论成立。如果 $x \in C_i$ 在 Γ^*_{per} 中至少有 k 个后继代（注意，这些后继代不必完全包含在 C_i 中），那么

$$\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) = \left| \ker \left(\prod_{j=0}^{k-1} \varphi_{i_{-k+j}} \right) : \ker \left(\prod_{j=0}^{k-2} \varphi_{i_{-k+j}} \right) \right| = \frac{\gcd\left(\prod_{j=0}^{k-1} \alpha_{i_{-k+j}}, s \right)}{\gcd\left(\prod_{j=0}^{k-2} \alpha_{i_{-k+j}}, s \right)},$$

独立于 x。

证明。该定理可视为定理 2.1.21（对应于 $\ell = 1$这一情形）的推广；同样，其证明也是定理 2.1.21 证明的推广。我们对 k进行归纳。对于 $k = 1$，我们注意到$C_{\overline{f_{\mathrm{per}}}^{-1}(i)} = C_{i_{-1}}$是唯一映射到 C_i的陪集。因此

$$\mathrm{proc}_1^{(\Gamma^*_{\mathrm{per}})}(x) = \#\text{ children of } x \text{ in } \Gamma^*_{\mathrm{per}} = |\{y \in \mathbb{Z}/s\mathbb{Z} : A_{i-1}(y) = x\}| = |\ker(\varphi_{i-1})|,$$

这就推出了 $k = 1$ 的情形，因为根据定义，群自同态的空乘积是恒等函数 id。

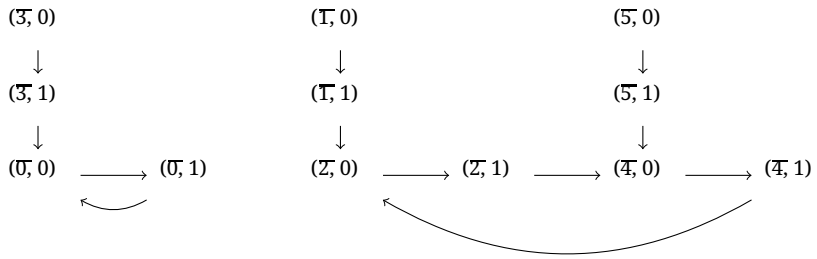
现在我们假设 $k \geq 2$，并假设对于C_{i_t} 中的点，当 $t \in \mathbb{Z}$ 任意时，该命题对不超过 $k-1$ 的情形均成立。对于 $h = 1$、2、.....、$k-1$ 和 $t \in \mathbb{Z}$，我们将 $\mathrm{proc}_{i_t,h}$ 记为所有在Γ^*_{per}中至少有 h 代后继的顶点$y \in C_{i_t}$ 的共同繁衍数 $\mathrm{proc}^{(\Gamma^*_{\mathrm{per}})}{}_h\,(y)$。对于每个 $h = 1$、2、.....、$k-1$，在 Γ^*_{per} 中从C_{i_t} 内至少有h 代后继的顶点出发、长度为 h 的路径的端点数为 $|\ker(\prod_{j=0}^{h-1} \varphi_{i_{t-h+j}})|$；通过对 h作简单归纳可知，该数也等于 $\prod_{j=0}^{h-1} \mathrm{proc}_{i_{t-j},h-j}$。由此可得，对于每个至少有 k 代后继的$x \in C_i = C_{i_0}$，有

$$\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) \cdot \left| \ker \left(\prod_{j=0}^{k-2} \varphi_{i_{-k+j}} \right) \right| = \mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) \cdot \prod_{j=0}^{k-2} \mathrm{proc}_{i_{-j-1},k-1-j} =$$

$$(\#\text{ endpoints of paths of length } k \text{ starting at } x) = \left| \ker \left(\prod_{j=0}^{k-1} \varphi_{i_{-k+j}} \right) \right|,$$

由此可立即得到 $\mathrm{proc}^{(\Gamma^*_{\mathrm{per}})}{}_k\,(x)$ 的所述公式。 □

下面的例子突出说明了一些通常并不成立的性质。例3.2.2。设 $q = 13$、 $d = 2$（从而 $s = 6$）以及 $\overline{f} =$ （0,1）（2），使得 $\Gamma_f = \Gamma_{\mathrm{per}}$。此外，我们假设 A_0（z）$= z$ 以及 A_1（z）$= 2z$。那么， $f_{|\mathbb{F}_q^*}$ 具有如下函数图，其中我们将 C_i 中与 $z \in \mathbb{Z}/6\mathbb{Z}$ 对应的点记为（z， i）：



我们注意到以下几点。

- C_0 中的周期顶点上的有根树与 C_1中的周期顶点上的有根树不同构。

- C_1 中的暂态顶点在 Γ_f 中的树高严格大于 C_1中的周期顶点。更具体地说，C_1 中的暂态顶点恰好是树高为 1 的顶点，而 Γ_f中的周期顶点是树高为 0 的顶点。

- 在 Γ_f 中位于 C_0 中顶点上方的可能树高集合为 $\{0, 2\}$，这不是一个整数区间。

3.3 有根树

我们采用递归方法，针对每个顶点 $x \in \mathbb{F}_q = \mathrm{V}(\Gamma_f)$，理解 $\mathrm{Tree}^{\Gamma_f}(x)$。我们根据唯一的 $i \in \{0, 1, \ldots, d\}$such that $x \in C_i$，分三步进行。除非 $i = d$，我们的目标是找到 $\mathbb{Z}/s\mathbb{Z}$ 的算术划分 $\mathcal{P}_i$，它通过双射 ι_i （我们也称之为 C_i 的算术划分）对应于 C_i 的一个划分；使得对于来自该划分同一块 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 的顶点 $x \in C_i$，$\mathrm{Tree}^{\Gamma_f}(x)$ 的同构类型保持不变，记作 $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$。我们还希望明确地用 $\vec{\nu}^{(\mathcal{P}_i)}$ 表示 $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$，并验证对于固定的 d， $\mathcal{P}_i$ 的最大（算术运算）复杂度（按照定义 1.5(3) 的含义）为 $O(d^2 \,\mathrm{mpe}(q-1)) \subseteq O(d^2 \log q)$，其中 $\mathrm{mpe}(m) := \max_p \nu_p(m)$，对 $m \in \mathbb{N}^+$ 而言（并且 p 遍历所有素数）。首先，我们引入一些记号。

- 对于 $M \subseteq \mathbb{F}_q$和 $x \in \mathbb{F}_q$， 记号树$^{\Gamma_f}(x, M)$表示 Γ_f 的子图的有向图同构-类型，该子图是一棵以 x为根的有根树，通过将所有有根树树$^{\Gamma_f}(y)$连接到该根上得到，其中 y是 f-暂态原像 x，且具有 $y \in M$。我们注意到，树$^{\Gamma_f}(x, \mathbb{F}_q) =$树$^{\Gamma_f}(x)$。

- 令 $M \subseteq \mathbb{F}_q$，并假设 $\mathcal{P}$ 是 C_i 的一个算术划分，且具有一组固定序列的生成同余式，使得对于 $x \in C_i$，有根树同构类型 $\mathrm{Tree}^{\Gamma_f}(x, M)$ 仅取决于 x 所在的 $\mathcal{P}$ 的块 $\mathcal{B}(\mathcal{P}, \vec{\nu})$ （而不取决于 x 本身）。于是我们用 $\mathrm{Tree}_i(\mathcal{P}, M, \vec{\nu})$ 表示该同构类型。我们还令 $\mathrm{Tree}_i(\mathcal{P}, \vec{\nu}) := \mathrm{Tree}_i(\mathcal{P}, \mathbb{F}_q, \vec{\nu})$。

• 若 $\mathfrak{I}_1, \mathfrak{I}_2, \ldots, \mathfrak{I}_N$ 是有根树的同构类型，则它们的和 $\mathfrak{I}_1 + \mathfrak{I}_2 + \cdots + \mathfrak{I}_N$ 定义为如下有根树同构类型：将这些有根树的互不相交副本在各自的根处粘合而成。此加法使有根树同构类型构成一个类大小幺半群，其单位元是平凡有根树同构类型（一个没有弧的单个顶点）。

• 若 $\mathfrak{I}$ 是一个有根树同构类型，我们用 $\mathfrak{I}^+$ 表示这样得到的有根树同构类型：将 $\mathfrak{I}$ 的一份副本通过一条从旧根指向新根的弧连接到一个新根上。例如，从平凡有根树同构类型开始反复进行这一操作，可以得到那些有限有向图，它们都是有向路径。

• 若 $\mathfrak{I}$ 是一个有根树同构类型，且 n 是一个非负整数，我们递归地定义其倍数 $n \cdot \mathfrak{I} = n\mathfrak{I}$ 如下。$0\mathfrak{I}$ 是平凡有根树同构类型，而 $(n+1)\mathfrak{I} := n\mathfrak{I} + \mathfrak{I}$。

• 鉴于前面两条要点，有根树同构类型的非负整数线性组合 $n_1\mathfrak{I}_1 + n_2\mathfrak{I}_2 + \cdots + n_N\mathfrak{I}_N$ 定义良好。

• 如果 $\mathcal{X}_1$、$\ldots \mathcal{X}_n$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分，那么 $\bigwedge_{k=1}^{n} \mathcal{X}_k$ 表示所有 $\mathbb{Z}/m\mathbb{Z}$ 划分构成的格中 $\mathcal{X}_k$ 的下确界（即 $\mathcal{X}_k$ 的最粗公共细化）。等价地，若为每个 $\mathcal{X}_k$ 固定一个生成 m 同余式序列，则 $\bigwedge_{k=1}^{n} \mathcal{X}_k$ 是由这些序列串接后生成的 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分。

在我们方法的第一步中，我们考虑 $i \neq d$ 和 C_i 不是周期陪集的情形（我们将这样的陪集称为暂态陪集）。对于这种情形，我们按照树 $\Gamma_{\mathcal{T}}(i)$ 的高度进行递归。递归的基情形是 i 是 $\Gamma_{\overline{f}}$ 中的叶。此时，$x \in C_i$ 是 Γ_f 中的叶，即 $x \notin \mathrm{im}(f)$。这意味着，树 $\Gamma_f(x)$ 由单个顶点 x 组成，且没有弧。因此，我们可以取 $\mathcal{P}_i$ 为 $\mathbb{Z}/m\mathbb{Z}$ 的平凡划分 $\mathcal{P}(\emptyset) = \{\mathbb{Z}/m\mathbb{Z}\}$，并取树 $_i(\mathcal{P}_i, \emptyset)$ 为平凡有根树同构类型。注意，$\mathrm{AC}(\mathcal{P}_i)$（即 $\mathcal{P}_i$ 的复杂度）为 $0 = \mathfrak{v}_i$，其中，对于一般的 $j \in \{0, 1, \ldots, d\}$，我们定义 $\mathfrak{v}_j := |\mathrm{V}(\text{树}\ \Gamma_{\mathcal{T}}(j))| - 1$，即在 $\Gamma_{\overline{f}}$ 的对应子树中严格位于 j 之上的顶点数。

现在我们假设 C_i 是暂态的，$\mathrm{Tree}^{\Gamma}\mathcal{T}(i)$ 的高度为 $h \geq 1$，并且所有高度小于 h 的暂态陪集都已通过算术划分 $\mathcal{P}_j$“处理完毕”，使得 $\mathrm{AC}(\mathcal{P}_j) \leq \mathfrak{v}_j$。特别地，如果 $\overline{f}^{-1}(\{i\}) = \{j_1, j_2, \ldots, j_K\}$，那么对于每个 $t = 1、2、\ldots\ldots、K$，我们都有 C_{j_t} 的一个算术划分 $\mathcal{P}_{j_t}$，并带有长度为 $m_{j_t} \leq \mathfrak{v}_{j_u}$ 的显式生成 s-同余式序列，使得对于从 $\mathcal{P}_{j_t}$ 的公共块 $\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ 中选取的所有顶点 $y \in C_{j_t}$，同构类型 $\mathrm{Tree}^{\Gamma_f}(y)$ 都相同，并且我们明确了解每一种这样的同构类型 $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$。现在，由于 $x \in C_i$ 在 f 下的每个原像 y（即，Γ_f^* 中 x 的每个子节点）都必须属于 C_{j_t} 中的某个陪集，其中 $t = 1、2、\ldots\ldots、K$，因此

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_f}(x, C_{j_t}).$$

此外，对于固定的 $t \in \{1, 2, \ldots, K\}$，我们可以写成

$$\mathrm{Tree}_{\Gamma_f}(x, C_{j_t}) = \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (|f^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})| \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+).$$

考虑引理 2.2.2 中的算术划分 $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$of C_i ，其具有长度为$m_{j_t} + 1$ 的显式生成 s-同余式序列。如果 x 位于块 $\mathcal{B}(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ 中，对于某个固定的 $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$，那么

$$|f^{-1}(\{x\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})| = \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$$

根据引理 2.2.2，因此

$$\mathrm{Tree}_{\Gamma_f}(x, C_{j_t}) = \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+),$$

独立于 x 本身。现在，令 $\mathcal{P}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$，将其视为 C_i 的算术划分，其张成序列的长度为 $m_i := \sum_{t=1}^{K} (m_{j_t} + 1)$。我们可以将每个逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$ 视为逻辑符号元组$_{\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}}$的串接，其中 $t = 1$取值为 2，…，K；若 $x \in C_i$属于$\mathcal{P}_i$的块 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$，则对于每个 $t = 1$取值为 2，…，K，$x \in C_i$也属于 $\mathcal{P}'_{j_t}$的块 $\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$，因此

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^+),$$

独立于 x 本身，正如所要求的那样。此外，我们注意到

$$\mathrm{AC}(\mathcal{P}_i) \le \sum_{t=1}^{K} \mathrm{AC}(\mathcal{P}'_{j_t}) \le \sum_{t=1}^{K} (\mathfrak{v}_{j_t} + 1) = \mathfrak{v}_i.$$

综上，我们得到如下结果。

命题3.3.1。对于每个 $\overline{f}$-暂态 $i \in \{0, 1, \ldots, d-1\}$，$C_i$ 的算术运算划-分 $\mathcal{P}_i$ 连同一个显式的 s-同余式张成序列，其长度为m_i，以及相关的有根树同构类型 树$_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 对于 $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$，可以如下通过对 $h_i :=$ ht(树$^{\Gamma}\mathcal{T}(i)$) 进行递归来定义。

（1）如果 $h_i = 0$，我们可以将 (a) $\mathcal{P}_i := \mathcal{P}(\emptyset)$、(b) $m_i := 0$ 和 (c) 树$_i(\mathcal{P}_i, \emptyset)$ 设为平凡有根树同构类型。

(2) 若 $h_i \ge 1$，令 $\overline{f}^{-1}(\{i\}) = \{j_1, j_2, \ldots, j_K\}$，并对$t = 1,2, \ldots, K$设定 $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$。然后定义

(a) $\mathcal{P}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$，

(b) $m_i := \sum_{t=1}^{K} (m_{j_t} + 1)$, *and*

(c) for $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$, *viewed as the concatenation of the logical sign tuples* $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$ *for* $t = 1, 2, \ldots, K$,

$$\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) := \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^{+}).$$

采用这一 $\mathcal{P}_i$ 选择后，我们有

$$\mathrm{AC}(\mathcal{P}_i) \le \mathfrak{v}_i := |\,\mathrm{V}(\mathrm{Tree}_{\Gamma_{\overline{f}}}(i))| - 1 \le d - 1 \in O(d) \subseteq O(d^2 \,\mathrm{mpe}(q-1)).$$

第二步是描述 $\mathrm{Tree}^{\Gamma_f}(0_{\mathbb{F}_q})$ 的同构类型，这与上文对暂态陪集的递归步骤类似。令 $\overline{f}^{-1}(\{d\}) = \{d, j_1, j_2, \ldots, j_K\}$（我们注意到，除非分圆形式 (1) 中某个系数 a_i 为 0，否则 $K = 0$ ）。$\mathrm{Tree}^{\Gamma_f}(0)^*$ 中 0 的子节点恰好是 Γ_f^* 中 0 的非零子节点，而每个这样的子节点都必须属于某个 $t \in \{1, 2, \ldots, K\}$ 中的 C_{j_t}。我们观察到，每个 C_{j_t} 都是一个暂态陪集，因此根据命题 3.3.1，我们已经知道 C_{j_t} 的一个合适的算术划分 $\mathcal{P}_{j_t}$，它具有一个长度为 m_{j_t} 的显式张成序列，并且我们已经明确了解 $\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$ 的有根树同构类型 $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$。此外，$C_{j_t}$ 中的所有顶点在 f 下都映射到 $0_{\mathbb{F}_q}$，因此

$$f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}) = \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}) = \mathbf{0}^{-1}(\{0_{\mathbb{Z}/s\mathbb{Z}}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$$

对所有 $\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$，其中 $\mathbf{0}: \mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}, z \mapsto 0 = 0z + 0$。因此，若令 $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, \mathbf{0})$，它按引理 2.2.2 中的标准形式由单个 s-同余式 $x \equiv 0 \pmod{s}$ 重复 $m_{j_t} + 1$ 次张成，则

$$|f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})| = \sigma_{\mathcal{P}_{j_t}, \mathbf{0}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, (\emptyset, \ldots, \emptyset)).$$

由于 $\mathrm{Tree}^{\Gamma_f}(0_{\mathbb{F}_q})$ 是通过对每个 $t = 1, 2, \ldots, K$ 以及每个 $\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}}$，将 $|f^{-1}(\{0_{\mathbb{F}_q}\}) \cap \mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})|$ 个 $\mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ 副本连接到同一个根而得到的，因此我们得到以下命题。

命题 3.3.2. 设 $\overline{f}^{-1}(\{d\}) = \{d, j_1, j_2, \ldots, j_K\}$。对于 $t = 1, 2, \ldots, K$，令 $\mathcal{P}_{j_t}$，m_{j_t} j_t … 和 $\mathrm{Tree}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$（$t = 1, 2, , K$）如命题 3.3.1 中所述。此外，我们用 0 表示常值零函数 $\mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$。则

$$\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q}) = \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}}} (\sigma_{\mathcal{P}_{j_t}, \mathbf{0}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, (\emptyset, \ldots, \emptyset)) \cdot \mathrm{Tree}_{j_t}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})^{+}).$$

在第三步也是最后一步中，我们考虑来自周期陪集 C_i（其中 $i < d$）的顶点 x。记 $\overline{f}^{-1}(\{i\}) = \{i', j_1, j_2, \ldots, j_K\}$，其中 i' 是 $\overline{f}$ 下 i 的唯一 $\overline{f}$-周期原像。因此，对于 $t = 1$、2、……、K，C_{j_t} 是一个暂态陪集；令 $\mathcal{P}_{j_t}$ 为命题 3.3.1 中定义的 C_{j_t} 的算术划分。此外，令

$(i_0,\ i_1,\ \ldots\ldots,\ i_{\ell-1})$ 为 $\overline{f}$ 下 $i = i_0$ 的循环，并定义 $i_k := i_{k \bmod \ell}$，其中 $k \in \mathbb{Z}$（特别地，$i' = i_{-1}$）。此外，令 $A_{i_k}:\ \mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z},\ \ x \mapsto \alpha_{i_k} x + \beta_{i_k}$，为描述 f 如何从 C_{i_k} 映射到 $C_{i_{k+1}}$ 的仿射映射。

这个情形更加复杂，我们需要根据另一个参数进行递归。与第3.2小节一样，我们将 Γ_f 在所有周期块 C_i的并集上诱导的子图记为 Γ_{per}，即 f_{per}的函数图。提醒读者，我们借助定理 3.2.1明确了解了 Γ_{per} 中周期顶点上的树。其想法是对一个称为 $\mathfrak{h}(x)$的参数进行递归，该参数的取值范围从0 到 Γ_{per} 中有根树的最大高度 H_i；这些有根树位于陪集 $C_{i_t} = C_{\overline{f}^t(i)}$ 中的周期顶点上，其中 $t = 0$ 取 $1, \ldots, \ell - 1$。该参数定义如下：

$$\mathfrak{h}(x) = \begin{cases} \mathrm{ht}(\mathrm{Tree}_{\Gamma_{\mathrm{per}}}(x)) \in \{0, 1, \ldots, H_i - 1\}, & \text{if } x \text{ is } f\text{-transient}, \\ H_i, & \text{if } x \text{ is } f\text{-periodic}. \end{cases} \tag{14}$$

根据例 3.2.2，当 $\mathfrak{h}(x) = H_i$ 时（即，当 x 是 f-周期的），$\mathfrak{h}(x)$ 与 $\mathrm{ht}(\text{树}_{\Gamma_{\mathrm{per}}}(x))$ 之间通常不存在任何关系。此外，$\mathfrak{h}$ 不必在给定陪集 C_{i_t} 上取遍 $\{0, 1, \ldots, H_i\}$ 中的所有值；事实上，$\mathfrak{h}(C_{i_t})$ 甚至不必是整数区间。不过，这些都不会影响我们的方法。

我们注意到，$H_i + 1$ 是最小的正整数 k，使得对于所有 $t \in \mathbb{Z}$，共同的繁衍数

$$\mathrm{proc}_k^{(\Gamma^*_{\mathrm{per}})}(x) = \frac{\gcd(\prod_{j=0}^{k-1} \alpha_{i_{t-k+j}}, s)}{\gcd(\prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}, s)}$$

所有 f-周期顶点$x \in C_{i_t}$ 的共同繁衍数等于 1；对于每个给定的 t，这等价于

$$\gcd(\prod_{j=0}^{k-1} \alpha_{i_{t-k+j}}, s) = \gcd(\prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}, s),$$

并且进一步等价于

$$\prod_{p \mid \gcd(\alpha_{i_{t-1}}, s)} p^{\nu_p(s)} \mid \prod_{j=0}^{k-2} \alpha_{i_{t-k+j}}.$$

令$\overline{\alpha}_i := \prod_{t=0}^{\ell-1} \alpha_{i_t}$（即第 3.1 小节记号中的 $\mathcal{A}_i$ 的线性系数），由此不难看出

$$H_i \le \ell \cdot \max_{p \mid \gcd(\overline{\alpha}_i, s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i)} \right\rceil \le \ell \,\mathrm{mpe}(s) \le d \,\mathrm{mpe}(q-1). \tag{15}$$

令 $t = 1$=2, $\ldots, K$ 时，设 $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$，并设 $\mathcal{R}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$。我们用 n_i 表示所用的 $\mathcal{R}_i$ 的生成同余序列的长度（通常为 $n_i = \sum_{t=1}^{K}(m_{j_t} + 1)$，但在具体例子中，这些同余式之间可能会有重复，因此可以删除其中一些）。一个简单的观察是，就暂态陪集对树 $\Gamma_f(x)$ 的贡献树 $\Gamma_f(x, \bigcup_{t=1}^{K} C_{j_t})$ 而言，一切都与步骤 1 相同。

命题 3.3.3。设 $i \in \{0,1,\ldots,d-1\}$ 是 $\overline{f}$-周期的，并设 $j_1, j_2, \ldots, j_K$ 是 i 在 $\overline{f}$ 下的 $\overline{f}$- 暂态原像。此外，设 $\mathcal{P}'_{j_t} := \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$，其中 $t = 1=2, \ldots, K$，且 $\mathcal{R}_i := \bigwedge_{t=1}^{K} \mathcal{P}'_{j_t}$。则有以下结论。

(1)对于 $x \in C_i$ 和 $t \in \{1,2,\ldots,K\}$，同构类型树$\Gamma_f(x, C_{j_t})$ 仅取决于 $\mathcal{P}'_{j_t}$-块 $\mathcal{B}(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$ (对于某个 $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$)，其中$x$ 位于其中。该同构类型记为 树$_i(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})})$，并可根据公式计算

$$\mathrm{Tree}_i\left(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right) = \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right) \mathrm{Tree}_{j_t}\left(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})}\right)^{+}.$$

（2）对于 $x \in C_i$，$\mathrm{Tree}^{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$ 的有根树同构类型仅取决于 x 所在的 $\mathcal{R}_i$-块 $\mathcal{B}\left(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)}\right)$（对于某个 $\vec{\nu}^{(\mathcal{R}_i)} \in \{\emptyset,\neg\}^{n_i}$）。该同构类型记作 $\mathrm{Tree}_i(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)})$，并可根据公式计算。

$$\mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)}\right) = \sum_{t=1}^{K} \mathrm{Tree}_i\left(\mathcal{P}'_{j_t}, C_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right)^{+}$$

其中，对于 $t = 1$、2、.....、K，$\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset,\neg\}^{m_{j_t}+1}$ 是唯一满足 $\mathcal{B}\left(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)}\right) \subseteq \mathcal{B}\left(\mathcal{P}'_{j_t}, \vec{\nu}^{(\mathcal{P}'_{j_t})}\right)$ 的逻辑符号元组；在标准情形下，其中 $n_i = \sum_{t=1}^{K}(m_{j_t}+1)$，元组 $\vec{\nu}^{(\mathcal{R}_i)}$ 只是 $t = 1$、2、.....、K 对应的 $\vec{\nu}^{(\mathcal{P}'_{j_t})}$ 的串接。

我们提醒读者，我们希望对参数$\mathfrak{h}(x)$进行递归，该参数在（14）中定义。这是由命题 3.3.3所启发的，因为如果 $\mathfrak{h}(x) = 0$，那么 x 在 $C_{i'}$中没有暂态子节点，因此树$^{\Gamma_f}(x) =$ 树$^{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$。一般来说，我们希望构造一个算术划分 $\mathcal{S}_{i,h}$，使得对于顶点$x \in C_i$，若 $\mathfrak{h}(x) = h$，则树$^{\Gamma_f}(x, C_{i'})$的同构类型只取决于包含 x 的$\mathcal{S}_{i,h}$-块，并且可以明确地理解。这样我们基本上就完成了，因为树$^{\Gamma_f}(x) =$ 树$^{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$ + 树$^{\Gamma_f}(x, C_{i'})$。

为了构造$\mathcal{S}_{i,\ h}$ 并证明其具有所需的性质，我们需要引入相当多的记号。

- 对于 $h \in \{0,1,\ldots,H_i\}$，顶点 $x \in C_i = C_{i_0}$ 当且仅当 x属于 $A_{i_{-h}} A_{i_{-h+1}} \cdots A_{i_{-1}}$的像，即仿射映射 $\mathcal{A}_{i,h} : z \mapsto \overline{\alpha}_{i,h} z + \overline{\beta}_{i,h}$ 的 $\mathbb{Z}/s\mathbb{Z}$ 中时，至少有 h 个后继代 Γ^*_{per}，其中

$$\overline{\alpha}_{i,h} = \prod_{t=1}^{h} \alpha_{i_{-t}} \text{ and } \overline{\beta}_{i,h} = \sum_{t=1}^{h} \beta_{i_{-t}} \prod_{k=1}^{t-1} \alpha_{i_{-k}}.$$

也就是说，x 在 Γ^*_{per} 中至少有 h 个后继代，当且仅当它满足以下 s-同余式，我们将其记为$\theta_{i,\ h}(x)$:

$$x \equiv \overline{\beta}_{i,h} \pmod{\gcd\left(\overline{\alpha}_{i,h}, s\right)}.$$

我们注意到，$\theta_{i,0}(x)$ 中的模数为 1，因此该同余式是平凡的。

• 接下来，对于每个 $h \in \{0, 1, \ldots, H_i\}$，我们描述一个简单系统 $\Theta_{i,h}(x)$，它至多包含两个 s-CC，使得对于所有 $x \in C_i$，等式 $\mathfrak{h}(x) = h$ 成立，当且仅当 $\Theta_{i,h}(x)$ 成立。若 $h < H_i$，则满足 $\mathfrak{h}(x) = h$ 的顶点 $x \in C_i$ 恰好是那些在 Γ^* 中具有恰好 h 个后继代的 f-暂态 $x \in C_i$。由此可知，对于这样的 h，$\mathfrak{h}(x) = h$ 成立，当且仅当 $\theta_{i,h}(x)$ 和 $\neg\theta_{i,h+1}(x)$ 均成立。现在假设 $h = H_i$。根据 $\mathfrak{h}$ 的定义，这当且仅当 x 是 f-周期的；而这（根据 H_i 的定义）等价于 x 在 Γ^* 中至少有 H_i 个后继代。因此，在这种情况下，仅条件 $\theta_{i,H_i}(x)$ 就能给出所需的刻画。

现在，再次注意到 $\theta_{i,0}(x)$ 是平凡的，因此可以从任何包含它的条件系统中省略，我们可以如下定义 $\Theta_{i,h}(x)$：

$$\Theta_{i,h}(x) := \begin{cases} \emptyset, & \text{if } H_i = (h =) 0, \\ \neg\theta_{i,1}(x), & \text{if } h = 0 < H_i, \\ \theta_{i,h}(x) \wedge (\neg\theta_{i,h+1}(x)), & \text{if } 0 < h < H_i, \\ \theta_{i,H_i}(x), & \text{if } h = H_i > 0. \end{cases}$$

如果 $\Theta(x)$ 是一个 m-CC 系统，那么 $\mathfrak{P}(\Theta(x))$表示由 $\Theta(x)$ 中条件的非否定版本所张成的 $\mathbb{Z}/m\mathbb{Z}$算术划分。例如，

$$\mathfrak{P}\begin{pmatrix} x \equiv 4 \pmod 6 \\ x \not\equiv 3 \pmod 9 \\ x \equiv 0 \pmod 2 \end{pmatrix} = \mathfrak{P}\begin{pmatrix} x \equiv 4 \pmod 6 \\ x \equiv 3 \pmod 9 \\ x \equiv 0 \pmod 2 \end{pmatrix}.$$

我们注意到，如果 $x \in \mathbb{Z}/m\mathbb{Z}$ 选自 $\mathfrak{P}(\Theta(x))$的一个固定块，那么 $\Theta(x)$中的每个条件的真值都与 x无关，因此 $\Theta(x)$本身的真值也与 x无关。我们还注意到，对每个 $h \in \{0, 1, \ldots, H_i\}$，上一条中定义的 s-CC 系统$\Theta_{i,k}(x)$都满足以下等式：

$$\bigwedge_{k=0}^{h} \mathfrak{P}(\Theta_{i,k}(x)) = \mathfrak{P}(\theta_{i,j}(x) : j = 1, 2, \ldots, \min\{h+1, H_i\}). \tag{16}$$

我们令 $\mathcal{U}_i := \mathfrak{P}(\theta_{i,k}(x) : k = 1, 2, \ldots, H_i)$.

• 对于 $k \in \{0, 1, \ldots, H_i\}$，我们用 $\vec{\xi}_{i,k}$ 表示逻辑符号元组（ν_1，ν_2，…，ν_{H_i}），当且仅当 $t \le k$时，$\nu_t = \emptyset$。根据此定义，若将 $\mathcal{U}_i$ 视为 C_i的一个算术划分，则块 $\mathcal{B}$（$\mathcal{U}_i$，$\vec{\xi}_{i,k}$）恰好由满足 $\mathfrak{h}(x) = k$ 的那些$x \in C_i$ 组成（并且 $\mathcal{U}_i$ 的每个块都具有这种形式，其中某个 k）。

• 设 $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j \text{(模 } \mathfrak{a}_j) : j = 1, 2, \ldots, K)$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的一个算术划分，并设 $A : z \mapsto az + b$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的一个仿射映射。根据 $\mathcal{P}$（实际上，根据 $\mathcal{P}$ 的生成同余式固定序列，而非 $\mathcal{P}$ 本身）和 A，我们定义 $\mathbb{Z}/m\mathbb{Z}$ 的另一个算术划分 $\lambda(\mathcal{P}, A)$ 如下：$\lambda(\mathcal{P}, A) := \mathfrak{P}(x \equiv a\mathfrak{b}_j + b \text{(模 } \gcd(a\mathfrak{a}_j, m)) : j = 1, 2, \ldots, K)$。以下列出关于此记号的重要事实（其中我们始终有 $m = s$）：

– 若 $\mathcal{P}$ 是 $C_{i'}$ 的一个算术划分，则

$$\mathfrak{P}'(\mathcal{P}, A_{i'}) = \lambda(\mathcal{P}, A_{i'}) \wedge \mathfrak{P}(\theta_{i,1}(x)).$$

– 对于 $h = 0,1,\ldots,H_{i'} = H_i$，我们有

$$\lambda(\mathfrak{P}(\theta_{i',h}(x)), A_{i'}) = \begin{cases} \mathfrak{P}(\theta_{i,h+1}(x)), & \text{if } h < H_i, \\ \mathfrak{P}(\theta_{i,H_i}(x)), & \text{if } h = H_i. \end{cases}$$

事实上，对于 $h < H_i$，根据 $\theta_{i,h}(x)$ 的定义，以及以下事实：对于所有 a，$a' \in \mathbb{Z}$，$A_{i'}(\overline{\beta}_{i',h}) = \overline{\beta}_{i,h+1}$ 且最大公约数$(a$ 最大公约数(a', s) $s) =$最大公约数(aa', s)，结论显然成立。此外，注意到我们的 $\theta_{i,h}(x)$ 定义在 $h > H_i$ 时也有意义，因此有 $\lambda(\mathfrak{P}(\theta_{i',H_{i'}}(x))) = \mathfrak{P}(\theta_{i,H_i+1}(x))$。然而，对于 $x \in C_i$，条件 $\theta_{i,H_i+1}(x)$ 成立，当且仅当 x 在 Γ^*_{per} 中至少有 $H_i + 1$ 个后继代；这又当且仅当 x 是周期的，即当且仅当 $\theta_{i,H_i}(x)$ 成立。因此，同余式 $\theta_{i,H_i+1}(x)$ 和 $\theta_{i,H_i}(x)$ 在 $\mathbb{Z}/s\mathbb{Z}$ 中有相同的解集，由此 $\mathfrak{P}(\theta_{i,H_i+1}(x)) = \mathfrak{P}(\theta_{i,H_i}(x))$。以上关于 $\lambda(\mathfrak{P}(\theta_{i',h}(x))\ A_{i'})$ 的公式证明完毕。

–根据前面两个项目以及等式 (16)，用 i' 替代 i 应用后，我们得到

$$\mathfrak{P}'(\bigwedge_{k=0}^{h} \mathfrak{P}(\Theta_{i',k}(x)), A_{i'}) = \bigwedge_{k=0}^{\min\{h+1,H_i\}} \Theta_{i,k}(x),$$

并且，尤其是 $\mathfrak{P}'(\mathcal{U}_{i'}, A_{i'}) = \mathcal{U}_i$。

• 令 $\mathcal{P}$ 为 C_i的一个算术划分。我们递归地定义记号 $\lambda_i^t(\mathcal{P})$，其中 t 为非负整数，如下所示：$\lambda_i^0(\mathcal{P}) := \mathcal{P}$，对于 $t \geq 1$，我们设$\lambda_i^t(\mathcal{P}) := \lambda(\lambda_i^{t-1}(\mathcal{P})\ A_{t-1})$。换言之，$\lambda_i^t(\mathcal{P})$是通过运算 λ，沿着 i 的 $\overline{f}$-循环将 $\mathcal{P}$ 向前推进 t 次，并在每一步使用相应的仿射函数 A_{i_k}，从而得到的 C_{i_t} 的算术划分。

对于 $h \in \{0, 1, \ldots, H_i\}$，我们引入以下 C_i 的算术划分：

- $\mathcal{S}_{i,h} := \bigwedge_{t=1}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}})$;
- $\mathcal{P}_{i,h} := \mathcal{R}_i \wedge \mathcal{S}_{i,h} = \bigwedge_{t=0}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}})$;
- $\mathcal{T}_{i,h} := \mathcal{S}_{i,h} \wedge \mathcal{U}_i = \bigwedge_{t=1}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}}) \wedge \mathcal{U}_i$;
- $\mathcal{Q}_{i,h} := \mathcal{R}_i \wedge \mathcal{T}_{i,h} = \bigwedge_{t=0}^{h} \lambda_{i_{-t}}^t(\mathcal{R}_{i_{-t}}) \wedge \mathcal{U}_i$.

考虑$\mathcal{S}_{i,h}$ 和 $\mathcal{P}_{i,h}$ 的动机在于：对于其中包含的 $\mathfrak{h}$-值 h 的顶点 $x \in C_i$，它们的块分别决定了 Tree$^{\Gamma_f}(x, C_{i'})$ 和 Tree$^{\Gamma_f}(x)$ 的有根树同构类型——见下文命题 3.3.4。命题 3.3.4 还给出了用包含 x 的 $\mathcal{S}_{i,h}$- 块表示 Tree$^{\Gamma_f}(x, C_{i'})$ 的显式公式；该公式涉及划分 $\mathcal{Q}_{i',k}$ for$0 \leq k < h$的分布数（如引理 2.2.2 中的分布数——见该引理之后的句子）。命题 3.3.4 的陈述中没有提及 $\mathcal{T}_{i,h}$ 形式的划分，但由于对于 $h \geq 1$，有 $\mathcal{T}_{i,h} = \mathfrak{P}'(\mathcal{Q}_{i',h-1}, A_{i'})$，它们在该命题的证明中发挥着重要作用。

• 我们将逻辑符号元组 $\vec{\nu}$ 和 $\vec{\nu'}$ 的连接记作 $\vec{\nu} \diamond \vec{\nu'}$。

现在我们可以详细阐述块 B of $\mathcal{S}_{i,h}$ 如何影响顶点 $x \in B$ with $\mathfrak{h}(x) = h$上方的有根树。

命题 3.3.4. 设 $h \in \{0, 1, \ldots, H_i\}$，且 $x \in C_i$ with $\mathfrak{h}(x) = h$。则有以下结论。

(1) 同构类型 $\mathsf{Tree}^{\Gamma_f}(x, C_{i'})$ 仅取决于 $\mathcal{S}_{i,h}$ 中包含 x 的块 $\mathcal{B}(\mathcal{S}_{i,h}, \vec{\nu}^{(\mathcal{S}_{i,h})})$，记作 $\mathsf{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$。(2) 同构类型 $\mathsf{Tree}^{\Gamma_f}(x)$ 仅取决于包含 x 的 $\mathcal{P}_{i,h}$-块 $\mathcal{B}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$，记作 $\mathsf{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$。

更具体地，对于 $h = 0$，其中 $\mathcal{S}_{i,h} = \mathcal{P}(\emptyset)$ 且 $\mathcal{P}_{i,h} = \mathcal{R}_i$，有根树 $\mathsf{树}_i^{(0)}(\mathcal{S}_{i,0}, C_{i'}, \emptyset)$ 是平凡的，且

$$\mathrm{Tree}_i^{(0)}(\mathcal{P}_{i,0}, \vec{\nu}^{(\mathcal{P}_{i,0})}) = \mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{P}_{i,0})}\right).$$

对于 $h \geq 1$，令 $\vec{\nu}^{(\mathcal{S}_{i,h})} = \diamond_{j=1}^{h} \vec{o_t'}$，其中 $\vec{o_t'} \in \{\emptyset, \neg\}^{n_{i-t}}$，我们有如下结论，其中$\vec{o_t}$对于 $t = 0,\ 1, \ldots, h-1$ 是取遍 $\{\emptyset, \neg\}^{n_{i-t-1}}$的变量:

$$\mathrm{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})}) = \\ \sum_{k=0}^{h-1} \sum_{\vec{o0}, \ldots, \vec{ok}} \sigma_{\mathcal{Q}_{i',k}, A_{i'}}(\diamond_{t=0}^{k} \vec{o_t} \diamond \vec{\xi}_{i',k}, \diamond_{t=1}^{k+1} \vec{o_t'} \diamond \vec{\xi_{i,h}}) \, \mathrm{Tree}_{i'}^{(k)}(\mathcal{P}_{i',k}, \diamond_{t=0}^{k} \vec{o_t})^{+},$$

并且，将 $\vec{\nu}^{(\mathcal{P}_{i,h})}$ 视为 $\vec{o_0'} \in \{\emptyset, \neg\}^{n_i}$ 与 $\vec{\nu}^{(\mathcal{S}_{i,h})}$的串接，我们有

$$\mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})}) = \mathrm{Tree}_i\left(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{o_0'}\right) + \mathrm{Tree}_i^{(h)}\left(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})}\right).$$

证明。$h = 0$ 的公式显然，因为满足 $\mathfrak{h}(x) = 0$ 的顶点 $x \in C_i$ 在 $C_{i'}$ 中没有 f-暂态子节点。因此，我们可以假设 $h \geq 1$。关于$\mathsf{树}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$，我们注意到，若 $x \in C_i$ 位于 $\mathcal{S}_i$ 的块 $\mathcal{B}(\mathcal{S}_{i,h}, \vec{\nu}^{(\mathcal{S}_{i,h})})$ 中，且满足 $\mathfrak{h}(x) = h$，则对每个 $k \in \{0, 1, \ldots, h-1\}$，都有 x 位于 $\mathcal{T}_i$ 的块 $\mathcal{B}(\mathcal{T}_{i,k+1}, \diamond_{j=1}^{k+1} \vec{o_j'} \diamond \vec{\xi}_{i,h})$ 中。根据定义，$\mathcal{T}_{i,k+1} = \mathcal{P}'(\mathcal{Q}_{i',k}, A_{i'})$。$x$ 在 $C_{i'}$ 中满足 $\mathfrak{h}(y) = k$ 的每个（f-暂态）原像 y，都位于 $\mathcal{Q}_{i',k}$ 的某个形如 $\mathcal{B}(\mathcal{Q}_{i',k}, \diamond_{j=0}^{k} \vec{o_j} \diamond \vec{\xi}_{i',k})$ 的块中，其中 $\vec{o_j} \in \{\emptyset, \neg\}^{n_{i-j-1}}$ 对于 $j = 0,\ 1,\ \ldots\ldots,\ k$。此外，根据引理 2.2.2，该块中此类原像的数量为

$$\sigma_{\mathcal{Q}_{i',k}, A_{i'}}(\diamond_{j=0}^{k} \vec{o_j} \diamond \vec{\xi}_{i',k}, \diamond_{j=1}^{k+1} \vec{o_j'} \diamond \vec{\xi}_{i,h}).$$

我们进一步观察到，在 $\mathcal{B}(\mathcal{Q}_{i',k}, \diamond_{j=0}^{k} \vec{o_j} \diamond \vec{\xi}_{i',k})$ 中的 $C_{i'}$ 的 x 的每个原像 y，也包含在 $\mathcal{B}(\mathcal{P}_{i',k}, \diamond_{j=0}^{k} \vec{o_j})$ 中，因此 $\mathsf{Tree}^{\Gamma_f}(y) \cong \mathsf{Tree}_{i'}^{(k)}(\mathcal{P}_{i',k}, \diamond_{j=0}^{k} \vec{o_j})$。至此，$\mathsf{Tree}_i^{(h)}(\mathcal{S}_{i,h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$ 的公式得证。

树$_i^{(h)}$ $(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$ 的公式很清楚，因为

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t}) + \mathrm{Tree}_{\Gamma_f}(x, C_{i'})$$

对所有 $x \in C_i$。 □

现在，令 $\mathcal{P}_i := \mathcal{Q}_{i,\ H_i} = \mathcal{P}_{i,\ H_i} \wedge \mathcal{U}_i$。综上，我们得到本小节的以下结论。

命题 3.3.5. 设 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 是 $\mathcal{P}_i$ 的一个块。我们可以将 $\vec{\nu}^{(\mathcal{P}_i)}$ 视为串接 $\diamond_{t=0}^{H_i} \vec{o_t'} \diamond \vec{\xi}$，其中 $\vec{o_t'} \in \{\emptyset, \neg\}^{n_{i-t}}$对 $t = 0, 1, \ldots, H_i$ 成立，且 $\vec{\xi} = \vec{\xi}_{i,\ h}$对应唯一的 $h \in \{0, 1, \ldots, H_i\}$。那么对于 $x \in \mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$，$\mathsf{Tree}^{\Gamma_f}(x)$ 的同构类型不依赖于 x，记作 $\mathsf{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$，并由公式给出

$$\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) = \mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h}, \diamond_{t=0}^{h} \vec{o_t'}).$$

此外，我们有

$$\mathrm{AC}(\mathcal{P}_i) \le d^2 \,\mathrm{mpe}(q-1) + d - 1 \in O(d^2 \,\mathrm{mpe}(q-1)).$$

证明。块 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 包含于 $\mathcal{B}(\mathcal{U}_i, \xi_{i,h})$中，因此 $\mathfrak{h}(v) = h$。此外，$\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 包含于 $\mathcal{B}(\mathcal{P}_{i,h}, \diamond_{t=0}^{h} \vec{o_t'})$ 中，所以根据命题 3.3.4(2)，关于树 $_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 的断言公式显然成立。对于复杂度界，我们注意到，根据 $\mathcal{P}_i = \mathcal{P}_{i,H_i} \wedge \mathcal{U}_i$ 的定义，有

$$\begin{aligned}\mathrm{AC}(\mathcal{P}_i) &\le \mathrm{AC}(\mathcal{P}_{i,H_i}) + \mathrm{AC}(\mathcal{U}_i) \le \sum_{t=0}^{H_i} \mathrm{AC}(\lambda_{i-t}^{t}(\mathcal{R}_{i-t})) + H_i \le (H_i+1)(d-1) + H_i \\ &= H_i d + d - 1 \le d^2 \,\mathrm{mpe}(q-1) + d - 1,\end{aligned}$$

按要求。 □

注释 3.3.6。省略本小节中详细推导的所有细节，我们基本上证明了，在每个陪集 C_i 上，都存在一种算术划分 $\mathcal{P}_i$，其算术复杂度 $\mathsf{AC}(\mathcal{P}_i) \in O(d^2 \,\mathsf{mpe}(q-1)) \subseteq O(d^2 \log q)$，能够“控制” C_i 中顶点上方的有根树。现在，由单元素块组成的 $C_i \cong \mathbb{Z}/s\mathbb{Z}$ 的平凡划分 $\mathcal{T}_s$，出于显而易见的理由，也能“控制”其各块上方的树。尽管为了实现我们的目的，尽可能将多个同构的有根树归入同一个块更为可取（因此一般来说 $\mathcal{P}_i$ 优于 $\mathcal{T}_s$），但一个有趣的问题是，就算术复杂度而言，$\mathcal{T}_s$ 是否至少能够“胜过” $\mathcal{P}_i$。

我们来讨论一般模数 $m \in \mathbb{N}^+$ 的情形（而不仅仅是 $s = (q-1)/d$）。我们考虑将 m 因式分解为两两互素的素数幂所得的分解 $m = p_1^{v_1} p_2^{v_2} \cdots p_K^{v_K}$。在由 $x \equiv b \left(\text{模 } p_j^{v_j}\right)$（其中 $j = 1$、2、.....、K）和 $b \in \{0, 1, \ldots, p_j^{v_j} - 2\}$ 构成的系统中，给 m-CC 添加逻辑符号，即可将 $\mathbb{Z}/m\mathbb{Z}$ 的每个单元素子集作为相应算术划分中的一个块。这

表明 AC($\mathcal{T}_m$) $\le p_1^{v_1}+\cdots+p_K^{v_K}-K$。当然，如果 m 是素数幂，那么这个界就是 $m-1$。另一方面，如果 $m=p_1p_2\cdots p_K=p_K\#$ 是前 primorial 数，那么由于 $p_K\# = \exp((1+o(1))K\log K)$，因此 $\log m\sim K\log K$；也就是说，回想备注 1.6 中 W 表示朗伯 W 函数，我们有

$$K\sim\frac{\log m}{W(\log m)}\sim\frac{\log m}{\log\log m},$$

而我们的界表明，AC($\mathcal{T}_m$) 至多为

$$\begin{aligned}p_1+\cdots+p_K-K\sim p_1+\cdots+p_K&\sim\frac{1}{2}n^2\log n\\&\sim\frac{1}{2}\frac{\log^2 m}{\log\log^2 m}(\log\log m-\log\log\log m)\sim\frac{1}{2}\frac{\log^2 m}{\log\log m},\end{aligned}$$

其不会超过固定 d 下的 $O(d^2\log m)$，更不用说 $O(d^2\,\mathrm{mpe}(m))=O(d^2)$了，并注意到 $\mathrm{mpe}(m)=\mathrm{mpe}(p_K\#)=1$。一个有趣的未解决问题是是否

$$\liminf_{m\to\infty}\frac{\mathrm{AC}(\mathcal{T}_m)}{\log^2(m)/\log\log m}>0,$$

另请参阅问题 6.3.1。

3.4 理解连通分量

现在，我们希望结合迄今为止建立的理论，全面理解 Γ_f 的连通分量。回顾引言中的内容，我们将每个 Γ_f的连通分量与一个由有根树同构类型组成的项链相关联，这一方法刻画了该分量的有向图同构类型。

设 $\mathcal{L}$ 是 f 的一个 CRL 列表（关于如何构造 $\mathcal{L}$，请参见第 3.1 小节）。提醒读者，$\mathcal{L}$ 中各对的第一项不仅是 f 中各循环的代表元，也是 Γ_f 中各连通分支的代表元。固定 $(r,l)\in\mathcal{L}$。我们旨在简洁地描述与包含 r 的 Γ_f 连通分支相关联的有根树同构类型的循环序列。

我们注意到，根据我们对 $\mathcal{L}$ 的构造，如果$r=0_{\mathbb{F}_q}$，那么 $l=1$，且该连通分支由附着在循环顶点 $0_{\mathbb{F}_q}$上的单个有根树组成。我们可以按照第3.3 小节中的描述确定该树，树$\Gamma_f(0_{\mathbb{F}_q})$。长度为 1的循环序列 [树$\Gamma_f(0_{\mathbb{F}_q})$] 确定了 $0_{\mathbb{F}_q}$ 的整个连通分支，因此我们今后可以假设$r\neq 0_{\mathbb{F}_q}$，包含在 C 的唯一陪集 C_i 中，该陪集位于$\mathbb{F}_q^*$中。

我们回顾一下，i必然是 $\overline{f}$的一个周期点，并令（i_0，i_1，…，$i_{\ell-1}$）且$i_0=i$为其循环。对于一般的 $t\in\mathbb{Z}$，我们定义 $i_t:=i_{t \bmod \ell}$。根据第 3.3 小节，在 $\mathbb{F}_q^*$中的 C 的每个陪集 C_j 上，都有一个算术划分 $\mathcal{P}_j$，使得树Γ_f（x）的同构类型保持不变；对于从 $\mathcal{P}_j$的同一块 $\mathcal{B}$（$\mathcal{P}_j$，$\vec{\nu}$）中选取的顶点 $x\in C_j$，该同构类型可被明确理解，我们将上述同构类型记为树j（$\mathcal{P}_j$，$\vec{\nu}$）。

让我们固定 $t\in\{0,1,\ldots,\ell-1\}$ 并回顾（来自上一小节）的记号 $\mathcal{A}_{i_t}=A_{i_t}A_{i_{t+1}}\cdots A_{i_{t+\ell-1}}$，它表示沿循环的所有仿射映射的乘积的

i_t。此外，我们回顾一下，f^ℓ稳定保持C_{i_t}，并且限制（f^ℓ）$^{|C_{i_t}}$对应于 $\mathbb{Z}/s\mathbb{Z}$的仿射映射$\mathcal{A}_{i_t}$。设 $r'_{i_t} := f^t(r)$。我们有 $\ell \mid l$，并且 r'_{i_t} 在$\mathcal{A}_{i_t}$ 下的循环是

$$(r'_{i_t}, \mathcal{A}_{i_t}(r'_{i_t}), \mathcal{A}_{i_t}^2(r'_{i_t}), \ldots, \mathcal{A}_{i_t}^{l/\ell-1}(r'_{i_t})) = (f^t(r), f^{t+\ell}(r), f^{t+2\ell}(r), \ldots, f^{t+((l/\ell)-1)\ell}(r)).$$

我们将定义在C_{i_t} 上的函数$\mathcal{B}_{i_t}$ 记为，它把$x \in C_{i_t}$ 映射到包含 x的$\mathcal{P}_{i_t}$ 的唯一块。我们的下一个目标是理解块序列。

$$(\mathcal{B}_{i_t}(r'_{i_t}), \mathcal{B}_{i_t}(\mathcal{A}_{i_t}(r'_{i_t})), \ldots, \mathcal{B}_{i_t}(\mathcal{A}_{i_t}^{l/\ell-1}(r'_{i_t})))$$

沿着每个 $t = 0$ 的 r'_{i_t} 的 $\mathcal{A}_{i_t}$-循环，因为这些序列组合成块序列 $(\mathcal{B}_{i_n}(f^n(r)))_{n=0,\ 1,\ \ldots,\ l-1}$，由此可以读出包含 r 的 Γ_f 的连通分支的有根树同构类型的循环序列。

现在，假设 $\mathcal{P}_{i_t} = \mathfrak{P}(x \equiv \mathfrak{b}_{i_t,j} (\text{模 } \mathfrak{a}_{i_t,j}) : j = 1, 2, \ldots, m_{i_t})$。理解块序列 $(\mathcal{B}_{i_t}(\mathcal{A}_{i_t}^m(r'_{i_t})))_{m=0,1,\ldots,l/\ell-1}$，就意味着理解同余式 $x \equiv \mathfrak{b}_{i_t,j}$ (模$\mathfrak{a}_{i_t,j}$) 的真值，其中 x 遍历 r'_{i_t} 在 $\mathcal{A}_{i_t}$ 下的循环。回顾上一小节可知，$\overline{\alpha}_{i_t} := \prod_{t=0}^{\ell-1} \alpha_{i_t}$ 是 $\mathcal{A}_{i_t}$ 的线性系数。引理 2.1.14 表明，$\mathcal{A}_{i_t}$ 的所有周期点（特别是 r'_{i_t} 在 $\mathcal{A}_{i_t}$ 下循环上的所有点）模 $\prod_p \gcd(\overline{\alpha}_{i_t,s})\, p^{\nu_p}(s)$ 都有一个特定且明确已知的值。这可能导致同余式$x \equiv \mathfrak{b}_{i_t,j}$ (模 $\mathfrak{a}_{i_t,j}$) 中有些式子对 $j \in \{1, 2, \ldots, m_{i_t}\}$ 在 $\mathcal{A}_{i_t}$ 的所有周期点上都具有恒定的真值。其余同余式可按如下方式处理。

我们计算 $^{(\mathfrak{a}_{i_t,j})}\mathcal{A}_{i_t}(r'_{i_t}, \mathfrak{b}_{i_t,j}) =: \mathfrak{l}_{i_t,j}$（关于这种离散对数记号，参见第2.4小节），以及 r'_{i_t} 在模 $\mathfrak{a}_{i_t,j}$ 意义下经过 $\mathcal{A}_{i_t}$ 作用后的周期长度 $l_{i_t,j}$。若 $\mathfrak{l}_{i_t,j} = \infty$（即 $\mathfrak{b}_{i_t,j}$ 不在 r'_{i_t} 在模 $\mathfrak{a}_{i_t,j}$ 意义下经过 $\mathcal{A}_{i_t}$ 作用所形成的循环上），则对模 s 下 r'_{i_t} 的 $\mathcal{A}_{i_t}$- 循环上的所有 x，同余式 $x \equiv \mathfrak{b}_{i_t,j}$ （$\mathrm{mod}\ \mathfrak{a}_{i_t,j}$）均不成立。否则，该同余式恰好对满足 $y \equiv \mathfrak{l}_{i_t,j}$ （$\mathrm{mod}\ l_{i_t,j}$）的那些 $x = \mathcal{A}_{i_t}^y (r'_{i_t})$ 成立。我们将那些满足以下条件的 $j \in \{1, 2, \ldots, m_{i_t}\}$ 构成的集合记为 I_{i_t}：$x \equiv \mathfrak{b}_{i_t,j}$ （$\mathrm{mod}\ \mathfrak{a}_{i_t,j}$）的真假值沿模 s 下 r'_{i_t} 的 $\mathcal{A}_{i_t}$-循环保持不变。块序列（$\mathcal{B}_{i_t}$（$\mathcal{A}_{i_t}^m$（r'_{i_t}））$)_{m=0,1,\ldots,l/\ell-1}$ 由以下内容确定

- 关于沿着$\mathcal{A}_{i_t}$-循环的 r'_{i_t} 模 s 同余式$x \equiv \mathfrak{b}_{i_t,j}$ (模 $\mathfrak{a}_{i_t,j}$)对于$j \in I_{i_t}$的恒定真值信息，以及
- 算术划分 $\mathcal{P}_{(i_t)} := \mathfrak{P}(y \equiv \mathfrak{l}_{i_t,j}$ (模$l_{i_t,j}$) : $j \notin I_{i_t}$) 的 $\mathbb{Z}/(l/\ell)\mathbb{Z}$，它编码了沿循环变化的其余同余式$x \equiv \mathfrak{b}_{i_t,j}$ (模 $\mathfrak{a}_{i_t,j}$) 的真值行为（对于 $j \notin I_{i_t}$）。

一旦对于每个 t，块序列 $(\mathcal{B}_{i_t}(\mathcal{A}_{i_t}^m(r'_{i_t})))_{m=0,1,\ldots,l/\ell-1}$ 以这种方式得到理解，编码包含 r 的 Γ_f 的连通分支的同构类型的实际项链，就由循环序列

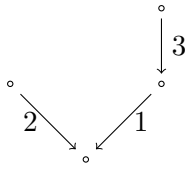
$$[\mathrm{Tree}_{i_n}(\mathcal{P}_{i_n}, \mathcal{B}_{i_n}(\mathcal{A}_{i_n}^{(n-(n \bmod \ell))/\ell}(r'_{i_n})))]_{n=0,1,\ldots,l-1} \tag{17}$$

其中，按照记号的惯用法，$\mathsf{Tree}_i(\mathcal{P}_i, B)$应理解为$B = \mathcal{B}(\mathcal{P}_i, \vec{\nu})$的$\mathsf{Tree}_i(\mathcal{P}_i, \vec{\nu})$。

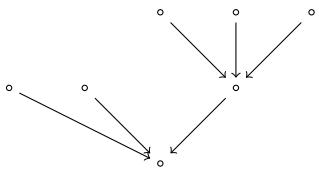
与 r 的两种不同取值相关联的 Γ_f 的连通分量同构，当且仅当对应的循环序列（17）相等。不过，我们注意到，如何高效地检验这一点似乎并不明显，正如如何高效地判断两个给定的算术划分是否相等也不甚明了——参见问题6.2.3和6.2.4。

4 计算与示例

本节的目标是通过一些具体的计算与示例/特殊情形来说明迄今为止建立的理论。我们首先介绍一个有用的记号，该记号将在第 4.1和 4.2小节中使用。我们考虑有限有向有根树（其中所有弧都朝向根），其边权为非负整数。这类图的同构是指保持弧的权重的底层非边加权有向图的同构。对于这类树的每个同构类型 $\mathfrak{I}$，我们关联一个非边加权的有限有向有根树的同构类型Expand($\mathfrak{I}$)，定义如下。若 y_1、y_2、…、y_n 是根 x 的邻接点，并且在这些邻接点上分别附有边加权有根树的（同构类型） $\mathfrak{I}_1$、$\mathfrak{I}_2$、…、$\mathfrak{I}_n$，而连接 x 与 y_j 的边权重为 $\mathfrak{w}_j \in \mathbb{N}_0$，则 Expand($\mathfrak{I}$) 递归定义为：取一个新根，并对每个 $j = 1$ = 1, 2, …, n，向其连接 $\mathfrak{w}_j$ 个 Expand($\mathfrak{I}_j$) 的副本。例如，如果 $\mathfrak{I}$ 是



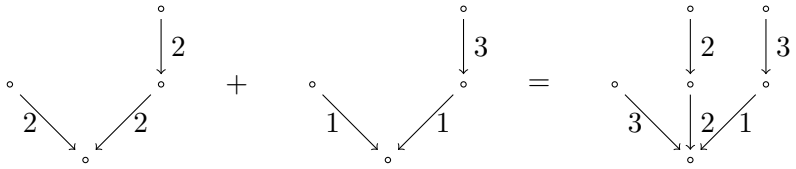
那么 Expand($\mathfrak{I}$) 为



通过对 $\mathfrak{I}$ 的高度进行递归法，我们定义 $\mathfrak{I}$ 如下进行简化。平凡同构类型 $\mathfrak{I}$ 是简化的；如果 $\mathfrak{I}$ 具有正高度，那么当所有附加到 $\mathfrak{I}$ 的根上的同构类型 $\mathfrak{I}'$（它们的高度都小于 $\mathfrak{I}$）都是简化的、两两不同的，并且其中没有任何一个通过边权为0的边附加到根上时，$\mathfrak{I}$ 就是简化的。有限边加权有向有根树的简化同构类型通过 Expand与有限有向有根树的同构类型构成双射（通过对高度进行归纳法即可轻易证明）。这使我们能够将任意有限边加权有向有根树的同构类型 $\mathfrak{I}$ 的简化形式 SF($\mathfrak{I}$) 定义为满足 Expand(SF($\mathfrak{I}$)) = Expand($\mathfrak{I}$) 的唯一简化同构类型。我们将 $\mathfrak{I} \sim \mathfrak{I}'$ 记作 SF($\mathfrak{I}$) = SF($\mathfrak{I}'$)（等价地，Expand($\mathfrak{I}$) = Expand($\mathfrak{I}'$)）。$\mathfrak{I}$ 的简化形式可以从 $\mathfrak{I}$ 通过关于树高的简单递归法显式构造（遍历附加到

$\mathfrak{I}$ 的根，计算其简化形式，并将属于同一 SF($\mathfrak{I}'$) 的边权相加。

在第 3.3 小节中，我们引入了非边加权有限有向有根树的同构类型之和（从而将它们的类构成一个类大小幺半群），并且存在唯一一种定义简化边加权有根树同构类型之和的方式，使得 Expand 成为幺半群同构（即，Expand($\mathfrak{I}_1 + \mathfrak{I}_2$) =Expand($\mathfrak{I}_1$) + Expand($\mathfrak{I}_2$)）。明确地说，$\mathfrak{I}_1 + \mathfrak{I}_2$ 可定义如下。选取一个新根 x，并考虑在 $\mathfrak{I}_1$ 或 $\mathfrak{I}_2$中连接到该根的边加权有根树 $\mathfrak{I}'$。令 $\mathfrak{w}_j$ 表示对于 $j = 1$，2，将 $\mathfrak{I}'$ 连接到 $\mathfrak{I}_j$ 中的根的弧的权重（如果不存在这样的弧，则将 $\mathfrak{w}_j$ 视为 0）。对于每个这样的 $\mathfrak{I}'$，通过一条权重为 $\mathfrak{w}_1 + \mathfrak{w}_2$的弧，将 $\mathfrak{I}'$ 的一个副本连接到 x。例如，



通过设定 $\mathfrak{I}_1 + \mathfrak{I}_2 :=$ SF($\mathfrak{I}_1$)+SF($\mathfrak{I}_2$)，这一扩展可以推广到有限边加权有向有根树的任意同构类型。为简洁起见，下文经常省略“同构类型”一词（也就是说，将有限（边加权）有向有根树与其同构类型视为相同）。

4.1 刚性繁衍下的有根树

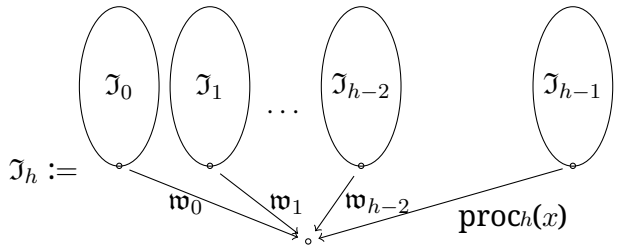
设 $\Gamma = \Gamma_g$ 是一个有限函数图，且 Γ^* 具有刚性繁衍（见定义 2.1.7(4)）。此外，设 $x \in$V(Γ) 是 g-周期的，并令 (proc$_k(x)$)$_{k\geq 1}$ 为 x 在 Γ^* 中的繁衍数序列（由于刚性繁衍，该序列与 x 无关）。命题 2.1.8 指出，有根树 Tree$_\Gamma(x)$ 的同构类型完全由这一繁衍数序列决定；该类型也不依赖于 g-周期顶点 x 的选择。我们希望明确了解如何从 (proc$_k(x)$)$_{k\geq 1}$ 推导出这一同构类型。

由于 Γ 是有限的，H 也有限：$=$ ht(Tree$_\Gamma(x)$)。我们注意到，x 在 Γ^* 中有 g-暂态子节点，其后至少有 $H-1$ 代后继，但没有后继至少有 H 代的此类子节点。这意味着，对于 $h = 1$、2、.....、H，proc$_h(x) > 1$，但 proc$_{H+1}(x) = 1$；x 在 Γ^* 中唯一的 g-周期子节点有无限多个后继代，因此对所有繁衍数的贡献均为 1。由此，我们可以仅根据序列 (proc$_k(x)$)$_{k\geq 1}$ 读出 H。

现在，引理 2.1.10 表明，对于所有 g-暂态顶点 $y \in$V(Γ)，若它们在 Γ 中上方的树高 $h \in \{0, 1, \ldots, H-1\}$ 固定，则有根树同构类型 Tree$_\Gamma(y)$ 始终相同。我们递归定义一个（未必经过简化的）边加权有向根树 $\mathfrak{I}_h = \mathfrak{I}_h$((proc$_k(x)$)$_{k\geq 1}$)，使得上述同构类型为 Expand($\mathfrak{I}_h$)。显然，$\mathfrak{I}_0$ 唯一的选择是不含弧的单个顶点。若 $h \in \{1, 2, \ldots, H-1\}$，则我们如下定义 $\mathfrak{I}_h$。我们固定一个新的根，并

- 对于 $k = 0$、1、.....、$h-2$，我们将一份 $\mathfrak{I}_k$ 连接到新根上，边权为 proc$_{k+1}$(x) −proc$_{k+2}(x)$ =: $\mathfrak{w}_k$; 且

• 我们将 $\mathfrak{I}_{h-1}$ 的一个副本附加到新的根上，边权为 $\mathrm{proc}_h(x)$。以下是该定义的示意图。



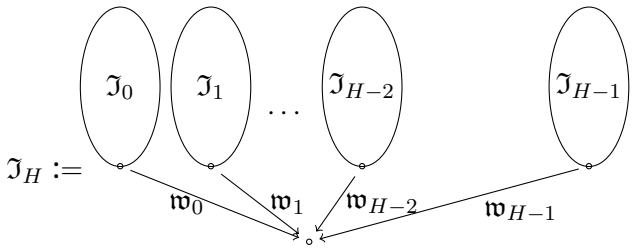
这个 $\mathfrak{I}_h$ 的定义足以解决问题，因为根据引理 2.1.10 的证明，对于每个 $k \in \{0,1,\ldots,h-1\}$，在 Γ^* 中，y 的子节点 z 里使 $\mathrm{Tree}_\Gamma(z)$ 的高度恰好为 $k-1$（因此根据归纳法与 $\mathfrak{I}_{k-1}$ 同构）的子节点数量等于

$$\mathrm{proc}_{k+1}(y) - \mathrm{proc}_{k+2}(y) = \begin{cases} \mathrm{proc}_{k+1}(x) - \mathrm{proc}_{k+2}(x) = \mathfrak{w}_k, & \text{if } k < h-1, \\ \mathrm{proc}_h(x) - 0 = \mathrm{proc}_h(x), & \text{if } k = h-1. \end{cases}$$

对于固定的周期顶点 x，$\mathrm{Tree}_\Gamma(x)$ 的确定方式类似，但必须考虑到 Γ^* 中的 x 有一个（唯一的） g-周期子节点，而该子节点不会出现在 $\mathrm{Tree}_\Gamma(x)$ 中。这意味着，将 $\mathfrak{I}_k$ （对于 $k \in \{0,1,\ldots,H-2\}$ ）附加到 $\mathfrak{I}_H$ 的根上时所用的权重为

$$(\mathrm{proc}_{k+1}(x) - 1) - (\mathrm{proc}_{k+2}(x) - 1) = \mathfrak{w}_k,$$

而 $\mathfrak{I}_{H-1}$所附着的权重为 $\mathrm{proc}_H(x) - 1 = \mathrm{proc}_H(x) - \mathrm{proc}_{H+1}(x) =: \mathfrak{w}_{H-1}$。简而言之，我们得到如下定义：$\mathfrak{I}_H$ 使得 $\mathrm{Expand}(\mathfrak{I}_H) \cong \mathrm{Tree}_\Gamma(x)$：



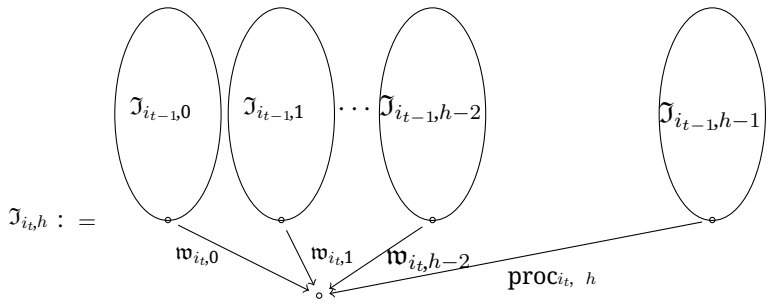
我们可以运用类似的思路，对每个指标为 d 的 $f \in \mathbb{F}_q$ 的广义分圆映射，描述 Γ_{per} 中非零周期顶点上方的有根树，即 Γ_f 在所有周期块 C_i 的并集上的诱导子图（特别地，当 $\overline{f}$ 是置换时，我们可以由此完整描述 Γ_f）。设 $i \in \{0,1,\ldots,d-1\}$ 为 $\overline{f}$-周期的，其 f-循环为（i_0，i_1，...，$i_{\ell-1}$），其中 $i_0 = i$。对于 $t \in \mathbb{Z}$，我们令 $i_t := i_t \bmod \ell$。定理 3.2.1 指出，固定 $t \in \mathbb{Z}$ 和 $h \in \mathbb{N}^+$ 后，如果 x、$y \in C_{i_t}$ 在 Γ^*_{per} 中各自至少有 h 代后继（即，如果 $\min\{\mathrm{proc}^{(\Gamma^*_{\mathrm{per}})\,h}(x), \mathrm{proc}^{(\Gamma^*_{\mathrm{per}})\,h}(y)\} > 0$），那么 $\mathrm{proc}^{(\Gamma^*_{\mathrm{per}})\,h}(x) = \mathrm{proc}^{(\Gamma^*_{\mathrm{per}})\,h}(y)$。因此，对于在 Γ^*_{per} 中至少有 h 代后继的任意 $x \in C_{i_t}$ （例如一个 f-周期的 x），我们可以设 $\mathrm{proc}_{i_t,h} := \mathrm{proc}^{(\Gamma^*_{\mathrm{per}})\,h}(x)$；此记号与定理 3.2.1 证明中使用的记号一致。

根据定理 3.2.1 前的注释，对于周期的$x \in C_{i_t}$，Tree$^{\Gamma_{\mathrm{per}}}(x)$ 的同构类型仅取决于 i_t 以及数值 $\mathrm{proc}_{i_{t'},\ h}$ （其中 $h \geq 1$ 和 $t' \in \mathbb{Z}$ ）（也就是说，它与 x 的选择无关）。下面说明如何根据这些数据读出这棵有根树。我们令

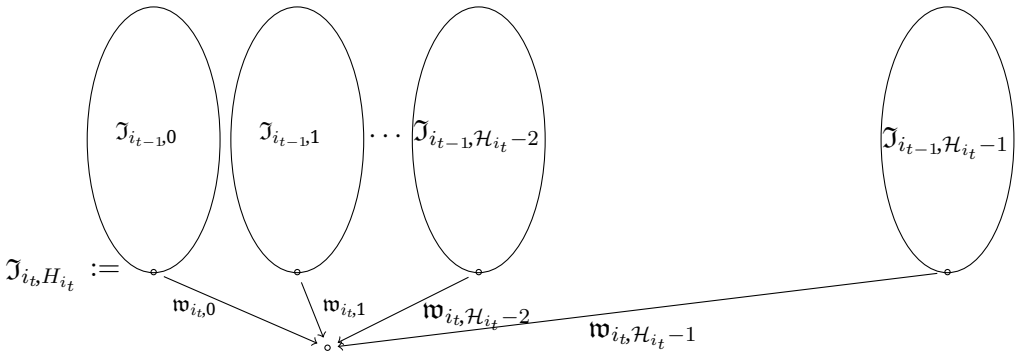
$$\mathcal{H}_{i_t} := \min\{h \in \mathbb{N}^+ : \mathrm{proc}_{i_t,h} = 1\} - 1,$$

Γ_{per} 中周期顶点C_{i_t}上有根树的共同高度。此外，与第 3.3 小节一样，我们令 $H_i := \max\{\mathcal{H}_{i_t} :\ t = 0, 1, \ldots, \ell - 1\}$，即沿着 i循环的此类树的最大高度。于是，作为对上述内容的推广，对于所有顶点$x \in C_{i_t}$，Tree$^{\Gamma_{\mathrm{per}}}(x)$的同构类型仅取决于$i_t$、数字 $\mathrm{proc}_{i_{t'},\ h}$ 以及 $\mathfrak{h}$的 x -值（ $\mathfrak{h}$的定义见第 3.3 小节中的公式 (14)）。需要指出的是，$\mathfrak{h}$ 在每个陪集C_{i_t} 上不一定会取遍所有可能的值 0、1、.....、 H_i（见例 3.2.2），但这并不影响我们的构造。

我们递归地定义边加权有根树 $\mathfrak{I}_{i_t,\ h}$ ，使得 Γ_{per} 中任意具有 $\mathfrak{h}(x) = h$ 的 $x \in C_{i_t}$ 上方的有根树都与 Expand($\mathfrak{I}_{i_t,\ h}$) 同构；显然，当该 $\mathfrak{h}$-值没有 $x \in C_{i_t}$ 时，这一性质必然成立。对于 $k \in \mathbb{N}_0$，我们令 $\mathfrak{w}_{i_t,\ k} := \mathrm{proc}_{i_t,\ k+1} - \mathrm{proc}_{i_t,\ k+2}$。我们将 $\mathfrak{I}_{i_t,\ 0}$ 定义为平凡有根树。如果 $h \in \{1, 2, \ldots, H_i - 1\}$（我们注意到，这种 $\mathfrak{h}$-值的 C_{i_t} 中的顶点是 f-暂态的），那么我们令



最后，C_{i_t} 中任意一个 f-周期的顶点（等价地，其 $\mathfrak{h}$值为 H_i）上方的有根树可构造如下：



4.2 一个说明性示例

在本小节中，我们沿用第 3 节的方法，推导出有根树同构类型的循环序列，以刻画下列指标为 $d = 5$ 的广义分圆映射 f of$\mathbb{F}_{2^8}$ 的函数图的连通分量：

$$f(x) = \begin{cases} 0, & \text{if } x = 0, \\ \omega^5 x^9, & \text{if } x \in C_0, \\ x^3, & \text{if } x \in C_1, \\ x^{17}, & \text{if } x \in C_2, \\ \omega^3 x^{34}, & \text{if } x \in C_3, \\ \omega^4 x^9, & \text{if } x \in C_4, \end{cases}$$

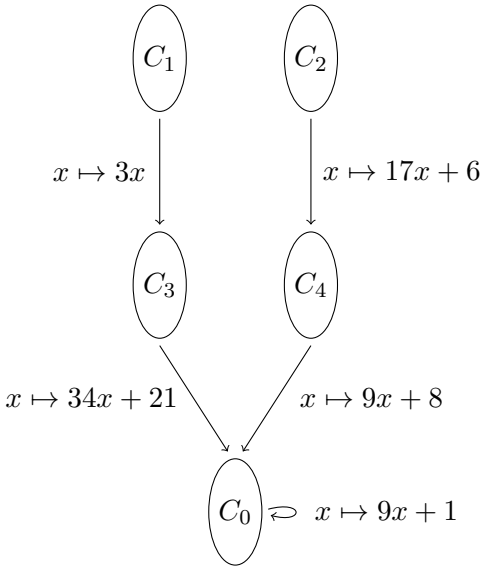
其中 ω 是 $\mathbb{F}_{2^8}$ 的任意一个固定本原元（ ω 在 $\mathbb{F}_2$ 上的最小多项式在此处并不相关）。这些循环序列也在我们的引言中通过 Γ_f 的图示推导出来（参见定义 1.4和1.5之间的正文段落），但第3节的方法通常在计算上更高效（参见第 5 节，尤其是定理 5.1.9）。

我们注意到，如果一个已知指标的有限域上的广义分圆映射不是以上述分圆形式给出，而是以多项式形式给出，那么在应用我们的方法之前，必须先将其转换为分圆形式。用于执行此转换的算法是 [13, 算法 1]。

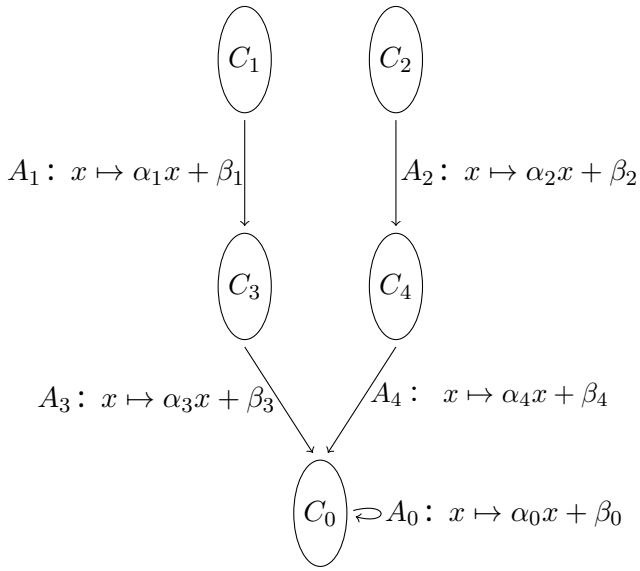
由于 $d = 5$，我们有 $s = (2^8 - 1)/5 = 51 = 3 \cdot 17$。我们将每个陪集 C_i 视为 $\mathbb{Z}/51\mathbb{Z}$ 的一个副本，通过双射 $\iota_i : \mathbb{Z}/51\mathbb{Z} \to C_i, k \mapsto \omega^{i+5k}$ 进行识别。下面我们来推导在这种识别下，f 在不同情形中的取值所对应的单项式公式。例如，若 $x \in C_4$，则对于某个 $k \in \mathbb{Z}$，有 $x = \omega^{4+5k}$，并且

$$f(x) = \omega^4 x^9 = \omega^{4+9\cdot 4+9\cdot 5k} = \omega^{0+5\cdot(9k+8)},$$

这表明 f通过仿射映射 $x \mapsto 9x + 8$将 C_4映射到 C_0。总之，将 C_i视为 $\mathbb{Z}/51\mathbb{Z}$的副本时，我们得到以下图示，描述了 f 在这些陪集之间的映射行为。



在另行通知之前，我们先搁置上文的具体函数 f，并假设更一般地，我们有一个有限域 $\mathbb{F}_q$，其中 $5 \mid q-1$，以及一个 $\mathbb{F}_q$ 的 5 阶广义圆分映射 f。将 C 的五个陪集视为 $\mathbb{Z}/s\mathbb{Z}$ 的副本时，该映射在它们之间的作用如下（其中 $s = (q-1)/5$）。



这使我们能够用一般系数 α_i 和 β_i来描述 Γ_f，从而更具启发性；实际上，可以看出相关参数的公式结构，例如$\mathcal{P}_i$的生成同余式的模数$\mathfrak{a}_{i,j}$ 和右端 $\mathfrak{b}_{i,j}$。与上文具体的广义分圆映射一样，我们假设 $\gcd(\alpha_0, s) = \gcd(\alpha_0^2, s) > 1$，这意味着在 C_0 上的 Γ_f 诱导子图中，附着于周期顶点的有根树高度为 $H_0 = 1$。对于 $\mathcal{P}_i$的每个块 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$，当 $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$时，我们描述算术划分 $\mathcal{P}_i = \mathfrak{P}(x \equiv \mathfrak{b}_{i,j} (\text{模 } \mathfrak{a}_{i,j}) \colon j = 1, 2, \ldots, m_i)$以及相应的有根树同构类型 $\mathsf{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$。

对于 i =1,2,3, 4，可以很容易地确定划分 $\mathcal{P}_i$ 及其相应的有根树。

- 对于 $i \in \{1,2\}$， C_i 中的每个顶点都是 Γ_f中的叶节点，因此我们可以选择 $\mathcal{P}_1 = \mathcal{P}_2 = \mathcal{P}(\emptyset)$（仅含一个块的平凡划分）。这里有且仅有一种有根树的同构类型，即 $\mathsf{Tree}_i(\mathcal{P}_i, \emptyset)$（其中 $\emptyset$ 表示空的逻辑符号序列，而非正逻辑符号），它由一个没有边的单个顶点构成。

- 对于 $i \in \{3,4\}$，由于 C_i 是一个暂态陪集（即它不位于 f作用下的陪集循环上），第 3.3 小节中的讨论表明，可以直接通过提升 $\mathfrak{P}'(\mathcal{P}_{i-2}, A_{i-2})$ 得到$\mathcal{P}_i$ 。根据引理 2.2.2，该提升的形式为 $\mathcal{P}_i = \mathfrak{P}(x \equiv \beta_{i-2} \pmod{\gcd(\alpha_{i-2}, s)})$。这个同余式的重要意义在于，它刻画了 $x \in C_i$ 在 C_{i-2}中是否存在至少一个关于f 的原像。我们注意到，若存在这样的原像，则 x 恰有 $\gcd(\alpha_{i-2}, s)$ 个此类原像，因为它们构成 $\mathbb{Z}/s\mathbb{Z}$中 $z \mapsto \alpha_{i-2}z$ 的核的一个陪集。因此，$\mathsf{Tree}_i(\mathcal{P}_i, (\neg))$是一个没有弧的单个顶点，而 $\mathsf{Tree}_i(\mathcal{P}_i, (\emptyset))$则由一个根及附着其上的 $\gcd(\alpha_{i-2}, s)$ 个顶点构成。

在我们关于 $\mathcal{P}_0$ 的讨论中，我们不指定与其自身的一个块 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 相关联的有根树 $\mathsf{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$，而是指定一个有限边加权有向根树（不一定是简化的同构类型） $\mathfrak{I} = \mathfrak{I}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$，使得 $\mathsf{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}) = \mathsf{Expand}(\mathfrak{I})$。但首先，让我们确定 $\mathcal{P}_0$ 本身。我们回忆，根据假设， $H_0 = 1$ 。根据周期的 $\mathcal{P}_i$ 的一般定义（该定义位于命题 3.3.5 之前），我们有

$$\mathcal{P}_0 = \mathcal{Q}_{0,1} = \mathcal{P}_{0,1} \wedge \mathcal{U}_0.$$

此外，注意到 $i = 0$ 位于 $\overline{f}$ 的一个长度为 1 的循环上（因此按照第 3.3 小节的记号，对于所有 $t \in \mathbb{Z}$ 都有 $i_t = 0$ ），我们得到

$$\mathcal{P}_{0,1} = \lambda_0^0(\mathcal{R}_0) \wedge \lambda_0^1(\mathcal{R}_0) = \mathcal{R}_0 \wedge \lambda(\mathcal{R}_0, A_0).$$

现在， $\mathcal{R}_0$ 是将 $\mathcal{P}_3$ 和 $\mathcal{P}_4$ 的 $\mathfrak{P}'$提升到 C_0后所得结果的下确界。为简便起见，用记号 $(n,\ m)$代替最大公约数 $\gcd(n,\ m)$，我们得出

$$\mathcal{R}_0 = \mathfrak{P}'(\mathcal{P}_3, A_3) \wedge \mathfrak{P}'(\mathcal{P}_4, A_4) = \mathfrak{P}\begin{pmatrix} x \equiv \alpha_3\beta_1 + \beta_3 \pmod{(\alpha_3(\alpha_1, s), s)} \\ x \equiv \beta_3 \pmod{(\alpha_3, s)} \\ x \equiv \alpha_4\beta_2 + \beta_4 \pmod{(\alpha_4(\alpha_2, s), s)} \\ x \equiv \beta_4 \pmod{(\alpha_4, s)} \end{pmatrix}$$

$$= \mathfrak{P}\begin{pmatrix} x \equiv \alpha_3\beta_1 + \beta_3 \left(\mathrm{mod}\ (\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1, s)})\right) \\ x \equiv \beta_3 \pmod{(\alpha_3, s)} \\ x \equiv \alpha_4\beta_2 + \beta_4 \left(\mathrm{mod}\ (\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2, s)})\right) \\ x \equiv \beta_4 \pmod{(\alpha_4, s)} \end{pmatrix}, \tag{18}$$

因此

$$\lambda(\mathcal{R}_0, A_0) = \mathfrak{P}\left(\begin{array}{l} x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_0(\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1,s)}), s)\right) \\ x \equiv \alpha_0\beta_3 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_3, s), s)) \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_0(\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2,s)}), s)\right) \\ x \equiv \alpha_0\beta_4 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_4, s), s)) \end{array}\right)$$

$$= \mathfrak{P}\left(\begin{array}{l} x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_1, s)(\alpha_0(\alpha_3, \frac{s}{(\alpha_1,s)}), \frac{s}{(\alpha_1,s)})\right) \\ x \equiv \alpha_0\beta_3 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_3, s), s)) \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_2, s)(\alpha_0(\alpha_4, \frac{s}{(\alpha_2,s)}), \frac{s}{(\alpha_2,s)})\right) \\ x \equiv \alpha_0\beta_4 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_4, s), s)) \end{array}\right)$$

此外，根据公式 (16) 以及紧随其后的 $\mathcal{U}_i$ 的定义，我们有

$$\mathcal{U}_0 = \mathfrak{P}(\theta_{0,1}) = \mathfrak{P}(x \equiv \beta_0 \ (\mathrm{mod}\ (\alpha_0, s))).$$

由此可得

$$\mathcal{P}_0 = \mathfrak{P}\left(\begin{array}{l} x \equiv \alpha_3\beta_1 + \beta_3 \ \left(\mathrm{mod}\ (\alpha_1, s)(\alpha_3, \frac{s}{(\alpha_1,s)})\right) \\ x \equiv \beta_3 \ (\mathrm{mod}\ (\alpha_3, s)) \\ x \equiv \alpha_4\beta_2 + \beta_4 \ \left(\mathrm{mod}\ (\alpha_2, s)(\alpha_4, \frac{s}{(\alpha_2,s)})\right) \\ x \equiv \beta_4 \ (\mathrm{mod}\ (\alpha_4, s)) \\ x \equiv \alpha_0\alpha_3\beta_1 + \alpha_0\beta_3 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_1, s)(\alpha_0(\alpha_3, \frac{s}{(\alpha_1,s)}), \frac{s}{(\alpha_1,s)})\right) \\ x \equiv \alpha_0\beta_3 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_3, s), s)) \\ x \equiv \alpha_0\alpha_4\beta_2 + \alpha_0\beta_4 + \beta_0 \ \left(\mathrm{mod}\ (\alpha_2, s)(\alpha_0(\alpha_4, \frac{s}{(\alpha_2,s)}), \frac{s}{(\alpha_2,s)})\right) \\ x \equiv \alpha_0\beta_4 + \beta_0 \ (\mathrm{mod}\ (\alpha_0(\alpha_4, s), s)) \\ x \equiv \beta_0 \ (\mathrm{mod}\ (\alpha_0, s)) \end{array}\right). \quad (19)$$

现在我们转而确定任意给定块 B 的 $\mathcal{P}_0$中顶点上方的有根树。更具体地说，我们有 $B = \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$，其中 $\vec{\nu}^{(\mathcal{P}_0)} = (\nu_1, \ldots, \nu_9) \in \{\emptyset, \neg\}^9$ 是 $\mathcal{P}_0$的九个生成同余式的逻辑符号元组。将 $\vec{\nu}^{(\mathcal{P}_0)}$ 分成若干段是有帮助的；也就是说，按照命题 3.3.5 中的记号，我们写作 $\vec{\nu}^{(\mathcal{P}_0)} = \vec{o_0'} \diamond \vec{o_1'} \diamond \vec{\xi}$其中

- $\vec{o_0'} = (\nu_1, \nu_2, \nu_3, \nu_4)$ 控制 $\mathcal{R}_0$ 的哪个块 $\mathcal{B}(\mathcal{R}_0, \vec{o_0'})$ 包含块 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$。根据命题 3.3.3，了解 o_0' 中的逻辑符号，就足以对所有 $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 一致地理解来自位于 $C_3 \cup C_4$ 中的 x 的那些原像（即映射到 C_0 的所有暂态陪集的并集）对树 $\Gamma_f(x)$ 的贡献 $\vec{\text{树}}\Gamma_f(x, C_3 \cup C_4) = \text{树}_0(\mathcal{R}_0, C_3 \cup C_4, o_0')$。

- $\vec{o_1'} = (\nu_5, \nu_6, \nu_7, \nu_8)$ 决定了 $\mathcal{S}_{0,1} = \lambda(\mathcal{R}_0, A_0)$ 的哪个块包含 $\vec{\mathcal{B}}(\mathcal{S}_{0,1}, o_1')$ 中的$\mathcal{P}_0$-块 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$。根据命题 3.3.4，只需知道 $\vec{o_1'}$ 中的逻辑符号，就足以统一理解所有 $\mathfrak{h}$-值为 1 的 $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$，也就是说，哪些是 f-周期的（或者等价地说，哪些在

Γ_{per}）中是非叶节点），$\mathsf{Tree}^{\Gamma_f}(x, C_0)$ 对 $\mathsf{Tree}^{\Gamma_f}(x)$ 的贡献 $\mathsf{Tree} = \mathsf{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o_1'})$，来自 x 的那些位于 C_0 中（映射到 C_0 的唯一周期陪集）的原像。注意，如果 $x \in C_0$ 的 $\mathfrak{h}$-值为 0，即如果 x 是 f-暂态的（或者等价地说，如果 x 是 Γ_{per} 中的叶节点），那么 $\mathsf{Tree}^{\Gamma_f}(x, C_0)$ 是平凡的，因为 x 在 C_0 中没有 f-暂态原像。

- $\vec{\xi} = (\nu_9)$ 控制 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 中顶点的 $\mathfrak{h}$ 值；若 $\nu_9 = \neg$，则这些顶点在 Γ 中每个 都是叶节点（即它们的 $\mathfrak{h}$ 值为 0），否则它们都是周期顶点（即它们的 $\mathfrak{h}$ 值为 $1 = H_0$）。

让我们更具体地说明对于 $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 而言，对树 $\Gamma_f(x)$ 的这些不同贡献。我们回顾一下，根据定义，

$$\mathcal{R}_0 = \mathcal{P}_3' \wedge \mathcal{P}_4' = \mathfrak{P}'(\mathcal{P}_3, A_3) \wedge \mathfrak{P}'(\mathcal{P}_4, A_4),$$

并注意到 $\vec{o_0'}$ 可以写成串联形式 $\vec{\nu}^{(\mathcal{P}_3')} \diamond \vec{\nu}^{(\mathcal{P}_4')}$，其中 $\vec{\nu}^{(\mathcal{P}_3')} = (\nu_1, \nu_2)$ 分别为 $\vec{\nu}^{(\mathcal{P}_4')} = (\nu_3, \nu_4)$，控制着 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 在 $\mathcal{P}_3'$-块以及 $\mathcal{P}_4'$-块中的包含关系。对于 $i \in \{3, 4\}$，了解 $\vec{\nu}^{(\mathcal{P}_i')}$ 中的逻辑符号，就足以统一地理解所有 $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 下，由那些位于 C_i 中的 x 的原像所产生的贡献 $\mathsf{Tree}^{\Gamma_f}(x, C_i) = \mathsf{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$ 对 $\mathsf{Tree}^{\Gamma_f}(x)$ 的贡献。

当然，对于每个给定的 $x \in C_0$，我们有

$$\mathrm{Tree}_{\Gamma_f}(x) = \mathrm{Tree}_{\Gamma_f}(x, C_0 \cup C_3 \cup C_4) = \sum_{i \in \{0,3,4\}} \mathrm{Tree}_{\Gamma_f}(x, C_i).$$

鉴于上文关于这三种对 $\mathsf{Tree}^{\Gamma_f}(x)$ 的不同贡献所述，我们有以下公式（也可以由命题 3.3.3(2)、命题 3.3.5 以及命题 3.3.4 中的最后一个公式推导得出）：

$$\begin{aligned}&\mathrm{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}) = \\ &\begin{cases} \sum_{i=3}^{4} \mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')}), & \text{if } \nu_9 = \neg, \\ \sum_{i=3}^{4} \mathrm{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')}) + \mathrm{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o_1'}), & \text{if } \nu_9 = \emptyset. \end{cases}\end{aligned} \tag{20}$$

特别地，$\vec{o_1'}$ 中的逻辑符号 $\nu_5, \nu_6, \nu_7, \nu_8$ 对 $\mathsf{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 的值无关紧要，只要 $\nu_9 = \neg$。

公式 (20) 使我们能够将确定 $\mathsf{Tree}_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 的任务分解为若干子任务。首先，对于 $i = 3, 4$，我们确定 $\mathsf{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$，这可以统一完成。我们注意到

$$\mathcal{P}_i' = \mathfrak{P}\left(\begin{array}{l} x \equiv \alpha_i\beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)} \\ x \equiv \beta_i \pmod{(\alpha_i, s)} \end{array}\right).$$

$\vec{\nu}^{(\mathcal{P}_i'}$的两个条目$) = (\nu_{2i-5}, \nu_{2i-4})$ 是分别对应于那两个同余式的逻辑符号，我们需要根据它们的真值进行分类讨论。我们可以在每种情况下，按照命题 3.3.3(1)，利用引理 2.2.2 中关于 $\sigma_{\mathcal{P}_i, A_i}(\vec{\nu}^{(\mathcal{P}_i)} \ \vec{\nu}^{(\mathcal{P}_i')})$ 的公式，“机械地”计算 $\mathsf{Tree}_0(\mathcal{P}_i', C_i, \vec{\nu}^{(\mathcal{P}_i')})$。不过，用直接论证推导出这些结果更有启发性（有兴趣的读者仍可自行采用公式化的方法，并进行比较）。

• 如果 $\nu_{2i-4} = \neg$，即，如果 $x \not\equiv \beta_i$ (模(α_i, s))，那么 x 在C_i中根本没有原像，因此 树$_0(\mathcal{P}'_i, C_i, \vec{\nu}_{(\mathcal{P}'_i)})$是一个没有弧的单个顶点（ ν_{2i-5}的值在此处无关紧要）。

• 如果 $\nu_{2i-4} = \emptyset$，也就是说，如果 $x \equiv \beta_i$ (模 (α_i, s))，那么 x 在 C_i 中恰有 (α_i, s) 个子节点（它们构成 $z \mapsto \alpha_i z$的核的一个陪集）。我们需要确定这些子节点在 $\mathcal{P}_i = \mathfrak{P}$的两个块中的分布（$x \equiv \beta_{i-2}$ (模 (α_{i-2}, s))），而该分布取决于 ν_{2i-5} 的真值

$$x \equiv \alpha_i\beta_{i-2} + \beta_i \pmod{(\alpha_i(\alpha_{i-2}, s), s)}. \tag{21}$$

事实上，按照引理 2.2.2 的证明， C_i 中 x 的原像 y，满足 $y \equiv \beta_{i-2}$ (模 (α_{i-2}, s)) （注意，它们恰好是 C_i 中 x 的那些不是 Γ_f中叶节点的原像），可由以下同余方程组刻画

$$\begin{aligned} y &\equiv \beta_{i-2} \pmod{(\alpha_{i-2}, s)} \\ \alpha_i y + \beta_i &\equiv x \pmod{s}, \end{aligned}$$

根据引理 2.2.2 的证明，当且仅当同余式 (21) 成立时，该系统才相容；此时该系统等价于模

$$\operatorname{lcm}\left((\alpha_{i-2}, s), \frac{s}{(\alpha_i, s)}\right).$$

因此，若 $\nu_{2i-5} = \emptyset$，即同余式 (21) 成立，则 x 恰有

$$\frac{s}{\operatorname{lcm}\left((\alpha_{i-2}, s), \frac{s}{(\alpha_i, s)}\right)} = \left(\frac{s}{(\alpha_{i-2}, s)}, (\alpha_i, s)\right)$$

原像 $y \in C_i$ 与 $y \equiv \beta_{i-2}$ （模（α_{i-2}, s）），这些正是位于 $\mathcal{B}$（$\mathcal{P}_i$,（$\emptyset$））中的原像。否则， x 在 C_i 中的所有原像都不同余于 β_{i-2}模（α_{i-2}, s），因而位于 $\mathcal{B}$（$\mathcal{P}_i$,（$\neg$））中。鉴于树$_i$（$\mathcal{P}_i$,（ν））关于 $\nu \in \{\emptyset, \neg\}$的已知值，我们发现，树$_0$（$\mathcal{P}'_i, C_i, \vec{\nu}^{(\mathcal{P}'_i)}$）是表3中所指定的边权重有向根树的 Expand 版本（不一定是简化的）。

$\mathcal{P}'_i = \mathfrak{P}'(\mathcal{P}_i, A_i)$的块	关联的树 $\Gamma_f(x, C_i)$
$\nu(x \equiv \alpha_i\beta_{i-2} + \beta_i$ (mod $(\alpha_i(\alpha_{i-2}, s)\ s)))$ $x \not\equiv \beta_i$ (模 (α_i, s)) $\nu \in \{\emptyset, \neg\}$	∘
$x \not\equiv \alpha_i\beta_{i-2} + \beta_i$ (模 $\alpha_i(\alpha_{i-2}, s)\ s$) $x \equiv \beta_i$ (模 (α_i, s))	∘ $\downarrow$ (α_i, s) ∘

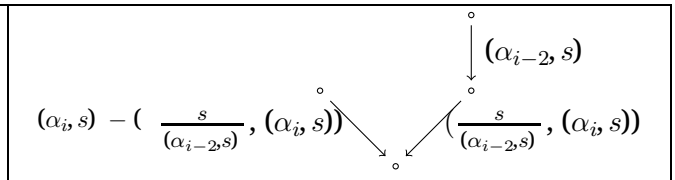
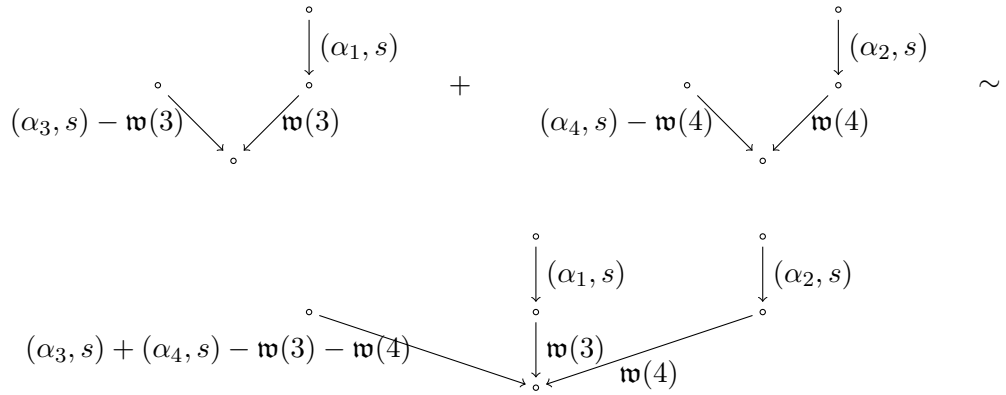
$x \equiv \alpha_i\beta_{i-2} + \beta_i$ (模 $\alpha_i(\alpha_{i-2}, s)$ s) $x \equiv \beta_i$ (模 (α_i, s))	

表3：仅使用暂态原像陪集中的原像的有根树 C_i 与 $i \in \{3, 4\}$。

这确定了公式（20）中两种情形下树$_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$的前两项。如果 $\nu_9 = \neg$ （即，如果相关的 $\mathcal{P}_0$-块由 f-暂态点组成），那么这些就是公式中的全部项，此时只需将表3中读出的、各个 $i \in \{3, 4\}$对应的树$_i(\mathcal{P}'_i, \vec{\nu}^{(\mathcal{P}'_i)})$（及其带边权的版本）相加，即可简单地得到树$_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$（的带边权版本）。例如，如果 $\nu_j = \emptyset$ 对于$j = 1$、2、.....、8，但 $\nu_9 = \neg$，那么树$_0(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$（的带边权版本）如下，其中对于 $i = 3$、4，设 $\mathfrak{w}(i) := (^{s}{}_{(\alpha_{i-2},s)}, (\alpha_i, s))$：



另一方面，如果 $\nu_9 = \emptyset$（这样 $\mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)}$ 中的所有顶点都是 f-周期的），那么我们还需要计算公式 (20) 中的第三个加数，树$_0^{(1)}(\mathcal{S}_0, 1, C_0, \vec{o_1})$，它表示来自 C_0 中 f- 暂态原像的贡献。根据命题 3.3.4，我们可以通过研究任意给定点 $x \in \mathcal{B}(\mathcal{P}_0, \vec{\nu}^{(\mathcal{P}_0)})$ 的原像在划分 $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathfrak{P}(x \equiv \beta_0 \pmod{(\alpha_0, s)})$ 的某些块上的分布来完成这项工作。更具体地说，我们注意到， x 的每个 f-暂态原像都包含在 $\mathcal{Q}_{0,0}$ 中形如 $\mathcal{B}(\mathcal{Q}_0, 0, \vec{o_0} \diamond (\neg))$ 的块内，其中 $\vec{o_0} \in \{\emptyset, \neg\}^4$ （我们还注意到，$\mathcal{Q}_{0,0}$ 中每个这种形式的块都完全由 f-暂态点组成，因为最后一个逻辑符号是 $\neg$）。能够统计 x 在 $\mathcal{Q}_{0,0}$ 的每个此类块中的原像数量，就足以了解树$_0^{(1)}(\mathcal{S}_0, 1, C_0, \vec{o_1})$，因为 $\mathcal{B}(\mathcal{Q}_{0,0}, \vec{o_0} \diamond (\neg)) \subseteq \mathcal{B}(\mathcal{R}_0, \vec{o_0})$，而且我们已经了解了给定块 $\mathcal{B}(\mathcal{R}_0, \vec{o_0}) = \mathcal{B}(\mathcal{R}_0, \vec{\nu}^{(\mathcal{P}'_3)} \diamond \vec{\nu}^{(\mathcal{P}'_4)})$（属于 $\mathcal{R}_0$）中 f-暂态顶点上方的有根树。

现在，我们来观察任意 f-周期点$x \in C_0$ 在 $\mathcal{Q}_{0,0}$ 各块上的原像分布，如何由 $j \in \{5, 6, 7, 8\}$对应的 ν_j 值控制，

→也就是说，由包含 x 的 $\mathcal{S}_{0,1}$ 中的块 $\mathcal{B}(\mathcal{S}_{0,1},o_1')$ 控制。这是因为 $x \in \mathcal{B}(\mathcal{T}_{0,1}, o_1' \diamond (\emptyset))$，其中 $\mathcal{T}_{0,1} = \mathcal{S}_{0,1} \wedge \mathcal{U}_0 = \mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$ 是一个划分。根据引理 2.2.2，该划分确实控制着 x 在 $\mathcal{Q}_{0,0}$ 各块上的原像分布。将此引理应用于此处，可得到命题 3.3.4 中 $\mathsf{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o_1'})$ 的公式。

例如，逻辑符号元组 $\vec{o_0} = (\neg, \neg, \emptyset, \emptyset)$对应于 $\mathcal{R}_0$ 的块$B := \mathcal{B}(\mathcal{R}_0, \vec{o_0})$。如果我们希望计算一个顶点$x \in C_i$（其 $x \equiv \alpha_0$（模 (α_0, s)）源自例如 $B' := \mathcal{B}(\mathcal{S}_{0,1}, o_1')$ 的块$\mathcal{S}_{0,1}$，且具有$o_1' = (\emptyset, \neg, \emptyset, \emptyset)$）有多少个暂态原像，那么我们需要计算

$$\sigma_{\mathcal{Q}_{0,0}, A_0}(\vec{o_0} \diamond (\neg), \vec{o_1'} \diamond (\emptyset)),$$

我们现在这样做是为了说明该方法。为避免读者产生混淆，我们指出，上述表达式与引理 2.2.2中使用的记号并不完全一致。事实上，这里我们对 $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathcal{T}_{0,0} = \mathcal{R}_0 \wedge \mathcal{U}_0$ 和 $\mathcal{T}_{0,1} = \mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$都使用长度为5 的张成同余序列。然而，在引理 2.2.2中，假设我们对 $\mathfrak{P}'(\mathcal{Q}_{0,0}, A_0)$ 使用张成同余序列的“标准格式”，该格式比 $\mathcal{Q}_{0,0}$ 的序列多包含一个同余式。这一差异的产生是因为我们将 $\mathcal{T}_{0,1}$ 写作 $\lambda(\mathcal{R}_0, A_0) \wedge \mathcal{U}_0$ —— 在“标准格式”中，它应改为

$$\begin{aligned}&\lambda(\mathcal{Q}_{0,0}, A_0) \wedge \mathfrak{P}(x \equiv \beta_0 \pmod{(\alpha_0, s)}) = \\ &\lambda(\mathcal{R}_0, A_0) \wedge \mathfrak{P}(x \equiv \beta_0(1+\alpha_0) \pmod{(\alpha_0^2, s)}) \wedge \mathcal{U}_0,\end{aligned}$$

但我们可以省略同余式 $x \equiv \beta_0(1+\alpha_0) \pmod{(\alpha_0^2, s)}$，它在第 3.3 小节的记号中为 θ_0,2(x)，因为（利用 $H_0 = 1$）它等价于$x \equiv \beta_0 \pmod{(\alpha_0, s)}$，即 $\mathcal{U}_0$ 的唯一生成同余式 θ_0,1(x)。为了应用引理 2.2.2，我们将 $\mathcal{T}_0$,1 写成不太简洁的标准格式，这要求→我们将块 $\mathcal{T}_0$,1 的逻辑符号序列 $o_1' \diamond (\emptyset) = (\emptyset, \neg, \emptyset, \emptyset, \emptyset)$ 替换为$\vec{\nu'} := \vec{o_1'} \diamond (\emptyset, \emptyset) = (\emptyset, \neg, \emptyset, \emptyset, \emptyset, \emptyset)$（即，我们将末尾的 $\emptyset$ 加倍），并将其第 j 个条目记为 ν_j'。块 $\mathcal{Q}_0$, 0 的逻辑符号序列保持为$\vec{\nu} := \vec{o_0} \diamond (\neg) = (\neg, \neg, \emptyset, \emptyset, \neg)$。现在我们的目标是计算

$$\sigma_{\mathcal{Q}_{0,0}, A_0}(\vec{\nu}, \vec{\nu'})$$

严格遵循引理 2.2.2。对于 $j = 1, \ldots, 5$，我们分别将 $\overline{\mathfrak{a}}_j$ 和 $\overline{\mathfrak{b}}_j$记为 $\mathcal{Q}_{0,0} = \mathcal{R}_0 \wedge \mathcal{U}_0$ 的第 j个生成同余式的模数和右端。也就是说，对于 $j \in \{1,2,3,4\}$，同余式 $x \equiv \overline{\mathfrak{b}}_j$ (模 $\overline{\mathfrak{a}}_j$)是 $\mathcal{P}_0$ 的公式 (19) 中第 j 个显示的同余式。此外，$\overline{\mathfrak{a}}_5 = (\alpha_0, s)$ 和 $\overline{\mathfrak{b}}_5 = \beta_0$。使用引理 2.2.2 中的记号，我们注意到

- $J_-(\vec{\nu}) = \{1,2,5\}$ (指标集合 $j \in \{1, \ldots, 5\}$，使得 $\vec{o_0} \diamond (\neg)$ 的第 j 项为 $\neg$);
- $J_+(\vec{\nu}) = \{1,2,3,4,5\} \setminus J_-(\vec{\nu}) = \{3,4\}$;
- $J_-(\vec{\nu'}) = \{2\}$;

• 对于 $J \subseteq J_{-}(\vec{\nu}) = \{1,2,5\}$，条件 $E(\vec{\nu}, J)$ 要求：“对于所有 $j_1, j_2 \in \{3,4\} \cup J$：最大公约数$(\overline{\mathfrak{a}}_{j_1}, \overline{\mathfrak{a}}_{j_2}) \mid \overline{\mathfrak{b}}_{j_1} - \overline{\mathfrak{b}}_{j_2}$”。

根据引理 2.2.2，我们有

$$\sigma_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}) = \sum_{J \subseteq J_{-}(\vec{\nu})} (-1)^{|J|} \kappa_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}, J)$$

其中

$$\kappa_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}, J) = \delta_{\nu'_6=\emptyset} \cdot \delta_{E(\vec{\nu},J)} \cdot \delta_{(J_{+}(\vec{\nu}) \cup J) \cap J_{-}(\vec{\nu'}) = \emptyset} \cdot \frac{s}{\operatorname{lcm}(\frac{s}{\gcd(\alpha_0, s)}, \overline{\mathfrak{a}}_{0,j} : j \in J_{+}(\vec{\nu}) \cup J)}.$$

在这个公式中，第一个克罗内克δ用于检查 $\vec{\nu'}$ 的最后一项是否为 $\emptyset$，而事实确实如此。第三个克罗内克δ当且仅当 $2 \notin J$ 时为 1，因此 $J \subseteq J_{-}$（$\vec{\nu}$）有四种可能取值：$\emptyset$、$\{1\}$、$\{5\}$、$\{1,5\}$；对于这些取值，$\kappa_{\mathcal{Q}_{0,\,0},\,A_0}$（$\vec{\nu}$、$\vec{\nu'}$、$J$）可能非零。由此可得

$$\begin{aligned}
&\sigma_{\mathcal{Q}_{0,0},A_0}(\vec{\nu}, \vec{\nu'}) = \\
&\delta_{E(\vec{\nu},\emptyset)} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \overline{\mathfrak{a}}_3, \overline{\mathfrak{a}}_4)} - \delta_{E(\vec{\nu},\{1\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \overline{\mathfrak{a}}_1, \overline{\mathfrak{a}}_3, \overline{\mathfrak{a}}_4)} \\
&- \delta_{E(\vec{\nu},\{5\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \overline{\mathfrak{a}}_3, \overline{\mathfrak{a}}_4, \overline{\mathfrak{a}}_5)} + \delta_{E(\vec{\nu},\{1,5\})} \frac{s}{\operatorname{lcm}(\frac{s}{(\alpha_0,s)}, \overline{\mathfrak{a}}_1, \overline{\mathfrak{a}}_3, \overline{\mathfrak{a}}_4, \overline{\mathfrak{a}}_5)}
\end{aligned}$$

是每个给定的 $x \in B' = \mathcal{B}(\mathcal{S}_{0,\,1},\ o'_1)$ 中 $B = \mathcal{B}(\mathcal{R}_0,\ \vec{o_0})$ 的 f-暂态子节点的数量。这些子节点各自提供一个

$$\operatorname{Tree}_0^{(0)}(\mathcal{P}_{0,0}, \vec{o_0}) = \operatorname{Tree}_0(\mathcal{R}_0, C_3 \cup C_4, \vec{o_0})$$

它连接在树 $\operatorname{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$ 的根上。若固定 $\vec{o'_1}$，并对 $\vec{o_0} \in \{\emptyset, \neg\}^4$ 的所有可能取值执行这一计算，就能得到 $\operatorname{Tree}_0^{(1)}(\mathcal{S}_{0,1}, C_0, \vec{o'_1})$ 的一幅“完整图景”。

现在，我们要将对有根树获得的明确理解具体应用到最初的实例中。首先，我们根据 $\mathcal{R}_0$ 的块来处理有根树 $\operatorname{Tree}^{\Gamma_f}(x, C_3 \cup C_4)$。将 $s = 51, \alpha_0 = 9, \alpha_1 = 3, \alpha_2 = 17, \alpha_3 = 34, \alpha_4 = 9, \beta_0 = 1, \beta_1 = 0, \beta_2 = 6, \beta_3 = 21, \beta_4 = 8$ 代入公式 (18)，得到

$$\mathcal{R}_0 = \mathfrak{P}\begin{pmatrix} x \equiv 21 \pmod{51} \\ x \equiv 4 \pmod{17} \\ x \equiv 11 \pmod{51} \\ x \equiv 2 \pmod{3} \end{pmatrix}.$$

这些同余式之间存在依赖关系。例如，第一个同余式蕴含第二个同余式，同时也蕴含第三个和第四个同余式的否定。表4列出了所有使 $\mathcal{B}(\mathcal{R}_0, o'_0)$ 非空的 $\vec{o'_0} \in \{\emptyset, \neg\}^4$，以及集合 $\mathcal{B}(\mathcal{R}_0, o'_0)$ 的描述，还有 $\mathcal{B}(\mathcal{R}_0, o'_0)$ 中每个点上方对应的有根树 $\operatorname{Tree}_0(\mathcal{R}_0, C_3 \cup C_4, o'_0)$（经简化的边加权形式），该树通过将 $i = 3$ 的有根树 $\operatorname{Tree}_0(\mathcal{P}'_i, C_i, \nu'_i)$（边加权形式）相加而得到；这些树可从表3中读出，4。

$\vec{o_0'}$	$B := \mathcal{B}(\mathcal{R}_0, \vec{o_0'})$	树$\Gamma_f(x, C_3 \cup C_4)$ 对于 $x \in B$
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \neg \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 2 \pmod 3\} \setminus \{4, 21\}$	∘
$\begin{pmatrix} \neg \\ \emptyset \\ \neg \\ \neg \end{pmatrix}$	$\{4\}$	∘ ↓ 17 ∘
$\begin{pmatrix} \emptyset \\ \emptyset \\ \neg \\ \neg \end{pmatrix}$	$\{21\}$	∘ ↓ 3 ∘ ↓ 17 ∘
$\begin{pmatrix} \neg \\ \neg \\ \neg \\ \emptyset \end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \equiv 2 \pmod 3\} \setminus \{11, 38\}$	∘ ↓ 3 ∘
$\begin{pmatrix} \neg \\ \neg \\ \emptyset \\ \emptyset \end{pmatrix}$	$\{11\}$	∘ ↓ 17 ∘ ↓ 3 ∘
$\begin{pmatrix} \neg \\ \emptyset \\ \neg \\ \emptyset \end{pmatrix}$	$\{38\}$	∘ ↓ 20 ∘

表4：仅使用暂态原像陪集中的原像的有根树。

现在，我们利用 x 所在的 $\mathcal{S}_{0,1}$-块 $\mathcal{B}(\mathcal{S}_{0,1}, o_1')$ 来描述 f-周期的 $x \in C_0$ 在 $\vec{}$ 中的 $\mathsf{Tree}^{\Gamma_f}(x, C_0)$。首先，将 s 、 α_i 和 β_i 的具体取值代入 $\mathcal{S}_{0,1}$ 的四个生成同余式，得到

$$\mathcal{S}_{0,1} = \mathfrak{P} \begin{pmatrix} x \equiv 37 \pmod{51} \\ x \equiv 37 \pmod{51} \\ x \equiv 49 \pmod{51} \\ x \equiv 1 \pmod{3} \end{pmatrix}.$$

无需严格按照引理 2.2.2 中描述的计算过程

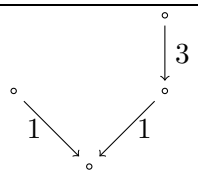
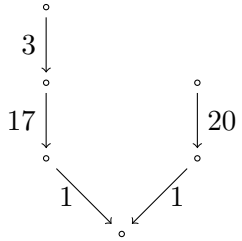
（如上所述）就能全面理解这里的树——我们改为给出一个概念性的论证。

我们注意到，根据引理 2.1.14， $\mathbb{Z}/51\mathbb{Z}$ 中在A_0： $x \mapsto 9x+1$ 作用下呈周期性的点，恰好是模 3 同余于 1 的那些点（模 $3 = \gcd(\alpha_0^L, s)$ 时 A_0 的唯一不动点，其中 L 如引理 2.1.14 所述）。因此，对于我们正在考虑的那些 x ， $\mathcal{S}_{0,1}$ 的最后一个生成同余式总是成立。我们注意到，$\mathcal{S}_{0,1}$ 的张成序列包含了用于刻画周期顶点的同余式，这纯属巧合——一般来说，若想通过划分块来控制这一点，就需要从“外部”补充这条信息，并改用划分 $\mathcal{T}_{0,1} = \mathcal{S}_{0,1} \wedge \mathcal{U}_0$ 。

由于 C_0 中 x 的子节点构成 $z \mapsto 9z$ 的核 $17\mathbb{Z}/51\mathbb{Z}$ 的一个陪集，因此 C_0 中的 x 恰有三个原像，每个模 3 同余类中各有一个。不过， C_0 中满足 $y \equiv 1 \pmod 3$ 的 x 的原像 y 是 f-周期的，因此不会出现在 $\mathsf{Tree}^{\Gamma_f}(x, C_0)$ 中。我们还注意到以下几点。

- 对于 $x = 37$，剩下的两个 f-暂态原像是 21 和 38。
- 对于 $x = 49$ ， f-暂态原像为 11 和 45。
- 对于所有其他 $x \equiv 1$ （模 3），在 $\mathcal{R}_0$的两个“通用”块 $\mathcal{B}(\mathcal{R}_0, (\neg, \neg, \neg, \neg))$ 和 $\mathcal{B}(\mathcal{R}_0, (\neg, \neg, \neg, \emptyset))$ 中，各有一个 f-暂态原像；这是因为除 $\{4\}$ 以外的所有其他块都已被“用尽”，并且 $4 \equiv 1$ （模 3）。

从表4中，我们可以读出表5中指定的树$\mathsf{Tree}^{\Gamma_f}(y, C_3 \cup C_4)$的形状：对于 C_0中的 x 的两个 f-暂态原像 y，分别为$\mathsf{Tree} = \mathsf{Tree}^{\Gamma_f}(y)$。

$\vec{o}_1'$	$B' := \mathcal{B}(\mathcal{S}_{0,1},\ \vec{o}_1')$	$x \in B'$ 的树$\Gamma_f(x, C_0)$，具有周期的
$\begin{pmatrix}\neg\\ \neg\\ \neg\\ \neg\end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 1 \text{ (模 3)}\}$	不适用（此块中没有周期的 x ）
$\begin{pmatrix}\neg\\ \neg\\ \neg\\ \emptyset\end{pmatrix}$	$\{x \in \mathbb{Z}/51\mathbb{Z} :\ x \equiv 1 \text{ (模 3)}\} \setminus \{37, 49\}$	
$\begin{pmatrix}\emptyset\\ \emptyset\\ \neg\\ \emptyset\end{pmatrix}$	$\{37\}$	

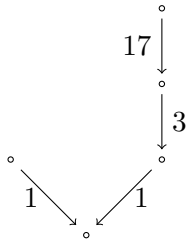
$\begin{pmatrix} \neg \\ \neg \\ \emptyset \\ \emptyset \end{pmatrix}$	$\{49\}$	

表5：仅使用 C_0 中原像的有根树。

最后，我们的公式(20)给出了Tree$^{\Gamma_f}$(x)的列表，我们在表6中列出了该列表；此外，我们还引入记号 $\mathfrak{I}_n$ 来表示 $n = 1$、2、3、4，以指代周期的 x所对应的Tree$^{\Gamma_f}$(x)的不同同构类型。另外，我们定义 $\mathfrak{I}_0$ 来表示平凡有根树的同构类型，该树由一个没有弧的单个顶点组成。

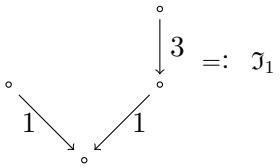
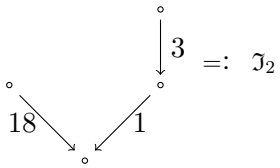
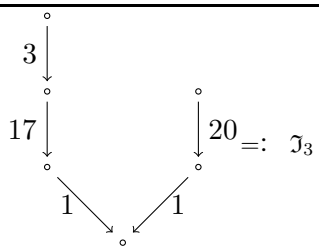
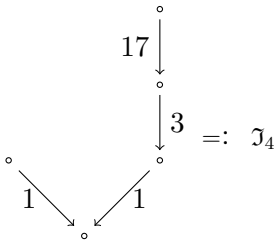
块 B 的 $\mathcal{P}_0$	$x \in B$的树$^{\Gamma_f}$(x)
$\{x \in \mathbb{Z}/51\mathbb{Z} : x \equiv 1 \text{ (模 3)}\} \setminus \{4, 37, 49\}$	
$\{4\}$	
$\{37\}$	
$\{49\}$	
$B \subseteq \{x \in \mathbb{Z}/51\mathbb{Z} : x \not\equiv 1 \text{ (模 3)}\}$	树$^{\Gamma_f}$($x, C_3 \cup C_4$)（见表4）

表6： $x \in C_0$ 的有根树 Tree$^{\Gamma_f}$(x)。

现在我们已经充分理解了 Γ_f 中各顶点上方的有根树，接下来来确定 f 的周期点，并为 f 构造 CRL 列表。按照第 3.1 小节中的讨论，f 的周期点包括域元素 0，以及唯一周期陪集 C_0 中 f 的所有周期点。我们已经在上文（利用引理 2.1.14）确定，这些周期点恰好是满足 $x \equiv 1 \pmod 3$ 的那些 $x \in \mathbb{Z}/51\mathbb{Z}$，因此 C_0 中有 $51/3 = 17$ 个周期点。周期点数量如此之少，直接用穷举法确定循环结构并构造 CRL 列表并不困难，但我们仍希望按照第 3.1 小节（以及第 3.1 小节所依据的第 2.3 小节）中描述的方法进行，以展示这一方法。

首先，我们注意到形如 $p^{\nu_p(51)}$ 的非平凡素数幂只有 3 和 17。按照第 2.3 小节的方法，我们需要确定 A_0 模 $p^{\nu_p(51)}$的每个双射约化的 CRL 列表；也就是说，这里只需确定 A_0模 17 的约化。我们可以从表 2 中直接读出这样的 CRL 列表。具体来说，由于$\nu_{17}^{(1)}(1) = 0 \geq 0 = \nu_{17}^{(1)}(9-1)$，且 ord(9)，即 9 模 17 的乘法阶，为 8，因此该表中的情形 1 告诉我们

$$\mathfrak{r} := 3 \text{ and } \mathfrak{f} := -\frac{1}{17^0} \cdot \mathrm{inv}_{17}\left(\frac{8}{17^0}\right) = -15 = 2$$

告诉我们

$$\{(3^0 17^0 + 2, 8), (3^1 17^0 + 2, 8), (3^0 17^1 + 2, 1)\} = \{(3, 8), (5, 8), (2, 1)\}$$

是 A_0 模 17 的 CRL 列表。模 3 时，A_0 的唯一周期点是 1，因此，为了得到 A_0 模 51 的 CRL 列表，我们只需将上述模 17 的 CRL 列表的首项通过函数 $\Lambda : \mathbb{Z}/17\mathbb{Z} \to \mathbb{Z}/51\mathbb{Z}$ 映射，其中 $\Lambda(x) \equiv x$ (模 17)，且 $\Lambda(x) \equiv 1$ (模 3)，由此得到以下 A_0 模 51 的 CRL 列表：

$$\{(37, 8), (22, 8), (19, 1)\}.$$

现在，我们只需通过迭代枚举 f 上循环中的元素，然后在表6 中查找相应的有根树同构类型，就可以将 Γ_f 的四个连通分量的同构类型描述为有限有向根树的循环序列（项链，即项链图的同构类型）。我们指出，这是一种穷举法，当 f 周期点的数量很大时并不可行，此时应改用第 3.4 小节中描述的方法。

- 域元素 0 的连通分支是带有一个循环的单个顶点，对应于以下有根树同构类型的循环序列：$[\mathfrak{I}_0]$。

- 由于 $22 \in \mathbb{Z}/51\mathbb{Z}$ 在 A_0 下的循环为 (22, 46, 7,13,16,43,31,25)，域元素 $\iota_0(22) = \omega^{5 \cdot 22} = \omega^{110}$ 的连通分支由循环序列 $[\mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1]$ 表示。

- 由于在 A_0 作用下，$37 \in \mathbb{Z}/51\mathbb{Z}$ 的循环为(37,28,49,34,1,10,40,4)，域元素 ω^{185} 的连通分支由循环序列 $[\mathfrak{I}_3, \mathfrak{I}_1, \mathfrak{I}_4, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_1, \mathfrak{I}_2]$ 表示。

- 最后，由于 19 的循环只是 (19)，域元素 ω^{95} 所属的连通分支表示为 $[\mathfrak{I}_1]$。

作为快速的合理性检查，我们注意到 $|\mathrm{V}(\mathfrak{I}_1)| = 6$、$|\mathrm{V}(\mathfrak{I}_2)| = 23$、$|\mathrm{V}(\mathfrak{I}_3)| = 91$ 以及 $|\mathrm{V}(\mathfrak{I}_4)| = 57$，因此 $\mathbb{F}_{2^8}^*$ 中三个连通分量的顶点数总和为 $14 \cdot 6 + 23 + 91 + 57 = 255$，这正如预期。还可以验证，这些正是我们在引言中给出的相同循环序列。

4.3 特殊情形：所有 A_i 都是置换

设 f 是 $\mathbb{F}_q$ 的一个指标为 d 的广义分圆映射，以分圆形式 (1) 给出。假设对于每个 $i = 0$、1、.....、$d-1$，都有

$$\gcd(r_i, s) = \gcd(r_i, \frac{q-1}{d}) = 1.$$

适用于此情形的一类重要函数，是指标为 d 的 $\mathbb{F}_q$ 阶分圆映射（即所有 r_i 都等于 1 的广义分圆映射）。

根据我们在注释 1.3 和定义 1.4 之间的说明，用于编码在 $a_i \neq 0$ 情况下的限制 $f_{|C_i}$ 的仿射映射 A_i of$\mathbb{Z}/s\mathbb{Z}$ 具有 $z \mapsto r_i z + \text{const}$ 的形式。因此，我们对 r_i 的假设等价于要求，对于每个满足 $a_i \neq 0$ 的 i（因而 A_i 有定义），函数 A_i 是 $\mathbb{Z}/s\mathbb{Z}$ 的仿射置换。

我们的目标是描述函数图 Γ_f，结果发现这尤其简单。让我们从有根树开始。

引理 4.3.1. 令 $x \in \mathbb{F}_q = \mathrm{V}(\Gamma_f)$.

(1)如果 $x \neq 0$，并且如果 i 表示 $\{0, 1, \ldots, d-1\}$ 中满足 $x \in C_i$的唯一指标，那么树$\Gamma_f(x)$ 同构于 $\mathfrak{I}_i :=$ 树$\Gamma_{\overline{f}}(i)$。

(2) If $x = 0$, *then* $\mathrm{Tree}_{\Gamma_f}(x) = \sum_{i \in \overline{f}^{-1}(\{d\}) \setminus \{d\}} s\mathfrak{I}_i^+$.

证明。陈述（1）可通过对 $h(x) :=$ ht(Tree$^{\Gamma}{}_{\overline{f}}(i)$) 归纳法证明。如果 $h(x) = 0$，那么 i 的所有 $\overline{f}$-原像（如果存在）都是 $\overline{f}$-周期的。特别地，x 在 f 下没有 f-暂态原像，因为每个这样的原像都必须位于一个陪集 C_j 中，其中 j 是 i 在 $\overline{f}$ 下的一个 $\overline{f}$-暂态原像。事实上，否则，i 在 $\overline{f}$ 下有一个 $\overline{f}$-周期原像，因此其自身也是 $\overline{f}$-周期的。根据假设，我们可以在 $C_{i'}$ 中选取 x 在 f 下的一个 f-暂态原像 y，其中 i' 是 i 在 $\overline{f}$ 下唯一的 $\overline{f}$-周期原像。如果 ℓ 表示 i 在 $\overline{f}$ 下的周期长度，那么 $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$ 表示 f^ℓ 在 C_i 上的限制。由于每个 A_{i_t} 都是双射，$\mathcal{A}_i$ 也是双射；换言之，C_i 中的每个点在 $\mathcal{A}_i$ 下都是周期的，因而在 f 下也是周期的（根据第 3.1 小节中的讨论）。特别地，x 是 f-周期的，设其周期长度为 l。因此，$f^{l-1}(x)$ 是 x 在 $C_{i'}$ 中的一个 f-原像，y 也如此。由于 $A_{i'}$ 表示限制 $f_{|C_{i'}}$：$C_{i'} \to C_i$，且它是单射，所以可得 $f^{l-1}(x) = y$，由此 y 是 f-周期的，这与我们的假设矛盾。综上所述，如果 $h(x) = 0$，那么 Tree$^{\Gamma}{}_f(x)$ 是平凡的，$\mathfrak{I}_i$ 也如此。

现在我们假设 $h(x) \geq 1$。令 $j_1, j_2, \ldots, j_K$ 为 $\overline{f}$-暂态的 i 在 $\overline{f}$ 下的不同原像。根据上一段的论证， x 在 f 下的每个 f-暂态原像必定属于陪集 C_{j_t} 之一，其中 $t = 1$ 为 $2, \ldots, K$；又由于对于每个 t， A_{j_t} 都是双射，因此可知，对于每个 $t = 1$ 为 $2, \ldots, K$， x 恰有一个（暂态）原像 $c_{j_t} \in C_{j_t}$。因此，利用归纳法，

$$\mathrm{Tree}_{\Gamma_f}(x) = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_f}(c_{j_t})^+ = \sum_{t=1}^{K} \mathfrak{I}_{j_t}^+ = \sum_{t=1}^{K} \mathrm{Tree}_{\Gamma_{\overline{f}}}(j_t)^+ = \mathrm{Tree}_{\Gamma_{\overline{f}}}(i).$$

对于陈述（2），令 $j_1, j_2, \ldots, j_K$ 为 $\Gamma^*_{\overline{f}}$ 中 d 的不同 $\overline{f}$-暂态子节点。等价地， j_t 是 $\overline{f}^{-1}(\{d\}) \setminus \{d\}$ 中的不同元素。Γ^*_f 中 $0_{\mathbb{F}_q}$ 的 f-暂态子节点恰好是 $\bigcup_{t=1}^{K} C_{j_t}$ 中的点。根据陈述（1），可得

$$\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q}) = \sum_{t=1}^{K} \sum_{y \in C_{j_t}} \mathrm{Tree}_{\Gamma_f}(y)^+ \cong \sum_{t=1}^{K} \sum_{y \in C_{j_t}} \mathfrak{I}_{j_t}^+ = \sum_{t=1}^{K} s\mathfrak{I}_{j_t}^+,$$

如所要求。 □

由于对于 $x \neq 0$， $\mathrm{Tree}^{\Gamma_f}(x)$ 只取决于 x 所在的陪集 C_i，并且可以直接从 $\Gamma_{\overline{f}}$ 中读出，因此，为了理解 Γ_f 的同构类型，我们只需要了解 $\Gamma_{\overline{f}}$ 以及 f 在每个陪集并集 $U_i := \bigcup_{t=0}^{\ell-1} C_{i_t}$ 上的循环结构，其中 $(i_0, i_1, \ldots, i_{\ell-1})$ 且 $i = i_0$ 是 i 的 $\overline{f}$-循环。这可以使用与第 3.1 小节中确定 CRL 列表时类似的思想来实现。

令 $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$。那么， $\mathcal{A}_i$ 是 $\mathbb{Z}/s\mathbb{Z}$ 的仿射置换，其循环类型 $\mathrm{CT}(\mathcal{A}_i)$ 可从 [13, 表 3 和 4] 中读出。此外，

$$\mathrm{CT}(f_{|U_i}) = \mathrm{BU}_\ell(\mathrm{CT}(\mathcal{A}_i))$$

其中 BU_ℓ（所谓的 ℓ 膨胀函数）是 $\mathbb{Q}[x_n$ 的唯一 $\mathbb{Q}$-代数端同态： $n \in \mathbb{N}^+]$，满足对所有 $n \in \mathbb{N}^+$， $\mathrm{BU}_\ell(x_n) = x_{\ell n}$。称作

$$\mathrm{CT}(f_{|U_i}) = x_1^{e_1} x_2^{e_2} \cdots x_{\ell s}^{e_{\ell s}}.$$

然后，将函数图的同构类型视为有限有向根树的同构类型所组成的循环序列（项链）的多重集（其中每个这样的序- 列编码一个连通分支的同构类型），我们有如下结论：

$$\Gamma_f^{(i)} := \Gamma_{f|U_i} = \bigsqcup_{1 \leq l \leq \ell s, \ell | l} \bigsqcup_{n=1}^{e_l} \{\diamond_{m=1}^{l/\ell}[\mathfrak{I}_{i_0}, \mathfrak{I}_{i_1}, \ldots, \mathfrak{I}_{i_{\ell-1}}]\},$$

并且，如果 $\mathcal{L}$ 是 $\overline{f}$的 CRL 列表，那么

$$\Gamma_f = \bigsqcup_{(i,\ell) \in \mathcal{L}} \Gamma_f^{(i)} \sqcup \{[\sum_{j \in \overline{f}^{-1}(\{d\}) \setminus \{d\}} s\mathfrak{I}_j]\}.$$

5 算法复杂度分析

本节旨在详细描述用于理解有限域上广义分圆映射的函数图结构的重要方面的算法，并分析其复杂度。在小节 5.1 中，我们首先详细描述计算模型，即所谓的双重模型，并介绍一些重要的辅助概念和结果。我们注意到，该双重模型由分别仔细跟踪三个不同的参数组成——位运算、基本量子门，以及比特到量子比特的转换和反向转换——据作者所知，这是一种新颖的方法，对于从事量子复杂度分析的读者而言，也可能具有独立而广泛的意义。第 5.2 小节包含定理 5.1.9 的证明，该定理给出了三个基本算法问题的复杂度界，可视为本节的主要结果。正如引言中所述，如何紧凑地编码广义分圆映射的函数图整体结构仍是一个开放问题；特别是，这些结果并未提供一种高效的通用算法，用于判定两个给定广义分圆映射的函数图是否同构。然而，在 5.3 中，我们讨论了四种特殊情形，在这些情形下可以高效地解决这一同构问题。

5.1 框架与辅助结果

在本节中，我们始终假设 f 是 $\mathbb{F}_q$ 的一个指标为 d 的广义分圆映射，具有分圆形式 (1)，其中满足以下条件之一：

- 每个 a_i 都表示为域元素 0，或表示为 $\mathbb{F}_q$ 的某个公共未知本原元 ω 的幂；或者
- 我们明确知道这样的 ω在素子域 $\mathbb{F}_p$上的最小多项式 $P(T)$，并且 a_i被表示为 $\mathbb{F}_p[T]/(P(T))$的元素。

本节的主要目标是分析以下算法问题的复杂性。

- 问题1：给定 f，计算一个紧凑参数化的 CRL列表 $\mathcal{L}$ 的 f（我们注意到，$|\mathcal{L}|$ 等于 f 在其周期点上的循环数，该数关于$\log q$可能是超多项式的，因此我们希望避免逐元素列出 $\mathcal{L}$）。
- 问题 2：给定 f，计算 f 的分区树寄存器，其含义见下文定义 5.1.2。

- 问题 3：给定 f、 f的分区树寄存器，以及一对(r, l)，使得$r \in \mathbb{F}_q$如果 f-周期的，并且 l是 r在 f下的周期长度，计算公式 (17) 中的有根树同构类型的循环序列的紧凑描述（该公式刻画了包含 r 的 Γ_f 的连通分支的有向图同构类型）。

f 的分区树寄存器是一种标准化方式，用于存储第 3.3 小节中构造的算术划分 $\mathcal{P}_i$ 以及与其各块相关联的有根树的信息。为定义它，我们先引入以下辅助概念。

定义 5.1.1. 递归树描述列表是一个有限序列 $(\mathfrak{D}_n)_{n=0,1,\ldots,N}$，由若干集合构成，并与一个有序序列 $(\mathfrak{I}_n)_{n=0,1,\ldots,N}$ 相对应；该序列中的有限根树同构类型两两不同。它们满足以下条件。

(1) $\mathfrak{I}_0$ 是平凡有根树同构类型，并且 $\mathfrak{D}_0 = \emptyset$。

(2) 对于 $n \geq 1$，附着在 $\mathfrak{I}_n$ 的根上的 $\mathfrak{I}_n$ 中的每棵有根树都同构于$\mathfrak{I}_m$，其中 $m \in \{0, 1, \ldots, n-1\}$。此外，$\mathfrak{D}_n$ 是所有形如 (m, k_m) 的有序对的集合，其中 $m \in \{0, 1, \ldots, n-1\}$ 是一个指标，使得 $\mathfrak{I}_m$ 至少一次附着在$\mathfrak{I}_n$ 的根上，而 k_m 是其附着的重数。

In a recursive tree description list, each set $\mathfrak{D}_n$ can be viewed as a compact description of $\mathfrak{I}_n$, referring to the rooted trees attached to the root of $\mathfrak{I}_n$ with their (earlier) indices m, rather than their full descriptions. The idea of encoding isomorphism types of rooted trees via numbers ("tree indices") to get more compact descriptions of larger rooted trees is not new; it appears, for example, in the decision algorithm for isomorphism of directed rooted trees described in [5, Example 3.2 on p. 84]. In contrast to that algorithm, which is linear in the number of vertices, we do not list tree indices m repeatedly, but rather, we specify their multiplicities k_m. In situations such as ours, where entire sets (here: arithmetic partition blocks) of vertices can be dealt with simultaneously, this modification is crucial to ensure the efficiency of our algorithms relative to their smaller input length (which lies in $O(d \log q)$). In implementations, we assume that each $\mathfrak{D}_n$ is represented by an array (ordered list) of pairs (m, k_m), sorted by increasing m. Moreover, m and k_m, both of which are at most q, are to be represented by bit strings of length $\lfloor \log_2 q \rfloor + 1$ (please note, however, that we use other conventions for the related notion of a type-I tree register, introduced in Definition 5.3.2.1(1)). We observe that with these conventions, all bit strings representing an element (m, k_m) of $\mathfrak{D}_n$ (for some n) have the same length, and the ordering of the elements of $\mathfrak{D}_n$ by increasing m corresponds to the lexicographic ordering of those bit string encodings.

借助递归树描述列表这一概念，我们可以使用第 3.3小节中引入的记号，将有限域的广义分圆映射的划分树寄存器定义如下。我们注意到，在这一算法部分中，我们经常将算术划分与其特定的张成同余序列视为同一对象。

定义 5.1.2. 设 f 是 $\mathbb{F}_q$的一个指标 d 广义分圆映射。如果$i \in \{0, 1, \ldots, d-1\}$是 $\overline{f}$-周期的，我们回顾一下，对于 $t \in \mathbb{Z}$，i_t表示 $\{0, 1, \ldots, d-1\}$中唯一的$\overline{f}$-周期指标，使得 $(\overline{f}_{\restriction \mathrm{per}(\overline{f})})^t(i) = i_t$。$f$的一个分区树寄存器是形如以下内容的有序对。

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n=0,1,\ldots,N})$$

使得以下条件成立。

(1) 对于每个 $i = 0, 1, \ldots, d-1$, $\mathcal{Z}_i$，情况如下。(a) 如果 i 是 $\overline{f}$-暂态，那么 $\mathcal{Z}_i = \mathcal{P}_i$ 由长度为 $m_i \in \mathbb{N}_0$ 的生成同余序列给出。

(b) 如果 i 是 $\overline{f}$-周期的，那么 $\mathcal{Z}_i$ 是一个 (H_i+2)-元组 $(\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$，满足 i. $\mathcal{X}_{i,-1}=(\theta_{i,h}(x))_{h=1,2,\ldots,H_i}$，以及 ii. $\mathcal{X}_{i,h}$ 对于 $h=0,1,\ldots,H_i$ 是算术划分 $\lambda^h_{i_{-h}}(\mathcal{R}_{i_{-h}})$ 的（生成同余序列），其长度为 $n_{i_{-h}}\in\mathbb{N}_0$。

(2) 序列 $(\mathfrak{D}_n)_{n=0,1,\ldots,N}$ 是一个递归树描述列表，其对应的有根树同构类型序列为 $(\mathfrak{I}_n)_{n=0,1,\ldots,N}$，使得 $\mathfrak{I}_n$ 恰好是具有以下某种形式的有根树同构类型

(a)树$_i(\mathcal{P}_i,\vec{\nu}^{(\mathcal{P}_i)})$对某个 $\overline{f}$-暂态 i 和某个 $\vec{\nu}^{(\mathcal{P}_i)}\in\{\emptyset,\neg\}^{m_i}$ 使得$\mathcal{B}(\mathcal{P}_i,\vec{\nu}^{(\mathcal{P}_i)})\neq\emptyset$;
(b) 树$\Gamma_f(0_{\mathbb{F}_q})$；或 (c) 树$_i^{(h)}(\mathcal{P}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})})$ 对某个 $\overline{f}$-周期的 $i\neq d$、某个 $h\in\{0,1,\ldots,H_i\}$和某个 $\vec{\nu}^{(\mathcal{P}_{i,h})}\in\{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}$使得 $\mathcal{B}(\mathcal{Q}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})}\diamond\vec{\xi}_{i,h})$，即 $\mathcal{B}(\mathcal{P}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})})$ 中所有 $\mathfrak{h}$-值为 h 的点的集合，非空。

(3)对象$S_{n,i}$ 满足以下条件。

(a) 如果 i 是 $\overline{f}$-暂态，则

$$S_{n,i}=\{\vec{\nu}^{(\mathcal{P}_i)}\in\{\emptyset,\neg\}^{m_i}:\mathcal{B}(\mathcal{P}_i,\vec{\nu}^{(\mathcal{P}_i)})\neq\emptyset\ and\ \mathrm{Tree}_i(\mathcal{P}_i,\vec{\nu}^{(\mathcal{P}_i)})\cong\mathfrak{I}_n\}.$$

(b) 如果 $i=d$，那么 $S_{n,i}=S_{n,d}\in\{\emptyset,\neg\}$ 是与同构关系树$\Gamma_f(0_{\mathbb{F}_q})\cong\mathfrak{I}_n$的真值相关联的逻辑符号。
(c) 若 $i\neq d$ 是 $\overline{f}$-周期的，则 $S_{n,i}=(S_{n,i,h})_{h=0,1,\ldots,H_i}$ 其中

$$S_{n,i,h}=\{\vec{\nu}^{(\mathcal{P}_{i,h})}\in\{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}:\\ \mathcal{B}(\mathcal{Q}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})}\diamond\vec{\xi}_{i,h})\neq\emptyset\ and\ \mathrm{Tree}_i^{(h)}(\mathcal{P}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})})\cong\mathfrak{I}_n\}.$$

在实现中，我们假设每个由 $\overline{f}$-暂态 i 构成的集合 $S_{n,i}$，以及每个由 $\overline{f}$-周期的 $i<d$ 和 $h\in\{0,1,\ldots,H_i\}$ 构成的集合 $S_{n,i,h}$，都表示为按字典序排列的比特串数组，其中比特 0 表示 $\emptyset$，比特 1 表示 $\neg$。我们注意到，尽管 f 的分区树寄存器没有明确列出 $\overline{f}$-周期索引 $i<d$ 对应的划分 $\mathcal{P}_i=\mathcal{Q}_{i,H_i}$，但很容易从寄存器中读出它们的生成同余序列及关联的有根树。具体而言，

- $\mathcal{Z}_i$ spans $\mathcal{P}_i$ 中同余序列的拼接；并且
- 根据命题 3.3.5，与 $\mathcal{P}_i$ 的一个块 $\mathcal{B}(\mathcal{P}_i,\vec{\nu}^{(\mathcal{P}_i)})$ 相关联的有根树具有如下形式：$\mathsf{Tree}^{(h)}_{\mathcal{P}_i\,h\ \vec{\nu}^{\,\mathcal{P}_i\,h}}$ $h\in\{0,1,\ldots,H_i\}$ $\vec{\nu}^{\,\mathcal{P}_i\,h}\in\{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n^{i}_{i_{-h}}}$，其中参数 h $\vec{\nu}^{\,\mathcal{P}_i\,h}$ $(,)$ 取适当值。相关参数以及 $(,)$ 可从逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_i)}$ 中读出，该元组刻画了 $\mathcal{P}_i$ 的块。

在开始对问题 1–3 进行实际的复杂度分析之前，我们先作一些说明，首先讨论我们的计算模型。

正如第2.4小节中已暗示的那样，为了让我们的算法有望实现多项式运行时间，至少在某些子任务中我们需要使用量子计算机，例如需要计算模乘法阶或离散对数时。话虽如此，我们只会将某些

定义明确且已有高效量子算法的任务交由量子计算机处理，而我们的其余算法则可以在经典计算机上执行。因此，我们采用以下两种计算模型：

- 一种针对经典计算机上尚无已知高效算法的任务（例如整数因数分解或离散对数计算）的位操作模型，该模型支持多种查询。在此模型中（下文称为查询模型，在该模型中运行的算法称为查询算法），复杂度以一个元组衡量，该元组记录查询之外所用的位运算数量，以及每种查询的调用次数；

- 一种经典计算机和量子计算机协同使用、并可相互“传递”输出的模型；我们将此类算法称为双重算法，将这种模型称为双重模型。

对于双重模型中计算的量子部分，我们具体采用量子电路模型，因此，凡是提到量子复杂度，均指量子（基本）门复杂度。我们使用的所有量子算法都基于 Shor 的奠基性论文 [63], ，并且都是拉斯维加斯算法，即其在给定输入上的运行时间会随机变化；所指定的位操作成本、门复杂度以及比特与量子比特之间的转换次数，均应理解为期望值。这也意味着，我们的所有双重算法整体上都是拉斯维加斯算法，其所指定复杂度的所有组成部分也都只是期望值。

另一方面，对于任一模型中计算的“经典”部分，我们采用一种基于随机存取内存的位操作模型，其方式类似于 [5,第 1 节2], : 如果 N 是需要访问的内存寄存器的地址（指标）的位长度，则内存访问需要 $O(N)$ 次位运算。例如，访问变量 y_k 的存储值（其中 k 是一个非负整数）需要$O(\log k)$ 次位运算——整个内存地址由字母“y”的位编码（假定其长度为 $O(1)$）与 k的标准二进制表示拼接而成。因此，我们假定，在地址扫描完毕后，“跳转”到内存中的某个位置是免费的。除了通过读取地址来访问内存寄存器外，我们还假定可以通过放置指针来保存寄存器内的某些位置（指针数量有限，但我们不指定其具体上限），从而能够以 $O(1)$ 次位运算的代价跳回内存中的特定位置（位）。此外，我们假定移动到内存中的相邻位置（包括移动到数组的下一项）需要 $O(1)$ 次位运算。我们将复杂度中的经典部分称为经典复杂度，或（同义地）位运算。

众所周知（例如参见 [75, 第 IV.3] 小节），每个经典电路都有一个等价的量子电路，其中基本门的数量至多增加一个常数因子。基于这一点，似乎可以直接在双重模型中对两类计算都使用电路，这样经典部分便可以并入量子部分（而不改变门复杂度的 Landau O 类），而且这似乎是量子计算中通常采用的方法，-

复杂度分析。例如，在 [35, 第 7.3节和第 7.4], 节中，特定量子算法（即不涉及黑盒群中的操作的算法）的复杂度分析仅包括对所涉及的量子门数量进行计数，而预处理和后处理的位操作（或经典门）成本则被忽略。对于 [35, 第 7.3节和第 7.4], 节中的算法，这样做完全没有问题，因为无论测量使用位操作还是经典门，该经典成本都是量子门数量的big-O 。然而，我们的算法所涉及的经典成本确实显著高于量子门，这一点从引理5.1.6的复杂度界中已经可以看出；我们注意到，尽管这些本质上是来自[35, 第7.3节和第 7.4], 节的算法，但如果希望它们成为拉斯维加斯算法（由于使用了AKS素性测试），它们最终确实会产生相对较大的经典成本。因此，分别追踪经典成本（无论是位操作还是经典门）和量子门的成本似乎是很自然的，尤其是因为目前尚不清楚量子计算机的大规模物理实现中每个量子门的实际时间成本。

至于我们为何在计算的经典部分使用位运算（而不是门），我们注意到，用于解决问题 2 和问题 3 的算法涉及大量“记账”工作，也就是内存访问；而在两种电路模型中，内存访问通常代价高昂。事实上，假设我们希望在经典电路模型中访问先前计算得到的变量 $y_k \in \{0,1\}$ 的值，其中指标 $k \in \{1,\ldots,N\}$ 也是先前一次计算的结果。构建电路时，我们无法预先知道相关的 N 根导线中哪一根携带所需信息，因此需要通过一个子电路来处理，其输入包括这些 N 根导线，以及承载 k（的二进制表示）的 $O(\log N)$ 根导线。但每个基本门只接受 $O(1)$ 个输入位，因此，执行内存访问的这个子电路至少必须由 cN 个基本门组成，其中 $c > 0$ 为某个常数；相比之下，在我们选用的位操作模型中，类似问题的内存访问代价为 $O(\log N)$。

当双重算法的经典部分与量子部分之间发生通信时，经典比特串 $\vec{x} \in \{0,1\}^N$ 需要转换为相应的量子比特寄存器 $|\vec{x}\rangle$ ，反之亦然。在我们处理 $\mathbb{F}_q$的广义分圆映射的算法中，每次此类转换的位长度 N 均为 $O(\log q)$。由于此类转换的成本尚不明确，因此我们有必要单独统计转换次数（两个转换方向合计），并将其称为相应双重算法的转换复杂度。将转换后的信息复制到下一个经典计算机或量子电路，以及在量子电路末尾进行的测量，均视为相应转换过程的一部分，其成本不单独统计。为统一标准，我们假设双重算法的原始输入和最终输出均为经典比特串。特别地，将纯量子算法视为双重算法时，除量子门数量外，还包含两次转换（分别在算法开始和结束时进行）。

我们也来谈谈 Grover 针对无结构数据库的量子算法。
s从 [25] 搜索。该算法因能提供二次加速而闻名。

经典线性搜索算法，并且鉴于我们的算法中前述的大量记账，使用它似乎是很自然的。然而，这里有一些需要考虑的细微之处，最终促使作者决定不在我们的分析中纳入Grover 算法。Grover算法通常的复杂度分析假设，要查找的列表（或者更确切地说，与我们想要在列表中找到的信息相关联的特征函数 χ ）被给定为某个酉算子 U_χ，称为相位反转算子；该算子随后被用作算法的量子电路的一部分，并被视为预言机。对于长度为 N 的列表进行查找时，著名的 Grover 复杂度 $O(\sqrt{N})$ 指的是在最终测量之前， U_χ （以及另一个所谓的相移算子）被应用的次数。然而，我们关注的是门复杂度，因此需要将 U_χ 的门复杂度作为额外因数纳入。现在， χ 可以是任意函数 $\{0,1,\ldots,N-1\}\to\{0,1\}$（通常使用的假设，即 $\chi(j)=1$对于唯一的指标 j 成立，并不适用于我们的情况），我们也可以将其视为 $n=\lfloor\log_2 N\rfloor$ 个变量上的（部分）布尔函数。这意味着，为了使 Grover算法的量子（门）复杂度“击败”线性搜索的位操作复杂度， n变量布尔函数的最坏情况量子门复杂度需要为 $o(2^{n/2})$，而这是否成立尚不明确。我们确实注意到，已知 n变量布尔函数的最坏情况经典门复杂度的量级为 $2^n/n$ （Shannon， [62, 第76页及后续页上的定理 6和7]），并且它与 n变量布尔函数的最坏情况量子门复杂度仅相差某个幂（Beals 等人， [10]）。

我们注意到，在双重模型中对量子算法的处理是理想化的，因为我们忽略了硬件故障和量子噪声导致错误的可能性。与许多作者一样，我们这样做是基于著名的量子阈值定理。该定理的要旨是，一旦将每个基本门的故障率降至某个固定阈值以下，就能以相较于理想化算法几乎不增加多少成本为代价，构造出鲁棒性任意高的量子算法。该定理最早见于 Shor 的一篇论文 [64], ，不过其中陈述的版本弱于如今所称的量子阈值定理。更强版本的若干变体（取决于所用的错误模型）分别由 Aharonov 和 Ben-Or [3], 、Knill、Laflamme 和 Zurek [37], 以及 Kitaev [36], 独立证明。综述 [24], 将该定理列为定理 10，并给出了统一证明。

前面的讨论引出了以下定义，这些算法复杂度概念将用于本节的结果。

定义 5.1.3. 我们引入以下概念和记号。

(1) 我们用 $\{0,1\}^{<\infty}$ 表示所有有限比特串组成的集合。形式上，

$$\{0,1\}^{<\infty}=\bigcup_{n\in\mathbb{N}_0}\{0,1\}^n.$$

(2) 算法问题是一个函数 $\mathfrak{L}$ ，其定义域为 $\mathfrak{L}_{\text{in}}$ 的子集 $\{0,1\}^{<\infty}$，并将每个比特串 $\vec{\mathfrak{r}}\in\mathfrak{L}_{\text{in}}$ 映射到某个非空有限子集 $\mathfrak{L}(\vec{\mathfrak{r}})\subseteq\{0,1\}^{<\infty}$。

(3) 在陈述 (2) 的情形中， $\mathfrak{L}_{\text{in}}$ 的元素称为 $\mathfrak{L}$ 的可容许输入；对于每个 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$， $\mathfrak{L}(\vec{\mathfrak{x}})$ 的元素称为 $\mathfrak{x}$ 的可接受输出（相对于 $\mathfrak{L}$）。

（4）设 $\mathfrak{L}$ 是一个算法问题， y、y_1、y_2、…、y_n 是与 $\mathfrak{L}$ 的可容许输入相关联的非负实参数；形式上， y 和 y_j 是函数 $\mathfrak{L}_{\text{in}} \to [0,\infty)$。

(a) 一个元组 $\vec{\mathcal{C}}^{(\text{qry})} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{fdl}}, \mathcal{C}_{\text{mdl}}, \mathcal{C}_{\text{mord}}, \mathcal{C}_{\text{prt}})$，其每个条目都是一个函数 $[0,\infty)^n \to [0,\infty)$，称为 $\mathfrak{L}$（相对于 $y_1, \ldots, y_n$）的 y-有界查询复杂度，如果存在一种查询算法，对每个输入 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$，都能通过使用，为 $\vec{\mathfrak{x}}$ 生成一个可接受的输出。

- $O(\mathcal{C}_{\text{类}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 除下列查询中列出的位运算外；
- $O(\mathcal{C}_{\text{fdl}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 次查询，用于在大小不超过 $y(\vec{\mathfrak{x}})$ 的有限域中计算离散对数；
- $O(\mathcal{C}\text{mdl}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 次查询，用于针对给定的 $x, z \in (\mathbb{Z}/m\mathbb{Z})^*$，其中 $m < y(\vec{\mathfrak{x}})2$，计算模离散对数 $\log(k)_x(z)$，其中 $k \in \{m, p^{\nu_p}(m) : p \mid m\}$，并输出一个列表，其中包含数对 $(m, \log(m)_x(z))$，以及对所有素数 $p \mid m$，包含四元组 $(p, \nu_p(m)\ p^{\nu_p}(m), \log(p^{\nu_p}(m))_x(z))$；
- $O(\mathcal{C}_{\text{mord}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 次查询，用于针对给定的单位元 $x \in (\mathbb{Z}/m\mathbb{Z})^*$，其中 $m < y(\vec{\mathfrak{x}})_2$，计算乘法阶 $\text{ord}_k(x)$，其中 $k \in \{m, p^{\nu_p(m)} : p \mid m\}$，并输出一个列表，其中包含数对 $(m, \text{ord}_m(x))$，以及对所有素数 $p \mid m$，包含四元组 $(p, \nu_p(m)\ p^{\nu_p(m)}, \text{ord}_{p^{\nu_p(m)}}(x))$；
- $O(\mathcal{C}_{\text{prt}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 次查询，以找出模每个奇素数幂因数 $p^{\nu_p(m)}$ 的原根 $\mathfrak{r}_{(p)}$，其中某个整数 $m < y(\vec{\mathfrak{x}})$ 满足 > 1，并输出相应的四元组列表 $(p, \nu_p(m)\ p^{\nu_p(m)}\ \mathfrak{r}_{(p)})$。

(b) $\mathfrak{L}$ 的 y 有界拉斯维加斯双重复杂度是一个三元组

$$\vec{\mathcal{C}}^{(\text{LV})} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{quant}}, \mathcal{C}_{\text{conv}})$$

其中每个条目都是一个函数 $[0,\infty)^n \to [0,\infty)$，并且存在一个（理想化的）双重算法，对每个输入 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$，都能在期望次数后终止。

- $O(\mathcal{C}_{\text{类}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 位运算，
- $O(\mathcal{C}_{\text{quant}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 基本量子门，以及
- $O(\mathcal{C}_{\text{转换}}(y_1(\vec{\mathfrak{x}})\ y_2(\vec{\mathfrak{x}}) \ldots, y_n(\vec{\mathfrak{x}})))$ 长度在 $O(\log y(\vec{\mathfrak{x}}))$ 中的比特串转换为量子比特寄存器，以及将长度为 $O(\log\ y(\vec{\mathfrak{x}}))$ 的量子比特寄存器转换为比特串，

为 $\mathfrak{x}$ 生成符合要求的输出。

在我们的算法中，定义 5.1.3 中的 $y(\vec{\mathfrak{x}})$ 的值始终等于相应的域大小 q。虽然我们的查询模型定义没有明确包含整数分解查询，但这类查询已涵盖在模离散

对数查询或乘法阶查询中。事实上，要对 $m \in \mathbb{N}^+$ 进行因数分解，其中 $m < y(\vec{\mathfrak{x}})^2$, , 只需查询模 m 下 x 的乘法阶：$= 1 \in (\mathbb{Z}/m\mathbb{Z})^*$。所得输出由满足 $k \in \{m, p^{\nu_p(m)} : p \mid m\}$ 的数对 $(k, 1)$ 构成，从中可以轻松读出 m 的素因数分解。同样，也可以进行模离散对数查询，参数为 $x := y := 1$。

根据定义 5.1.3(2) 的假设， $\mathfrak{L}$ 的每个可接受输入只能对应有限个可接受输出；对于我们的分析而言，这并不失一般性。该假设简化了下面引理 5.1.4 的表述。引理 5.1.4 很容易证明，其基本内容是：查询复杂度关于算法问题的复合具有可加性。算法问题的复合定义如下。若 $\mathfrak{L}$ 和 $\mathfrak{L}'$ 是算法问题，且对每个 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$ 都有 $\mathfrak{L}(\vec{\mathfrak{x}}) \subseteq \mathfrak{L}'_{\text{in}}$，那么 $\mathfrak{L}$ 和 $\mathfrak{L}'$ 的复合（记作 $\mathfrak{L}\mathfrak{L}'$ 或 $\mathfrak{L}' \circ \mathfrak{L}$）是输入集为 $\mathfrak{L}_{\text{in}}$ 的算法问题，其定义为 $(\mathfrak{L}' \circ \mathfrak{L})(\vec{\mathfrak{x}}) := \mathfrak{L}'(\mathfrak{L}(\vec{\mathfrak{x}}))$（集合 $\mathfrak{L}(\vec{\mathfrak{x}})$ 在函数 $\mathfrak{L}'$ 下的逐元素像）。

引理 5.1.4。设 $\mathfrak{L}$ 和 $\mathfrak{L}'$ 是算法问题，使得 $\mathfrak{L}(\vec{\mathfrak{x}}) \subseteq \mathfrak{L}'_{\text{in}}$ 对每个 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$ 均成立，并设 y、y_1、y_2、……、y_n 分别为与 $\mathfrak{L}$ 的可容许输入相关联的非负实参数，y'、y'_1、y'_2、……、$y'_{n'}$ 分别为与 $\mathfrak{L}'$ 的可容许输入相关联的非负实参数。对于 $j' = 1$、2、……、n'，我们定义

$$y_{n+j'} : \mathfrak{L}_{\text{in}} \to [0, \infty),$$
$$\vec{\mathfrak{x}} \mapsto \max\{y'_{j'}(\vec{\mathfrak{y}}) : \vec{\mathfrak{y}} \in \mathfrak{L}(\vec{\mathfrak{x}})\}.$$

由于复合映射 $\mathfrak{L}\mathfrak{L}'$ 的输入集为 $\mathfrak{L}_{\text{in}}$，我们可以将每个 y_j（对于 $1 \le j \le n+n'$）视为 $\mathfrak{L}\mathfrak{L}'$ 的参数。此外，我们定义

$$y^+ : \mathfrak{L}_{\text{in}} \to [0, \infty),$$
$$\vec{\mathfrak{x}} \mapsto \max\{y(\vec{\mathfrak{x}}), y'(\vec{\mathfrak{y}}) : \vec{\mathfrak{y}} \in \mathfrak{L}(\vec{\mathfrak{x}})\}.$$

最后，我们令 $\vec{\mathcal{C}}^{(\text{qry})}$ 分别为关于 $y_1, \ldots, y_n$ 的一个 y-有界查询复杂度，$\vec{\mathcal{C}'}^{(\text{qry})}$ 分别为关于 $y'_1, \ldots, y'_{n'}$ 的一个 y'-有界查询复杂度。对于 $k = 1, \ldots, 5$，我们将 $\mathcal{C}_k$ 分别记为 $\vec{\mathcal{C}}^{(\text{qry})}$ 的 k 个条目，将 $\mathcal{C}'_k$ 分别记为 $\vec{\mathcal{C}'}^{(\text{qry})}$ 的 k 个条目。然后，定义

$$\mathcal{C}^+_k : [0, \infty)^{n+n'} \to [0, \infty),$$
$$(z_1, \ldots, z_{n+n'}) \mapsto \mathcal{C}_k(z_1, \ldots, z_n) + \mathcal{C}'_k(z_{n+1}, \ldots, z_{n+n'}),$$

元组 $\vec{\mathcal{C}^+}^{(\text{qry})} := (\mathcal{C}^+_1、\ldots、\mathcal{C}^+_5)$ 是关于 $y_1、\ldots、y_{n+n'}$ 的 $\mathfrak{L}\mathfrak{L}'$ 的 y^+-有界查询复杂度。

另一方面，对于每个给定的算法问题 $\mathfrak{L}$ 以及与 $\mathfrak{L}$ 的可容许输入相关联的非负实参数 y，只要 y 可以根据其他参数 y_j 进行界定，就可以从 $\mathfrak{L}$ 的 y-有界查询复杂度推导出 $\mathfrak{L}$ 的 y-有界拉斯维加斯双重复杂度；见下文引理 5.1.7。

通常，在具体情况下说明复杂度时，我们将函数与其定义项等同起来。例如，如果 d 和 q 是相关参数

与我们的输入相关，我们可以将一种受 q 限制的拉斯维加斯双重复杂度指定为

$$(d^3 \log q, d \log^{2+o(1)} q, \log^{1+o(1)} q),$$

而不是为三个分量中的函数命名。在此背景下，我们还要指出，$\log^k x$ 始终表示算术幂 $(\log x)^k$，而非 log 的 k 次迭代在 x 处的函数值。我们始终将迭代对数写全（log log、log log log 等）。

以下引理将在本节中反复使用，它给出了一些基本算法问题的复杂度。

引理 5.1.5。以下结论成立。

(1) Addition and subtraction of integers of absolute value less than m, as well as addition and subtraction modulo m cost $O(\log m)$ bit operations each.

(2) Addition and subtraction in the finite field $\mathbb{F}_q$ cost $O(\log q)$ bit operations each.

(3) Multiplication of positive integers less than m, multiplication modulo m and division of positive integers less than m with remainder each cost $O(\log^{1+o(1)} m)$ bit operations.

(4) Let $x \in (\mathbb{Z}/m\mathbb{Z})^*$. The computation of $\mathrm{inv}_m(x)$, the multiplicative inverse of x modulo m, costs $O(\log^{1+o(1)} m)$ bit operations.

(5) Multiplication, multiplicative inversion and division in the finite field $\mathbb{F}_q$ each cost $O(\log^{1+o(1)} q)$ bit operations.

(6) Let $x \in \mathbb{Z}/m\mathbb{Z}$ and $e \in \mathbb{N}_0$. The computation of the power x^e modulo m costs $O(\log(e) \log^{1+o(1)} m)$ bit operations.

(7) Let $x \in \mathbb{F}_q$ and $e \in \mathbb{N}_0$. The computation of x^e costs $O(\log(e) \log^{1+o(1)} q)$ bit operations.

(8) The computations of the gcd *and* lcm *of two positive integers that are at most m cost $O(\log^{1+o(1)} m)$ bit operations each.*

(9) Checking deterministically whether a given positive integer m is a prime costs $O(\log^{6+o(1)} m)$ bit operations.

(10) An array of n bit strings, each of length k, can be lexicographically sorted within $O(kn \log n)$ bit operations.

(11) We assume given two lexicographically sorted arrays of bit strings (not necessarily all of the same length) and consider the algorithmic problem of finding the lexicographically sorted version of their concatenation (i.e., the problem of merging *those sorted arrays). For $j = 1, 2$, say the j-th array has N_j entries, and the sum of the bit lengths of the strings stored in it is $l^{(j)}_{\mathrm{total}}$. Then those two sorted arrays can be merged within $O(N_1 + N_2 + l^{(1)}_{\mathrm{total}} + l^{(2)}_{\mathrm{total}})$ bit operations (and thus within $O(l^{(1)}_{\mathrm{total}} + l^{(2)}_{\mathrm{total}})$ bit operations if all bit strings in question are non-empty).*

证明。对于陈述（1），众所周知（而且很容易验证），使用加法与减法的传统算法会产生所指定的复杂度。

对于陈述（2），令 $q = p^m$。我们参考第 84], 页的表 2.8，并注意到，在 $\mathbb{F}_q$ 中进行加法/减法时，涉及的唯一（$\mathbb{Z}/p\mathbb{Z}$）操作是模加法与模减法。因此，陈述（1）意味着，在 $\mathbb{F}_q$ 中加法与减法的代价为 $O(m\log\ p) = O(\log\ q)$，如所要求的那样。

对于陈述（3），根据 Schönhage-Strassen 算法 [60] 或 Harvey 和 van der Hoeven 提出的（稍快的）算法 [28]，两个小于 m 的正整数相乘需要 $O(\log^{1+o(1)} m)$ 位操作。此外，若使用牛顿-拉夫森算法进行整数带余除法，其代价也为 $O(\log^{1+o(1)} m)$ 位操作；参见第 1.3] 小节。模 m 乘法可以通过先对相关的两个整数进行（非模）乘法（所得结果为一个含有 $O(\log\left(m^2\right)) = O(\log\ m)$ 位的数），再进行模约简（这是带余除法的一部分）来实现。因此，模 m 乘法总共也只需要 $O(\log^{1+o(1)} m)$ 位操作。

对于陈述（4），我们注意到，模 m 求逆可以通过扩展欧几里得算法完成；采用 Sch¨onhage 提出的加速变体时（该变体基于 Knuth 早期的思想），需要 $O(\log^{1+o(1)} m)$ 次位运算 [59]；另请参阅 [74], ，其中对这一算法作了推广，而且由于是用英语撰写的，可能更容易理解。

对于陈述 (5)，我们注意到 $\mathbb{F}_q$ 给定为 $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$，其中 $P(T)$ 是一个次数为 $m := \log_p q$ 的首一本原不可约多项式。为了将 $\mathbb{F}_q$ 中的两个元素 $Q_1(T) + (P(T))$ 和 $Q_2(T) + (P(T))$ 相乘，需要计算多项式乘积 $Q_1(T)Q_2(T) \in (\mathbb{Z}/p\mathbb{Z})[T]$，然后求其除以 $P(T)$ 的余数。正如第 2], 节第二段 [23, 中所述，对于次数至多为 n 的 $\mathbb{Z}/p\mathbb{Z}$ 上的多项式，其乘法以及带余除法的快速方法需要进行 $O(n^{1+o(1)})$ 次运算（加法、减法、乘法、乘法求逆）。根据陈述 (1) 和 (3)，这相当于 $\mathbb{Z}/p\mathbb{Z}$ 上 $O(n^{1+o(1)}\log^{1+o(1)} p)$ 的位操作成本。因此，计算 $Q_1(T)Q_2(T)$，以及随后计算其模 $P(T)$ 的余数，都需要 $O(m^{1+o(1)}\log^{1+o(1)} p) = O(\log^{1+o(1)} q)$ 次位运算，这正是 $\mathbb{F}_q$ 中乘法满足所述复杂度界所需的。

现在，由于 $\mathbb{F}_q$ 中的除法由一次乘法逆元运算后接一次乘法组成，因此只需说明 $\mathbb{F}_q$ 中乘法逆元的位操作成本为 $O(\log^{1+o(1)} q)$，即可完成该命题的证明。假设 $P(T) \nmid Q(T)$，则可通过将 $1 = \gcd(P(T)\ Q(T))$ 表示为 $P(T)$ 与 $Q(T)$ 的 $(\mathbb{Z}/p\mathbb{Z})[T]$-线性组合，并将该线性组合中 $Q(T)$ 的标量对 $P(T)$ 取模，来计算 $Q(T)$ 模 $P(T)$ 的乘法逆元。 [5, 第 8.9] 节中描述的算法使用 $O(\log^{1+o(1)}(p^m) \cdot \log m) = O(\log^{1+o(1)} q)$ 次位操作来计算 $\gcd(P(T)\ Q(T))$（根据 [5, 定理 8.19]）。在此过程中，可以存储 HGCD 过程从 [5, 第 8.7 图第 304] 页中进行的 $K \in O(\log m)$ 次调用所输出的 (2×2)-矩阵 R_1、R_2、.....、R_K（其系数属于 $(\mathbb{Z}/p\mathbb{Z})[T]$），这些矩阵按列出的顺序输出。设 $P_0(T) = P(T)\ P_1(T) = Q(T)\ P_2(T)$ .....、$P_N(T) = \gcd(P(T)\ Q(T)) = 1$ 为将欧几里得算法的经典 “ 缓慢 ” 版本应用于 $(P(T)\ Q(T))$ 时出现的连续余数，并且对于 $t \in \{0, 1, \ldots, N -$

1}，令 $Q_t(T)$ 为 $P_t(T)$ 除以 $P_{t+1}(T)$所得的多项式除法的商群，满足 $\deg Q_t(T) = \deg P_t(T) - \deg P_{t+1}(T)$。根据第 303 页关于 HGCD 输出形式为 R_{0j} 的 [5, 陈述, 以及第 302], 页例 8.10 之前的 R_{ij} 定义，每个矩阵 R_j 都是以下形式矩阵的乘积。

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q_t(T) \end{pmatrix}$$

对于两两不同的 t。因此， $R_j(T)$ 的每个条目都是关于 $(\mathbb{Z}/p\mathbb{Z})[T]$ 的次数至多为的多项式

$$\sum_{t=0}^{N-1} \deg Q_t(T) = \sum_{t=0}^{N-1} (\deg P_t(T) - \deg P_{t+1}(T)) = \deg P(T) - \deg 1 = \deg P(T).$$

此外，由 [5, 引理 8.5(a)], 可知，对每个 $k \in \{1, 2, \ldots, K\}$，均有

$$R_k R_{k-1} \cdots R_1 \cdot \begin{pmatrix} P(T) \\ Q(T) \end{pmatrix} = \begin{pmatrix} P_j(T) \\ P_{j+1}(T) \end{pmatrix}$$

对某些 $j = j(k) \in \{0, 1, \ldots, N-1\}$，具体而言

$$R_K R_{K-1} \cdots R_1 \cdot \begin{pmatrix} P(T) \\ Q(T) \end{pmatrix} = \begin{pmatrix} P_{N-1}(T) \\ P_N(T) \end{pmatrix} = \begin{pmatrix} P_{N-1}(T) \\ 1 \end{pmatrix}.$$

这个后一个等式给出了 1作为 $P(T)$和$Q(T)$ 的线性组合的表示，其中， $Q(T)$ 的（模 $P(T)$约化后的）标量等于(2×2)-矩阵$R_K R_{K-1} \cdots R_1$的右下系数的（模 $P(T)$ 约化后的）值。计算该矩阵乘积需要 $O(K) \subseteq O(\log m)$次加法、乘法和带余数的除法（在 $(\mathbb{Z}/p\mathbb{Z})[T]$ 中），这也对应于 $O(\log\ m \cdot \log^{1+o(1)} p^m) = O(\log^{1+o(1)} q)$ 的位操作成本，正如所要求的那样。

对于陈述（6），我们注意到，使用“平方乘”，幂 x^e模 m 可以用 $O(\log e)$ 次乘法模 m计算，因此陈述（3）得出该结论。

对于陈述（7），其证明与陈述（6）的证明类似，但用陈述（5）代替陈述（3）。

对于陈述（3），可以使用欧几里得算法计算最大公约数，并参考 [59] 或 [74]。此外，$\text{lcm}(x, y) = xy/$最大公约数(x, y)，因此陈述（3）完成了这一论断的证明。

对于陈述（9），所声称的复杂度可由 Lenstra 和Pomerance 设计的 AKS 素性测试变体达到，参见 [2] 和 [43]。

关于命题 (10)，请参阅第 78f. 页的 [5, 算法 3.1,], 并注意，该算法中的变量 m 在我们的情形下取值为 2。我们注意到，尽管第 79] 页的 [5, 定理 3.1 指出，该算法需要 $O(kn)$ 次位操作，但这一结论依赖于我们的计算模型并不具备的一项假设。具体而言，第 78f. 页的 [5, 算法 3.1 为每个比特串使用一个“指针”，这与我们使用该词的含义不可混淆。在我们的模型中，指针是一个捷径，只需 $O(1)$ 次位操作，就能跳转到内存中先前保存的位置；而

我们只能使用 $O(1)$ 个这样的指针（也就是说，随着输入长度趋于 ∞，所用指针的数量不得趋于 ∞）。另一方面，在第 78f. 页的 [5, 算法 3.1 中，“指针”一词似乎指的是相应比特串的内存地址。第 78f. 页的 [5, 算法 3.1 明确指出，该书作者假设，按照他们的定义，指针可以在 $O(1)$ 次位操作内处理（即存储并用于跳转到相应的比特串）。然而，在我们的模型中，由于需要 n 这些内存地址，处理一个地址需要 $O(\log n)$ 次位操作，因此我们的成本会额外增加因子 $\log n$。

对于陈述（11），我们注意到，只要使用指针（在我们对该词的意义下）立即跳回数组中保存的位置，就很容易证明第[38, 页的合并算法 M 158] 能够达到这一复杂度，这避免了复杂度中额外的对数因子。事实上，在 [38, 第 159], 页的讨论中指出，达到的复杂度为 $O(N_1+N_2)$，但这使用了每个存储的位字符串具有常数有界位长度这一假设（从而对于 $j=1$，2，导致 $l^{(j)}_{\text{总计}} \in O(N_j)$）。

□

下一个引理基本上给出了我们查询模型中查询问题的拉斯维加斯对偶复杂度。它用于将查询复杂度转换为拉斯维加斯对偶复杂度；见下文的引理 5.1.7。

引理 5.1.6。以下结论成立。

(1) 正整数 m 的素因数分解可以通过一个具有 m 界复杂度（$\log^{7+o(1)} m, \log^{3+o(1)} m, \log m$）的拉斯维加斯双重算法完成。(2) $x \in (\mathbb{Z}/m\mathbb{Z})^*$ 的乘法阶可以通过一个具有（期望） m 界双重复杂度（$\log^{7+o(1)} m, \log^{3+o(1)} m, \log m$）的拉斯维加斯双重算法计算。(3) 设 m 为正整数。对于 $x, y \in (\mathbb{Z}/m\mathbb{Z})^*$，模离散对数 $\log_x^{(m)}(y)$ 可以通过一个具有 m 界复杂度（$\log^{7+o(1)} m, \log^{3+o(1)} m, \log m$）的拉斯维加斯双重算法计算。(4) 设 q 为素数幂。对于 $x,y \in \mathbb{F}_q^*$，离散对数 $\log_x(y)$ 可以通过一个具有 q 界复杂度的拉斯维加斯双重算法计算

$$(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q).$$

(5) 设 p 为奇素数，k 为正整数。对于输入(p,k)，可以使用拉斯维加斯双重算法找到模 p^k 的原根，其复杂度为 p-有界复杂度 $(\log^{7+o(1)} p, \log^{3+o(1)} p, \log p)$。

证明。对于命题（1）和（2）的证明，我们采用[35, 第 7 节3], 中描述的方法，该方法最初由 Miller [48] 和 Shor [63]提出。为此，我们首先需要分析 [35, 第 137], 页所述阶查找算法的复杂度；该算法结合使用量子电路和一些经典后处理。需要注意的是，该算法各部分复杂度的界都是绝对界（即并非仅为期望值），但其输出仅以概率

至少为 0.399 的概率正确，因此它属于蒙特卡罗算法，而非我们最终希望构造的拉斯维加斯算法。

首先，我们需要分析连分数算法的复杂度（ [35, 定理 7.17] 提到该算法具有多项式复杂度，但未说明次数）。假设给定一个（采用二进制格式表示的）浮点数，它表示有理数 $k/2^n$，其中 $k \in \{0, 1, \ldots, 2^n - 1\}$。那么就存在一个分数序列

$$\mathrm{conv}_j(k, 2^n) = \frac{\mathrm{num}_j(k, 2^n)}{\mathrm{den}_j(k, 2^n)}$$

对于 $j = 1,2, .. ., N \in O(n)$， $k/2^n$的（主）渐近分数，它们是相对于其分母大小而言的 $k/2^n$ 的最佳逼近 $\mathrm{den}_j(k, 2^n) \leq 2^n$；有关详细信息，请参阅 [35, 定理 7.1.7和习题7.1.7] 以及 [40,第一章]。我们稍后需要用到的一个重要性质是，任意一个既约整数分数 y/z 使得

$$\left| \frac{k}{2^n} - \frac{y}{z} \right| \leq \frac{1}{2z^2}$$

是 $k/2^n$ 的一个渐近分数；见第 11] 页的 [40, 推论 2。根据第 2], 页的 [40, 定理 1，两个序列 $(\mathrm{num}_j(k, 2^n))_{j=1,\ldots,N}$ 和 $(\mathrm{den}_j(k, 2^n))_{j=1,\ldots,N}$ 均通过简单的递归定义（每一步递归都涉及 $O(1)$ 次整数加法和乘法），其定义基于 $k/2^n$ 的所谓连分数系数，这些系数正是对 $(2^n, k)$ 应用欧几里得算法时所进行的除法所得的整数商值。鉴于第 937] 页提出的 [18, 问题 31-2（另见 [15]，其中使用望远镜求和论证给出了该问题的完整解答），可以用 $O(n^2)$ 次位运算计算并存储 $k/2^n$ 的连分数系数。随后，根据引理 5.1.5 的陈述 (1) 和 (3)，计算所有相关 j 对应的 $\mathrm{num}_j(k, 2^n)$ 和 $\mathrm{den}_j(k, 2^n)$ 还需要 $O(n \cdot n^{1+o(1)}) \subseteq O(n^{2+o(1)})$ 次位运算。

Having analyzed the continued fractions algorithm, let us now turn to the order-finding algorithm described in [35, p. 137]. In accordance with our notation, we assume that this algorithm is used to find the multiplicative order of x modulo m (in [35], the variable a, respectively N, is used in place of x, respectively m). The algorithm starts by computing $m' := \lceil 2 \log m \rceil$, which takes $O(\log m)$ bit operations. We observe that $\mathrm{ord}(x)$, the multiplicative order of x modulo m, is at most $2^{(m'-1)/2}$. After this, we need to initialize two m'-qubit registers, which in our dual model formally takes $O(\log m)$ bit operations for printing the length m' bit strings $0 \cdots 0$ and $0 \cdots 01$, followed by $2 \in O(1)$ conversions of these strings into the corresponding qubit registers $|0\rangle^{\otimes m'}$ and $|0 \cdots 01\rangle$. Steps 4–6 of the algorithm in [35, p. 137] are applications of quantum circuits to those registers, which according to the analysis in [35, pp. 138f.] consist of $O(\log^{2+o(1)} m)$ elementary gates. Next, the measurement described in Step 7 of [35, p. 137] corresponds to one more conversion in our model (this time from qubits to classical bits), and with high probability, it leads to a "good estimate" (see below) $k_1/2^{m'}$ of a random integer multiple $t/\operatorname{ord}_m(x)$ of $1/\operatorname{ord}_m(x)$, with $t \in \{0, 1, \ldots, \operatorname{ord}_m(x) - 1\}$. Considering Step 8 of [35, p. 137] next, we believe that there is a mistake in the formulation of this step, the first sentence of which

我们认为应当将其表述为（使用其中的记号）：“使用连分数算法获得整数 $c_1 \geq 0$ 和 r_1，使得 $1 \leq r_1 \leq 2^{(n-1)/2}$ 满足$|x_1/2^n - c_1/r_1| \leq 1/2^{n+1}$。”无论如何，这是一个可行的表述。事实上，切换回我们的记号，只要步骤 7的输出 $k_1/2^{m'}$ 是某个 $t \in \{0, 1, \ldots, \mathsf{ord}(x) - 1\}$的良好估计 $t/\mathsf{ord}(x)$，根据“良好估计”的定义（见小节 7.1.1], 的开头，尤其参见 [35,定理 7.1.4]），即可推出:

$$\left| \frac{t}{\mathrm{ord}(x)} - \frac{k_1}{2^{m'}} \right| \leq \frac{1}{2^{m'+1}} < \frac{1}{2^{m'}} \leq \frac{1}{2\,\mathrm{ord}(x)^2}$$

因此，如上所述，对于某个 j，根据第 11]页的推论 2，我们有 $t/\mathsf{ord}(x) = \mathsf{conv}_j(k_1, 2^{m'})$ [40,。根据我们上述对连分数算法的分析，我们因此可以使用 $O(\log^{2+o(1)} m)$ 位运算，求得一个指标 j 以及相关的值 $\mathsf{num}_j(k_1, 2^{m'})$ 和 $\mathsf{den}_j(k_1, 2^{m'})$，其中 $\mathsf{den}_j(k_1, 2^{m'}) \leq 2^{(m'-1)/2}$，使得

$$\left| \mathrm{conv}_j(k_1, 2^{m'}) - \frac{k_1}{2^{m'}} \right| = \left| \frac{\mathrm{num}_j(k_1, 2^{m'})}{\mathrm{den}_j(k_1, 2^{m'})} - \frac{k_1}{2^{m'}} \right| \leq \frac{1}{2^{m'+1}},$$

除非我们在第 7 步的输出方面运气不佳。如果是这样，放弃计算并按照第 [35, 步第 137] 页上的说明输出 “ 失败 ” 是合理的。现在，可以得到

$$\begin{aligned} \left| \frac{t}{\mathrm{ord}(x)} - \mathrm{conv}_j(k_1, 2^{m'}) \right| &\leq \left| \frac{t}{\mathrm{ord}(x)} - \frac{k_1}{2^{m'}} \right| + \left| \frac{k_1}{2^{m'}} - \mathrm{conv}_j(k_1, 2^{m'}) \right| \\ &\leq \frac{1}{2^{m'+1}} + \frac{1}{2^{m'+1}} = \frac{1}{2^{m'}} \leq \min\left(\frac{1}{2\,\mathrm{ord}(x)^2}, \frac{1}{2\mathrm{den}_j(k_1, 2^{m'})^2} \right). \end{aligned}$$

Using [35, Exercise 7.1.7(b)], this implies that $t/\,\mathrm{ord}(x) = \mathrm{conv}_j(k_1, 2^{m'})$, as required for the correctness of the algorithm from [35, p. 137]. Step 9 of [35, p. 137] is just a repetition of Steps 1–8, hence does not make a difference for the O-class of the complexity. Finally, Steps 10 and 11 of [35, p. 137] take $O(\log^{2+o(1)} m)$ bit operations by statements (6) and (8) of Lemma 5.1.5. In summary, the order-finding algorithm from [35, p. 137] may be viewed as a dual algorithm which, on input $x \in (\mathbb{Z}/m\mathbb{Z})^*$ and m, outputs the multiplicative order of x modulo m with probability at least 39.9% (see [35, Theorem 7.3.2]), and does so taking $O(\log^{2+o(1)} m)$ bit operations and elementary quantum gates each, as well as $O(1)$ conversions of $O(\log m)$-bit strings to $O(\log m)$-qubit registers or vice versa. As noted in [35, Theorem 7.3.2] (and is clear from Step 11), unless the output of that algorithm is “FAIL”, it always outputs at least an integer multiple of $\mathrm{ord}(x)$. This concludes the preparation for the proofs of statements (1) and (2), which we tackle next.

对于陈述（1），我们假设给定一个正整数 m。我们希望求出 m 的素因数分解。具体而言，我们希望输出由数对 $(p, \nu_p(m))$ 组成的列表，其中 p 遍历 m 的素因子。首先，我们描述并分析一种确定性（经典）算法，用于判断 m 是否是单个素数 p 的幂；如果是，则输出 $(p, \nu_p(m))$；另请参见 [35, 练习 7.3.3]。该算法

该方法基于一个确定性算法，用于判断给定的 $m \in \mathbb{N}^+$是否为正整数 $n < m$ 的 n^k 次幂；若是，则输出使 $k \geq 2$取最小可能值时的 (n, k)。注意，若对于某个 $n \in \{1, 2, \ldots, m-1\}$ 和某个 $k \in \mathbb{N}^+$，有 $m = n^k$，则 $k \leq \lfloor \log_2 m \rfloor$。因此，我们遍历 $k = 2$、3、.....、$\lfloor \log_2 m \rfloor$；对每个固定的 k，利用函数 $x \mapsto x^k$的严格单调性，对 n进行二分查找。具体而言，我们初始化 $n := 2$，并在 $n^k < m$成立时，将 n 不断加倍，直到 $n^k = m$ 或 $n^k > m$成立。若后一种情况成立，则在 $\frac{n}{2}$ 和 n之间开始二分查找。采用这种方法，我们计算的幂 n^k 不会超过 $2^{\lfloor \log_2 m \rfloor} m \leq m^2$，因此，过程中每次单独的幂计算所需时间.....

$$O(\log(k) \log^{1+o(1)}(m^2)) = O(\log \log m \log^{1+o(1)} m) = O(\log^{1+o(1)} m)$$

根据引理 5.1.5 的陈述(6) 进行位运算。由于我们对 k的 $O(\log m)$ 个值进行循环，并且对于每个 k，对 n 的二分查找需要 $O(\log m)$ 次迭代，因此可知它需要

$$O(\log m \cdot \log m \cdot \log^{1+o(1)} m) = O(\log^{3+o(1)} m)$$

使用位运算查找 k 及其关联的 $n = \sqrt[k]{m}$ 的最小可行值，或确定它们不存在。

如上所述，我们可以使用这个求根算法来检查给定的 $m \in \mathbb{N}^+$ 是否为素数幂；如果是，则将其写成这种形式——下面说明具体方法。首先，通过迭代幂查找例程，可以将 $m = n^k$ 写成最大的$k \in \{1, 2, \ldots, \lfloor \log_2 m \rfloor\}$，使得 m 具有一个整数 k 次根；这需要

$$O\left(\sum_{t=0}^{\infty} \log^{3+o(1)}\left(m^{1/2^t}\right)\right) = O\left(\sum_{t=0}^{\infty} \frac{1}{2^{3t}} \cdot \log^{3+o(1)} m\right) = O(\log^{3+o(1)} m)$$

位运算。于是问题归结为检查 n 是否为素数，根据引理 5.1.5 的陈述（9），这需要 $O(\log^{6+o(1)} n) \subseteq O(\log^{6+o(1)} m)$ 次位运算。总之，我们有一个复杂度为 $O(\log^{6+o(1)} m)$ 的确定性算法，用于判断 m 是否为素数幂，并在是素数幂的情况下将其写成这种形式。

Miller 利用归约思想提出的 Shor 通用拉斯维加斯对偶方法，用于对 $m \in \mathbb{N}^+$ 进行因数分解，其概要见 [35, 第 132 页及后续页]。我们先从 m中分离出因数 $2^{\nu_2(m)}$。由于 m 以二进制表示给出，这只需 $O(\log^{1+o(1)} m)$ 次位运算，相当于计数器增加 $O(\nu_2(m))$ 次 $\subseteq O(\log m)$；该计数器始终不超过 $O(\log m)$（因此有 $O(\log \log m) \subseteq O(\log^{o(1)} m)$ 个二进制位）。现在，我们令 $m' := m/2^{\nu_2(m)}$。在陈述（1）的证明其余部分中，我们只处理奇正整数。我们描述一个拉斯维加斯例程，用于判定给定的奇正整数 n是否为素数幂，然后执行以下操作：

- 如果 n 是素数幂，则将 n 写作 $p^{\nu_p(n)}$；
- 如果 n 不是素数幂，它将找到 n 的因式分解，形式为 $n = n' \cdot n''$，其中 $1 < n'$、$n'' < n$。

我们已经在上文描述了如何判定 $n = p^{\nu_p(n)}$，并在判定为是时，如何使用 $O(\log^{6+o(1)} n)$ 位运算将其写成这种形式，因此我们首先应用该例程，然后

may henceforth assume that it returned that n is *not* a prime power. Following [35, p. 133], we wish to draw an integer $x \in \{2, 3, \ldots, n-1\}$ uniformly at random. Letting $N := \lfloor \log_2(n-3) \rfloor + 1$, we aim to draw the N-bit integer $y \in \{0, 1, \ldots, n-3\}$ uniformly at random, then set $x := y+2$. To draw y, we initialize an N-qubit register to $|0\rangle^{\otimes N}$, then pass it through an N-dimensional Hadamard circuit (with elementary gate complexity $N \in O(\log n)$) to get a uniform superposition of all N-bit strings, so that a simple measurement returns (the binary representation of) a random integer in $\{0, 1, \ldots, 2^N - 1\}$. The probability that this integer lies in the range for y is at least $1/2$, so we only need to iterate this procedure an expected number of $O(1)$ times until we get a suitable value for y, using $O(\log n)$ bit operations and elementary quantum gates as well as $O(1)$ conversions to and from $O(\log n)$-bit strings. Following that, we compute $x = y + 2$ and $\gcd(x, n)$, taking $O(\log^{1+o(1)} n)$ bit operations by statements (1) and (8) of Lemma 5.1.5. If $\gcd(x, n) > 1$, we may output the factorization $n = n' \cdot n''$ with $n' = \gcd(x, n)$, taking just another $O(\log^{1+o(1)} n)$ bit operations to compute n'' by division, and are done. Otherwise, x is a (uniformly random) unit modulo n, and we proceed to apply the order-finding routine from [35, p. 137] to get an output o which is either a number or the string "FAIL", and is equal to $\mathrm{ord}(x)$ with probability at least 0.399. If o is "FAIL", we repeat this routine on the same value of x until we get an output that is actually a number (only $O(1)$ repetitions needed by expectancy). Then, if o is *not* even, we abandon this value of x and choose y anew (because we want $2 \mid \mathrm{ord}(x)$, and even if o may not be equal to $\mathrm{ord}(x)$, it is an integer multiple of $\mathrm{ord}(x)$, as was noted above) until we get an x such that either $\gcd(x, n) > 1$ or the associated alleged multiplicative order $o \in \mathbb{N}^+$ is even. This, too, only requires an expected number of $O(1)$ attempts, because for a randomly selected $x \in (\mathbb{Z}/n\mathbb{Z})^*$, the order of x is even with probability at least $1/2$. We then compute $z := x^{o/2} \bmod n$, taking $O(\log^{2+o(1)} n)$ bit operations by statement (6) of Lemma 5.1.5. Because n is not a prime power, we have $\gcd(z-1, n) > 1$ with probability at least $1/2$, so after expectedly $O(1)$ more tries, we will indeed have found a nontrivial factorization of n. Taking into account the complexity of the order-finding routine from [35, p. 137] which we analyzed above, this process expectedly takes $O(\log^{6+o(1)} n)$ bit operations, $O(\log^{2+o(1)} n)$ elementary quantum gates, and $O(1)$ conversions to and from $O(\log n)$-bit strings.

现在让我们回到对奇正整数 $m' = m/2^{\nu_2(m)}$ 进行因数分解的问题上。通过反复应用我们刚才描述的因数寻找例程，该例程需要应用 $O(\log m)$次，陈述（1）随即成立（某些必要的确定性后处理的复杂度，例如将出现在多个所得因数中的素数的指数相加，显然包含在 $O(\log^{7+o(1)} m)$ 之下）。

对于陈述（2），假设给定一个模数 m 和一个单位元 $x \in (\mathbb{Z}/m\mathbb{Z})^*$，我们希望给出一种计算 $\mathrm{ord}(x)$ 的拉斯维加斯算法。为此，我们首先利用陈述（1）中的算法，在 m-有界的拉斯维加斯双重复杂度 $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$ 下对 m进行因式分解。已知 m 的因式分解后，我们可以在相同的 m-有界拉斯维加斯双重复杂度下计算欧拉函数值 $\phi(m)$，并对其进行因式分解。现在，对于每个素数 $p \mid \phi(m)$，我们可以求出 $\nu_p(\mathrm{ord}(x))$，即满足以下条件的最小 $v_p \in \{0, 1, \ldots, \nu_p(\phi(m))\}$

$$x^{p^{v_p}\phi(m)_{p'}} \equiv 1 \pmod{m},$$

其中 $\phi(m)_{p'} := \phi(m)/p^{\nu_p(\phi(m))}$。更具体地说，我们可以使用二分查找来求出$\nu_p$(ord($x$))；根据引理 5.1.5 的陈述（6），其代价为

$$O(\log\log m \cdot \log^{2+o(1)} m) = O(\log^{2+o(1)} m)$$

对单个 p，需要进行位运算来求ν_p(ord(x))，因此求出所有这些赋值需要 $O(\log^{3+o(1)} m)$次位运算。最后，我们将 ord(x) 本身计算为所有素数幂 $p^{\nu_p(\mathrm{ord}(x))}$ 的乘积，其中 p 遍历$\phi(m)$ 的素因子 $\in$ ord(x)$\mathbb{Z}$。这还需要进行 $O(\log^{3+o(1)} m)$ 次位运算，从而证明了陈述 (2)。

对于陈述（3）和（4）的证明，我们还需要做一些准备。在[35, 第7节4], 中，讨论了一种计算离散对数的一般方法，该方法适用于从任意黑盒群 G 中选取的元素，其中每个元素都具有唯一编码。对于给定的 x、$y \in G$，满足 $y = x^t$（其中 $t \in \mathbb{Z}$），并且已知 x 在 G 中的阶，该方法以高概率返回唯一的$t \in \{0, 1, \ldots, \mathrm{ord}(x) - 1\}$，使得 $y = x^t$；这称为 y 以 x为基的离散对数（在G中），记作 log$_x$(y)。然而，如果 y 不是 x 在 G中的幂，看来这种方法无法提供一种确定无疑地确认这一点的手段，而这正是我们所需的拉斯维加斯算法所要求的。这意味着，除了 [35, 第7节4], 中的离散对数算法之外，我们还需要一个拉斯维加斯例程，首先检查 y 是否为 x 在其中的幂。我们逐一分析这些算法，先从计算 log$_x$(y)的例程开始；当 y 是 x 的幂时，对于 $G =$ （$\mathbb{Z}/m\mathbb{Z}$）* 或$G = \mathbb{F}_q^*$均适用。与第2.4小节中一样，我们通过规定log$_x$(y) : $= \infty$ （当 y 不是 x的幂时），将记号 log$_x$(y) 扩展到任意的 x、$y \in G$。

一个重要的观察是，[35, p. 144] 中描述的离散对数算法仅在 ord(x) 为素数时才有效（见 [35, p. 143]上公式 (7.4.2.) 后的第二段）。对于一般情形，我们遵循 [35, pp. 244f. 中“方法1”的做法，从推论 A.2.2]之后开始。首先令 $m_0 := 1$ 并令 rem$_0 := 0$。那么显然，log$_x$(y) $\equiv$ rem$_0$ (mod m_0)。目标是递归定义整数 $m_1 \mid m_2 \mid \cdots \mid m_N =$ ord(x) 和 rem$_1$, rem$_2$, . . . , rem$_N$，使得 rem$_N$ =log$_x$(y)，并且始终满足 log$_x$(y) $\equiv$rem$_k$ (mod m_k)。如 [35, p. 244 中推论 A.2.2], 之后紧接着所述，从 [35, p. 144] 运行该算法，可以让我们以很高的概率求出 m_{k+1} 和 rem$_{k+1}$ （根据 $m_k <$ ord(x) 和 rem$_k$ ），而且不会有得到错误值的风险，只可能出现“失败”。在每一步中，这都涉及

- O(1) 引理 5.1.5 的陈述 (1)–(8) 中涉及的算术运算；若 $G = (\mathbb{Z}/m\mathbb{Z})^*$，则分别需要 $O(\log^{2+o_{(1)}m})$ 次位运算；若 $G = \mathbb{F}_q^*$，则需要 $O(\log^{2+o_{(1)}q})$ 次位运算；
- $O(\log^{2+o_{(1)}m})$，以及 $O(\log^{2+o_{(1)}} q)$ 个基本量子门；
- O(1) 与 $O(\log m)$ 位字符串之间的转换，以及与 $O(\log q)$ 位字符串之间的转换。

如 [35, 第 245], 页所述， N 该循环的迭代次数为 O(logord(x))，因此分别为 O(log m) 和 O(log q)。因此，当 log$_x$(y) 未知，且 ord(x) 已知时，我们可以使用以下 m 有界、分别为 q 有界的拉斯维加斯双重复杂度来计算它：

- $(\log^{3+o(1)} m, \log^{3+o(1)} m, \log m)$ if $G = (\mathbb{Z}/m\mathbb{Z})^*$;

- $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$ if $G = \mathbb{F}_q^*$.

至此，我们为证明陈述 (3) 和 (4) 所做的准备工作就完成了。

对于陈述（3），我们假设给定了 $x, y \in (\mathbb{Z}/m\mathbb{Z})^*$。为了计算 $\log_x(y)$，我们首先检查 y 起初是否是 x 的幂。我们首先对 m 进行因数分解，这需要 m-有界的拉斯维加斯双重复杂度

$$(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$$

根据陈述（1）。对于 m 的素因子 p，我们设 $v_p := \nu_p(m)$ 和 $m_p := p^{v_p}$。我们希望对每个素数 $p \mid m$，计算以 x 为底的 y 模 m_p 的离散对数 $\log^{(m_p)}{}_x(y)$，并在接下来的两段中说明如何计算。我们用记号 $\mathrm{ord}_n(z)$ 表示 z 模 n 的乘法阶。

首先，我们假设 $p > 2$。根据陈述（2），我们以 m_p-有界拉斯维加斯双重复杂度（$\log^{7+o(1)} m_p, \log^{3+o(1)} m_p, \log m_p$）计算 $\mathrm{ord}_{m_p}(x)$ 和 $\mathrm{ord}_{m_p}(y)$。由于单位元群 $(\mathbb{Z}/m_p\mathbb{Z})^*$ 是循环的，因此 y 是模 m_p 意义下 x 的幂，当且仅当 $\mathrm{ord}_{m_p}(y)$ 整除 $\mathrm{ord}_{m_p}(x)$。根据引理 5.1.5 的陈述（3），只需 $O(\log^{1+o(1)} m_p)$ 次位操作即可检查这一点。如果该整除关系不成立，那么 $\log^{(m_p)}{}_x(y)$ 是 ∞，$\log_x^{(m)}(y)$ 也一样，因此问题解决。否则，我们使用 [35, 第 144 页和附录 A.2], 中的拉斯维加斯例程计算 $\log^{(m_p)}{}_x(y)$，其拉斯维加斯双重复杂度为 m_p-有界（$\log^{3+o(1)} m_p, \log^{3+o(1)} m_p, \log m_p$）；需要注意的是，此时我们确实知道 $\mathrm{ord}_{m_p}(x)$，因为它已事先计算过。

现在我们假设 $p = 2$。我们按照处理 $p > 2$ 时的类似方法进行，即先检查 $\log^{(m_2)}{}_x(y)$ 是否为 ∞；如果不是，则以 m_2 有界的拉斯维加斯双重复杂度计算其精确整数值，复杂度为

$$(\log^{3+o(1)} m_2, \log^{3+o(1)} m_2, \log m_2),$$

或者 $(\log^{7+o(1)} m_2, \log^{3+o(1)} m_2, \log m_2)$，如果此时尚未计算 $\mathrm{ord}_{m_2}(x)$。不过，检查 $\log^{(m_2)}{}_x(y) = \infty$ 是否成立，比检查 $p > 2$ 更复杂一些，因为 $(\mathbb{Z}/m_2\mathbb{Z})^*$ 不一定是循环的。我们可以假设 $m_2 > 2$（否则，$\log^{(m_2)}{}_x(y)$ 就等于 1），并区分几种情况。

- 如果 $x \equiv y \equiv 1 \pmod 4$，检查这一点只需 $O(1)$ 次位运算，因为 x 和 y 均以二进制给出，那么 x 和 y 都属于由单位元 5 生成的 $(\mathbb{Z}/m_2\mathbb{Z})^*$ 的循环子群（若 $m_2 = 4$，该单位元 5 等于单位元 1）。因此，与处理 $p > 2$ 时一样，我们有：y 是 x 的幂，当且仅当 $\mathrm{ord}_{m_2}(y) \mid \mathrm{ord}_{m_2}(x)$。
- 若 $x \equiv 1$ (模 4) 且 $y \equiv 3$ (模 4)，则 y 不可能是模 m_2 下的 x 的幂。

如果 $x \equiv 3$（模 4）且 $y \equiv 1$（模4），那么 y 是 x 模 m_2 的幂，当且仅当 y 是 x^2 模 m_2的幂。由于 $x^2 \equiv 1$（模4），我们得出，y 是 x 模 m_2 的幂，当且仅当 $\mathrm{ord}_{m_2}(y) \mid \mathrm{ord}_{m_2}(x^2)$。

如果 $x \equiv y \equiv 3$（模4），那么 y 是 x 模 m_2 的幂，当且仅当 $-y$是 $-x$ 模 m_2 的奇指数幂。因此，为了检查 y 是否是 x 模 m_2 的幂，我们首先检查 $\mathrm{ord}_{m_2}(-y) \mid$

$\mathrm{ord}_{m_2}(-x)$。如果不是这样，那么 y 肯定不是模 m_2 下 x 的幂。否则，我们计算 $\log_{-x}^{(m_2)}(-y)$，并检查它是否为奇数。

总之，由于

$$\sum_{p|m} \log^k m_p \in O(\log^k m)$$

对于所有实指数 $k \geq 1$，我们得出结论：对每个素数 $p \mid m$ 计算 $\log_x^{(m_p)}(y)$，对所有 p 总共需要 m-有界拉斯维加斯双重复杂度 $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$。如上所述，如果这些“主离散对数”中任意一个是 ∞，那么 y 不是模 m x 的幂，即 $\log_x^{(m)}(y) = \infty$，问题便迎刃而解。否则，注意到对于每个整数 t，我们有

$$y \equiv x^t \pmod{m} \text{ if and only if } y \equiv x^t \pmod{m_p} \text{ for all primes } p \mid m,$$

我们发现，$\log_x^{(m)}(y) < \infty$ 当且仅当该同余方程组

$$t \equiv \log_x^{(m_p)}(y) \pmod{\mathrm{ord}_{m_p}(x)} \text{ for all primes } p \mid m$$

关于变量 t 的同余方程组是相容的，此时 $\log_x^{(m)}(y)$ 是其在 0 和 $\mathrm{lcm}\left(\{\mathrm{ord}_{m_p}(x) : p \mid m\}\right) - 1 = \mathrm{ord}_m(x) - 1$ 之间的唯一解。我们已经计算出该同余方程组的右端和模数，可能除外 $\mathrm{ord}_{m_2}(x)$；计算它需要 m-有界拉斯维加斯双重复杂度 $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$。接下来，我们可以利用命题 2.2.1 中陈述（1）与陈述（2）的等价性来检验该同余方程组是否相容，这需要对小于 m 的整数进行 $O(\log m)$ 次减法、最大公因数计算和除法，因此可通过 $O(\log^{2+o(1)} m)$ 次位操作完成。若该方程组相容，则可通过求解该方程组（这是确定性的，当然可以用 $O(\log^{7+o(1)} m)$ 次位操作完成，但此处不详述）来确定 $\log_x^{(m)}(y)$ 的整数值；也可以计算 $\mathrm{ord}_m(x)$，并运行前文第 [35, 页及附录 A 中提到的例程，再花费另一个 m-有界拉斯维加斯双重复杂度 $(\log^{7+o(1)} m, \log^{3+o(1)} m, \log m)$。至此，陈述（3）的证明完成。

对于陈述（4），我们假设已给定 x、$y \in \mathbb{F}_q^*$。根据本节开头所述广义分圆映射的两种表示形式，我们考虑求解 $\log_x(y)$ 这一计算问题的两个不同版本：

- 版本 1：x、y 给出为同一个未指定本原元 ω 的幂，其中该本原元属于 $\mathbb{F}_q$；
- 版本 2：已知一个在 $\mathbb{Z}/p\mathbb{Z}$ 上次数为 $\log_p q$ 的本原不可约多项式 $P(T)$，并将 $\mathbb{F}_q$ 视为 $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$；$x$、$y$ 给出为该商环中的元素，且采用标准形式（即表示为变量 T 的 $\mathbb{Z}/p\mathbb{Z}$ 上次数小于 $\log_p q$ 的多项式）。

在这两种情形下，我们都证明可以判断 y 是否为 x 的幂；若是，则可以用总计为 q 有界的拉斯维加斯双重复杂度 $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$，求出 $\log_x(y)$ 的整数值。事实上，我们注意到，对于任意给定的本原元

$\omega \in \mathbb{F}_q$，函数 $\log_\omega : \mathbb{F}_q^* \to \mathbb{Z}/(q-1)\mathbb{Z}$ 是群同构。在计算问题的第一个版本中，ω 的某个（未指定）值已预先固定；在第二个版本中，我们令 $\omega := T$（视为 $(\mathbb{Z}/p\mathbb{Z})[T]/(P(T))$ 的一个元素）。在这两种情况下，我们都可以求出 $\log_\omega(x)$ 和 $\log_\omega(y)$；在第一种情形中，x和 y 明确给定为 ω的幂；在第二种情形中，我们使用 [35, 第 144 页和附录 A.2] 中的例程计算 $\log_\omega(x)$ 和 $\log_\omega(y)$。之所以可行，是因为 x 和 y 是 ω 的幂，并且我们知道 $\operatorname{ord}(\omega) = q-1$。在这两种情况下，这两个离散对数都可以用 q 有界的拉斯维加斯双重复杂度 $(\log^{3+o(1)} q, \log^{3+o(1)} q, \log q)$ 计算出来（事实上，在第一种情形中，只需从输入中复制它们，因此仅需 $O(\log q)$ 次位操作）。

求出$\log_\omega(x)$和 $\log_\omega(y)$后，问题归结为检查 $\log_\omega(y)$ 是否是 $\mathbb{Z}/(q-1)\mathbb{Z}$ 中 $\log_\omega(x)$ 的倍数，即是否

$$\log_\omega(y) \equiv k \cdot \log_\omega(x) \pmod{q-1} \tag{22}$$

对于某个 $k \in \mathbb{Z}$。但当且仅当最大公约数$(\log_\omega(x)\ q-1)\ |\log_\omega(y)$时，这一点才成立，而这可以使用 $O(\log^{1+o(1)} q)$ 位运算进行检查。如果成立，那么 $\log_x(y)$ 是同余式 (22) 的唯一解 $k \in \{0, 1, \ldots, q-2\}$，即模 $q-1$

$$\log_x(y) = \frac{\log_\omega(y)}{\gcd(\log_\omega(x), q-1)} \cdot \operatorname{inv}_{\frac{q-1}{\gcd(\log_\omega(x), q-1)}} \left(\frac{\log_\omega(x)}{\gcd(\log_\omega(x), q-1)} \right),$$

可通过最后一批 $O(\log^{1+o(1)} q)$ 位运算求值。至此，陈述（4）的证明完成。

For statement (5), we first aim to find a primitive root modulo p. A polynomial-time probabilistic classical algorithm for doing so is [20, Algorithm 1], which is a refinement of an earlier algorithm by Bach [9], itself based on Itoh's idea of using partial factorizations of $p-1$ to find a primitive root with high probability. We follow this approach, but since we can factor $p-1$ completely, our situation is easier. We start by factoring $p-1$, taking p-bounded Las Vegas dual complexity $(\log^{7+o(1)} p, \log^{3+o(1)} p, \log p)$ by statement (1). Say $p-1 = \prod_{j=1}^{K} p_j^{v_j}$ is the said factorization. For each prime divisor p_j of $p-1$, we wish to find a unit $y_j \in (\mathbb{Z}/p\mathbb{Z})^* = \{1, 2, \ldots, p-1\}$ such that $p_j^{v_j}$ divides $\operatorname{ord}_p(y_j)$. Equivalently, y_j should *not* be a p_j-th power in $\mathbb{Z}/p\mathbb{Z}$. The proportion of units that satisfy this is $1 - \frac{1}{p_j} \geq \frac{1}{2}$, so if we pick $y_j \in (\mathbb{Z}/p\mathbb{Z})^*$ at random, then check whether $y_j^{(p-1)/p_j} \not\equiv 1 \pmod{p}$, it only takes an expected number of $O(1)$ tries until we succeed at finding y_j. As in the proof of statement (1), we perform this random drawing of y_j using a simple quantum circuit consisting of a $(\lfloor \log_2(p-2) \rfloor + 1)$-qubit Hadamard gate. This means that the expected p-bounded Las Vegas dual complexity of finding y_j is $(\log^{2+o(1)} p, \log p, 1)$ for a single j, and $(\log^{3+o(1)} p, \log^2 p, \log p)$ for all $j = 1, 2, \ldots, K$ together. Once the y_j have been found, the unit

$$\mathfrak{r} := \prod_{j=1}^{K} y_j^{(p-1)/p_j^{v_j}} \in (\mathbb{Z}/p\mathbb{Z})^*,$$

它可以通过额外的 $O(\log^{3+o(1)} p)$ 位运算计算得到，是模 p的一个原根。事实上，该乘积中的第 j 个因数的阶为 $p_j^{v_j}$，并且由于

数 $p_j^{v_j}$ 两两互素，且群 $(\mathbb{Z}/p\mathbb{Z})^*$ 是阿贝尔群，因此 $\mathrm{ord}_p(\mathfrak{r}) = \prod_{j=1}^{K} p_j^{v_j} = p-1$。

根据 $\mathfrak{r}$，不难构造模 p^k 的原根 $\mathfrak{r}^+$。事实上，如果 $k=1$，我们只需令 $\mathfrak{r}^+ := \mathfrak{r}$。此外，模 p^2 的原根对每个 $k \geq 2$ 都是模 p^k 的原根，而 $\mathfrak{r}$ 或 $\mathfrak{r}+p$ 是模 p^2 的原根。因此，如果 $k > 1$，我们只需检查 $\mathfrak{r}^{p-1} \equiv 1 \pmod{p^2}$ 是否成立，这需要 $O(\log^{2+o(1)} p^2) = O(\log^{2+o(1)} p)$ 次位运算。若是，则令 $\mathfrak{r}^+ := \mathfrak{r}+p$；否则，令 $\mathfrak{r}^+ := \mathfrak{r}$。至此，陈述 (5) 的证明以及引理 5.1.6 的整体证明均告完成。 □

我们对引理5.1.6的首次应用是以下前述结果：将查询复杂度转换为拉斯维加斯对偶复杂度。

引理 5.1.7. 设 $\mathfrak{L}$ 是一个算法问题，y、y_1、...、y_n 是与 $\mathfrak{L}$ 的可容许输入相关联的非负实参数，满足

$$y(\vec{\mathfrak{x}}) \leq h(y_1(\vec{\mathfrak{x}}), \ldots, y_n(\vec{\mathfrak{x}}))$$

对所有 $\vec{\mathfrak{x}} \in \mathfrak{L}_{\text{in}}$ 均成立，其中 h 是一个固定函数 $[0,\infty)^n \to [0,\infty)$。此外，设

$$\vec{\mathcal{C}}^{\text{qry}} = (\mathcal{C}_{\text{class}}, \mathcal{C}_{\text{fdl}}, \mathcal{C}_{\text{mdl}}, \mathcal{C}_{\text{mord}}, \mathcal{C}_{\text{prt}})$$

是关于 y_1、...、y_n 的 $\mathfrak{L}$ 有界查询复杂度。那么

$$\vec{\mathcal{C}} = (\mathcal{C}'_{\text{class}}, \mathcal{C}_{\text{quant}}, \mathcal{C}_{\text{conv}}),$$

其中 $\mathcal{C}'_{\text{class}}$、$\mathcal{C}_{\text{quant}}$、$\mathcal{C}_{\text{conv}}$：$[0,\infty)^n \to [0,\infty)$ 如下文所定义，是 $\mathfrak{L}$ 的 y 有界拉斯维加斯双重复杂度。我们用 $\vec{z}$ 作为 $(z_1, z_2, \ldots, z_n) \in [0,\infty)^n$ 的简写。

$$\begin{aligned}
&\mathcal{C}'_{\text{class}}(\vec{z}) := \mathcal{C}_{\text{class}}(\vec{z}) + \log^{7+o(1)}(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})) \\
&\quad + \log^{3+o(1)}(h(\vec{z}))\mathcal{C}_{\text{fdl}}(\vec{z}); \\
&\mathcal{C}_{\text{quant}}(\vec{z}) := \log^{3+o(1)}(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{fdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})); \\
&\mathcal{C}_{\text{conv}}(\vec{z}) := \log(h(\vec{z})) \cdot (\mathcal{C}_{\text{mdl}}(\vec{z}) + \mathcal{C}_{\text{fdl}}(\vec{z}) + \mathcal{C}_{\text{mord}}(\vec{z}) + \mathcal{C}_{\text{prt}}(\vec{z})).
\end{aligned}$$

证明. 由引理 5.1.6 及相关概念的定义可轻易得出。例如，在计算 $\mathcal{C}'_{\text{class}}(\vec{z})$ 时，我们不仅要考虑在特殊查询之外所耗费的位运算（由加项 $\mathcal{C}_{\text{class}}(\vec{z})$ 表示），还要考虑执行查询所耗费的位运算；为完成这些查询，需使用引理 5.1.6 的证明中讨论的算法。例如，对于输入 $\vec{\mathfrak{x}}$，在计算 $\vec{\mathfrak{x}}$ 的一个可容许输出的过程中，需要进行的每个 $O(\mathcal{C}_{\text{fdl}}(y_1(\vec{\mathfrak{x}}), \ldots, y_n(\vec{\mathfrak{x}})))$ 有限域离散对数查询，都是在计算大小至多为 $h(y_1(\vec{\mathfrak{x}}), \ldots, y_n(\vec{\mathfrak{x}}))$ 的有限域中的离散对数。因此，根据引理 5.1.6(4)，这些“fdl 查询”合计耗费

$$O(\mathcal{C}_{\text{fdl}}(y_1(\vec{\mathfrak{x}}), \ldots, y_n(\vec{\mathfrak{x}})) \cdot \log^{3+o(1)}(h(y_1(\vec{\mathfrak{x}}), \ldots, y_n(\vec{\mathfrak{x}}))))$$

位运算，因此在 $\mathcal{C}'_{\text{class}}(\vec{z})$ 的定义中包含了项 $\mathcal{C}_{\text{fdl}}(\vec{z}) \cdot \log^{3+o(1)}(h(\vec{z}))$。对于其他三类查询，可以结合引理 5.1.6 和以下事实：

$$\sum_{p \mid m} \log^t \left(p^{\nu_p(m)} \right) \in O(\log^t m)$$

对于每个正整数 m 和每个实数指数 $t \geq 1$。例如，对于一次“原根查询”，我们需要对每个整除 m 的奇素数 p，求出模 $p^{\nu_p(m)}$ 的一个原根。我们先对 m 进行因数分解，然后应用引理 5.1.6(5) 中的算法。根据引理 5.1.6(1,5)，这一过程所需的位运算次数为

$$O\left(\log^{7+o(1)} m + \sum_{2<p|m} \log^{7+o(1)} p\right) \subseteq O\left(\log^{7+o(1)} m + \sum_{p|m} \log^{7+o(1)}\left(p^{\nu_p(m)}\right)\right)$$
$$= O(\log^{7+o(1)} m) \subseteq O(\log^{7+o(1)} y(\vec{\mathfrak{r}})) \subseteq O(\log^{7+o(1)} h(y_1(\vec{\mathfrak{r}}), \ldots, y_n(\vec{\mathfrak{r}}))),$$

根据引理 5.1.5(6)，这也涵盖了对所有 p，从 p 和 $\nu_p(m)$ 计算 $p^{\nu_p(m)}$ 的成本。计算过程中所需的基本量子门数量以及比特-量子比特转换次数，也可以类似地处理。此外，类似的方法也适用于“MDL查询”和“MORD查询”，其中还需要对 m 进行因数分解（见引理 5.1.6(1)）；并且由于定义 5.1.3(4,a)中 $y(\vec{\mathfrak{r}})^2$ 对 m 的上界，最终在对数幂中会得到一个 $h(y_1(\vec{\mathfrak{r}}) \ldots, y_n(\vec{\mathfrak{r}}))^2$ 形式的参数，但可以将其替换为 $h(y_1(\vec{\mathfrak{r}}) \ldots, y_n(\vec{\mathfrak{r}}))$，而不改变整体表达式的 O- 类。

□

鉴于引理 5.1.7，从这里开始我们主要研究查询复杂度，只在一些主要结果中将其转换为拉斯维加斯对偶复杂度。为了运用本文发展的理论，解决本节开头提出的关于指标为 d 广义分圆映射f 的三个算法问题，我们首先需要计算诱导函数 $\overline{f}$: $\{0,1,\ldots,d\} \to \{0,1,\ldots,d\}$；此外，对于每个满足广义分圆形式 (1) 中 f 的系数 a_i 非零的 $i \in \{0,1,\ldots,d-1\}$，我们需要计算 $\mathbb{Z}/s\mathbb{Z}$ 的仿射映射 A_i，该映射编码了在通过引言中描述的双射 ι_j 将 C_j 与 $\mathbb{Z}/s\mathbb{Z}$ 对应起来后得到的限制 $f_{|C_i}$: $C_i \to C_{\overline{f}(i)}$。接下来的目标是分析这些任务的查询复杂度。

命题5.1.8。给定 f，可以计算诱导函数$\overline{f}$ 以及相关的仿射映射 A_i，其查询复杂度有 q 界

$$(d\log^{1+o(1)} q, d, 0, 0, 0).$$

证明。关于 $\overline{f}$，我们知道 $\overline{f}(d) = d$，因此只需计算 $i \in \{0,1,\ldots,d-1\}$ 的 $\overline{f}(i)$ 值。这些是 $O(d)$种情况。根据引言中的讨论，我们有 $\overline{f}(i) = (e_i + r_i i) \bmod d$，其中 $e_i = \log_\omega(a_i)$；根据本节开头的假设，该离散对数要么通过 a_i直接给出，要么通过一次（有限）域离散对数（fdl）查询进行计算。计算出 e_i 后，根据引理 5.1.5(1,3)，还需进行另一次 $O(\log^{1+o(1)} q)$ 位运算来计算$(e_i + r_i i) \bmod d$，从而计算出 $\overline{f}(i)$。总之，计算 $\overline{f}$ 的 q 有界查询复杂度为

$$(d\log^{1+o(1)} q, d, 0, 0, 0).$$

一旦确定了 $\overline{f}$，计算 A_i 就很容易；对于所讨论的每个 i 的 $O(d)$ 个取值，我们注意到，对所有 $x \in \mathbb{Z}/s\mathbb{Z}$，$A_i(x) = \alpha_i x + \beta_i$，其中

α_i、$\beta_i \in \mathbb{Z}/s\mathbb{Z}$ 是常数。计算 A_i 就是计算 α_i 和 β_i；根据引言中的讨论，我们有

$$\alpha_i = r_i, \quad \beta_i = \frac{e_i + r_i i - \overline{f}(i)}{d} \bmod s.$$

根据 f的定义 (1)，我们可以直接读出 r_i ；根据引理 5.1.5(1,3)，计算 β_i 需要$O(\log^{1+o(1)} q)$次位运算。因此，在求出 $\overline{f}$之后，计算所有 α_i 和β_i需要 $O(d\log^{1+o(1)} q)$ 次位运算，结论得证。

□

关于本节开头的问题 3，我们注意到，高效地解决这个问题，可以让我们快速了解给定的每个 Γ_f 连通分量。虽然若能有一种高效算法，对 Γ_f 的所有连通分量的同构类型作全局了解会更理想，但甚至不清楚这种算法的输出会是什么样子。在定义 5.3.2.8 中，我们引入了树项链列表的概念，它以列表形式列出 Γ_f 的所有连通分量的同构类型及其重数。在某些情况下（例如小小节 5.3.2 和 5.3.3 中讨论的情形），这样的树项链列表是 Γ_f 同构类型的一种紧凑编码，但尚不清楚是否总是如此；另请参见注 5.3.2.9 之后的讨论。

本节的以下主要结果给出了本节开头提到的三个算法问题的 q-有界查询复杂度和拉斯维加斯对偶复杂度。回顾一下，$\mathrm{mpe}(q-1) = \max_{p|q-1} \nu_p(q-1)$ 表示 $q-1$ 的素因数分解中素数的最大指数。

定理 5.1.9. 关于本节开头的三个算法问题，以下结论成立。

(1) 问题 1 的查询复杂度有界，为 q

$$(d^2 \log^2 d + d^2 \log^{1+o(1)} q + d\log^{2+o(1)} q, d, d\log q, d, 1)$$

以及 q-有界的拉斯维加斯双重复杂度

$$(d^2 \log^2 d + d^2 \log^{1+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q).$$

(2) 问题 2 具有 q 有界查询复杂度。

$$(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q, d, 0, 0, 0)$$

以及 q-有界的拉斯维加斯双重复杂度

$$(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q + d\log^{3+o(1)} q, d\log^{3+o(1)} q, d\log q).$$

此外，可以选择所计算的树划分寄存器，使其底层的递归树描述列表长度属于 $O(\min\{d2^{d^2 \operatorname{mpe}(q-1)+d}, q\})$，其中列表中的每个树描述本身都是一个长度属于

$$O(\min\{d2^{d^2 \operatorname{mpe}(q-1)+d}, q\}),$$

其中每个条目都是位长度为 $O(\log q)$ 的有序对。

(3) 问题 3 具有 q-有界查询复杂度

$$(d\log^{2+o(1)} q + d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d^3\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q,\\ d, d^3\,\mathrm{mpe}(q-1), d^3\,\mathrm{mpe}(q-1), 0)$$

以及 q 有界的拉斯维加斯双重复杂度

$$(d^3\,\mathrm{mpe}(q-1)\log^{7+o(1)} q + d^3\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q,\\ d^3\,\mathrm{mpe}(q-1)\log^{3+o(1)} q, d^3\,\mathrm{mpe}(q-1)\log q).$$

我们将在第 5.2 小节中证明定理 5.1.9。在此之前，我们进一步讨论定理 5.1.9 中三个算法问题的复杂度。一个显而易见的问题是，在陈述 (2) 和 (3) 中纳入参数 $\mathrm{mpe}(q-1)$ 会如何影响复杂度。乍一看，这似乎相当不利，因为一般而言，$\mathrm{mpe}(q-1) \le \lfloor \log_2(q-1) \rfloor \sim \log_2 q$，且当且仅当 q 是费马素数时，这一界可以达到。由于在定理 5.1.9 的陈述 (2) 和 (3) 中，$\mathrm{mpe}(q-1)$ 出现在以 2 为底的幂的指数中，这意味着在最坏情况下，问题 2 和问题 3 的给定复杂度相对于输入长度呈指数级增长（输入长度为 $O(d\log q)$），其中 d 固定。

尽管如此，事实证明，“在大多数情况下”，$\mathrm{mpe}(q-1)$实际上被一个适当大的常数所界定，正如下述结果所述。MathOverflow 用户“Dr. Pi”在回应第一作者发布于 MathOverflow[1]上的问题时，友好地指出了这一结果及其证明。

命题 5.1.10。存在绝对常数 $c_{\mathrm{mpe}} > 0$，使得对于所有 $x \ge 2$，都有

$$\left(\sum_{q\le x} 1\right)^{-1} \sum_{q\le x} \mathrm{mpe}(q-1) \le c_{\mathrm{mpe}}$$

其中变量 q 遍历素数幂。特别地，以下结论成立。

(1)对每个 $\epsilon > 0$，存在常数 $c_\epsilon > 0$，使得对于所有素数幂 q，除了所占比例小于 ϵ 的渐近部分之外，都有 $\mathrm{mpe}(q-1) < c_\epsilon$。(2) 令 $h : [0,\infty) \to [0,\infty)$为一个函数，使得 $h(x) \to \infty$，当 $x \to \infty$ 时。那么对于渐近几乎所有素数幂 q，都有 $\mathrm{mpe}(q-1) \le h(q)$。

证明。我们首先注意到，不超过 x 的真（即非素数）素数幂的数量渐近等价于 $2x^{1/2}/\log x$。事实上，不超过 x 的素数平方的数量为

$$\pi\left(x^{1/2}\right) \sim \frac{x^{1/2}}{\log\left(x^{1/2}\right)} = \frac{2x^{1/2}}{\log x}.$$

此外，满足 $k \ge 3$ 且 $k \le \lfloor \log_2 x \rfloor$的素数幂 $p^k \le x$，对于每个固定的 k，此类素数幂的数量至多为 $x^{1/k} \le x^{1/3}$。因此，所有

[1]参见 https://mathoverflow.net/questions/436134/average-value-of-the-prime-omega-function-omega-on-

不超过 x 的真素数幂数量为

$$\pi\left(x^{1/2}\right)+O\left(x^{1/3}\log x\right)\sim\frac{2x^{1/2}}{\log x}.$$

这意味着以下两点。
(1) 不超过 x 的所有素数幂的数量 $\sum_{q\le x}1$ 渐近等价于 $x/\log\ x$，与 $\pi(x)$ 相同。
(2) 在和式 $\sum_{q\le x}\mathrm{mpe}(q-1)$ 中，来自真素数幂的总贡献至多为

$$O\left(\log x\cdot\frac{2x^{1/2}}{\log x}\right)=O\left(x^{1/2}\right)\subseteq o\left(\sum_{q\le x}1\right).$$

因此，我们可以把重点放在来自素数的贡献 $\sum_{p\le x}\mathrm{mpe}(p-1)$ 上。对于每个 $v=1,2,\ldots,\lfloor\log_2 x\rfloor$，我们给出素数数量的一个 O 上界，这些素数 $p\le x$ 满足 $\mathrm{mpe}(p-1)=v$。对于 $v=1$，我们使用平凡上界

$$O(\pi(x))=O\left(\frac{x}{\log x}\right)=O\left(\frac{x}{2\log x}\right).$$

现在假设 $v\ge 2$。为了推导此类 v 的上界，我们采用 Montgomery 和 Vaughan 证明的 Brun-Titchmarsh 定理的更强形式 [51,定理 2]。该结果指出，对于 $\mathfrak{a}\in\mathbb{N}^{+}$、$\mathfrak{b}\in\mathbb{Z}$ 以及任意实数 $x>\mathfrak{a}$，满足 $p\equiv\mathfrak{a}\pmod{\mathfrak{b}}$ 的素数 $p\le x$ 的数量至多为

$$\frac{2x}{\phi(\mathfrak{a})\log(x/\mathfrak{a})}.$$

现在，一个素数 $p\le x$ 满足 $\mathrm{mpe}(p-1)=v$，模 $\mathfrak{p}^v$ 同余于1，其中 $\mathfrak{p}^v$ 是某个素数 $\mathfrak{p}<x^{1/v}$。若固定 $\mathfrak{p}^v\le x^{1/2}$，则布伦-蒂奇马什定理意味着，满足 $p\equiv 1\pmod{\mathfrak{p}^v}$ 的素数 $p\le x$ 的数量至多为

$$\frac{2x}{\phi(\mathfrak{p}^v)\log(x/\mathfrak{p}^v)}\le\frac{4x}{\mathfrak{p}^{v-1}(\mathfrak{p}-1)\log x}\in O\left(\frac{x}{\mathfrak{p}^v\log x}\right).$$

另一方面，如果 $x^{1/2}<\mathfrak{p}^v<x$，那么满足 $p\equiv 1\pmod{\mathfrak{p}^v}$ 且为 $p\le x$ 的素数个数至多为 $x/\mathfrak{p}^v<x^{1/2}$。由此可得，满足 $\mathrm{mpe}(p-1)=v=2$ 的素数 $p\le x$ 的个数为 in

$$O\left(\frac{x}{\log x}\cdot\sum_{\mathfrak{p}\le x^{1/4}}\frac{4}{\mathfrak{p}(\mathfrak{p}-1)}+x^{1/2}\cdot\frac{2x^{1/2}}{\log x}\right)=O\left(\frac{x}{\log x}\right)=O\left(\frac{x}{4\log x}\right)$$

并且，如果 $v>2$，该数在彩票娱乐

$$O\left(\frac{x}{\log x}\cdot\sum_{\mathfrak{p}\le x^{1/(2v)}}\frac{4}{\mathfrak{p}^{v-1}(\mathfrak{p}-1)}+x^{1/2}\cdot x^{1/3}\right)=O\left(\frac{x}{2^v\log x}\right).$$

总之，我们已经证明，对于每个 $v = 1$、2、.....、 $\lfloor \log_2 x \rfloor$，满足 $\mathrm{mpe}(p-1) = v$ 的素数$p \le x$ 的数量为 $O\left(x/(2^v \log x)\right)$，因此

$$\sum_{p \le x} \mathrm{mpe}(p-1) \in O\left(\sum_{v=1}^{\lfloor \log_2 x \rfloor} \frac{vx}{2^v \log x}\right) = O\left(\frac{x}{\log x} \sum_{v=1}^{\lfloor \log_2 x \rfloor} \frac{v}{2^v}\right) = O\left(\frac{x}{\log x}\right),$$

由此

$$\left(\sum_{q \le x} 1\right)^{-1} \sum_{q \le x} \mathrm{mpe}(q-1) \sim \left(\sum_{p \le x} 1\right)^{-1} \sum_{p \le x} \mathrm{mpe}(p-1)$$
$$\in O\left(\frac{1}{x/\log x} \cdot \frac{x}{\log x}\right) = O(1),$$

这就是该命题的主要陈述。第一个“特别是”陈述可由此轻易得出，只需注意到，量 $\left(\sum_{q \le x} 1\right)^{-1} \sum_{q \le x} \mathrm{mpe}(q-1)$ 是 $\mathrm{mpe}(q-1)$ 在素数幂 $q \le x$ 上的平均值。最后，第二个“特别是”陈述可由第一个轻易推出。

□

在应用方面，特征 2 的有限域尤其值得关注。据作者所知，目前尚无关于 $\mathrm{mpe}(2^v - 1)$ 在 $v \to \infty$ 时渐近行为的严格结果；不过，我们在表 7 中概述了 $v \in \{1, 2, \ldots, K\}$ 时 $\mathrm{mpe}(2^v - 1)$ 的最大值和平均值，其中 $K \in \{100, 200, \ldots, 1000\}$。这些结果是利用 GAP [22] 以及坎宁安项目[68]中的信息获得的。更具体地说，对于 $v \in \{929, 947, 991\}$，GAP 似乎难以对 $2^v - 1$ 进行因式分解；但查阅坎宁安因数分解表后可知，对于这三个值中的每一个 v，$\mathrm{mpe}(2^v - 1) = 1$。

K	最大$\{\mathrm{mpe}(2^v - 1)$: $1 \le v \le K\}$	$K^{-1}\sum_{v=1}^{K} \mathrm{mpe}(2^v - 1)$ 的四舍五入值
100	4	1.28
200	5	1.325
300	5	1.3267
400	5	1.3325
500	6	1.336
600	6	1.3383
700	6	1.3371
800	6	1.34
900	6	1.3389
1000	6	1.341

表 7: $\mathrm{mpe}(2^v - 1)$ 的最大值和平均值。

基于此，我们猜想，对于 $1 \le v \le x$，$\mathrm{mpe}(2^v - 1)$ 的平均值始终小于 2，参见猜想 6.1.1。

5.2 定理5.1.9的证明

我们详细描述了解决问题 1–3 的算法，并分析了它们的查询复杂度（定理 5.1.9 中规定的拉斯维加斯对偶复杂度可利用引理 5.1.7 直接得到）。我们提供的细节应足以使这些算法易于实现。在这三种情况下，我们首先需要计算$\overline{f}$以及仿射映射 A_i，根据命题 5.1.8，这需要查询复杂度 $(d\log^{1+o(1)} q, d, 0, 0, 0)$。我们假设在讨论每个具体问题开始时，这一步已经完成。每当需要对一个正整数进行因数分解时，我们都将其计入一次 mdl 查询（计入查询复杂度的第三项）。

5.2.1 证明陈述（1）

要完整详细地提供这一算法，需要使用相当多的记号。为方便读者，我们在本次讨论中首次出现这些记号的相应段落开头，以带下划线的形式列出本次讨论中新引入的记号名称，以及少数在前文引入但此后很少使用的记号名称。为了阅读流畅起见，这些带下划线的部分需要忽略。当然，这些记号也收录在附录的表 11 中。

$\underline{\overline{\mathcal{L}}.}$ Before computing $\mathcal{L}$ properly, we need to compute a CRL-list $\overline{\mathcal{L}}$ for $\overline{f}$. Because $\overline{f}$ can be any function $\{0,1,\dots,d\} \to \{0,1,\dots,d\}$ with $\overline{f}(d) = d$, we use a general, brute-force algorithm for this, assuming that the indices $i \in \{0,1,\dots,d\}$ are processed as non-negative integers in binary representation, with $\lfloor \log d \rfloor + 1$ digits each. Going through them to compute $\mathrm{im}(\overline{f})$ as a (not necessarily repetition-free) list of its elements takes $O(d \log d)$ bit operations, and checking whether $\mathrm{im}(\overline{f}) = \{0,1,\dots,d\}$ uses $O(d \log^2 d)$ bit operations, for sorting $\mathrm{im}(\overline{f})$ according to Lemma 5.1.5(10), then checking that there are no repeated entries. If $\mathrm{im}(\overline{f}) \neq \{0,1,\dots,d\}$, we continue by computing $\mathrm{im}(\overline{f}^2)$ and checking whether $\mathrm{im}(\overline{f}) = \mathrm{im}(\overline{f}^2)$, using another $O(d \log^2 d)$ bit operations, and so on. After $O(d)$ iterations of this, and thus after $O(d^2 \log^2 d)$ bit operations in total, we have found the periodic point set $\mathrm{per}(\overline{f})$ as the first iterated image $\mathrm{im}(\overline{f}^n)$ such that $\mathrm{im}(\overline{f}^{n+1}) = \mathrm{im}(\overline{f}^n)$. Finally, it takes another $O(d \log d)$ bit operations to compute $\overline{\mathcal{L}}$ through iteration of $\overline{f}$ on $\mathrm{per}(\overline{f})$ by brute force, and in the process, we can actually store each cycle of $\overline{f}$ in full, which will be useful shortly. In total, these computations require $O(d^2 \log^2 d)$ bit operations.

$\underline{\mathcal{L}_i, U_i, \mathrm{par}_i, Y_i}$。为计算 $\mathcal{L}$ 所需的参数化，我们遍历 $\in \overline{\mathcal{L}}$ 中的元素 (i,ℓ)，它们对应的 $\overline{f}$-循环为 $(i_0, i_1, \dots, i_{\ell-1})$，并计算限制 $f_{|U_i}$ 的 CRL 列表 $\mathcal{L}_i$ 的参数化，其中 $U_i = \bigcup_{t=0}^{\ell-1} C_{i_t}$。这样做是可行的，因为根据命题 3.1.1，$\mathcal{L}$ 只是这些 $\mathcal{L}_i$ 的（不相交）并集。具体而言，我们计算一个定义双射函数 $\mathrm{par}_i : Y_i \to \mathcal{L}_i$ 的公式，其中 Y_i 是依赖于 i 的"简单"集合。对于 $(i,\ell) = (d,1)$，它在处理其他数对 (i,ℓ) 的循环之外单独处理，我们有 $\mathcal{L}_d = \{(0_{\mathbb{F}_q}, 1)\}$，并令$Y_d := \{(\emptyset,\emptyset)\}$（为符合集合 Y_i 对 $i < d$ 的格式——这两个 $\emptyset$ 都应视为空元组），并定义 $\mathrm{par}_d(\emptyset,\emptyset) := (0_{\mathbb{F}_q}, 1)$。这只需 $O(\log d)$ 次位运算（而不是 $O(1)$，因为定义左侧的指标 d 必须明确写出）。

定义的一侧需要明确写出）。

$\mathfrak{r}_p, \mathcal{A}_i, \overline{\alpha}_i, \overline{\beta}_i$，奇素数 $', \mathcal{L}'_i, \mathfrak{P}_i$。接下来，我们通过一次 q有界的模离散对数（MDL）查询对 $s = (q-1)/d$ 进行因数分解（提醒读者，我们将因式分解归入 MDL 查询）。我们还通过一次 q有界的原根（PRT）查询，为 s 的每个奇素数因子 p 求出模 $p^{\nu_p(s)}$ 的原根 $\mathfrak{r}_p$。随后，我们遍历满足 $i < d$ 的元素 $(i, \ell) \in \overline{\mathcal{L}}$，并对其中每个元素执行以下操作。我们计算

$$\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}, \quad \mathcal{A}_i(z) = \overline{\alpha}_i z + \overline{\beta}_i.$$

这需要对已计算出的仿射映射进行 $O(d)$次乘法，其中每次根据引理 5.1.5(1,3) 使用公式的成本为$O(\log^{1+o(1)} q)$ 次位运算

$$(z \mapsto \alpha z + \beta)(z \mapsto \alpha' z + \beta') = (z \mapsto \alpha\alpha' z + \alpha'\beta + \beta').$$

因此，总体而言，计算 $\mathcal{A}_i$ 需要 $O(d \log^{1+o(1)} q)$ 次位运算。我们的下一个目标是计算 CRL 列表 $\mathcal{L}'_i$ 的参数化 $\mathrm{par}'_i : Y_i \to \mathcal{L}'_i$，该列表用于 $\mathcal{A}_i$；由此，只需将 par'_i 的图像中的所有第二项（周期长度）乘以因数 ℓ，即可得到 $\mathrm{par}_i : Y_i \to \mathcal{L}_i$。作为即将进行的遍历 s 的素因子的准备工作，我们初始化 $\mathfrak{P}_i := \emptyset$（最终，$\mathfrak{P}_i$ 将是一个列表，其中包含 s 的那些不整除 $\overline{\alpha}_i$ 的素因子）。我们还针对 s 的每个素因子 p 计算 $\mathrm{ord}_{p^{\nu_p(s)}}(\overline{\alpha}_i)$，这只需要一次 q 有界的乘法阶（mord）查询。

κ_p、$\overline{\mathcal{A}}_{i,p}$，参数$'_{i,p}$、$Y_{i,p}$、$\mathcal{L}'_{i,p}$接下来，我们遍历 p的所有素因子，并对其中每一个执行以下操作。首先，我们检查 $p \mid \overline{\alpha}_i$，如果是，则跳到 p的下一个值。否则，我们将 p 添加到 $\mathfrak{P}_i$中，然后从先前计算的 s 的因式分解中读出 $\kappa_p := \nu_p(s)$ 和 p^{κ_p}。随后，我们计算 $\overline{\mathcal{A}}_{i,p} := \mathcal{A}_i \bmod p^{\kappa_p}$（也就是说，我们计算 $\overline{\alpha}_i \bmod p^{\kappa_p}$ 和 $\overline{\beta}_i \bmod p^{\kappa_p}$）；根据引理 5.1.5(3)，这需要$O(\log^{1+o(1)} q)$次位运算。我们注意到，由于 p 不整除$\overline{\alpha}_i$，函数 $\overline{\mathcal{A}}_{i,p}$ 是 $\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$上的仿射置换，并且根据我们的表 2，可以读出一个紧凑参数化 参数$'_{i,p} : Y_{i,p} \to \mathcal{L}'_{i,p}$，它是 $\overline{\mathcal{A}}_{i,p}$的 $\mathcal{L}'_{i,p}$ 的 CRL 列表，其中所有指定的循环长度都已完全进行因式分解。具体细节见下方的表 8；该表中的每个编号行都对应于表 2 中具有相同编号的情形。下一段将介绍一些其他记号，这些记号用于表 2，并且必须在能够打印参数$'_{i,p}$的描述之前计算出来。

$\kappa_{i,p}, \mathfrak{n}_p, \mathfrak{p}_{p,k}, v_{p,k}, v'_{i,p,k}, v'_{i,p}, v''_{i,2}, \mathfrak{r}_p, \mathfrak{f}_{i,p}$。回想一下，$\nu_p^{(v)}(m) := \min\{\nu_p(m),\ v\}$，我们设定

$$\kappa_{i,p} := \nu_p^{(\kappa_p)}(\overline{\beta}_i) = \nu_p^{(\kappa_p)}(\overline{\beta}_i \bmod p^{\kappa_p}),$$

这可以使用 $O(\kappa_p) = O(\log p^{\kappa_p})$ 次通过 p 进行的整数除法来计算，其位操作成本为 $O(\log^{2+o(1)} p^{\kappa_p})$。如果 $p > 2$，接下来我们使用 $2 \in O(1)$ 次MORD查询，计算$p-1$ 的因式分解以及 $\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)$ 的因式分解。我们将这些因式分解具体写成如下形式：

$$p - 1 = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}}$$

and

$$\operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i) = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}} \cdot p^{v'_{i,p}}.$$

注意，指数 v'_i、$_p$、$_k$ 或 v'_i、$_p$ 可能为 0。另一方面，如果 $p = 2$（其中 $\mathfrak{n}_p = 0$），我们记 $\mathsf{ord}_{2^{\kappa_2}}(\overline{\alpha}_i) = 2^{v'_i}$,2，这与上文 $p > 2$ 的记号相符，而 $\mathsf{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i) = 2^{v''_i}$,2。通常，$v''_i$、$_2 = v'_i$,2，因为 $\mathsf{ord}_{2^{\kappa_2}}(\overline{\alpha}_i) = \mathsf{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i)$；但如果 $\overline{\alpha}_i \equiv \pm 1 \pmod{2^{\kappa_2}}$，那么 v'_i、$_2 \in \{0, 1\}$ 和 v''_i、$_2 = 1 - v_i$,2。最后，无论 $p > 2$ 是否成立，我们都检查 $\overline{\mathcal{A}}_i$、$_p$ 是否有不动点，即是否

$$\gcd(\overline{\alpha}_i - 1, p^{\kappa_p}) = \gcd((\overline{\alpha}_i \bmod p^{\kappa_p}) - 1, p^{\kappa_p}) \mid \overline{\beta_i} \bmod p^{\kappa_p},$$

这可以使用 $O(\log^{1+o(1)} p^{\kappa_p})$ 位运算完成。我们存储此信息，并且每当 $\overline{\mathcal{A}}_{i,p}$ 有一个不动点时，就通过命题 2.3.6中的公式计算出一个不动点，记作 $\mathfrak{i}_{i,p}$，这需要另外进行 $O(\log^{1+o(1)} p^{\kappa_p})$ 位运算。

$\underline{u, u', \vec{u}.}$ 现在我们可以给出 CRL 列表 $\mathcal{L}'_{i,p}$ （$\mathcal{A}'_{i,p}$）的双射参数化 $\mathsf{par}'_{i,p} : Y_{i,p} \to \mathcal{L}'_{i,p}$ 的表格定义。注意，集合 $Y_{i,p}$ 总是具有以下两种形式之一，这一点在后文中会很重要。

- $Y_{i,\ p}$ 是一个整数区间，其一般元素记为 u；或
- $Y_{i,\ p}$ 中的元素是整数对（u，u'），其中 u 遍历一个整数区间，并且对每个固定的 u 值，第二个元素 u' 也遍历一个整数区间。

为统一记号，在两种情况下，我们也可以将 $Y_{i,p}$ 中的一个元素记作 $\vec{u}$。例如，要推导表 8 第一种情况下的公式，我们采用表 2 中的第一种情况，其中 $v := \kappa_p$, $t := u$, $j := u'$, $a := \overline{\alpha}_i$ 以及 $b := \overline{\beta}_i$。由此可清楚看出，u 的取值范围与表 2 中 t 的取值范围相同。关于所声称的 u'取值范围，我们注意到

$$\frac{\phi(p^{\kappa_p})}{\operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)} = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k} - v'_{i,p,k}} \cdot p^{\kappa_p - 1 - v'_{i,p}}$$

and

$$\phi(p^{\kappa_p - u}) = \left((p-1)p^{\kappa_p - u - 1}\right)^{\delta_{[u < \kappa_p]}} = \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}} \cdot p^{\kappa_p - u - 1}\right)^{\delta_{[u < \kappa_p]}},$$

由此可以推得

$$\gcd\left(\frac{\phi(p^{\kappa_p})}{\operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)}, \phi(p^{\kappa_p - u})\right) = \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k} - v'_{i,p,k}}\right)^{\delta_{[u < \kappa_p]}} \cdot p^{\min\{\kappa_p - 1 - v'_{i,p}, \delta_{[u < \kappa_p]}(\kappa_p - u - 1)\}},$$

如所需。最后，第1种情形中周期长度（$\mathsf{par}'_{i,p}(\vec{u})$ 的第二项）的公式成立，因为

$$\frac{\phi(p^{\kappa_p - u})}{\gcd\left(\frac{\phi(p^{\kappa_p})}{\operatorname{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)}, \phi(p^{\kappa_p - u})\right)} = \frac{\left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}} \cdot p^{\kappa_p - u - 1}\right)^{\delta_{[u < \kappa_p]}}}{\left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k} - v'_{i,p,k}}\right)^{\delta_{[u < \kappa_p]}} \cdot p^{\min\{\kappa_p - 1 - v'_{i,p}, \delta_{[u < \kappa_p]}(\kappa_p - u - 1)\}}}$$

$$= \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}} \right)^{\delta_{[u<\kappa_p]}} \cdot p^{\delta_{[u<\kappa_p]}(\kappa_p-u-1-\min\{\kappa_p-1-v'_{i,p},\kappa_p-u-1\})}$$

$$= \left(\prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}} \right)^{\delta_{[u<\kappa_p]}} \cdot p^{\delta_{[u<\kappa_p]}\max\{v'_{i,p}-u,0\}}$$

表 8 中的其他情况也可以采用类似方法处理。为避免读者混淆，我们注意到，在表 2 的情况7和情况 8 中，指定的 CRL列表由若干具有不同公式的互不相交部分组成。由于我们希望 u 遍历一个整数区间，因此这里对这些部分进行了略微重新排列，并将它们“拼接”在一起。例如，在这里的情况 7中， u的范围 $\{0,1,\ldots,\kappa_2-2\}$ 和 $\{-\kappa_2+1,-\kappa_2+2,\ldots,-1\}$分别对应于表2情况7中代表元为 $5^j2^t+\mathfrak{f}$和$-5^j2^t+\mathfrak{f}$ 的部分。在后一个区段中， u的范围不等于表2 中对应的 t 范围，这解释了变量替换 $u \to -u-1$ ，尽管表2 中对应的周期长度公式相同。在情况 10中，我们使用 1（而不是 0）作为 u的可接受值，同样是为了将 u 的范围转换为整数区间。

No.	u	u'	$\mathrm{par}'_{i,p}(\vec{u})$
1	$0,\ldots,\kappa_p$	$0,\ldots,p^{\min\{\kappa_p-1,\ v'_{i,p}+\delta_{[u<\kappa_p]}(\kappa_p-u-1)\}}-1$	$(\mathfrak{r}_p^{u'}p^u+\mathfrak{f}_{i,p},\ \left(\prod_{k=1}^{\mathfrak{n}_p}\mathfrak{p}_{p,k}^{v'_{i,p,k}}\right)^{\delta_{[u<\kappa_p]}}\cdot p^{\delta_{[u<\kappa_p]}\max(v'_{i,p}-u,0)})$
2	$0,\ldots,p^{\kappa_{i,p}}-1$	n/a	$(u,\ p^{\kappa_p-\kappa_{i,\ p}})$
3	$0,\ldots,2^{\kappa_{i,2}}-1$	n/a	$(u,2^{\kappa_2-\kappa_{i,2}})$
4	$0,1,2$	n/a	$(u,-u^2+2u+1)$
5	$0,1,2$	n/a	$(2^u-1,\frac{1}{2}u^2-\frac{3}{2}u+2)$
6	$0,1$	n/a	$(2u,2)$
7	$-\kappa_2,\ldots,\kappa_2-1$	若 $u\in\{-\kappa_2,\kappa_2-1\}$: 0; 否则: $0,\ldots,2^{\kappa_2-2-\text{最大值}(v'_{i,2},u)}-1$	若 $u=\kappa_2-1$: $(\mathfrak{f}_{i,\ 2},\ 1)$; 若 $u=-\kappa_2$: $(2^{\kappa_2-1}+\mathfrak{f}_{i,\ 2},\ 1)$; 如果 $0\le u<\kappa_2-1$: $(5^{u'}2^u+\mathfrak{f}_{i,2},2^{\max(v'_{i,2}-u,0)})$; 如果 $-\kappa_2<u<0$: $(-5^{u'}2^u+\mathfrak{f}_{i,2},2^{\max(v'_{i,2}+u+1,0)})$.
8	$-v''_{i,2},\ldots,v''_{i,2}$	if $u=v''_{i,2}$: $0,\ldots,2^{\kappa_2-v''_{i,\ 2}-1}$; 否则: $0,\ldots,2^{\kappa_2-v''_{i,2}-2}-1$.	如果 $u=v''_{i,2}$ 且 $u'\in\{0,2^{\kappa_2-v''_{i,2}-1}\}$: $(u'2^{v''_{i,2}}+\mathfrak{f}_{i,2},1)$; 如果 $u=v''_{i,2}$ 且 $0<u'<2^{\kappa_2-v''_{i,\ 2}-1}$: $(u'2^{v''_{i,\ 2}}+\mathfrak{f}_{i,2},2)$; 如果 $0\le u<v''_{i,2}$: $(5^{u'}2^u+\mathfrak{f}_{i,2},2^{v''_{i,2}-u})$; 如果 $u<0$: $(5^{u'}2^{-u-1}+\mathfrak{f}_{i,\ 2},2^{v''_{i,\ 2}+u+1})$
9	$0,\ \ldots,\ 2^{\kappa_{i,2}}-1$	n/a	$(u,2^{\kappa_2-\kappa_{i,2}})$
10	$1,\ldots,2^{\kappa_2-v''_{i,2}-1}$	n/a	如果 $u=1$:$(0,2^{v''_{i,\ 2}+1})$; 否则: $(\overline{\beta}_i u,2^{v''_{i,2}+1})$.

表 8：有限主循环群的仿射置换的显式参数化 CRL 列表。

我们的算法为所有满足 $p \nmid \overline{\alpha}_i$ 的素数 $p \mid s$ 打印并存储$\mathrm{par}'_{i,p}(\vec{u})$ 的参数化描述。对于给定的 p，此参数化描述需要 $O(\log p^{\kappa_p})$ 位来存储（这是因为打印 $n \in \mathbb{N}^+$ 的素因数分解需要 $O(\log n)$ 位），因此所有描述合起来可以使用$O(\log s) \subseteq O(\log q)$ 位来存储。

$\mathrm{par}''_i, \mathcal{L}''_i, \mathcal{A}'_i, s'_i, \vec{\overline{u}}, \vec{u}_p, u_p, u'_p, \overline{Y}_i, \mathrm{proj}_j, \vec{r}_i(\vec{\overline{u}})$ $r_{i,p}(\vec{u}_p)$ $B_{\vec{r}_i(\vec{\overline{u}})}$ $\mathcal{I}_{i,\vec{\overline{u}}}$. Next, based on the parametrizations $\mathrm{par}'_{i,p}$ of the CRL-lists $\mathcal{L}'_{i,p}$ of $\overline{\mathcal{A}}_{i,p}$ for $p \in \mathfrak{P}_i$, we construct a parametrization $\mathrm{par}''_i : Y_i \to \mathcal{L}''_i$ of a CRL-list $\mathcal{L}''_i$ for $\mathcal{A}'_i := \mathcal{A}_i \bmod \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$. We start by setting $s'_i := \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$, which takes $O(\log^{2+o(1)} q)$ bit operations to compute, carrying out $|\mathfrak{P}_i| \in O(\log q)$ integer multiplications, each with a bit operation cost in $O(\log^{1+o(1)} q)$ (we note that the powers p^{κ_p} themselves do not need to be computed, as they are specified, alongside the pairs (p, κ_p), in the output of the mdl query that gave the factorization of s). We follow the approach described at the end of Subsection 2.3. More specifically, we identify $\mathbb{Z}/s'_i\mathbb{Z}$ with $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$, and $\mathcal{A}'_i$ with $\bigotimes_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}$. We consider tuples $\vec{\overline{u}} = (\vec{u}_p)_{p\in\mathfrak{P}_i} \in \prod_{p\in\mathfrak{P}_i} Y_{i,p} =: \overline{Y}_i$. We can either write $\vec{u}_p = u_p$ or $\vec{u}_p = (u_p, u'_p)$. For $j = 1, 2$, we denote by proj_j the (class-sized) function that maps an ordered pair to its j-th entry. Associated with each parameter tuple $\vec{\overline{u}} \in \overline{Y}_i$, we have the tuple

$$\vec{r}_i(\vec{\overline{u}}) := (r_{i,p}(\vec{u}_p))_{p\in\mathfrak{P}_i} := (\mathrm{proj}_1(\mathrm{par}'_{i,p}(\vec{u}_p)))_{p\in\mathfrak{P}_i}$$

相关周期代表元的$\overline{\mathcal{A}}_{i,p}$。根据我们在第2.3 节末尾的讨论，这些元组 $\vec{r}_i(\vec{\overline{u}})$参数化了某个划分的块 $B_{\vec{r}_i(\vec{\overline{u}})} \prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$，其中每个块都是 $\mathcal{A}'_i$ 的循环的并集。就 $\vec{\overline{u}}$ 而言，我们希望明确描述将 $\mathcal{A}'_i$ 限制到 $B_{\vec{r}_i(\vec{\overline{u}})}$ 后的CRL列表。为此，我们需要展示一个 $\vec{r}_i(\vec{\overline{u}})$-可容许索引函数 $\mathcal{I}_{i,\vec{\overline{u}}}$ （其含义见定义2.3.8(1)），并理解其相关的良好元组集合（其含义见定义 2.3.8(2)）。

$l_{i,p,\vec{u}_p}, l_{i,\vec{\overline{u}},\mathfrak{P}'_i}$。现在，根据可容许索引函数的定义，$\mathcal{I}_{i,\vec{\overline{u}}}$ 的定义域是所有至少整除一个分量周期长度的素数的集合$l_{i,p,\vec{u}_p} := \mathrm{proj}_2(\mathrm{par}'_{i,p}(\vec{u}_p))$，其中 $p \in \mathfrak{P}_i$；等价地，它们整除$l_{i,\vec{\overline{u}}} := \mathrm{lcm}\{l_{i,p,\vec{u}_p} : p \in \mathfrak{P}_i\}$。我们通过扫描所展示的$l_{i,p,\vec{u}_p}$的参数因式分解，计算其位长度为 $O(\log^{1+o(1)} q)$的参数化定义，以供后续使用；这一过程需要 $O(\log^{2+o(1)} q)$次位操作。根据定义，$\mathcal{I}_{i,\vec{\overline{u}}}$ 的定义域是$\mathfrak{P}'_i := \mathfrak{P}_i \cup \pi(\prod_{p\in\mathfrak{P}_i}(p-1))$的子集。我们将$\mathfrak{P}'_i$ 计算为一个列表（含有 $O(\log q)$个条目）；这项计算包括 $O(\log q)$次包含性检查，每次检查涉及 $O(1)$次复制过程，复制长度为 $O(\log q)$的比特串，以及$O(\log q)$次位比较和内存地址扫描，每个地址的长度为 $O(\log\log q)$。因此，我们可以使用 $O(\log^{2+o(1)} q)$次位操作来计算$\mathfrak{P}'_i$ 。在我们的算法方法中，我们将$\mathcal{I}_{i,\vec{\overline{u}}}$ 视为一个函数，其定义域为整个 $\mathfrak{P}'_i$;

额外的素数 $\mathfrak{p}$ 是那些不整除任何分量周期长度的素数，因此在每个分量中的赋值均为 0，而值 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$可以在 $\mathfrak{P}_i$中任意选取。我们注意到，这一改变不会影响良好元组的相关概念，并确保$\mathcal{I}_{i,\vec{u}}$ 的定义域不依赖于 $\vec{u}$。

对于每个 $\mathfrak{p} \in \mathfrak{P}'_i$，值 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 是一个素数 $p' \in \mathfrak{P}_i$（可视为 $\vec{u}$ 的一个分量的指标），使得 $\nu_{\mathfrak{p}}(l_{i,p',\vec{u}})$ 在所有满足 $p \in \mathfrak{P}_i$ 的 $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ 中取最大值。我们回顾一下，我们已经求出了正整数 $l_{i,p,\vec{u}_p}$ 关于 $\vec{u}_p$ 的显式因式分解（见表 8）。如果 $\mathfrak{p} \notin \mathfrak{P}_i$，那么对于每个 $p \in \mathfrak{P}_i$，$\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ 的值都是常数，不依赖于 $\vec{u}_p$。沿长度为 $O(\log q)$ 的参数化描述进行扫描，并比较相关指数 $\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$（每个指数的位长度为 $O(\log \log q)$），即可选出适合 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 的值。对于给定的 $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$，这一过程需要 $O(\log^{1+o(1)} q)$ 次位操作；对所有 $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$ 执行这一过程则需要 $O(\log^{2+o(1)} q)$ 次位操作。

$\mathcal{J}_{i,p,k}, \mathfrak{m}_{i,p}, \mathcal{J}'_{i,p,k}$。我们仍需讨论 $\mathfrak{p} \in \mathfrak{P}_i$ 时的处理方法。即便如此，$\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})$ 并不依赖于 $\vec{u}_p$，除非 $p = \mathfrak{p}$，在这种情况下，以下情形之一适用。

- $\nu_p(l_{i,p,\vec{u}})$ 也不依赖于 $\vec{u}_p$（例如参见表 8 中的情形 2），因此我们可以按照上述说明计算 $\mathcal{I}_{i,\vec{u}}(p)$ 的一个常数值。
- $\nu_p(l_{i,p,\vec{u}})$确实取决于 $\vec{u}_p$，具体如下：它只取决于u_p（而非u'_p），并且可以将u_p 的取值范围划分为至多五个子区间$\mathcal{J}_{i,p,1}, \ldots, \mathcal{J}_{i,p,\mathfrak{m}_{i,p}}$（表 8 中的情况 8 确实需要 $\mathfrak{m}_{i,p} = 5$），使得对于固定的 $k \in \{1, \ldots, \mathfrak{m}_{i,p}\}$，在情况$u_p \in \mathcal{J}_{i,p,k}$下，$\nu_p(l_{i,p,\vec{u}})$ 的值要么为常数，要么由关于u_p的线性表达式给出，要么由关于u_p 的线性表达式与 0 中的较大者给出。这样，我们便能指定$\mathcal{J}_{i,p,k}$的一个子区间（事实上是一个初始段或末尾段）$\mathcal{J}'_{i,p,k}$（其边界点为常数），使得若$u_p \in \mathcal{J}'_{i,p,k}$，则$\mathcal{I}_{i,\vec{u}}(p)$可取为 p；而若$u_p \in \mathcal{J}_{i,p,k} \setminus \mathcal{J}'_{i,p,k}$，则$\mathcal{I}_{i,\vec{u}}(p)$必须取 $\mathfrak{P}_i$ 中的另一个常数值（对每个k都相同）。对于给定的每个 p，写出$\mathcal{I}_{i,\vec{u}}(p)$的显式定义（其中包含至多两种情况的分类讨论），需要我们扫描分量像的参数化描述 $\mathrm{par}'_{i,p}(\vec{u}_p)$，并执行一些低成本的计算，例如对素数的指数进行加减运算（这些指数是位长度为 O的数字，其中 O 为 $\log \log q$）。对于所有相关的 $\mathfrak{p}$ 取值，这可以用 $O(\log^{2+o(1)} q)$次位运算完成。

我们注意到，对于每个给定的 $\mathfrak{p} \in \mathfrak{P}'_i$，我们刚刚推导出的 $\mathcal{I}_{i,\,\vec{u}}$ 的参数化定义的位长度为 $O(\log q)$。因此，由于 $\mathcal{I}_{i,\,\vec{u}}$的定义域大小为 $O(\log q)$，存储 $\mathcal{I}_{i,\,\vec{u}}$所有函数值的参数化定义需要 $O(\log^2 q)$ 位。

$\mathcal{I}_i, \mathfrak{P}_{i,p,\vec{u}}, \vec{k}, k_p, \mathfrak{d}_{i,p,\vec{u}}, \vec{k}', k'_p, K'_{i,\vec{u}}$。在继续论证之前，我们需要引入另一个记号。对于 $\mathfrak{p} \in \mathfrak{P}'_i$，如果 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 当$\vec{u}$ 遍历 $\overline{Y}_i$ 时只取一个不同的值，我们令 $\mathcal{I}_i(\mathfrak{p}) := \mathcal{I}_{i,\vec{u}}(\mathfrak{p})$，其中 $\vec{u} \in \overline{Y}_i$ 为任意值。另一方面，如果 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 取两个不同的值，其中一个是 $\mathfrak{p}$，我们令 $\mathcal{I}_i(\mathfrak{p})$ 为唯一的元素

$\{\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 中与 $\mathfrak{p}$ 不同的 $\vec{u} \in \overline{Y}_i\}$。这定义了一个函数 $\mathcal{I}_i\colon \mathfrak{P}'_i \to \mathfrak{P}_i$，它独立于 $\vec{u}$，且可以从函数值 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 的参数化定义中轻松推导出来（需要 $O(\log^{2+o(1)} q)$ 次位操作）。利用函数 $\mathcal{I}_i$，我们可以给出 $\mathfrak{P}_i$ 的单元素子集在 $\mathcal{I}_{i,\vec{u}}$ 下原像的以下紧凑参数化定义：

$$\mathfrak{P}_{i,p,\vec{u}} := \mathcal{I}_{i,\vec{u}}^{-1}(\{p\}) = \left(\mathcal{I}_i^{-1}(\{p\}) \setminus \{\mathfrak{p} \in \mathfrak{P}_i \setminus \{p\} : u_{\mathfrak{p}} \in \bigcup_{k=1}^{\mathfrak{m}_{i,\mathfrak{p}}} \mathcal{J}'_{i,p,k}\}\right) \cup \{\mathfrak{p} \in \{p\} : u_{\mathfrak{p}} \in \bigcup_{k=1}^{\mathfrak{m}_{i,\mathfrak{p}}} \mathcal{J}'_{i,p,k}\}. \quad (23)$$

我们注意到，我们的算法只是为每个 $p \in \mathfrak{P}_i$ 生成这个定义公式 $\mathfrak{P}_{i,\ p,\ \vec{u}}$，这在复杂度方面无伤大雅——即便在每种情况下都显式写出 $\mathcal{I}_i^{-1}(\{p\})$，每个 p 也只需 $O(\log^{1+o(1)} q)$ 次位运算和存储空间，因此总共需要 $O(\log^{2+o(1)} q)$ 次位运算和存储空间。也可以尝试为 $\mathfrak{P}_{i,\ p,\ \vec{u}}$ 提供分类定义，其中每种情况对应 $\mathfrak{P}_{i,\ p,\ \vec{u}}$ 的一个常数值，但这会破坏复杂度，因为一般需要遍历 $2^{O(\log q)}$ 种情况。同样，容易验证，只要足够谨慎地写出这些参数化，生成本论证其余部分所述的每个参数化定义都需要 $O(\log^{2+o(1)} q)$ 次位运算。

明确给出原像 $\mathfrak{P}_{i,\ p,\ \vec{u}}$ 十分重要，因为我们需要借此建立 $\mathcal{I}_{i,\ \vec{u}}$-良好元组的参数化。回顾上文的记号 $l_{i,\ p,\ \vec{u}_p}$，它表示 $r_{i,\ p}(\vec{u}_p)$ 在 $\vec{r}_i(\vec{u})$ 中由 p 指标标记的分量里的代表元的周期长度。$\mathcal{I}_{i,\ \vec{u}}$-良好元组是满足 $k_p \in \mathbb{Z}/l_{i,\ p,\ \vec{u}_p}\mathbb{Z} = \{0, 1, \ldots, l_{i,p,\vec{u}_p} - 1\}$ 的元组 $\vec{k} = (k_p)_{p\in\mathfrak{P}_i}$，其中 k_p 可被整除

$$\mathfrak{d}_{i,p,\vec{u}} := \prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,\vec{u}})}.$$

我们可以用 $O(\log^{1+o(1)} q)$ 次位运算和存储空间计算 $\mathfrak{d}_{i,\ p,\ \vec{u}}$ 及 $l_{i,\ p,\ \vec{u}_p}/\mathfrak{d}_{i,\ p,\ \vec{u}}$ 的参数化定义（每个 p），因此总共需要 $O(\log^{2+o(1)} q)$ 次位运算和存储空间。此外，我们可以如下对 $\mathcal{I}_{i,\ \vec{u}}$-良好元组的集合进行参数化：

$$\mathrm{Good}_{\vec{r}_i(\vec{u})}(\mathcal{I}_{i,\vec{u}}) = \{(k'_p\mathfrak{d}_{i,p,\vec{u}})_{p\in\mathfrak{P}_i} : \vec{k'} = (k'_p)_{p\in\mathfrak{P}_i} \in \prod_{p\in\mathfrak{P}_i} \mathbb{Z}/\frac{l_{i,p,\vec{u}_p}}{\mathfrak{d}_{i,p,\vec{u}}}\mathbb{Z} =: K'_{i,\vec{u}}\}.$$

$r'_{i,\ \vec{u}}(\vec{k'})$。现在，对于每个 $\vec{k'} = (k'_p)^{p\in\mathfrak{P}_i} \in K'_{i,\ \vec{u}}$ 及其对应的 $\mathcal{I}_{i,\ \vec{u}}$-优良元组 $(k'_p\mathfrak{d}_{i,\ p,\ \vec{u}})_{p\in\mathfrak{P}_i}$，其周期代表元为 $\left(\overline{\mathcal{A}}_{p,\ \ i}^{k'_p\mathfrak{d}_{i,\ p,\ \vec{u}},\ ,\ }(r_{i,\ p}(\vec{u}_p))\right)_{p\in\mathfrak{P}_i}$，它属于 $\mathcal{A}'_i$；更确切地说，它是与之对应的置换 $\bigotimes_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,\ p}$ 在 $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ 中的周期代表元。严格来说，$\mathcal{A}'_i$ 定义为 $\mathbb{Z}/s'_i\mathbb{Z} = \mathbb{Z}/\prod_{p\in\mathfrak{P}_i} p^{\kappa_p}\mathbb{Z}$ 的一个仿射置换。因此，与 $\vec{k'}$ 对应的 $\mathcal{A}'_i$ 的实际周期代表元是

$$r'_{i,\vec{u}}(\vec{k'}) := \sum_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))\frac{s'_i}{p^{\kappa_p}}\,\mathrm{inv}_{p^{\kappa_p}}\left(\frac{s'_i}{p^{\kappa_p}}\right),$$

$\mathbb{Z}/s_i'\mathbb{Z}$ 中与 $\overline{\mathcal{A}}_{,}^{k_p'\mathfrak{d}_i}{}_{p\,\vec{u}\,,\,,\,p}(r_i,\,{}_p(\vec{u}_p))$ 模 p^{κ_p} 同余的唯一元素，对每个 $p \in \mathfrak{P}_i$ 都成立。我们注意到，表达式 $\overline{\mathcal{A}}_{,\quad i}^{k_p'\mathfrak{d}_i}{}_{p\,\vec{u}\,,\,,\,p}(r_i,\,{}_p(\vec{u}_p))$ 可以明确展开如下。

$$\overline{\mathcal{A}}_{i,p}^{k_p'\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p)) = \begin{cases} r_{i,p}(\vec{u}_p) + k_p'\mathfrak{d}_{i,p,\vec{u}}\overline{\beta}_i, & \text{if } \overline{\alpha}_i = 1, \\ \overline{\alpha}_i^{k_p'\mathfrak{d}_{i,p,\vec{u}}} r_{i,p}(\vec{u}_p) + \overline{\beta}_i \frac{\overline{\alpha}_i^{k_p'\mathfrak{d}_{i,p,\vec{u}}}-1}{\overline{\alpha}_i-1}, & \text{otherwise,} \end{cases}$$

其中，第二种情形中的分数应理解为整数除法，但整个表达式表示 $\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ 中的一个元素（需要将整数值与其模 p^{κ_p} 的约化相对应）。计算并存储 r_i', $\vec{u}$ 的参数化定义需要 $O(\log^{2+o(1)} q)$ 次位操作和相应的存储空间。

L_j。最后，现在我们可以给出 $\mathcal{L}_i''$ of$\mathcal{A}_i'$ 的 CRL 列表以及随后 $\mathcal{L}_i'$ of $\mathcal{A}_i$ 的 CRL 列表的参数化定义。具体而言，$Y_i := \bigcup_{\vec{u}\in\overline{Y}_i} \left(\{\vec{u}\} \times K'_{i,\vec{u}}\right)$；对于 $(\vec{u},\vec{k'}) \in Y_i$，我们定义

$$\mathrm{par}_i''(\vec{u},\vec{k'}) := (r'_{i,\vec{u}}(\vec{k'}), l_{i,\vec{u}}).$$

那么，$\mathcal{L}_i'' = \{\mathrm{par}_i''(\vec{u},\vec{k'}) : (\vec{u},\vec{k'}) \in Y_i\}$。为了得到 par_i' 和 $\mathcal{L}_i'$，我们只需将 $\mathcal{L}_i''$ 中各元素（par_i'' 的像）的首项从 $\mathbb{Z}/s_i'\mathbb{Z}$ 提升到 $\mathbb{Z}/s\mathbb{Z}$，使得每个提升模 s/s_i' 的约化，都是 $\mathbb{Z}/(s/s_i')\mathbb{Z}$ 中 $\mathcal{A}_i \bmod (s/s_i')$ 的唯一周期点。根据引理 2.1.14，我们可以如下计算该周期点。令 L_i 表示满足下列条件的最小非负整数

$$\gcd\left(\alpha_i^{L_i}, s\right) = \prod_{p|\gcd(\alpha_i,s)} p^{\kappa_p},$$

其满足

$$L_i = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i)} \right\rceil = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i \bmod p^{\nu_p(s)})} \right\rceil \le \mathrm{mpe}(s) \in O(\log q)$$

并且可以通过以下方式求得：对每个 $p \mid \gcd(\overline{\alpha}_i, s)$，计算值 $\overline{\alpha}_i \bmod p^{\nu_p(s)}$（对于$\nu_p(s)$，可查阅上面计算出的 s 的因式分解），然后在 0 和 $\nu_p(s)$ 之间通过二分查找求出 $\nu_p(\overline{\alpha}_i \bmod p^{\nu_p(s)})$（每一步都涉及一次幂运算和一次最大公因数计算）。总的来说，这需要

$$O\left(\log q \cdot \left(\log^{1+o(1)} q + \sum_{p|\gcd(\overline{\alpha}_i,s)} \left(\log\log q \cdot \log^{2+o(1)} p^{\nu_p(s)}\right)\right)\right) = O(\log^{2+o(1)} q)$$

根据引理 5.1.5(6,8)进行位运算。$\mathcal{A}_i$ 模（s/s_i'）的唯一周期点是.....的约化

$$\sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i = \begin{cases} L_i\overline{\beta}_i, & \text{if } \overline{\alpha}_i = 1, \\ \frac{\overline{\alpha}_i^{L_i}-1}{\overline{\alpha}_i-1}\overline{\beta}_i, & \text{otherwise,} \end{cases}$$

模 s/s'_i，并可利用 $\overline{\alpha}_i^{L_i}$的位长度为 $O(\log^2 q)$，在 $O(\log^{2(1+o(1))} q) = O(\log^{2+o(1)} q)$ 次位运算中计算得到。我们得到如下公式：对于 par'_i（其定义域为 Y_i，与par''_i 相同），使得 $\mathcal{L}'_i = \{\mathrm{par}'_i(\vec{u}, \vec{k'}) : (\vec{u}, \vec{k'}) \in Y_i\}$：

$$\mathrm{par}'_i(\vec{u}, \vec{k'}) = \left(r'_{i,\vec{u}}(\vec{k'}) \frac{s}{s'_i} \mathrm{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i s'_i \, \mathrm{inv}_{s/s'_i}(s'_i), l_{i,\vec{u}} \right).$$

打印这个参数化定义需要占用 $O(\log^{2+o(1)} q)$ 位存储空间。如前所述，$f_{|U_i}$ 的 CRL 列表$\mathcal{L}_i$ 的（双射）参数化 $\mathrm{par}_i : Y_i \to \mathcal{L}_i$，可以通过将 par'_i 的像中的第二个分量放大 $\ell = \ell_i$ 倍来得到（ i的 $\overline{f}$-循环长度）。也就是说，

$$\mathrm{par}_i(\vec{u}, \vec{k'}) = \left(r'_{i,\vec{u}}(\vec{k'}) \frac{s}{s'_i} \mathrm{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i-1} \overline{\alpha}_i^z \overline{\beta}_i s'_i \, \mathrm{inv}_{s/s'_i}(s'_i), \ell \cdot l_{i,\vec{u}} \right)$$

以及 $\mathcal{L}_i = \{\mathrm{par}_i(\vec{u}, \vec{k'}) : (\vec{u}, \vec{k'}) \in Y_i\}$.

最后，表达式 $\mathrm{par}_i(\vec{u}, \vec{k'})$，其中

- $i \in \mathrm{proj}_1(\overline{\mathcal{L}})$;
- $\vec{u} \in \overline{Y}_i$ (带有 $\overline{Y}_d := \{\emptyset\}$); 以及
- $\vec{k'} \in K'_{i,\vec{u}}$ (其中 $K'_{d,\emptyset} := \{\emptyset\}$)

构成所需的CRL列表 $\mathcal{L}$ 的双射参数化，可利用 $O(d \log^{2+o(1)} q)$ 次位运算将其从先前计算的结果中拼接得到。

接下来，我们将通过概述该算法各步骤来结束本小小节。在每个步骤的描述末尾，我们都会注明其查询复杂度（计算成本）；对于循环，如果适用，则逐分量计算循环各迭代步骤的查询复杂度在相应分量中的各项之和，将所得表达式替换为生成相同 O-类的更简单表达式，再乘以循环迭代次数的 O 上界。由此概述可知，问题 1 的查询复杂度如定理 5.1.9 的陈述（1）所述，而拉斯维加斯双重复杂度的公式则由此及引理 5.1.7 得出。

1 Compute the induced function $\overline{f}$ on $\{0, 1, \ldots, d\}$ and the affine maps A_i of $\mathbb{Z}/s\mathbb{Z}$.
QC: $(d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a CRL-list $\overline{\mathcal{L}}$ for $\overline{f}$, storing the cycles of $\overline{f}$ in full in the process.
QC: $(d^2 \log^2 d, 0, 0, 0, 0)$.

3 Compute and store the parametrization $\mathrm{par}_d : Y_d \to \mathcal{L}_d$ where $Y_d = \{(\emptyset, \emptyset)\}$ and $\mathrm{par}_d(\emptyset, \emptyset) = (0_{\mathbb{F}_q}, 1)$.
QC: $(\log d, 0, 0, 0, 0)$.

4 Compute and factor $s = (q-1)/d$.
QC: $(\log^{1+o(1)} q, 0, 1, 0, 0)$.

5 Find a primitive root $\mathfrak{r}_p$ modulo $p^{\nu_p(s)}$ for each odd prime $p \mid s$.
QC: $(\log q, 0, 0, 0, 1)$.

6 For each $(i, \ell) \in \overline{\mathcal{L}} \setminus \{(d, 1)\}$, with associated $\overline{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$ which was already computed in Step 2, do the following.
QC: $(d^2 \log^{1+o(1)} q + d \log^{2+o(1)} q, 0, d \log q, d, 0)$.

6.1 Compute the forward cycle product $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}} : z \mapsto \overline{\alpha}_i z + \overline{\beta}_i$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2 Initialize $\mathfrak{P}_i := \emptyset$.
QC: $(\log d, 0, 0, 0, 0)$.

6.3 Compute $\mathrm{ord}_{p^{\nu_p(s)}}(\overline{\alpha}_i)$ for each prime $p \mid s$.
QC: $(\log q, 0, 0, 1, 0)$.

6.4 For each prime $p \mid s$, do the following.
QC: $(\log^{2+o(1)} q, 0, \log q, 0, 0)$.

6.4.1 Check whether $p \mid \overline{\alpha}_i$, and if not, skip to the next p.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

6.4.2 Add p to $\mathfrak{P}_i$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

6.4.3 Read off $\kappa_p = \nu_p(s)$ and p^{κ_p} from the factorization of s computed in Step 4.
QC: $(\log q, 0, 0, 0, 0)$.

6.4.4 Compute $\overline{\mathcal{A}}_{i,p} = \mathcal{A}_i \mod p^{\kappa_p}$, i.e., compute $\overline{\alpha}_i \bmod p^{\kappa_p}$ and $\overline{\beta}_i \bmod p^{\kappa_p}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

6.4.5 Compute $\kappa_{i,p} := \nu_p^{(\kappa_p)}(\overline{\beta}_i) = \nu_p^{(\kappa_p)}(\overline{\beta}_i \bmod p^{\kappa_p})$, using $O(\kappa_p)$ divisions by p and increasing a counter.
QC: $(\log^{2+o(1)} p^{\kappa_p}, 0, 0, 0, 0)$.

6.4.6 If $p > 2$ then do the following.
QC: $(\log q + \log^2 p^{\kappa_p}, 0, 1, 0, 0)$.

6.4.6.1 Compute factorizations of $p - 1$ and of $\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i)$:

$$p - 1 = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v_{p,k}} \text{ and } \mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i) = \prod_{k=1}^{\mathfrak{n}_p} \mathfrak{p}_{p,k}^{v'_{i,p,k}} \cdot p^{v'_{i,p}}.$$

QC: $(\log q, 0, 1, 0, 0)$.

6.4.6.2 Check whether $\overline{\mathcal{A}}_{i,p}$ has a fixed point and, if so, store this information and compute a fixed point $\mathfrak{f}_{i,p}$ of it according to Proposition 2.3.6. The check can be done by testing whether $\gcd((\overline{\alpha}_i \bmod p^{\kappa_p}) - 1, p^{\kappa_p})$ divides $\overline{\beta}_i \bmod p^{\kappa_p}$.
QC: $(\log^{1+o(1)} p^{\kappa_p}, 0, 0, 0, 0)$.

6.4.7 Else do the following.

6.4.7.1 Compute $v'_{i,2} = \nu_2(\mathrm{ord}_{2^{\kappa_2}}(\overline{\alpha}_i))$ and $v''_{i,2} = \nu_2(\mathrm{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i))$. To avoid making another mord query, we note that $v''_{i,2} = v'_{i,2}$ unless $\overline{\alpha}_i \equiv \pm 1 \pmod{2^{\kappa_2}}$, in which case $v''_{i,2} = 1 - v'_{i,2}$.

计算成本：（$\log^{2+o(1)} 2^{\kappa_2}$，0，0，0，0）。6.4.7.2 检查 $\overline{\mathcal{A}}_{i,2}$ 是否有不动点；如果有，则存储该信息，并计算其一个不动点 $\mathfrak{f}_{i,2}$（参见步骤 6.4.6.2）。计算成本：（$\log^{1+o(1)} 2^{\kappa_2}$，0，0，0，0）。6.4.8 参照表 8，详细说明 CRL 列表 $\mathcal{L}'_{i,p}$ 的双射参数化 $\mathrm{par}'_{i,p} : Y_{i,p} \to \mathcal{L}'_{i,p}$，其中 $\overline{\mathcal{A}}_{i,p}$ 的所有指定周期长度都已完成因数分解。这需要检查表 2 中的哪种情形适用；我们已在步骤 6.4.6.2 和 6.4.7.2 中存储了与此相关的部分信息。注意，$Y_{i,p}$ 的一般元素记作 $\vec{u}_p$，并且它要么等于 u_p，要么等于 (u_p, u'_p)，其中 u_p 和 u'_p 是整数参数，u_p 在一个固定区间内取值，而对于 u_p 的每个固定值，u'_p 在一个区间内取值（区间边界关于 u_p 的显式公式已给出）。计算成本：（$\log\ p^{\kappa_p}$，0，0，0，0）。

6.5计算 $s'_i = \prod_{p\in\mathfrak{P}_i} p^{\kappa_p}$。计算成本: $(\log^{2+o(1)} q, 0, 0, 0, 0)$。

6.6计算 $l_{i,\ \vec{u}}$ 的参数化定义，即 $\mathcal{A}'_i$ 的周期长度（或者更确切地说，置换 $\bigotimes_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,\ p}$ 在与其对应的 $\prod_{p\in\mathfrak{P}_i} \mathbb{Z}/p^{\kappa_p}\mathbb{Z}$ 上的周期长度），其中点 $\vec{r}_i(\vec{u})$ 由 $\vec{u} = (\vec{u}_p)_{p\in\mathfrak{P}_i}$ 表示。这可以通过扫描完全进行因数分解的周期长度的参数化定义来完成

$$l_{i,p,\vec{u}_p} = \mathrm{proj}_2(\mathrm{par}'_{i,p}(\vec{u}_p)),$$

其中 $l_{i,\ \vec{u}}$ 是最小公倍数，并对位长度为 O（$\log\log\ q$）的数进行低计算成本运算。计算成本：$(\log^{2+o(1)} q, 0, 0, 0, 0)$。

6.7 使用 $O(\log q)$ 次包含性检查设置 $\mathfrak{P}'_i := \mathfrak{P}_i \cup \pi(\prod_{p\in\mathfrak{P}_i}(p-1))$，每次检查涉及 $O(1)$ 次复制长度为 $O(\log q)$ 的比特串的过程，以及 $O(\log q)$ 次位比较和对内存地址的扫描，每个内存地址的长度为 $O(\log\log q)$。计算成本：$(\log^{2+o(1)} q, 0, 0, 0, 0)$。

6.8 对于 $\mathfrak{p} \in \mathfrak{P}'_i$，求出函数值 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 的参数化定义，其中该函数是 $\vec{r}_i(\vec{u})$-可容许索引函数 $\mathcal{I}_{i,\vec{u}}$。此定义由至多两种情形的分类讨论构成（每种情形下 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 的值为常数）。对于 $\mathfrak{p} \notin \mathfrak{P}_i$，始终只有一种情形；对于 $\mathfrak{p} \in \mathfrak{P}_i$，则根据 $u_{\mathfrak{p}}$ 是否包含在若干特定区间的并集中来划分情形（每个 $\mathfrak{p}$ 至多对应五个这样的区间）。此外，只要有两种情形，其中一种就对应于 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p}) = \mathfrak{p}$。若只有一种情形，则对于任意给定的 $\vec{u}$，令 $\mathcal{I}_i(\mathfrak{p}) := \mathcal{I}_{i,\vec{u}}(\mathfrak{p})$；否则，令 $\mathcal{I}_i(\mathfrak{p})$ 为 $\mathcal{I}_{i,\vec{u}}(\mathfrak{p})$ 中异于 $\mathfrak{p}$ 的唯一元素。

计算成本：$(\log^{2+o(1)} q, 0, 0, 0, 0)$.

6.9 对于 $p \in \mathfrak{P}_i$，计算原像集 $\mathfrak{P}_{i,\ p,\ \vec{u}} := \mathcal{I}_i^{-1}$ 的参数化描述

$_{,\vec{u}}(\{p\})$，使用公式 (23)。在此参数化描述中，包含关系

素数 $\mathfrak{p} \in \mathfrak{P}'_i \setminus \mathfrak{P}_i$ 属于 $\mathfrak{P}_{i,\ p,\ \vec{u}}$ 与 $\vec{u}$无关，而素数 $p' \in \mathfrak{P}_i$各自都有一个条件，即根据第 6.8 步中提到的区间，对 $u_{p'}$ corre- 作出界限析取，以判定是否 $p' \in \mathfrak{P}_{i,\ p,\ \vec{u}}$。计算成本： $(\log^{2+o(1)} q, 0, 0, 0, 0)$。

6.10 基于步骤 6.9，计算参数化描述的

$$\mathfrak{d}_{i,p,\vec{u}} = \prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,\vec{u}})} = \prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})}$$

和 $l_{i,p,\vec{u}}/\mathfrak{d}_{i,p,\vec{u}}$ 对于每个 $p \in \mathfrak{P}_i$。鉴于步骤 6.9，可以在指数中使用适当的克罗内克δ来实现这一点。
计算成本: $(\log^{2+o(1)} q, 0,0,0, 0)$.

6.11 计算参数化描述

$$r_{i,\vec{u}}(\vec{k'}) = \sum_{p\in\mathfrak{P}_i} \overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p))\frac{s'_i}{p^{\kappa_p}}\operatorname{inv}_{p^{\kappa_p}}\left(\frac{s'_i}{p^{\kappa_p}}\right)$$

的周期代表元 $r_{i,\vec{u}}(\vec{k'})$ 的 $\mathcal{A}_{i'}$ ，与参数元组 $(\vec{u},\vec{k'})$ 相关联，其中 $\vec{u} \in \overline{Y}_i$ 且

$$\vec{k'} = (k'_p)_{p\in\mathfrak{P}_i} \in K'_{i,\vec{u}} = \prod_{p\in\mathfrak{P}_i} \mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{u}})\mathbb{Z}.$$

在此表达式中，应将（未显式给出的）仿射映射迭代值 $\overline{\mathcal{A}}^{k'_p\mathfrak{d}_{i,\ \ p\,\vec{u}\,,p}}_{i}(r_{i,p}(\vec{u}_p))$ 替换为显式公式

$$\overline{\mathcal{A}}_{i,p}^{k'_p\mathfrak{d}_{i,p,\vec{u}}}(r_{i,p}(\vec{u}_p)) = \begin{cases} r_{i,p}(\vec{u}_p) + k'_p\mathfrak{d}_{i,p,\vec{u}}\overline{\beta}_i, & \text{if } \overline{\alpha}_i = 1, \\ \overline{\alpha}_i^{k'_p\mathfrak{d}_{i,p,\vec{u}}} r_{i,p}(\vec{u}_p) + \overline{\beta}_i\frac{\overline{\alpha}_i^{k'_p\mathfrak{d}_{i,p,\vec{u}}}-1}{\overline{\alpha}_i-1}, & \text{otherwise.} \end{cases}$$

计算成本：$(\log^{2+o(1)} q, 0,0,0, 0)$。

6.12 计算

$$L_i = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i)} \right\rceil = \max_{p|\gcd(\overline{\alpha}_i,s)} \left\lceil \frac{\nu_p(s)}{\nu_p(\overline{\alpha}_i \bmod p^{\nu_p(s)})} \right\rceil,$$

满足最大公约数$(\overline{\alpha}_i^{L_i}, s) = \prod_{p|\text{最大公约数}(\overline{\alpha}_i,s)} p^{\kappa_p}$的最小非负整数。为此，对于每个 $p \mid$ 最大公约数$(\overline{\alpha}_i, s)$，通过一次除法计算 $\overline{\alpha}_i$ 对 $p^{\nu_p(s)}$ 取模，然后在 0 和$\nu_p(s)$ 之间使用二分查找找到 $\nu_p(\overline{\alpha}_i$ 对 $p^{\nu_p(s)})$。计算成本：$(\log^{2+o(1)} q, 0, 0, 0, 0)$

6.13 计算参数化描述

$$\operatorname{par}_i(\vec{u},\vec{k'}) = \left(r'_{i,\vec{u}}(\vec{k'})\frac{s}{s'_i}\operatorname{inv}_{s'_i}\left(\frac{s}{s'_i}\right) + \sum_{z=0}^{L_i-1} \overline{\alpha}_i^z\overline{\beta}_i s'_i \operatorname{inv}_{s/s'_i}(s'_i), \ell \cdot l_{i,\vec{u}} \right)$$

与 $(\vec{u},\vec{k'})$ 相关联的 $\mathcal{L}_i$ 中的元素（ $f_{|U_i}$ 的 CRL 列表，其中 $U_i = \bigcup_{t=0}^{\ell-1} C_{i_t}$）。
计算成本：$(\log^{2+o(1)} q, 0, 0, 0, 0)$。

7输出与 $(i,\vec{u},\vec{k'})$ 相关联的 $\mathcal{L}$（f 的CRL列表）中元素的参数化描述$\mathrm{par}_i(\vec{u},\vec{k'})$，其中 $i \in \mathrm{proj}_1(\mathcal{L})$，$\vec{u} \in \overline{Y}_i$且 $k' \in K'_{i,\vec{u}}$（约定 $\overline{Y}_d = \{\emptyset\}$ 且 $K'_{d,\emptyset} = \{\emptyset\}$），然后停止。计算成本：$(d\log^{2+o(1)} q, 0, 0, 0, 0)$。

5.2.2 命题（2）的证明

我们注意到，我们针对问题 2 的算法中唯一需要量子计算机的部分是在开始时，此时需要计算 $\overline{f}$ 和 A_i。算法的其余部分（我们将在下文中介绍）仅使用位运算。

除了计算 $\overline{f}$ 和 A_i，并且与陈述（1）的证明开头一样，我们还需要根据索引 $i \in \{0,1,\ldots,d-1\}$ 包含于 $\overline{f}$的迭代像中的情况，计算不同的“层”，总共需要 $O(d^2 \log^2 d)$ 次位运算。

我们沿用第 3.3 小节的方法，分三个连续步骤进行。步骤 1：暂态 i。我们的目标是计算

- $\mathcal{Z}_i = \mathcal{P}_i$对所有 $\overline{f}$-暂态 $i \in \{0,1,\ldots,d-1\}$;
- 一份有根树描述列表（$\mathfrak{D}_0$、$\mathfrak{D}_1$、...、$\mathfrak{D}_{N_1}$），涵盖形如 $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$的所有有根树同构类型，其中 $\overline{f}$-暂态 $i \in \{0,1,\ldots,d-1\}$ 和逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset,\neg\}^{m_i}$ 满足 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$；以及
- 相应的逻辑符号元组数据 $S_{n,i}$.

在算法的任意给定时刻（不仅仅是在这一步中），所有使 $\mathfrak{D}_n$ 有定义的 $n \in \mathbb{N}_0$ 构成 $\mathbb{N}_0$ 的一个初始段 $\{0,1,\ldots,N'\}$，记作 $\mathcal{N}$（这是一个在整个过程中不断更新的变量）。

为了完成上述列出的计算，我们对 $h_i = \mathrm{ht}(\text{树}^{\Gamma}\pi(i))$ 进行递归。首先，我们假设 $h_i = 0$。然后，根据第3.3小节，对于所有这样的 i，我们令 $\mathcal{P}_i := \mathfrak{P}(\emptyset)$，通过其描述 $\mathfrak{D}_0 := \emptyset$ 引入平凡有根树同构类型 $\mathfrak{I}_0$，并令 $S_{0,i} := \{\emptyset\}$（其中 $\emptyset$ 视为空逻辑符号元组）；而所有 $S_{n,i}$ 对于稍后引入的值 $n > 0$ 都将定义为空集。这就解决了 $h_i = 0$ 的情形。

现在我们假设 $h_i = h > 0$，并且所有 j 中带有 $h_j < h$的瞬态指标都已处理完毕。对于每个给定的 i，我们首先需要找到 i的$\overline{f}$-原像 j_1、j_2、.....、j_K，这需要 $O(d\log d)$次位运算。随后，我们为$\mathcal{P}_i = \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$计算一个生成同余序列。这涉及简单的算术运算（包括最大公因数计算），并且需要

$$O\left(\sum_{t=1}^{K} m_{j_t} \log^{1+o(1)} q\right) \subseteq O(d \log^{1+o(1)} q)$$

位运算。随后，我们按字典序遍历逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset,\neg\}^{m_i}$，检查 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$，若成立，则计算 $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 的紧凑描述 $\mathfrak{D}$。要检查块 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 是否非空，我们注意到，根据命题 3.3.2 之前的论证，其基数

$|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})|$ 等于分布数 $\sigma_{\mathcal{P}_i,0}(\vec{\nu}^{(\mathcal{P}_i)},(\emptyset,\ldots,\emptyset))$，其中 0 是常值零函数 $\mathbb{Z}/s\mathbb{Z} \to \mathbb{Z}/s\mathbb{Z}$。为了了解计算该分布数的成本，我们参考以下引理。

引理 5.2.2.1。设 $\mathcal{P}$ 为 $\mathbb{Z}/m\mathbb{Z}$的算术划分，由长度为 $k \in \mathbb{N}^+$的显式生成 m-同余式序列给出。此外，设 A为 $\mathbb{Z}/m\mathbb{Z} \to \mathbb{Z}/m\mathbb{Z}$上的仿射函数。则对于任意给定的长度分别为 k和 $k+1$的逻辑符号元组 $\vec{\nu}^{(\mathcal{P})}$ 和 $\vec{\nu}^{(\mathcal{P}')}$，计算单个分布数 $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$ 需要 $O(k2^k \log^{1+o(1)} m)$ 次位运算。

证明。根据引理 2.2.2 中的公式，计算 $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$ 需要将所有 $J \subseteq J_-(\vec{\nu}^{(\mathcal{P})})$ 对应的加项 $(-1)^{|J|}\kappa_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')}, J)$ 相加，而这类加项共有 $O(2^k)$ 个。计算其中一个加项包括

- 只需简单查找 $\vec{\nu}_{(\mathcal{P}')}$ 的最后一个分量（位操作成本：扫描相应内存地址需要 $O(\log k)$）；
- $O(k)$ 次整数整除检查，继最大公因数计算和减法之后进行，总位操作成本为 $O(k \log^{1+o_{(1)}} m)$，根据引理 5.1.5(1,3,8)；
- 检查两个子集 $J_+(\vec{\nu}_{(\mathcal{P})}) \cup J$和 $J_-(\vec{\nu}_{(\mathcal{P}')}) \ \{1, 2, \ldots, k\}$是否不相交，这涉及查找 $\vec{\nu}_{(\mathcal{P})}$ 和 $\vec{\nu}_{(\mathcal{P}')}$ 中的条目；如果使用指针，则总共需要 $O(k)$ 次位运算；以及

- 执行 $O(k)$次最大公因数计算、整数除法和最小公倍数计算，总复杂度为 $O(k\log^{1+o_{(1)}} m)$。

因此，计算所有求和项 $(-1)^{|J|}\kappa_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P}_{j_t})}\ \vec{\nu}^{(\mathcal{P}'_{j_t})}, J)$ 需要 $O(k2^k\log^{1+o(1)} m)$ 位运算，其成本高于将这些求和项相加的成本，因此也是计算 $\sigma_{\mathcal{P},A}(\vec{\nu}^{(\mathcal{P})}, \vec{\nu}^{(\mathcal{P}')})$ 的复杂度。 □

特别地，计算 $|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})|$，以检查 $\mathcal{P}_i$ 的该块是否为空，需要 $O(m_i 2^{m_i} \log^{1+o(1)} q) \subseteq O(d2^d \log^{1+o(1)} q)$ 次位运算。

现在假设 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 的结果非空。于是，我们希望计算 $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$ 的紧凑描述 $\mathfrak{D}$。为此，我们写作 $\vec{\nu}^{(\mathcal{P}_i)} =_{\diamond}^{K}_{t=1} \vec{\nu}^{(\mathcal{P}'_{j_t})}$，其中 $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$。根据命题 3.3.1，我们可以设

$$\mathfrak{D} := \left\{ \left(n, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t},A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \right) : n \in \mathcal{N} \right\} \setminus (\mathbb{N} \times \{0\}).$$

注意到，命题 3.3.1 中求和指标的取值范围包含一些逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_{j_t})}$，对于这些元组，$\mathcal{B}(\mathcal{P}_{j_t}, \vec{\nu}^{(\mathcal{P}_{j_t})})$ 为空，但可以忽略它们（本文即如此处理），因为所有对应的分布数 $\sigma_{\mathcal{P}_{j_t},A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}\ \vec{\nu}^{(\mathcal{P}'_{j_t})}\right)$ 均为 0。根据引理 5.2.2.1，计算单个分布数 $\sigma_{\mathcal{P}_{j_t},A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})}\ \vec{\nu}^{(\mathcal{P}'_{j_t})}\right)$ 需要

$$O(2^{m_{j_t}} m_{j_t} \log^{1+o(1)} q),$$

次位运算。注意到，对于每个固定的 $t \in \{1, 2, \ldots, K\}$，有

$$\bigcup \{S_{n,j_t} : n \in \mathcal{N}\} \subseteq \{\emptyset, \neg\}^{m_{j_t}},$$

最终得到的总位操作成本为

$$O\left(\sum_{t=1}^{K} (2^{m_t} \cdot 2^{m_{j_t}} m_{j_t} \log^{1+o(1)} q)\right) \subseteq O(d4^d \log^{1+o(1)} q)$$

用于计算 $\mathfrak{D}$，其成本上界是检查 $\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)}) \neq \emptyset$ 是否成立的成本。此外，如果计算 $\mathfrak{D}$ 时按递增顺序遍历数字 $n \in \mathcal{N}$，则表示 $\mathfrak{D}$ 的数组中的元素会按递增的 n 排列，这正是我们所期望的。

接下来，我们需要检查由 $\mathfrak{D}$ 描述的有根树 $\mathfrak{I}$ 是否出现在由 $\mathfrak{D}_n$ 描述的有根树 $\mathfrak{I}_n$ 中；这些树此前已经引入。此类树的数量至多等于所有 $\mathcal{P}_j$ 中不同（非空）块的总数，其中 j 是 $\overline{f}$-暂态，而该数量为 $O(\min\{d2^d, q\})$。由于每个 $\mathfrak{D}_n$ 以及 $\mathfrak{D}$ 都是按字典序排列的列表，长度为 $O(\min\{d2^d, q\})$，且每个条目都是长度为 $O(\log q)$ 的位串，因此检查是否存在某个 n 使得 $\mathfrak{D} = \mathfrak{D}_n$ 成立，需要 $O(d^2 4^d \log q)$ 次位运算。若成立，我们就将 $\vec{\nu}^{(\mathcal{P}_i)}$ 作为一个新元素添加到$S_{n,i}$ 中（添加到数组末尾，从而使该数组按字典序排列）。否则，我们将 $\mathfrak{D}$ 创建为一个新的树描述 $\mathfrak{D}_{n'}$，其中 $n' = \max \mathcal{N} + 1$，并初始化

$$S_{n',j} := \begin{cases} \{\vec{\nu}^{(\mathcal{P}_i)}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

对于给定的 $\overline{f}$-暂态 i，满足 $h_i = h$ 时，此循环需要

$$O(2^{m_i}(d4^d \log^{1+o(1)} q + d^2 4^d \log q))$$

位运算。现在，具有 $h_i = h$ 的不同索引 i 在$\overline{f}$下具有互不相交的迭代原像集，因此，对所有这类 i 而言，数值 m_i 的总和至多为 d。因此，我们得到的总位操作成本为

$$O(d8^d \log^{1+o(1)} q + d^2 8^d \log q)$$

用于处理所有满足 h_i 具有给定值的 i。总共处理所有 $\overline{f}$-暂态 i 需要

$$O(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q)$$

位运算。

步骤 2：$i = d$。现在，$\overline{f}$瞬态指标 i 已处理完毕，可以根据命题 3.3.2 计算 $\mathsf{Tree}^{\Gamma_f}(0_{\mathbb{F}_q})$ 的描述 $\mathfrak{D}$。这与步骤 1 中循环的一次迭代类似，需要 $O(d4^d \log^{1+o(1)} q)$ 次位运算。之后，我们检查 $\mathfrak{D}$是否出现在现有描述$\mathfrak{D}_n$（在步骤 1 中引入）中；与步骤 1 类似，这需要 $O(d^2 4^d \log q)$ 次位运算。如果是，则对相应的唯一项设置 $S_{n,\ d} := \emptyset$（正逻辑符号）

n，并对所有其他 m设置 $S_{m,\,d} := \neg$。如果不是，则将 $\mathfrak{D}$ 作为新的树描述引入$\mathfrak{D}_{n'}$，其中 $n' = \max\mathcal{N} + 1$，并对所有 $m < n'$设置 $S_{n',\,d} := \emptyset$ 和 $S_{m,\,d} := \neg$。此步骤的总体复杂度不超过步骤 1 的复杂度。

第 3 步：$\overline{f}$-周期的 $i < d$。最后，我们讨论如何处理 $\overline{f}$-周期的索引 $i \in \{0, 1, \ldots, d-1\}$。让我们使用第 3.3 小节第 3 步中的记号。例如，$(i_0, i_1, \ldots, i_{\ell-1})$ 是 $i = i_0$ 的 $\overline{f}$- 循环，并且我们有 $i_t = i_{t \bmod \ell}$，对于 $t \in \mathbb{Z}$ 以及 $i' = i_{-1}$ 也是如此。

我们首先计算 H_i，即在 Γ_f 中、形如C_{i_t}（$t \in \mathbb{Z}$）的陪集里，位于每个 $\overline{f}$-周期 i上方的最大树高。按照第3.3 小节中的论证，我们回顾 $H_i \le \ell\,\mathrm{mpe}(s) \le \ell\lfloor \log_2 s \rfloor$。此外，若对于固定的 $k \in \mathbb{Z}$，记 $h'_{i,k}$ 为满足以下条件的最小正整数

$$\gcd\left(\prod_{t=0}^{h'-1} \alpha_{i_{k-h'+t}}, s\right) = \gcd\left(\prod_{t=0}^{h'-2} \alpha_{i_{k-h'+t}}, s\right), \tag{24}$$

然后 $H_i = \max\{h'_{i,k} : k = 0, 1, \ldots, \ell-1\} - 1$。在进入循环遍历 k 以查找$h'_{i,\,k}$ 之前，我们计算 $\overline{\alpha}_i = \prod_{t=0}^{\ell-1} \alpha_{i_t}$ 模 s，需要进行 $O(\ell \log^{1+o(1)} s) \subseteq O(d \log^{1+o(1)} q)$ 次位运算。然后我们进入循环遍历 $k = 0, 1, \ldots, \ell-1$。对于每个 k，我们旨在通过在 1 和 $\ell\lfloor \log_2 s \rfloor + 1$ 之间的范围内进行二分查找，找到 $h'_{i,\,k}$ 的正确值。假设我们固定了一个试探值 h'。那么对于 $H' \in \{h'-1, h'-2\}$，我们有

$$\prod_{t=0}^{H'} \alpha_{i_{k-h'+t}} = (\overline{\alpha}_i)^{\lfloor (H'+1)/\ell \rfloor} \cdot \prod_{t=0}^{(H'+1) \bmod \ell} \alpha_{i_{k-h'+t}},$$

对于 H' 的两个取值，均可模 s 计算得出。

$$O(\log\log s \log^{1+o(1)} s + \ell \log^{1+o(1)} s) = O(\ell \log^{1+o(1)} s) \subseteq O(d \log^{1+o(1)} q)$$

位运算。接下来，我们计算并比较方程（24）中的两个最大公约数，这需要 $O(\log^{1+o(1)} s) \subseteq O(\log^{1+o(1)} q)$ 次位运算。该二分查找对每个 k 需要 $O(\log(\ell \log s)) \subseteq O(\log d + \log\log q)$ 次迭代，并且需要处理 k 的 $\ell \in O(d)$ 个值。总的来说，计算 H_i 需要

$$O(d \cdot (\log d + \log\log q) \cdot d \log^{1+o(1)} q) = O(d^2 \log d \log^{1+o(1)} q).$$

对每个单独的 i执行位运算，以及

$$O(d^3 \log d \log^{1+o(1)} q)$$

对所有 $\overline{f}$-周期的 $i < d$ 一起执行位运算。

在找到所有 H_i 后，我们旨在计算 $\mathcal{Z}_i = (\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$，对于每个 $\overline{f}$-周期的i。我们通过对所有 $\overline{f}$-周期的 i 一起依次计算 $\mathcal{X}_{i,h}$ 来完成这一点，计算范围为 $h = -1, 0, \ldots, \mathfrak{H} := \max\{H_i : i \in \mathrm{per}(\overline{f}) \setminus \{d\}\}$（对于 h 的每个固定值，我们跳过满足 $h > H_i$ 的那些 i）。现在，$\mathcal{X}_{i,-1} = (\theta_{i,h}(x))_{h=1,2,\ldots,H_i}$ 由 H_i 个同余式组成，并且根据$\theta_{i_t,h}(x)$ 的定义，这些同余式可以在每一步中使用简单的算术运算递归计算。根据 i，这需要

$$O((H_i+1)\log^{1+o(1)} q) \subseteq O((d\,\mathrm{mpe}(s)+1)\log^{1+o(1)} q) \subseteq O(d\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$$

位运算，并且对于所有 $\overline{f}$-周期的 i 一起，计算 $\mathcal{X}_{i,\,-1}$ 需要 $O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$ 次位操作。计算 $\mathcal{X}_{i,\,0}=\mathcal{R}_i$ 与计算 $\mathcal{Z}_j=\mathcal{P}_j$（针对 $\overline{f}$-暂态 j，见步骤 1）类似，每个 i 需要 $O(d\log^{1+o(1)} q)$ 次位操作，对于所有 $\overline{f}$-周期的 i 总共需要 $O(d^2\log^{1+o(1)} q)$。随后，生成同余序列为

$$\mathcal{X}_{i,h}=\lambda_{i_{-h}}^{h}(\mathcal{R}_{i_{-h}})=\lambda(\lambda_{i_{-h}}^{h-1}(\mathcal{R}_{i_{-h}}),A_{i_{-1}})=\lambda(\mathcal{X}_{i_{-1},h-1},A_{i_{-1}})$$

通过递归处理 $\mathcal{X}_{i_{-1},\,h-1}$来获得 $h=1$、2、.....、H_i，使用简单的算术运算；同样，对于全部i，每一步的复杂度为 $O(d^2\log^{1+o(1)} q)$。总体而言，在计算出每个 H_i 后，计算 $\mathcal{Z}_i$ 的位操作成本为

$$\begin{aligned}&O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q+(d\,\mathrm{mpe}(s)+1)\cdot d^2\log^{1+o(1)} q)\\ \subseteq&O(d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)\end{aligned}$$

对于所有 $\overline{f}$-周期的 $i<d$ 一起。

最后，我们需要

- 将第 1 步和第 2步中 $\mathfrak{D}_n$ 生成的有根树描述列表扩展为最终版本，该版本还包含形如 $\text{树}_{(h)}$ $\{0,1,\dots,H_i\}$ $\vec{\nu}^{\mathcal{P}_{i,h}}\in\{\emptyset,\neg\}^{n_{i_{-h}}+n_{i_{-h+1}}+\cdots+n_{i_0}}$ 的所有有根树同构类型的描述，其中 为 周期长度，$,_{(,)}$，以及 $(,,_{(,),})$；并且 i $\mathcal{P}_{i,h}$ $\vec{\nu}$ h $i<d$ $\overline{f}$ $h\in(,,_{(\mathcal{P}_{i,})})$ $\mathcal{B}$ $\mathcal{Q}_{i,h}$ $\vec{\nu}^{\mathcal{P}_{i,h}}\diamond\vec{\xi}_{i,h}\neq\emptyset$
- 计算相应的逻辑符号集 $S_{n,\,i,\,h}$。

我们按照之前相同的方式递归进行，也就是说，依次为$h=0$、1、. . . 、 $\mathfrak{H}$ 计算所有对应的$\overline{f}$周期的 $i<d$ 的相关数据。对于$h=0$，其中$\mathcal{P}_{i,\,h}=\mathcal{R}_i$，这基本上与步骤 1 中相应的计算相同，整体位操作成本也相同，为 $O(d^2 8^d\log^{1+o(1)} q+d^3 8^d\log q)$。

现在我们假设 $h>0$ 并且所有更小的值都已经处理完毕。我们按照字典序对带有 $\sum_{t=0}^{h} n_{i_{-t}}$ 个条目的逻辑符号元组 $\vec{\nu}^{(\mathcal{P}_{i,h})}$ 进行循环，并注意到共有 $O(2^{(h+1)d})$ 个这样的元组。对于每个元组 $\vec{\nu}^{(\mathcal{P}_{i,h})}$，我们首先检查 $\mathcal{B}(\mathcal{Q}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})}\diamond\vec{\xi}_{i,h})$ 是否非空。由于此处用于 $\mathcal{Q}_{i,h}$ 的生成同余序列的长度属于 $O((h+1)d+H_i)$，引理 5.2.2.1 表明，这项检查需要

$$\begin{aligned}&O(((h+1)d+H_i)2^{(h+1)d+H_i}\log^{1+o(1)} q)\\ \subseteq&O(d^2\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d\,\mathrm{mpe}(q-1)+d}\log^{1+o(1)} q)\end{aligned}$$

位运算。如果 $\mathcal{Q}_{i,h}$ 这一块确实非空，我们需要计算 $\mathsf{Tree}_i(\mathcal{P}_{i,h},\vec{\nu}^{(\mathcal{P}_{i,h})})$ 的紧凑描述 $\mathfrak{D}$。令 $j_1,j_2,\dots,j_K$ 为 $\overline{f}$ 下 i 的 $\overline{f}$ 阶暂态原像（确定这些原像需要 $O(d\log d)$ 次位操作）。将$\vec{\nu}^{(\mathcal{P}_{i,h})}=\diamond_{t=0}^{h}\vec{o_t'}$写作 $\vec{o_t'}\in\{\emptyset,\neg\}^{n_{i_{-t}}}$，并将 $\vec{o_0'}=\diamond_{t=1}^{K}\vec{\nu}^{(\mathcal{P}'_{j_t})}$写作 $\vec{\nu}^{(\mathcal{P}'_{j_t})}\in\{\emptyset,\neg\}^{m_{j_t}+1}$，根据命题 3.3.3 和 3.3.4，我们可以如下选择 $\mathfrak{D}$：

$$\mathfrak{D}:=\{(m,\sum_{t=1}^{K}\sum_{\vec{\nu}^{(\mathcal{P}_{j_t})}\in S_{m,j_t}}\sigma_{\mathcal{P}_{j_t},A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})},\vec{\nu}^{(\mathcal{P}'_{j_t})}\right)$$

$$+\sum_{k=0}^{h-1}\sum_{\vec{\nu}^{(\mathcal{P}_{i',k})}\in S_{m,i',k}}\sigma_{\mathcal{Q}_{i',k},A_{i'}}\left(\vec{\nu}^{(\mathcal{P}_{i',k})}\diamond\vec{\xi}_{i',k},\diamond_{\mathfrak{t}=1}^{k+1}\vec{o_{\mathfrak{t}}'}\diamond\vec{\xi_{i,h}}\right)):m\in\mathcal{N}\}$$
$$\setminus(\mathbb{N}\times\{0\}).$$

计算第一个和，

$$\sum_{t=1}^{K}\sum_{\vec{\nu}^{(\mathcal{P}_{j_t})}\in S_{m,j_t}}\sigma_{\mathcal{P}_{j_t},A_{j_t}}\left(\vec{\nu}^{(\mathcal{P}_{j_t})},\vec{\nu}^{(\mathcal{P}'_{j_t})}\right),$$

对于所有 $m\in\mathcal{N}$ 一起，与步骤 1 中相应的论证类似（也应用引理5.2.2.1，其中 $k:=m_{j_t}$ 和 $m:=q$），并需要进行 $O(d4^d\log^{1+o(1)}q)$ 次位运算。至于计算第二个和的复杂度，

$$\sum_{k=0}^{h-1}\sum_{\vec{\nu}^{(\mathcal{P}_{i',k})}\in S_{m,i',k}}\sigma_{\mathcal{Q}_{i',k},A_{i'}}\left(\vec{\nu}^{(\mathcal{P}_{i',k})}\diamond\vec{\xi}_{i',k},\diamond_{\mathfrak{t}=1}^{k+1}\vec{o_{\mathfrak{t}}'}\diamond\vec{\xi}_{i,h}\right),$$

我们注意到，对于固定的 k 和 $\vec{\nu}^{(\mathcal{P}_{i',k})}$，引理 5.2.2.1 与 $k:=(k+1)d+H_i$ 和 $m:=q$ 一起意味着，计算单个分布数

$$\sigma_{\mathcal{Q}_{i',k},A_{i'}}(\vec{\nu}^{(\mathcal{P}_{i',k})}\diamond\vec{\xi}_{i',k},\diamond_{\mathfrak{t}=1}^{k+1}\vec{o_{\mathfrak{t}}'}\diamond\vec{\xi}_{i,h})$$

取

$$O(2^{(k+1)d+H_i}((k+1)d+H_i)\log^{1+o(1)}q)$$

位运算，因此

$$\begin{aligned}&O(2^{(k+1)d}\cdot 2^{(k+1)d+H_i}((k+1)d+H_i)\log^{1+o(1)}q)\\=&O(2^{2(k+1)d+H_i}((k+1)d+H_i)\log^{1+o(1)}q)\end{aligned}$$

对于固定的 k，同时计算所有 m 的这些数值需要进行位运算。对所有 m 一起完整计算第二个和需要

$$O\left(\sum_{k=0}^{h-1}2^{2(k+1)d+H_i}((k+1)d+H_i)\log^{1+o(1)}q\right)\subseteq O(2^{2hd+H_i}(hd+H_i)\log^{1+o(1)}q),$$

次位运算，其数量上界超过了计算第一个和所需的总体位操作成本，因此也是计算 $\mathfrak{D}$ 的成本。

接下来，我们需要检查 $\mathfrak{D}$ 是否出现在已经引入的描述$\mathfrak{D}_n$中（对于 $n\in\mathcal{N}$）。这些描述的数量 $|\mathcal{N}|$ 至多等于以下形式之一的算术划分中不同（非空）块的数量之和：

- $\mathcal{P}_j$，其中 j 是 $\overline{f}$-暂态；或者
- $\mathcal{P}_{j,\ k}$ 其中 $j<d$ 是 $\overline{f}$-周期的，且 $k\leq h$，

且该和属于

$$O\left(d2^d + d\sum_{k=0}^{h} 2^{(k+1)d}\right) = O(d2^{(h+1)d}).$$

此外， $\mathfrak{D}_n$ 中 $n \in \mathcal{N}$ 以及 $\mathfrak{D}$ 的每个描述都是一个按字典序排序的列表，长度为 $O(\min\{d2^{(h+1)d}, q\})$，其中每个条目都是长度为 $O(\log\ q)$的位字符串，因此检查 $\mathfrak{D} = \mathfrak{D}_n$ 是否对于某个 $n \in \mathcal{N}$成立需要进行 $O(d^2 4^{(h+1)d} \log q)$次位操作。若是，则将 $\vec{\nu}^{(\mathcal{P}_{i,h})}$ 作为新元素添加到$S_{n,i,h}$ 中。否则，我们创建$\mathfrak{D}$ 作为新的树描述 $\mathfrak{D}_{n'}$，其中 $n' = \max\mathcal{N} + 1$，并初始化

$$S_{n',j,h} := \begin{cases} \{\vec{\nu}^{(\mathcal{P}_{i,h})}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

对于给定的 $\overline{f}$ 周期的 $i < d$，此循环的位操作成本为

$$\begin{aligned} &O(2^{(h+1)d} \cdot (2^{2hd+H_i}(hd + H_i)\log^{1+o(1)} q + 4^{(h+1)d} d^2 \log q)) \\ &\subseteq O(2^{3hd+H_i+d}(hd + H_i)\log^{1+o(1)} q + 8^{(h+1)d} d^2 \log q), \end{aligned}$$

并对固定的 h 值的所有此类 i 执行此操作需要

$$O(d(hd + \mathfrak{H})2^{3hd+\mathfrak{H}+d}\log^{1+o(1)} q + d^3 8^{(h+1)d}\log q).$$

位运算。总的来说，步骤 3 的位操作成本为

$$\begin{aligned} &O(d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q \\ &\quad + \sum_{h=1}^{\mathfrak{H}} (d(hd + \mathfrak{H})2^{3hd+\mathfrak{H}+d}\log^{1+o(1)} q + d^3 8^{(h+1)d}\log q)) \\ &\subseteq O(d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q \\ &\quad + d(d+1)\mathfrak{H}2^{3\mathfrak{H}d+\mathfrak{H}+d}\log^{1+o(1)} q + d^3 8^{(\mathfrak{H}+1)d}\log q) \\ &\subseteq O(d^3\,\mathrm{mpe}(q-1)2^{(3d^2+d)\,\mathrm{mpe}(q-1)+d}\log^{1+o(1)} q + d^3 8^{(d\,\mathrm{mpe}(q-1)+1)d}\log q) \\ &\subseteq O(d^3\,\mathrm{mpe}(q-1)2^{(3d^2+d)\,\mathrm{mpe}(q-1)+2d}\log^{1+o(1)} q), \end{aligned}$$

它支配步骤 1 和 2 的成本，因此正如所声称的，它就是问题 2 算法的总体位操作成本。关于所构造的递归树描述列表的长度 $|\mathcal{N}|$ 以及各个有根树描述的内存成本，可以如下推导。应用上述在带有$h = \mathfrak{H} \le d\mathrm{mpe}(q-1)$的循环第 h 次迭代期间始终成立的 $|\mathcal{N}|$ 上界，可得

$$|\mathcal{N}| \in O(d2^{(\mathfrak{H}+1)d}) \subseteq O(d2^{(d\,\mathrm{mpe}(q-1)+1)d}) = O(d2^{d^2\,\mathrm{mpe}(q-1)+d}).$$

此外， $|\mathcal{N}| \le q$ 由于构造的原因，相关递归树描述列表中的每个有根树同构类型都具有树$\Gamma_f(x)$的形式，其中某个$x \in \mathbb{F}_q = \mathrm{V}(\Gamma_f)$。每个树描述 $\mathfrak{D}_n$ 都是由成对的

形式为 (m,k)，其中首项 m 是 $\mathcal{N}$ 的两两不同元素，因此 $\mathfrak{D}_n$ 作为列表的长度至多为 $|\mathcal{N}| \in O(\min\{d2^{d^2}\operatorname{mpe}(q-1)^{+d}, q\})$。最后，存储单个对 (m,k) 的比特成本为 $O(\log q)$，如所要求。

我们以伪代码的形式，对该算法的各个步骤进行详细概述，以此结束本小小节，格式与第 5.2.1 小节末尾的格式相同。

1计算 $\overline{f}$在 $\{0,1,\ldots,d\}$ 上的诱导函数以及 A_i的仿射映射。计算成本：$(d\log^{1+o(1)} q,\ d,\ 0,\ 0,\ 0)$。

2 对于 $i \in \{0,1,\ldots,d-1\}$，令

$$h_i := \begin{cases} \operatorname{ht}(\operatorname{Tree}_{\Gamma_{\overline{f}}}(i)), & \text{if } i \text{ is } \overline{f}\text{-transient}, \\ \infty, & \text{otherwise.} \end{cases}$$

对于 h_i的每个可取值 h，计算其对应的指标列表 Layer_h es i.

QC: $(d^2\log^2 d, 0, 0, 0, 0)$.

3 Set $\mathcal{N} := \emptyset$.

QC: $(1, 0, 0, 0, 0)$.

4 Set $\vec{\mathfrak{D}} := \emptyset$.

QC: $(1, 0, 0, 0, 0)$.

5 Set $\vec{\imath} := \emptyset$.

QC: $(1, 0, 0, 0, 0)$.

6 For $h = 0, 1, \ldots, \max\{h_i : i \in \{0,1,\ldots,d-1\} \setminus \operatorname{per}(\overline{f})\}$, do the following.

QC: $(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q, 0, 0, 0, 0)$.

6.1 If $h = 0$, then do the following.

6.1.1 Set $N' := 0$.
QC: $(1, 0, 0, 0, 0)$.

6.1.2 Add N' to $\mathcal{N}$ as a new element.
QC: $(1, 0, 0, 0, 0)$.

6.1.3 Set $\mathfrak{D}_0 := \emptyset$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(1, 0, 0, 0, 0)$.

6.1.4 For each $i \in \mathrm{Layer}_0$, do the following.
QC: $(d\log d, 0, 0, 0, 0)$.

6.1.4.1 Set $\mathcal{Z}_i := \mathcal{P}_i := \mathfrak{P}(\emptyset)$ and $m_i := 0$.
QC: $(\log d, 0, 0, 0, 0)$.

6.1.4.2 Set $S_{0,i} := \{\emptyset\}$.
QC: $(\log d, 0, 0, 0, 0)$.

6.1.4.3 Add i to $\vec{\imath}$ as a new element.
QC: $(\log d, 0, 0, 0, 0)$.

6.2 Else do the following.

6.2.1 For each $i \in \mathrm{Layer}_h$, do the following.
QC: $(d8^d \log^{1+o(1)} q + d^2 8^d \log q, 0, 0, 0, 0)$.

6.2.1.1 Compute the $\overline{f}$-pre-images $j_1, j_2, \ldots, j_K$ of i.
QC: $(d \log d, 0, 0, 0, 0)$.

6.2.1.2 Compute a spanning congruence sequence, of length m_i, for $\mathcal{Z}_i := \mathcal{P}_i := \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.3 For each $n \in \mathcal{N}$, do the following.
QC: $(d2^d \log q, 0, 0, 0, 0)$.

6.2.1.3.1 Initialize $S_{n,i} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4 For each $\vec{\nu}^{(\mathcal{P}_i)} \in \{\emptyset, \neg\}^{m_i}$, do the following.
QC: $(2^{m_i}(d4^d \log^{1+o(1)} q + d^2 4^d \log q), 0, 0, 0, 0)$.

6.2.1.4.1 Check whether $|\mathcal{B}(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})| = \sigma_{\mathcal{P}_i, \mathbf{0}}(\vec{\nu}^{(\mathcal{P}_i)}, (\emptyset, \ldots, \emptyset)) = 0$, and if so, skip to the next tuple $\vec{\nu}^{(\mathcal{P}_i)}$.
QC: $(d2^d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.4.2 Writing $\vec{\nu}^{(\mathcal{P}_i)} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$ with $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{m_{j_t}+1}$, compute

$$\mathfrak{D} := \{(n, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t}, A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})} \ \vec{\nu}^{(\mathcal{P}'_{j_t})})) : n \in \mathcal{N}\} \backslash (\mathbb{N} \times \{0\}),$$

a compact description of $\mathrm{Tree}_i(\mathcal{P}_i, \vec{\nu}^{(\mathcal{P}_i)})$.
QC: $(d4^d \log^{1+o(1)} q, 0, 0, 0, 0)$.

6.2.1.4.3 Check whether $\mathfrak{D} = \mathfrak{D}_n$ for some (unique) $n \in \mathcal{N}$, and store this information (the truth value and, if applicable, n).
QC: $(d^2 4^d \log q, 0, 0, 0, 0)$.

6.2.1.4.4 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

6.2.1.4.4.1 Add $\vec{\nu}^{(\mathcal{P}_i)}$ to $S_{n,i}$ as a new element.
QC: $(\log q + d, 0, 0, 0, 0)$.

6.2.1.4.5 Else do the following.

6.2.1.4.5.1 Set $N' := N' + 1$, and add it to $\mathcal{N}$ as a new element.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4.5.2 Set $\mathfrak{D}_{N'} := \mathfrak{D}$, and add it to $\vec{\mathfrak{D}}$ as a new element.
QC: $(d2^d \log q, 0, 0, 0, 0)$.

6.2.1.4.5.3 For $j \in \vec{\imath}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

6.2.1.4.5.3.1 Set $S_{N',j} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

6.2.1.4.5.4 Set $S_{N',i} := \{\vec{\nu}^{(\mathcal{P}_i)}\}$.
QC: $(\log q + d, 0, 0, 0, 0)$.

6.2.1.5 Add i to $\vec{\imath}$ as a new element.
QC: $(\log d, 0, 0, 0, 0)$.

7 计算 $\overline{f}$-暂态 $\overline{f}$-原像 $j_1, j_2, \ldots, j_K$ 的 d。计算成本: ($d$log d, 0, 0, 0, 0)。
8 计算树$\Gamma_f(0_{\mathbb{F}_q})$的紧凑描述。计算成本:
($d4^d \log^{1+o(1)} q$, 0,0,0,0)。

$$\mathfrak{D} := \{(n, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{n,j_t}} \sigma_{\mathcal{P}_{j_t},\mathbf{0}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, (\emptyset, \ldots, \emptyset))) : n \in \mathcal{N}\} \setminus (\mathbb{N} \times \{0\}),$$

9检查 $\mathfrak{D} = \mathfrak{D}_n$ 是否对某个（唯一的） $n \in \mathcal{N}$成立，并存储此信息（真值以及在适用时的 n）。计算成本：($d^2 4^d$log q, 0,0,0,0)。

10 如果对于某个 $n \in \mathcal{N}$ 有 $\mathfrak{D} = \mathfrak{D}_n$，则执行以下操作。10.1对所有 $m \in \mathcal{N} \setminus \{n\}$，设置$S_{n,d} := \emptyset$，以及 $S_{m,d} := \neg$。计算成本：($d2^d$对数 q，0，0，0，0)。11否则执行以下操作。11.1设置 $N' := N' + 1$，并将其作为新元素添加到 $\mathcal{N}$ 中。计算成本：(对数 q，0，0，0，0)。 $\vec{}$11.2设置 $\mathfrak{D}_{N'} := \mathfrak{D}$，并将其作为新元素添加到 $\mathfrak{D}$ 中。计算成本：($d2^d$对数 q，0，0，0，0)。11.3 对于 $j \in \vec{\imath} = \{0, 1, \ldots, d-1\} \setminus \mathrm{per}(\overline{f})$，执行以下操作。计算成本：($d$对数 q，0，0，0，0)。 11.3.1设置$S_{N',j} := \emptyset$。计算成本：(对数 q，0，0，0，0)。11.4 对所有 $n \in \mathcal{N} \setminus \{N'\}$，设置 $S_{N',d} := \emptyset$，以及 $S_{n,d} := \neg$。计算成本：($d2^d$对数 q，0，0，0，0)。 12对于 $i \in \mathrm{per}(\overline{f}) \setminus \{d\}$，执行以下操作。计算成本：($d^3$对数 d对数$^{1+o(1)} q$，0，0，0，0)。12.1 计算 H_i，即上述周期的顶点中树的最大树高 Γ_f。

es in

形如 C_{i_t}（其中 $t \in \mathbb{Z}$）的陪集。详情请参见上文讨论。 ls.
计算成本:(d^2 log $d$log$^{1+o(1)} q$,0, 0,0, 0)。13计算 $\mathfrak{H} := \max\{H_i : i \in \mathrm{per}(\overline{f}) \setminus \{d\}\}$。计算成本: ($d$ log d, 0, 0, 0, 0)。14对于 $h = -1$,0, 1, .. . , $\mathfrak{H}$, 执行以下操作。计算成本:(d^3 mpe$(q-1)$log$^{1+o(1)} q$, 0,0,0,0)。14.1 如果 $h = -1$, 则执行以下操作。
14.1.1 对于 $i \in \mathrm{per}(\overline{f}) \setminus \{d\}$, 执行以下操作。计算成本:($d^2$ mpe$(q-1)$ log$^{1+o(1)} q$, 0, 0, 0, 0)。

14.1.1.1 Compute and store the $\overline{f}$-pre-images of i, including the periodic one, i'.
QC: $(d \log d, 0, 0, 0, 0)$.

14.1.1.2 Compute $\mathcal{X}_{i,-1} := (\theta_{i,k}(x))_{k=1,2,\ldots,H_i}$.
QC: $(d \operatorname{mpe}(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2 Else do the following.

14.2.1 If $h = 0$, then do the following.

14.2.1.1 For $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d^2 \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.1.1.1 Compute $\mathcal{X}_{i,0} = \mathcal{R}_i = \bigwedge_{t=1}^{K} \mathfrak{P}'(\mathcal{P}_{j_t}, A_{j_t})$, with a spanning sequence of length n_i, where $j_1, j_2, \ldots, j_K$ are the $\overline{f}$-transient $\overline{f}$-pre-images of i (this can be handled analogous to Steps 6.2.1.1 and 6.2.1.2).
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.2 Else do the following.

14.2.2.1 For $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d^2 \log^{1+o(1)} q, 0, 0, 0, 0)$.

14.2.2.1.1 If $h \le H_i$, then compute $\mathcal{X}_{i,h} = \lambda(\mathcal{X}_{i',h-1}, A_{i'})$ (we observe that as a spanning congruence sequence, $\mathcal{X}_{i,h}$ has length $n_{i'}$, same as $\mathcal{X}_{i',h-1}$). Otherwise, skip to the next value of i.
QC: $(d \log^{1+o(1)} q, 0, 0, 0, 0)$.

15 For $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d^3 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

15.1 Set $\mathcal{Z}_i := (\mathcal{X}_{i,h})_{h=-1,0,\ldots,H_i}$.
QC: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

16 For $h = 0, 1, \ldots, \mathfrak{H}$, do the following.
QC: $(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q, 0, 0, 0, 0)$.

16.1 Initialize all sets $S_{n,i,h}$, where $n \in \mathcal{N}$ and $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$ such that $H_i \ge h$, to be $\emptyset$.
QC: $(d^2 2^{hd}, 0, 0, 0, 0)$.

16.2 If $h = 0$, then do the following.

16.2.1 Extend $\mathcal{N}$ and the associated list $\vec{\mathfrak{D}}$ of rooted tree descriptions $\mathfrak{D}_n$ such that all rooted tree isomorphism types of the form $\operatorname{Tree}_i(\mathcal{P}_{i,0}, \vec{\nu}^{(\mathcal{P}_{i,0})}) = \operatorname{Tree}_i(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{P}_{i,0})})$ for $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$ are covered, and compute the associated logical sign tuple sets $S_{n,i,0}$. This is analogous to Step 6.2.1.4, but carried out for the $O(d)$ values of $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$ together.
QC: $(d^2 8^d \log^{1+o(1)} q + d^3 8^d \log q, 0, 0, 0, 0)$.

16.3 Else do the following.

16.3.1 For each $i \in \operatorname{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d(hd + \mathfrak{H}) 2^{3hd+\mathfrak{H}+d} \log^{1+o(1)} q + d^3 8^{(h+1)d} \log q, 0, 0, 0, 0)$.

16.3.1.1 Check whether $h \le H_i$. If not, skip to the next i.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2 回想一下 $\underset{\{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-h}}}}{\mathcal{P}_i}$ $_t$ $\in$, $_h = \bigwedge_{t=0}^{h} \mathcal{X}_{i,}$,
对每个 $\vec{\nu}^{(\mathcal{P}_{i,h})}$执行以下操作。计算成本：((hd + H_i)2^{3hd+H_i+d}log^{1+o(1)} q + d^2 8^{(h+1)d}log q, 0,0,0,0)。

16.3.1.2.1 回想一下 $\mathcal{Q}_{i,\ h} = \mathcal{P}_{i,\ h} \wedge \mathcal{U}_i$, 其中 $\mathcal{U}_i = \mathfrak{P}'(\theta_{i,\ k}(x) : \ k$ =1, 2, . . . , H_i)（以及第 47 页上 $\vec{\xi}_{i,\ h} \in \{\emptyset, \neg\}^{H_i}$的定义），检查是否

$$|\mathcal{B}(\mathcal{Q}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h})| = \sigma_{\mathcal{Q}_{i,h},\mathbf{0}}(\vec{\nu}^{(\mathcal{P}_{i,h})} \diamond \vec{\xi}_{i,h}, (\emptyset, \ldots, \emptyset)) = 0,$$

如果是，则跳到下一个元组 $\vec{\nu}^{(\mathcal{P}_{i,h})}$。
计算成本: $(d^2\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d\mathrm{mpe}(q-1)^{+d}}\log^{1+o(1)} q$, 0, 0, 0, 0).

16.3.1.2.2将 $\vec{\nu}^{(\mathcal{P}_{i,h})} = \diamond_{t=0}^{h} \vec{o_t'}$写作 $\vec{o_t'} \in \{\emptyset, \neg\}^{n_{i-t}}$, 并将 $\vec{o_0'} = \diamond_{t=1}^{K} \vec{\nu}^{(\mathcal{P}'_{j_t})}$写作 $\vec{\nu}^{(\mathcal{P}'_{j_t})} \in \{\emptyset, \neg\}^{n_{j_t}+1}$ （其中 j_1、 j_2、 . . . 、 j_K 是 i 的 $\overline{f}$- 暂态 $\overline{f}$- 原像, 在步骤 14.1.1.1 中计算得到），计算

$$\begin{aligned}
&\mathfrak{D} := \\
&\{(m, \sum_{t=1}^{K} \sum_{\vec{\nu}^{(\mathcal{P}_{j_t})} \in S_{m,j_t}} \sigma_{\mathcal{P}_{j_t},A_{j_t}}(\vec{\nu}^{(\mathcal{P}_{j_t})}, \vec{\nu}^{(\mathcal{P}'_{j_t})}) \\
&+ \sum_{k=0}^{h-1} \sum_{\vec{\nu}^{(\mathcal{P}_{i',k})} \in S_{m,i',k}} \sigma_{\mathcal{Q}_{i',k},A_{i'}}(\vec{\nu}^{(\mathcal{P}_{i',k})} \diamond \vec{\xi}_{i',k}, \diamond_{\mathrm{t}=1}^{k+1} \vec{o_{\mathrm{t}}'} \diamond \vec{\xi_{i,h}})) : \\
&m \in \mathcal{N}\} \setminus (\mathbb{N} \times \{0\}).
\end{aligned}$$

QC: $(2^{2hd+H_i}(hd + H_i)\log^{1+o(1)} q, 0, 0, 0, 0)$.

16.3.1.2.3检查是否对于某个（唯一的） $n \in \mathcal{N}$有 $\mathfrak{D} = \mathfrak{D}_n$, 并存储此信息（真值以及在适用时的 n）。计算成本：($d^2 4^{(h+1)d}$log q, 0,0,0,0)。

16.3.1.2.4 如果对于某个 $n \in \mathcal{N}$, $\mathfrak{D} = \mathfrak{D}_n$, 则执行以下操作。
16.3.1.2.4.1 将 $\vec{\nu}^{(\mathcal{P}_{i,h})}$ 作为新元素添加到 $S_{n\ i\ h}$。
QC: $(\log q + (h + 1)d, 0, 0, 0, 0)$.
16.3.1.2.5 否则，执行以下操作。
16.3.1.2.5.1 设置 $N' := N' + 1$, 并将其作为新元素添加到 $\mathcal{N}$ 中。计算成本：(log q, 0, 0, 0, 0)。¯16.3.1.2.5.2 设置 $\mathfrak{D}_{N'} := \mathfrak{D}$, 并将其作为新元素添加到 $\mathfrak{D}$ 中。计算成本：($d2^{(h+1)d}$log q, 0, 0, 0, 0)。16.3.1.2.5.3 对于 $j \in \vec{\imath} = \{0, 1, \ldots, d-1\} \setminus \mathrm{per}(\overline{f})$, 执行以下操作。计算成本：($d$log q, 0, 0, 0, 0)。16.3.1.2.5.3.1 设置 $S_{N',j} := \emptyset$。计算成本：(log q, 0, 0, 0, 0)。16.3.1.2.5.4 设置 $S_{N',d} := \neg$。计算成本：(log q, 0, 0, 0, 0)。16.3.1.2.5.5 对于 $k = 0$, 1, . . . , $h - 1$, 执行以下操作。计算成本：(dhlog q, 0, 0, 0, 0)。

16.3.1.2.5.5.1 For $j \in \mathrm{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

16.3.1.2.5.5.1.1 Set $S_{N',j,k} := \emptyset$.
QC: $(\log q, 0, 0, 0, 0)$.

16.3.1.2.5.6 For $j \in \mathrm{per}(\overline{f}) \setminus \{d\}$, do the following.
QC: $(d \log q + (h+1)d^2, 0, 0, 0, 0)$.

16.3.1.2.5.6.1 Set

$$S_{N',j,h} := \begin{cases} \{\vec{\nu}^{(\mathcal{P}_{i,h})}\}, & \text{if } j = i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

QC: $(\log q + (h+1)d, 0, 0, 0, 0)$.

17 对于 $n \in \mathcal{N}$，执行以下操作。
计算成本: $(d^4 \,\mathrm{mpe}(q-1)4^{d^2 \,\mathrm{mpe}(q-1)^{+d}}, 0, 0, 0, 0)$; 请注意，尽管第16步的计算成本项只涉及 d^3，而不涉及 d^4，但它仍然支配这一项。
对于 $i \in \mathrm{per}(\overline{f}) \setminus \{d\}$，执行以下操作。计算成本：$(d^3 \,\mathrm{mpe}(q-1)2^{d^2 \,\mathrm{mpe}(q-1)^{+d}}, 0, 0, 0, 0)$。17.1.1 设置 $S_{n,i} := (S_{n,i,h})_{h=0,1,\ldots,H_i}$。计算成本：$(d^2 \,\mathrm{mpe}(q-1)2^{d^2 \,\mathrm{mpe}(q-1)^{+d}}, 0, 0, 0, 0)$，用于复制，利用

$$\begin{aligned} &\sum_{h=0}^{H_i} 2^{(h+1)d}(h+1)d \in O((H_i+1)d2^{(H_i+1)d}) \\ \subseteq &O(d^2 \,\mathrm{mpe}(q-1)2^{d^2 \,\mathrm{mpe}(q-1)+d}) \end{aligned}$$

18 输出分区树寄存器 $((\mathcal{Z}_i)_{i=0,\,1,\,\ldots,\,d-1}, ((\mathfrak{D}_n, (S_{n,\,i})_{i=0,\,1,\,\ldots,\,d})_{n \in \mathcal{N}})$，然后停止。

计算成本：$(d^2 4^{d^2} \,\mathrm{mpe}(q-1)^{+d} \log q + d^4 \,\mathrm{mpe}(q-1)4^{d^2} \,\mathrm{mpe}(q-1)^{+d}, 0, 0, 0, 0)$ 用于复制。注意，对于固定的 i，$\mathcal{X}_i$、$_{-1} = (\theta_i$、$_h(x))_{h=1}$、2、...、$_{H_i}$ 的位存储成本为 $O(H_i \log q) \subseteq O(d\,\mathrm{mpe}(q-1) \log q)$；对于固定的 i 和 h，$\mathcal{X}_i$、$_h$ 的位存储成本为 $O((h+1)d \log q)$；递归树描述列表 $(\mathfrak{D}_n)_{n \in \mathcal{N}}$ 的总位存储成本为 $O(|\mathcal{N}|^2 \log q) \subseteq O(d^2 4^{d^2} \,\mathrm{mpe}(q-1)^{+d} \log q)$；对于 $n \in \mathcal{N}$ 和 $i \in \{0, 1, \ldots, d\}$，$S_n$、$_i$ 的总位存储成本为 $O(d^4 \,\mathrm{mpe}(q-1)4^{d^2} \,\mathrm{mpe}(q-1)^{+d})$（步骤 17 的位操作成本；对于每个 $\overline{f}$ 的 S_n、$_i$ 的组合存储成本，$i \notin$ 或 $i = d$，由每个 $\overline{f}$ 的 $i \in$ 所对应的成本主导 $\setminus \{d\}$），并且

$$\begin{aligned} &O(dH_i \log q + d\sum_{h=0}^{H_i}(h+1)d \log q + d^2 4^{d^2 \,\mathrm{mpe}(q-1)+d} \log q \\ &\quad + d^4 \,\mathrm{mpe}(q-1)4^{d^2 \,\mathrm{mpe}(q-1)+d}) \\ &\quad \subseteq O(d^2 4^{d^2 \,\mathrm{mpe}(q-1)+d} \log q + d^4 \,\mathrm{mpe}(q-1)4^{d^2 \,\mathrm{mpe}(q-1)+d}). \end{aligned}$$

5.2.3 陈述（3）的证明

我们遵循第3.4小节的方法。如果 $r = 0_{\mathbb{F}_q}$，那么我们搜索唯一的 $n \in \mathcal{N} = \{0, 1, \ldots, N\}$ 使得 $S_{n,d} = \emptyset$（而不是 $\neg$）。此过程需要

$O(N) \subseteq O(d2^{d^2}\,\mathrm{mpe}(q-1)^{+d})$ 位运算——关于这一界限，请参阅陈述(2)。找到 n 后，可以直接读出 $\mathfrak{D}_n$ 关于 $\mathfrak{I}_n$ =树$\Gamma_f(0_{\mathbb{F}_q})$ 的描述，并输出相关循环序列的描述；复制需要 $O(d2^{d^2}\mathrm{mpe}(q-1)^{+d}\log\ q)$ 位运算。

此后，我们假设 $r \neq 0_{\mathbb{F}_q}$。回顾引理 5.1.6 的陈述（4）的证明，对于 $\mathbb{F}_q$ 的本原元，自然有一种选择 ω。我们通过一次 fdl 查询计算 $\mathfrak{l} := \log_\omega(r)$（如果 (r,l) 是由问题 1 的算法参数化的 CRL 列表中的一个元素，那么 r 已明确指定为 ω 的幂，因此可以跳过此计算）。然后，我们令 $i := \mathfrak{l} \bmod d$（根据引理 5.1.5(2)，计算需要 $O(\log^{1+o(1)} q)$ 次位运算），从而 $r \in C_i$，并确定 $i = i_0$ 在 $\overline{f}$ 作用下的循环 $(i_0, i_1, \ldots, i_{\ell-1})$，这需要 $O(d\log\ d)$ 次位运算（另见陈述（1）论证的开头）。此外，我们令

$$r'_{i_0} = r'_i := \iota_i^{-1}(r) = \frac{\mathfrak{l}-i}{d} \in \mathbb{Z}/s\mathbb{Z} \ \text{ and } \ r'_{i_t} := A_{i_{t-1}}(r'_{i_{t-1}}) \text{ for } t = 1, 2, \ldots, \ell-1,$$

总共需要 $O(d\log^{1+o(1)} q)$ 次位运算。最后，对于 $t = 0$、1、.....、$\ell-1$，我们计算

$$\mathcal{A}_{i_t} := A_{i_t} A_{i_{(t+1) \bmod \ell}} \cdots A_{i_{(t+\ell-1) \bmod \ell}},$$

对于所有 t，总共需要 $O(d^2\log^{1+o(1)} q)$ 次位运算（另见陈述（1）的证明开头部分）。

在另行通知之前，我们假设 t 固定。采用第 3.4 小节的记号，我们有 $\mathcal{A}_{i_t}(x) = \overline{\alpha}_{i_t} x + \overline{\beta}_{i_t}$。我们计算 $\gcd(\overline{\alpha}_{i_t}, s)$，找出满足 $\gcd(\overline{\alpha}_{i_t}^{L_{i_t}}, s) = \gcd(\overline{\alpha}_{i_t}^{L_{i_t}+1}, s) =$ 的最小非负整数 L_{i_t}： s''_{i_t}；计算 $s'_{i_t} := s/s''_{i_t}$ 和 $\mathfrak{f}_{i_t} := \sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z \overline{\beta}_{i_t} \bmod s''_{i_t}$（$\mathcal{A}_{i_t}$ 模 s''_{i_t} 的约化的唯一周期点）。若使用二分查找来确定 L_{i_t}，这些任务总共可在 $O(\log^{2+o(1)} q)$ 次位运算内完成（另见陈述（1）的证明中引入 L_i 的段落）。

正如我们在第 3.4 小节中已经观察到的，由于 r'_{i_t} 在 $\mathcal{A}_{i_t}$ 作用下的循环上的每个点（对应于 $f^t(r)$ 在 f^ℓ 作用下的循环）都与 $\mathfrak{f}_{i_t}$ 模 s''_{i_t} 同余，因此 $\mathcal{P}_{i_t} = \mathfrak{P}(x \equiv \mathfrak{b}_{i_t,\ j}$ (模 $\mathfrak{a}_{i_t,\ j}$)： $j = 1, 2, \ldots, m_{i_t})$ 的某些生成同余式的真假值可能在该循环上保持不变。更具体地说，

- 如果 $\mathfrak{f}_{i_t} \not\equiv \mathfrak{b}_{i_t,j}$ $\left(\text{模 } \gcd(\mathfrak{a}_{i_t,j}, s''_{i_t})\right)$，那么 $\mathcal{P}_{i_t}$ 的第 j 个生成同余式沿循环恒为假。对于这样的 j，我们令 $\nu_j := \neg$，并称 j 为 I 型。

- 若$\mathfrak{a}_{i_t,j} \mid s''_{i_t}$ 且 $\mathfrak{f}_{i_t} \equiv \mathfrak{b}_{i_t,j}$ (模 $\mathfrak{a}_{i_t,j}$)，则 j阶生成同余式在 $\mathcal{P}_{i_t}$的循环上始终成立。对于这样的 j，我们令 $\nu_j := \emptyset$，并称 j 为 II型。

遍历$\mathcal{P}_{i_t}$ 的生成同余式，并分别创建属于I 型和II型的完整列表，需要 $O(m_{i_t}\log^{1+o(1)} q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$位运算。对于$m_{i_t} \in O(d^2\,\mathrm{mpe}(q-1))$ 的界，我们从第 3.3 小节回忆起，$\mathcal{P}_{i_t} = \mathcal{Q}_{i_t,\ H_{i_t}} = \bigwedge_{k=0}^{H_{i_t}} \lambda_{i_{t-k}}^k(\mathcal{R}_{i_{t-k}}) \wedge \mathcal{U}_{i_t}$，并且该

$\lambda^{k}_{i_{t-k}}(\mathcal{R}_{i_{t-k}})$ 的“标准”生成同余序列的长度（存储在分区树寄存器中，为$\mathcal{X}_{i_t,\ k}$）为 $n_{i_{t-k}} \in O(d)$，而 $\mathcal{U}_{i_t} = \mathfrak{P}(\theta_{i_t,\ h}(x)\colon h = 0, 1, \ldots, H_{i_t})$的生成同余序列存储为 $\mathcal{X}_{i_t,\ -1}$，其长度为 H_{i_t}。此外，$H_i \in O(d\mathrm{mpe}(q-1))$。

接下来，对于那些既非I 型也非II型的$j \in \{1, 2, \ldots, m_{i_t}\}$，我们根据第2.4小节中的定义，计算仿射离散对数 $\mathfrak{l}_{i_t,j} := \log^{(\mathfrak{a}_{i_t,j})}{}_{\mathcal{A}_{i_t}}(r'_{i_t}, \mathfrak{b}_{i_t,j})$；根据第2.4小节中的约化论证，这至多需要 $O(m_{i_t}) \subseteq O(d^2\mathrm{mpe}(q-1))$ MDL查询，以及$O(m_{i_t}\log^{1+o(1)} q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$ 位运算。如果 $j \in \{1, 2, \ldots, m_{i_t}\}$ 不是 I 型或 II型且 $\mathfrak{l}_{i_t,j} = \infty$，那么沿着 r'_{i_t} 在$\mathcal{A}_{i_t}$下的循环，$\mathcal{P}_{i_t}$ 的第 j 个生成同余式始终为假；对于这样的 j，我们设 $\nu_j := \neg$，并称 j 为 III型。

按照第 3.4 小节的记号，集合 I_{i_t} 恰好由属于类型 I–III 之一的那些 $j \in \{1, 2, \ldots, m_{i_t}\}$组成。现在，对于每个满足 $j \notin I_{i_t}$的 $j \in \{1, 2, \ldots, m_{i_t}\}$，我们通过$\mathcal{A}_{i_t}$模 $\mathfrak{a}_{i_t,\ j}$的约化论证计算 r'_{i_t}的周期长度 $l_{i_t,\ j}$——根据第2.4小节的归约论证，这需要 $O(m_{i_t}) \subseteq O(d^2\,\mathrm{mpe}(q-1))$ 次 MORD查询，以及 $O(m_{i_t}\log^{1+o(1)} q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q)$ 次位运算。

我们回顾一下，r 的 f-循环周期长度 l 作为输入的一部分提供给我们。对于每个 $j \in \{1, 2, \ldots, m_{i_t}\} \setminus I_{i_t}$，我们都有 (l/ℓ)-同余式 $y \equiv \mathfrak{l}_{i_t,\ j} \pmod{l_{i_t,\ j}}$；当代入 $x := \mathcal{A}^{y}_{i_t}(r'_{i_t})$ 时，该同余式恰好对满足以下条件的那些 $y \in \mathbb{Z}/(l/\ell)\mathbb{Z}$ 成立：$\mathcal{P}_{i_t}$ 的第 j 个生成同余式变为真。与第 3.4 小节一样，我们设

$$\mathcal{P}^{(i_t)} := \mathfrak{P}(y \equiv \mathfrak{l}_{i_t,j} \pmod{l_{i_t,j}} : j \in \{1, 2, \ldots, m_{i_t}\} \setminus I_{i_t}),$$

$\mathbb{Z}/(l/\ell)\mathbb{Z}$的算术划分。用于参数化 $\mathcal{P}^{(i_t)}$ 的块的逻辑符号元组来自集合 $\{\emptyset, \neg\}^{\{1,2,\ldots,m_{i_t}\}\setminus I_{i_t}}$，其形式为 $\vec{\nu} = (\nu_j)^{j\in\{1,\ldots,m_{i_t}\}\setminus I_{i_t}}$。对于这样的元组 $\vec{\nu}$，我们设 $\vec{\nu}^{+} := (\nu_j)_{j=1,2,\ldots,m_{i_t}}$（由 $j \in I_{i_t}$ 索引的位置填入此前定义的、针对这类 j 的常值逻辑符号）。反之，对于 $\vec{\nu'} \in \{\emptyset, \neg\}^{m_{i_t}}$，我们将 $\vec{\nu'}$到 $\{\emptyset, \neg\}^{\{1,2,\ldots,m_{i_t}\}\setminus I_{i_t}}$的投影记为 $\vec{\nu'}^{-}$。

由此可知，如果 $y \in \mathbb{Z}/(l/\ell)\mathbb{Z}$ 包含于 $\mathcal{B}(\mathcal{P}^{(i_t)}, \vec{\nu})$的块中，则$\mathcal{A}^{y}_{i_t}(r'_{i_t})$ 总是包含于 $\mathcal{B}(\mathcal{P}_{i_t}, \vec{\nu}^{+})$ 的块中，因此 树$\Gamma_f(f^{t+\ell y}(r)) = \mathfrak{T}_{n'_{i_t}(\vec{\nu})}$，其中 $n'_{i_t}(\vec{\nu})$ 是满足 $\vec{\nu}^{+} \in S_{i_t,n}$ 的唯一 $n \in \{0, 1, \ldots, N\}$。

我们提醒读者，我们希望输出一个紧凑描述，即刻画包含 r 的 Γ_f 的连通分支的有根树同构类型的循环序列。为此，我们计算对于 $n \in \{0, 1, \ldots, N\}$（以及我们仍固定的 t）定义如下的集合$\overline{S}_{n,\ i_t}$。$\overline{S}_{n,\ i_t}$ 的元素恰好是满足 $n'_{i_t}(\vec{\nu}) = n$ 的那些逻辑符号元组 $\vec{\nu} \in \{\emptyset, \neg\}^{I_{i_t}}$。

让我们描述如何计算这些集合$\overline{S}_{n,i_t}$。我们遍历 $N+1 \in O(d2^{d^2}\mathrm{mpe}(_{q-1})^{+d})$ 个 n 的值（另请参见本小小节开头关于$r = 0_{\mathbb{F}_q}$ 的论证），对于每个 n，我们按照列出的阶（即字典序）遍历 $|S_{n,i_t,H_{i_t}}|$ 逻辑符号元组ν'' 中的 $S_{n,i_t,H_{i_t}}$（S_{n,i_t}的最后一项）。对于每个 $\vec{\nu''}$，令 $\vec{\nu'} := \vec{\nu''} \diamond \vec{\xi}_{i_t,H_{i_t}} = (\nu'_j)_{j=1,2,\ldots,m_{i_t}}$。我们检查对于所有 $j \in I_{i_t}$是否有 $\nu'_j = \nu_j$（这需要 $O(m_{i_t}\log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log q)$次位运算）。如果 $\vec{\ }\ \vec{\ }\ \vec{\ }^{-}$不成立，我们就转到下一个 ν'' 及其关联的 ν'；否则，我们将 ν' 加入$\overline{S}_{n,i_t}$，作为

一个新元素（再次进行 $O(m_{i_t}\log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)\log\ q)$ 位运算以完成复制）。由于 $N \in O(d2^{d^2\,\mathrm{mpe}(q-1)+d})$ 和 $\sum_{n=0}^{N}|S_{n,i_t,H_{i_t}}| \le 2^{(H_{i_t}+1)d} \le 2^{d^2\mathrm{mpe}(q-1)+d}$，并且我们只需进行 $O(1)$ 次位运算来处理一个 n ，对于该对象有$S_{n,i_t}=\emptyset$，因此我们得出，计算集合 $\overline{S}_{n,i_t}$（对于我们固定的 t）的总体位操作成本为

$$O(d2^{d^2\,\mathrm{mpe}(q-1)+d} + m_{i_t}2^{(H_{i_t}+1)d}\log q) \subseteq O(d^2\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q),$$

且表示这些集合的构造数组按字典序排列。

现在，我们终于再次“取消固定” t。对于 $t \in \{0,1,\ldots,\ell-1\}$ 的所有 $O(d)$ 值合在一起，在同意固定 t 之后所描述的所有计算的 q-有界查询复杂度是

$$\begin{aligned}&(d\log^{2+o(1)} q + d^3\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d^3\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d}\log q,\\ &\ 0, d^3\,\mathrm{mpe}(q-1), d^3\,\mathrm{mpe}(q-1), 0).\end{aligned} \tag{25}$$

此处，位操作分量中的第一项加上指定的查询，涵盖了除计算集合$\overline{S}_{n,i_t}$ 本身之外的所有开销，而后者则由位操作分量中的第二项涵盖（无需任何查询）。考虑到所有其他相关计算的查询复杂度（见下文算法的详细步骤），我们发现，为了获得用于求解问题 3的整个算法的有效 q-有界查询复杂度，只需将公式 (25) 中的第二个条目 0 替换为 d 即可。

与前两个小小节一样，我们最后给出该算法的一些伪代码，其中列出了每一步的 q-有界查询复杂度。

1 如果 $r = 0_{\mathbb{F}_q}$，则搜索唯一的 $n \in \mathcal{N} = \{0,1,\ldots,N\}$，使得 $S_{n,d} = \emptyset$，并输出以下内容后停止：“编码所讨论连通分支的有向图同构类型的有根树同构类型循环序列为 $[\mathfrak{D}_n]$。”计算成本：$(d2^{d^2}\,\mathrm{mpe}(q-1)^{+d}\log\ q, 0, 0, 0, 0)$。 2 计算 $\overline{f}$ 在 $\{0,1,\ldots,d\}$ 上的诱导函数以及 A_i 的仿射映射。计算成本：$(d\log^{1+o(1)} q, d, 0, 0, 0)$。 3 令 ω 为 $\mathbb{F}_q$ 的“自然”本原元选择（见引理 5.1.6(4) 的证明），并确定 $\ell := \log_\omega(r)$。计算成本：$(\log\ q, 1, 0, 0, 0)$。 4 设 $i := \ell \bmod d$。计算成本：$(\log^{1+o(1)} q, 0, 0, 0, 0)$。 5 确定 $i = i_0$ 在 $\overline{f}$ 下的循环 $(i_0, i_1, \ldots, i_{\ell-1})$。计算成本：$(d\log\ d, 0, 0, 0, 0)$。 6 对于 $t = 1, 2, \ldots, \ell-1$，计算 $r'_{i_0} := \frac{\ell-i}{d}$ 和 $r'_{i_t} := A_{i_{t-1}}(r'_{i_{t-1}})$。计算成本：$(d\log^{1+o(1)} q, 0, 0, 0, 0)$。

对于 $t = 0, 1, \ldots, \ell - 1$, 计算 $\mathcal{A}_{i_t} := A_{i_t} A_{i_{(t+1) \bmod \ell}} \cdots A_{i_{(t+\ell-1) \bmod \ell}}$: $x \mapsto \overline{\alpha}_{i_t} x + \overline{\beta}_{i_t}$.
计算成本 : $(d^2 \log^{1+o(1)} q, 0,0,0,0)$.

8 对于每个 $t = 0$、1、...、$\ell - 1$, 执行以下操作。计算成本:

$$(d \log^{2+o(1)} q + d^3 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^3 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} \log q,$$
$$0, d^3 \operatorname{mpe}(q-1), d^3 \operatorname{mpe}(q-1), 0).$$

8.1 计算 $\gcd(\overline{\alpha}_{i_t}, s)$。计算成本:
$(\log^{1+o(1)} q, 0,0,0, 0)$。
8.2 找出满足以下条件的最小非负整数 L_{i_t}

$$\gcd(\overline{\alpha}_{i_t}^{L_{i_t}}, s) = \gcd(\overline{\alpha}_{i_t}^{L_{i_t}+1}, s) =: s''_{i_t}.$$

计算成本: $(\log^{2+o(1)} q, 0,0,0, 0)$。

8.3 计算 $s'_{i_t} := s / s''_{i_t}$。
计算成本: $(\log^{1+o(1)} q, 0,0,0, 0)$。
8.4 计算 $\mathfrak{f}_{i_t} := \sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z \overline{\beta}_{i_t} \bmod s''_{i_t}$。为此，不要将所有加数相加，而应使用等比求和公式

$$\sum_{z=0}^{L_{i_t}-1} \overline{\alpha}_{i_t}^z = \begin{cases} \frac{\overline{\alpha}_{i_t}^{L_{i_t}} - 1}{\overline{\alpha}_{i_t} - 1}, & \text{if } \overline{\alpha}_{i_t} \neq 1, \\ L_{i_t} \overline{\alpha}_{i_t}, & \text{if } \overline{\alpha}_{i_t} = 1, \end{cases}$$

以及 $L_{i_t} \in O(\log q)$ 这一事实。
计算成本: $(\log^{2+o(1)} q, 0,0,0, 0)$。

8.5 从给定的分区树寄存器中读出 $\mathcal{P}_{i_t} = \mathfrak{P}(x \equiv \mathfrak{b}_{i_t,j} (\bmod\ \mathfrak{a}_{i_t,j})$: $j = 1,2, \ldots, m_{i_t})$。

计算成本: $(d^2 \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$。

8.6 对于 $j = 1$、2、.....、m_{i_t}, 执行以下操作。
计算成本: $(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q, 0, d^2 \operatorname{mpe}(q-1)\ d^2 \operatorname{mpe}(q-1), 0)$。

8.6.1 Check whether $\mathfrak{f}_{i_t} \not\equiv \mathfrak{b}_{i_t,j} \pmod{\gcd(\mathfrak{a}_{i_t,j}, s''_{i_t})}$. If so, set $\nu_j := \neg$, and $\mathrm{Type_I}(j) := \text{true}$, $\mathrm{Type_{II}}(j) := \text{false}$ and $\mathrm{Type_{III}}(j) := \text{false}$. If not, just set $\mathrm{Type_I}(j) := \text{false}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

8.6.2 如果 $\mathrm{Type_I}(j)$ =false，则检查是否

$$\mathfrak{a}_{i_t,j} \mid s''_{i_t} \text{ and } \mathfrak{f}_{i_t} \equiv \mathfrak{b}_{i_t,j} \pmod{\mathfrak{a}_{i_t,j}}.$$

若是，则设置 $\nu_j := \emptyset$，并将 $\mathrm{Type_{II}}(j) := \text{true}$ 和 $\mathrm{Type_{III}}(j) := \text{true}$。
若否，则只将 $\mathrm{Type_{II}}(j)$:=false。计算成本: $(\log^{1+o(1)} q, 0,0,0,0)$。

8.6.3 如果 $\mathrm{Type_I}\ j = \mathrm{Type_{II}}(j)$ = 为 false，则计算 $\mathfrak{l}_{i_t,\ j} := \log^{(\mathfrak{a}_{i_t,j})} \mathcal{A}_{i_t}(r'_{i_t}, \mathfrak{b}_{i_t,\ j})$，并检查 $\mathfrak{l}_{i_t,\ j} = \infty$ 是否成立。如果成立，则设置 $\nu_j := \neg$，并令 $\mathrm{Type_{III}}(j) :=$ true。否则，只需设置 $\mathrm{Type_{III}}(j)$:=false。计算成本: （$\log^{1+o(1)} q$, 0, 1, 0, 0）。

8.6.4 如果 Type$_{\text{I}}(j)$ =Type$_{\text{II}}(j)$ = Type$_{\text{III}}(j)$ =为 false，则计算 $l_{i_t,\ j}$ 在 $\mathcal{A}_{i_t}$ 模$\alpha_{i_t,\ j}$下的周期长度。计算成本：（$\log^{1+o(1)} q$, 0,0, 1, 0）。

8.7 设置计算成本：(d^2 mpe$(q-1)$ log q, 0, 0, 0, 0). 8.8 对于每个 $n \in \mathcal{N} = \{0, 1, \ldots, N\}$，执行以下操作。计算成本：($d^2$ mpe$(q-1)2^{d^2}$ mpe$(q-1)^{+d}$log q, 0, 0, 0, 0). 8.8.1 初始化集合 S_{n,i_t} 为 $\emptyset$。计算成本：(log q, 0, 0, 0, 0). 8.8.2 对于每个 $\nu'' \in S_{n,i_t,H_{i_t}}$（来自输入中的划分树寄存器的 S_{n,i_t} 的最后一个条目），执行以下操作。计算成本：(d^2 mpe$(q-1)2^{d^2}$ mpe$(q-1)^{+d}$log q, 0, 0, 0, 0). 8.8.2.1 设置 $\vec{\nu'} := \vec{\nu''} \diamond \vec{\xi}_{i_t,H_{i_t}} = (\nu'_j)_{j=1,2,\ldots,m_{i_t}}$。计算成本：($d$ mpe$(q-1)$ log q, 0, 0, 0, 0). 8.8.2.2 检查对于所有满足以下条件之一的 $j \in \{1, 2, \ldots, m_{i_t}\}$，$\nu'_j = \nu_j$ 是否成立：Type$_{\text{I}}(j)$、Type$_{\text{II}}(j)$ 或 Type$_{\text{III}}(j)$。如果成立，则将 $\vec{\nu'}^{-}$（即删除了对应于这三种类型之一的所有分量后的 $\vec{\nu'}$）作为新元素添加到 $\overline{S}_{n,i_t}$ 中。计算成本：(d^2 mpe$(q-1)$ log q, 0, 0, 0, 0).

$\mathcal{P}^{(i_t)} := \mathfrak{J}(y \equiv l_{i,\ j} \pmod{l_{i,\ j}} : 1 \le j \le m_{i_t}$, Type$_{\text{I}}(j)$ = Type$_{\text{II}}(j)$ = Type$_{\text{III}}(j)$ = false).

9 输出以下内容：“考虑一次迭代 $f^z(r)$，其中 $z \in \mathbb{Z}/l\mathbb{Z}$。令 $t := z$ 模 ℓ，且 $y := (z-t)/\ell \in \mathbb{Z}/(l/\ell)\mathbb{Z}$。根据 t，我们得到 $\mathbb{Z}/(l/\ell)\mathbb{Z}$ 的算术划分 $\mathcal{P}^{(i_t)}$，而 y 所属的 $\mathcal{P}^{(i_t)}$ 中的块决定了 Tree$^{\Gamma_f}(f^z(r))$ 的同构类型。更确切地说，对于每个 $n \in \mathcal{N}$，我们有逻辑符号元组集合 $\overline{S}_{n,i_t}$，使得当且仅当存在某些 $\vec{\nu} \in \overline{S}_{n,i_t}$，满足 $y \in \mathcal{B}(\mathcal{P}^{(i_t)}\vec{\nu})$ 时，Tree$^{\Gamma_f}(f^z(r)) = \mathfrak{I}_n$。算术划分 $\mathcal{P}^{(i_t)}$ 及其对应的逻辑符号元组集合 $\overline{S}_{n,i_t}$ 如下所示。”随后输出（计算得到的） $\mathcal{P}^{(i_t)}$ 的生成同余序列，以及每个$t = 0$（$1, \ldots, \ell-1$）对应的集合 $\overline{S}_{n,i_t}$。然后停止。

计算成本: (d^3 mpe$(q-1)2^{d^2\ \text{mpe}(q-1)^{+d}}$log q, 0,0,0,0).

5.3 广义分圆映射的函数图同构问题

在本小节中，我们考虑这样一个算法问题：判定给定的两个 $\mathbb{F}_q$（每个具有固定的指标，但两者的指标不一定相同）的广义分圆映射的函数图是否为同构有向图。我们注意到，得益于Babai [8]（另见 Helfgott的科普文章 [29], 或其英文译本 [30]），已有一种用于判定（有向）图同构的通用算法，其关于顶点数量是准多项式的。假设这些图

问题中的图是通过逐一指定其边（作为顶点的有序对或无序对）给出的，该算法关于输入长度是准多项式的（不妨假设不存在孤立顶点，因此顶点数和边数彼此处于二次界内）。

然而，我们并不以这种形式指定我们的函数图 Γ_f，而是以其分圆形式(1) 指定 f，因此我们的输入长度仅为 $O(d\log q)$，而顶点数为 $|\mathbb{F}_q| = q$。我们认为，判定两个定义在 $\mathbb{F}_q$ 上、且其指标至多为 d 的广义分圆映射的函数图是否同构是一个困难问题（即，对于固定的 d，该问题通常不能在 $\log q$ 的多项式时间内求解）。然而，对于某些特殊情形，可以开发出高效算法，本小节旨在介绍这些特殊情形及相关的判定算法。

5.3.1 特殊情形：指标 1

$\mathbb{F}_q$ 的指标 1 广义分圆映射与单项式映射$x \mapsto ax^r$相同，其中 $a \in \mathbb{F}_q$ 且 $r \in \{1, \dots, q-1\}$（如果公式 (1) 中有 $r_0 = 0$ 且$d = 1$，则可以将其替换为 $q-1$，从而得到一个适用于所有 $\mathbb{F}_q$的公式）。若要涵盖 $\mathbb{F}_q$的所有单项式映射，就必须包括“$r = 0$”这一情形（此时$0_{\mathbb{F}_q}$ 不一定是固定的），本小小节将对此进行讨论。多亏了 Deng [19], 的早期研究，我们很容易判断 $\mathbb{F}_q$的两个单项式映射（例如 $f\colon x \mapsto ax^r$ 和 $f'\colon x \mapsto a'x^{r'}$）的函数图是否同构，下面将对此作出说明。

首先，我们注意到，如果 f 为常数——当且仅当 $r = 0$ 或 $a = 0_{\mathbb{F}_q}$ 时会出现这种情况——那么 $\Gamma_f \cong \Gamma_{f'}$ 当且仅当 f' 也为常数，即当且仅当 $r' = 0$ 或 $a' = 0_{\mathbb{F}_q}$。检查是否适用这一特殊情形，只需进行 $O(\log q)$ 次位运算（扫描 a、a'、r、r' 的值）。

因此，我们可以假设 r、r'、a、a' 均非零。由于 a、$a' \neq 0_{\mathbb{F}_q}$，我们有 $f(\mathbb{F}_q^*) \cup f'(\mathbb{F}_q^*) \subseteq \mathbb{F}_q^*$。按照引言中所述的过程，我们可以分别将一个仿射映射 A、A' 与 f、f' 关联起来，它们分别定义在 $\mathbb{Z}/(q-1)\mathbb{Z}$ 上，并满足 $\Gamma_f \cong \Gamma_{f'}$ 当且仅当 $\Gamma_A \cong \Gamma_{A'}$。计算 A 和 A' 具有 q-有界查询复杂度 $(\log^{1+o(1)} q, 1, 0, 0, 0)$；除了一些简单的算术运算外，还需要计算以 ω 为底的 a 和 a' 的离散对数（ω 是 $\mathbb{F}_q$ 的“自然选择”本原元——参见引理 5.1.6(4) 的证明）。

为了判断 $\Gamma_A \cong \Gamma_{A'}$，我们使用以下结果，它是 [19,定理 11]的一个变体。

定理 5.3.1.1. 设 $m = p_1^{v_1} \cdots p_K^{v_K}$为一个正整数，其素因数分解如下所示。令 a、a'、b、$b' \in \mathbb{Z}/m\mathbb{Z}$，并分别将 A和 A'记为 $x \mapsto ax + b$和 $x \mapsto a'x + b'$在 $\mathbb{Z}/m\mathbb{Z}$上的仿射映射。此外，定义

- $\mathfrak{Y} := \{p_j\colon j \in \{1, 2, \dots, K\}$、$p_j \nmid a$、$A$模 $p_j^{v_j}$ 有一个不动点}；对于 $\mathfrak{Y}'$同理，只需用 a'和 A'分别替代 a和 A；并且

分别令 A 和 A' 在各自的周期点上的最小周期长度为• l 和 l'。

那么，当且仅当以下两项均成立时， $\Gamma_A \cong \Gamma_{A'}$：

(1) 最大公约数(a, m) =最大公约数(a', m)，以及 (2) $\mathfrak{Y} = \mathfrak{Y}'$, $l = l'$，$\mathrm{ord}_{p^{\nu_p(m)}}(a^l)$ $= \mathrm{ord}_{p^{\nu_p(m)}}((a')^l)$ 对所有 $p \in \mathfrak{Y}$，并且如果 $2 \in \mathfrak{Y}$ 和$\nu_2(m) > 1$，那么 $\mathrm{ord}_4(a^l) = \mathrm{ord}_4((a')^l)$。

证明。这与 [19, 定理 11] 相同，只是条件(2)中，我们不要求对于 $\mathfrak{Y}$中素因子属于 $\mathfrak{Y}$ 的 m 的所有因子 t ，均有$\mathrm{ord}_t(a^l) = \mathrm{ord}_t((a')^l)$。然而，我们的条件 (2) 已经足够，因为如果该条件成立且 $t = \prod_{p\in\mathfrak{Y}} p^{v'_p}$整除 m，那么对于所有 $p \in \mathfrak{Y}$，都有

$$\mathrm{ord}_{p^{v'_p}}(a^l) = \mathrm{ord}_{p^{v'_p}}((a')^l), \tag{26}$$

因此

$$\mathrm{ord}_t(a^l) = \mathrm{lcm}_{p\in\mathfrak{Y}} \mathrm{ord}_{p^{v'_p}}(a^l) = \mathrm{lcm}_{p\in\mathfrak{Y}} \mathrm{ord}_{p^{v'_p}}((a')^l) = \mathrm{ord}_t((a')^l).$$

要看出公式 (26) 成立，我们进行分类讨论。

- 如果 $p > 2$，设 $\mathfrak{r}$ 是 $\mathbb{Z}/p^{\nu_p(m)}\mathbb{Z}$的一个原根。由于 $\mathrm{ord}_{p^{\nu_p(m)}}(a^l)$ $=\mathrm{ord}_{p^{\nu_p(m)}}((a')^l)$，可知模 $p^{\nu_p(m)}$ 下有 $a^l = \mathfrak{r}^k$ 以及 $(a')^l = \mathfrak{r}^{k'}$，其中 $\gcd(k, \phi(p^{\nu_p(m)})) = \gcd(k', \phi(p^{\nu_p(m)}))$。由于 $\phi(p^{v'_p})$ 整除 $\phi(p^{\nu_p(m)})$，可知 $\gcd(k, \phi(p^{v'_p}))$ $=\gcd(k', \phi(p^{v'_p}))$，因此 a^l和 $(a')^l$模 $p^{v'_p}$ 的乘法阶也相同，证毕。

- If $p = 2$, then formula (26) holds by assumption if $\nu_2(m) \le 2$, so let us assume that $\nu_2(m) \ge 3$. We may also assume that $v'_2 < \nu_2(m)$, because there is nothing to show if $v'_2 = \nu_2(m)$. By the structure of $\mathbb{Z}/2^{\nu_2(m)}\mathbb{Z}$, modulo $2^{\nu_2(m)}$ we can write $a^l = (-1)^\epsilon 5^k$ and $(a')^l = (-1)^{\epsilon'} 5^{k'}$ with $\epsilon, \epsilon' \in \{0, 1\}$. Since a^l and $(a')^l$ have (by assumption) the same multiplicative order modulo 4, we infer that $\epsilon = \epsilon'$. It is not hard to see that if $\mathrm{ord}_{2^{\nu_2(m)}}(a^l) = \mathrm{ord}_{2^{\nu_2(m)}}((a')^l) \in \{1, 2\}$, then $\mathrm{ord}_{2^{v'_2}}(a^l) = \mathrm{ord}_{2^{v'_2}}((a')^l) = 1$. Indeed, this is clear if both orders modulo $2^{\nu_2(m)}$ are equal to 1, and if both orders modulo $2^{\nu_2(m)}$ are equal to 2, then $\mathrm{ord}_{2^{\nu_2(m)}}(5^k), \mathrm{ord}_{2^{\nu_2(m)}}(5^{k'})$ both divide 2, whence due to $v'_2 < \nu_2(m)$, one has $5^k \equiv 5^{k'} \equiv 1 \pmod{2^{v'_2}}$, as follows by comparing the multiplicative orders of 5 modulo $2^{\nu_2(m)}$ and modulo $2^{v'_2}$, respectively. We may thus assume that the common multiplicative order modulo $2^{\nu_2(m)}$ of a^l and $(a')^l$ is strictly greater than 2. Then

  $$\mathrm{ord}_{2^{\nu_2(m)}}(5^k) = \mathrm{ord}_{2^{\nu_2(m)}}(a^l) = \mathrm{ord}_{2^{\nu_2(m)}}((a')^l) = \mathrm{ord}_{2^{\nu_2(m)}}(5^{k'}),$$

  并且通过与“$p > 2$”情形类似的论证，我们得出 $\mathrm{ord}_{2^{v'_2}}(5^k)$ $=\mathrm{ord}_{2^{v'_2}}(5^{k'})$。因此，

  $$\begin{aligned}\mathrm{ord}_{2^{v'_2}}(a^l) &= \mathrm{lcm}(\mathrm{ord}_{2^{v'_2}}((-1)^\epsilon), \mathrm{ord}_{2^{v'_2}}(5^k)) = \mathrm{lcm}(\mathrm{ord}_{2^{v'_2}}((-1)^\epsilon), \mathrm{ord}_{2^{v'_2}}(5^{k'})) \\ &= \mathrm{ord}_{2^{v'_2}}((a')^l),\end{aligned}$$

  如要求。

□

利用定理 5.3.1.1，我们可以证明如下结论。

推论5.3.1.2。设 m 为正整数，并设 A 和 A' 为 $\mathbb{Z}/m\mathbb{Z}$ 的仿射映射。判定 $\Gamma_A \cong \Gamma_{A'}$ 是否具有 m-有界查询复杂度

$$(\log^{2+o(1)} m, 0, \log m, 1, 0)$$

且受 m限制的拉斯维加斯双重复杂度

$$(\log^{8+o(1)} m, \log^{4+o(1)} m, \log^2 m).$$

证明。我们采用定理 5.3.1.1 中的记号，包括 $A(x) = ax + b$ and$A'(x) = a'x + b'$。我们证明，在指定的 m 有界查询复杂度内，可以验证定理 5.3.1.1 中给出的条件（利用引理 5.1.7，即可直接推出所断言的拉斯维加斯双重复杂度）。首先，我们计算并比较 gcd(a, m) 和 gcd(a', m)；根据引理 5.1.5(8)，这需要 $O(\log^{1+o(1)} m)$ 次位操作。接下来，我们通过一次有界 mdl 查询对 $m = p_1^{v_1} \cdots p_K^{v_K}$ 进行因数分解（另见第 5.2 小节开头）。

接下来，我们计算集合 $\mathfrak{Y}$ 和 $\mathfrak{Y}'$ 并进行比较。为此，我们注意到， $A \bmod p_j^{v_j}$ 有不动点，当且仅当同余式 $(a-1)x \equiv -b \pmod{p_j^{v_j}}$ 可解；同样， $A' \bmod p_j^{v_j}$ 有不动点，当且仅当同余式 $(a'-1)x \equiv -b' \pmod{p_j^{v_j}}$ 可解。这些条件分别等价于 gcd($a-1, p_j^{v_j}$) $\mid b$ 和 gcd($a'-1, p_j^{v_j}$) $\mid b'$。为了检验这些条件是否成立，我们从上述 mdl 查询的输出中读出 $p_j^{v_j}$ 的二进制表示，然后分别进行相关算术运算。根据引理 5.1.5(1,3,8)，对单个 j 执行这些操作需要 $O(\log^{1+o(1)} m)$ 次位运算。由于 j 有 $O(\log m)$ 个不同取值，计算 $\mathfrak{Y}$ 和 $\mathfrak{Y}'$ 并检查它们是否相等，总共需要 $O(\log^{2+o(1)} m)$ 次位运算。

如果 $\mathfrak{Y} = \mathfrak{Y}'$，我们接下来计算最小周期长度 l 和 l'，并检查它们是否相等。由我们的表 2（或 [13, 表 3 和 4], ，其中展示的是有限主循环群上仿射映射的循环类型，而不是 CRL 列表，并且可以从中更直接地读出周期长度）可知，模素数幂时，仿射置换的周期长度在整除关系下是线性有序的。因此， l分别是 l'的最小周期长度的最小公倍数，模 $p_j^{v_j}$ 下满足 $p_j \nmid a$、分别满足 $p_j \nmid a'$的那些 $j \in \{1, 2, \ldots, K\}$ 的最小周期长度。可以根据我们的表 2（或 [13, 表 3 和 4]）计算这些最小周期长度。更具体地说，我们遍历$j = 1$、2、.....、K，并对其中每个值执行以下操作。

- 我们检查 $p_j \nmid a$（分别为 $p_j \nmid a'$）是否需要 O（$\log^{1+o_{(1)}} m$）位运算。

- 若是这样，我们检查表2（或 [13, 表3 或4，分别地]）适用的情况，这同样需要 O ($\log^{1+o_{(1)}} m$) 次位运算。
- 最后，我们根据针对具体情况的公式计算最小周期长度并对其进行因数分解，使用一次 mdl 查询和 $O(\log^{1+o_{(1)}} m)$ 次位运算。

对于所有 j，这一过程的查询复杂度有界，为 m

$$(\log^{2+o(1)} m, 0, \log m, 0, 0).$$

接下来，我们确定所计算周期长度的最小公倍数。这些长度已经完成因数分解，因此只需遍历$O(\log m)$个素数，这些素数至少整除其中一个周期长度（注意，每个素数都是某个素数 $p \mid m$ 的 $p(p-1)$ 的因数），并对每个素数计算其出现时的最大指数。为此，对于每个 j，我们需要扫描所得的模$p_j^{v_j}$ 最小周期长度的因式分解，该因式分解是长度为 $O(\log p_j^{v_j})$ 的位串；同时，我们需要将存储的中- 间最大值与其中的某个素数指数进行比较，而该指数是长度为 $O(\log\log p_j^{v_j})$ 的位串。总而言之，计算 l 和 l' 需要 $O(\log^{2+o(1)} m)$次位运算，而检查 l 和 l' 是否相等只需 $O(\log\ m)$次位运算。

若 $l = l'$，我们接下来计算 $a^l \bmod m$ 和$(a')^l \bmod m$，这需要 $O(\log^{2+o(1)} m)$ 位运算。随后，对于每个 $p \in \mathfrak{Y}$，我们计算 $\mathrm{ord}_{p^{\nu_p(m)}}(a^l)$ 和 $\mathrm{ord}_{p^{\nu_p(m)}}((a')^l)$，这可以通过仅两次MORD查询对所有 p 一起完成，并检查它们是否相等；对所有 p 一起进行检查需要 $O(\log m)$ 位运算。最后，如有必要，我们计算 $\mathrm{ord}_4(a^l)$ 和$\mathrm{ord}_4((a')^l)$ 并检查它们是否相等，这可以通过 $O(\log^{1+o(1)} m)$ 位运算完成（无需查询）。 □

关于我们对次数为 1 的广义分圆映射的应用，我们注意到推论 5.3.1.2 的以下结果。

推论 5.3.1.3. 设 f 和 f' 是 $\mathbb{F}_q$ 的单项式映射，以多项式形式给出。在 q 有界查询复杂度下

$$(\log^{2+o(1)} q, 1, \log q, 1, 0),$$

或 q-有界的拉斯维加斯双重复杂度

$$(\log^{8+o(1)} q, \log^{4+o(1)} q, \log^2 q)$$

可以判定是否 $\Gamma_f \cong \Gamma_{f'}$。

证明。正如本小小节开头所解释的，首先通过对输入进行简单扫描来处理 f 或 f' 中至少一个为常数的情形，耗费 $O(\log\ q)$位运算。否则， $f(x) = ax^r$ 和 $f'(x) = a'x^{r'}$，其中 a、a'、r、r' 均非零。在取对数变换下， f分别限制到 $\mathbb{F}_q^*$上，以及 f' 限制到 $\mathbb{F}_q^*$上，对应于 $\mathbb{Z}/(q-1)\mathbb{Z}$的仿射映射 $A(x) = rx+\log_\omega(a)$ 和 $A'(x) = r'x+\log_\omega(a')$，计算这些仿射映射需要 q-有界查询复杂度$(\log q, 1, 0, 0, 0)$。最后，应用推论 5.3.1.2 证明中的算法，在 q-有界查询复杂度 $(\log^{2+o(1)} q, 0, \log\ q, 1, 0)$ 内检查 $\Gamma_A \cong \Gamma_{A'}$ 是否成立，这等价于 $\Gamma_f \cong \Gamma_{f'}$。

□

将我们的定理 5.3.1.1 与 Deng 的原始版本 [19, 定理 11], 进行比较，我们注意到，额外的简化在于实际上只需检验等式

$\mathrm{ord}_t(a^l) = \mathrm{ord}_t((a')^l)$ 对于 m 的那些形如 $p^{\nu_p(m)}$ 的因子 t 成立；与检查 m 的所有因子相比，这对于使相关算法达到多项式复杂度至关重要。这是因为 m 的不同（正）因子数 $\tau(m)$ 可能关于 $\log m$ 呈超多项式增长。

另一方面，狄利克雷证明了一个结果，该结果表明，τ 在 $\{1, 2, \ldots, N\}$ 的初始段 $\mathbb{N}^+$ 上的平均值渐近等价于 $\log N$ [7, 定理 3.3]，因此特别地，对于每个常数 $\epsilon > 0$，满足 $\tau(m) \geq \log^{1+\epsilon} m$ 的正整数 m 集合的渐近密度为 0，因为否则，如果上述渐近密度为 $\delta > 0$，则存在无穷多个 $N \in \mathbb{N}^+$ 使得

$$N^{-1} \sum_{m \leq N} \tau(m) \geq N^{-1} \sum_{m=1}^{\lfloor \delta N \rfloor} \log^{1+\epsilon} m \geq (2N)^{-1} \frac{\delta}{2} N \log^{1+\epsilon} \left(\frac{\delta}{2} N \right) = \frac{\delta}{4} \log^{1+\epsilon} \left(\frac{\delta}{2} N \right)$$
$$\geq \frac{\delta}{8} \log^{1+\epsilon} N >> \log N,$$

矛盾。关于 $\tau(q-1)$ 的平均值，其中 q 遍历素数幂，我们有以下结果，该结果用于子小节 5.3.2。在此背景下，作者谨向 Ofir Gorodetsky致谢，他友好地指出了 Halberstam 的关键论文 [26] 以及命题 5.3.1.4的部分证明；这些内容是在第一作者于MathOverflow[2] 上发布的一个问题的回答中指出的。

命题 5.3.1.4. 对每个 $\epsilon > 0$，都存在常数 c_ϵ、$c'_\epsilon > 0$，使得对所有素数幂 q，除去渐近占比小于 ϵ 的部分，以下结论均成立：

(1) $\mathrm{mpe}(q-1) < c_\epsilon$；以及 (2) $q-1$ 的不同的素因子的个数小于 $c'_\epsilon \log \log q$。

特别地，对所有这类素数幂 q，有 $\tau(q-1) < \log^{c''_\epsilon} q$，其中 $c''_\epsilon := \log(c_\epsilon + 1) c'_\epsilon$。

证明。在整个证明中，变量 q 遍历素数幂，而 p 遍历素数。陈述（1）与命题 5.1.10(1) 相同。对于陈述（2），照例记 $\omega(m)$ 为 $m \in \mathbb{N}^+$ 的不同的素因子的个数。Halberstam 证明了

$$\left(\sum_{p \leq x} 1 \right)^{-1} \sum_{p \leq x} \omega(p-1) \sim \log \log x \tag{27}$$

如 $x \to \infty$ 所示，参见 [26, 定理 1]。现在，固定 $\epsilon > 0$，并假设存在某个常数 $c > 0$，使得对至少占所有素数幂 q 的 ϵ 的渐近比例，有 $\omega(q-1) \geq c \log \log q$。为了证明结论 (2)，我们需要用 ϵ 来界定 c。现在，由于真素数幂在所有素数幂中构成一个密度为 0 的子集（参见命题 5.1.10 的证明），我们得出，对至少占所有素数 p 的 ϵ 的渐近比例，也有 $\omega(p-1) \geq c \log \log p$。我们固定一个足够大的 $x \geq 2$，使得对至少占所有素数 $p \leq x$ 的 $\epsilon/2$ 的比例，有

[2]参见 https://mathoverflow.net/questions/436134/average-value-of-the-prime-omega-function-omega-on-

$\omega(p-1) \geq c\log\log p$。令 p_m 表示满足 $m \in \mathbb{N}^+$ 的第 m 个素数。由于当 m 足够大时，$p_m \geq \frac{1}{2}m\log m$，且素数 $p \leq x$ 的总数至少为 $x/(2\log x)$，因此

$$\begin{aligned}\sum_{p\leq x}\omega(p-1) &\geq \sum_{m=1}^{\lfloor \epsilon x/(4\log x)\rfloor} c\log\log p_m \geq \sum_{m=\lceil \epsilon x/(8\log x)\rceil}^{\lfloor \epsilon x/(4\log x)\rfloor} c\log\log\left(\frac{1}{2}m\log m\right)\\ &\geq \frac{\epsilon x}{16\log x}\cdot c\log\log\left(\frac{1}{2}\cdot\frac{\epsilon x}{8\log x}\cdot\log\left(\frac{\epsilon x}{8\log x}\right)\right),\end{aligned}$$

由此

$$\begin{aligned}&\left(\sum_{p\leq x}1\right)^{-1}\sum_{p\leq x}\omega(p-1)\\ &\geq \left(2\frac{x}{\log x}\right)^{-1}\cdot\frac{\epsilon x}{16\log x}\cdot c\log\log\left(\frac{1}{2}\cdot\frac{\epsilon x}{8\log x}\cdot\log\left(\frac{\epsilon x}{8\log x}\right)\right)\\ &\sim\frac{\epsilon c}{32}\log\log x,\end{aligned}$$

这意味着

$$\left(\sum_{p\leq x}1\right)^{-1}\sum_{p\leq x}\omega(p-1) \geq \frac{\epsilon c}{64}\log\log x$$

如果 x 足够大。因此，为了不与Halberstam 的 (27) 相矛盾，我们必须有 $c \leq 64/\epsilon$，即关于 ϵ 的 c 上界，这是陈述 (2) 按阶成立所要求的。

最后，对于“特别地”这一陈述，我们注意到，由于如果 $q-1 = p_1^{v_1}p_2^{v_2}\cdots p_K^{v_K}$ 是 $q-1$ 的素因数分解，则 $\tau(q-1) = (v_1+1)(v_2+1)\cdots(v_K+1)$，因此可以按如下方式从上界估计 $\tau(q-1)$：

$$\begin{aligned}\tau(q-1) &\leq (\mathrm{mpe}(q-1)+1)^{\omega(q-1)} \leq (c_\epsilon+1)^{c'_\epsilon\log\log q} = \exp(c'_\epsilon\log\log q\log(c_\epsilon+1))\\ &= (\log q)^{\log(c_\epsilon+1)c'_\epsilon}.\end{aligned}$$

□

我们以两组伪代码结束本小小节。首先，给出用于检验 $\Gamma_A \cong \Gamma_{A'}$ where $A: x \mapsto ax+b$and $A': x \mapsto a'x+b'$ 是否为 $\mathbb{Z}/m\mathbb{Z}$ 的仿射映射的算法伪代码（参见推论 5.3.1.2 及其证明）。与上一小节一样，我们注明每个步骤的（m-有界）查询复杂度（计算成本）。

1 计算最大公约数(a,m)和最大公约数(a',m)，并检查它们是否相等。如果不相等，输出“false”并停止。计算成本：$(\log^{1+o(1)} m$,0, 0, 0, 0)。 2 对 $m = p_1^{v_1}\cdots p_K^{v_k}$ 进行因数分解。计算成本：$(\log\ m$,0, 1, 0, 0)。

3 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} m, 0, 0, 0, 0)$.

3.1 Check whether it is the case that $p_j \nmid a$ and $\gcd(a-1, p_j^{v_j}) \mid b$. If so, set $\text{test}_j :=$ true, otherwise set $\text{test}_j :=$ false.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

3.2 Check whether it is the case that $p_j \nmid a'$ and $\gcd(a'-1, p_j^{v_u}) \mid b'$. If so, set $\text{test}'_j :=$ true, otherwise set $\text{test}'_j :=$ false.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

3.3 If $\text{test}_j \neq \text{test}'_j$, then output "false" and halt.
QC: $(\log \log m, 0, 0, 0, 0)$.

4 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} m, 0, \log m, 0, 0)$.

4.1 Check whether $p_j \nmid a$. If not, set $\text{Test}_j :=$ false and skip to Step 4.4. Otherwise, set $\text{Test}_j :=$ true.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.2 Check which case in Table 2 (or [13, Table 3 or 4 respectively]) applies to $A \bmod p_j^{v_j}$, using simple arithmetic.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.3 Determine the minimal cycle length l_j of $A \bmod p_j^{v_j}$ according to Table 2 (or [13, Table 3 or 4 respectively]) and factor it.
QC: $(\log^{1+o(1)} m, 0, 1, 0, 0)$.

4.4 Check whether $p_j \nmid a'$. If not, set $\text{Test}'_j :=$ false and skip to the next j. Otherwise, set $\text{Test}'_j :=$ true.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.5 Check which case in Table 2 (or [13, Table 3 or 4 respectively]) applies to $A' \bmod p_j^{v_j}$, using simple arithmetic.
QC: $(\log^{1+o(1)} m, 0, 0, 0, 0)$.

4.6 Determine the minimal cycle length l'_j of $A' \bmod p_j^{v_j}$ according to Table 2 (or [13, Table 3 or 4 respectively]) and factor it.
QC: $(\log^{1+o(1)} m, 0, 1, 0, 0)$.

5 计算

$$l := \text{lcm}(l_j : 1 \le j \le K, \text{Test}_j = \text{true}) \text{ and}$$
$$l' := \text{lcm}(l'_j : 1 \le j \le K, \text{Test}'_j = \text{true})$$

使用上述步骤 4.3 和 4.6 中计算出的 l_j 和 l'_j 的因式分解。
计算成本： $(\log^{2+o(1)} m, 0, 0, 0, 0)$。

6 检查是否为 $l = l'$。如果不是，则输出"false"并停止。
计算成本：$(\log\ m, 0, 0, 0, 0)$.

7计算 $a^l \bmod m$ 和 $(a')^l \bmod m$。
计算成本： $(\log^{2+o(1)} m, 0, 0, 0, 0)$。

8分别对满足 $p \nmid a$ 的所有素数 $p \mid m$ 计算ord$_{p^{\nu_p(m)}}(a^l)$和ord$_{p^{\nu_p(m)}}((a')^l)$，特别地，对所有 $p \in \mathfrak{Y} = \mathfrak{Y}' = \{p_j :$ 测试j =真$\}$。计算成本：(log m,0, 0, 1, 0)。9 检查对于所有 $p \in \mathfrak{Y}$，ord$_{p^{\nu_p(m)}}(a^l)$ = ord$_{p^{\nu_p(m)}}((a')^l)$ 是否成立。如果不成立，输出“假”并停止。计算成本：(log m,0, 0, 0, 0)。10 如果 $4 \mid m$，执行以下操作。10.1检查 ord$_4(a^l)$ = ord$_4((a')^l)$是否成立。如果不成立，输出“假”并停止。计算成本：(log$^{1+o(1)}$ m,0,0,0,0)。10.2输出“真”并停止。计算成本：(1, 0,0, 0, 0)。11否则执行以下操作。11.1 输出“真”并停止。计算成本：(1, 0, 0, 0, 0)。

最后，我们给出用于检验 $\Gamma_f \cong \Gamma_{f'}$的算法的伪代码，其中 f：$x \mapsto ax^r$和 f'：$x \mapsto a'x^{r'}$是 $\mathbb{F}_q$的单项式映射（其中 a、$a' \in \mathbb{F}_q$和r、$r' \in \{0, 1, \ldots, q-1\}$）。

1 如果是 $a = 0_{\mathbb{F}_q}$ 或 $r = 0$，则执行以下操作。

1.1 检查是否满足 $a' = 0_{\mathbb{F}_q}$ 或 $r' = 0$。如果是，则输出“true”并停止。否则，输出“false”并停止。计算成本：(log q, 0, 0, 0, 0)。

2 否则，执行以下操作。

2.1 检查 $a' = 0_{\mathbb{F}_q}$ 或 $r' = 0$ 是否成立。若成立，则输出“false”并停止。计算成本：(log q, 0, 0, 0, 0)。 2.2 计算 b:= log$_\omega(a)$ 和 b' := log$_\omega(a')$，其中 ω 是 $\mathbb{F}_q$ 的“自然”本原元选择（参见引理 5.1.6(4) 的证明）。分别将由公式$A(x)$ $= rx + b$和 $A'(x) = r'x + b'$给出的 $\mathbb{Z}/(q-1)\mathbb{Z}$ 的仿射映射记为A和 A'。计算成本：(log q, 1, 0, 0, 0)。 2.3 使用上文的算法检查 $\Gamma_A \cong \Gamma_{A'}$是否成立。若成立，则输出“true”并停止。否则，输出“false”并停止。计算成本：(log $^{2+o(1)}$ q, 0, log q, 1, 0)。

5.3.2 特殊情形：树只取决于陪集

在本小小节中，我们讨论有限域 $\mathbb{F}_q$ 的两类特殊广义圆分映射，使得如果 f_1 和 f_2 各自属于其中一类（不一定属于同一类），且它们的指标分别为 d_1 和 d_2，那么可以判定

是否能在 q-有界查询复杂度下判定 $\Gamma_{f_1} \cong \Gamma_{f_2}$，其中每一项关于参数 $\max\{d_1, d_2\}$、$\log\ q$ 和 $\tau(q-1)$（$q-1$ 的除数个数）均为多项式。在特别情况下，该问题的 q-有界拉斯维加斯双重复杂度相对于输入规模 $O(\max\{d_1, d_2\} \log\ q)$ 始终是次指数的，因为对于每个 $\epsilon > 0$ [7, p, $\tau(m) \in o(m^{\epsilon})$。 296]。此外，根据命题 5.3.1.4，对于每个 $\epsilon > 0$，在所有有限域 $\mathbb{F}_q$ 中，除去渐近比例小于 ϵ 的部分后，上述 q-有界拉斯维加斯双重复杂度关于输入规模是多项式的（次数取决于 ϵ）。

我们考察的广义分圆映射 f，即 $\mathbb{F}_q$的指标为 d的两类映射如下。

- 第 1 类: f将每个陪集 C_i（对 $i \in \{0, 1, \ldots, d-1\}$）映射到$C_d = \{0_{\mathbb{F}_q}\}$，或双射地映射到$C_{\overline{f}\ (i)}$（即，只要仿射函数 A_i 关于 $\mathbb{Z}/s\mathbb{Z}$ 有定义，它就是 $\mathbb{Z}/s\mathbb{Z}$的一个置换）。若此情形成立，我们称 f 为特殊类型 I。这与第 4.3 小节中的情形相同。

- 第 2 类: 诱导函数 $\overline{f}$是 $\{0, 1, \ldots, d\}$ 的一个置换（即， f将 C的陪集置换）。若此情形成立，我们称 f 为特殊类型 II。使用第 3.2 小节中的记号，这意味着 $\Gamma_f = \Gamma_{\text{置换}}$，因此该小节中的讨论适用。

这两种情形共有的关键性质是，根据引理 4.3.1 和定理 3.2.1，分别位于 $i \in \{0, 1, \ldots, d\}$ 的块 C_i 中的 Γ_f 上方的有根树只取决于该块。这使我们能够对 Γ_f 的有向图同构类型给出一种相对紧凑的描述。为此，我们可以对定义 5.1.2 中引入的分区树寄存器概念作如下调整。

定义 5.3.2.1. 设 f 是 $\mathbb{F}_q$ 上的一个指标为 d 的广义分圆映射。

(1) 我们假设 f 是特殊类型 I，因此给定块 C_i 中的所有顶点（不只是 f-周期顶点）在 Γ_f 中上方都有同构的有根树。 f 的 I 型树寄存器是一个有序序列 $((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ N}$，满足以下条件。

(a)集合 $\mathfrak{D}_n$ 构成递归树描述列表，其关联的有根树同构类型为 $\mathfrak{I}_n$（参见定义 5.1.1）；(b) 对于每个 n，集合 S_n 非空，并且恰好由那些满足以下条件的 $i \in \{0, 1, \ldots, d\}$ 组成：对于任意 $x \in C_i$，树$^{\Gamma_f}(x) \cong$ 树$^{\Gamma}(i) \cong \mathfrak{I}_n$。

(2) 我们假设 f 是特殊类型 II，因此对于每个 $i \in \{0, 1, \ldots, d-1\}$， $x \in C_i$ 的有根树同构类型 Tree$^{\Gamma_f}(x)$ 仅取决于 i 以及 x 的$\mathfrak{h}$ 值（见第 4.1 小节，第 56 页起）。对于 f，II 型树寄存器是一个有序序列 $((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ N}$，满足以下条件。

(a) 集合 $\mathfrak{D}_n$ 构成递归树描述列表，其关联的有根树同构类型为 $\mathfrak{I}_n$；(b) $\mathfrak{I}_n$ 正是那些在形如 Expand($\mathfrak{J}_{i,\ h}$)的有根树中出现的同构类型，其中 $i \in \{0, 1, \ldots, d-1\}$ 和 $h \in \{0, 1, \ldots, H_i\}$（有关$\mathfrak{J}_{i,\ h}$的定义，见第 4.1 小节第 56 页起）；以及

(c) 对于每个 n，有 $S_n = (\mathrm{ht}(\mathfrak{I}_n)S_{n,\ \mathrm{trans}},\ S_{n,\ \mathrm{per}})$，其中

$$S_{n,\mathrm{trans}} = \{i \in \{0,1,\ldots,d-1\} : \mathrm{ht}(\mathfrak{I}_n) < H_i \ \textit{and}\ \mathfrak{I}_n = \mathrm{Expand}(\mathfrak{I}_{i,\mathrm{ht}(\mathfrak{I}_n)})\}$$

and

$$S_{n,\mathrm{per}} = \{i \in \{0,1,\ldots,d-1\} : \mathfrak{I}_n = \mathrm{Expand}(\mathfrak{I}_{i,H_i})\}.$$

In an implementation, we assume that the sets S_n from Definition 5.3.2.1(1), as well as the sets $S_{n,\mathrm{trans}}$ and $S_{n,\mathrm{per}}$ from Definition 5.3.2.1(2), are represented by *sorted* arrays each entry of which is a binary digit representation of a number $i \in \{0,1,\ldots,d\}$ with bit length exactly $\lfloor \log_2 d \rfloor + 1$. Moreover, in a type-I tree register, only one of the descriptions $\mathfrak{D}_n$ corresponds to $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, and it is the only description in which the second entries of elements may be larger than d. For the sake of efficiency, we make the convention that all descriptions $\mathfrak{D}_n$ except that one use $\lfloor \log_2 d \rfloor + 1$ digits for representing each entry m or k_m of an element (m, k_m) of $\mathfrak{D}_n$. On the other hand, in the description corresponding to $\mathrm{Tree}_{\Gamma_f}(0_{\mathbb{F}_q})$, we use $\lfloor \log_2 d \rfloor + 1$ digits for the first entry m, and $\lfloor \log_2 q \rfloor + 1$ digits for the second entry k_m. In contrast to this, in a type-II tree register, we know that $m \le d^2\,\mathrm{mpe}(q-1) + d \le d^2 \lfloor \log_2 q \rfloor + d$ (see the first paragraph in the proof of Lemma 5.3.2.2(4) below), while there is a priori no upper bound on k_m other than the trivial one, q. Hence, in such a register, we use $\lfloor \log_2(d^2 \lfloor \log_2 q \rfloor + d) \rfloor + 1$ digits for representing m, and $\lfloor \log_2 q \rfloor + 1$ digits for representing k_m. In either case, the entries of a given description $\mathfrak{D}_n$ all have the same bit length, and different descriptions $\mathfrak{D}_n$ use the same bit length for the first entries of their elements. As before, we assume that the array representing a given description $\mathfrak{D}_n$ is lexicographically ordered (corresponding to the ordering of the elements (m, k_m) of $\mathfrak{D}_n$ by increasing m).

我们注意到，在定义 5.3.2.1(2,c) 中，集合 $S_{n,\mathrm{trans}}$ 恰好由满足以下条件的 $i \in \{0,1,\ldots,d-1\}$ 组成：对所有满足 $\mathfrak{h}(x) = \mathrm{ht}(\mathfrak{I}_n) < H_i$ 的 $x \in C_i$，都有 $\mathfrak{I}_n = \text{树}^{\Gamma_f}(x)$，而不论这样的 x 是否存在；我们从例 3.2.2 中回顾，并非 $\{0,1,\ldots,H_i-1\}$ 中的所有值都必然会由给定陪集 C_i 中处于 f-暂态的点上的 $\mathfrak{h}$ 取到。我们还提醒读者，f-暂态的 $x \in C_i$ 由不等式 $\mathfrak{h}(x) < H_i$ 刻画，并且对于所有这样的 x，都有 $\mathrm{ht}(\text{树}^{\Gamma_f}(x)) = \mathrm{ht}(\mathfrak{I}_{i,\mathfrak{h}(x)}) = \mathfrak{h}(x)$；参见第 4.1 小节第 56 页起关于 $\mathfrak{I}_{i,h}$ 的递归定义。此外，集合 $S_{n,\mathrm{per}}$ 由满足以下条件的 $i \in \{0,1,\ldots,d-1\}$ 组成：对于 f-周期的 $x \in C_i$，$\mathfrak{I}_n$ 是唯一的同构类型树 $^{\Gamma_f}(x) = \mathrm{Expand}(\mathfrak{I}_{i,H_i})$（而 $\mathrm{Expand}(\mathfrak{I}_{i,H_i})$ 的高度不一定是 H_i，但其高度为 $\mathcal{H}_i$）。

接下来，我们讨论以下重要引理。

引理5.3.2.2。设 f 是 $\mathbb{F}_q$的一个指标为 d 的广义分圆映射，以分圆形式 (1) 给出。

(1) 检查 f 是否为特殊类型 I 需要 q 有界查询复杂度

$$(d \log^{1+o(1)} q, d, 0, 0, 0).$$

(2) 如果 f 是特殊类型 I，那么对于 f，具有 $N \in O(d)$ 的 I 型树寄存器 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ \ldots\ldots,\ N}$ 可在 q-有界查询复杂度内计算

$$(d^3 \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0).$$

(3) 检查 f 是否为特殊类型 II 的查询复杂度是 q-有界的

$$(d\log^2 d + d\log^{1+o(1)} q, d, 0, 0, 0).$$

(4) 若 f 属于特殊类型 II，则可在 q-有界查询复杂度内计算 II 型树寄存器 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ \ldots,\ N}$，其中 f 满足 $N \in O(d^2\mathrm{mpe}(q-1))$

$$(d^5 \log^2 d\,\mathrm{mpe}(q-1)^3 + d^5\,\mathrm{mpe}(q-1)^3 \log q + d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q, d, 0, 0, 0).$$

特别地，可在 q-有界查询复杂度内计算

$$(d^{5+o(1)} \log^4 q, d, 0, 0, 0).$$

证明。对于陈述（1），我们首先计算 $\overline{f}$ 和仿射映射 $A_i\colon x \mapsto \alpha_i x + \beta_i$，这需要 q-有界查询复杂度（$d\log^{1+o(1)} q, d, 0, 0, 0$），根据命题 5.1.8。注意，当且仅当对于所有使 A_i 有定义（即 f 的分圆形式（1）中的系数 $a_i \in \mathbb{F}_q$ 非零）的 i，$\gcd(\alpha_i, s) = 1$ 时，f 属于特殊类型 I；根据引理 5.1.5(3,8)，可用 $O(d\log^{1+o(1)} q)$ 次位运算进行检查。

对于陈述（2），我们首先计算 $\overline{f}$、数目 $s = (q-1)/d$，以及（如第 5.2.1 小节第 98 页所示）“层”。

$$\mathrm{Layer}_h := \begin{cases} \mathrm{im}(\overline{f}^h) \setminus \mathrm{im}(\overline{f}^{h+1}), & \text{if } h \in \{0, 1, \ldots, \overline{H} - 1\}, \\ \mathrm{im}(\overline{f}^{\overline{H}}) = \mathrm{per}(\overline{f}), & \text{if } h = \infty, \end{cases}$$

关于$\overline{f}$的迭代排序形式，且不含重复项，其中 $\overline{H}$ 是 $\Gamma_{\overline{f}}$中的最大树高。总的来说，这需要 q-有界查询复杂度（$d\log^{1+o(1)} q + d^2\log^2 d, d, 0, 0, 0$）。之后，我们开始构建寄存器。在该过程中给定的任意时刻，我们至少拥有一个作为中间结果的“部分寄存器”，其中包括有根树 $\mathfrak{I}_n$ 的描述 $\mathfrak{D}_n$ 的定义，适用于所有 $n \in \mathcal{N}$，以及 $\mathbb{N}_0$ 的一个初始段。由于每个 $n \in \mathcal{N}$ 都有一个与之关联的 $\{0, 1, \ldots, d\}$ 的非空子集 S_n，且这些集合两两不相交，我们得出 $|\mathcal{N}| \le d + 1 \in O(d)$。特别地，最终有 $N \in O(d)$，正如所断言的那样。

现在，为了构建寄存器，我们依次对 $h = 0$、1、.....、$\overline{H}$ –1、∞执行以下操作。我们遍历索引 $i \in \mathrm{Layer}_h$，并对其中每一个索引以排序形式计算原像集 $\overline{f}^{-1}(\{i\})$，每个索引需要进行 $O(d\log\ d)$ 次位运算。令j_1、j_2、.....、j_K为$\overline{f}$ 下 $\overline{f}$的 i 暂态原像；如果 $h < \infty$，它们就只是 i的全部原像；如果 $h = \infty$，则还可以通过在 $\mathrm{per}(\overline{f})$ 的 $= \mathrm{Layer}_\infty$ 中确定 i 唯一的 $\overline{f}$ 原像来确定它们。对于所有 $i \in \mathrm{Layer}_\infty$ 总计需要 $O(d\cdot|\overline{f}^{-1}(\{i\})|\cdot\log^2 d) \subseteq O(d^2\log^2 d)$ 次位运算，利用二分查找即可，因为 Layer_∞ 已经排好序。

在下文中，我们假设 i 固定。每个 j_t 都属于唯一的层 $\mathrm{Layer}^{h_{j_t}}$，且满足 $h_{j_t} < h_i = h$，因此存在唯一的非负整数 $\overline{n}_{j_t} \in \mathcal{N}$，使得 $j_t \in S_{\overline{n}_{j_t}}$。计算 $\overline{n}_{j_t}$ 需要 $O(|\mathcal{N}| \cdot \log^2 d) \subseteq O(d\log^2 d)$ 位运算，对于一个

单个 t，因此总共需要 $O(d^2\log^2 d)$ 次位运算（对于这个固定的 i）。现在，根据引理 4.3.1，任何 $x \in C_i$ 上方的有根树都同构于

$$\begin{cases} \mathrm{Tree}_{\Gamma_{\overline{f}}}(i) \cong \sum_{t=1}^{K} \mathfrak{I}^{+}_{\overline{n}_{j_t}}, & \text{if } i < d, \\ \sum_{t=1}^{K} s\mathfrak{I}^{+}_{\overline{n}_{j_t}}, & \text{if } i = d, \end{cases}$$

因此，我们可以为这棵树选择以下紧凑描述 $\mathfrak{D}$。

- 如果 $i < d$，我们设

$$\mathfrak{D} := \{(n, m) : n \in \{\overline{n}_{j_t} : 1 \le t \le K\}, m = |\{t \in \{1, \dots, K\} : \overline{n}_{j_t} = n\}| > 0\}.$$

- 如果 $i = d$，我们设

$$\mathfrak{D} := \{(n, sm) : n \in \{\overline{n}_{j_t} : 1 \le t \le K\}, m = |\{t \in \{1, \dots, K\} : \overline{n}_{j_t} = n\}| > 0\}.$$

在确定数字$\overline{n}_{j_t}$ 后计算 $\mathfrak{D}$ 需要创建$\overline{n}_{j_t}$ 的不同值及其重数列表。使用引理 5.1.5(10)中的排序算法时，这可以在$O(K \log K \log d) \subseteq O(d\log^2 d)$ 位运算内完成。如果 $i < d$，那么对于该i而言，这也是计算 $\mathfrak{D}$ 的总体复杂度；而如果 $i = d$，则计算 $\mathfrak{D}$ 的复杂度为 $O(d\log^2 d + d\log^{1+o(1)} q)$，因为其中还涉及整数乘法。计算出 $\mathfrak{D}$ 后，我们检查是否存在一个 $n \in \mathcal{N}$使得 $\mathfrak{D} = \mathfrak{D}_n$；无论 i的值如何，这都需要 $O(|\mathcal{N}| \cdot d \log d) \subseteq O(d^2 \log d)$ 位运算。事实上，如果 $i = d$，此时 $\mathfrak{D}$ 中元素的第二项的位长度不一定属于 $O(\log d)$，则可以如下进行。注意到第二项只有在这个 i取该值时才会如此之大，首先检查 $s > d$，这只需 $O(\log d)$ 次位运算即可完成（注意，s 本身已在开始时计算完毕）。如果是这样，则可知对于任意 $n \in \mathcal{N}$都有 $\mathfrak{D} \neq \mathfrak{D}_n$；否则，即使对于 $i = d$，$\mathfrak{D}$ 中元素第二项的位长度也属于 $O(\log d^2) = O(\log d)$，因此可以按照处理 $i < d$的方式进行。无论哪种情况，如果对于某个 $n \in \mathcal{N}$有 $\mathfrak{D} = \mathfrak{D}_n$，那么我们通过合并分别对应于集合 S_n和 $\{i\}$的有序列表，将 i 作为新元素添加到 S_n 中；根据引理 5.1.5(11)，这需要 $O(d\log d)$ 位运算。否则，我们创建 $\mathfrak{D}$ 作为一个新的描述$\mathfrak{D}_{n'}$，其中 $n' = \max \mathcal{N} + 1$，并初始化 $S_{n'} := \{i\}$。

由于对于所有此类 i，我们都需要在声明 i 固定后执行所述计算，因此这些计算的总位操作成本为 $O(d^3 \log^2 d + d\log^{1+o(1)} q)$，由此可知，计算 f 的 I 型树寄存器的总体 q-有界查询复杂度正如所述。

对于陈述（3），我们只需计算 $\overline{f}$ 并检查 $\mathrm{im}(\overline{f}) = \{0, 1, \dots, d\}$。前一项任务的查询复杂度为 q-有界

$$(d \log^{1+o(1)} q, d, 0, 0, 0)$$

根据命题 5.1.8，后一项任务需要 $O(d\log^2 d)$ 次位运算（参见第 5.2.1 小节论证的开头）。

对于陈述（4），我们首先计算 $\overline{f}$、仿射映射 $A_i\colon\ x \mapsto \alpha_i x + \beta_i$，以及（如第 5.2.1 小节所示）$\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$ 和 $\overline{f}$ 的循环，并进行 q 有界查询

在($d\log^{1+o(1)} q + d^2\log^2 d, d$,0, 0,0) 中的复杂度。此后，我们开始构建树寄存器，并且如陈述(2)的证明中一样，对于该过程中的一个给定点，我们将由 $\mathcal{N}$ 表示由 $\mathbb{N}_0$ 的所有 n 组成的初始段，其中 $\mathfrak{D}_n$在该点有定义。由于相关的有根树同构类型 $\mathfrak{I}_n$ 两两不同，且是集合 $\{\text{Expand}(\mathfrak{I}_{i,h}) : i \in \{0,1,\ldots,d-1\}, h \in \{0,1,\ldots,H_i\}\}$ 的元素，因此我们有

$$|\mathcal{N}| \le d \cdot (\max_i H_i + 1) \le d^2 \operatorname{mpe}(q-1) + d \in O(d^2 \operatorname{mpe}(q-1))$$

（关于 H_i 的界，参见第 3.3 小节中的公式 (15)）。特别是，最后 $N \in O(d^2\text{mpe}(q-1))$，正如所断言的那样。

为了构建寄存器，我们遍历元素 $(i,\ell) \in \overline{\mathcal{L}}$ with $i < d$ （注意，$\text{Tree}^{\Gamma_f}(0_{\mathbb{F}_q})$ 显然是平凡的，根据定义甚至不会记录在寄存器中），并对其中每个元素执行以下操作。首先，我们计算 H_i 的精确值，即 $\bigcup_{t=0}^{\ell-1} C_{i_t}$ 中周期顶点上方的最大树高，其中 $(i_0, i_1, \ldots, i_{\ell-1})$ with $i = i_0$ 是 i 的 $\overline{f}$-循环。我们注意到，根据定理 3.2.1 及其前面的段落（已在第 4.1 小节中详细推导），有以下结论：对于每个 $t \in \{0,1,\ldots,\ell-1\}$，$C_{i_t}$ 中周期顶点上方的树两两同构，因此具有相同的高度 $\mathcal{H}_{i_t}$。我们有 $H_i = \max\{\mathcal{H}_{i_t} : t = 0, 1, \ldots, \ell-1\}$，因此我们计算数值 $\mathcal{H}_{i_t}$ 以得到 H_i。此时我们注意到，事实上，在第 5.2.2 小小节中，我们已经描述了如何通过二分查找找到 H_i 。然而，这里我们还希望存储 $t \in \{0,1,\ldots,\ell-1\}$ 和 $k \in \{1,2,\ldots,\mathcal{H}_{i_t}\}$ 的繁衍数 $\text{proc}_{i_t,\,k}$，因此不使用二分查找来跳过步骤。

我们提醒读者，对于任意 $t \in \mathbb{Z}$，记号 i_t 是$i_{t \bmod \ell}$的简写。对于 $t = 0,1,\ldots,\ell-1$ 以及依次对于 $k = 1,2,\ldots,$ 我们计算（并存储）繁衍数（参见定理 3.2.1）

$$\text{proc}_{i_t,k} = \frac{\gcd(\prod_{r=0}^{k-1} \alpha_{i_{t-k+r}}, s)}{\gcd(\prod_{r=0}^{k-2} \alpha_{i_{t-k+r}}, s)}$$

直到给定 t 时首次 $\text{proc}_{i_t,\,k} = 1$ ，这恰好发生在$k = \mathcal{H}_{i_t} + 1$时。如果我们存储公式中出现的两个乘积的值，以计算 $\text{proc}_{i_t,\,k}$，那么每个 $\text{proc}_{i_t,\,k}$ 的计算只涉及 $O(1)$ 次乘法，因此根据引理 5.1.5(3,8)，其位操作成本为 $O(\log^{1+o(1)} q)$。因此，由于 $\mathcal{H}_{i_t} \in O(\ell\text{mpe}(q-1))$ ），我们可以用 $O(\ell \operatorname{mpe}(q-1) \log^{1+o(1)} q)$ 次位操作计算每个单独的 $\mathcal{H}_{i_t}$，由此计算总共需要 $O(\ell^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q)$ 次位操作。

计算出 H_i 后，我们开始将与 i 的$\overline{f}$-循环相关的信息添加到树寄存器中。具体来说，我们依次对$h = 0$、1、…、H_i执行以下操作。如果 $h = 0$，那么对于所有 t，$\text{Expand}(\mathfrak{I}_{i_t,h})$ 都是平凡的，因此在 $\mathcal{N} = \emptyset$的情况下（这只会发生在我们考虑的第一个对 $(i,\ell) \in \overline{\mathcal{L}}$ 中），我们设 $\mathfrak{D}_0 := \emptyset$，使得$\mathfrak{I}_0$ 成为平凡有根树（特别是，$\text{ht}(\mathfrak{I}_0) = 0$），并按如下方式初始化一些变量。

- 我们令

$$S_{0,\text{trans}} := \begin{cases} \{i_0, i_1, \ldots, i_{\ell-1}\}, & \text{if } H_i > 0, \\ \emptyset, & \text{otherwise.} \end{cases}$$

- We set $S_{0,\mathrm{per}} := \{i_t : t \in \{0,1,\dots,\ell-1\}, \mathcal{H}_{i_t} = 0\}$.

提醒读者，我们希望表示$S_{0,\ \mathrm{trans}}$ 和 $S_{0,\ \mathrm{per}}$ 的数组已排序，因此应应用引理 5.1.5(10) 中的排序算法，该算法对每个数组需要进行$O(d\log^2 d)$次位运算。

在另一种情况下，其中 $\mathcal{N} \neq \emptyset$，我们进行如下操作。

- 如果 $H_i > 0$，我们将 i_0、i_1.....、$i_{\ell-1}$ 添加到已定义的集合 $S_{0,\ \mathrm{trans}}$ 中作为新元素（严格来说，我们对 $\{i_0, i_1, \dots, i_{\ell-1}\}$ 进行排序，并将其与 $S_{0,\ \mathrm{trans}}$合并）。
- 我们还将所有满足 $\mathcal{H}_{i_t} = 0$ 的指标 i_t（其中 $t \in \{0,1,\dots,\ell-1\}$）作为新元素添加到已定义的集合$S_{0,\ \text{每个}}$ 中。

利用引理 5.1.5(10,11)可知，将情况 $h = 0$ 作为一个整体处理，只需进行 $O(\ell \log^2 d + d\log d)$ 次位运算（用于复制信息和排序/合并，以及对值 $\mathcal{H}_{i_t}$ 进行简单查找）。

现在我们假设 $h \geq 1$。边加权根树（同构类型）$\mathfrak{I}_{i_t,h}$ 绘制于第 4.1 小节末尾（我们提请读者注意 $h < H_i$ 与 $h = H_i$ 之间的分类讨论），并如下计算 Expand($\mathfrak{I}_{i_t,h}$) 的描述$\mathfrak{D} = \mathfrak{D}(i_t, h)$。首先，我们设

$$h' := \begin{cases} h, & \text{if } h < H_i, \\ \mathcal{H}_{i_t}, & \text{if } h = H_i, \end{cases}$$

这就是 Expand($\mathfrak{I}_{i_t,h}$) 的高度。它也是 $\mathfrak{I}_{i_t,h}$ 中以根为其终端顶点的边的数量（即该根的无权入度）。我们注意到，其中一些边的权重可能为 0。对于 $k = 0$、1、.....、$h' - 1$，我们计算

$$\mathfrak{w}_k := \begin{cases} \mathfrak{w}_{i_t,k} = \mathrm{proc}_{i_t,k+1} - \mathrm{proc}_{i_t,k+2}, & \text{if } h = H_i, \text{ or } h < H_i \text{ and } k < h' - 1, \\ \mathrm{proc}_{i_t,h}, & \text{if } h < H_i \text{ and } k = h' - 1, \end{cases}$$

即$\mathfrak{I}_{i_t,h}$ 中（从图中的左侧数起）以根为其终端顶点的第（$k+1$）条边的权重。这些计算只需要

$$O(\ell \cdot h' \cdot \log q) \subseteq O(\ell^2 \,\mathrm{mpe}(q-1) \log q)$$

对所有 t 一起进行位运算。然后我们可以设置

$$\mathfrak{D} := \{(\overline{n}_{i_t,k}, \mathfrak{w}_k) : k = 0, 1, \dots, h' - 1\}$$

其中$\overline{n}_{i_t,k}$ 是唯一的 $n \in \mathcal{N}$ 满足 $\mathfrak{I}_n$ = Expand($\mathfrak{I}_{i_t,k}$) 的元素，即满足 $i_t \in S_{n,\mathrm{trans}}$ 且ht($\mathfrak{I}_n$) $= k$。假设各种 $\mathfrak{I}_n$ 的高度在寄存器更新时都会被存储，则计算 $\mathfrak{D}$ 需要

$$\begin{aligned} &O(h' \cdot (|\mathcal{N}| \cdot \log^2 d + \log\log q)) \\ \subseteq &O(\ell \,\mathrm{mpe}(q-1) \cdot (d^2 \,\mathrm{mpe}(q-1) \cdot \log^2 d + \log\log q)) \\ \subseteq &O(\ell d^2 \log^2 d \,\mathrm{mpe}(q-1)^2 + \ell \,\mathrm{mpe}(q-1) \log\log q) \end{aligned}$$

针对单个 t 的位运算（用于确定 $\overline{n}_{i_t,k}$），因此

$$O(\ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3 + \ell^2 d \operatorname{mpe}(q-1)^2 \log\log q)$$

对所有 t 和 h 一起进行位运算。

一旦 $\mathfrak{D}$ 已计算完成，我们需要检查它是否已经出现在 $n \in \mathcal{N}$ 的 $\mathfrak{D}_n$ 中（并相应更新寄存器）。如果我们按字典序对 $\mathfrak{D}$ 排序，就可以通过线性比较各项将其与给定的 $\mathfrak{D}_n$ 进行比较，因此，检查对于某个 $n \in \mathcal{N}$ 是否有 $\mathfrak{D} = \mathfrak{D}_n$ 需要

$$\begin{aligned} &O(h' \log h' \log q + |\mathcal{N}| h' \log q) \subseteq O(d^2 \operatorname{mpe}(q-1) \cdot \ell \operatorname{mpe}(q-1) \cdot \log q) \\ =&O(\ell d^2 \operatorname{mpe}(q-1)^2 \log q) \end{aligned}$$

单个 t 的位运算，因此

$$O(\ell H_i \cdot \ell d^2 \operatorname{mpe}(q-1)^2 \text{对数 } q) \subseteq O(\ell^2 d^3 \operatorname{mpe}(q-1)^3 \text{对数 } q) \subseteq O(\ell d^4 \operatorname{mpe}(q-1)^3 \text{对数 } q)$$

对所有 t 和 h 的位运算。最后这个 O-表达式支配了本复杂度分析中给出的所有其他位操作成本，但计算 t 的成本以及

$$O(\ell^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q) \subseteq O(\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q)$$

计算描述 $\mathfrak{D}$的总成本 $O(\ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)$ 除外。因此，这些计算的位操作成本为

$$O(\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q + \ell d^4 \operatorname{mpe}(q-1)^3 \log q + \ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)$$

对于给定的 (i,ℓ) 和 h。利用 $\sum_{(i,\ell)\in\overline{\mathcal{L}}} \ell = d$，计算 f 的 II 型树寄存器的总位操作成本为

$$\begin{aligned} &O(d \log^{1+o(1)} q + d^2 \log^2 d \\ &+ \sum_{(i,\ell)\in\overline{\mathcal{L}}} (\ell d \operatorname{mpe}(q-1) \log^{1+o(1)} q + \ell d^4 \operatorname{mpe}(q-1)^3 \log q + \ell^2 d^3 \log^2 d \operatorname{mpe}(q-1)^3)) \\ &\subseteq O(d^2 \operatorname{mpe}(q-1) \log^{1+o(1)} q + d^5 \operatorname{mpe}(q-1)^3 \log q + d^5 \log^2 d \operatorname{mpe}(q-1)^3), \end{aligned}$$

正如所断言的那样。“特别是”这一说法成立，是因为 $\operatorname{mpe}(q-1) \in O(\log q)$。 □

到目前为止，所有内容的查询复杂度都有一个关于 q的上界，且关于 $\log q$和 d 是多项式的。量 $\tau(q-1)$ 通常关于 $\log q$是超多项式的，它通过以下辅助结果引入。

命题 5.3.2.3.设 m 为一个正整数，且设 $A : x \mapsto ax+b$ 为 $\mathbb{Z}/m\mathbb{Z}$ 的一个仿射映射。$A_{|\operatorname{per}(A)}$ 的循环类型，记作 $\operatorname{CT}(A_{|\operatorname{per}(A)})$，可以在 m-有界查询复杂度内计算

$$(\tau(m) \log^{2+o(1)} m + \tau(m)^2 \log m, 0, \log m, 1, 0).$$

证明。利用一次 m 有界的 mord 查询（即 m 有界查询复杂度 (log m, 0, 0, 1, 0)），我们对 m 进行因数分解，并计算所有满足 $p \nmid a$ 的素数 $p \mid m$ 的 ord$_{p^{\nu_p(m)}}(a)$。令 $m'' := \prod_{p|\gcd(a,m)} p^{\nu_p(m)}$，并令 $m' := m/m''$（实际上我们无需计算它），我们观察到如下结论。由于 $A \bmod m''$ 有唯一的周期点（见引理 2.1.14），我们得到 CT($A_{|\mathrm{per}(A)}$) = CT($A \bmod m'$)，因此计算后者即可。这使我们可以不失一般性地假设 A是 $\mathbb{Z}/m\mathbb{Z}$的一个置换。对于每个素数 $p \mid m$，我们令 $A_{(p)} := A \bmod p^{\nu_p(m)}$。由于$A$ 由其系数 a 和 b 给出，计算 $A_{(p)}$ 所需的开销仅为每个 p 执行 $O(\log^{1+o(1)} m)$ 次位操作，以求出 a 和 b 除以 $p^{\nu_p(m)}$所得的余数。因此，计算所有约化 $A_{(p)}$ 需要 $O(\log^{2+o(1)} m)$ 次位操作。

CT($A_{(p)}$) 的公式见 [13, 表 3 和 4], ，并且由于我们通过初始的 mord 查询已知所有 $p \mid m$ 的 ord$_{p^{\nu_p(m)}}(a)$，这些公式使我们能够在 m-有界查询复杂度内，针对给定的 p 计算 CT($A_{(p)}$)

$$(\log^{2+o(1)}(p^{\nu_p(m)}) + \nu_p(m)\log^{1+o(1)}(p^{\nu_p(m)}), 0, 1, 0, 0).$$

事实上，仔细查看这些公式可知，最初我们需要对 ord$_{p^{\nu_p(m)}}(a)$ 进行因数分解并计算一次幂，再结合较简单的算术运算（查询复杂度为 $p^{\nu_p(m)}$-有界（$\log^{2+o(1)}(p^{\nu_p(m)})$, 0, 1, 0, 0 ））， 随后执行循环 $O(\nu_p(m))$ 次，每次循环包含 $O(1)$ 次基本算术运算，每次运算耗费 $O(\log^{1+o(1)}(p^{\nu_p(m)}))$ 次位操作。对于所有 p，计算 CT($A_{(p)}$) 的查询复杂度为 m-有界（$\log^{2+o(1)} m$, 0, $\log m$, 0, 0）。我们还注意到，每个 CT($A_{(p)}$) 都是一个单项式，至多包含 $\nu_p(m)$ 个 $+1$ 因数分解项；并且我们的计算过程允许将 CT($A_{(p)}$) 连同完全因数分解的所有周期长度一并存储。

现在，我们可以通过公式 CT(A) $= \ast_{p|m}$CT($A_{(p)}$) 计算 CT(A)，其中$\ast$ 表示第 182 页及后续页的定义 2.2 中的 Wei-Xu 乘积]。具体做法是遍历由以下方式构成的 $O(\prod_{p|m}(\nu_p(m) + 1)) \subseteq O(\tau(m))$ 个元组：从每个 CT($A_{(p)}$) 的因式分解中各选取一个变量幂，并根据引理 2.3(b) 中的 [77,公式 (2.9)]计算这些变量幂（其本身也是一个变量幂）的 Wei-Xu 乘积。为此，我们需要计算所涉周期长度的最小公倍数；由于这些周期长度已经完全分解，这需要 $O(\log^{2+o(1)} m)$ 次位运算（另请参见推论 5.3.1.2 证明中关于 l 的计算的段落）。随后还需要 $O(\log m)$ 次整数乘法和除法来计算指数；这些运算合计也需要 $O(\log^{2+o(1)} m)$ 次位运算。总之，计算所有相关变量幂的 Wei-Xu 乘积需要 $O(\tau(m)\log^{2+o(1)} m)$ 次位运算。完成后，我们需要计算这些变量幂的乘积，也就是说，需要进行 $O(\tau(m))$ 次运算：将一个首一单项式与一个这样的变量幂相乘，其中首一单项式含有 $O(\tau(m))$ 个不同的变量幂因数，每个因数的指标和指数都属于 $\{1, 2, \ldots, m\}$。每次这样的乘法需要 $O(\tau(m)\log m)$ 次位运算，因此这些计算的总体复杂度（其结果为 CT(A)）为 $O(\tau(m)^2 \log m)$。

□

注记 5.3.2.4. 根据命题 5.3.2.3 的证明，A 上的仿射映射 $\mathbb{Z}/m\mathbb{Z}$ 的循环类型是至多 $\tau(m)$ 个变量幂的乘积，因此 A 至多有 $\tau(m)$ 个不同的周期长度。

关于 $\mathbb{Z}/m\mathbb{Z}$ 的仿射映射的最大不同周期长度数的下界，我们固定一个正整数 K 和素数$2 < p_1 < p_2 < \cdots < p_K$，使得$\gcd(p_j, p_{j'} - 1) = 1$对于 $1 \le j < j' \le K$成立（根据狄利克雷关于算术级数中素数的定理，对于每个 K 都存在这样的素数，见[7, 第 7]章）。对于可变的正整数 v_1、v_2、.....、v_K，我们令 $m := p_1^{v_1} \cdots p_K^{v_K}$，并考虑一个自同构 $A: x \mapsto ax$ of $\mathbb{Z}/m\mathbb{Z}$，使得对于每个 j， a 是模$p_j^{v_j}$ 的原根（由于中国剩余定理，可以这样选择 a ）。根据 [13, 表 3], ， A 模$p_j^{v_j}$ 的周期长度恰好是形如$(p_j - 1)p_j^{v_j'}$的数，其中 $v_j' \in \{0, 1, \ldots, v_j - 1\}$。因此， A 的周期长度恰好是形如 $\prod_{j=1}^{K}(p_j - 1) \cdot \prod_{j=1}^{K} p_j^{v_j'}$的数，从而$A$具有 $v_1 \cdots v_K$个不同的周期长度。我们注意到，当 $\min\{v_1, \ldots, v_K\} \to \infty$时，这一周期长度计数渐近等价于 $(v_1 + 1) \cdots (v_K + 1) = \tau(m)$。此外，我们注意到 $\log m = v_1 \log p_1 + \cdots + v_K \log p_K \le (v_1 + \cdots + v_K) \log p_K$。现在，假设 $v_1 = v_2 = \cdots = v_K =: v \to \infty$。那么， A的不同周期长度数为

$$v^K = \left(\frac{\log p_K \cdot K \cdot v}{\log p_K \cdot K}\right)^K \ge \left(\frac{\log m}{\log p_K \cdot K}\right)^K = \frac{\log^K m}{(\log p_K \cdot K)^K} = c(K, p_K) \cdot \log^K m,$$

其中 $c(K, p_K) := (K \log p_K)^{-K}$。由于我们可以针对每个 K 构造这样一类例子，因此 $\mathbb{Z}/m\mathbb{Z}$ 上仿射映射的不同周期长度的最大数量通常不受 $\log m$ 的多项式上界限制。

在继续之前，我们还需要引入一个辅助概念和一个结果。

定义 5.3.2.5. 设 $\vec{\mathfrak{x}} = (\mathfrak{x}_1, \mathfrak{x}_2, \ldots, \mathfrak{x}_n)$ 是一个有限序列。 $\vec{\mathfrak{x}}$ 的一个周期长度是 n 的正因数 m，满足 $\vec{\mathfrak{x}} = \diamond_{t=1}^{n/m}(\mathfrak{x}_1, \mathfrak{x}_2, \ldots, \mathfrak{x}_m)$，其中 $\diamond$ 表示串接（如第 3.3 小节所述）。 $\vec{\mathfrak{x}}$ 的最小正周期长度记作 $\mathrm{minperl}(\vec{\mathfrak{x}})$。

注 5.3.2.6。关于定义 5.3.2.5，我们注意到以下几点。

(1) 数目 $\mathrm{minperl}(\vec{\mathfrak{x}})$ 定义良好，因为至少长度 $n$of $\vec{\mathfrak{x}}$ 是它的一个周期长度。(2) $[\vec{\mathfrak{x}}]$ 中的所有元素（即序列 $\vec{\mathfrak{x}}$ 的循环等价类，参见定义 1.4 之后的第二段）与 $\vec{\mathfrak{x}}$ 具有相同的周期长度，尤其是相同的 minperl 值。我们用 $\mathrm{minperl}([\vec{\mathfrak{x}}])$ 表示这个共同的 minperl 值。我们可以如下界定计算 $\mathrm{minperl}(\vec{\mathfrak{x}})$ 的复杂度。

引理 5.3.2.7。设 $\vec{\mathfrak{x}} \in \{0, 1, \ldots, N-1\}^n$ 是给定的一个长度为 n 的列表，其中包含位长度为 l_{bit} 的非负整数。那么 $\mathrm{minperl}(\vec{\mathfrak{x}})$ 可以通过以下方式计算。

$$O(n \log n \log\log n(\log n + l_{\mathrm{bit}})) \subseteq O(n^{1+o(1)} l_{\mathrm{bit}})$$

位运算。

证明。我们首先确定元组 $\vec{\mathfrak{x}}$ 中条目的数量 n，以及其具有 $\lfloor \log_2 n \rfloor + 1$ 位的二进制表示，这需要 $O(n\log\ n)$ 次位运算。随后，我们以确定性方式进行 n 的因数分解。目前，这项工作的位操作成本纪录为 $O(n^{1/5+o(1)})$，由 Harvey [27]，在 Hittmeir [31] 早先的一项突破性成果基础上取得（我们也可以使用 mdl 查询进行这次因式分解，但这样一来该算法就不完全是经典算法，而且在本分析中，$n^{1/5+o(1)}$ 本来就被其他项所控制）。算法其余部分类似于引理 5.1.6(2) 的证明中所述的确定性过程，用于计算模乘法阶。更具体地说，若 $n = p_1^{v_1} \cdots p_K^{v_K}$ 是所得的 n 的因式分解，那么对于 $j = 1$、2、.....、K，我们执行二分查找，以找到使 $p_j^{v_j'} \prod_{k \neq j} p_k^{v_k}$ 成为 $\vec{\mathfrak{x}}$ 的周期长度的最小 $v_j' \in \mathbb{N}_0$，这意味着 $v_j' = \nu_{p_j}(\text{minperl}(\vec{\mathfrak{x}}))$。对于每个给定的 j，二分查找循环需要 $O(\log v_j) \subseteq O(\log \log n)$ 次迭代才能找到 v_j'，且每次迭代的成本为 $O(n(\log n + l_{\text{bit}}))$ 次位运算。由于 $K \in O(\log n)$，这意味着计算这些数 v_j' 的总位操作成本为 $O(n\log\ n\log \log\ n(\log\ n + l_{\text{bit}}))$；而根据引理 5.1.5(3,6)，计算 $\text{minperl}(\vec{\mathfrak{x}}) = \prod_{j=1}^{K} p_j^{v_j'}$ 需要 $O(\log n \cdot \log^{2+o(1)} n) = O(\log^{3+o(1)} n)$ 次位运算。

□

现在我们给出我们旨在获得的 Γ_f 同构类型的紧凑描述的精确定义。

定义 5.3.2.8. 设 f 是一个函数 $X \to X$，其中 X 是某个有限集，并且令 $\vec{\mathfrak{I}} = (\mathfrak{I}_n)_{n=0,\ 1,\ldots,N}$ 为一个两两不同的有限根树同构类型序列，其中包含所有形如 $\text{Tree}_{\Gamma_f}(x)$、$x \in$ 属于 $\text{per}(f)$ 的同构类型。相对于 $\vec{\mathfrak{I}}$，f 的树项链列表是唯一的集合 $\{([\vec{\mathfrak{n}}_k], l_k, \mathfrak{m}_k) : k = 1, 2, \ldots, N'\}$，满足以下条件。

(1) $[\vec{\mathfrak{n}}_k] = [\mathfrak{n}_{k,1}, \mathfrak{n}_{k,2}, \ldots, \mathfrak{n}_{k,\ell_k'}]$ 是 $\{0, 1, \ldots, N\}$ 中的数字循环序列，使得最小周期长度$([\vec{\mathfrak{n}}_k]) = \ell_k'$。(2) l_k 和 $\mathfrak{m}_k$ 是正整数，且 l_k 是 ℓ_k'的倍数。(3) 在 Γ_f中，恰有 $\mathfrak{m}_k$ 个连通分量，分别对应于有根树同构类型的循环序列 $[\diamond_{t=1}^{l_k/\ell_k'}(\mathfrak{I}_{\mathfrak{n}_{k,1}}, \mathfrak{I}_{\mathfrak{n}_{k,2}}, \ldots, \mathfrak{I}_{\mathfrak{n}_{k,\ell_k'}})]$。(4)对于 Γ_f的每个连通分量，都存在一个 $k \in \{1, 2, \ldots, N'\}$，使得上述连通分量对应于 $[\diamond_{t=1}^{l_k/\ell_k'}(\mathfrak{I}_{\mathfrak{n}_{k,1}}, \mathfrak{I}_{\mathfrak{n}_{k,2}}, \ldots, \mathfrak{I}_{\mathfrak{n}_{k,\ell_k'}})]$。

如果 f 是有限域 d 上的一个指标广义分圆映射，使得 f分别属于特殊类型 I 或 II，并且如果 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ 是 f的I 型或II型树寄存器，那么与 $\mathfrak{R}$相关联的，是根树同构类型的序列（$\mathfrak{I}_n$）$_{n=0,1,\ldots,N}$，其中 $\mathfrak{I}_n$ 具有紧凑描述 $\mathfrak{D}_n$。在该情况下，相对于 $\mathfrak{I}$ 的 f 树项链列表也称为相对于 $\mathfrak{R}$的树项链列表。

注 5.3.2.9。我们对定义 5.3.2.8 作出以下评论。

(1) 要证明相对于 $\vec{\mathfrak{I}}$ 的 f 的树项链列表具有唯一性并不难，但确实需要对所有 k都有 $\text{minperl}([\vec{\mathfrak{n}}_k]) = \ell_k'$。例如，如果没有这一

性质，对于任意长度为 $N+1 \geq 2$的 $\vec{\mathfrak{J}}$，对于某个合适的函数 f，

{ [0,1],4,1 } { [0,1,0,1],4,1 } () 和 () 都会是相对于 $\mathfrak{I}$ 的树项链列表。

(2) 对于 $j=1$，2，令 X_j 为一个有限集，f_j 为一个函数 $X_j \to X_j$。此外，令 $\vec{\mathfrak{J}}$ 为一个有限序列，其中包含两两不同的有限根树同构类型，使得对于 $j=1$，2，每个 $\mathsf{Tree}^{\Gamma_{f_j}}(x)$（其中 $x \in \mathsf{per}(f_j)$）都出现在 $\vec{\mathfrak{J}}$ 中。对于 $j=1$，2，令 $\vec{\mathfrak{N}}_j$ 为相对于 $\mathfrak{I}$ 的 f_j 唯一的树项链列表。不难证明，$\mathfrak{N}_1 = \mathfrak{N}_2$ 当且仅当 $\Gamma_{f_1} \cong \Gamma_{f_2}$。

(3) 令 f 为一个指标为 d 的 $\mathbb{F}_q$ 上的广义分圆映射，且属于特殊类型 I 或 II。我们需要为 f 的树项链列表中的元素确定合适的比特串编码，并作出以下约定。根据假设，若 $i \in \{0,1,\ldots,d\}$ 在 $\overline{f}$ 下的周期长度为 ℓ，则对 Γ_f 中任意一个 C_i 的 f- 周期顶点，其所在连通分支的有根树同构类型循环序列的最小周期是 ℓ 的某个因数。特别是，$\vec{\mathfrak{N}}$ 所述最小周期总是不大于 d。此外，我们假设 $\vec{\mathfrak{I}}$ 源自递归树描述列表 $\mathfrak{D}$，该列表属于 f 的 I 型或 II 型 $\vec{\mathfrak{N}}$ 树寄存器 $\mathfrak{R}$。在 $\mathfrak{D}$ 中，表示来自 $\{0,1,\ldots,N\}$ 的数时，比特长度相同（见定义 5.3.2.1 后的说明）；我们将该位长度记为 l_{bit}，并注意到

$$l_{\mathrm{bit}} = \begin{cases} \lfloor \log_2 d \rfloor + 1 \in O(\log d), & \text{if } \mathfrak{R} \text{ has type I,} \\ \lfloor \log_2 (d^2 \lfloor \log_2 q \rfloor + d) \rfloor + 1 \in O(\log d + \log\log q), & \text{if } \mathfrak{R} \text{ has type II.} \end{cases}$$

如上所述的循环序列 $[\vec{\mathfrak{n}}] = [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{\ell'}]$ 是相对于 $\mathfrak{R}$的树项链列表中某个元素的第一项；我们假设关联的有序序列 $\vec{\mathfrak{n}} = (\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{\ell'})$ 在其循环等价类 $[\vec{\mathfrak{n}}]$ 的所有有序序列中字典序最小。我们按如下方式编码 $[\vec{\mathfrak{n}}]$。我们取有序序列 $\vec{\mathfrak{n}}$，并用条目 -1（一个虚拟值）将其填充，直到它包含 d 个条目。然后，我们输出一个比特字符串，该字符串是此扩展序列中各条目编码的串联。我们使用 $l_{\mathrm{bit}}+1$ 个比特表示每个条目，其中除 -1 以外的条目由一个辅助位 1 表示，后面跟着该条目的长度为 l_{bit} 的二进制位表示，该表示来自 $\mathfrak{R}$。另一方面，条目 -1 由一串 $l_{\mathrm{bit}}+1$ 个零表示。例如，如果 $l_{\mathrm{bit}}=3$ 和 $d=5$，那么对应树项链列表中 $[6,3,4] = [3,4,6]$ 的比特字符串编码为

10111100111000000000.

另一方面，f 的任意树项链列表中元素的第二项和第三项都是不超过 q 的正整数，我们直接使用它们标准的二进制表示，并以包含 $\lfloor \log_2 q \rfloor + 1$ 位来表示它们；这些表示可以直接与 $[\vec{\mathfrak{n}}]$ 的比特串编码连接起来。按照这些约定，给定的 f 的树项链列表中的所有元素都是具有相同位长度的比特串，即

$$d(l_{\mathrm{bit}}+1) + 2(\lfloor \log_2 q \rfloor + 1) \in \begin{cases} O(d\log d + \log q), & \text{if } \mathfrak{R} \text{ has type I,} \\ O(d\log d + d\log\log q + \log q), & \text{if } \mathfrak{R} \text{ has type II,} \end{cases}$$

这使我们能够使用引理5.1.5(10)中的排序算法按字典序对它们进行排序。此外，这些比特串的字典序对应于相关（抽象）树项链列表中元素的“自然”字典序（第一分量采用循环序列的按字典序最小代表元之间的字典序，第二和第三分量采用整数的通常线性序）。应当注意的是，我们的方法涉及一些填充，而通过使用 [5, 第89页及后续页面上的算法 3.2], 即可避免这一点；该算法是一种更一般的字典序排序算法，不要求输入中的比特串具有相同长度。然而，就我们所推导出的复杂度界的 O-类而言，这并没有区别。

注 5.3.2.9(2) 保证，树项链列表是对函数图的有向图同构类型的单射描述，但它们未必总是紧凑的。实际上，其元素数量与所述函数图中连通分量的不同同构类型的数量一样多；而对于固定指标 d 的 $\mathbb{F}_q$ 上广义分圆映射 f 的函数图 Γ_f，这类连通分量的最大数量至少等于 $\mathbb{Z}/s\mathbb{Z}$ 的仿射置换所能达到的不同周期长度的最大数量。后一个数量原则上可能是关于 $s=(q-1)/d$ 的超多项式级（因而当 d 固定时，也是关于 q 的超多项式级），见注记 5.3.2.4。不过我们注意到，注记 5.3.2.4 中考虑的模数具有特殊形式；如果还要求所有构造出的模数都必须是某个满足 $d \mid q-1$ 的素数幂 q 所对应的形式 $(q-1)/d$，其中 $d \in \mathbb{N}^+$ 固定，那么注记 5.3.2.4 中的构造是否仍可实现尚不清楚。此外，根据命题 5.3.1.4，只要愿意排除一小部分渐近正比例的素数幂，那么 $\tau(q-1)$（这是 $\mathbb{Z}/s\mathbb{Z}$ 上仿射映射的不同周期长度数量的一个上界，见命题 5.3.2.3 的证明）就是 $\log q$ 的多项式级。即便对于这样的素数幂 q，Γ_f 本身的连通分量的不同同构类型数量仍可能是 $\log q$ 的超多项式级。我们将寻找一个能证实这一点的具体无限例类的问题留作开放问题；另见问题 6.3.3 和 6.3.4。

尽管如此，如果 f 属于特殊类型 I 或 II，那么以下关键结果表明，可以计算出 f 的树寄存器 $\mathfrak{R}$，随后在相对于 $\mathfrak{R}$ 的 f 上，以关于 $\log q$、d 和 $\tau(q-1)$ 呈多项式级的 q有界查询复杂度，计算并输出树项链列表。

定理 5.3.2.10. 设 f 是 $\mathbb{F}_q$上的指标为 d 的广义分圆映射。
(1) 假设 f 属于特殊类型 I。那么，可以在查询复杂度以 q为界的条件下计算

$$(d^3\log^2 d + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q,\\ d, d\log q, d, 0),$$

或其拉斯维加斯双重复杂度有界为 a q

$$(d^3\log^2 d + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q\\ + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

一个 I 型树寄存器 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ \ldots,\ N}$ 对于 f 且满足 $N \in O(d)$，以及相对于 $\mathfrak{R}$的 f 的树项链列表。

(2) 我们假设 f 属于特殊类型 II。然后可以在一个 q-有界查询复杂度内计算

$$(d^5 \log^2 d \operatorname{mpe}(q-1) + (d^5 \operatorname{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^2 \operatorname{mpe}(q-1)\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

或一个 q有界的拉斯维加斯双重复杂度的

$$(d^5 \log^2 d \operatorname{mpe}(q-1) + (d^5 \operatorname{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^2 \operatorname{mpe}(q-1)\log^{1+o(1)} q + d\tau(q-1)\log^{2+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

一个 II 型树寄存器 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$，属于具有 $N \in O(d^2 \operatorname{mpe}(q-1))$ 的 f，以及相对于 $\mathfrak{R}$ 的 f 树项链列表。

证明。我们同时证明这两个陈述，用“情形 I”和“情形 II”分别指陈述（1）和陈述（2）中描述的情形。首先，我们计算 $\overline{f}$、仿射映射 A_i，以及一个树寄存器 $\mathfrak{R}$，使其具有所需类型，且满足所断言的 N 上界，并具有 q 有界查询复杂度

- $(d^3 \log^2 d + d\log^{1+o(1)} q, d, 0, 0, 0)$ in case I,
- $(d^2 \operatorname{mpe}(q-1)\log^{1+o(1)} q + d^5 \operatorname{mpe}(q-1)^3 \log q + d^5 \log^2 d \operatorname{mpe}(q-1)^3, d, 0, 0, 0)$ in case II

根据命题 5.1.8 和引理5.3.2.2(2,4)。然后，我们计算一个CRL列表 $\overline{\mathcal{L}}$，关于 $\overline{f}$，以及 $\overline{f}$的循环，依据第5.2.1小节开头的论证，耗用 $O(d^2 \log^2 d)$次位运算。对于每个 $(i, \ell) \in \overline{\mathcal{L}}$，令$(i_0, i_1, \ldots, i_{\ell-1})$ 与 $i = i_0$ 表示先前确定的 i 的 $\overline{f}$-循环，我们如下计算相对于 $\mathfrak{R}$ 的树项链列表 $\mathfrak{N}_i$，其针对 f 到 $\bigcup_{t=0}^{\ell-1} C_{i_t}$ 的限制。

若 $i = d$，我们简单地设 $\mathfrak{N}_i := \{([\mathfrak{n}], 1, 1)\}$，其中 $\mathfrak{n} \in \{0, 1, \ldots, N\}$ 是在 $\mathfrak{R}$ 中表示树$\Gamma_f(0_{\mathbb{F}_q})$ 的正-整数。我们注意到，在情形II中，当树$\Gamma_f(0_{\mathbb{F}_q})$是平凡的时，必然有 $\mathfrak{n} = 0$；而在情形 I中，数 $\mathfrak{n}$ 由包含关系 $d \in S_{\mathfrak{n}}$ 唯一确定，因此可以用$O(d \log d)$ 次位运算确定（利用每个 $\mathfrak{n}$ 都由长度为 $O(\log d)$ 的位字符串表示这一事实）。

现在我们假设 $i < d$。然后我们计算 $\mathcal{A}_i = A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$，耗费$O(d\log^{1+o(1)} q)$ 次位运算（见第 5.2.1 小节，第 99 页），以及循环类型 $\operatorname{CT}((\mathcal{A}_i)_{|\operatorname{per}(\mathcal{A}_i)}) = x^{e_{i,1}}{}_1 x^{e_{i,2}}{}_2 \cdots x^{e_{i,s}}{}_s$（其中某些 $e_{i,j}$ 可能为 0），具有q-有界查询复杂度

$$(\tau(q-1)\log^{2+o(1)} q + \tau(q-1)^2 \log q, 0, \log q, 1, 0)$$

由命题 5.3.2.3。此外，我们计算该数列。

$$\vec{\mathfrak{n}}_i = (\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\ell-1}})$$

其中，$\mathfrak{n}_{i_t}$ 由包含关系唯一刻画

$$\begin{cases} i_t \in S_{\mathfrak{n}_{i_t}}, & \text{in case I,} \\ i_t \in S_{\mathfrak{n}_{i_t},\mathrm{per}}, & \text{in case II,} \end{cases}$$

也就是说，$\mathfrak{n}_{i_t}$ 是正整数，它在寄存器 $\mathfrak{R}$中表示 Γ_f 的包含于C_{i_t}中的周期顶点上方唯一的有根树同构类型。确定（并设置其值）$\mathfrak{n}_{i_t}$ 对所有$t = 0$、1、.....、$\ell - 1$所需的位运算次数可按如下方式界定：

- 在情况 I 中，它位于

$$O\left(d \cdot \sum_{n=0}^{N} |S_n| \cdot \log d + d l_{\mathrm{bit}}\right) \subseteq O(d^2 \log d);$$

- 在情况 II 中，它位于

$$O\left(d\left(N + \sum_{n=0}^{N} |S_{n,\mathrm{per}}|\right) \log d + d l_{\mathrm{bit}}\right) \subseteq O(d^3 \log d \,\mathrm{mpe}(q-1) + d \log\log q).$$

接下来，我们将 $\vec{\mathfrak{n}}_i$ 覆盖为同一循环等价类中唯一的字典序最小数字序列。我们可以通过将 $O(\ell) \subseteq O(d)$ 个 $\vec{\mathfrak{n}}_i$ 的循环移位写出，然后按照引理5.1.5(10)中的方式按字典序排列，并取排序列表中的第一个序列来完成此操作。此过程需要

$$O(d \log d \cdot d l_{\mathrm{bit}}) = O(d^2 \log d l_{\mathrm{bit}}) \subseteq \begin{cases} O(d^2 \log^2 d), & \text{in case I,} \\ O(d^2 \log^2 d + d^2 \log d \log\log q), & \text{in case II} \end{cases}$$

位运算。接下来，我们计算 minperl($\vec{\mathfrak{n}}_i$)，根据引理 5.3.2.7，这需要进行以下数量的位运算：

$$\begin{aligned} & O(d \log d \log\log d \cdot (\log d + l_{\mathrm{bit}})) \\ \subseteq & \begin{cases} O(d \log^2 d \log\log d), & \text{in case I,} \\ O(d \log d \log\log d (\log d + \log\log q)), & \text{in case II.} \end{cases} \end{aligned}$$

我们观察到，以下是 $\mathfrak{N}_i$ 的一个有效选择：

$$\mathfrak{N}_i := \{([\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\mathrm{minperl}(\vec{\mathfrak{n}}_i)-1}}], \ell \cdot l', e_{i,l'}) : l' \in \{1, 2, \ldots, s\}, e_{i,l'} > 0\}.$$

为了进一步处理，与其将 $\mathfrak{N}_i$ 原样存储为此列表，不如分别存储 $\vec{\mathfrak{n}}'_i$:
$= [\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\mathrm{minperl}(\vec{\mathfrak{n}}_i)-1}}]$和列表 $\mathfrak{N}'_i := \{(\ell \cdot l', e_{i,l'}) : \ l' \in \{1, 2, \ldots, s\}, e_{i,l'} > 0\}$，这样占用

$$\begin{aligned} & O(\mathrm{minperl}(\vec{\mathfrak{n}}_i) l_{\mathrm{bit}} + \tau(q-1) \log^{1+o(1)} q) \\ \subseteq & \begin{cases} O(d \log d + \tau(q-1) \log^{1+o(1)} q), & \text{in case I,} \\ O(d(\log d + \log\log q) + \tau(q-1) \log^{1+o(1)} q), & \text{in case II} \end{cases} \end{aligned}$$

用于执行乘法 $\ell \cdot l'$ 和复制数据的位运算。这就完成了我们对如何计算 $\mathfrak{N}_i$ 的分析，总体而言，其查询复杂度为 q-有界

$$(\tau(q-1)\log^{2+o(1)} q + (d+\tau(q-1))\log^{1+o(1)} q + \tau(q-1)^2 \log q + E(d,q), 0, \log q, 1, 0)$$

其中

$$\begin{aligned} E(d,q) &:= Nd\log d + d^2 \log dl_{\mathrm{bit}} \\ &\in \begin{cases} O(d^2 \log^2 d), & \text{in case I,} \\ O(d^3 \log d\, \mathrm{mpe}(q-1) + d^2 \log^2 d + d^2 \log d \log\log q), & \text{in case II} \end{cases} \end{aligned}$$

对每个给定的 i，以及将所有 i 合在一起时，该复杂度的（逐分量） d 次倍。

最后，我们需要计算相对于 $\mathfrak{R}$ 的 f 的实际树项链列表 $\mathfrak{N}$。首先令 $\mathfrak{M} := \mathfrak{N} := \emptyset$，并令 $\vec{\mathfrak{n}}' := \{(\vec{\mathfrak{n}}'_i, i, \ell) : (i,\ell) \in \overline{\mathcal{L}}\}$（其中 $\vec{\mathfrak{n}}'_d := [\mathfrak{n}]$ 是 $\mathfrak{N}_d$ 中唯一三元组的第一项），然后按字典序排序 $\vec{\mathfrak{n}}'$。总计需要 $O(d^2 \log dl_{\mathrm{bit}})$ 次位操作。在下文所述的整个过程中，$\mathfrak{N}$ 是将要输出的树项链列表的一个（按字典序排序的）初始段，而 $\mathfrak{M}$ 是从多重集 $\{\vec{\mathfrak{n}}'_i : (i,\ell) \in \overline{\mathcal{L}}\}$ 中删除重复项后所得列表的一个初始段。特别地，$\mathfrak{M}$ 有 $O(d)$ 个元素，每个元素都是一个长度属于 $O(d)$ 的循环序列，其中每一项的位长度属于 $O(l_{\mathrm{bit}})$，且该序列由其字典序最小代表元给出。我们注意到，这样的两个循环序列相等，当且仅当它们的代表元相等，因此验证这种相等性需要 $O(dl_{\mathrm{bit}})$ 次位操作。

现在，我们遍历三元组 $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$，并对其中每一个执行以下操作。首先，我们检查是否$\vec{\mathfrak{n}}'_i \in \mathfrak{M}$，这需要进行 $O(d^2 l_{\mathrm{bit}})$ 次位操作。如果是，则跳到 $\vec{\mathfrak{n}}'$ 中的下一个三元组；否则，执行如下操作。我们将 $\vec{\mathfrak{n}}'_i$ 作为新元素添加到 $\mathfrak{M}$中（复制需要进行 $O(dl_{\mathrm{bit}})$ 次位操作），并设置 $\mathfrak{V} := \mathfrak{N}'_i$，这需要进行 $O(\tau(q-1)\log q)$ 次位操作。

然后，我们遍历 $(\vec{\mathfrak{n}}'_i, i, \ell)$ 之后的元素 $(\vec{\mathfrak{n}}'_j, j, \ell') \in \vec{\mathfrak{n}}'$，并对其中每一个执行以下操作。我们检查是否 $\vec{\mathfrak{n}}'_j = \vec{\mathfrak{n}}'_i$，这需要进行 $O(dl_{\mathrm{bit}})$ 次位操作。如果不是，则跳到 $(\vec{\mathfrak{n}}'_j, j, \ell')$ 的下一个值；否则，执行如下操作。我们遍历 $\mathfrak{N}'_j$ 的元素 (l,k)，并对其中每一个检查是否 l 作为 $\mathfrak{V}$ 中某个二元组 (l,k') 的第一个条目出现；每次此类检查需要进行$O(|\mathfrak{V}|\log q) \subseteq O(d\tau(q-1)\log q)$ 次位操作。如果是，则将 $\mathfrak{V}$ 中形如 (l,k') 的唯一元素替换为 $(l, k+k')$；否则，将 (l,k) 作为新元素添加到 $\mathfrak{V}$ 中。

总体而言，所描述的遍历 $\mathfrak{N}'_j$ 中元素的循环需要

$$O(|\mathfrak{N}'_j| \cdot d\tau(q-1)\log q) \subseteq O(d\tau(q-1)^2 \log q)$$

位运算，因此，对 $(\vec{\mathfrak{n}}'_j, j, \ell')$ 的循环需要 $O(d^2 l_{\mathrm{bit}} + d^2\tau(q-1)^2 \log q)$ 次位运算。该循环结束后，我们通过对每个 $(l,k) \in \mathfrak{V}$，将三元组 $([\vec{\mathfrak{n}}'_i], l, k)$ 作为新元素添加到 $\mathfrak{N}$ 中，从而完成对 $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$的循环——这一复制过程需要

$$O(|\mathfrak{V}| \cdot (dl_{\mathrm{bit}} + \log q)) \subseteq O(d\tau(q-1)(dl_{\mathrm{bit}} + \log q))$$

次位运算。

在对 $(\vec{\mathfrak{n}}'_i、\ i、\ \ell) \in \vec{\mathfrak{n}}'$ 的循环结束时，变量 $\mathfrak{N}$ 已取得其所需的值，而该循环的总体位操作成本为

$$O(d \cdot (d^2 l_{\mathrm{bit}} + d^2\tau(q-1)l_{\mathrm{bit}} + d^2\tau(q-1)^2 \log q)) = O(d^3\tau(q-1)^2 \log q)$$

□

最后，我们讨论有向图同构问题的复杂度。设 f_1和 f_2 是 $\mathbb{F}_q$的广义分圆映射，且各自属于特殊类型 I 或 II 之一。需要注意的是， f_1 和 f_2 的指标不必相同，它们也不一定都属于相同的特殊类型。为了判断 $\Gamma_{f_1} \cong \Gamma_{f_2}$，我们希望将为 f_1 计算出的树项链列表与为 f_2计算出的树项链列表进行比较。为此，这些树项链列表必须“同步”，使得每个 n 在两种情况下都表示相同的有根树同构类型 $\mathfrak{I}_n$ 。下面给出一个精确定义。

定义 5.3.2.11。设 $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ 和 $\vec{\mathfrak{D}}' = (\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$ 是递归树描述列表，其关联的有根树同构类型序列为 $(\mathfrak{I}_n)_{n=0,1,\ldots,N}$ 和 $(\mathfrak{I}'_n)_{n=0,1,\ldots,N'}$。 $\vec{\mathfrak{D}}$ 和 $\vec{\mathfrak{D}}'$ 的同步是满足以下条件的一对 $(\vec{\mathfrak{D}}^+, \mathfrak{i})$。

(1) $\vec{\mathfrak{D}}^+ = (\mathfrak{D}^+_n)_{n=0,1,\ldots,N^+}$ 是一个递归树描述列表，其中 $\vec{\mathfrak{D}}$ 是初始段（特别是，$N \le N^+$）。我们将与 $\vec{\mathfrak{D}}^+$相关联的唯一有根树同构类型序列记为$(\mathfrak{I}^+_n)_{n=0,1,\ldots,N^+}$。

(2) $\mathfrak{i}$是一个函数 $\{0,1,\ldots,N'\} \to \{0,1,\ldots,N^+\}$，满足 $\{N+1, N+2, \ldots, N^+\} \subseteq \mathrm{im}(\mathfrak{i})$，使得对于每个 $n \in \{0,1,\ldots,N'\}$，都有$\mathfrak{I}'_n \cong \mathfrak{I}^+_{\mathfrak{i}\ (n)}$。

在实现中，我们假设每个描述 $\mathfrak{D}_n$ 或 $\mathfrak{D}'_n$ 都按其元素的首项递增排序。我们还假设，对于数字 n的二进制表示， $\vec{\mathfrak{D}}$ 和 $\vec{\mathfrak{D}}'$ 各自使用统一的位长度，分别记作 l_{bit} 和 l'_{bit}。由于 $N^+ \le N + N'$，我们在同步 $\vec{\mathfrak{D}}^+$中为数字 n 使用 $\max\{l_{\mathrm{bit}}, l'_{\mathrm{bit}}\} + 1 \in O(l_{\mathrm{bit}} + l'_{\mathrm{bit}})$ 位。我们注意到，同步的定义具有不对称性，也就是说，$\vec{\mathfrak{D}}$和 $\vec{\mathfrak{D}}'$ 的一个同步不一定也是 $\vec{\mathfrak{D}}'$ 和 $\vec{\mathfrak{D}}$的一个同步。从复杂度来看，以下引理表明，采用 $N' \le N$ 会稍微有利一些。

引理 5.3.2.12. 设 $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,1,\ldots,N}$ 和 $\vec{\mathfrak{D}}' = (\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$ 为递归树描述列表，使得对于所有 n，

- $\mathfrak{D}_n$ 或 $\mathfrak{D}'_n$ 中每个元素的第二项都用长度至多为 m的比特串表示（该数量不依赖于 n）；

  $\mathfrak{D}_n$中每个元素的第一项，以及 $\mathfrak{D}'_n$中每个元素的第一项，分别用长度恰好为 l比特和 l'比特的比特串表示，并且
- 在给定描述 $\mathfrak{D}_n$（分别为 $\mathfrak{D}'_n$）中，该描述元素的第二个条目具有相同的位长度（因此，$\mathfrak{D}_n$（分别是 $\mathfrak{D}'_n$）中的所有配对都具有相同的位长度，该长度属于 $O(l$位 $+ m)$（分别属于 $O(l'$ 位 $+ m$ ））。

计算 $\vec{\mathfrak{D}}$和 $\vec{\mathfrak{D}}'$的同步需要 $O((NN'\min\{N,N'\}+(\max\{N,N'\})^2+(N')^2 l'_{\mathrm{bit}})(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$ 次位运算。

证明。令 $\mathfrak{I}_n$分别为由$\mathfrak{D}_n$描述的有根树同构类型， $\mathfrak{I}'_n$为由 $\mathfrak{D}'_n$描述的有根树同构类型。为了计算同步，我们对 $n=0$、1、.....、N' 进行循环。在过程中的每一个给定点，有根树同构类型 $\mathfrak{I}_k^+$ 的描述 $\mathfrak{D}_k^+$ 都为所有 $k\in\mathcal{N}$定义； $k\in\mathcal{N}$ 是$\mathbb{N}_0$的一个初始段，起初为 $\{0,1,\ldots,N\}$，其中对于每个 $k\in\{0,1,\ldots,N\}$ 都有 $\mathfrak{D}_k^+ := \mathfrak{D}_k$，并且最终将为 $\{0,1,\ldots,N^+\}$。我们注意到，这需要

$$O(N^2(l_{\mathrm{bit}}+\log m))\subseteq O((\max\{N,N'\})^2(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$$

位运算（花费在复制上）以设置集合 $\mathcal{N}$ 和 $\mathfrak{D}_k^+$，用于将 $k\in\{0,1,\ldots,N\}$ 向上。算法中的任意时刻，描述 $\mathfrak{D}_k^+$ 形成一个递归树描述列表，记作 $\vec{\mathfrak{D}}^+$ （最终将具有所需的值）。我们还不断更新 $\mathfrak{i}$ 的值： $\{0,1,\ldots,n\}\to\mathcal{N}$，它最初是空函数 $\emptyset$。

对于 $n=0$，其中 $\mathfrak{D}'_n=\emptyset$ 及其关联的有根树是平凡的，我们简单地设置$\mathfrak{i}(0):=0$，而不更新 $\mathcal{N}$。现在假设 $n\geq 1$。基于 $\mathfrak{D}'_n$，我们通过将 $\mathfrak{D}'_n$ 中给定的每一对的第一个条目 $k<n\leq N'$ 替换为 $\mathfrak{i}(k)$，计算新的有根树描述 $\mathfrak{D}$。由于 $\mathfrak{D}'_n$ 至多包含 $n+1\in O(n)$ 个不同的对，并且此处处理的是位长度为 $O(l_{\mathrm{bit}}+l'_{\mathrm{bit}})$ 的非负整数，因此，将 $\mathfrak{D}$ 作为未排序列表进行计算，对于给定的 n，总共需要 $O(n\cdot(l_{\mathrm{bit}}+l'_{\mathrm{bit}}))\subseteq O(N'(l_{\mathrm{bit}}+l'_{\mathrm{bit}}))$ 次位运算；对 $\mathfrak{D}$ 进行排序还需要另外 $O(N'l'_{\mathrm{bit}}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$ 次位运算。

一旦 $\mathfrak{D}$ 已按排序形式计算完成，我们需要检查由其相对于 $\vec{\mathfrak{D}}^+$ 所描述的有根树 $\mathfrak{I}'_n$ 是否已经出现在 $\mathfrak{I}_k^+$ 中。如果 $k>N$，那么对于某个 $l\in\{0,1,\ldots,n-1\}$有 $\mathfrak{I}_k^+\cong\mathfrak{I}'_l$，从而 $\mathfrak{I}_k^+\ncong\mathfrak{I}'_n$，因为根据假设， $\mathfrak{I}'_t$ 同构类型两两不同。因此，我们只需要检查$\mathfrak{I}_k^+\cong\mathfrak{I}'_n$ 与 $k\leq N$的同构，这等价于$\mathfrak{I}_k\cong\mathfrak{I}'_n$，进而等价于$\mathfrak{D}_k=\mathfrak{D}$。对于给定的 k，通过线性扫描检查 $\mathfrak{D}_k=\mathfrak{D}$ 需要 $O(\min\{N,N'\}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$次位运算（利用 $\mathfrak{D}_k$ 和 $\mathfrak{D}$ 均已排序），因此可以用 $O(N\min\{N,N'\}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$次位运算检查对于某个（唯一的） $k\in\{0,1,\ldots,N\}$是否$\mathfrak{I}'_n\cong\mathfrak{I}_k^+$。如果是，则将 $\mathfrak{i}(n):=k$，而不更新$\mathcal{N}$；否则，我们用新元素 $n':=\max\mathcal{N}+1$ 扩展 $\mathcal{N}$，并设置 $\mathfrak{D}_{n'}^+:=\mathfrak{D}$以及 $\mathfrak{i}(n):=n'$。

所述循环的总体位操作成本为 $O((NN'\min\{N,N'\}+(N')^2 l'_{\mathrm{bit}})(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$。我们通过输出 $(\vec{\mathfrak{D}}^+,\mathfrak{i})$ 来结束该算法，其中$\vec{\mathfrak{D}}^+:=(\mathfrak{D}_k^+)_{k\in\mathcal{N}}$。这需要

$$\begin{aligned}&O((N+N')\max\{N,N'\}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))\\=&O((\max\{N,N'\})^2(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))\end{aligned}$$

用于复制的位运算。 □

推论 5.3.2.13。设 f_1 和 f_2 是同一个有限域 $\mathbb{F}_q$ 上的广义分圆映射，分别具有指标 d_1 和 d_2，并设 $d:=\max\{d_1,d_2\}$。

(1) 如果每个 f_j 都是特殊类型 I 或 II（不一定都属于同一类型），那么其查询复杂度有界，为 q。

$$((d^6\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^4\,\mathrm{mpe}(q-1)^2\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

或有界于 q 的拉斯维加斯双重复杂度

$$((d^6\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^4\,\mathrm{mpe}(q-1)^2\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

以判定 $\Gamma_{f_1} \cong \Gamma_{f_2}$。

(2) 如果两个 f_j 都是特殊类型 I，那么其查询复杂度有界于 q

$$(d^3\log^2(d) + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

或有界于 q 的拉斯维加斯双重复杂度

$$(d^3\log^2(d) + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q + d\log^{8+o(1)} q, d\log^{4+o(1)} q, d\log^2 q),$$

判断是否 $\Gamma_{f_1} \cong \Gamma_{f_2}$。

证明。我们将陈述（1）中的情形称为“情形（1）”，将陈述（2）中的情形称为“情形（2）”；不得将它们与定理 5.3.2.10 证明中的情形 I 和 II 混淆。首先，对于 $j = 1$、2，我们计算一个合适的包含 $N_j + 1$ 个条目的 f_j 的树寄存器 $\mathfrak{R}_j$，以及与 f_j 相关联的树项链列表 $\mathfrak{N}_j$。根据定理 5.3.2.10，这需要 q 有界查询复杂度

$$(d^5\log^2 d\,\mathrm{mpe}(q-1) + d^3\log d\tau(q-1) + d^3\tau(q-1)\log\log q \\ + (d^5\,\mathrm{mpe}(q-1)^3 + d^3\tau(q-1)^2)\log q + d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

在情况 (1) 中，或

$$(d^3\log^2 d + d^3\log d\tau(q-1) + d^3\tau(q-1)^2\log q + d^2\log^{1+o(1)} q \\ + d\tau(q-1)\log^{2+o(1)} q, d, d\log q, d, 0),$$

在情况（2）中。

随后，我们将 $\mathfrak{R}_j$ 的底层递归树描述列表同步。根据引理 5.3.2.12，取 $m := \log q$，并利用以下事实：两种情况下均有 $l_{\mathrm{bit}} \in O(\log q)$，且

$$\min\{N_1, N_2\} \le \max\{N_1, N_2\} \in \begin{cases} O(d^2\,\mathrm{mpe}(q-1)), & \text{in case (1)}, \\ O(d), & \text{in case (2)}, \end{cases}$$

我们可以看出，这可以通过使用

$$\begin{aligned}&O((d^6\operatorname{mpe}(q-1)^3+d^4\operatorname{mpe}(q-1)^2\log\log q)(\log d+\log\operatorname{mpe}(q-1)+\log q))\\=&O((d^6\operatorname{mpe}(q-1)^3+d^4\operatorname{mpe}(q-1)^2\log\log q)\log q)\end{aligned}$$

在情形 (1) 中需要进行位运算，或在情形 (2) 中需要进行 $O(d^3\log q)$ 次位运算。按照我们对同步中描述的索引 n 的位长度 $\mathfrak{D}_n^+$ 所采用的约定，计算得到的同步使用公共位长度 l^+ 位，在情形 (1) 中该长度为 $O(\log d+\log\log q)$，在情形 (2) 中为 $O(\log d)$。

接下来，基于 $\mathfrak{N}_2$，我们针对 f_2，根据计算得到的同步 $(\vec{\mathfrak{D}}^+,\mathfrak{i})$，通过将 $\mathfrak{N}_2$ 中某个循环序列里作为条目出现的每个数 $\mathfrak{n}$替换为 $\mathfrak{i}(\mathfrak{n})$，再将底层有序序列替换为同一循环等价类中字典序最小的代表元，从而计算修改后的树项链列表 $\mathfrak{N}_2'$。以未排序形式计算 $\mathfrak{N}_2'$ 需要

$$O(|\mathfrak{N}_2|\cdot(dl_{\text{bit}}^+ +\log q)+d\cdot d\log dl_{\text{bit}}^+)\subseteq O(d^2\tau(q-1)\log q+d^2\log d\log q)$$

次位运算；随后我们按字典序对 $\mathfrak{N}_2'$ 进行排序，这需要 $O(d^3\tau(q-1)\log q)$ 次位运算，方法是依次合并对应于相同重写后首项的 $O(d)$ 个片段（参见引理 5.1.5(11)）。

最后，我们需要检查是否 $\mathfrak{N}_2'=\mathfrak{N}_1$。由于 $\mathfrak{N}_2'$ 和 $\mathfrak{N}_1$ 都按字典序排序，因此只需线性扫描即可（需要注意的是，$\mathfrak{N}_1$ 中的索引 n 的位长度可能与$\mathfrak{N}_2'$中的不同，但这当然不是问题）。其位操作成本为

$$O(\min\{|\mathfrak{N}_1|,|\mathfrak{N}_2'|\}\cdot(dl_{\text{bit}}^+ +\log q))\subseteq O(d\tau(q-1)\cdot d\log q)=O(d^2\tau(q-1)\log q).$$

□

正如我们在第 5.2 小节和第5.3.1子小节中所做的那样，我们以本小小节中介绍的所有相关算法的伪代码结束本小小节，并说明每一步的查询复杂度（根据上下文，为q-有界或 m-有界）。我们从引理5.3.2.2(1)中的算法开始，该算法用于判定给定指标 d 广义分圆映射 f of $\mathbb{F}_q$ 是否属于特殊类型 I。

1计算与 f相关的 $\mathbb{Z}/s\mathbb{Z}$ 的仿射映射 $A_i:\ x\mapsto\alpha_i x+\beta_i$。计算成本：($d$ $\log^{1+o(1)}q$, d, 0, 0,0)。2对于每个 $i\in\{0,1,\ldots,d-1\}$，执行以下操作。
计算成本：($d\log^{1+o(1)}q$, 0, 0,0, 0)。2.1 如果 $a_i=0_{\mathbb{F}_q}$，则跳至下一个 i。
计算成本：($\log\ d$,0, 0, 0, 0)。2.2如果最大公约数$(\alpha_i, s)>1$，则输出“假”并停止。计算成本：($\log^{1+o(1)}q$, 0, 0, 0, 0)。

3 输出“true”并停止。

接下来，我们列出引理5.3.2.2(2)中的算法步骤。该算法针对给定的指标 d 广义分圆映射 f，其中 $\mathbb{F}_q$ 是特殊类型I，计算 I 型树寄存器 $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$，其对应于 f 且满足 $N \in O(d)$。

1 Compute $s = (q-1)/d$, the induced function $\overline{f}$ on $\{0, 1, \ldots, d\}$ and the layers Layer_h, for $h \in \{0, 1, \ldots, \overline{H} - 1, \infty\}$, of $\overline{f}$ with respect to iteration.
QC: $(d^2 \log^2 d + d \log^{1+o(1)} q, d, 0, 0, 0)$.

2 Set $\mathcal{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

3 For each $h = 0, 1, \ldots, \overline{H} - 1, \infty$, do the following.
QC: $(d^3 \log^2 d + d \log^{1+o(1)} q, 0, 0, 0, 0)$.

3.1 If $h = 0$ or $\overline{H} = 0$, then do the following.

3.1.1 Add 0 to $\mathcal{N}$ as a new (the first) element, set $\mathfrak{D}_0 := \emptyset$ and $S_0 := \mathrm{Layer}_h$.
QC: $(d \log d, 0, 0, 0, 0)$.

3.1.2 If $\overline{H} = 0$, then output $\mathfrak{R} := ((\mathfrak{D}_0, S_0))$ and halt.
QC: $(d \log d, 0, 0, 0, 0)$.

3.2 Else do the following.

3.2.1 For each $i \in \mathrm{Layer}_h$, do the following.
QC: $(|\mathrm{Layer}_h| d^2 \log^2 d + \delta_{d \in \mathrm{Layer}_h} d \log^{1+o(1)} q, 0, 0, 0, 0)$.

3.2.1.1 Determine the full pre-image set $\overline{f}^{-1}(\{i\})$ and the $\overline{f}$-transient $\overline{f}$-pre-images $j_1, j_2, \ldots, j_K$ of i.
QC: $(d \log d, 0, 0, 0, 0)$ if $h < \infty$; $(d^2 \log^2 d, 0, 0, 0, 0)$ if $h = \infty$.

3.2.1.2 For each $t = 1, 2, \ldots, K$, do the following.
QC: $(d^2 \log^2 d, 0, 0, 0, 0)$.

3.2.1.2.1 Determine the unique number $\overline{n}_{j_t} \in \mathcal{N}$ such that $j_t \in S_{\overline{n}_{j_t}}$.
QC: $(d \log^2 d, 0, 0, 0, 0)$.

3.2.1.3 If $i < d$, then do the following.

3.2.1.3.1 Set

$$\mathfrak{D} := \{(n, m) : n \in \{\overline{n}_{j_t} : 1 \le t \le K\}, \\ m = |\{t \in \{1, \ldots, K\} : \overline{n}_{j_t} = n\}| > 0\}.$$

QC: $(d \log^2 d, 0, 0, 0, 0)$.

3.2.1.4 Else do the following.

3.2.1.4.1 Set

$$\mathfrak{D} := \{(n, sm) : n \in \{\overline{n}_{j_t} : 1 \le t \le K\}, \\ m = |\{t \in \{1, \ldots, K\} : \overline{n}_{j_t} = n\}| > 0\}$$

QC: $(d \log^2 d + d \log^{1+o(1)} q, 0, 0, 0, 0)$.

3.2.1.5 Check whether there is a (unique) $n \in \mathcal{N}$ such that $\mathfrak{D} = \mathfrak{D}_n$, and store this information (the truth value and n).
QC: $(d^2 \log d, 0, 0, 0, 0)$.

3.2.1.6 如果对于某个 $n \in \mathcal{N}$ 有 $\mathfrak{D} = \mathfrak{D}_n$，则执行以下操作。
3.2.1.6.1 合并 $\{i\}$ 和 S_n 有序列表。计算成本：($d\log\ d$, 0, 0, 0, 0)。3.2.1.7 否则执行以下操作。(a) 设置 $n' := \max\mathcal{N} + 1$，并将 n' 作为新元素添加到 $\mathcal{N}$ 中。计算成本：($\log\ d$, 0, 0, 0, 0)。(b) 设置 $\mathfrak{D}_{n'} := \mathfrak{D}$，并初始化 $S_{n'} := \{i\}$。如果 $i < d$，计算成本：($d\log\ d$, 0, 0, 0, 0)；如果 $i = d$，计算成本：($d\log\ q$, 0, 0, 0, 0)。

4 输出 $\mathfrak{R} := ((\mathfrak{D}_n, S_n))_{n=0,\ 1,\ \ldots\ldots\max\mathcal{N}}$并停止。
计算成本：($d^2\log\ d + d\log\ q$, 0, 0, 0, 0)。

接下来，我们给出引理 5.3.2.2(3)中算法的（简单）伪代码，用于检查给定的指标 d 广义分圆映射 f of $\mathbb{F}_q$ 是否属于特殊类型 II。

1 计算诱导函数 $\overline{f}$：$\{0, 1, \ldots, d\} \to \{0, 1, \ldots, d\}$。计算成本：($d\log^{1+o(1)} q, d$, 0, 0, 0)。2 对 $\mathrm{im}(\overline{f})$ 进行排序，并检查其是否等于 $\{0, 1, \ldots, d\}$。计算成本：($d\log^2 d$, 0, 0, 0, 0)。

用于计算给定指标 d 广义分圆映射 f 的 $\mathbb{F}_q$ 特殊类型 II 的II 型树寄存器的、来自引理 5.3.2.2(4) 的算法，其伪代码如下。

1 Compute the induced function $\overline{f}$ and the affine maps $A_i : x \mapsto \alpha_i x + \beta_i$.
QC: $(d\log^{1+o(1)} q, d, 0, 0, 0)$.

2 Compute a CRL-list $\overline{\mathcal{L}}$ of $\overline{f}$ and, in the process, store the cycles of $\overline{f}$.
QC: $(d^2\log^2 d, 0, 0, 0, 0)$.

3 Set $\mathcal{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

4 For each $(i, \ell) \in \overline{\mathcal{L}}$, do the following.
QC: $(d^5\log^2 d\,\mathrm{mpe}(q-1)^3 + d^5\,\mathrm{mpe}(q-1)^3\log q + d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1 For each $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(\ell^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.1 Set $\mathrm{prod}_{\mathrm{upper}} := \alpha_{i_{t-1}}$.
QC: $(\log q, 0, 0, 0, 0)$.

4.1.2 Set $\mathrm{prod}_{\mathrm{lower}} := 1$.
QC: $(1, 0, 0, 0, 0)$.

4.1.3 For each $k = 1, 2, \ldots$ (no upper bound a priori), do the following.
QC: $(\ell\,\mathrm{mpe}(q-1)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.3.1 Compute

$$\mathrm{proc}_{i_t,k} := \frac{\gcd(\mathrm{prod}_{\mathrm{upper}}, s)}{\gcd(\mathrm{prod}_{\mathrm{lower}}, s)}.$$

QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1.3.2 If $\mathrm{proc}_{i_t,k} = 1$, then do the following.
4.1.3.2.1 Set $\mathcal{H}_{i_t} := k - 1$.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$.
4.1.3.2.2 Exit the loop for k, and skip to the next t.
QC: $(1, 0, 0, 0, 0)$.
4.1.3.3 Else do the following.
(a) Set $\mathrm{prod}_{\mathrm{lower}} := \mathrm{prod}_{\mathrm{upper}}$.
QC: $(\log q, 0, 0, 0, 0)$.
(b) Set $\mathrm{prod}_{\mathrm{upper}} := \mathrm{prod}_{\mathrm{upper}} \cdot \alpha_{i_{t-k-1}}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.
4.2 Compute $H_i := \max\{\mathcal{H}_{i_t} : t = 0, 1, \ldots, \ell - 1\}$.
QC: $(\ell(\log d + \log\log q), 0, 0, 0, 0)$.
4.3 For $h = 0, 1, \ldots, H_i$, do the following.
QC: $(\ell^2 d^3 \log^2 d \,\mathrm{mpe}(q-1)^3 + \ell d^4 \,\mathrm{mpe}(q-1)^3 \log q, 0, 0, 0, 0)$.
4.3.1 If $h = 0$, then do the following.
4.3.1.1 If $\mathcal{N} = \emptyset$, then do the following.
4.3.1.1.1 Set $\mathfrak{D}_0 := \emptyset$ and $\mathrm{ht}_0 := 0$.
QC: $(1, 0, 0, 0, 0)$.
4.3.1.1.2 If $H_i > 0$, then set $S_{0,\mathrm{trans}} := \{i_0, i_1, \ldots, i_{\ell-1}\}$, sorted. Otherwise, set $S_{0,\mathrm{trans}} := \emptyset$.
QC: $(\ell \log^2 d, 0, 0, 0, 0)$.
4.3.1.1.3 Set $S_{0,\mathrm{per}} := \{i_t : \mathcal{H}_{i_t} = 0\}$, sorted.
QC: $(\ell \log^2 d, 0, 0, 0, 0)$.
4.3.1.2 Else do the following.
4.3.1.2.1 If $H_i > 0$, then sort $\{i_0, i_1, \ldots, i_{\ell-1}\}$ and merge it with $S_{0,\mathrm{trans}}$.
QC: $(\ell \log^2 d + d \log d, 0, 0, 0, 0)$.
4.3.1.2.2 Create a list of all indices i_t, for $t = 0, 1, \ldots, \ell - 1$, such that $\mathcal{H}_{i_t} = 0$, then sort it and merge it with $S_{0,\mathrm{per}}$.
QC: $(\ell \log^2 d + d \log d, 0, 0, 0, 0)$.
4.3.2 Else do the following.
4.3.2.1 For $t = 0, 1, \ldots, \ell - 1$, do the following.
QC: $(\ell^2 d^2 \log^2 d \,\mathrm{mpe}(q-1)^2 + \ell d^3 \,\mathrm{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.
4.3.2.1.1 If $h < H_i$, then set $h' := h$; otherwise, set $h' := \mathcal{H}_{i_t}$.
QC: $(\log d + \log\log q, 0, 0, 0, 0)$.
4.3.2.1.2 For $k = 0, 1, \ldots, h' - 1$, set

$$\mathfrak{w}_k := \begin{cases} \mathrm{proc}_{i_t,k+1} - \mathrm{proc}_{i_t,k+2}, & \text{if } h = H_i, \text{ or } h < H_i \text{ and } k < h' - 1, \\ \end{cases}$$

$\mathrm{proc}_{i_t,h}$, 如果 $h < H_i$ 且 $k = h' - 1$.计算成本：$(\ell\,\mathrm{mpe}(q-1)\log q$, 0, 0, 0, 0)。4.3.2.1.3 对于 $k = 0$、1、.....、$h' - 1$, 求 $\overline{n}_{i_t,k}$, 唯一的 $n \in \mathcal{N}$使得$i_t \in S_{n,\mathrm{trans}}$ 且 $\mathrm{ht}_n = k$。计算成本：$(\ell d^2 \log^2 d\mathrm{mpe}(q-1)^2 + \ell\,\mathrm{mpe}(q-1)\log\log q$, 0, 0, 0, 0)。

4.3.2.1.4 Set $\mathfrak{D} := \{(\overline{n}_{i_t,k}, \mathfrak{w}_k) : k = 0, 1, \ldots, h' - 1\}$, sorted lexicographically.
QC: $(\ell \operatorname{mpe}(q-1)(\log \ell + \log \operatorname{mpe}(q-1)) \log q, 0, 0, 0, 0)$.

4.3.2.1.5 Check whether $\mathfrak{D} = \mathfrak{D}_n$ some (unique) $n \in \mathcal{N}$, and if so, store this information (the truth value and n).
QC: $(\ell d^2 \operatorname{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.

4.3.2.1.6 If $\mathfrak{D} = \mathfrak{D}_n$ for some $n \in \mathcal{N}$, then do the following.

4.3.2.2.6.1 If $h < H_i$, then merge the sorted lists $\{i_t\}$ and $S_{n,\mathrm{trans}}$. Otherwise, merge the sorted lists $\{i_t\}$ and $S_{n,\mathrm{per}}$.
QC: $(d \log d, 0, 0, 0, 0)$.

4.3.2.1.7 Else do the following.

(1) Set $n' := \max \mathcal{N} + 1$, $\mathfrak{D}_{n'} := \mathfrak{D}$, $\mathrm{ht}_{n'} := h'$, and add n' to $\mathcal{N}$ as a new element.
QC: $(d \operatorname{mpe}(q-1) \log q, 0, 0, 0, 0)$.

(2) Initialize $S_{n',\mathrm{trans}} := \emptyset$ and $S_{n',\mathrm{per}} := \emptyset$.
QC: $(\log d + \log \log q, 0, 0, 0, 0)$

(3) If $h < H_i$, then add i_t to $S_{n',\mathrm{trans}}$ as a new element. Otherwise, add i_t to $S_{n',\mathrm{per}}$ as a new element.
QC: $(\log d + \log \log q, 0, 0, 0, 0)$.

5 Compute $\mathfrak{H} := \max\{H_i : (i, \ell) \in \overline{\mathcal{L}}\}$.
QC: $(d(\log d + \log \log q), 0, 0, 0, 0)$.

6 For $n \in \mathcal{N}$, do the following.
QC: $(d^3 \log d \operatorname{mpe}(q-1) + d^2 \operatorname{mpe}(q-1) \log \log q, 0, 0, 0, 0)$.

6.1 Set $S_n := (\mathrm{ht}_n, S_{n,\mathrm{trans}}, S_{n,\mathrm{per}})$.
QC: $(d \log d + \log \log q, 0, 0, 0, 0)$.

7 Output $\mathfrak{R} := ((\mathfrak{D}_n, S_n))_{n \in \mathcal{N}}$ and halt.
QC: $(d^4 \operatorname{mpe}(q-1)^2 \log q, 0, 0, 0, 0)$.

接下来，我们给出命题5.3.2.3中算法的伪代码，该算法计算给定仿射映射 $A : x \mapsto ax + b$ 的 $\mathbb{Z}/m\mathbb{Z}$ 的循环类型 $\mathsf{CT}(A_{|\mathrm{per}(A)})$。

1 对 m 进行因数分解，并对所有满足 $p \nmid a$ 的素数 $p \mid m$ 计算 $\mathsf{ord}_{p^{\nu_p(m)}}(a)$。计算成本：$(\log m, 0, 1, 0, 0)$。 **2** 对每个满足 $p \nmid a$ 的素数 $p \mid m$，执行以下操作。计算成本：$(\log^{2+o(1)} m, 0, \log m, 0, 0)$。 **2.1** 计算 $A_{(p)} := A \bmod p^{\nu_p(m)}$。计算成本：$(\log^{1+o(1)} m, 0, 0, 0, 0)$。 **2.2** 使用 [13, 表 3 和 4], 计算

$$\mathrm{CT}(A_{(p)}) = x_{\overline{l}_{p,1}}^{e_{p,1}} x_{\overline{l}_{p,2}}^{e_{p,2}} \cdots x_{\overline{l}_{p,K_p}}^{e_{p,K_p}} \ (\text{all } e_{p,j} > 0)$$

所有周期长度均已完全因数分解为 $\overline{l}_{p,j}$。这涉及对 $\mathsf{ord}_{p^{\nu_p(m)}}(a)$ 进行因数分解、一次幂计算，以及 $O(\nu_p(m))$ 次较简单的算术运算。计算成本：$(\log^{2+o(1)} p^{\nu_p(m)} + \nu_p(m) \log^{1+o(1)} p^{\nu_p(m)}, 0, 1, 0, 0)$。

3 对每个 $\vec{j}=(j_p)_{p\mid m,p\nmid a}\in\prod_{p\mid m,p\nmid a}\{1,2,\dots,K_p\}$，执行以下操作。

计算成本：$(\tau(m)\log^{2+o(1)}m,\ 0,\ 0,\ 0,\ 0)$。

3.1 计算变量幂的 Wei-Xu 乘积

$$\mathrm{WX}(\vec{j}):=\divideontimes_{p\mid m,p\nmid a}x_{\bar{l}_{p,j_p}}^{e_{p,j_p}}=x_{\mathrm{lcm}(\bar{l}_{p,j_p}:p\mid m,p\nmid a)}^{\prod_{p\mid m,p\nmid a}(e_{p,j_p}\bar{l}_{p,j_p})/\,\mathrm{lcm}(\bar{l}_{p,j_p}:p\mid m,p\nmid a)}.$$

QC: $(\log^{2+o(1)}m,0,0,0,0)$.

4 计算并输出

$$\mathrm{CT}(A)=\divideontimes_{p\mid m,p\nmid a}\mathrm{CT}(A_{(p)})=\prod_{\vec{j}}\mathrm{WX}(\vec{j}),$$

然后停止。

计算成本：$(\tau(m)^2\log\ m,\ 0,\ 0,\ 0,\ 0)$。

以下是引理 5.3.2.7 中算法的伪代码，用于计算给定 $\vec{\mathfrak{r}}\in\{0,1,\dots,N-1\}^n$ 时的 minperl($\vec{\mathfrak{r}}$)，其中每个 $n\in\{0,1,\dots,N-1\}$ 的位长度为 l_{bit}。

1 计算 n 的二进制表示。计算成本：$(n\log\ n,\ 0,\ 0,\ 0,\ 0)$。2 确定性地对 $n=p_1^{v_1}\cdots p_K^{v_K}$ 进行因数分解。计算成本：$(n^{1/5+o(1)},\ 0,\ 0,\ 0,\ 0)$。3 对每个 $j=1$、2、.....、K，执行以下操作。计算成本：$(n\log\ n\log\log\ n(\log\ n+l_{\mathrm{bit}}))$。3.1 使用二分查找，找出 minperl($\vec{\mathfrak{r}}$) 中的 $v_j'=\nu_{p_j}$，使得 $p_j^v\prod_{k\neq j}p_k^{v_k}$ 是 $\vec{\mathfrak{r}}$ 的周期长度，且 $v\in\{0,1,\dots,v_j\}$ 尽可能小。计算成本：$(n\log\log\ n(\log\ n+l_{\mathrm{bit}}))$。

4 计算并输出

$$\mathrm{minperl}(\vec{\mathfrak{r}})=\prod_{j=1}^{K}p_j^{v_j'},$$

然后停止。

计算成本: $(\log^{3+o(1)}n$,0, 0, 0, 0).

接下来，我们给出定理 5.3.2.10 的伪代码，该定理关注如何为给定指标d 的 $\mathbb{F}_q$ 特殊类型 I 或 II 的广义分圆映射 f 计算树寄存器，以及与之关联的树项链列表。由于这两种情况的处理过程类似，我们只给出一个同时处理两者的算法。

1 检查 f 是否为特殊类型 I，并存储此信息。在此过程中，还要计算并存储 $\overline{f}$ 和仿射映射 A_i，以供后续使用。计算成本：$(d\log^{1+o(1)}q,d,0,0,0)$。

2 If f is *not* of special type I, then check whether f is of special type II and store this information.
QC: $(d\log^2 d, 0, 0, 0, 0)$ because $\overline{f}$ has already been computed.

3 If f is neither of special type I nor II, then output "fail" and halt.
QC: $(1, 0, 0, 0, 0)$.

4 If f is of special type I, then do the following.

4.1 Compute a type-I tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d)$.
QC: $(d^3\log^2 d + d\log^{1+o(1)} q, 0, 0, 0, 0)$ because $\overline{f}$ and the A_i have already been computed.

5 Else do the following.

5.1 Compute a type-II tree register $\mathfrak{R} = ((\mathfrak{D}_n, S_n))_{n=0,1,\ldots,N}$ for f with $N \in O(d^2\,\mathrm{mpe}(q-1))$, where $S_n = (\mathrm{ht}_n, S_{n,\mathrm{trans}}, S_{n,\mathrm{per}})$.
QC: $(d^2\,\mathrm{mpe}(q-1)\log^{1+o(1)} q + d^5\,\mathrm{mpe}(q-1)^3\log q + d^5\log^2 d\,\mathrm{mpe}(q-1)^3, 0, 0, 0, 0)$ because $\overline{f}$ and the A_i have already been computed.

6 Compute a CRL-list $\overline{\mathcal{L}}$ of $\overline{f}$ and, in the process, store the cycles of $\overline{f}$.
QC: $(d^2\log^2 d, 0, 0, 0, 0)$.

7 For each $(i, \ell) \in \overline{\mathcal{L}}$, with associated $\overline{f}$-cycle $(i_0, i_1, \ldots, i_{\ell-1})$, do the following.
QC: $(d\tau(q-1)\log^{2+o(1)} q + d^2\log^{1+o(1)} q + d\tau(q-1)^2\log q + Nd^2\log d + d^3\log dl_{\mathrm{bit}}, 0, d\log q, d, 0)$.

7.1 If $i = d$, then do the following.

7.1.1 If f is of special type I, then do the following.

7.1.1.1 Set $\mathfrak{n}$ to be the unique $n \in \{0, 1, \ldots, N\}$ such that $d \in S_n$.
QC: $(d\log d, 0, 0, 0, 0)$.

7.1.2 Else do the following.

7.1.2.1 Set $\mathfrak{n} := 0$.
QC: $(1, 0, 0, 0, 0)$.

7.1.3 Set $\mathfrak{N}_d := \{([\mathfrak{n}], 1, 1)\}$, $\vec{\mathfrak{n}}'_d := [\mathfrak{n}]$ and $\mathfrak{N}'_d := \{(1, 1)\}$, then skip to the next pair (i, ℓ).
QC: $(\log d, 0, 0, 0, 0)$.

7.2 Else do the following.

7.2.1 Compute $\mathcal{A}_i := A_{i_0}A_{i_1}\cdots A_{i_{\ell-1}}$.
QC: $(d\log^{1+o(1)} q, 0, 0, 0, 0)$.

7.2.2 Compute $\mathrm{CT}((\mathcal{A}_i)_{|\mathrm{per}(\mathcal{A}_i)}) = x_1^{e_{i,1}}x_2^{e_{i,2}}\cdots x_s^{e_{i,s}}$.
QC: $(\tau(q-1)\log^{2+o(1)} q + \tau(q-1)^2\log q, 0, \log q, 1, 0)$.

7.2.3 If f is of special type I, then do the following.

7.2.3.1 For each $n = 0, 1, \ldots, N$, do the following.
QC: $(d^2\log d, 0, 0, 0, 0)$.

7.2.3.1.1 For each $t = 0, 1, \ldots, \ell-1$, do the following.
QC: $(|S_n|d\log d, 0, 0, 0, 0)$.

7.2.3.1.1.1如果 $i_t \in S_n$，则设置$\mathfrak{n}_{i_t} := n$。计算成本：($|S_n| \log d$,0, 0,0,0)。7.2.4 否则执行以下操作。7.2.4.1 对每个 $n = 0, 1, \ldots, N$，执行以下操作。计算成本：($d^3 \log d$mpe$(q-1) + dl_{\text{bit}}$,0, 0, 0, 0)。7.2.4.1.1 对每个 $t = 0, 1, \ldots, \ell - 1$，执行以下操作。计算成本：(max$\{1, |S_{n,\text{per}}|\} d \log d + |S_{n,\text{per}} \cap \{i_0, \ldots, i_{\ell-1}\}| l_{\text{bit}}$, 0, 0, 0, 0)。7.2.4.1.1.1 如果 $i_t \in S_{n,\text{per}}$，则设置$\mathfrak{n}_{i_t} := n$。计算成本：(max$\{1, |S_{n,\text{per}}|\} \log d + l_{\text{bit}}$,0,0, 0,0)。7.2.5 设置 $\vec{\mathfrak{n}}_i :=$ ($\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\ell-1}}$)。计算成本：($dl_{\text{bit}}$,0,0,0, 0)。7.2.6 用同一循环等价类中字典序最小的序列覆盖 $\vec{\mathfrak{n}}_i$。计算成本：($d^2 \log dl_{\text{bit}}$, 0, 0, 0,0)。7.2.7 计算minperl($\vec{\mathfrak{n}}_i$)。计算成本：($d \log d \log\log d (\log d + l_{\text{bit}})$, 0, 0, 0, 0)。7.2.8 设置

- $\vec{\mathfrak{n}}'_i :=$ $[\mathfrak{n}_{i_0}, \mathfrak{n}_{i_1}, \ldots, \mathfrak{n}_{i_{\text{最小周期长度}(\vec{\mathfrak{n}}_i)-1}}]$,
- $\mathfrak{N}'_i :=$ $\{(\ell \cdot l', e_{i,\ \iota'}) :\ l' \in \{1, 2, \ldots, s\}, e_{i,\ \iota'} > 0\}$, 以及
- $\mathfrak{N}_i := \{\vec{\mathfrak{n}}'_i\} \times \mathfrak{N}'_i$.

QC: $(dl_{\text{bit}} + \tau(q-1) \log^{1+o(1)} q, 0, 0, 0, 0)$.

8 Set $\mathfrak{M} := \mathfrak{N} := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

9 Set $\vec{\mathfrak{n}}' := \{(\vec{\mathfrak{n}}'_i, i, \ell) : (i, \ell) \in \overline{\mathcal{L}}\}$, and sort it lexicographically.
QC: $(d^2 \log dl_{\text{bit}}, 0, 0, 0, 0)$.

10 For each $(\vec{\mathfrak{n}}'_i, i, \ell) \in \vec{\mathfrak{n}}'$, do the following.
QC: $(d^3 \tau (q-1)^2 \log q, 0, 0, 0, 0)$.

10.1 Check whether $\vec{\mathfrak{n}}'_i \in \mathfrak{M}$, and if so, skip to the next triple $(\vec{\mathfrak{n}}'_i, i, \ell)$.
QC: $(d^2 l_{\text{bit}}, 0, 0, 0, 0)$.

10.2 Add $\vec{\mathfrak{n}}'_i$ to $\mathfrak{M}$ as a new element.
QC: $(dl_{\text{bit}}, 0, 0, 0, 0)$.

10.3 Set $\mathfrak{V} := \mathfrak{N}'_i$.
QC: $(\tau(q-1) \log q, 0, 0, 0, 0)$.

10.4 For each $(\vec{\mathfrak{n}}'_j, j, \ell') \in \vec{\mathfrak{n}}'$ that comes after $(\vec{\mathfrak{n}}'_i, i, \ell)$, do the following.
QC: $(d^2 l_{\text{bit}} + d^2 \tau (q-1)^2 \log q, 0, 0, 0, 0)$.

10.4.1 Check whether $\vec{\mathfrak{n}}'_i = \vec{\mathfrak{n}}'_j$, and if not, skip to the next triple $(\vec{\mathfrak{n}}'_j, j, \ell')$.
QC: $(dl_{\text{bit}}, 0, 0, 0, 0)$.

10.4.2 For each $(l, k) \in \mathfrak{N}'_j$, do the following.
QC: $(d\tau (q-1)^2 \log q, 0, 0, 0, 0)$.

10.4.2.1检查 l 是否作为某个对 $(l,k')\in\mathfrak{V}$的第一项出现，并存储此信息。计算成本：$(d\tau(q-1)\log q, 0, 0,0, 0)$。10.4.2.2如果对于某个 k'，$(l,k')\in\mathfrak{V}$，则执行以下操作。10.4.2.2.1将 $\mathfrak{V}$ 中形如(l,k') 的唯一元素替换为$(l,k+k')$。计算成本：$(\log q, 0,0,0, 0)$。10.4.2.3 否则执行以下操作。10.4.2.3.1将 (l,k) 添加到 $\mathfrak{V}$ 中作为新元素。计算成本：$(\log q, 0, 0,0,0)$。10.5对于每个$(l,k)\in\mathfrak{V}$，执行以下操作。计算成本：$(d\tau(q-1)(dl_{\mathrm{bit}}+\log q), 0, 0, 0, 0)$。10.5.1将 $([\vec{\mathfrak{n}}'_i],l,k)$ 添加到 $\mathfrak{N}$ 中作为新元素。计算成本：$(dl_{\mathrm{bit}}+\log q, 0, 0, 0, 0)$。11 输出 $\mathfrak{R}$和 $\mathfrak{N}$，然后停止。

QC: $(N^2\log q+d\tau(q-1)(dl_{\mathrm{bit}}+\log q),0,0,0,0)$.

以下是引理5.3.2.12中的算法的伪代码。对于给定的递归树描述列表 $\vec{\mathfrak{D}}=(\mathfrak{D}_n)_{n=0,1,\ldots,N}$ 和 $\vec{\mathfrak{D}}'=(\mathfrak{D}'_n)_{n=0,1,\ldots,N'}$，满足引理 5.3.2.12的假设，该算法计算 $\vec{\mathfrak{D}}$ 和 $\vec{\mathfrak{D}}'$ 的同步 $(\vec{\mathfrak{D}}^+,\mathfrak{i})$。

1 Set $\mathfrak{N}:=\{0,1,\ldots,N\}$, and for $k\in\mathfrak{N}$, set $\mathfrak{D}_k^+:=\mathfrak{D}_k$. Moreover, let $\mathfrak{i}$ be the empty function $\emptyset$.
QC: $(N^2(l_{\mathrm{bit}}+m),0,0,0,0)$.

2 For each $n=0,1,\ldots,N'$, do the following.
QC: $((NN'\min\{N,N'\}+(N')^2l'_{\mathrm{bit}})(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m),0,0,0,0)$.

2.1 If $n=0$, then do the following.

2.1.1 Set $\mathfrak{i}(0):=0$.
QC: $(1,0,0,0,0)$.

2.2 Else do the following.

2.2.1 Let $\mathfrak{D}$ be the set of pairs obtained from $\mathfrak{D}'_n$ through replacing each first entry k of each pair in $\mathfrak{D}'_n$ by $\mathfrak{i}(k)$ (one may simply overwrite the corresponding entries of $\mathfrak{D}'_n$, so one does not need to handle the second entries of bit length in $O(m)$).
QC: $(N'(l_{\mathrm{bit}}+l'_{\mathrm{bit}}),0,0,0,0)$.

2.2.2 Sort $\mathfrak{D}$.
QC: $(N'l'_{\mathrm{bit}}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m))$.

2.2.3 Check whether $\mathfrak{D}=\mathfrak{D}_k^+$ for some (unique) $k\in\{0,1,\ldots,N\}$, and store this information (the truth value and k).
QC: $(N\min\{N,N'\}(l_{\mathrm{bit}}+l'_{\mathrm{bit}}+m),0,0,0,0)$.

2.2.4 If $\mathfrak{D}=\mathfrak{D}_k^+$ for some $k\in\{0,1,\ldots,N\}$, then do the following.

2.2.4.1 Set $\mathfrak{i}(n):=k$.
QC: $(l_{\mathrm{bit}}+l'_{\mathrm{bit}},0,0,0,0)$.

2.2.5 否则执行以下操作。2.2.5.1设置 $n' := \max \mathcal{N} + 1$，将 n' 添加到 $\mathcal{N}$ 作为新元素，设置 $\mathfrak{D}^+_{n'} := \mathfrak{D}$以及 i($n$):= n'。计算成本：($N'(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m)$，0，0，0，0)。3 设置 $\vec{\mathfrak{D}}^+ := (\mathfrak{D}^+_n)_{n\in\mathcal{N}}$，输出 ($\vec{\mathfrak{D}}^+$, i)并停止。计算成本：(($N + N'$) $\max\{N, N'\}(l_{\mathrm{bit}} + l'_{\mathrm{bit}} + m)$，0，0，0，0)。

最后，我们给出推论5.3.2.13 中算法的伪代码。对于给定的 $\mathbb{F}_q$ 的广义分圆映射 f_1 和 f_2，其指标分别为 d_1 和 d_2，且每个 f_j 都是特殊类型I 或 II（不一定二者属于相同的特殊类型），该算法判定$\Gamma_{f_1} \cong \Gamma_{f_2}$是否成立。在整个讨论过程中，我们有 $d := \max\{d_1, d_2\}$。

1 对于 $j = 1$、2，执行以下操
作。计算成本：

- ($d\log^{1+o_{(1)}} q, d$, 0, 0, 0)，如果 f_1 和 f_2 均为特殊类型 I;
- (d^2 对数 $d + d$对数$^{1+o_{(1)}} q, d$, 0, 0,0) 否则。1.1检查 f_j 是否属于特殊类型 I，并存储此信息以及 $\overline{f_j}$的诱导函数和与 f_j相关联的 $\mathbb{Z}/((q-1)/d_j)$上的仿射映射。计算成本：(d对数$^{1+o_{(1)}} q, d$, 0, 0, 0)。1.2 如果 f_j 不属于特殊类型 I，则检查 f_j 是否属于特殊类型II，并存储此信息。计算成本：(d对数^2 d,0,0, 0,0)，因为 $\overline{f_j}$和仿射映射已经计算完成。1.3 如果 f_j 既不属于特殊类型I，也不属于II型，则输出“失败”并停止。计算成本：(1, 0, 0, 0, 0)。

对于 $j = 1$,2，请执行以下操作。

QC:

- (d^3 对数^2 d + d^3 对数 $d\tau(q-1)$ + $d^3\tau(q-1)^2$ 对数 q + d^2 对数$^{1+o_{(1)}}$ q + $d\tau$(q −1) 对数$^{2+o_{(1)}} q, d, d$对数 q, d, 0) 如果 f_1 且 f_2 均为特殊类型 I;
- ($d^5 \log^2 d$ mpe($q-1$) + (d^5 mpe$(q-1)^3 + d^3\tau(q-1)^2$) log $q + d^2$ mpe(q −1) log$^{1+o_{(1)}} q + d\tau(q-1)$ log$^{2+o_{(1)}} q, d, d$ log q, d, 0) 否则。

2.1 如果 f_j 是特殊类型 I，则执行以下操作。

2.1.1计算I 型树寄存器 $\mathfrak{R}_j$ 的 f_j，以及相对于 $\mathfrak{R}_j$的树项链列表 $\mathfrak{N}_j$用于 f_j。计算成本：(d^3log^2 $d + d^3$log $d\tau(q-1) + d^3\tau(q-1)^2$log $q + d^2$log $^{1+o(1)}$ q +$d\tau(q-1)$log$^{2+o(1)}$ q, d, d log q, d, 0)。

2.2 否则，执行以下操作。

2.2.1 计算 f_j 的 II 型树寄存器 $\mathfrak{R}_j$，以及相对于 $\mathfrak{R}_j$ 的 f_j 的树项链列表 $\mathfrak{N}_{j\circ}$ 计算成本：(d^5 log^2 $d$mpe($q-1$)+(d^5mpe$(q-1)^3$+$d^3\tau(q-1)^2$) log q+d^2mpe($q-$ 1 $^{1+o(1)}$ $q + d\tau$ $q-1$ $^{2+o(1)}$ q d d q d 0) log ( ) log , , log , ,)。

3 对于 $j = 1$，令 2，设 $\vec{\mathfrak{D}}^{(j)} = (\mathfrak{D}_n^{(j)})_{n=0,\ 1,\ldots,N_j}$ 为 $\mathfrak{R}_j$ 的底层递归树描述列表。计算 $\vec{\mathfrak{D}}^{(1)}$ 和 $\vec{\mathfrak{D}}^{(2)}$ 的一个同步 $(\vec{\mathfrak{D}}^{+}, \mathfrak{i})$。

QC:

- $(d^3 \log\ q, 0,0,0,0)$，如果 f_1 和 f_2 都属于特殊类型 I；
- $(d^6\, \mathrm{mpe}(q-1)^3 \log\ q + d^4\, \mathrm{mpe}(q-1)^2 \log^{1+o_{(1)}} q, 0,0,0,0)$，否则。

4 创建一个修改后的版本 $\mathfrak{N}_2'$：对 $\mathfrak{N}_2$ 中的每个元素 $([\vec{\mathfrak{n}}], l, \mathfrak{m})$ 的首项$[\vec{\mathfrak{n}}]$ 中的每个条目 $\mathfrak{n}$，将其替换为 $\mathfrak{i}(\mathfrak{n}) \in \mathfrak{N}_2$；然后，将列表中由此产生的$O(d)$个不同三元组首项，分别改写为同一循环等价类中字典序最小的数序列；最后，通过合并 $\mathfrak{N}_2'$ 中对应于相同三元组首项的 $O(d)$ 个不同片段，按字典序对 $\mathfrak{N}_2'$进行排序。

计算成本：$(d^3\tau(q-1) \log\ q, 0, 0, 0, 0)$。

5 检查 $\mathfrak{N}_2' = \mathfrak{N}_1$，输出相应的真值，然后停止。计算成本：$(d^2\tau(q-1)\,\mathrm{log}\ q, 0, 0, 0, 0)$。

5.3.3 短期块行为及所有周期都很短的特殊情形

设 f 是 $\mathbb{F}_q$ 的指标为 d 的广义分圆映射。对于一个 f-周期的 $x \in \mathbb{F}_q$ 和 $t \in \mathbb{Z}$，令 $x^{(t)} := (f_{|\mathrm{per}(f)})^t(x)$；并回顾一下记号 $i_t := (\overline{f}_{|\mathrm{per}(\overline{f})})^t(i)$，它用于 $\overline{f}$-周期的 $i \in \{0, 1, \ldots, d-1\}$ 和 $t \in \mathbb{Z}$，以及 $i' := i_{-1}$。

在第 3.3 小节中，对于每个 $i \in \{0, 1, \ldots, d-1\}$，我们构造了 C_i 的一个算术划分 $\mathcal{P}_i$，使得对于 $x \in C_i$，$\mathrm{Tree}^{\Gamma_f}(x)$ 的同构类型仅取决于 x 所属的 $\mathcal{P}_i$-块。这里我们将细化这一构造。回顾一下，$\mathcal{P}_i = \mathcal{Q}_{i,\ H_i}$，其中 H_i 是$\bigcup_{t\in\mathbb{Z}} C_{i_t}$ 中某个 f-周期点上方的 Γ_{per}（即 Γ_f 在$\bigcup_{j\in\mathrm{per}(\overline{f})} C_j$ 上的诱导子图）的最大树高。C_i 的算术划分 $\mathcal{Q}_{i,\ h}$ 对所有 $h \in \mathbb{N}_0$ 均有定义（尽管我们在第 3.3 小节中只考虑了 $h \in \{0, 1, \ldots, H_i\}$ 的情形），并且满足

$$\mathcal{Q}_{i,h+1} = \mathcal{R}_i \wedge \mathfrak{P}'(\mathcal{Q}_{i',h}, A_{i'}). \tag{28}$$

这个公式是我们构造的关键。事实上，我们在此考虑的 C_i 的精细算术划分只是$\mathcal{Q}_{i,\ H_i+L-1}$，其中某个 $L \in \mathbb{N}^+$，而不是$\mathcal{P}_i = \mathcal{Q}_{i,H_i}$。虽然 $\mathcal{P}_i$ 的块控制了 Γ_f 中上述（周期的）顶点上的有根树的同构类型，但对于$\mathcal{Q}_{i,H_i+L-1}$，以下更一般的陈述成立。

引理5.3.3.1。设 $x \in C_i$ 是 f-周期的，且设 $L \in \mathbb{N}_0$。包含 x 的$\mathcal{Q}_{i,H_i+L-1}$-块唯一确定（长度为 L 的）序列

$$(\mathrm{Tree}_{\Gamma_f}(x^{(t)}))_{t=0,-1,\ldots,-L+1}$$

有根树同构类型。

证明。我们可以将 x 的 $\mathcal{Q}_{i,H_i+L-1}$-块写作 $\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i})$，其中 $o'_t \in \{\emptyset,\neg\}^{n_{i-t}}$。注意到

$$\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i}) \subseteq \mathcal{B}(\mathcal{P}_i, \diamond_{t=0}^{H_i}\vec{o'_t}\diamond\vec{\xi}_{i,H_i}),$$

我们看到，树$\Gamma_f(x)$ = 树$_i(\mathcal{P}_i, \diamond_{t=0}^{H_i}\vec{o'_t}\diamond\vec{\xi}_{i,H_i})$ 被唯一确定。

由公式 (28) 可知，对于 $\mathcal{Q}_{i',H_i+L-2}$ 的每个块 $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t}\diamond\vec{\xi})$，该块中 x 的 f-原像数量是常数

$$\sigma_{\mathcal{Q}_{i',H_i+L-2},A_{i'}}(\diamond_{t=0}^{H_i+L-2}\vec{o_t}\diamond\vec{\xi}, \diamond_{t=1}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i}). \tag{29}$$

Now we assume that $\vec{\xi} = \vec{\xi}_{i',H_i}$ (which is actually the same as $\vec{\xi}_{i,H_i}$). The union of all blocks of $\mathcal{Q}_{i',H_i+L-2}$ of the form $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t}\diamond\vec{\xi}_{i',H_i})$ (where $\vec{o_t}$ ranges over $\{\emptyset,\neg\}^{n_{i-t-1}}$ for each $t \in \{0,1,\dots,H_i+L-2\}$) is just the subset of $C_{i'}$ consisting of all f-periodic points in it. Since x has precisely one f-periodic pre-image (which lies in $C_{i'}$), it follows that the value of (29) for $\vec{\xi} = \vec{\xi}_{i',H_i}$ is 0 for all $\diamond_{t=0}^{H_i+L-2}\vec{o_t} \in \{\emptyset,\neg\}^{n_{i-1}+n_{i-2}+\cdots+n_{i-H_i-L+1}}$ except for one, for which the constant (29) has value 1. If $\diamond_{t=0}^{H_i+L-2}\vec{o_t}$ is that unique logical sign tuple, then the unique f-periodic pre-image $x^{(-1)}$ of $x \in \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i})$ always lies in $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \diamond_{t=0}^{H_i+L-2}\vec{o_t}\diamond\vec{\xi}_{i',H_i})$, and so $\mathrm{Tree}_{\Gamma_f}(x^{(-1)}) \cong \mathrm{Tree}_{i'}(\mathcal{P}_{i'}, \diamond_{t=0}^{H_i}\vec{o_t}\diamond\vec{\xi}_{i',H_i})$ is also uniquely determined. Continuing this process inductively, we get the statement of the lemma. □

为了后续的复杂度分析，我们需要引理 5.3.3.1 的一个更明确版本。对于每个 $\overline{f}$-周期的 $i \in \{0,1,\dots,d-1\}$ 以及每个 $L \in \mathbb{N}^+$，我们关联集合

$$\mathcal{O}_{i,L} := \{\diamond_{t=0}^{H_i+L-1}\vec{o'_t} \in \{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i-L+1}}} : \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i}) \neq \emptyset\}$$

的逻辑符号元组，这些元组对应于由 f-周期点组成的非空块$\mathcal{Q}_{i,\ H_i+L-1}$。引理 5.3.3.1的证明表明，只要 $L \geq 2$，我们就可以通过以下方式隐式定义一个（满射）函数 $\mathfrak{u}_{i,\ L}:\mathcal{O}_{i,\ L} \to \mathcal{O}_{i',\ L-1}$：

$$\sigma_{\mathcal{Q}_{i',H_i+L-2},A_{i'}}(\mathfrak{u}_{i,L}(\diamond_{t=0}^{H_i+L-1}\vec{o'_t})\diamond\vec{\xi}_{i',H_i}, \diamond_{t=1}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i}) = 1.$$

然后，对于每个（f-周期的） $x \in \mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t}\diamond\vec{\xi}_{i,H_i})$，位于 $C_{i'}$中的 x 的唯一 f-周期的原像 $x^{(-1)}$包含于 $\mathcal{B}(\mathcal{Q}_{i',H_i+L-2}, \mathfrak{u}_{i,L}(\diamond_{t=0}^{H_i+L-1}\vec{o'_t})\diamond\vec{\xi}_{i',H_i})$。记 $\mathrm{proj}_{i,L}$ 为投影

$$\mathcal{O}_{i,L} \to \{\emptyset,\neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i}}}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t} \mapsto \diamond_{t=0}^{H_i}\vec{o'_t},$$

因此，我们得到引理 5.3.3.1 的以下更明确的表述。

引理 5.3.3.2. 设 $L \in \mathbb{N}^+$，且 $x \in C_i$ 是 f-周期的，假设其包含于 $\mathcal{B}(\mathcal{Q}_{i,H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o'_t})$ 中。那么对于每个 $k = 0,-1,\dots,-L+1$，我们有

$$\mathrm{Tree}_{\Gamma_f}(x^{(k)}) \cong \mathrm{Tree}_{i_k}(\mathcal{P}_{i_k}, (\mathrm{proj}_{i_k,L+k}\circ\mathfrak{u}_{i_{k+1},L+k+1}\circ\mathfrak{u}_{i_{k+2},L+k+2}\circ\cdots\circ\mathfrak{u}_{i_0,L})(\diamond_{t=0}^{H_i+L-1}\vec{o'_t})).$$

我们注意到，引理 5.3.3.2 中公式里形如 $\mathfrak{u}_{i_l,L+t}$ 的函数复合在如果 $k=0$ （按指标来看，它应当从 1 递增到 0，这是没有意义的）时是空的。具体而言，关于 $k=0$ 的引理 5.3.3.2指出

$$\mathrm{Tree}_{\Gamma_f}(x)=\mathrm{Tree}_{\Gamma_f}(x^{(0)})=\mathrm{Tree}_{i_0}(\mathcal{P}_{i_0},\mathrm{proj}_{i_0,L}(\diamond_{t=0}^{H_i+L-1}\vec{o_t})),$$

对于 $k=-1$，它指出 that

$$\mathrm{Tree}_{\Gamma_f}(x^{(-1)})=\mathrm{Tree}_{i_{-1}}(\mathcal{P}_{i_{-1}},\mathrm{proj}_{i_{-1},L-1}\circ\mathfrak{u}_{i_0,L}(\diamond_{t=0}^{H_i+L-1}\vec{o_t})),$$

等等。

接下来，假设 f 的所有循环长度均不超过 L。我们考虑一个周期长度为 ℓ 的 $\overline{f}$- 周期指标 i。根据引理 5.3.3.2，对于每个 $x\in C_i$，包含 x 的 $\mathcal{Q}_{i,H_i+L-1}$ 块，以及 x 的精确 f-循环长度，共同完全确定了包含 x 的 Γ_f 连通分支的有向图同构类型。通过向 $\mathcal{Q}_{i,H_i+L-1}$ 的生成同余式添加适当的 s-同余式，我们可以构造一个更细的算术划分，以下记为 $\mathcal{W}_{i,L}$，其每个块都由具有相同 f-循环长度的点组成。因此，对于每个 f-周期点 $x\in C_i$，包含 x 的 Γ_f 连通分支的有向图同构类型完全由包含 x 的 $\mathcal{W}_{i,L}$-块确定。

下面讨论如何构造 $\mathcal{W}_{i,L}$ 的细节。回顾第 3.3 小节，对于每个 $l\in\mathbb{N}^+$， f^l 对 $C_{i_{-l}}$ 的限制（将其映射到 C_i）由仿射映射 $\mathcal{A}_{i,l}:x\mapsto\overline{\alpha}_{i,l}x+\overline{\beta}_{i,l}$ 表示（$\overline{\alpha}_{i,l}$ 和 $\overline{\beta}_{i,l}$ 的公式见命题 3.3.3 之后的第一个项目符号）。因此，将点 $x\in C_i$ 视为 $\mathbb{Z}/s\mathbb{Z}$ 的一个元素时，它是 f^l 的不动点，当且仅当 ℓ 整除 l（从而 $i_{-l}=i$），且 $\overline{\alpha}_{i,l}x+\overline{\beta}_{i,l}\equiv x$（模 s）。该同余式有解，当且仅当 $\gcd(s,\overline{\alpha}_{i,l}-1)\mid\overline{\beta}_{i,l}$；此时它等价于 s-同余式

$$x\equiv-\frac{\overline{\beta}_{i,l}}{\gcd(s,\overline{\alpha}_{i,l}-1)}\cdot\mathrm{inv}_{\frac{s}{\gcd(s,\overline{\alpha}_{i,l}-1)}}\left(\frac{\overline{\alpha}_{i,l}-1}{\gcd(s,\overline{\alpha}_{i,l}-1)}\right)\ \left(\mathrm{mod}\ \frac{s}{\gcd(s,\overline{\alpha}_{i,l}-1)}\right),$$

我们以下将其记为$\eta_{i,\ l}(x)$。注意，只有当 $\gcd(s,\overline{\alpha}_{i,\ l}-1)\mid\overline{\beta}_{i,\ l}$ 时，$\eta_{i,\ l}(x)$ 才有定义。令

$$\mathfrak{C}_{i,L}:=\{l\in\{1,2,\dots,L\}:\ell\mid l\text{ and }\gcd(s,\overline{\alpha}_{i,l}-1)\mid\overline{\beta}_{i,l}\}$$

并定义

$$\mathcal{V}_{i,L}:=\mathfrak{P}(\eta_{i,l}(x):l\in\mathfrak{C}_{i,L})\text{ and }\mathcal{W}_{i,L}:=\mathcal{Q}_{i,H_i+L-1}\wedge\mathcal{V}_{i,L}.$$

将 $\mathcal{V}_{i,L}$ 视为 C_i的一个算术划分，我们断言，其各块恰好是 C_i 的那些子集：每个子集由任意给定 f周期长度的所有点组成。事实上，设$l\in\{1,2,\dots,L\}$。若 $l\notin\mathfrak{C}_{i,L}$，则 f^l 在 C_i中没有不动点，特别是，f在 C_i中没有周期长度为 l 的点。另一方面，若 $l\in\mathfrak{C}_{i,L}$，则周期长度恰为 l 的 f点$x\in C_i$（若存在）恰好是满足同余式$\eta_{i,l'}(x)$的那些点，其中 $l'\in\mathfrak{C}_{i,L}$ 恰为 l的倍数。换言之，若对于$l'\in\mathfrak{C}_{i,L}$，我们定义

$$\nu_{l,l'}:=\begin{cases}\emptyset, & \text{if } l\mid l',\\ \neg, & \text{otherwise,}\end{cases}$$

并令 $\vec{\nu}_{i,L,l} := (\nu_{l,l'})_{l' \in \mathfrak{C}_{i,L}}$，那么集合 $\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$（可能为空）恰由周期长度为 l 的那些 $x \in C_i$ 构成。总之，我们得到以下结果。

命题 5.3.3.3. 设 $L \in \mathbb{N}^+$ 满足 f 的所有周期长度至多为 L，并设 $i \in \{0, 1, \ldots, d-1\}$ 是 $\overline{f}$- 周期的。我们将 $\mathcal{V}_{i,\,L}$ 和 $\mathcal{W}_{i,\,L}$ 视为 C_i 的算术划分。则有以下结论。

(1)$\mathcal{V}_{i,L}$ 的每个块分别是以下形式之一： $\mathcal{B}(\mathcal{V}_{i,L}, (\neg, \neg, \ldots, \neg))$，以及$\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$，其中$l \in \mathfrak{C}_{i,L}$；它分别恰好由 C_i中的 f-暂态点，以及 C_i 中具有恰好为 l的 f-周期长度的那些 f-周期点组成。

(2) $\mathcal{W}_{i,\,L}$ 的每个块要么完全由 f-周期点组成，要么完全由 f-暂态点组成。此外，$\mathcal{W}_{i,\,L}$ 中元素为 f-周期的每个块具有以下形式

$$\mathcal{B}(\mathcal{W}_{i,L}, \diamond_{t=0}^{H_i+L-1} \vec{o'_t} \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l})$$

$\vec{}$对于某些 $o'_t \in \{\emptyset, \neg\}^{n_{i-t}}$ 和 $l \in \mathfrak{C}_{i,\,L}$，在这种情况下，对于该块中的任意给定点 x，包含 x 的 Γ_f的连通分支的有向图同构类型由以下循环序列表示

$$[\mathrm{Tree}_{i_k}(\mathcal{P}_{i_k}, (\mathrm{proj}_{i_k,L+k} \circ \mathfrak{u}_{i_{k+1},L+k+1} \circ \mathfrak{u}_{i_{k+2},L+k+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{t=0}^{H_i+L} \vec{o'_t}))]_{k=-l+1,-l+2,\ldots,0}$$

有根树同构类型。

命题 5.3.3.3 是证明以下定理的基础。

定理 5.3.3.4。设 f 是 $\mathbb{F}_q$的指标为 d 的广义分圆映射。此外，设 $L \in \mathbb{N}^+$ 满足 $L \le q-1$，使得 f 的所有周期长度均不超过 L。那么，在 q有界查询复杂度内

$$(8^{d^2\,\mathrm{mpe}(q-1)+dL} 2^{d\,\mathrm{mpe}(q-1)} (d^3 L\,\mathrm{mpe}(q-1) + d^2L^2) \log^{1+o(1)} q, d, 0, 0, 0),$$

从而在 q有界拉斯维加斯双重复杂度内

$$(8^{d^2\,\mathrm{mpe}(q-1)+dL} 2^{d\,\mathrm{mpe}(q-1)} (d^3 L\,\mathrm{mpe}(q-1) + d^2L^2) \log^{1+o(1)} q + d \log^{3+o(1)} q, d \log^{3+o(1)} q, d \log q),$$

可以计算

- 一个递归树描述列表 $\vec{\mathfrak{D}} = (\mathfrak{D}_n)_{n=0,\ 1,\ \ldots,\ N}$ 其中 $N \in O(d2^{d^2\,\mathrm{mpe}(q-1)+d})$，$\vec{}$其关联的有根树同构类型序列记为 $\mathfrak{I} = (\mathfrak{I}_n)_{n=0,\ 1,\ \ldots,\ N}$，使得描述 $\mathfrak{D}_n$ 中元素的索引 $n \in \{0, 1, \ldots, N\}$ 以及第二项均由长度为 $l_{\text{比特}} := \lfloor \log_2 q \rfloor + 1$ 的比特串表示；并且→

- 相对于 $\mathfrak{I}$的树项链列表 $\mathfrak{N}$， f 按照定义 5.3.2.8，以字典序排序的形式，其中包含 $O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL})$ 个不同的元素（三元组），并且按照约定，

– 将其元素的第一项 $[\vec{\mathfrak{n}}] = [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ 按照类似注 5.3.2.9(3) 的方式填充，但加入 $L - l'$ 个虚拟项 -1，使得 $[\vec{\mathfrak{n}}]$ 的位串表示始终具有 $L(l_{\mathrm{bit}} + 1) = L(\lfloor \log_2 q \rfloor + 2)$ 的长度；– 使用 $\lfloor \log_2 L \rfloor + 1$ 位表示其元素的第二项；并且 – 使用 $\lfloor \log_2 q \rfloor + 1$ 位表示其元素的第三项。

证明。首先，我们计算 $\overline{f}$、仿射映射 A_i 以及一个分区树寄存器

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1}, ((\mathfrak{D}_n, (S_{n,i})_{i=0,1,\ldots,d}))_{n=0,1,\ldots,N})$$

对于带有 $N \in O(d2^{d^2 \mathrm{mpe}(q-1)+d})$ 的 f；所需的递归树描述列表 $\vec{\mathfrak{D}}$ 是其中的一部分。根据命题 5.1.8 和定理 5.1.9(2)，这些计算可以在 q 有界查询复杂度内完成

$$(d^3 \operatorname{mpe}(q-1) 2^{(3d^2+d)\operatorname{mpe}(q-1)+2d} \log^{1+o(1)} q, d, 0, 0, 0),$$

该复杂度被计算 $\vec{\mathfrak{D}}$ 和 $\mathfrak{N}$ 所需的整体 q 有界查询复杂度所支配（正是这一项使得必须在整体位操作成本的界中纳入因数分解 $2^{d \operatorname{mpe}(q-1)}$）。此外，根据定理 5.1.9(2) 的证明，以下内容也作为其中的一部分被计算出来（并且可以存储）：

- $\overline{f}$ 的循环以及 $\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$；
- 每个 $\overline{f}$-周期的指标 $i \in \{0, 1, \ldots, d-1\}$ 的参数 H_i。

在另行通知之前，我们假设$(i, \ell) \in \overline{\mathcal{L}}$ 其中 $i < d$ 是固定的（$i = d$ 的情形很容易单独处理，并将在本证明末尾“附带处理”）。按照惯例，我们令$(i_0, i_1, \ldots, i_{\ell-1})$ 其中 $i = i_0$ 为 i的 $\overline{f}$-循环，并对任意$t \in \mathbb{Z}$设 $i_t := i_{t \bmod \ell}$。我们分析计算与$\bigcup_{t=0}^{d-1} C_{i_t}$ 相交的 Γ_f 的连通分支的同构类型时的位操作成本（即，这些连通分支可以由陪集 C_{i_t} 中的某个周期顶点表示）。我们注意到，对于每个 $t \in \{0, 1, \ldots, \ell-1\}$，都可以直接从上面计算出的分区树寄存器中读出 $\mathcal{U}_{i_t}$的一个长度为 $H_i \le d\operatorname{mpe}(q-1)$的生成同余序列。接下来，我们需要为 $t = 0$，1，…，$L-1$ 确定一个 $\mathcal{Q}_{i_{-t},\ H_i+L-t-1}$的生成同余序列。根据第 3.3小节中 $\mathcal{R}_i$ 和 $\mathcal{Q}_{i,\ h}$ 的定义，我们有

$$\mathcal{Q}_{i_{-t}, H_i+L-t-1} = \mathcal{R}_{i_{-t}} \wedge \mathfrak{P}'(\mathcal{Q}_{i_{-t-1}, H_i+L-t-2}, A_{i_{-t-1}}).$$

此外，我们观察到

$$\mathfrak{P}'(\mathcal{Q}_{i_{-t-1}, H_i+L-t-2}, A_{i_{-t-1}}) = \mathfrak{P}'\left(\bigwedge_{j=0}^{H_i+L-t-2} \lambda^j_{i_{-t-1-j}}(\mathcal{R}_{i_{-t-1-j}}) \wedge \mathcal{U}_{i_{-t-1}}, A_{i_{-t-1}}\right)$$
$$= \bigwedge_{j=0}^{H_i+L-t-2} \lambda^{j+1}_{i_{-t-1-j}}(\mathcal{R}_{i_{-t-1-j}}) \wedge \mathcal{U}_{i_{-t}} = \bigwedge_{j=1}^{H_i+L-t-1} \lambda^j_{i_{-t-j}}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}},$$

由此

$$\mathcal{Q}_{i_{-t}, H_i+L-t-1} = \bigwedge_{j=0}^{H_i+L-t-1} \lambda^j_{i_{-t-j}}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}}.$$

现在，我们可以从分区树寄存器中直接读出
$\lambda_{i_t}^{j}\ \mathcal{R}_{i_t}$ $n_{i_t} \le d$ $t = 0\ 1\ \ldots\ \ell - 1$ $j =$( ) 的生成同余
$0\ 1\ \ldots\ H_i$ $O\ d^3$ $q-1$ q
序列，长度为，对于每个，，，以及每个，，，，而复制这些序列总共只需 (mpe( ) log)次位运算。此外，对于任意固定的 $t \in \{0, 1, \ldots, \ell - 1\}$，我们可以计算一个生成同余序列，用于

$$\lambda_{i_t}^{j}(\mathcal{R}_{i_t}) = \lambda(\lambda_{i_t}^{j-1}(\mathcal{R}_{i_t}), A_{i_{t+j-1}})$$

依次对 $j = H_i + 1$、$H_i + 2$、.....、$H_i + L - 1$进行。对于给定的每个 t和 j，这需要 $O(d\log^{1+o(1)} q)$ 次位运算，因此对所有 t和 j合计而言，需要$O(d^2L\log^{1+o(1)} q)$ 次位运算。一旦计算出所有这些生成同余序列，就可以将这样的序列拼接起来，得到一个形如 $\mathcal{Q}_{i_{-t},\ H_i+L-t-1}$的单个划分，所需位运算次数为 $O((H_i + L)d\log\ q) \subseteq O((d^2$ mpe$(q-1)\ + dL)\log\ q)$。对所有 $t = 0$、1、.....、$L - 1$ 执行此操作，需要 $O((d^2L$ mpe$(q-1)\ + dL^2)\log\ q)$ 次位运算。

我们的下一个目标是计算这个函数

$$\mathfrak{u}_{i_{-t},L-t} : \mathcal{O}_{i_{-t},L-t} \to \mathcal{O}_{i_{-t-1},L-t-1}$$

对于 $t = 0,1, \ldots, L-2$。为此，我们首先计算集合

$$\begin{aligned}&\mathcal{O}_{i_{-t},L-t} = \\ &\{\diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \in \{\emptyset, \neg\}^{n_{i_{-t}}+n_{i_{-t+1}}+\cdots+n_{i_{-H_i}-L+1}} : \\ &\mathcal{B}(\mathcal{Q}_{i_{-t},H_i+L-t-1}, \diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \diamond \vec{\xi}_{i_{-t},H_i}) \neq \emptyset\}\end{aligned}$$

对于 $t = 0,1, \ldots, L-1$。为此，我们遍历这些

$$O(2^{d(H_i+L-t)}) \subseteq O(2^{d^2\ \mathrm{mpe}(q-1)+dL-dt})$$

元组 $\diamond_{k=0}^{H_i+L-t-1}\vec{o_k}$，并对每个元组计算块的基数。 k

$$\mathcal{B}(\mathcal{Q}_{i_{-t},H_i+L-t-1}, \diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \diamond \vec{\xi}_{i_{-t},H_i}).$$

按照导出命题 3.3.2 的思路，该基数等于分-布数

$$\sigma_{\mathcal{Q}_{i_{-t},H_i+L-t-1},\mathbf{0}}(\diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \diamond \vec{\xi}_{i_{-t},H_i}, (\emptyset, \ldots, \emptyset)) \tag{30}$$

的$\mathcal{Q}_{i_{-t},\ H_i+L-t-1}$ 在恒为零的仿射函数 0 下。根据引理 5.2.2.1 的证明以及以下事实

- 我们使用的$\mathcal{Q}_{i_{-t},\ H_i+L-t-1}$ 生成同余式的数量至多为$d(H_i + L)\ + H_i \le d^2$ mpe$(q-1)\ + d$mpe$(q-1)\ + dL \in O(d^2$ mpe$(q-1)\ + dL)$,
- 我们需要遍历的子集 J 中，绝不会包含对应于某个 H_i 生成同余式中逻辑符号的指标$\vec{}$，因为$\xi_{i_{-t},\ H_i}$仅由正逻辑符号构成，而且

- $d(H_i + L)\ \le d^2$mpe$(q-1)\ + dL$,

我们得出结论，计算分布数 (30) 的复杂度属于

$$O(2^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)}q). \tag{31}$$

总之，计算所有集合 $\mathcal{O}_{i_{-t},L-t}$ 对于 $t=0,1,\ldots,L-1$ 需要

$$\begin{aligned}&O\left(\sum_{t=0}^{L-1}4^{d^2\,\mathrm{mpe}(q-1)+dL}2^{-dt}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)}q\right)\\ \subseteq&O(4^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)}q).\end{aligned}$$

位运算。

关于函数$\mathfrak{u}_{i_{-t},L-t}$ 本身的计算，我们注意到，对于给定的 t 和$\mathfrak{u}_{i_{-t},L-t}$ 的参数 $\diamond_{k=0}^{H_i+L-t-1}\vec{o_k'}$，相关函数值$\mathfrak{u}_{i_{-t},L-t}(\diamond_{k=0}^{H_i+L-t-1}\vec{o_k'})$ 是满足以下条件的唯一元组 $\diamond_{k=0}^{H_i+L-t-2}\vec{o_k}\in\mathcal{O}_{i_{-t-1},L-t-1}$，即分布数

$$\sigma_{\mathcal{Q}_{i_{-t-1},H_i+L-t-2},A_{i_{-t-1}}}(\diamond_{k=0}^{H_i+L-t-2}\vec{o_k}\diamond\vec{\xi}_{i_{-t-1},H_i},\diamond_{k=1}^{H_i+L-t-1}\vec{o_k'}\diamond\vec{\xi}_{i_{-t},H_i})$$

等于 1（并且根据引理5.3.3.1 的证明，对于 $\diamond_{k=0}^{H_i+L-t-2}\vec{o_k}$的所有其他选择，该分布数均为 0）。因此，为了计算每个函数 $\mathfrak{u}_{i_{-t},L-t}$for $t=0,1,\ldots,L-2$ 的每个值，我们需要进行

$$O\left(\sum_{t=0}^{L-2}4^{d^2\,\mathrm{mpe}(q-1)+dL-dt}2^{-d}\right)\subseteq O\left(4^{d^2\,\mathrm{mpe}(q-1)+dL}\right)$$

上述形式的分布数 $\mathcal{Q}_{i_{-t-1},\ H_i+L-t-2}$ 的计算，并且如上所述（这次利用了 $\vec{\xi}_{i_{-t-1},\ H_i}$ 的所有条目都是正逻辑符号这一点），每次此类分布数计算的位操作成本为 (31)。因此，我们得到的总位操作成本为

$$O(8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)}q)$$

用于计算所有函数$\mathfrak{u}_{i_{-t},L-t}$。

接下来，我们计算集合 $\mathfrak{C}_{i,L}$，并对每个 $l\in\mathfrak{C}_{i,L}$，计算 s-同余式$\eta_{i,l}(x)$。根据引理 5.1.5(1,3,4,8)，若对每个满足 $\ell\mid l$ 的 l 存储线性系数 $\overline{\alpha}_{i,l}$ 和常数系数 $\overline{\beta}_{i,l}$，那么只要 $\mathcal{A}_{i,\ l}=(A_{i_0}A_{i_1}\cdots A_{i_{\ell-1}})^{l/\ell}=\mathcal{A}_i^{l/\ell}$，这些计算总共需要 $O((d+L)\log^{1+o(1)}q)$ 次位运算；随后利用公式 $\mathcal{A}_{i,l+\ell}=\mathcal{A}_{i,l}\mathcal{A}_i$，在 $\ell\mid l$ 时为下一个较大的相关值 $l+\ell$ 进行计算。通过这些计算，我们为 $\mathcal{W}_{i,L}$ 建立了一个生成同余序列，其长度至多为 $d(H_i+L)+H_i+L\le d^2\,\mathrm{mpe}(q-1)+d\mathrm{mpe}(q-1)+dL+L$。现在，我们遍历 $O(2^{d(H_i+L)}L)\subseteq O(2^{d^2\,\mathrm{mpe}(q-1)+dL}L)$ 个逻辑符号元组 $\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\nu}_{i,L,l}$，它们参数化了 $\mathbb{Z}/s\mathbb{Z}$ 的形如 $\mathcal{B}(\mathcal{W}_{i,L},\ \diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\xi}_{i,H_i}\diamond\vec{\nu}_{i,L,l})$ 的子集——我们注意到，非空的这类集合恰好是 $\mathcal{W}_{i,L}$ 中由 f-周期点组成的那些块。对于每个这样的元组，我们计算

$$\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\nu}_{i,L,l}):=|\mathcal{B}(\mathcal{W}_{i,L},\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\xi}_{i,H_i}\diamond\vec{\nu}_{i,L,l})|$$

作为分布数

$$\sigma_{\mathcal{W}_{i,L},\mathbf{0}}(\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\xi}_{i,H_i}\diamond\vec{\nu}_{i,L,l},(\emptyset,\emptyset,\ldots,\emptyset)),$$

其计算成本为

$$\begin{aligned}&O(2^{d(H_i+L)+L}(d(H_i+L)+H_i+L)\log^{1+o(1)}q)\\&\subseteq O(2^{d^2\,\mathrm{mpe}(q-1)+dL+L}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)}q)\end{aligned}$$

位运算。若 $\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\nu}_{i,\ L,\ l})=0$，则舍弃该情形。否则，存储计算得到的块大小。总体而言，该过程需要

$$\begin{aligned}&O(4^{d^2\,\mathrm{mpe}(q-1)+dL}2^L(d^2L\,\mathrm{mpe}(q-1)+dL^2)\log^{1+o(1)}q)\\\subseteq&O(8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2L\,\mathrm{mpe}(q-1)+dL^2)\log^{1+o(1)}q)\end{aligned}$$

位运算。

我们回顾一下，根据引理 5.3.3.1 和引理 5.3.3.2，由 f-周期点构成的 $\mathcal{W}_{i,\ L}$ 的块决定了其中任意给定点所属连通分支的有向图同构类型。对于每个 $\mathfrak{m}_i$-值（即相应的块大小）非零的元组 $\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\nu}_{i,\ L,\ l}$，我们使用引理 5.3.3.2 计算唯一的循环序列 $[\mathfrak{n}_1,\mathfrak{n}_2,\ldots,\mathfrak{n}_{l'}]$，其元素取自 $\{0,1,\ldots,N\}$，且周期长度最小为 l'；该循环序列的有根树同构类型序列刻画相应的有向图同构类型，并且等于

$$[\diamond_{m=1}^{l/l'}(\mathfrak{I}_{\mathfrak{n}_1},\mathfrak{I}_{\mathfrak{n}_2},\ldots,\mathfrak{I}_{\mathfrak{n}_{l'}})].$$

为此，对于固定的 $\diamond_{k=0}^{H_i+L-1}\vec{o_k'}\diamond\vec{\nu}_{i,L,l}$，我们计算该序列。

$$(\mathfrak{n}^{(-l+1)},\mathfrak{n}^{(-l+2)},\ldots,\mathfrak{n}^{(0)})$$

其中，$\mathfrak{n}^{(-t)}$ 对于 $t\in\{0,1,\ldots,l-1\}$ 是 $\{0,1,\ldots,N\}$ 中使得

$$\begin{aligned}&\mathrm{Tree}_{i_{-t}}(\mathcal{P}_{i_{-t}},(\mathrm{proj}_{i_{-t,L-t}}\circ\mathfrak{u}_{i_{-t+1},L-t+1}\circ\mathfrak{u}_{i_{-t+2},L-t+2}\circ\cdots\circ\mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o_k'}))\\\cong&\mathfrak{I}_{\mathfrak{n}^{(-t)}}.\end{aligned}$$

对于每个固定的 t，这要求我们计算逻辑符号元组

$$(\mathrm{proj}_{i_{-t,L-t}}\circ\mathfrak{u}_{i_{-t+1},L-t+1}\circ\mathfrak{u}_{i_{-t+2},L-t+2}\circ\cdots\circ\mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o_k'}).\tag{32}$$

现在，我们可以计算

$$(\mathfrak{u}_{i_{-t+1},L-t+1}\circ\mathfrak{u}_{i_{-t+2},L-t+2}\circ\cdots\circ\mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1}\vec{o_k'})$$

只需查找函数 $\mathfrak{u}_{i_{-j},L-j}$ 的预计算值即可，这需要 $O(t(H_i+L)d)\subseteq O(d^2L\mathrm{mpe}(q-1)+dL^2)$ 次位运算。该元组的 $\mathrm{proj}_{i_t,L-t}$ 值是到一个初始段上的投影，可以使用

$O(d(H_i+1)) \subseteq O(d^2\mathrm{mpe}(q-1))$ 次位运算。最后，我们需要查找分区树寄存器中的数值 $\mathfrak{n}^{(-t)}$，它由包含关系刻画

$$(\mathrm{proj}_{i_{-t},L-t} \circ \mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1} \vec{o_k'}) \in S_{\mathfrak{n}^{(-t)},i_{-t},H_i}.$$

因此，为了确定并存储 $\mathfrak{n}^{(-t)}$，我们需要依次遍历两两不相交的集合 S_{n,i_{-t},H_i} 对于 $n = 0$、1、.. .、N，直到找到包含逻辑符号元组 (32) 的集合。由于 $\sum_{n=0}^{N} |S_{n,i_{-t},H_i}| \le 2^{d(H_i+1}) \in O(2^{d^2}{}_{\mathrm{mpe}(q-1)^{+d}})$，且每个集合 S_{n,i_{-t},H_i} 中的每个元素都是长度属于 $O(d(H_i+1)) \subseteq O(d^2\mathrm{mpe}(q-1))$ 的逻辑符号元组，这需要

$$O(N + 2^{d^2\,\mathrm{mpe}(q-1)+d} d^2\,\mathrm{mpe}(q-1) + \log q) \subseteq O(d^2\,\mathrm{mpe}(q-1) 2^{d^2\,\mathrm{mpe}(q-1)+d} + \log q)$$

位运算来确定 $\mathfrak{n}^{(-t)}$ 对于我们的固定值 t。总之，对于固定值 $\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}$，计算序列 $\vec{\mathfrak{n}} = (\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(0)})$需要

$$O(d^2L^2\,\mathrm{mpe}(q-1) + dL^3 + d^2L\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d} + L\log q)$$

位运算。我们要找的循环序列 $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ 就是 $[\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(-l+l')}]$，其中 $l' = \mathrm{minper}(\vec{\mathfrak{n}})$，可通过.....计算得出。

$$O(l^{1+o(1)} l_{\mathrm{bit}}) \subseteq O(L^{1+o(1)} \log q)$$

依据引理 5.3.2.7 的位运算。最后，在表示此循环序列时，我们希望将 $(\mathfrak{n}_1,\ldots,\mathfrak{n}_{l'})$ 替换为同一循环等价类中的字典序最小有序序列。对于每个此类序列，这还需要 $O(L^2 \log L \log q)$ 次位运算。总计，对于所有的 $O(2^{d(H_i+L)}L) \subseteq O(2^{d^2}{}_{\mathrm{mpe}(q-1)^{+dL}}L)$ 个逻辑符号元组 $\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}$，循环序列 $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$ 的计算需要

$$\begin{aligned} &O(2^{d^2\,\mathrm{mpe}(q-1)+dL} L \cdot (d^2L^2\,\mathrm{mpe}(q-1) + dL^3 + d^2L\,\mathrm{mpe}(q-1)2^{d^2\,\mathrm{mpe}(q-1)+d} \\ &+ L^2 \log L \log q)) \subseteq O(8^{d^2\,\mathrm{mpe}(q-1)+dL} \log^{1+o(1)} q) \end{aligned}$$

位运算。总之，在固定 $(i,\ell) \in \overline{\mathcal{L}} \setminus \{(d,1)\}$ 后，对所有这些 $O(d)$ 对 (i,ℓ) 进行的计算，其位操作成本为

$$O(8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^3L\,\mathrm{mpe}(q-1) + d^2L^2)\log^{1+o(1)} q).$$

此时，对于每个 $(i,\ell) \in \overline{\mathcal{L}} \setminus \{(d,1)\}$，我们已经计算出一个形如 $\mathfrak{N}_i$ 的集合

$$\begin{aligned} &\mathfrak{N}_i = \\ &\{(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}, [\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], \mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l})) : \\ &\vec{o_k'} \in \{\emptyset, \neg\}^{n_{i-k}} \text{ for } k = 0,1,\ldots,H_i+L-1, l \in \mathfrak{C}_{i,L}, \text{ and} \\ &\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}) > 0\}. \end{aligned}$$

我们注意到，对于 $\mathfrak{N}_i$中的每个元素，都有 $l' \mid l$，且 $\mathsf{minper}([\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]) = l'$。我们还观察到，尽管相对于 $\mathfrak{I}$而言， $\mathfrak{N}_i$ 并不是将 f 限制在所有陪集 C_j 的并集上的树项链列表（其中 j 是包含 i的 $\Gamma_{\overline{f}}$的连通分支中的一个顶点），但该树项链列表可以根据 $\mathfrak{N}_i$ 轻松推导如下。固定 l和 $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$，并将 $\mathfrak{N}_i$ 中相应三元组的所有第三项相加（即第一项具有终止片段 $\vec{\nu}_{i,\ L,\ l}$且第二项为 $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$的那些三元组），所得结果恰好是 Γ_f 的连通分量数量；这些连通分量由有根树同构类型的循环序列 $[\diamond_{m=1}^{l/l'}(\mathfrak{I}_{\mathfrak{n}_1}, \ldots, \mathfrak{I}_{\mathfrak{n}_{l'}})]$ 刻画，并且包含在所有陪集 C_j （其中 j 属于包含 i的 $\Gamma_{\overline{f}}$ 的连通分支）的并集中。该观察还意味着，我们可以如下计算相对于 $\mathfrak{I}$ 的 f 的完整树项链列表 $\mathfrak{N}$ 。

我们从设定 $\mathfrak{N} := \emptyset$ 开始。在下文所述的整个过程中， $\mathfrak{N}$ 是由三元组 $([\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], l, \mathfrak{m})$ 构成的集合，使得

- $|\mathfrak{N}| \le \sum_{(i,\ell)\in\overline{\mathcal{L}}\setminus\{(d,1)\}} |\mathfrak{N}_i| \in O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL})$;
- $\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'} \in \{0, 1, \ldots, N\}$;
- $\mathsf{minper}([\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]) = l'$;
- $[\diamond_{m=1}^{l/l'}(\mathfrak{I}_{\mathfrak{n}_1}, \mathfrak{I}_{\mathfrak{n}_2}, \ldots, \mathfrak{I}_{\mathfrak{n}_{l'}})]$ 是刻画包含于$\mathbb{F}_q^*$中的 Γ_f 的至少一个连通分支的有根树同构类型的循环序列；并且
- $\mathfrak{m} \in \mathbb{N}^+$ 是包含在$\mathbb{F}_q^*$中且由 $[\diamond_{m=1}^{l/l'}(\mathfrak{I}_{\mathfrak{n}_1}, \mathfrak{I}_{\mathfrak{n}_2}, \ldots, \mathfrak{I}_{\mathfrak{n}_{l'}})]$刻画的 Γ_f 的连通分量的确切数量。

我们注意到， $\mathfrak{N}$ 中每个三元组的第一项都是一个长度为 $O(L)$ 的循环序列，其中每一项的位长度为 $O(\log N) \subseteq O(d^2\,\mathrm{mpe}(q-1))$；这类三元组的第二项和第三项则分别可以表示为长度为 $O(\log L)$ 和 $O(\log q)$ 的比特串。现在，为了得到 $\mathfrak{N}$ 的近似最终形式，我们遍历数对 $(i, \ell) \in \overline{\mathcal{L}} \setminus \{(d, 1)\}$，并针对每个这样的数对遍历 $\mathfrak{N}_i$ 中的元素；这个双重循环包含 $O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL})$ 次单独的迭代。在每次迭代中，我们最多向 $\mathfrak{N}$ 添加一个新元素，这解释了整个过程中始终成立的上述 $|\mathfrak{N}|$ 上界。具体来说，一次迭代由以下步骤组成。对于我们正在考察的 $\mathfrak{N}_i$ 中的三元组，我们有参数 l，它们可以通过 $O(l \log l) \subseteq O(L^{1+o(1)})$ 次位操作，从三元组的第一项 $\diamond_{k=0}^{H_i+L} \vec{o}_k \diamond \vec{\nu}_{i,L,l}$ 中读取出来（扫描以找到 $\vec{\nu}_{i,L,l}$ 中第一个正逻辑符号，并在此过程中递增计数器），以及 $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$。我们检查这些是否已经作为某个 $\mathfrak{N}$ 元素的前两项出现，这需要

$$O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL} \cdot L \log q) = O(dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL} \log q)$$

位运算。如果是这样，我们就结束当前迭代，不向 $\mathfrak{N}$ 添加任何内容。否则，我们计算与参数 l 和 $[\mathfrak{n}_1, \ldots, \mathfrak{n}_{l'}]$ 相关联的所有三元组中第三个元素的总和 $\mathfrak{m}$，这些三元组位于$\bigcup_{(j,\ell')\in\overline{\mathcal{L}}\setminus\{(d,1)\}} \mathfrak{N}_j$。这需要

$$O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}(L \log L + L \log q) = O(dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL} \log q)$$

位运算。然后，我们将 $([\mathfrak{n}_1,\dots,\mathfrak{n}_{l'}],l,\mathfrak{m})$ 作为新元素添加到 $\mathfrak{N}$ 中，并结束当前迭代。总体而言，这个双重循环需要

$$O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}\cdot dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)=O(d^2L^3 4^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$
$$\subseteq O(8^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$

位运算。

在双重循环结束时，$\mathfrak{N}$ 几乎等于相对于 $\mathfrak{I}$的 $\vec{f}$树项链列表；只有包含$0_{\mathbb{F}_q}$的 Γ_f 连通分支尚未计入，该分支由循环序列 $[\text{树}\Gamma_f(0_{\mathbb{F}_q})]$,刻画。为了求出相对于我们的分区树寄存器而言，树$\Gamma_f(0_{\mathbb{F}_q})$的指标 $\mathfrak{n}\in\{0,1,\dots,N\}$，我们注意到 $\mathfrak{n}$ 由等式$S_{\mathfrak{n},d}=\emptyset$（正逻辑符号）刻画。因此，我们只需额外进行 $O(N)\subseteq O(d2^{d^2\,\mathrm{mpe}(q-1)+d})$位运算即可求出 $\mathfrak{n}$。接下来，我们需要检查 $[\mathfrak{n}]$ 和1是否已经作为某个三元组的前两个条目出现在 $\mathfrak{N}$中，这需要.....

$$O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}\cdot\log q)\subseteq O(8^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$

位运算。如果是这样，我们将该三元组的第三个条目加 1，然后停止。否则，我们将 $([\mathfrak{n}],1,1)$ 作为新元素添加到 $\mathfrak{N}$ 中。

最后，我们将表示 $\mathfrak{N}$的计算数组按字典序排序。根据引理 5.1.5(10)，由于 $|\mathfrak{N}|\in O(dL2^{d^2\mathrm{mpe}(q-1)+dL})$，且数组中的每个条目的位长度为 $O(L\log q)$，这需要

$$O(dL^2(d^2\,\mathrm{mpe}(q-1)+dL)2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)\subseteq O(8^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$

位运算。最后输出 $(\vec{\mathfrak{O}},\mathfrak{N})$ 并终止。 □

当然，除非能够高效计算广义分圆映射的最大循环长度，否则定理 5.3.3.4 在实践中并无用处。下面的命题解决了这一问题。

命题5.3.3.5。设 f 是 $\mathbb{F}_q$的一个指标为 d 的广义分圆映射。 $f_{|\mathrm{per}(f)}$ 的最大循环长度可以在 q有界查询复杂度内计算

$$(d\log^{2+o(1)}q+d^2\log^2 d,d,1,d,0).$$

证明。这与命题 5.3.2.3 的思路类似，但由于这里不需要详细列出完整的循环类型，因此我们可以得到比直接使用该命题更好的界。首先，我们计算 $\overline{f}$、 A_i 的仿射映射、 $\overline{f}$ 的循环以及 $\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$。根据命题 5.1.8 以及第 5.2.1 小小节的开头部分，这需要 q 有界的查询复杂度

$$(d\log^{1+o(1)}q+d^2\log^2 d,d,0,0,0).$$

我们还对 s 进行因数分解，只使用一次 q 有界的 mdl 查询（即花费 q 有界查询复杂度（log q, 0,1,0,0））。

现在，我们遍历这些对 $(i,\ell) \in \overline{\mathcal{L}} \setminus \{(d,1)\}$，并对其中每一对执行以下操作。首先，我们计算 $\mathcal{A}_i = A_{i_0}A_{i_1}\cdots A_{i_{\ell-1}}$：$x \mapsto \overline{\alpha}_i x + \overline{\beta}_i$，根据第 5.2.1 小节的开头，这需要进行$O(\ell \log^{1+o(1)} q)$ 次位操作（针对 (i,ℓ)）。接下来，我们确定 $\mathcal{A}_i$ 在 per($\mathcal{A}_i$) $\subseteq \mathbb{Z}/s\mathbb{Z}$上的最大周期长度。为此，我们注意到，$\mathcal{A}_i$ 的每个周期长度都是以下主分量的周期长度的最小公倍数：$\overline{\mathcal{A}}_{i,p} = \mathcal{A}_i \bmod p^{\nu_p(s)}$，其中 $p \mid s$ 且 $p \nmid \overline{\alpha}_i$。但根据 [13,表 3 和 4]（或我们的表 2），$\overline{\mathcal{A}}_{i,p}$ 的最大周期长度等于 $\overline{\mathcal{A}}_{i,p}$ 在 Sym($\mathbb{Z}/p^{\nu_p(s)}\mathbb{Z}$) 中的阶，也就是 $\overline{\mathcal{A}}_{i,p}$ 的所有周期长度的最小公倍数。因此，$\mathcal{A}_i$ 的最大周期长度等于 ord($\mathcal{A}_i \bmod s_i'$)，其中$s_i' = \prod_{p\mid s, p\mid\overline{\alpha}_i} p^{\nu_p(s)}$。我们按如下方式计算这个阶。

我们设定 $s_i' := 1$，并遍历整除 s 的 $O(\log q)$ 个素数 p（这些素数可从上文计算出的 s 的因式分解中直接读出）。对于每个 p，我们检查$p \mid \overline{\alpha}_i$是否成立；根据引理 5.1.5(3)，这需要 $O(\log^{1+o(1)} q)$ 次位运算。若成立，则跳到下一个 p；否则，我们将 $s_i' := s_i' \cdot p^{\nu_p(s)}$ 覆盖为新值，其中乘法需要 $O(\log^{1+o(1)} q)$ 次位运算（无需计算幂 $p^{\nu_p(s)}$，因为它是用于对 s 进行因式分解的 mdl 查询的输出之一）。结束对 p 的循环后，变量 s_i' 便具有所需的值；该循环的位操作成本为 $O(\log^{2+o(1)} q)$。现在，我们计算 $\overline{\alpha}_i' := \overline{\alpha}_i \bmod s_i'$ 和 $\overline{\beta}_i' := \overline{\beta}_i \bmod s_i'$，这需要 $O(\log^{1+o(1)} q)$ 次位运算，从而得到 $\mathbb{Z}/s_i'\mathbb{Z}$ 在模 s_i' 下的仿射映射 $\mathcal{A}_i' = \mathcal{A}_i$：$x \mapsto \overline{\alpha}_i' x + \overline{\beta}_i'$。我们希望计算 ord($\mathcal{A}_i'$)。为此，首先通过一个以 q 为界的 mord 查询，计算（乘法）阶 $\mathrm{ord}_{s_i'}(\overline{\alpha}_i')$。注意，$\mathrm{ord}_{s_i'}(\overline{\alpha}_i')$ 整除 ord($\mathcal{A}_i'$)，因为对于所有 $x \in \mathbb{Z}/s_i'\mathbb{Z}$ 和所有 $t \in \mathbb{Z}$，$(\mathcal{A}_i')^t(x) = (\overline{\alpha}_i')^t x + c(\overline{\alpha}_i', \overline{\beta}_i', t)$。因此，

$$\mathrm{ord}(\mathcal{A}_i') = \mathrm{ord}_{s_i'}(\overline{\alpha}_i') \cdot \mathrm{ord}((\mathcal{A}_i')^{\mathrm{ord}_{s_i'}(\overline{\alpha}_i')}).$$

但 $(\mathcal{A}_i')^{\mathrm{ord}_{s_i'}(\overline{\alpha}_i')}$ 是满足以下条件的平移 $x \mapsto x + \overline{\beta}_i''$

$$\overline{\beta}_i'' = \begin{cases} \overline{\beta}_i', & \text{if } \overline{\alpha}_i' = 1, \\ \overline{\beta}_i' \frac{(\overline{\alpha}_i')^{\mathrm{ord}_{s_i'}(\overline{\alpha}_i')} - 1}{\overline{\alpha}_i' - 1}, & \text{otherwise,} \end{cases}$$

其中，第二种情况下的公式需在环 $\mathbb{Z}$ 中求值，但结果应视为 $\mathbb{Z}/s_i'\mathbb{Z}$ 的一个元素。由于 ord($(\mathcal{A}_i')^{\mathrm{ord}_{s_i'}(\overline{\alpha}_i')}$) 等于 $\overline{\beta}_i''$ 模 s_i' 的加法阶，我们可以通过另外 $O(\log^{2+o(1)} q)$ 次位运算，求出 ord($\mathcal{A}_i'$)；它与 $\mathcal{A}_i$ 的最大周期长度相同（计算 $\overline{\beta}_i''$ ——这可能涉及幂计算——并通过最大公约数计算和除法求出其模 s_i' 的加法阶）。我们通过设定 $l_i := \ell \cdot \mathrm{ord}(\mathcal{A}_i')$ 来结束此循环；计算它需要 $O(\log^{1+o(1)} q)$ 次位运算，它是 f 在 $\bigcup_{t=0}^{\ell-1} C_{i_t}$ 中周期点上的最大周期长度。至此，我们完成了对 $(i,\ell) \in \overline{\mathcal{L}} \setminus \{(d,1)\}$ 的循环的描述；总体而言，其查询复杂度为 q-有界 $(d\log^{2+o(1)} q, 0, 0, d, 0)$，其中用到了 $\sum_{(i,\ell)\in\overline{\mathcal{L}}} \ell \le d$。

最后，f 的最大循环长度就是满足 $(i,\ell)\in\overline{\mathcal{L}}$的所有 l_i 中的最大值，其中 $l_d := 1$，计算该值需要 $O(d\log q)$ 位运算。 □

最后，我们给出定理 5.3.3.4 的以下推论，它可视为本小小节的主要结果。

推论5.3.3.6. 设 f_1 和 f_2 是 $\mathbb{F}_q$的广义分圆映射，指标分别为d_1 和 d_2，并设 $d := \mathsf{max}\{d_1, d_2\}$。此外，设 $L \in \mathbb{N}^+$，并分别以 L_1 和 L_2 表示 f_1和 f_2 在其周期点上的最大循环长度。那么，如果 $\mathsf{min}\{L_1, L_2\} \leq L$，则可以在 q-有界查询复杂度内决定 $\Gamma_{f_1} \cong \Gamma_{f_2}$

$$\begin{aligned}&(8^{d^2\,\mathrm{mpe}(q-1)+dL}2^{d\,\mathrm{mpe}(q-1)}(d^3L\,\mathrm{mpe}(q-1)+d^2L^2)\log^{1+o(1)}q\\&\quad+d^2 4^{d^2\,\mathrm{mpe}(q-1)+d}\log^2 q+d\log^{2+o(1)}q,\\&\quad d,1,d,0),\end{aligned}$$

因此，其拉斯维加斯双重复杂度在 q 范围内有界

$$\begin{aligned}&(8^{d^2\,\mathrm{mpe}(q-1)+dL}2^{d\,\mathrm{mpe}(q-1)}(d^3L\,\mathrm{mpe}(q-1)+d^2L^2)\log^{1+o(1)}q\\&\quad+d^2 4^{d^2\,\mathrm{mpe}(q-1)+d}\log^2 q+d\log^{2+o(1)}q+d\log^{7+o(1)}q,\\&\quad d\log^{3+o(1)}q, d\log q).\end{aligned}$$

证明。首先，我们计算 L_1 和 L_2，其查询复杂度在 q 范围内有界

$$(d\log^{2+o(1)}q + d^2\log^2 d, d, 1, d, 0)$$

根据命题 5.3.3.5。我们检查 $L_1 = L_2$ 是否成立，这需要 $O(\log q)$ 次位运算。如果不成立，则 $\Gamma_{f_1} \ncong \Gamma_{f_2}$，因此我们可以输出“false”并停止。否则，继续计算对于 $j = 1, 2,$

- 一个递归树描述列表 $\vec{\mathfrak{D}}_{(j)} = (\mathfrak{D}_{(j)n})_{n=0,1,\ldots,N_j}$，其中

$$N_j \in O(d2^{d^2\,\mathrm{mpe}(q-1)+d})$$

以及相关的有根树同构类型列表 $\vec{\mathfrak{J}}^{(j)}$；以及

- 树项链列表 $\mathfrak{N}_j$ 的 f_j 相对于 $\vec{\mathfrak{J}}_{(j)}$ 使得

$$|\mathfrak{N}_j| \in O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}).$$

根据定理 5.3.3.4，这可以在 q-有界查询复杂度内完成。

$$(8^{d^2\,\mathrm{mpe}(q-1)+dL}2^{d\,\mathrm{mpe}(q-1)}(d^3L\,\mathrm{mpe}(q-1)+d^2L^2)\log^{1+o(1)}q, d, 0, 0, 0).$$

接下来，我们按照定义 5.3.2.11 的含义，计算 $\vec{\mathfrak{D}}^{(1)}$ 和 $\vec{\mathfrak{D}}^{(2)}$的同步 $(\vec{\mathfrak{D}}^+, \mathfrak{i})$。根据引理 5.3.2.12，这需要

$$\begin{aligned}&O((d^3 8^{d^2\,\mathrm{mpe}(q-1)+d}+d^2 4^{d^2\,\mathrm{mpe}(q-1)+d}\log q)\cdot\log q)\\=\,&O(d^3 8^{d^2\,\mathrm{mpe}(q-1)+d}\log q+d^2 4^{d^2\,\mathrm{mpe}(q-1)+d}\log^2 q)\end{aligned}$$

位运算。接下来，我们使用循环等价类的字典序最小表示，覆盖 $\mathfrak{N}_2$ 中每个三元组的首项 $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$，替换为 $[\mathfrak{i}(\mathfrak{n}_1)\ \mathfrak{i}(\mathfrak{n}_2)\ldots, \mathfrak{i}(\mathfrak{n}_{l'})],$，从而得到修改后的、未排序的树项链列表 $\mathfrak{N}'_2$。这需要

$$O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}\cdot L^2\log L\log q) = O(dL^3\log L 2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$

总体位运算。之后，我们按字典序对 $\mathfrak{N}_2'$ 排序，这需要

$$O(dL^2(d^2\,\mathrm{mpe}(q-1)+dL)2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q)$$

位运算（另见定理5.3.3.4 证明的结尾）。最后，我们注意到 $\Gamma_{f_1}\cong\Gamma_{f_2}$ 当且仅当 $\mathfrak{N}_1=\mathfrak{N}_2'$，因此我们确定后者的真值；使用线性扫描，这需要 $O(dL^2 2^{d^2\mathrm{mpe}(q-1)^{+dL}}\log\ q)$ 次位运算。然后我们输出该真值并停止。

□

与前面的小小节一样，我们以讨论的算法伪代码结束本小小节，并注明每一步的 q 界查询复杂度（计算成本）。我们先介绍定理 5.3.3.4 中的算法：给定输入 (f,L)，其中 f 是 $\mathbb{F}_q$ 上的一个指标为 d 的广义分圆映射 f，且 f 的所有周期长度均不超过 L，该算法计算一对 $(\vec{\mathfrak{D}},\mathfrak{N})$，使得 $\vec{\mathfrak{D}}=(\mathfrak{D}_n)_{n=0,1,\ldots,N}$ 是一个递归树描述列表，其 $N\in O(d2^{d^2\mathrm{mpe}(q-1)^{+d}})$，关联序列 $\vec{\mathfrak{N}}$ 是有根树同构类型的序列 $\mathfrak{I}$，而 $\mathfrak{N}$ 是相对于 $\mathfrak{I}$ 的 f 的树项链列表，其 $|\mathfrak{N}|\in O(dL2^{d^2\mathrm{mpe}(q-1)^{+dL+d}})$。

1 计算 $\overline{f}$在 $\{0,1,\ldots,d\}$ 上的诱导函数以及仿射映射 A_i（关于$\mathbb{Z}/s\mathbb{Z}$）。计算成本：$(d\log^{1+o(1)} q,\ d,\ 0,\ 0,\ 0)$。

2 计算分区树寄存器

$$((\mathcal{Z}_i)_{i=0,1,\ldots,d-1},((\mathfrak{D}_n,(S_{n,i})_{i=0,1,\ldots,d})_{n=0,1,\ldots,N})$$

f的 $N\in O\left(d2^{d^2\,\mathrm{mpe}(q-1)^{+d}}\right)$。在此过程中，存储一个 $\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$、$\overline{f}$ 的循环，以及每个 $\overline{f}$ 周期的 $i\in\{0,1,\ldots,d-1\}$ 的参数 H_i。
计算成本: $(d^3\,\mathrm{mpe}(q-1)2^{(3d^2+d}{}_{)\mathrm{mpe}(q-1)^{+2d}}\log^{1+o(1)} q, 0, 0, 0, 0)$，因为 $\overline{f}$和该A_i 已经计算完成。

3 集合 $\vec{\mathfrak{D}}:=(\mathfrak{D}_n)_{n=0,1,\ldots,N}$.
计算成本：$(d^2 4^{d^2\,\mathrm{mpe}(q-1)^{+d}}\log\ q, 0,0,0,0)$.

4 对于每个 $(i,\ell)\in\overline{\mathcal{L}}\setminus\{(d,1)\}$，执行以下操作。
计算成本: $(8^{d^2\,\mathrm{mpe}(q-1)^{+dL}}(d^3L\,\mathrm{mpe}(q-1)\ +d^2L^2)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.1 对每个 $t=0,\ 1,\ldots,\ell-1$，执行以下操作。计算成本：$(d^3\,\mathrm{mpe}(q-1)\log\ q+d^2L\log^{1+o(1)} q, 0, 0, 0, 0)$。4.1.1 从 $\mathcal{Z}_{i_t}$ 中读取出一个针对 $\mathcal{U}_{i_t}$ 、长度为$H_i\le d\mathrm{mpe}(q-1)$ 的生成同余序列。计算成本：$(d\mathrm{mpe}(q-1)\log\ q, 0, 0, 0, 0)$。4.1.2 对每个 $j=0,\ 1,\ldots,H_i+L-1$，执行以下操作。计算成本：$(d^2\,\mathrm{mpe}(q-1)\log\ q+dL\log^{1+o(1)} q, 0, 0, 0, 0)$。4.1.2.1 如果 $j\le H_i$，则执行以下操作。4.1.2.1.1 从 $\mathcal{Z}_{i_{t+j}}$ 中读取出一个针对 $\lambda_{i_t}^{j}(\mathcal{R}_{i_t})$、长度为 $n_{i_t}\le d$ 的生成同余序列。计算成本：$(d\log\ q, 0, 0, 0, 0)$。

4.1.2.2 否则，执行以下操作。

4.1.2.2.1 计算.....的生成同余序列

$$\lambda_{i_t}^{j}(\mathcal{R}_{i_t}) = \lambda(\lambda_{i_t}^{j-1}(\mathcal{R}_{i_t}), A_{i_{t+j-1}})$$

长度为 $n_{i_t} \le d$。
计算成本: ($d\log^{1+o(1)} q$, 0, 0, 0, 0)。

4.2 对每个 $t = 0$、1、.....、$L-1$，执行以下操作。
计算成本:(($d^2 L\,\mathrm{mpe}(q-1) + dL^2)\log q$, 0, 0, 0, 0).

4.2.1 根据步骤 4.1 中存储的数据，拼接出一个生成同余式序列，用于

$$\mathcal{Q}_{i_{-t},H_i+L-t-1} = \bigwedge_{j=0}^{H_i+L-t-1} \lambda_{i_{-t-j}}^{j}(\mathcal{R}_{i_{-t-j}}) \wedge \mathcal{U}_{i_{-t}}.$$

计算成本:(($d^2\,\mathrm{mpe}(q-1) + dL)\log q$, 0, 0, 0, 0).

4.3 对每个 $t = 0$、1、. . .、$L-1$，执行以下操作。
计算成本: ($4^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1) + dL)\log^{1+o(1)} q$, 0, 0, 0, 0).

4.3.1集合 $\mathcal{O}_{i_{-t},L-t} := \emptyset$.计算成本:($\log d + \log L$, 0, 0, 0,0).4.3.2 对于每个 $\diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \in \{\emptyset, \neg\}^{n_{i_{-t}}+n_{i_{-t-1}}+\cdots+n_{i_{-H_i-L+1}}}$，执行以下操作。计算成本: ($4^{d^2\,\mathrm{mpe}(q-1)+dL}2^{-dt}(d^2\,\mathrm{mpe}(q-1) + dL)\log^{1+o(1)} q$, 0, 0, 0, 0)。

4.3.2.1 检查是否

$$\sigma_{\mathcal{Q}_{i_{-t},H_i+L-t-1},\mathbf{0}}(\diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \diamond \vec{\xi}_{i_{-t},H_i}, (\emptyset, \ldots, \emptyset)) > 0,$$

如果是这样，则将 $\diamond_{k=0}^{H_i+L-t-1}\vec{o_k}$ 添加到$\mathcal{O}_{i_{-t},\ L-t}$ 作为新元素。
QC: $(2^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1) + dL)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.4 对每个 $t = 0$、1、.....、$L-2$，执行以下操作。
计算成本: ($8^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1) + dL)\log^{1+o(1)} q$, 0, 0, 0, 0).

4.4.1 对每个 $\diamond_{k=0}^{H_i+L-t-1}\vec{o_k} \in \mathcal{O}_{i_{-t},L-t}$，执行以下操作。
计算成本: ($8^{d}{}^{2}{}_{\mathrm{mpe}(q-1)+dL}2^{-d(t+1)}2^{-dt}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q$, 0, 0, 0, 0).

4.4.1.1 对于每个 $\diamond_{k=0}^{H_i+L-t-2}\vec{o_k} \in \mathcal{O}_{i_{-t-1},\ L-t-1}$，执行以下操作。
计算成本：($4^{d^2\,\mathrm{mpe}(q-1)+dL}2^{-d(t+1)}(d^2\,\mathrm{mpe}(q-1)+dL)\log^{1+o(1)} q$, 0, 0, 0, 0)。

4.4.1.1.1 检查分布数是否

$$\sigma_{\mathcal{Q}_{i_{-t-1},H_i+L-t-2},A_{i_{-t-1}}}(\diamond_{k=0}^{H_i+L-t-2}\vec{o_k} \diamond \vec{\xi}_{i_{-t-1},H_i}, \diamond_{k=1}^{H_i+L-t-1}\vec{o_k} \diamond \vec{\xi}_{i_{-t},H_i})$$

等于 1。如果是，则设置

$$\mathfrak{u}_{i_{-t},L-t}(\diamond_{k=0}^{H_i+L-t-1}\vec{o_k}) := \diamond_{k=0}^{H_i+L-t-2}\vec{o_k}.$$

QC: $(2^{d^2\,\mathrm{mpe}(q-1)+dL}(d^2\,\mathrm{mpe}(q-1) + dL)\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.5 计算 $\mathcal{A}_i := A_{i_0} A_{i_1} \cdots A_{i_{\ell-1}}$。计算成本：(d log $^{1+o(1)}$ q, 0, 0, 0, 0)。4.6 设置 $\mathfrak{C}_{i,L} := \emptyset$。计算成本：(log d + log L,0, 0, 0, 0)。4.7 对每个 $l \in \{1, 2, \ldots, L\}$，执行以下操作。计算成本：($L \log^{1+o(1)} q$, 0, 0, 0, 0)。4.7.1 检查 $\ell \mid l$ 是否成立。如果不成立，则跳到下一个 l。计算成本：($\log^{1+o(1)} q$, 0, 0, 0, 0)。4.7.2 计算并存储仿射迭代

$$\mathcal{A}_{i,l} = \mathcal{A}_i^{l/\ell} = \begin{cases} \mathcal{A}_i, & \text{if } l = \ell, \\ \mathcal{A}_{i,l-\ell}\mathcal{A}_i, & \text{if } \ell \mid l > \ell, \end{cases}$$

线性系数 $\overline{\alpha}_{i,l}$ 和常数系数 $\overline{\beta}_{i,l}$。

计算成本：(log q, 0,0,0,0)，如果 $l = \ell$（只需复制步骤 4.5 中的信息）；否则为 ($\log^{1+o(q)} q$, 0, 0, 0, 0)。

4.7.3 检查最大公约数(s,$\overline{\alpha}_{i,l} - 1$)$\mid \overline{\beta}_{i,l}$。如果是，则将 l 添加到 $\mathfrak{C}_{i,L}$ 中作为新元素，并将 $\eta_{i,l}(x)$ 存储为 s-同余式

$$x \equiv -\frac{\overline{\beta}_{i,l}}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \cdot \operatorname{inv}_{\frac{s}{\gcd(s, \overline{\alpha}_{i,l}-1)}} \left(\frac{\overline{\alpha}_{i,l} - 1}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \right) \left(\operatorname{mod} \frac{s}{\gcd(s, \overline{\alpha}_{i,l} - 1)} \right).$$

计算成本：($\log^{1+o(1)} q$, 0,0,0,0)。

4.8 集合 $\mathfrak{N}_i := \emptyset$。

计算成本: (log d,0, 0, 0, 0)。

4.9 对每个 $\diamond_{k=0}^{H_i+L-1} \vec{o}_k \in \mathcal{O}_{i,\ L}$，执行以下操作。

计算成本: ($4^{d^2 \operatorname{mpe}(q-1)+dL} 2^L (d^2 L \operatorname{mpe}(q-1) + dL^2) \log^{1+o(1)} q$, 0, 0, 0, 0).

4.9.1 对于每个 $l \in \mathfrak{C}_{i,L}$，执行以下操作。

QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL+L}(d^2 L \operatorname{mpe}(q-1) + dL^2) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1.1 计算逻辑符号元组$\vec{\nu}_{i,\ L,\ l}$（参见命题 5.3.3.3 之前的段落）。

计算成本：($L \log^{1+o(1)} q$, 0, 0, 0, 0)。

4.9.1.2 设定

$$\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o}_k \diamond \vec{\nu}_{i,L,l}) := \sigma_{\mathcal{W}_{i,L},\mathbf{0}}(\diamond_{k=0}^{H_i+L-1} \vec{o}_k \diamond \vec{\xi}_{i,H_i} \diamond \vec{\nu}_{i,L,l}, (\emptyset, \ldots, \emptyset)).$$

QC: $(2^{d^2 \operatorname{mpe}(q-1)+dL+L}(d^2 \operatorname{mpe}(q-1) + dL) \log^{1+o(1)} q, 0, 0, 0, 0)$.

4.9.1.3 如果 $\mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o}_k \diamond \vec{\nu}_{i,\ L,\ l}) = 0$，则跳到下一个 l。

QC: $(d^2 \operatorname{mpe}(q-1) + dL, 0, 0, 0, 0)$.

4.9.1.4 对每个 $t = 0$、1、...、$l - 1$，执行以下操作。

计算成本：($dL^3 + d^2 L \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}} (q-1)^{+d} + L$ 日志 q, 0, 0, 0, 0).

4.9.1.4.1 Compute the logical sign tuple

$$(\mathrm{proj}_{i_{-t},L-t} \circ \mathfrak{u}_{i_{-t+1},L-t+1} \circ \mathfrak{u}_{i_{-t+2},L-t+2} \circ \cdots \circ \mathfrak{u}_{i_0,L})(\diamond_{k=0}^{H_i+L-1} \vec{o_k'}).$$

QC: $(d^2 L \operatorname{mpe}(q-1) + dL^2, 0, 0, 0, 0)$.

4.9.1.4.2 For each $n = 0, 1, \ldots, N$, do the following.
QC: $(d^2 \operatorname{mpe}(q-1) 2^{d^2 \operatorname{mpe}(q-1)+d} + \log q, 0, 0, 0, 0)$.

4.9.1.4.2.1 If the logical sign tuple computed in Step 4.9.1.4.1 is an element of S_{n,i_{-t},H_i} (the last entry of the tuple $S_{n,i_{-t}}$ from the partition-tree register computed in Step 2), then set $\mathfrak{n}^{(-t)} := n$ and skip to the next t.
QC: $(\max\{1, |S_{n,i_{-t},H_i}| d^2 \operatorname{mpe}(q-1)\}, 0, 0, 0, 0)$ if the condition is *not* satisfied; $(\max\{1, |S_{n,i_{-t},H_i}| d^2 \operatorname{mpe}(q-1)\} + \log q, 0, 0, 0, 0)$ if the condition *is* satisfied (the additional $O(\log q)$ bit operations are from copying the value of n; we do not need to process the $O(\log q)$-bit indices n during the loop over them, because we may jump to a neighboring address in memory in $O(1)$ bit operations).

4.9.1.5 Set $\vec{\mathfrak{n}} := (\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(0)})$.
QC: $(L \log q, 0, 0, 0, 0)$.

4.9.1.6 Set l' to be $\operatorname{minperl}(\vec{\mathfrak{n}})$.
QC: $(L^{1+o(1)} \log q, 0, 0, 0, 0)$.

4.9.1.7 Set $\vec{\mathfrak{n}}'$ to be the lexicographically minimal ordered sequence in the same cyclic equivalence class as $(\mathfrak{n}^{(-l+1)}, \mathfrak{n}^{(-l+2)}, \ldots, \mathfrak{n}^{(-l+l')})$.
QC: $(L^2 \log L \log q, 0, 0, 0, 0)$.

4.9.1.8 Add

$$(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}, [\vec{\mathfrak{n}}'], \mathfrak{m}_i(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}))$$

to $\mathfrak{N}_i$ as a new element.
QC: $(d^2 \operatorname{mpe}(q-1) + dL + L \log q, 0, 0, 0, 0)$.

5 集合 $\mathfrak{N} := \emptyset$。计
算成本：(1,0,0,0,0)。

6对于每个 $(i, \ell) \in \overline{\mathcal{L}} \setminus \{(d,1)\}$，执行以下操作。
计算成本: $(d^4 L^4 \operatorname{mpe}(q-1) 4^{d^2 \operatorname{mpe}(q-1)+dL} + d^2 L^2 4^{d^2 \operatorname{mpe}(q-1)^+ dL}$ 对数 q, 0,0,0,0).

6.1 对于每个 $(\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}, [\vec{\mathfrak{n}}], \mathfrak{m}') \in \mathfrak{N}_i$，执行以下操作。计算成本：($d^3 L^4 \operatorname{mpe}(q-1) 4^{d^2 \operatorname{mpe}(q-1)+dL} + dL^2 4^{d^2 \operatorname{mpe}(q-1)^+ dL} \log q$, 0, 0, 0, 0)。

6.1.1 从第一个条目 $\diamond_{k=0}^{H_i+L-1} \vec{o_k'} \diamond \vec{\nu}_{i,L,l}$，确定 l的二进制表示。计算成本：($L \log L$,0, 0, 0, 0)。

6.1.2 检查 $[\vec{\mathfrak{n}}]$ 和 l 是否已经作为 $\mathfrak{N}$ 中某个元素的前两项出现。如果是，则跳到 $\mathfrak{N}_i$ 的下一个元素。计算成本：($dL^2 2^{d^2 \operatorname{mpe}(q-1)^+ dL} \log q$, 0, 0, 0, 0)。

6.1.3 Compute the sum $\mathfrak{m}$ of the third entries of all triples in $\bigcup_{(j,\ell')\in\overline{\mathcal{L}}\setminus\{(d,1)\}}\mathfrak{N}_j$ that have the parameters l and $[\vec{\mathfrak{n}}]$ associated with them.
QC: $(dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

6.1.4 Add $([\vec{\mathfrak{n}}], l, \mathfrak{m})$ to $\mathfrak{N}$ as a new element.
QC: $(L\log q, 0, 0, 0, 0)$.

7 For each $n \in \{0, 1, \ldots, N\}$, do the following.
QC: $(d2^{d^2\,\mathrm{mpe}(q-1)+d} + \log q, 0, 0, 0, 0)$.

7.1 Check whether $S_{n,d} = \emptyset$. If so, set $\mathfrak{n} := n$ and exit the loop.
QC: $(\log q, 0, 0, 0, 0)$ if the condition is satisfied (which happens only once), $(1, 0, 0, 0, 0)$ otherwise.

8 Check whether $[\mathfrak{n}]$ and 1 already occur as the first two entries of some (unique) triple in $\mathfrak{N}$, and store this information (the truth value and, if applicable, the position of that triple in $\mathfrak{N}$).
QC: $(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

9 If $[\mathfrak{n}]$ and 1 occur as the first two entries of a triple $([\mathfrak{n}], 1, \mathfrak{m})$ in $\mathfrak{N}$, then do the following.

9.1 Overwrite the entry $([\mathfrak{n}], 1, \mathfrak{m})$ of the list $\mathfrak{N}$ with $([\mathfrak{n}], 1, \mathfrak{m}+1)$.
QC: $(\log q, 0, 0, 0, 0)$.

10 Else do the following.

10.1 Add $([\mathfrak{n}], 1, 1)$ to $\mathfrak{N}$ as a new element.
QC: $(L\log q, 0, 0, 0, 0)$.

11 Sort $\mathfrak{N}$ lexicographically.
QC: $(dL^2(d^2\,\mathrm{mpe}(q-1) + dL)2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

12 Output $(\vec{\mathfrak{D}}, \mathfrak{N})$ and halt.
QC: $((d^2 4^{d^2\,\mathrm{mpe}(q-1)+d} + dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL})\log q, 0, 0, 0, 0)$.

接下来，我们给出命题 5.3.3.5 中算法的伪代码。对于给定的指标 d 、 f 上的广义分圆映射 $\mathbb{F}_q$ ，该算法输出 f的最大循环长度。

1 计算 $\{0, 1, \ldots, d\}$ 上的诱导函数 $\overline{f}$以及$\mathbb{Z}/s\mathbb{Z}$的仿射映射 A_i。计算成本：$(d\log^{1+o(1)} q, d, 0, 0, 0)$。 **2** 计算 $\overline{f}$ 的 CRL 列表 $\overline{\mathcal{L}}$ 以及 $\overline{f}$的循环。计算成本：$(d^2\log^2 d, 0, 0, 0, 0)$。 **3** 对$s = p_1^{v_1}p_2^{v_2}\cdots p_K^{v_K}$进行因数分解。计算成本：$(\log q, 0, 1, 0, 0)$。 **4** 对每个 $(i, \ell) \in \overline{\mathcal{L}}\setminus\{(d, 1)\}$，执行以下操作。计算成本：$(d\log^{2+o(1)} q, 0, 0, d, 0)$。 **4.1** 计算 $\mathcal{A}_i = A_{i_0}A_{i_1}\cdots A_{i_{\ell-1}}$： $x \mapsto \overline{\alpha}_i x + \overline{\beta}_i$。计算成本：$(\ell\log^{1+o(1)} q, 0, 0, 0, 0)$。

4.2 Set $s'_i := 1$.
QC: $(\log d, 0, 0, 0, 0)$.

4.3 For each $j = 1, 2, \ldots, K$, do the following.
QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

4.3.1 Check whether $p_j \mid \overline{\alpha}_i$, and if so, skip to the next j.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.3.2 Set $s'_i := s'_i \cdot p_j^{v_j}$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.4 Set $\overline{\alpha}'_i := \overline{\alpha}_i \bmod s'_i$ and $\overline{\beta}'_i := \overline{\beta}_i \bmod s'_i$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.5 Compute $\operatorname{ord}_{s'_i}(\overline{\alpha}'_i)$.
QC: $(\log q, 0, 0, 1, 0)$.

4.6 Compute

$$\overline{\beta}''_i := \begin{cases} \overline{\beta}'_i, & \text{if } \overline{\alpha}'_i = 1, \\ \overline{\beta}'_i \frac{(\overline{\alpha}'_i)^{\operatorname{ord}_{s'_i}(\overline{\alpha}'_i)} - 1}{\overline{\alpha}'_i - 1}, & \text{otherwise.} \end{cases}$$

QC: $(\log^{2+o(1)} q, 0, 0, 0, 0)$.

4.7 Compute the additive order of $\overline{\beta}''_i$ modulo s'_i, which is equal to $s'_i / \gcd(s'_i, \overline{\beta}''_i)$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

4.8 Set $l_i := \ell \cdot \operatorname{ord}_{s'_i}(\overline{\alpha}'_i) \cdot s'_i / \gcd(s'_i, \overline{\beta}''_i)$.
QC: $(\log^{1+o(1)} q, 0, 0, 0, 0)$.

5 Set $L := 1$.
QC: $(\log d, 0, 0, 0, 0)$.

6 For each $(i, \ell) \in \overline{\mathcal{L}} \setminus \{(d, 1)\}$, do the following.
QC: $(d \log q, 0, 0, 0, 0)$.

6.1 If $l_i > L$, then set $L := l_i$.
QC: $(\log q, 0, 0, 0, 0)$.

7 Output L and halt.
QC: $(\log q, 0, 0, 0, 0)$.

最后，我们给出一个来自推论5.3.3.6 中算法的变体的伪代码。对于输入(L, f_1, f_2)，其中 $L \in \mathbb{N}^+$ and f_j，对于 $j = 1$，2，是 $\mathbb{F}_q$ 的指标为 d_j 的广义分圆映射；如果 f_1 的最大周期长度和 f_2 的最大周期长度都不至多为 L，该算法输出“失败”。否则，它输出有向图同构关系 $\Gamma_{f_1} \cong \Gamma_{f_2}$ 的真值。

1 对于 $j = 1,2$，计算 f_j 的最大周期长度 L_j。计算成本：$(d \log^{2+o(1)} q + d^2 \log^2 d, d, 1, d, 0)$。2 检查 $\min\{L_1, L_2\} \le L$，并存储此信息。计算成本：$(\log q, 0,0,0,0)$。3 如果 $\min\{L_1, L_2\} > L$，则执行以下操作。

3.1 输出“失败”并停止。计算成本：(1, 0, 0, 0, 0)。4 否则执行以下操作。4.1 检查 $L_1 = L_2$。如果不是，则输出“false”并停止。计算成本：(log q, 0, 0, 0, 0)。5 对于 $j = 1, 2$，计算

- 一个递归树描述列表 $\vec{\mathfrak{D}}_{(j)} = (\mathfrak{D}_{(j)n})_{n=0,1,\ldots,N_j}$ 以及相关的有根树同构类型序列 $\vec{\mathfrak{J}}_{(j)}$; 以及
$$N_j \in O(d2^{d^2\,\mathrm{mpe}(q-1)+d})$$

- the tree necklace list $\mathfrak{N}_j$ of f_j relative to $\vec{\mathfrak{J}}^{(j)}$, with
$$|\mathfrak{N}_j| \le O(dL2^{d^2\,\mathrm{mpe}(q-1)+dL}).$$

QC: $(8^{d^2\,\mathrm{mpe}(q-1)+dL}2^{d\,\mathrm{mpe}(q-1)}(d^3L\,\mathrm{mpe}(q-1)+d^2L^2)\log^{1+o(1)} q, d, 0, 0, 0)$.

6 Compute a synchronization $(\vec{\mathfrak{D}}^{+}, \mathfrak{i})$ of $\vec{\mathfrak{D}}^{(1)}$ and $\vec{\mathfrak{D}}^{(2)}$.
QC: $(d^3 8^{d^2\,\mathrm{mpe}(q-1)+d}\log q + d^2 4^{d^2\,\mathrm{mpe}(q-1)}\log^2 q, 0, 0, 0, 0)$.

7 Set $\mathfrak{N}_2' := \emptyset$.
QC: $(1, 0, 0, 0, 0)$.

8 For each $([\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}], l, \mathfrak{m}) \in \mathfrak{N}_2$, do the following.
QC: $(dL^3\log L 2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

8.1 Compute the lexicographically minimal representative of the cyclic equivalence class $[\mathfrak{n}_1, \mathfrak{n}_2, \ldots, \mathfrak{n}_{l'}]$, representing that class by it.
QC: $(L^2\log L\log q, 0, 0, 0, 0)$.

8.2 Add $([\mathfrak{i}(\mathfrak{n}_1)\ \mathfrak{i}(\mathfrak{n}_2)\ \ldots, \mathfrak{i}(\mathfrak{n}_{l'})], l, \mathfrak{m})$ to $\mathfrak{N}_2'$ as a new element.
QC: $(L\log q, 0, 0, 0, 0)$.

9 Sort $\mathfrak{N}_2'$ lexicographically.
QC: $(dL^2(d^2\,\mathrm{mpe}(q-1)+dL)2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

10 Check whether $\mathfrak{N}_1 = \mathfrak{N}_2'$ as sets, output the truth value of this equality, and halt.
QC: $(dL^2 2^{d^2\,\mathrm{mpe}(q-1)+dL}\log q, 0, 0, 0, 0)$.

6 个开放问题

我们以讨论与本文结果和方法相关的未决问题作为结尾。

6.1 mpe 和 τ 在素数幂上的渐近行为

命题 5.1.10指出，当 q 遍历所有素数幂的一个初始段时，mpe($q-1$)（ $q-1$的完全分解中素数的最大指数）的平均值有一个与该初始段无关的常数上界。这引出了一个重要观察：当 d 固定时，对于渐近意义下几乎所有有限域 $\mathbb{F}_q$，定理 5.1.9(2,3)中的复杂度都是log q的多项式。随后，在小节 5.1的末尾，我们提出了一个针对2的幂的类似问题；我们收集到的计算证据（以表 7的形式呈现）引出了以下猜想。

猜想 6.1.1. 当 k 遍历 $\mathbb{N}^+$ 的一个初始段时，mpe(2^k-1) 的平均值总是小于 2。形式上，该猜想断言，对于每个 $K \in \mathbb{N}^+$，都有

$$\frac{1}{K}\sum_{k=1}^{K} \operatorname{mpe}(2^k-1) < 2.$$

事实上，观察表 7，甚至可以猜想所述平均值始终小于 $3/2$，这意味着对于超过半数的所有 $k \in \mathbb{N}^+$，mpe(2^k-1) $=1$ （即 2^k-1 无平方因子）。

子小节 5.3.2 中的复杂度界也涉及 $\tau(q-1)$，即 $q-1$ 的除数个数。借助命题 5.3.1.4，我们能够证明，对于除渐近比例小于 ϵ 以外的所有素数幂 q，都有$\tau($ $q-1) < \log^{c''_\epsilon} q$，其中 c''_ϵ 是依赖于 ϵ 的适当常数。这意味着，对于固定的 d 和所有此类素数幂 q，子小节 5.3.2 中的复杂度界是 $\log q$ 的多项式，其次数依赖于 ϵ。在引出命题 5.3.1.4 的评论中，我们提到了狄利克雷的结果 [7, 定理 3.3], ，该结果意味着当 $x \to \infty$ 时，有搭配

$$\frac{1}{x}\sum_{n \le x} \tau(n) \sim \log x,$$

其中求和指标 n 遍历任意正整数（而不仅仅是形如 $q-1$ 、其中 q 为素数幂的数）。鉴于这一结果，我们提出以下问题，探讨命题 5.3.1.4 是否可以加强。

问题 6.1.2. 证明或证伪：当 $x \to \infty$ 时，有

$$\left(\sum_{q \le x} 1\right)^{-1} \sum_{q \le x} \tau(q-1) \in O(\log x),$$

w这里 q 遍历素数幂。如果事实证明这不成立，是否至少
c存在某个绝对常数 $c>0$，使得

$$\left(\sum_{q \le x} 1\right)^{-1} \sum_{q \le x} \tau(q-1) \in O(\log^c x)?$$

我们用下表中的计算证据来支持问题 6.1.2；这些证据是使用 GAP [22] 得到的。

x	$(\sum_{q\le x}1)^{-1}\sum_{q\le x}\tau(q-1)/\log x$ 四舍五入
10^3	1.56316
10^4	1.70011
10^5	1.74922
10^6	1.78351
10^7	1.8069
10^8	1.82506

表 9： $\tau(q-1)$ 的平均值与自然对数的关系。

6.2 算术划分的高效比较

我们回顾第 5 节开头提出的问题 3：给定 $\mathbb{F}_q$ 上的广义分圆映射 f，以及一对 (r,l)，其中 $r\in\mathbb{F}_q$ 是周期长度为 l 的 f-周期映射，任务是获得包含 r 的 Γ_f 所在连通分支的有向图同构类型的紧凑描述，并将其视为由有根树同构类型组成的项链。在问题 3 中，还假设给定了 f 的分区树寄存器（定义 5.1.2 意义下），这样我们就可以用寄存器中的编号 $n\in\{0,1,\dots,N\}$ 来指代这些有根树，而不必逐一完整描述它们。

假设 $r\neq 0_{\mathbb{F}_q}$（“$r=0_{\mathbb{F}_q}$”的情形可以单独轻松处理），我们在定理 5.1.9(3)的证明中为解决这一问题所采用的算法，其背后的主要思想是：将 r 的 f-循环上位于给定陪集 C_j中的位置，与 $\mathbb{Z}/(l/\ell)\mathbb{Z}$的元素相对应，其中 ℓ 是相关的陪集周期长度（即，对于满足 $r\in C_i$的唯一 $i\in\{0,1,\dots,d-1\}$，i 的 $\overline{f}$- 循环长度），并为满足条件的 $t\in\{0,1,\dots,\ell-1\}$ 推导出 $\mathbb{Z}/(l/\ell)\mathbb{Z}$ 的算术划分 $\mathcal{P}^{(i_t)}$，使得循环上 C_{i_t} 中属于 $\mathcal{P}^{(i_t)}$ 同一块的顶点，在 Γ_f中位于其上的树相同。形式化地说，我们可以通过标记函数 lab_{i_t}：$\mathcal{P}^{(i_t)}\to\{0,1,\dots,N\}$ 来表示这一点，该函数将 $\mathcal{P}^{(i_t)}$的每个块映射到其对应的有根树编号。

现在，以这种方式得到的包含 r 的 Γ_f 连通分支描述，并不能唯一编码其同构类型。也就是说，同构连通分量最终可能得到不同的描述，而要高效判定两个给定描述是否属于同一种同构类型，似乎是一个非平凡的计算问题。本小节旨在更详细地讨论这个开放问题，并将其归结为一些有待解答的具体问题。

当然，在我们针对问题 3 的算法的实际实现中，上文提到的每个算术划分 $\mathcal{P}^{(i_t)}$ 都通过一个长度为 $\overline{m}_{i_t}$（设为此值）的生成同余序列表示，而 lab_{i_t} 可以表示为一个函数$\{\emptyset,\neg\}^{\overline{m}_{i_t}}\to\{-1,0,1,\dots,N\}$；其中，若关联集合 $\mathcal{B}(\mathcal{P}^{(i_t)},\vec{v})$ 为空，则 -1 是要赋给逻辑符号元组 $\vec{v}$ 的虚设值。滥用记号时，我们也将此函数称为 lab_{i_t}。我们为形如 $(\mathcal{P}^{(i_t)},\mathsf{lab}_{i_t})$ 的有序对赋予一个特殊名称。

定义 6.2.1. 设 $m,N\in\mathbb{N}_0$ with $m>0$。$\mathbb{Z}/m\mathbb{Z}$ 的一个带 N 标签的算术划分是一个有序对 $(\mathfrak{P}(x\equiv\mathfrak{b}_j\pmod{\mathfrak{a}_j}:\ j=1,2,\dots,K),\mathsf{lab})$，由一个

对 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分，具有固定的生成同余序列和所谓的标记函数lab: $\{\emptyset, \neg\}^K \to \{-1, 0, 1, \ldots, N\}$，使得lab($\vec{\nu}$) $= -1$ 当且仅当 $\mathcal{B}(\mathcal{P}, \vec{\nu}) = \emptyset$。

为了刻画算法针对问题3 得到的两个紧凑描述何时表示同构连通分量，我们首先考虑 $\ell = 1$ 的特殊情形。此时，我们只需关注陪集 C_i 以及与之关联的 N标记的算术划分（$\mathcal{P}^{(i)}$，lab_i），该划分属于 $\mathbb{Z}/l\mathbb{Z}$。

我们需要理解，对应的有根树同构类型序列应用循环移位会如何影响带 N 标记的算术划分。令 $\mathfrak{t}$ 为 $\mathbb{Z}/l\mathbb{Z}$的平移 $x \mapsto x + 1$。它生成一个阶为 l 的循环子群 $\mathfrak{T}$，该子群是 $\mathbb{Z}/l\mathbb{Z}$ 在自身上的正则表示的像，即对称群($\mathbb{Z}/l\mathbb{Z}$)的子群。这个群 $\mathfrak{T}$还通过 $M^t := t(M) = \{y^t : y \in M\}$自然地作用于 $\mathbb{Z}/l\mathbb{Z}$的幂集。在同样的方式下，这导致了 $\mathfrak{T}$对 $\mathbb{Z}/l\mathbb{Z}$ 的幂集的幂集的作用（即，将 $\mathbb{Z}/l\mathbb{Z}$ 的子集族变换为其他此类子集族的作用），并且该作用限制在所有算术划分的集合上。事实上，如果 $\mathcal{P} = \mathfrak{P}(x \equiv \mathfrak{b}_j$(模 $\mathfrak{a}_j) : j = 1, 2, .., K)$是 $\mathbb{Z}/l\mathbb{Z}$的一个算术划分，那么 $\mathcal{P}^{\mathfrak{t}}$，即将 $\mathcal{P}$ 的所有块向右平移一个单位元所得到的 $\mathbb{Z}/l\mathbb{Z}$ 的划分，恰好是 $\mathbb{Z}/l\mathbb{Z}$的算术划分 $\mathfrak{P}(x \equiv \mathfrak{b}_j + 1$ (模 $\mathfrak{a}_j) : j = 1, 2, \ldots, K)$。

如果我们假设 $\mathcal{P}$ 被标记为 N，并且 $\mathcal{P}^t$ 的每个块（其中 $t \in \mathfrak{T}$）都带有与其移出的 $\mathcal{P}$ 块相同的标签，那么我们最终在 $\mathbb{Z}/l\mathbb{Z}$ 的带 N 标签的算术划分集合上得到 $\mathfrak{T}$ 的作用，这对我们的刻画很有用。形式上，该作用定义为

$$(\mathfrak{P}(x \equiv \mathfrak{b}_j \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K), \mathrm{lab})^{\mathfrak{t}^n} := (\mathfrak{P}(x \equiv \mathfrak{b}_j + n \pmod{\mathfrak{a}_j} : j = 1, 2, \ldots, K), \mathrm{lab}).$$

在我们当前讨论的“$\ell = 1$”特殊情形中，对与 N 标记的算术划分（$\mathcal{P}^{(i)}$，lab_i）相关联的有根树同构类型的循环序列进行 n 个单位的右循环移位，相当于将（$\mathcal{P}^{(i)}$，lab_i）替换为（$\mathcal{P}^{(i)}$，lab_i）$^{\mathfrak{t}^n}$。这启发了下面的定义。

定义6.2.2。设 m、$N \in \mathbb{N}_0$ 满足 $m > 0$，并设（$\mathcal{P}$,lab）和（$\mathcal{P}'$,lab′）是N标记的算术划分。这些划分是等价的，当且仅当存在一个 $t \in \mathfrak{T}$，使得（$\mathcal{P}'$, lab′）^{t} =（$\mathcal{P}$, lab）。在这种情况下，使得（$\mathcal{P}'$,lab′）$^{\mathfrak{t}^n}$ =（$\mathcal{P}$, lab）成立的最小正整数 n，称为（$\mathcal{P}$,lab）和（$\mathcal{P}'$,lab′）的平移数。如果（$\mathcal{P}$,lab）和（$\mathcal{P}'$, lab′）不等价，则定义它们的平移数为 ∞。

我们注意到，平移数对其两个自变量并不具有对称性。相反，如果 ($\mathcal{P}$, lab) 和 ($\mathcal{P}'$, lab′) 等价，那么 ($\mathcal{P}'$,lab′) 和 ($\mathcal{P}$, lab) 的平移数就是稳定子阶的差异

$$|\mathrm{Stab}_{\mathfrak{T}}((\mathcal{P}, \mathrm{lab}))| = |\{t \in \mathfrak{T} : (\mathcal{P}, \mathrm{lab})^t = (\mathcal{P}, \mathrm{lab})\}| = |\mathrm{Stab}_{\mathfrak{T}}((\mathcal{P}', \mathrm{lab}'))|$$

以及($\mathcal{P}$, lab)和 ($\mathcal{P}'$, lab′) 的平移数。例如，如果($\mathcal{P}$,lab)$^{\mathfrak{t}^3}$ =($\mathcal{P}$,lab) 且 ($\mathcal{P}'$, lab′)$^{\mathfrak{t}}$ =($\mathcal{P}$, lab)，那么 ($\mathcal{P}$, lab)$^{\mathfrak{t}^2}$ =($\mathcal{P}'$, lab′)。

正如我们在定义6.2.2 之前刚刚提到的，在 $\ell = 1$ 的情况下，形如 $(\mathcal{P}^{(i)}$,lab$_i)$和$(\mathcal{P}^{(j)}$,lab$_j)$ 的两个带有 N标记的算术划分，当且仅当它们等价时，才对应于 Γ_f 的同构连通分量。因此，我们提出以下算法问题。

问题 6.2.3。对于给定的 m、$N \in \mathbb{N}_0$，其中 $m > 0$ 和 N 标记的算术划分 $(\mathcal{P}, \mathsf{lab})$ 和 $(\mathcal{P}', \mathsf{lab}')$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的划分，求一种高效算法来计算 $(\mathcal{P}, \mathsf{lab})$ 与 $(\mathcal{P}', \mathsf{lab}')$ 的平移数。

请注意，根据我们对不等价的 N 标记算术划分的平移数所采用的约定，这样的算法尤其可以用于高效判断 $(\mathcal{P}$,lab$)$ 和 $(\mathcal{P}', \mathsf{lab}')$ 最初是否等价。此外，它还可以用于确定给定 N 标记算术划分 $(\mathcal{P}$,lab$)$ 在 $\mathfrak{T}$ 中的稳定子，因为该稳定子由 $\mathfrak{t}^n$ 生成，其中 n 是 $(\mathcal{P}$,lab$)$ 与其自身的平移数。

现在我们从“$\ell = 1$”回到一般情形。根据我们将 r 的 f 循环上的 l 个位置与 ℓ 个互不相交的 $\mathbb{Z}/(l/\ell)\mathbb{Z}$ 副本中的元素相对应的具体过程（见第 5.2.3 小小节），不难看出，一般而言，对与序列 $((\mathcal{P}^{(i_t)}$,lab$_{i_t}))_{t=0,\ 1,\ \ell-1}$ 相关联的有根树同构类型序列施加向右循环移位 n 个单位，相当于将该序列替换为

$$
\begin{aligned}
&(((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j}))_{j=0,1,\ldots,\ell-1})^{\mathfrak{t}^n} := \\
&((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j})^{\mathfrak{t}^{n''+1}})_{j=\ell-n',\ell-n'+1,\ldots,\ell-1} \diamond ((\mathcal{P}^{(i_j)}, \mathrm{lab}_{i_j})^{\mathfrak{t}^{n''}})_{j=0,1,\ldots,\ell-n'-1}
\end{aligned}
$$

其中 $n' := n$ 模 ℓ 以及 $n'' := (n - n')/\ell$。这定义了 $\mathfrak{T}$ 在所有长度为 ℓ 、由 N标记的 $\mathbb{Z}/(l/\ell)\mathbb{Z}$的算术划分序列集合上的作用；至于$\ell = 1$，如果两个这样的序列可以在此作用下相互映射，我们就称它们是等价的。为了普遍判定我们的算法针对问题 3 生成的两个描述是否对应于同构连通分量，我们需要判定这些描述在这种更一般的意义下是否等价。然而，事实证明，如果我们有一个如问题6.2.3中所述的算法，就可以高效地完成这一判定。下面解释其中的原因。

我们假设 $(\mathcal{X}_k, \mathsf{lab}^{(k)})_{k=0,1,\ldots,\ell-1}$ 和 $(\mathcal{Y}_k, \mathsf{Lab}^{(k)})_{k=0,1,\ldots,\ell-1}$ 是长度为 ℓ 的 N-标记算术划分序列，其中 $\mathbb{Z}/m\mathbb{Z}$ 是某个 $m \in \mathbb{N}^+$。当且仅当存在 $n' \in \{0, 1, \ldots, \ell - 1\}$ 和 $n'' \in \mathbb{Z}$，使得 $((\mathcal{Y}_k$,Lab$^{(k)}))_{k=0,1,\ldots,\ell-1}$ 等于

$$
((\mathcal{X}_t, \mathrm{lab}^{(t)})^{\mathfrak{t}^{n''+1}})_{t=\ell-n',\ell-n'+1,\ldots,\ell-1} \diamond ((\mathcal{X}_t, \mathrm{lab}^{(t)})^{\mathfrak{t}^{n''}})_{t=0,1,\ldots,\ell-n'-1}. \tag{33}
$$

为了检查是否如此，我们假设 n' 固定（在最坏情况下，我们需要尝试 n' 的 $\ell \in O(d)$ 个取值）。对于 $t = 0$，1，.....，$\ell - 1$，我们计算数量

$$
\mathfrak{b}_t := \begin{cases} \text{translation number of } (\mathcal{Y}_t, \mathrm{Lab}^{(t)}) \text{ and } (\mathcal{X}_{\ell-n'+t}, \mathrm{lab}^{(\ell-n'+t)}), & \text{if } t < n', \\ \text{translation number of } (\mathcal{Y}_t, \mathrm{Lab}^{(t)}) \text{ and } (\mathcal{X}_{t-n'}, \mathrm{lab}^{(t-n')}), & \text{otherwise.} \end{cases}
$$

使用问题6.2.3中的算法。如果这些数中的任何一个是 ∞，那么所选的 n' 值就不起作用。否则，我们使用上述算法计算 $\mathfrak{a}_t$，即 $\mathfrak{T}$ 中($\mathcal{Y}_t$,Lab$^{(t)}$)的稳定子的群阶（这里 $\mathfrak{T}$ 是一个阶为 m的循环群）。问题是是否存在 $n'' \in \mathbb{Z}$ 使得

$$\begin{aligned} n''+1 &\equiv \mathfrak{b}_0 \pmod{\mathfrak{a}_0} \\ n''+1 &\equiv \mathfrak{b}_1 \pmod{\mathfrak{a}_1} \\ &\vdots \\ n''+1 &\equiv \mathfrak{b}_{n'-1} \pmod{\mathfrak{a}_{n'-1}} \\ n'' &\equiv \mathfrak{b}_{n'} \pmod{\mathfrak{a}_{n'}} \\ n'' &\equiv \mathfrak{b}_{n'+1} \pmod{\mathfrak{a}_{n'+1}} \\ &\vdots \\ n'' &\equiv \mathfrak{b}_{\ell-1} \pmod{\mathfrak{a}_{\ell-1}} \end{aligned}$$

因为这些同余式刻画了何时 (($\mathcal{Y}_t$,Lab$^{(t)}$))$_{t=0,\ 1,\ \ldots \ell-1}$ 等于 (33)。将其视为关于单个变量 n'' 的 m 个同余式组成的方程组，可利用命题 2.2.1 轻松判断 n'' 是否存在。

我们通过指出以下事实来结束本小节：问题6.2.3 中的算法可用于判定两个（带 N 标记的）算术划分是否相等，即是否具有相同的（带标记的）块。这是因为，$\mathbb{Z}/m\mathbb{Z}$ 的两个带 N标记的算术划分（$\mathcal{P}$, lab）和（$\mathcal{P}'$, lab′）相等，当且仅当（$\mathcal{P}$, lab）与（$\mathcal{P}'$,lab′）的平移数等于（$\mathcal{P}$, lab）与其自身的平移数。此外，$\mathcal{P}$ 和 $\mathcal{P}'$ 相等，当且仅当（$\mathcal{P}$,0）和（$\mathcal{P}'$,0）相等，其中（在这两种情形中）0表示恒为零的标记函数。不过，验证给定的两个 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分是否相等这一算法问题本身就很有趣，即使问题6.2.3 无法解决，它也可能有高效算法解决方案，因此我们将其单独提出。

问题 6.2.4。寻找一种高效算法，对于给定的 $m \in \mathbb{N}^+$ 和（生成 m-同余式序列的）算术划分 $\mathcal{P}$ 和 $\mathcal{Q}$ of $\mathbb{Z}/m\mathbb{Z}$，判定是否$\mathcal{P} = \mathcal{Q}$。

6.3 关于渐近增长率的更多问题

设 $m = p_1^{v_1} \cdots p_K^{v_K}$ 是一个已显示其因式分解的正整数。根据定义 1.5(3)，最小的、覆盖 $\mathcal{P}$ 的 m-同余式数量称为 $\mathcal{P}$ 的（算术）复杂度，记作 AC($\mathcal{P}$)。在注释 3.3.6 中，我们观察到，$\mathbb{Z}/m\mathbb{Z}$ 的平凡划分 $\mathcal{T}_m$ （其所有块都是单元素集）满足 AC($\mathcal{T}_m$) $\leq \sum_{j=1}^{K} p_j^{v_j} - K$。当 m 是素数幂时，这个界等于 $m-1$；我们还在注释 3.3.6 中观察到，当 m 是前 primorial 数时，这个界渐近等价于 $\log^2 m/(2 \log \log m)$。由此引出一个问题：对于 m 的某个合适的无限值类，$\mathcal{T}_m$ 的实际复杂度是否可能显著小于该界?

问题 6.3.1. 以下命题是否成立：

$$\liminf_{m\to\infty} \frac{\mathrm{AC}(\mathcal{T}_m)}{\log^2 m/\log\log m} > 0?$$

我们观察到，$\mathcal{T}_m$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的唯一一种能够达到最大可能块数 m的（算术）划分。更一般地，我们可以探讨是否存在一些非平凡界，能够将 $\mathbb{Z}/m\mathbb{Z}$的算术划分 $\mathcal{P}$ 的算术复杂度与其块数联系起来。显然，$\mathcal{P}$ 的不同块数至多为$2^{\mathrm{AC}(\mathcal{P})}$，且当 AC($\mathcal{P}$) $\in \{0,1\}$时，这个界可以达到。事实上，对于任意给定的值$k \in \mathbb{N}_0$，都存在一个 $m \in \mathbb{N}^+$ 以及 $\mathbb{Z}/m\mathbb{Z}$的一个算术划分 $\mathcal{P}$，使得 AC($\mathcal{P}$) $= k$ 且 $\mathcal{P}$ 有 2^k 个不同的块：只需令 m 为第 k个前 primorial 数 $p_k\# = p_1 p_2 \cdots p_k$，其中$p_j$表示第 j小的素数，并令 $\mathcal{P} := \mathfrak{P}(x \equiv 0 \pmod{p_j} : j = 1, 2, \ldots, k)$即可。然而，当 AC($\mathcal{P}$) 相对于 m足够大时，$\mathcal{P}$的块数就会低于 $2^{\mathrm{AC}(\mathcal{P})}$；最迟在 AC($\mathcal{P}$) $> \log_2 m$ 时便会如此，因为 $\mathcal{P}$ 的块数不可能大于 m。在刚才给出的例子中，$m = p_k\#$，我们有 AC($\mathcal{P}$) $= k \sim (\log m/\log\log m)$，这引出了以下未解决问题。

问题 6.3.2. 找出函数 f, g：$[0,\infty) \to [0,\infty)$，使得

(1) (x/对数 x/对数 对数 x) $\lesssim f(x) \in o$(对数 x), (2) $g(x) \in o(2^x)$,以及(3)对于每个正整数 m 和每个关于 $\mathbb{Z}/m\mathbb{Z}$的算术划分 $\mathcal{P}$，若AC($\mathcal{P}$) $\geq f(m)$，则 $\mathcal{P}$ 的块数量至多为 g(AC($\mathcal{P}$)),

或者证明这类函数不存在。

在子小节 5.3.2 中，我们注意到，使用树项链列表来构造一个通用的高效算法，以判定给定的两个 $\mathbb{F}_q$ 上的广义分圆映射的函数图是否同构，可能会遇到一个障碍。也就是说，即使它们的指标固定，广义分圆映射 f 的函数图中连通分支的不同同构类型也可能过多。具体而言，我们提出以下问题。

问题 6.3.3。证明或反驳：对于每个 $d \in \mathbb{N}^+$，存在一个常数 $c = c(d)$，使得对于每个素数幂 q 和每个指标 d，f 的广义分圆映射 $\mathbb{F}_q$的连通分支的不同同构类型数量属于 $O(\log^c q)$。

我们观察到，对于 $d = 1$，非零 f-周期点之上的所有有根树都是同构的；参见定理 2.1.5，并注意到 $\Gamma_{f_{|\mathbb{F}_q^*}} \cong \Gamma_{A_0}$，除非 f 恒为零，在这种情况下，所述命题显然成立。因此，对于 $d = 1$，问题6.3.3等价于证明或否定：当 q遍历所有素数幂时，$\mathbb{Z}/(q-1)\mathbb{Z}$的仿射映射的不同周期长度的数量是否有某个固定多项式（以 $\log q$ 为变量）作为上界。尽管有注记 5.3.2.4，这甚至似乎仍是一个未解决的问题。

鉴于命题5.3.1.4，我们接受这样一个事实：为了使来自子小节5.3.2 的算法高效，需要排除素数幂 q 中任意小但为正的渐近比例。在此背景下，我们注意到以下问题，该问题比问题 6.3.3 更难，但也更有趣。

问题 6.3.4. 对于 $d \in \mathbb{N}^+$ 和 $c > 0$，我们用 $\epsilon(d, c)$ 表示满足 $d \mid q-1$ 的所有素数幂 q 中，存在一个指标为 d 的 $\mathbb{F}_q$ 上的广义分圆映射 f，使得 Γ_f 的连通分量具有多于 $\log^c q$ 种不同同构类型的渐近比例。证明或否证：当 d 固定且 $c \to \infty$ 时，有 $\epsilon(d, c) \to 0$。

6.4 扩展到其他陪集仿射函数

一个指标为 d 的 f $\mathbb{F}_q$广义分圆映射，以分圆形式 (1) 给出，若所有 a_i 和 r_i 都非零，则限制为一个函数 $\mathbb{F}_q^* \to \mathbb{F}_q^*$。该函数具有“逐陪集仿射”的性质：它在指标为 d的$\mathbb{F}_q^*$ 子群 C 的任意给定陪集 C_i 上的限制，会通过C 循环群的一个仿射映射映到另一个陪集 $C_{\overline{f}(i)}$ （此处我们采用“仿射映射”一词的一般群论含义，见定义 2.1.15）。我们可以将 f 拆分为更小且易于处理的部分，这对于理解本文所介绍的 Γ_f 方法至关重要。

在本小节中，我们旨在推广这一想法。更具体地说，我们将$\mathbb{F}_q^*$ 替换为某个群 G（通常是有限群，但不一定），并将 C 替换为G的一个子群 H。我们考虑以下两种陪集仿射函数的概念。

定义6.4.1. 设 G 为一个群，且设 H 为 G的子群。

(1) 仿射函数 $H \to G$ 是形如 $h \mapsto h^{\varphi} b$ 的函数 $H \to G$，其中 $\varphi : H \to G$ 是群同态，且 $b \in G$是某个元素。(2) 如果对于 G中 H 的每个右陪集 $C = Hr_C$，都存在一个仿射函数 $A_C : H \to G$，使得对于所有 $h \in H$， $f(hr_C) = A_C(h)$，则称函数 $f\colon G \to G$ 是广义的 H-陪集逐块仿射函数。(3) 如果对于 G中 H 的每个右陪集 $C = Hr_C$，都存在 H 的一个仿射映射 A_C 和一个$t_C \in G$，使得对于所有 $h \in H$， $f(hr_C) = A_C(h)t_C$，则称函数 $f : G \to G$ 是狭义的 H-陪集逐块仿射函数。

与狭义的 H-陪集仿射函数相比，广义的 H-陪集仿射函数不必将 H 的每个右陪集映射到单个这样的陪集。这使得研究广义的 H-陪集仿射函数在迭代下的行为变得困难。其中最著名的例子是考拉兹函数 g，对应于 $G = \mathbb{Z}$ 和 $H = 2\mathbb{Z}$，其陪集仿射公式为

$$g(x) = \begin{cases} x/2, & \text{if } x \in 2\mathbb{Z}, \\ 3x+1, & \text{if } x \in 2\mathbb{Z}+1. \end{cases}$$

另一方面，任意函数 $g : \mathbb{Z} \to \mathbb{Z}$使得 g 在指标为 n 的子群 $n\mathbb{Z}$的每个陪集$k + n\mathbb{Z}$ 上，与一个仿射函数 $x \mapsto a^{(k)}x + b^{(k)}$ 一致，该函数具有

整数系数 $a^{(k)}$ 和 $b^{(k)}$，是 $n\mathbb{Z}$-陪集仿射函数的狭义形式，因此适用于本小节第一段中提到的思想。今后，我们将把注意力限制在 H-陪集仿射函数的狭义形式上，为简便起见，我们简称其为 H-陪集仿射函数。

一个重要的特殊情形是 $G = (\mathbb{F}_q, +)$，且 H 是 $\mathbb{F}_p$ 的G 子空间；对于这种函数类，[12] 已经对其进行了研究。我们预计，用于理解广义分圆映射的函数图的方法，对于 H-陪集仿射函数也基本能够以类似方式奏效，但有一个重要的注意事项：在引理2.2.2（我们的“主引理”）的证明中，我们实质性地使用了命题2.2.1中陈述（1）和（3）的等价性。在当前小节更一般的群论背景下，这种等价性需要由群 H 的以下性质所取代。

定义6.4.2.令 H 为一个群。我们称 H 是两两同余式相容的，如果在 H上的任意给定同余方程组，

$$\begin{aligned} x &\equiv h_1 \pmod{N_1} \\ x &\equiv h_2 \pmod{N_2} \\ &\vdots \\ x &\equiv h_K \pmod{N_K} \end{aligned}$$

其中 $h_1, h_2, \ldots, h_K \in H$和 $N_1, N_2, \ldots, N_K$是 H 的正规子群，当且仅当系统中的每一对同余式都相容。

命题 2.2.1 中陈述 (1) 和 (3) 的等价性可以改述为“有限循环群是两两同余一致的”。在上文提到的特殊情形“$G = (\mathbb{F}_q, +)$”中，群 H 具有 $\mathbb{F}_p^n$ 的形式，即它是一个有限初等阿贝尔 p-群。如果我们的方法要在该情形下完全类似地进行，就需要有限初等阿贝尔群是两两同余一致的。然而，事实并非如此，正如下面的结果所示（注意，所有阿贝尔群都是幂零的）。

定理 6.4.3. 设 G 是一个有限幂零群。以下命题等价：

(1) G两两同余一致。(2) G是循环的。

我们将在本小节末尾证明定理 6.4.3。在此之前，我们再补充两点说明。

首先，我们注意到，存在非循环的有限两两同余一致群；其中既有不可解的例子，例如任意非阿贝尔有限单群（这是显而易见的），也有可解的例子，例如对于任意素数幂 q，$\mathrm{AGL}_1(q) = \mathbb{F}_q \rtimes \mathbb{F}_q^*$。要说明后一类群两两同余一致，我们注意到，$\mathrm{AGL}_1(q)$ 有唯一的非平凡极小正规子群 $\mathbb{F}_q$，因此，$\mathrm{AGL}_1(q)$ 上的任意同余式都对应于循环群 $\mathbb{F}_q^*$ 上的一个同余式（即经典数论意义上的同余式），使得 $\mathrm{AGL}_1(q)$ 上该同余式的解集是

规范投影 $\mathrm{AGL}_1(q) \to \mathbb{F}_q^*$ 下，对应的数论同余式解集的原像。特别地，如果给定的 $\mathrm{AGL}_1(q)$ 上的同余方程组中任意两个同余式都相容，那么对应的 $\mathbb{F}_q^*$ 上的同余方程组也满足这一条件，因此根据命题 2.2.1，该方程组相容；由此，原来的 $\mathrm{AGL}_1(q)$ 上的方程组也必定有解。

其次，我们提出以下两个开放问题，它们的提出动机源于定理 6.4.3 以及引出该定理的讨论。

问题 6.4.4。分类所有两两同余一致的有限群。

问题 6.4.5. 对于不属于有限两两同余一致群类的重要有限群类（例如有限（初等）阿贝尔群类），设计高效算法，判定该类中的群上给定的同余方程组是否一致。

为了证明定理 6.4.3，我们先考察群的以下性质。

定义6.4.6. 设 G 是一个群。若 G 具有两两陪集相交性质（简称 PCIP），则称其满足以下条件：对于任意正整数 m 以及 G 的子群的（左或右）陪集序列 $(C_1, C_2, \ldots, C_m)$，如果对于所有 $1 \le j < k \le m$ 都有 $C_j \cap C_k \neq \emptyset$，那么 $\bigcap_{j=1}^{m} C_j \neq \emptyset$。满足 PCIP 的群也简称为 PCIP 群。

以下命题可由以下观察直接得到：在群 G 上，同余式 $x \equiv g$(模 N) 的解集是 N 的陪集 $gN = Ng$。

命题 6.4.7. 设 G 为一个群。以下条件等价。

(1) G是两两同余一致的。(2) 对任意正整数 m 以及 G的正规子群的陪集序列 $(C_1, C_2, \ldots, C_m)$，如果对于所有 $1 \le j < k \le m$ 都有 $C_j \cap C_k \neq \emptyset$，那么 $\bigcap_{j=1}^{m} C_j \neq \emptyset$。

命题 6.4.7 有两个重要推论。

推论 6.4.8。以下结论成立。

(1) 每个 PCIP 群都是两两同余一致的。(2) 阿贝尔群满足 PCIP，当且仅当它是两两同余一致的。

根据推论 6.4.8，以下结果证明了定理 6.4.3 对阿贝尔群成立。

定理 6.4.9。设 G 是一个有限群。以下命题等价。

(1) G满足 PCIP。(2) G
是循环的。

证明。“(2) $\Rightarrow$ (1)”由命题 2.2.1 得出，因此我们重点讨论“(1)$\Rightarrow$ (2)”。

不难证明，PCIP群的所有子群和商群本身都是 PCIP群。因此，对于蕴含关系“(1) $\Rightarrow$ (2)”而言，一个极小反例将是一个有限的非循环群 G，其所有真子群都是循环群。这些群由 Miller 和 Moreno [49] 分类如下。

(1) $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z}$，其中 p 是素数；(2)四元数群 Q_8；或(3) 后循环群 $\mathbb{Z}/q^n\mathbb{Z} \ltimes \mathbb{Z}/p\mathbb{Z} = \langle x, y : \ x^p = y^{q^n} = 1, y^{-1}xy = x^r \rangle$，其中$r \equiv 1 \pmod{q}$且 $r^q \equiv 1 \pmod{p}$，但 $r \not\equiv 1 \pmod{p}$，因为否则该群是循环的或同构于 $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/p\mathbb{Z}$。

只需证明这些群中没有一个满足 PCIP。为此，我们现在为 $j = 1$、2、3 分别指定三个子集 $C_j \subseteq G$；其中每个子集都是 G 中某个子群的左陪集和右陪集，并且这三个子集两两相交，而 $\bigcap_{j=1}^{3} C_j = \emptyset$。

- 对于第一种群，其中的元素是形如（x， y）的有序对，且 x、$y \in \mathbb{F}_p$，令

$$C_1 := \{(x, y) \in \mathbb{F}_p^2 : x = 1\}, C_2 := \{(x, y) \in \mathbb{F}_p^2 : y = 1\}, \text{ and}$$
$$C_3 := \{(x, y) \in \mathbb{F}_p^2 : x + y = 1\}.$$

- 对于 $Q_8 = \{\pm 1, \pm i, \pm j, \pm k\}$，令

$$C_1 := \langle i \rangle j = j\langle i \rangle = \{\pm j, \pm k\}, C_2 := \langle j \rangle i = i\langle j \rangle = \{\pm i, \pm k\}, \text{ and}$$
$$C_3 := \langle k \rangle j = j\langle k \rangle = \{\pm i, \pm j\}.$$

- 我们注意到，第三种群的中心商群具有相同的形式，但其中 $n = 1$；因此不妨作此假设。令

$$C_1 = \langle x \rangle y = y\langle x \rangle = \{y, yx, yx^2, \ldots, yx^{p-1}\}, C_2 = \langle y \rangle = \{1, y, y^2, \ldots, y^{q-1}\},$$
$$\text{and } C_3 = \langle yx \rangle = \{1, yx, y^2x^{1+r}, y^3x^{1+r+r^2}, \ldots, y^{q-1}x^{1+r+r^2+\cdots+r^{q-2}}\}.$$

由于 C_1 的所有元素中 y—指数均为 1，因此有 $C_1 \cap C_2 = \{y\}$ 和$C_1 \cap C_3 = \{yx\}$。此外， $C_2 \cap C_3 = \{1\}$，因为 C_2 和 C_3 是阶为素数的不同子群。于是得到 $C_1 \cap C_2 \cap C_3 = \emptyset$，如所要求。

□

定理 6.4.3 的证明。与定理 6.4.9 的证明一样，根据命题 2.2.1，蕴含“(2) $\Rightarrow$(1)”显然成立，因此我们着重证明“(1) $\Rightarrow$ (2)”。

设 G 是一个两两同余一致的有限幂零群。若 G 是阿贝尔群，则由推论 6.4.8(2) 和定理 6.4.9 可知 G 必为循环群，因此我们反设 G 非阿贝尔。于是，对某个素数 p， G 的（唯一）Sylow p 子群 S_p 非阿贝尔。由于 G 是其 Sylow 子群的直积，我们可知 S_p 是 G 的一个商群。此外，不难

证明，两两同余一致群的商群本身也两两同余一致。因此， S_p 是两两同余一致的。

但是 S_p 是一个非阿贝尔有限 p-群，因此 Burnside 基定理表明，其 Frattini 商群 $S_p/\Phi(S_p)$ 是有限初等阿贝尔 p-群 $\mathbb{F}_p^n$，其中 $n > 1$ 是 S_p 的生成集的最小大小。再次利用两两同余一致这一性质在取商群时保持不变，可得 $\mathbb{F}_p^n$ 是两两同余一致的。由于 $\mathbb{F}_p^n$ 是阿贝尔的，推论 6.4.8 因而表明 $\mathbb{F}_p^n$ 满足 PCIP，这与定理 6.4.9 矛盾。 □

6.5 推广到变换图

本文研究函数图，它们是集合上各个函数的自然可视化形式，有助于理解离散动力系统的长期行为。作为一种推广，可以考虑这样的情形：动力系统并非确定性地演化，而是对于某些$n \in \mathbb{N}^+$，每个系统状态 x 有 n 种后继状态的可能性（可能存在重复），这些可能性以不同的概率发生，并由函数$g_j\colon\ X \to X$的值$g_1(x)$ $g_2(x)$. .. , $g_n(x)$表示。令 $\mathcal{G}\colon = \{g_1, g_2, \ldots, g_n\}$。研究此类系统行为的第一步，是理解所谓的变换图 TRAG$(X, \mathcal{G})$。它被定义为一个顶点集为 X 的边标记有向图，其弧的形式为 $x \xrightarrow{g_j} g_j(x)$，其中 $x \in X$ 和 $j = 1$，2，.. . , n。术语“变换图”源自 Annexstein、Baumslag 和Rosenberg 的论文 [6]。这一概念还与作用密切相关，作用是半群在集合上的作用 [17, 第 11节1] （类比于 [6], 中的术语“群作用图”，也可以将变换图称为“作用图”），并且与确定性有限自动机 [33, 第 2.2 小节1]密切相关。事实上，TRAG$(X, \mathcal{G})$类似于一个状态集为 X 、输入符号集为 $\mathcal{G}$的确定性有限自动机，但没有声明起始状态和接受状态。

我们注意到，除边标签外，TRAG$(X, \{g\})$与我们记号中的函数图 Γ_g 相同。正如 [6, 第 2.1],小节开头所指出的，对于给定的 x、$x' \in X$，也可以考虑TRAG$(X, \mathcal{G})$的简单（即没有重弧 $x \to x'$ ）无标签版本，我们将其记为 STRAG$(X, \mathcal{G})$。当然，对于给定的集合 X，任何顶点集为 X 且每个顶点都具有正出度（可能包括 ∞）的有向图，都是某个适当的非空$\mathcal{G} \subseteq X^X$所对应的 STRAG$(X, \mathcal{G})$ 形式。

探讨本文中发展的方法是否能够扩展到处理形如TRAG$(\mathbb{F}_q, \mathcal{F})$ 和STRAG$(\mathbb{F}_q, \mathcal{F})$ 的图将会很有意义，其中 $\mathcal{F}$ 是 $\mathbb{F}_q$的广义分圆映射集合，假设它们具有一个公共的小指标 d （这也涵盖了一些指标并不统一的情形，因为如果对于 f_j，2，. . .， n，$j = 1$是 $\mathbb{F}_q$ 上指标为 d_j的广义分圆映射，那么每个 f_j的指标也都是 lcm$(d_1, d_2, \ldots, d_n)$ ）。具体而言，我们提出以下问题。

问题 6.5.1. 对于给定的素数幂 q 以及指标为 d 的 $\mathbb{F}_q$ 的广义分圆映射集合 $\mathcal{F}$，设计高效算法（例如，当 d 固定时，运行时间关于 log q 为多项式），以求得

(1) 通过代表顶点对 TRAG($\mathbb{F}_q, \mathcal{F}$) 的连通分量（等价地，对 STRAG($\mathbb{F}_q, \mathcal{F}$)）进行紧凑参数化；以及 (2) 给出由第 (1) 点参数化映射的像中的一个顶点时，分别紧凑描述 TRAG($\mathbb{F}_q, \mathcal{F}$) 的一个连通分量以及 STRAG($\mathbb{F}_q, \mathcal{F}$) 的一个连通分量的同构类型。

据作者所知，即使对于 $d = 1$ 和 $|\mathcal{F}| = 2$而言，这仍是一个未解决的问题（即，考虑每个都基于 $\mathbb{F}_q$上的两个单项式函数的变换图时）。

问题 6.5.2。对于指标为 d 的广义分圆映射的某些集合类别，设计高效算法，以分别判定对于该类别中的集合 $\mathcal{F}_1$、$\mathcal{F}_2$，TRAG($\mathbb{F}_q, \mathcal{F}_1$) $\cong$ TRAG($\mathbb{F}_q, \mathcal{F}_2$)，以及 STRAG($\mathbb{F}_q, \mathcal{F}_1$) $\cong$ STRAG($\mathbb{F}_q, \mathcal{F}_2$)。

参考文献

[1] M. Agrawal, *CS681: Computational Number Theory and Algebra. Lecture 1 & 2: Integer and Modular Arithmetic*, lecture notes (2009), `https://www.cse.iitk.ac.in/users/manindra/CS681/lecture1and2.pdf`.

[2] M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P, *Ann. of Math. (2)* **160**(2): 781–793, 2004.

[3] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error rate, *SIAM J. Comput.* **38**(4): 1207–1282, 2008.

[4] S. Ahmad, Cycle Structure of Automorphisms of Finite Cyclic Groups, *J. Combin. Theory* **6**: 370–374, 1969.

[5] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley (Addison-Wesley Series in Computer Science and Information Processing), Reading, 2nd printing 1975.

[6] F. Annexstein, M. Baumslag and A.L. Rosenberg, Group action graphs and parallel architectures, *SIAM J. Comput.* **19**(3): 544–569, 1990.

[7] T.M. Apostol, *Introduction to analytic number theory*, Springer (Undergraduate Texts in Mathematics), New York, 1976.

[8] L. Babai, Graph Isomorphism in Quasipolynomial Time. Version 2.5, preprint (2018), `https://people.cs.uchicago.edu/~laci/quasi25.pdf`.

[9] E. Bach, Comments on search procedures for primitive roots, *Math. Comp.* **66**(220): 1719–1727, 1997.

[10] R. Beals、H. Buhrman、R. Cleve、M. Mosca 和 R. de Wolf，《通过多项式得到的量子下界》，J. ACM 48(4)：778–797，2001年。

[11] A. Bors, On the dynamics of endomorphisms of finite groups, *Appl. Algebra Engrg. Comm. Comput.* **28**(3): 205–214, 2017.

[12] A. Bors and Q. Wang, Coset-wise affine functions and cycle types of complete mappings, *Finite Fields Appl.* **83**: article ID 102088, 2022.

[13] A. Bors and Q. Wang, Generalized cyclotomic mappings: Switching between polynomial, cyclotomic, and wreath product form, *Commun. Math. Res.* **38**(2): 246–318, 2022.

[14] A. Caranti, Quasi-inverse endomorphisms, *J. Group Theory* **16**(5): 779–792, 2013.

[15] P.Y. Chen, *Solutions to* Introduction to Algorithms Third Edition. *31-2 Analysis of bit operations in Euclid's algorithm*, online resource, `https://walkccc.me/CLRS/Chap31/Problems/31-2/`.

[16] W.-S. Chou and I.E. Shparlinski, On the cycle structure of repeated exponentiation modulo a prime, *J. Number Theory* **107**(2): 345–356, 2004.

[17] A.H. Clifford and G.B. Preston, *The algebraic theory of semigroups. Vol. II*, American Mathematical Society (Mathematical Surveys, vol. 7), Providence, 1967.

[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, 3rd edn. 2009.

[19] G. Deng, Isomorphic digraphs from affine maps of finite cyclic groups, *ISRN Combinatorics*, **2013**: article ID 398641, 2013.

[20] J. Dubrois and J.-G. Dumas, Efficient polynomial time algorithms computing industrial-strength primitive roots, *Inform. Process. Lett.* **97**(2): 41–45, 2006.

[21] P. Flajolet and A.M. Odlyzko, Random mapping statistics, in: J.J. Quisquater and J. Vandewalle (eds.), *Advances in Cryptology – EUROCRYPT '89*, Springer (Lecture Notes in Comput. Sci., vol. 434), Berlin, 1990, pp. 329–354.

[22] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.12.0 released on 18 August 2022*, `http://www.gap-system.org`.

[23] J. von zur Gathen and D. Panario, Factoring Polynomials Over Finite Fields: A Survey, *J. Symb. Comput.* **31**: 3–17, 2001.

[24] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation, in: S.J. Lomonaco, Jr. (ed.), *Quantum information science and its contributions to mathematics. Proceedings of the American Mathematical Society Short Course held in Washington, DC, January 3–4, 2009*, American Mathematical Society (Proc. Sympos. Appl. Math., vol. 68), Providence, 2010, pp. 13–58.

[25] L.K. Grover, A fast quantum mechanical algorithm for database search, in: G.L. Miller (ed.), *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, ACM Press, New York, 1996, pp. 212–219.

[26] H. Halberstam, On the distribution of additive number-theoretic functions. III, *J. London Math. Soc.* **31**: 14–27, 1956.

[27] D. Harvey, An exponent one-fifth algorithm for deterministic integer factorisation, *Math. Comp.* **90**(332): 2937–2950, 2021.

[28] D. Harvey and J. van der Hoeven, Integer multiplication in time $O(n \log n)$, *Ann. of Math. (2)* **193**(2): 563–617, 2021.

[29] H.A. Helfgott, Isomorphismes de graphes en temps quasi-polynomial (d'après Babai et Luks, Weisfeiler-Leman...), in: *Séminaire Bourbaki. Vol. 2016/2017. Exposés 1120–1135*, *Astérisque* **2019**(407): i–ix and 1–584, pp. 135–182.

[30] H.A. Helfgott, J. Bajpai and D. Dona, Graph isomorphisms in quasi-polynomial time, preprint (2017), `https://arxiv.org/abs/1710.04574`.

[31] M. Hittmeir, A time-space tradeoff for Lehman's deterministic integer factorization method, *Math. Comp.* **90**(330): 1999–2010, 2021.

[32] W.M.L. Holcombe, *Algebraic automata theory*, Cambridge University Press (Cambridge studies in advanced mathematics, vol. 1), Cambridge, 1982.

[33] J.E. Hopcroft, R. Motwani and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Boston, 3rd edn. 2007.

[34] A.S. Jarrah, R. Laubenbacher, B. Stigler and M. Stillman, Reverse-engineering of polynomial dynamical systems, *Adv. in Appl. Math.* **39**(4): 477–489, 2007.

[35] P. Kaye, R. Laflamme and M. Mosca, *An Introduction to Quantum Computing*, Oxford University Press, Oxford, 2007.

[36] A.Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Physics* **303**(1): 2–30, 2003.

[37] E. Knill, R. Laflamme and W.H. Zurek, Resilient Quantum Computation, *Science* **279**(5349): 342–345, 1998.

[38] D.E. Knuth, *The Art of Computer Programming. Volume 3. Sorting and Searching*, Addison-Wesley, Reading, 2nd edn. 1998.

[39] R. Kumanduri and C. Romero, *Number Theory with Computer Applications*, Prentice Hall, Upper Saddle River, 1998.

[40] S. Lang, *Introduction to Diophantine Approximations. New Expanded Edition*, Springer, New York, 2nd edn. 1995.

[41] R. Laubenbacher and B. Pareigis. Equivalence relations on finite dynamical systems, *Adv. in Appl. Math.* **26**: 237–251, 2001.

[42] M. LeBorgne, A. Benveniste and P. LeGuernic, Polynomial dynamical systems over finite fields, in: G. Jacob and F. Lamnabhi-Lagarrigue (eds.), *Algebraic Computing in Control. Proceedings of the First European Conference held in Paris, March 13–15, 1991*, Springer (Lecture Notes in Control and Inform. Sci., vol. 165), Berlin, 1991, pp. 212–222.

[43] H.W. Lenstra Jr. and C. Pomerance, Primality testing with Gaussian periods, preprint (2011), `https://math.dartmouth.edu/~carlp/aks041411.pdf`.

[44] L. Lovász, *Combinatorial Problems and Exercises*, North-Holland, Amsterdam, 2nd edn., 1993.

[45] M. Martelli, *Introduction to Discrete Dynamical Systems and Chaos*, Wiley (Wiley-Interscience Series in Discrete Mathematics and Optimization), New York, 1999.

[46] R.S.V. Martins and D. Panario, On the heuristic of approximating polynomials over finite fields by random mappings, *Int. J. Number Theory* **12**(7): 1987–2016, 2016.

[47] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press (CRC Press Series on Discrete Mathematics and its Applications), Boca Raton, 1997.

[48] J.C.P. Miller, On factorisation, with a suggested new approach, *Math. Comp.* **29**: 155–172, 1975.

[49] G.A. Miller and H.G. Moreno, Non-abelian groups in which every subgroup is abelian, *Trans. Amer. Math. Soc.* **4**: 398–404, 1903.

[50] D. Milligan and M. Wilson, The Behaviour of Affine Boolean Sequential Networks, *Connection Science* **5**(2): 153–167, 1993.

[51] H.L. Montgomery and R.C. Vaughan, The large sieve, *Mathematika* **20**: 119–134, 1973.

[52] H. Niederreiter and A. Winterhof, Cyclotomic $\mathcal{R}$-orthomorphisms of finite fields, *Discrete Math.* **295**(1-3): 161–171, 2005.

[53] D. Panario and L. Reis, The functional graph of linear maps over finite fields and applications, *Des. Codes Cryptogr.* **87**: 437–453, 2019.

[54] A. Peinado, F. Montoya, J. Muñoz and A. Yuste, Maximal periods of $x^2 + c$ in $\mathbb{F}_q$, in: S. Boztaş and I.E. Shparlinski (eds.), *Applied algebra, algebraic algorithms and error-correcting codes. Proceedings of the 14th International Symposium (AAECC-14) held in Melbourne, November 26–30, 2001*, Springer (Lecture Notes in Comput. Sci., vol. 2227), Berlin, 2001, pp. 219–228.

[55] J.M. Pollard, A Monte Carlo method for factorization, *Nordisk Tidskr. Informationsbehandling (BIT)* **15**(3): 331–334, 1975.

[56] C. Qureshi and D. Panario, Rédei Actions on Finite Fields and Multiplication Map in Cyclic Group, *SIAM J. Discrete Math.* **29**(3): 1486–1503, 2015.

[57] C. Qureshi and D. Panario, The graph structure of Chebyshev polynomials over finite fields and applications, *Des. Codes Cryptogr.* **87**(2-3): 393–416, 2019.

[58] G. Robin, Grandes valeurs de la fonction somme des diviseurs et hypothèse de Riemann, *J. Math. Pures Appl. (9)* **63**(2): 187–213, 1984.

[59] A. Schönhage, Schnelle Berechnung von Kettenbruchentwicklungen, *Acta Inform.* **1**: 139–144, 1971.

[60] A. Schönhage and V. Strassen, Schnelle Multiplikation großer Zahlen, *Computing (Arch. Elektron. Rechnen)* **7**: 281–292, 1971.

[61] M. Sha, Digraphs from endomorphisms of finite cyclic groups, *J. Combin. Math. Combin. Comput.* **83**: 105–120, 2012.

[62] C.E. Shannon, The synthesis of two-terminal switching circuits, *Bell System Tech. J.* **28**: 59–98, 1949.

[63] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: S. Goldwasser (ed.), *35th Annual Symposium on Foundations of Computer Science. Proceedings of the IEEE Symposium held in Santa Fe, NM, November 20–22, 1994*, IEEE Computer Society Press, Los Alamitos, 1994, pp. 124–134.

[64] P.W. Shor, Fault-tolerant quantum computation, in: *37th Annual Symposium on Foundations of Computer Science. Held in Burlington, VT, October 14–16, 1996*, IEEE Computer Society Press, Los Alamitos, 1996, pp. 56–65.

[65] S. Ugolini, Graphs associated with the map $x \mapsto x + x^{-1}$ in finite fields of characteristic three and five, *J. Number Theory* **133**: 1207–1228, 2013.

[66] T. Vasiga and J. Shallit, On the iteration of certain quadratic maps over GF(p), *Discrete Math.* **277**(1-3): 219–240, 2004.

[67] A. Veliz-Cuba and R. Laubenbacher, On the computation of fixed points in Boolean networks, *J. Appl. Math. Comput.* **39**(1-2): 145–153, 2012.

[68] S. Wagstaff, *The Cunningham Project*, online database (December 13, 2022), `https://homes.cerias.purdue.edu/~ssw/cun/`.

[69] D. Wan and R. Lidl, Permutation polynomials of the form $x^r f(x^{(q-1)/d})$ and their group structure, *Monatsh. Math.* **112**: 149–163, 1991.

[70] Q. Wang, Cyclotomic mapping permutation polynomials over finite fields, in: S.W. Golomb et al. (eds.), *Sequences, subsequences, and consequences. Revised invited papers from the International Workshop (SSC 2007) held at the University of Southern California, Los Angeles, CA, May 31–June 2, 2007*, Springer (Lecture Notes in Comput. Sci., vol. 4893), Berlin, 2007, pp. 119–128.

[71] Q. Wang, Cyclotomy and permutation polynomials of large indices, *Finite Fields Appl.* **22**: 57–69, 2013.

[72] Q. Wang, A note on inverses of cyclotomic mapping permutation polynomials over finite fields, *Finite Fields Appl.* **45**: 422–427, 2017.

[73] Q. Wang, Polynomials over finite fields: an index approach, in: K.-U. Schmidt and A. Winterhof (eds.), *Combinatorics and Finite Fields. Difference Sets, Polynomials, Pseudorandomness and Applications*, De Gruyter (Radon Series on Computational and Applied Mathematics, vol. 23), Berlin, 2019, pp. 319–348.

[74] X. Wang and V.Y. Pan, Acceleration of Euclidean algorithm and rational number reconstruction, *SIAM J. Comput.* **32**(2): 548–556, 2003.

[75] J. Watrous, Quantum computational complexity, in: R.A. Meyers (ed.), *Computational complexity. Vols. 1-6*, Springer, New York, 2012, pp. 2361–2387.

[76] W.-D. Wei, X.-H. Gao and B.-F. Yang, Equivalence relation on the set of subsets of Z_v and enumeration of the equivalence classes (Research Announcement), *Adv. Math.* **17**: 326–327, 1988.

[77] W.-D. Wei and J.-Y. Xu, Cycle index of direct product of permutation groups and number of equivalence classes of subsets of Z_v, *Discrete Math.* **123**: 179–188, 1993.

[78] Y. Zheng, Y. Yu, Y. Zhang and D. Pei, Piecewise constructions of inverses of cyclotomic mapping permutation polynomials, *Finite Fields Appl.* **40**: 1–9, 2016.

附录：记号和术语一览表

以下两张表包含本文中出现的所有记号和术语。我们先从表10中基于数学符号的记号开始，这份列表相当简短；如果将其放在更长的表11中会很难查找。表11中的条目按字母顺序列出（拉丁字母排在希腊字母之前，小写字母排在相应的大写字母之前，标准字体的字母排在书法体字母之前，而书法体字母又排在哥特体字母之前）。

记号	page	补充说明
$\{0,1\}^{<\infty}$	77	所有有限比特串的集合
$\divideontimes$	18	$\mathbb{Q}[x_n$ 上的某个 $\mathbb{Q}$-双线性乘积：$n \in \mathbb{N}^+]$, 最初由 Wei 和 Xu 定义，使得 $\mathsf{CT}(\psi_1 \otimes \psi_2) = \mathsf{CT}(\psi_1) \divideontimes \mathsf{CT}(\psi_2)$
$\bigotimes_{j\in I} g_j$	13	当每个 g_j 都是一个函数 $X_j \to X_j$时定义；它是该函数在 $\prod_{j\in I} X_j$ 上由 g_j逐分量作用得到
gg'	20	函数 g 和 g'的复合（先 g，再 g'）；同义词：$g' \circ g$
$\langle g_1, \ldots, g_n\rangle$	29	若 $g_1, \ldots, g_n$ 是群 G 的元素，这表示由 g_j 生成的 G 的子群
$G = H \ltimes N$	21	表示 G 是（内）半直积 H 与 N 的直积
$\lvert G_1 : G_2\rvert$	17	G_2 在 G_1中的子群指标
$G_1 \wr G_2$	8	非本原置换 wreath 积
$H \leq G$	29	“H 是 G的子群”的简写
$\mathfrak{I} \sim \mathfrak{I}'$	53	$\mathsf{SF}(\mathfrak{I}) = \mathsf{SF}(\mathfrak{I}')$ 的简写
$\mathcal{P} \wedge \mathcal{Q}$	42	这些划分的下确界（最粗的公共细化）$\mathcal{P}$ 和 $\mathcal{Q}$；如果它们是算术划分，那么 $\mathcal{P} \wedge \mathcal{Q}$ 可表示为由.....张成的算术划分。任意选取的生成同余式的串联用于 $\mathcal{P}$ 和 $\mathcal{Q}$的序列
x^g	20	如果 x 是函数 g 的一个参数，则此尤其可以表示函数值 $g(x)$ 与复合记号 gg' 联用
$\bigotimes_{j\in I} \Gamma_j$	13	Γ_j 的有向图张量积
$\vec{\nu} \diamond \vec{\nu}'$	49	$\vec{\nu}$ 与 $\vec{\nu}'$的拼接

表10：基于数学符号的记号。

记号/术语	page	补充评论
a	13	表示仿射映射的线性系数的变量 $A: x \mapsto ax + b$ 之 $\mathbb{Z}/m\mathbb{Z}$
$\mathfrak{a}$	10	用于表示同余式（或非同余式）模数的变量
$\mathfrak{a}_{i,j}$	52	用于第 3.4 小节和第 5.2.3 小小节; $\mathcal{P}_i$ 的第 j 个生成同余式的模数
$\overline{\mathfrak{a}}_j$	64	（用于第 4.2 小节中的示例）；模数为第 j个生成同余式，属于$\mathcal{Q}_{0,0}$
A	13	表示群上的仿射映射的变量（大多为 $\mathbb{Z}/m\mathbb{Z}$）
$\mathsf{AC}(\mathcal{P})$	11	算术复杂度的 $\mathcal{P}$

可容许的指标编排函数	36	none
可接受的输入	78	none
可接受的输出	78	none
仿射函数 (群之间)	190	none
仿射映射（群的）	20	none
a_i	4	定义于 $i \in \{0,1,\ldots,d-1\}$； $\mathbb{F}$ 的一个元素 q 与 f 相关； $f(x) = a_i x^{r_i}$ 对所有 $x \in C_i$
A_i	8	对每个满足 $a_i \neq 0$ 的 $i \in \{0,1,\ldots,d-1\}$ 定义；它是与 $f_{\mid C_i}$ 对应的 $\mathbb{Z}/s\mathbb{Z}$ 的唯一仿射映射： $C_i \to C_{\overline{f}(i)}$ 通过 ι_i、 ι 将 C_i、 $C_{\overline{f}(i)}$ 与 $\mathbb{Z}/s\mathbb{Z}$ 对应起来时 $\overline{f}(i)$, 并且它由以下公式给出： $A_i(x) = r_i x + {}^{e_i + r_i i - \overline{f}(i)\, d}$
$\mathcal{A}_i$	39	$A_{i_0} A_{i_1} \cdots A_{i_{\ell-}}$ 的仿射映射 $_1$ $\mathbb{Z}/s\mathbb{Z}$ 的
$\mathcal{A}_{i,h}$	46	用于第 3.3 小节；为 $i \in \mathsf{per}(\overline{f}) \setminus \{d\}$ 定义 以及 $h \in \mathbb{N}_0$；其定义为 $A_{i_-}{}_h A_{i_{-h+1}} \cdots A_{i_{-1}}$, $\mathbb{Z}/s\mathbb{Z}$ 的仿射映射
算法问题	77	none
$\overline{\mathcal{A}}_{i,p}$	99	用于第 5.2.1 和 5.3.3 小小节；它是约化 $\mathcal{A}_i$ 模 $p^{\nu_p(s)}$, 一个 $\mathbb{Z}/p^{\nu_p(s)}\mathbb{Z}$
$\mathcal{A}'_i$	102	$\mathbb{Z}/s'_i\mathbb{Z}$ 的仿射置换 $\mathcal{A}_i \bmod s'_i$
aord(x)	30	m 的加法阶（ $x \in \mathbb{Z}/m\mathbb{Z}$ 必须能从上下文中明确判断）
arc	2	none
(算术运算) 复杂度	11	none
算术划分	11	none
Aut(G)	29	群 G 的自同构群
b	13	表示该仿射映射的常数系数的变量 A： $x \mapsto ax + b$ $\mathbb{Z}/m\mathbb{Z}$的；也更广泛地用于群 G上的仿射映射在 1_G 处的值
$\mathfrak{b}$	10	用于表示右端的变量 一个同余式或不相容式
$\mathfrak{b}_{i,j}$	52	用于第 3.4 小节和第 5.2.3 小小节； $\mathcal{P}_i$ 的第 j 个生成同余式的右端
$\overline{\mathfrak{b}}_j$	64	（用于第 4.2 小节中的示例）； 第 j 个生成同余式的右端，属于 $\mathcal{Q}_{0,0}$
B	24	表示（通常为算术）划分中一个块的变量； 参见记号 $B_{\cdot}$ $_r$ 在问题 2.3.7之后引入
$\mathcal{B}_i$	52	在第 3.4 小节中使用；将 $x \in C_i$ 映射到 的函数 到包含 x的唯一 $\mathcal{P}_i$-块

$\mathcal{B}(\mathcal{P},\vec{\nu})$	11	算术划分中的块 $\mathcal{P}$ 与逻辑符号元组 $\vec{\nu}$ 相关联
B_m	11	第 m 个贝尔数
位操作	75	none
爆炸函数	71	none
有界拉斯维加斯算法 对偶复杂度	78	none
有界查询复杂度	78	none
手链	10	none
手镯图	10	none
BU_m	71	m-膨胀函数；唯一的 $\mathbb{Q}$-代数端同态 满足以下条件的 $\mathbb{Q}[x_n:\ n\in\mathbb{N}^+]$ 使得 $\mathrm{BU}_m(x_n)=x_{mn}$ 对所有 $n\in\mathbb{N}^+$
C	4	$\mathbb{F}^*$ 的指标为 d 的子群 $_q$
子节点	15	同义词：后继
C_i	4	$\mathbb{F}$ 的某个子集 $_q$ 定义于 $i\in\{0,1,\ldots,d\}$; 即，$C_i:=\omega^i C$ 若 $i\in\{0,1,\ldots,d-1\}$，以及 $C_d:=\{0_{\mathbb{F}_q}\}$
$\mathcal{C}_{\mathrm{class}}$	78	经典复杂度（即比特运算次数）； 用于表示 $\vec{\mathcal{C}}^{(\mathrm{qry})}$ 或 $\vec{\mathcal{C}}^{(\mathrm{LV})}$ 的第一项
$\mathcal{C}_{\mathrm{conv}}$	78	一种转换复杂度（即转换次数 从比特到量子比特，反之亦然）； 用于表示 $\vec{\mathcal{C}}^{(\mathrm{LV})}$ 的第三项
$\mathcal{C}_{\mathrm{fact}}$	78	整数因式分解查询的计数; 用于表示 $\vec{\mathcal{C}}^{(\mathrm{qry})}$ 的第二个条目
$\mathcal{C}_{\mathrm{fdl}}$	78	（有限）域离散对数查询的数量; 用于表示 $\vec{\mathcal{C}}^{(\mathrm{qry})}$ 的第四个条目
$\mathcal{C}_{\mathrm{mdl}}$	78	模离散对数查询次数; 用于表示 $\vec{\mathcal{C}}^{(\mathrm{qry})}$ 的第三项
$\mathcal{C}_{\mathrm{mord}}$	78	模乘法阶查询的次数; 用于表示 $\vec{\mathcal{C}}^{(\mathrm{qry})}$ 的第五个条目
$\mathcal{C}_{\mathrm{prt}}$	78	原根查询次数; 用于表示.....的第六项 $\vec{\mathcal{C}}^{(\mathrm{qry})}$
$\mathcal{C}_{\text{量子}}$	78	量子复杂度（即基本量子门的数量） 门）；用于表示.....的第二个条目 $\vec{\mathcal{C}}^{(\mathrm{LV})}$
$\vec{\mathcal{C}}^{(\mathrm{LV})}$	78	表示拉斯维加斯双重复杂度的变量，它 本身是由不同类型的复杂度分量构成的三元组
$\vec{\mathcal{C}}^{(\mathrm{qry})}$	78	表示查询复杂度的变量，它本身 一个由不同类型的分量复杂度组成的 6-元组

$\mathfrak{C}_{i,L}$	166	所有满足 l 是倍数的 $l \in \{1,2,\ldots,L\}$ 的集合 ℓ的（ i的 $\overline{f}$-周期长度）和最大公约数$(s, \overline{\alpha}_{i,l} - 1) \mid \overline{\beta}_{i,l}$; 这些正是满足 $\eta_{i,l}(x)$ is 的那些 $l \in \{1,2,\ldots,L\}$ 定义良好，并刻画了对于 $x \in C_i$，何时 $f^l(x) = x$
经典复杂度	75	none
复杂度 （划分的）	11	none
复合（ 算法问题)	79	none
以x共轭	21	群 G上的 $x \in G$ 共轭, 即自同构 $y \mapsto x^{-1}yx$ 的 G
连通分支 （函数图的）	2	none
一致的 (由 m构成的CC系统)	23	none
连分数 系数	84	none
收敛的	84	none
转换复杂度	76	none
陪集上的仿射函数	191	与狭义的逐陪集仿射函数相同
陪集逐一仿射函数 在狭义上	190	none
逐陪集仿射函数 广义上	190	none
CRL 列表	7	“循环代表元和长度列表”的缩写
CT(ψ)	3	ψ的循环类型
循环类型	3	none
相关联的循环序列 与 $\vec{\tau}$ 搭配	9	none
循环序列	9	none
分圆映射	4	none
d	3	f的指标
$\mathfrak{d}$	37	表示除数的变量
$\mathfrak{d}_{i,p,\vec{u}}$	104	$\prod$ 的简写 $\prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_i,\vec{u})}$，这与 $\prod_{\mathfrak{p}\in\mathfrak{P}_{i,p,\vec{u}}} \mathfrak{p}^{\nu_{\mathfrak{p}}(l_{i,p,\vec{u}_p})}$ 且总是整除 $l_{i,p,\vec{u}_p}$
$\mathfrak{D}$	73	一个表示某个.....的紧凑描述的变量 有限有向有根树同构类型 相对于固定的递归树描述列表
二面体序列	10	none
有向根树	2	我们假设所有弧都指向根

离散对数 （在群中）	88	none
离散动力系统	2	none
分布数	25	none
双重算法	75	none
对偶有向图	15	none
双重模型	75	none
e	18	用于循环类型中变量的指数（次数）； 需要与记号 e_i 区分开来
e_i	8	对满足 $a_i \neq 0$ 的那些 $i \in \{0, 1, \ldots, d-1\}$ 定义； 这是以 ω 为底的 a_i 的离散对数，即 $a_i = \omega^{e_i}$
E	15	表示有向图的边（弧）集的变量
E^{-1}	15	E的逆关系
$E(\vec{\nu}, J)$	24	某种整除条件的合取式
等价标记的 算术划分	186	none
等价序列的 带标记的算术划分	187	none
等价系统 个 m-CC	23	none
Expand(Δ)	53	有限边加权的“Expand 版本” 有向根树 Δ
f	3	$\mathbb{F}_q$ 的广义分圆映射
f_{per}	39	f 到 $\bigcup_{i \in \mathrm{per}(\overline{f})} C_i$ 的限制
$\overline{f}$	8	f 在 $\{0, 1, \ldots, d\}$ 上诱导的函数， 通过 $f(C_i) \subseteq C_{\overline{f}(i)}$ 隐式定义
$\overline{f_{\mathrm{per}}}$	39	f_{per}的诱导函数，等于 $\overline{f}_{\lvert \mathrm{per}(\overline{f})}$
$\mathfrak{f}$	28	表示不动点的变量；在命题 2.3.6 中 以及基于此的讨论，这是一个特别 有限循环群的仿射映射所定义的不动点
$\mathfrak{f}_i$	123	$\mathcal{A}_i \bmod s_i''$ 的唯一周期点
$\mathfrak{f}_{i\,p}$	100	在$\mathcal{A}_{i,p}$ 有不动点时定义；某个不动点 $\mathcal{A}_{i,p}$的点，由命题 2.3.6 中的公式给出
$\mathcal{F}$	194	一组广义分圆映射 定义在同一个有限域上
$\mathbb{F}_q$	2	大小为 q 的有限域
有限动力系统	2	none
函数图	2	none
g	2	一个函数 $X \to X$
G	20	表示抽象群的变量
$\mathcal{G}$	194	一组函数 $X \to X$

广义分圆 映射	3	none
良好$_{\vec{r}}(\mathcal{I})$	36	所有 $\mathcal{I}$-良好元组的集合（与 $\vec{r}$ 相关，或者更确切地说， 其循环长度元组 $(l_j(\vec{r}))_{j=1,2,\ldots,n}$ 对于该 $\vec{r}$-可容许索引函数 $\mathcal{I}$
良好的元组	36	none
$g_{\|Y}$	7	g 在子集 $Y \subseteq X$ 上的限制
h_i	43	定义于 $i \in \{0,1,\ldots,d-1\}$；如果 i 是 $\overline{f}$-暂态的，那么 $h_i =$ht(树$_\Gamma$ $_{\overline{f}}(i)$）；如果 i 是 $\overline{f}$-周期的，则 $h_i = \infty$
$h'_{i,k}$	113	定义于 $i < d$ 是 $\overline{f}$-周期的且 $k \in \mathbb{Z}$时； 它是满足以下条件的最小正整数 h' 最大公约数$(\prod_{t=0}^{h'-1} \alpha_{i_{k-h'+t}}, s)$ = 最大公约数$(\prod_{t=0}^{h'-2} \alpha_{i_{k-h'+t}}, s)$; 有 $H_i =$max$\{h'$ $_{i,k}$ ： $k = 0, 1, \ldots, \ell-1\}$
$\mathfrak{h}(x)$	45	第 3.3 小节中的某个技术参数； 当 $x \in C_i$ 对于一个 $\overline{f}$-周期的 $i < d$ 时，我们的 对 Tree$^\Gamma{}_f(x)$ 的理解通过对 $\mathfrak{h}(x)$ 的递归而获得 （先处理 $\overline{f}$-瞬态指标 i 和 $i = d$）
H_i	45	Γ_{per} 上方有根树的最大高度 f-周期的顶点，位于陪集 C_{i_t} 中，适用于 $t \in \mathbb{Z}$
$\overline{H}$	138	$\Gamma_{\overline{f}}$ 中的最大树高
$\mathcal{H}_i$	56	当 $\overline{f}$ 是一个置换时，为 $i < d$ 定义； 这是 f-周期顶点上方的公共树高。 在 C_i 中；我们注意到 $H_i = \max_{t\in\mathbb{Z}} \mathcal{H}_{i_t}$
$\mathfrak{H}$	113	max$\{H_i$ 的简写： $i \in$ per$(\overline{f})$ $\setminus \{d\}\}$
高度	16	有限有向有根树的属性 ， 表示其中有向路径的最大长度
ht(Δ)	16	Δ 的高度
超核	21	none
i	4	取值范围为 $\{0,1,\ldots,d\}$ 的指标
i_t	39	为 $i \in$ per$(\overline{f})$ 和 $t \in \mathbb{Z}$ 定义的记号； （$\overline{f}$ 的简写 $_{\|\mathrm{per}(\overline{f})}$）$^t(i)$；特别是， $(i_0, i_1, \ldots, i_{\ell-1})$ 是 $i = i_0$ 的 $\overline{f}$-循环
i'	44	i_{-1} 的简写
$\mathfrak{i}$	151	表示一个单射函数的变量
I	13	表示指标集的变量
I_i	52	用于第 3.4 小节；那些 $j \in \{1,2,\ldots,m_i\}$ 的集合 使得$x \equiv \mathfrak{b}_{i,j}$ （模$\mathfrak{a}_{i,j}$）的真值 沿着模 s 下 r_i 的 $\mathcal{A}_i$-循环为常数
$\mathcal{I}$	36	表示一个 $\vec{r}$-可容许 索引函数的变量（对于某个 $\vec{r}$）

$\mathcal{I}_i$	104	某个函数 $\mathfrak{P}'_i \to \mathfrak{P}_i$；如果 $\mathcal{I}_i(\mathfrak{p})$ 只有一个不同于的值 for $\vec{u} \in \overline{Y}_i$，那么 $\mathcal{I}_i(\mathfrak{p})$ 就是该值，否则，$\mathcal{I}_i(\mathfrak{p})$ 是一个不同于 $\mathfrak{p}$ 的唯一值，其中 $\mathcal{I}_i(\mathfrak{p}, \vec{u})$ 取值为
$\mathcal{I}_{i,\vec{u}}$	102	某个 $\vec{r}_i(\mathfrak{p}, \vec{u})$-可容许索引函数，将其延拓，使其定义域为 $\mathfrak{P}'_i$
$\mathfrak{I}$	17	表示有限……同构类型的变量 有向有根树（无论是否带边权重，取决于上下文）
$\mathfrak{I}^+$	42	通过附加所得到的有根树同构类型 将单个 $\mathfrak{I}$ 复制到新的根
$\mathfrak{I}_i$	70	（用于第4.3小节）Tree_Γ 的简写 $\overline{f}(i)$
$\mathcal{I}$-优良元组	36	none
由 f 诱导	8	指代函数 $\overline{f}$ 的属性
$\mathrm{im}(g)$	42	函数 g 的像（也称为值域）
下确界（针对划分）	42	none
$\mathrm{inv}_n(u)$	26	模 n 的乘法逆元 的 $u \in \mathbb{Z}$，其最大公约数$(u, n) = 1$
顶点-同构 带标记的有向图	9	none
有限域的同构 边权有向 有根树	53	none
迭代原像	2	none
J	24	表示指标集的变量
$J_+(\vec{\nu})$	24	指标集合 j 使得 $\vec{\nu}$ 的第 j项为 $\emptyset$
$J_-(\vec{\nu})$	24	指标集合 j，使得 $\vec{\nu}$ 的第 j 个条目为 $\neg$
$\ker^{(k)}(\varphi)$	17	对 $k \in \mathbb{N}_0$ 和群自同态 φ of G定义； 它是正规子群 $\{x \in G : \varphi^k(x) = 1_G\}$ of G
$\mathcal{J}_{i,p,k}$	103	u_p 值域中最多五个子区间之一 使得 $\nu_p(l_{i,p,\vec{u}_p})$ 可由 u 中的一个简单公式给出。p 每当 u_p 属于固定的 $\mathcal{J}_{i,p,k}$
$\mathcal{J}'_{i,p,k}$	103	一个子区间 $\mathcal{J}_{i,p,k}$，使得 $\mathcal{I}_{i,\vec{u}}(p) = p$ 当且仅当 $u_p \in \bigcup_{k=1}^{\mathfrak{m}_{i,p}} \mathcal{J}'_{i,p,k}$
k_p	104	（第 5.2.1 小节中）p 标记的条目 $\vec{k}$ （$\mathbb{Z}/l_i$ 的一个元素 $\vec{u}\mathbb{Z}$）
$\vec{k}$	104	（第 5.2.1 小节中）表示一个 $\mathcal{I}_i$ 的变量 $\vec{u}$-良好元组
k'_p	104	（在第 5.2.1 小节中）p 标记的条目 $\vec{k}'$ （一个元素属于 $\mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{u}})\mathbb{Z}$）

$\vec{k}'$	104	一个取值遍历 $\prod$ 的元组 $_{\mathfrak{p}\in\mathfrak{P}_i}\mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{u}})\mathbb{Z}$, 对于固定的 $\vec{u}\in\overline{Y}_i$ 用作参数 （某个双射参数化的自变量） 良好$_{\vec{r}_i(\vec{u})}(\mathcal{I}_{i,\vec{u}})$，用于.....的良好元组集合 $_{\text{久}}^{\vec{r}_i(\text{综合久}\,\vec{u})}$-可容许索引函数 $\mathcal{I}_{i\quad,\vec{u}}$
$K'_{i,\vec{u}}$	104	集合 $\prod_{\mathfrak{p}\in\mathfrak{P}_i}\mathbb{Z}/(l_{i,p,\vec{u}_p}/\mathfrak{d}_{i,p,\vec{u}})\mathbb{Z}$，由此 $\vec{k}'$ 源自
l	7	周期长度为 f 或 g
$l_{i,j}$	52	用于第 3.4 小节和第 5.2.3 小小节; 在 $\mathcal{A}_i$ 模 $\mathfrak{a}_{i,\,j}$ 下 r'_i 的周期长度
$l_{i,\vec{u}}$	102	周期长度 $\operatorname{lcm}\{l_{i,p,\vec{u}_p}:p\in\mathfrak{P}_i\}$ of $\bigotimes_{p\in\mathfrak{P}_i}\mathcal{A}_{i\,p}$ 相关于 $\vec{u}$
$l_{i\,p,\vec{u}_p}$	102	“分量周期长度” proj$_2$(par$'_{i,p}(\vec{u}_p)$)
ℓ	39	表示 $\overline{f}$ 的周期长度的变量，与一个循环相对。 f 的长度，记作 l；如果 $i\in\{0,1,\ldots,d\}$ 一旦确定，则 ℓ 始终是 i的 $\overline{f}$-周期长度
$\mathfrak{l}$	123	表示$\log_\omega(r)$ 的简写，其中 $r\in\mathbb{F}_q$ 是一个 f-周期点 表示 Γ_f 的一个连通分支
$\mathfrak{l}_{i,j}$	52	用于第 3.4 小节和第 5.2.3 小小节; 它表示仿射离散对数值 $\log^{(\mathfrak{a}_{i,j})}{}_{\mathcal{A}_i}(r_i,\mathfrak{b}_{i,j})$
L_i	105	使得.....的最小非负整数 $\text{最大公约数}\left(\overline{\alpha}_i^{L_i},s\right)=\prod_{p\mid\text{最大公约数}(\overline{\alpha}_i,s)}p^{\kappa_p}$, 用于 $\mathcal{A}_i$ 模 (s/s'_i) 的唯一周期点公式
$\mathcal{L}$	7	表示 CRL 列表的变量
$\overline{\mathcal{L}}$	38	$\overline{f}$的 CRL 列表
$\mathcal{L}^{(1)}$	35	CRL 列表 $\mathcal{L}$ 中各对的首项集合
$\mathcal{L}_i$	39	一个从 f 到 $\bigcup_{t=0}^{\ell-1}C_{i_t}$ 的限制 CRL 列表
$\mathcal{L}'_i$	39	$\mathcal{A}$的 CRL 列表 $_i$
$\mathcal{L}'_{i,p}$	99	一个由 $\mathcal{A}_{i,p}$ 组成的 CRL 列表，其中所有指定的周期长度 已完全进行因数分解；具体定义见表 8
$\mathcal{L}''_i$	102	$\mathcal{A}'$的 CRL 列表$_i$
$\mathcal{L}(p^v,a)$	29	$\mathbb{Z}/p^v\mathbb{Z}$ 的自同构 μ_a 的某个 CRL 列表, 在表1中定义，并且（对于 $p>2$）取决于 一个原根 $\mathfrak{r}$ 模 p^v 的选择
$\mathcal{L}(p^v,a,b)$	32	仿射置换 $x\mapsto ax+b$ 的某个 CRL 列表 的 $\mathbb{Z}/p^v\mathbb{Z}$，定义于表 2 中，并且（对于 $p>2$） 取决于模 p^v 的原根 $\mathfrak{r}$ 的选择
$\mathfrak{L}$	77	表示算法问题的变量
$\mathfrak{L}_{\text{in}}$	77	算法问题 $\mathfrak{L}$ 的可容许输入集合; 正式地说，这是函数 $\mathfrak{L}$ 的定义域

$\mathfrak{L}\mathfrak{L}'$	79	算法问题 $\mathfrak{L}$ 和 $\mathfrak{L}'$的复合（先 $\mathfrak{L}$，再 $\mathfrak{L}'$）；同义记号：$\mathfrak{L}' \circ \mathfrak{L}$
$\mathfrak{L}' \circ \mathfrak{L}$	79	参见 $\mathfrak{L}\mathfrak{L}'$
lab	186	表示标记函数的变量（即，带标记的算术划分的第二项）
lab_i	185	算术划分 $\mathcal{P}^{(i)}$ 的某个标记函数，用于两种略有不同的含义（另见定义 6.2.1 前的段落）
Lab	187	表示标记函数的变量（如 lab）
带标记的算术划分	185	none
拉斯维加斯算法	75	none
层$_h$	117	满足 $h_i = h$ 的那些 $i \in \{0, 1, \ldots, d-1\}$ 的集合
左正则表示	20	none
$\log_a^{(m)}(x)$	37	以 a 为底，x 模 m 的离散对数
$\log_A^{(m)}(x, y)$	37	一个“仿射离散对数”，在第2.4小节开头形式化定义
$\log_x(y)$	88	如果 x 和 y 是群 G 的元素，则这表示以 y 为底的离散对数（如果 y 不是 x 的幂，则定义为 ∞）
m	2	正整数；用作一般模数
m_i	42	定义于 $i \in \{0, 1, \ldots, d-1\}$；表示长度固定的 s-同余生成序列的 $\mathcal{P}_i$
$\overline{m}_i$	185	$\mathcal{P}^{(i)}$ 的标准生成同余序列的长度；用第 3.4 小节的记号，有 $\overline{m}_i = m_i - \lvert I_i \rvert$
m_p	89	所谓的 p-部分 $m \in \mathbb{N}^+$；定义为 $p^{\nu_p(m)}$
$m_{p'}$	88	所谓的 p'-部分 $m \in \mathbb{N}^+$；定义为 $m/p^{\nu_p(m)}$
m'	28	所有素数幂 $p^{\nu_p(m)}$ 的乘积，其中 p 不整除某个由上下文可知的其他整数（通常是某个仿射映射的线性系数 of $\mathbb{Z}/m\mathbb{Z}$); 不要与记号 i' 混淆
m''	28	商群 m/m'
$\mathfrak{m}$	145	表示同构类型数量的变量函数图的连通分支
$\mathfrak{m}_i$	171	一个用于定理 5.3.3.4 证明中的函数，它编码了 $\mathcal{W}_i$的块大小 $_{,L}$
$\mathfrak{m}_{i\,p}$	103	区间数量 $\mathcal{J}_{i,p,k}(k = 1, 2, \ldots, \mathfrak{m}_{i,p})$
M	41	表示全集的一个子集的变量（通常 $\mathbb{Z}/m\mathbb{Z}$ 或 $\mathbb{F}_q$）
M^c	25	M 的补集（在其相应的全集中）
m-CC	10	“m-同余条件”的缩写

m-(不)同余式	10	其模数整除 m 的（不）同余式
合并（ 有序数组)	80	none
minperl($\vec{\mathfrak{r}}$)	144	有限序列 $\vec{\mathfrak{r}}$的最小周期长度
蒙特卡罗算法	84	none
mpe(m)	20	m的最大素数指数, 即，$\mathsf{mpe}(m) = \max_p \nu_p(m)$
一个有根 树同构类型的倍数	42	none
n_i	45	$\mathcal{R}_i$ 的一个固定生成 s-同余式序列的长度
$\overline{n}_i$	138	用于引理 5.3.2.2(2) 的证明中；它表示 唯一的 $n \in \mathcal{N}$，使得对每个 $x \in C_i$，$\mathfrak{I}_n = \mathsf{Tree}^{\Gamma_f}(x)$
$\overline{n}_{i,k}$	141	用于引理 5.3.2.2(4) 的证明中；它表示 唯一的 $n \in \mathcal{N}$，使得 $\mathsf{Expand}(\mathfrak{I}_{i,\ k}) = \mathfrak{I}_n$ （即 $i \in S_{n,\ \mathrm{trans},\ k}$）
$n_i'\ (\vec{\nu})$	124	第 5.2.3 小小节中使用的一种特殊记号； 它表示唯一的 $n \in \mathcal{N}$，使得 $\vec{\nu}^+ \in S_{i,\ n}$
$\mathfrak{n}_p$	100	用于第 5.2.1 小节；它表示 $p-1$ 的不同的素因子个数
$\mathcal{N}$	110	在小小节 5.2.2 和 5.3.2 中使用；它表示 由 $\mathbb{N}_0$的初始段构成，其中包含满足以下条件的 n 。 紧凑树描述 $\mathfrak{D}_n$ 已定义 (在相关算法中的相应位置)
$\mathfrak{N}$	146	表示树项链列表的变量
$\mathbb{N}^+$	3	正整数的集合
$\mathbb{N}_0$	4	非负整数集合
项链	9	none
项链图	9	none
nil(φ)	21	φ 的超核
$\mathcal{O}_{i,L}$	165	为 $\overline{f}$-周期的 $i \in \{0,1,\ldots,d-1\}$ 定义 和 $L \in \mathbb{N}_0$；它是所有逻辑符号元组的集合 $\diamond_{t=0}^{H_i+L} \vec{o_t} \in \{\emptyset, \neg\}^{n_{i_0}+n_{i_{-1}}+\cdots+n_{i_{-H_i-L}}}$ 使得 相关块 $\mathcal{B}(\mathcal{Q}_{i,\ H_i+L}, \diamond_{t=0}^{H_i+L} \vec{o_t} \diamond \vec{\xi}_{i,\ H_i})$ 的 $\mathcal{Q}_{i,\ H_i+L}$ 由 f-周期点组成，且非空
ord(x)	28	群元素 $x \in G$ 的阶；群 G 必须 从上下文中应当清楚；对于 $\mathbb{Z}/m\mathbb{Z}$ 中的元素，这始终 表示乘法阶（因此仅定义良好 如果 x 是模 m 的单位元）——另请参见记号 aord
ord$_n$(x)	89	x 模 n的乘法阶
p	14	表示一个素数的变量，不一定是 q 的素数基数； 使用 p作为求和指标意味着只有素数指标 应使用满足明确规定约束条件的内容

$\mathfrak{p}$	103	用于表示 $\mathfrak{P}_i'$ 的一般元素的变量（以区别于用于表示 $\mathfrak{P}_i$ 的元素的 p）
$\mathfrak{p}_{p,k}$	100	（在固定的因式分解中） $p-1$ 的第 k 个素因子
$P(T)$	72	变量 T 中的多项式
$\mathcal{P}$	11	表示一个划分的变量，通常是一个算术运算划分
$\mathcal{P}_i$	10	在 $C_i \cong \mathbb{Z}/s\mathbb{Z}$中定义的某种算术划分 第 3.3 小节 ，它“控制”了 C 中顶点上方的树 $_i$
$\mathcal{P}^{(i)}$	52	用于第 3.4 小节和第 5.2.3 小小节；某个对循环的（部分）进行编码的算术划分 表示.....的有根树同构类型序列 包含 r_i' 的 Γ_f 连通分支
$\mathcal{P}_i'$	43	$\mathfrak{P}'(\mathcal{P}_i, A_i)$ 的简写
$\mathcal{P}_{i,h}$	48	一种特定的 C_i 的算术划分，使得对于 $x \in C_i$ 其中 $\mathfrak{h}(x) = h$，同构类型为树$\Gamma_f(x)$ 仅取决于包含 x 的 $\mathcal{P}_{i,h}$-块
$\mathfrak{P}$	10	一个用于记号的算子 $\mathfrak{P}(x \equiv \mathfrak{b}_k \text{(模 } \mathfrak{a}_k\text{)} : k = 1, 2, \ldots, K)$, 这表示 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分（对于 隐含地清楚 $m \in \mathbb{N}^+$) 由那些 m-同余式张成
$\mathfrak{P}_i$	99	所有素数 $p \mid s$ 使得 $p \nmid \overline{\alpha}_i$ （即，使得 $\mathcal{A}_{i\,p}$ 是 $\mathbb{Z}/p^{\nu_p}$的置换$_{(s)}\mathbb{Z}$)
$\mathfrak{P}_{i,p,\vec{u}}$	104	原像集 $\mathcal{I}^{-1}_{i,\vec{u}}(\{p\})$
$\mathfrak{P}_i'$	102	扩展定义域 $\mathfrak{P}_i \cup \pi(\prod_{p\in\mathfrak{P}_i}(p-1))$ 的 $\mathcal{I}_{\vec{\nu}\vec{u}}$
$\mathfrak{P}(\Theta(x))$	47	$\mathbb{Z}/m\mathbb{Z}$ 的算术划分由以下内容生成 $\Theta(x)$ 中 m-CC 的未否定版本
$\mathfrak{P}'(\mathcal{P}, A)$	24	当 $\mathcal{P}$ 是 $\mathbb{Z}/m\mathbb{Z}$ 的算术划分并且具有一个固定的生成同余序列，且 A 是一个仿射映射时定义 $\mathbb{Z}/m\mathbb{Z}$的；它是 $\mathbb{Z}/m\mathbb{Z}$ 的一种特定算术划分 使得交集大小 $\lvert A^{-1}(\{x\}) \cap B\rvert$，其中 $B = \mathcal{B}(\mathcal{P}, \vec{\nu})$ 是 $\mathcal{P}$ 的一个固定块，为常数（等于 $\sigma_{\mathcal{P},\,A}(\vec{\nu},\vec{\nu'})$) 当 x 遍历固定块 $\mathcal{B}(\mathcal{P}', \vec{\nu'})$ $\mathcal{P}' = \mathfrak{P}'(\mathcal{P}, A)$ 的
两两同余一致	191	none
两两陪集交性质	192	none
par_i	98	$\mathcal{L}_i$ 的双射参数化；对于 $i < d$，可通过以下方式获得 将 par_i' 中所有循环的周期长度扩大因数 ℓ（i 的 $\overline{f}$-循环长度）
par_i'	99	$\mathcal{L}_i'$ 的双射参数化(对于 $i < d$)
par_i''	102	$\mathcal{L}_i''$ 的双射参数化

$\mathrm{par}'_{i,p}$	99	$\mathcal{L}'$的双射参数化 $_{i,p}$
分区树寄存器	73	none
PCIP	192	“两两陪集相交性质”的缩写
PCIP 群	192	none
per(g)	2	X 中的 g-周期点集合
周期长度（一个函数下参数的周期长度）	2	none
周期的	2	同义用法：在 f 下周期的； f-周期的
（有限序列的周期长度）	144	none
perl$_g(x)$	2	$x \in X$ 在 g下的周期长度
相位反转	77	none
pperl$_g(x)$	2	$x \in X$ 在 g下的预周期长度
预周期长度 (长度)	2	none
预周期	2	none
proc$_{i,h}$	40	一个辅助（繁殖）数，用于定理 3.2.1 的证明
proc$_k(x)$	15	Γ of $x \in$ V(Γ)中的第 k 个繁衍数；为使表述更清晰，也写作 $\mathrm{proc}_k^{(\Gamma)}(x)$
繁衍数	15	none
proj$_{i,\ L}$	165	将 $\mathcal{O}_{i,\ L}$ 中的每个元组映射到的投影 其长度为 n_i 的初始段 $_0 + n_{i-1} + \cdots + n_{i-H_i}$
proj^j	102	对 $j \in \{1,2\}$ 定义，它是（按类大小划分的）将有序对映射到其第 j 个分量的函数
q	2	一个素数幂
$\mathcal{Q}_{i,h}$	48	算术划分 $\mathcal{P}_{i,h} \wedge \mathcal{U}_i$ of C_i；起辅助作用在命题 3.3.4 的证明中发挥作用；此外， $\mathcal{P}_i := \mathcal{Q}_{i,H}$ $_i$
量子复杂度	75	none
查询算法	75	none
查询模型	75	none
r	7	在 f 或 g 下表示一个循环的周期点的函数；例外：作为指数使用于“r 阶分圆映射”（参见记号 r_i）
r_i	4	为 $i \in \{0,1,\ldots,d-1\}$ 定义；一个非负整数介于 0 和 $q-2$ 之间，与 f 相关；有 对于所有 $x \in C_i$， $f(x) = a_i x^{r_i}$
$r_{i,\ p}(\vec{u}_p)$	102	与 proj$_1$(par′ 相同$_{i,p}(\vec{u}_p)$)， p标记的 $\vec{r}_i$条目仪("")]//?$\vec{\pi}$)
r'_i	52	用于第 3.4 小节和第 5.2.3 小小节；某个 C_i 的陪集代表元

$r'_{i,\vec{u}}(\vec{k'})$	105	与 $\mathbb{Z}/s'_i\mathbb{Z}$ 模同余的唯一元素，且其满足。 $\mathcal{A}^{k'_p \mathfrak{d}_{i,p,\vec{u}}}_{i}{}_{,\,p}(r_{i,\,p}(\vec{u}_p))$ 模 p^{κ_p}，对于每个 $p \in \mathfrak{P}_i$；它是由公式给出 $\sum_{p\in\mathfrak{P}_i} \mathcal{A}^{k'_p \mathfrak{d}_{i,\,p\,\vec{u}\,,\,,\,p}(r_{i,\,p}(\vec{u}_p))\,\frac{s'_i}{p^{\kappa_p}}}_{p^{\kappa_p}}\left(\frac{i_{\vec{u}}}{\frac{s'_i}{p^{\kappa_p}}}\right)$ 逆
$\vec{r}$-可容许的 索引函数	36	none
~~元素(proj$_i$(par$'_{i,p}(\vec{u}_p)$))$_{p\in\mathfrak{P}_i}$ 中的 $\prod_{p\in\mathfrak{P}_i}\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$~~ $(\vec{u})^{\vec{r}_i}$	102	与.....相关的 $\vec{u}$; 这些元素是.....的代表元 自然（逐分量）作用的轨道 of $\langle \mathcal{A}_{i,p} :\ p \in \mathfrak{P}_i \rangle$ 于 $\prod_{p\in\mathfrak{P}_i}\mathbb{Z}/p^{\kappa_p}\mathbb{Z}$
$r^{\langle\psi\rangle}$	35	r 的 ψ-循环上的点集，是.....的特殊情形。 记号 x^G 表示 $x \in X$ 在自然作用下的轨道 $G \leq$对称群(X)的作用
$\mathfrak{r}$	29	表示原根的变量
$\mathfrak{r}_p$	99	模 $p^{\nu_p(m)}$ 的一个原根，其中 $m \in \mathbb{N}^+$ 显而易见。 上下文（在第 5.2.1 小节中，$m = s$）
$\mathcal{R}_i$	45	定义在 $i \in \{0, 1, \ldots, d-1\}$ 上；它是算术划分 $\bigwedge_{t=1}^{K} \mathcal{P}_{j_t}$ 其中 $j_1, j_2, \ldots, j_K$ 是 $\overline{f}$-暂态原像 在 $\overline{f}$ 下的 i；如果 i 本身是 $\overline{f}$-暂态的，那么 $\mathcal{R}_i = \mathcal{P}_i$
$\mathfrak{R}$	137	表示 I 型或 II 型树寄存器的变量
递归树 描述列表	73	none
右正则 表示	20	none
刚性繁衍	15	none
有根树	2	对我们而言，与“有向根树”同义
rt(Δ)	16	Δ 的根
s	8	C 的群阶，即 $s = (q-1)/d$
s'_i	102	$\prod$ 的简写 $_{p\in\mathfrak{P}_i} p^{\kappa_p}$
s''_i	123	s/s'_i 的简写
$\mathfrak{s}$	35	表示有限.....的移位（加法平移）的变量 阿贝尔群；尤其参见记号 $\mathfrak{s}_{\vec{r}}$ 的引入 问题 2.3.7 之后
S_n	136	类型 I 和类型 II 树寄存器中使用的记号；它包含了 寄存器中的树在 Γ_f 中出现的信息
$S_{n,i}$	73	出现在分区树寄存器的定义中；它是 一个特定的逻辑符号元组，或其集合，或由此类集合构成的元组 ；它们共同编码了这些信息 $\mathcal{P}_i$ 中的哪个块对应于哪棵有根树
$\overline{S}_{n,i}$	124	第 5.2.3 小小节中使用的一种特殊记号； $S_{n,\,i}$ 在.....下的逐元素像 $\vec{\nu'} \mapsto \vec{\nu'}^{-}$

$S_{n,\ \mathrm{per}}$	137	S_n 在 II 型树寄存器中的部分；它存储了相关信息 其有根树同构 $i \in \{0, 1, \ldots, d-1\}$ 类型 $\mathfrak{I}_n$ 出现在 C_i中周期长度为 f的顶点之上
$S_{n,\mathrm{trans}}$	137	是 II 型树寄存器中 S_n 的一部分；它存储信息 对于 $i \in \{0, 1, \ldots, d-1\}$ 有根树同构 类型 $\mathfrak{I}_n$ 出现在 C_i 中的 f-暂态顶点之上
$\mathcal{S}_{i,h}$	46	定义当 $i \in \mathrm{per}(\overline{f}) \setminus \{d\}$ 且 $h \in \{0, 1, \ldots, H_i\}$；它是一个 某种算术划分 C_i，使得对于 $x \in C_i$ 和 $\mathfrak{h}(x) = h$，同构类型 $\mathrm{Tree}^{\Gamma_f}(x, C_{i'})$ 仅取决于 $\mathcal{S}$ 所在的块 $_{i,h}$ 其中包含 x
半直积	21	none
SF($\mathfrak{I}$)	53	$\mathfrak{I}$ 的简化形式
简化的	53	none
简化形式	53	none
由.....张成的	11	none
特殊类型 I	136	不要与“I 型”混淆
特殊类型 II	136	不要与“II型”混淆
$\mathrm{Stab}_G(x)$	186	当 G 是作用于集合 X 和 $x \in X$ 上的群时定义； 它表示 G 中 x 的稳定子
状态空间	2	none
STRAG($X, \mathcal{G}$)	194	TRAG($X, \mathcal{G}$) 的底层有向图，其中没有边 标签且无多重边 $x \to x'$ 对于固定的 x、$x' \in X$
后继	15	synonym: 子节点
后继代	15	none
有根树之和 同构类型	42	none
简化后的总和 边加权有根 树同构类型	54	none
对称群(n)	8	与对称群($\{0, 1, \ldots, n-1\}$)相同
对称群(X)	8	X上的对称群
同步	151	none
t	186	循环群 $\mathbb{Z}/m\mathbb{Z}$的平移 $x \mapsto x + 1$ （m 须根据上下文明确）
T	72	一个形式变量，用于多项式环
$\mathcal{T}_m$	50	$\mathbb{Z}/m\mathbb{Z}$ 的平凡划分（所有块都是单元素集）
$\mathcal{T}_{i,h}$	48	算术划分$\mathcal{S}_{i,\ h} \wedge \mathcal{U}_i$ 的 C_i；起辅助作用 命题 3.3.4 证明中的作用
$\mathfrak{T}$	186	由 t 生成的对称群($\mathbb{Z}/m\mathbb{Z}$)的循环子群
张量积 （有向图的）	13	none

TRAG$(X, \mathcal{G})$	194	与集合 X 相关联的变换图和 函数集 $\mathcal{G}$ $X \to X$
暂态	2	同义用法： f 下的暂态； f-暂态
转移函数	2	none
平移数	186	none
树$_i(\mathcal{P}_i, B)$	52	在 $B = \mathcal{B}(\mathcal{P}_i, \vec{\nu})$ 时定义； 与树$_i(\mathcal{P}_i, \vec{\nu})$相同
树$_i(\mathcal{P}_i, \vec{\nu})$	12	与该块相关的有根树同构类型 $\mathcal{B}(\mathcal{P}_i, \vec{\nu})$ 的 $\mathcal{P}_i$
树$_i(\mathcal{P}, M, \vec{\nu})$	41	当 $x \in C_i$ 只依赖于 $\mathsf{Tree}^{\Gamma_f}(x, M)$ 时定义的 包含 x 的 $\mathcal{P}$-块（在此对应关系下 的 C_i 与 $\mathbb{Z}/s\mathbb{Z}$ 通过 ι_i）； 它表示共同的根 树同构类型树$^{\Gamma_f}(x, M)$ 对于 $x \in \mathcal{B}(\mathcal{P}, \vec{\nu})$
树$_i(\mathcal{P}, \vec{\nu})$	41	树$_i(\mathcal{P}, \mathcal{B}(\mathcal{P}, \vec{\nu}))$的简写； .....的自然扩展 将记号树$_i(\mathcal{P}_i, \vec{\nu})$推广到任意算术运算 对 C_i的划分
树$_i(\mathcal{P}'_j, C_j, \vec{\nu}^{(\mathcal{P}'_j)})$	46	当 $i \in \{0, 1, \ldots, d-1\}$ 和 j 是一个 $\overline{f}$ 暂态时定义 $\overline{f}$下 i 的原像； 它表示公用的 对于 $x \in \mathcal{B}(\mathcal{P}'_j, \vec{\nu}^{(\mathcal{P}'_j)}) \subseteq C_i$， 其同构类型为 $\mathsf{Tree}^{\Gamma_f}(x, C_j)$
树$_i(\mathcal{R}_i, \bigcup_{t=1}^{K} C_{j_t}, \vec{\nu}^{(\mathcal{R}_i)})$	46	当 $i \in \{0, 1, \ldots, d-1\}$ 和 j_t 是 $\overline{f}$下 i的 $\overline{f}$ 暂态原像； 它表示公用的 同构类型 $\mathsf{Tree}^{\Gamma_f}(x, \bigcup_{t=1}^{K} C_{j_t})$， 适用于 $x \in \mathcal{B}(\mathcal{R}_i, \vec{\nu}^{(\mathcal{R}_i)}) \subseteq C_i$
树$_i^{(h)}$ $(\mathcal{P}_{i,h}, \vec{\nu}^{(\mathcal{P}_{i,h})})$	49	定义于 $i < d$ 为 $\overline{f}$-周期的且 $h \in \{0, 1, \ldots, H_i\}$； 一种有根树同构类型， 所有 $\mathsf{Tree}^{\Gamma_f}(x)$ 均属于该类型 对于 $x \in \mathcal{B}(\mathcal{P}_{i,\ h}, \vec{\nu}^{(\mathcal{P}_{i,h})}) \subseteq C_i$， 当 $\mathfrak{h}(x) = h$ 相等时
树$_i^{(h)}$ $(\mathcal{S}_{i,\ h}, C_{i'}, \vec{\nu}^{(\mathcal{S}_{i,h})})$	49	当 $i < d$ 是 $\overline{f}$-周期的且 $h \in \{0, 1, \ldots, H_i\}$ 时定义； 一种有根树同构类型， 所有 $\mathsf{Tree}^{\Gamma_f}(x, C_{i'})$ 都属于该类型 对于 $x \in \mathcal{B}(\mathcal{S}_{i,\ h}, \vec{\nu}^{(\mathcal{S}_{i,h})})$， $\subseteq C_i$ 若 $\mathfrak{h}(x)$， 则 $= h$ 相等
树$_\Gamma(x)$	8	Γ 中上述的树 x
树$_\Gamma(x, M)$	41	类似于树$_\Gamma(x)$， 但删除所有子图树$_\Gamma(y)$ 其中 $y \to x$ 和 $y \notin M$
上面的树 x 在 Γ	8	none
树项链列表（相对于 有根树序列 同构类型）	145	none
树项链列表（相对 到树寄存器)	145	none
I 型	123	不要与“特殊类型 I”混淆 or “I 型树寄存器”
I 型树寄存器	136	none

II型	123	不要与“特殊类型 II”混淆 or “II 型树寄存器”
II 型树寄存器	136	none
III型	124	none
u	100	组成 $\vec{u}$的至多两个参数中的第一个
u_p	102	构成 $\vec{u}_p$的最多两个参数中的第一个
u'	100	构成 $\vec{u}$的至多两个参数中的第二个
u'_p	102	组成 $\vec{u}_p$的至多两个参数中的第二个
$\vec{u}$	100	参数集 Y_i 中元素的一般形式 p （取决于 关于 p，在此记号中省略；另见 $\vec{u}_p$）
$\vec{u}_p$	102	标记为 p 的分量的 $\vec{u}$；它是 Y_i的一个元素$_{,p}$
$\vec{\vec{u}}$	102	参数集 $\overline{Y}_i$ 中元素的一般形式
$\mathfrak{u}_{i,L}$	165	函数 $\mathcal{O}_{i,\ L} \to \mathcal{O}_{i',\ L-1}$ 将每个元组映射 $\diamond_{t=0}^{H_i+L}\vec{o_t} \in \mathcal{O}_{i,\ L}$ 对应唯一元组 $\diamond_{t=0}^{H_i+L-1}\vec{o_t} \in \mathcal{O}_{i',\ L-1}$ 使得对于每个（f-周期的）点 $x \in \mathcal{B}(\mathcal{Q}_{i,\ H_i+L}, \diamond_{t=0}^{H_i+L}\vec{o_t} \diamond \vec{\xi}_{i,\ H_i})$，唯一的 f-周期的 $x^{(-1)} \in C_{i'}$ 在 f 下的 x 的原像位于块 $\mathcal{B}(\mathcal{Q}_{i',\ H_i+L-1}, \diamond_{t=0}^{H_i+L-1}\vec{o_t} \diamond \vec{\xi}_{i',\ H_i})$
U_i	71	为 $\overline{f}$-周期的 i 定义，设其周期长度为 ℓ；它是.....的并集 $\bigcup_{t=0}^{\ell\ 1} C_{i_t}$ 对于 i 的 $\overline{f}$ 循环上的所有指标 j，块 C_j
U_χ	77	与 χ 相关联的相位反转算子
$\mathcal{U}_i$	47	当 $i \in \{0, 1, \ldots, d-1\}$ 是 $\overline{f}$-周期的时定义；它是 C_i 的算术划分 $\mathfrak{P}(\theta_{i,h}(x)\colon\ h = 1,\ 2,\ \ldots,\ H_i)$，其块是由具有以下特征的点组成的 C_i 子集 具有相同 $\mathfrak{h}$ 值的点
v	14	表示素因数分解中指数的变量
$v_{p,k}$	100	ν 的简写 $\mathfrak{p}_{p,k}(p-1)$
$v'_{i,p}$	100	$\nu_p(\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i))$ 的简写
$v'_{i,p,k}$	100	ν 的简写 $\mathfrak{p}_{p,k}(\mathrm{ord}_{p^{\kappa_p}}(\overline{\alpha}_i))$
$v''_{i,2}$	99	简写为 $\nu_2(\mathrm{ord}_{2^{\kappa_2}}(-\overline{\alpha}_i)$
$\mathfrak{v}_i$	42	在 $\Gamma_{\overline{f}}$ 中严格位于 i 上方的顶点数； 换言之，$\mathfrak{v}_i = \lvert\mathrm{V}(\text{树}_{\Gamma\ \ \ \overline{f}}(i))\rvert - 1$
V	15	表示有向图顶点集的变量
$\mathrm{V}(\Gamma)$	9	Γ的顶点集
$\mathcal{V}_{i,L}$	166	C_i的算术划分 $\mathfrak{P}(\eta_{i,\ l}(x) : l \in \mathfrak{C}_{i,\ L})$； 如果 C_i 中所有 f-周期点的循环长度 至多为 L，那么 $\mathcal{V}_{i,\ L}$ 是唯一的算术划分 C_i，使得 $\mathcal{V}_{i,\ L}$ 的一个块由所有 f-暂态 C_i 中的点组成，而 $\mathcal{V}_{i,\ L}$ 的其他每个块由 所有具有相同 f-周期长度的 f 周期点
w	18	表示同一底集元素的变量 如 x、y、z

$\mathfrak{w}$	53	表示边权的变量
$\mathfrak{w}_k$	141	引理 5.3.2.2(4)证明中使用的记号；对于固定的 $i \in \{0,1,\ldots,d-1\}$ 和 $t \in \mathbb{Z}$，它表示权重 $\mathfrak{I}_{i_t}$图示中从左起第 k条边的，k 在第 4.1 小节中
W	11	朗伯 W 函数
$\mathcal{W}_{i,L}$	166	算术划分 $\mathcal{V}_{i,L} \wedge \mathcal{Q}_{i,H_i+L-1}$ 的 C_i；如果全部 C_i 中的f周期点的 f周期长度至多为 L，则 对于每个 f 周期的 $x \in C_i$，其中的 $\mathcal{W}_{i,L}$-块，x 所包含的内容决定了.....的同构类型。 包含 x 的 Γ_f 的连通分支
印 wreath 积	8	none
x	2	基集中的一个元素，例如 X 或 $\mathbb{F}_q$，或一个变量 在同余式中，具体含义取决于上下文
x_n	3	一个形式变量，用于表示循环类型
$x^{(t)}$	164	对 $x \in \mathrm{per}(f)$ 和 $t \in \mathbb{Z}$ 有定义；表示 $(f_{\mid \mathrm{per}(f)})^t(x)$
$\mathfrak{x}$	9	$\mathfrak{X}$的一个元素
$[\vec{\mathfrak{x}}]$	9	与有限序列 $\vec{\mathfrak{x}}$相关的循环序列 在 $\mathfrak{X}$上；也写作 $[\mathfrak{x}_0,\ldots,\mathfrak{x}_{L-1}]$ 如果 $\vec{\mathfrak{x}} = (\mathfrak{x}_0,\ldots,\mathfrak{x}_{L-1})$
$\lvert\vec{\mathfrak{x}}\rangle$	76	当 $\vec{\mathfrak{x}} \in \{0,1\}^n$ 时定义；它是编码 $\vec{\mathfrak{x}}$ 的 n 量子比特寄存器； 物理学家将这个对象称为右矢
X	2	一个集合（通常假设为有限集）
X^X	8	所有函数的变换半群 $X \to X$
$\mathcal{X}$	42	表示算术划分的变量；例如用于 枚举指标（$\mathcal{X}_1$、$\mathcal{X}_2$、...），以避免与 记号 $\mathcal{P}_i$
$\mathfrak{X}$	9	用于表示一个被视为字母表的集合的变量， 有限序列由该集合中的元素构成
y	9	表示与 x 属于同一基础集合的元素的变量
Y_i	98	par_i、par'_i 和par''_i的定义域
$Y_{i,p}$	99	par'_i的定义域，p
$\overline{Y}_i$	102	参数集 $\prod_{p \in \mathfrak{P}_i} Y_{i\,p}$
$\mathcal{Y}$	187	表示算术划分的变量（如 $\mathcal{X}$）
$\mathfrak{Y}$	128	（用于定理 5.3.1.1）一个特殊的素数集合
z	13	表示与.....属于同一基集中的一个元素的变量 x 或 y
$\mathcal{Z}_i$	73	出现在分区树寄存器的定义中； 一种经过专门定义、能够编码足够信息的对象 重构 $\mathcal{P}_i$ （有时更多）
$\mathbb{Z}/m\mathbb{Z}$	2	模 m 的剩余环，其底层集合 $\{0,1,\ldots,m-1\}$
$(\mathbb{Z}/m\mathbb{Z})^*$	29	$\mathbb{Z}/m\mathbb{Z}$ 的单位乘法群

α_i	40	A_i的线性系数
$\overline{\alpha}_i$	45	$\mathcal{A}$的线性系数 $_i$
$\overline{\alpha}_{i,h}$	46	$\mathcal{A}_i$的线性系数 $_{,h}$
$\overline{\alpha}'_i$	175	$\mathcal{A}'$的线性系数 $_i$
β_i	40	常数系数的 A_i
$\overline{\beta}_i$	99	常数系数的 $\mathcal{A}_i$
$\overline{\beta}_{i,h}$	46	$\mathcal{A}_i$ 的常数系数 $_{,h}$
$\overline{\beta}'_i$	175	常数系数的 $\mathcal{A}'$ $_i$
$\overline{\beta}''_i$	175	$(\mathcal{A}'_i)$ 的常数系数 $\operatorname{ord}_{s'_i}(\overline{\alpha}'_i)$
γ	11	欧拉–马歇罗尼常数
Γ	2	表示一个（通常是有限）有向图的变量
$[\Gamma]_{\cong}$	16	Γ 的有向图同构类型
Γ^*	15	Γ 的对偶有向图
Γ_g	2	g 的函数图
$\Gamma_f^{(i)}$	71	$f_{\lvert U_i}$ 的函数图
Γ_{per}	12	的诱导子图 Γ_f 在$\bigcup_{i\in \mathrm{per}(f)} C_i$
Δ	16	表示有限有向有根树的变量
ζ	28	表示函数的循环的变量
$\eta_{i,l}(x)$	166	某个 s 同余式，在 l 是 ℓ 的倍数的情况下（i 的 $\overline{f}$-周期长度）刻画了 f 周期点 $x \in C_i$ 与 $f^l(x) = x$
$\theta_{i,h}(x)$	46	一个刻画 $x \in \mathbb{Z}/s\mathbb{Z}$ 何时成立的 s-同余式位于 $\mathcal{A}_{i,\ h}$ 的像中
$\Theta(x)$	47	单变量 x 中的 m-CC 系统
$\Theta_{i,h}(x)$	47	一个（至多两个） s-CC 的系统，用于刻画何时 $x \in C_i \cong \mathbb{Z}/s\mathbb{Z}$ 的 $\mathfrak{h}$值为 h
ι	35	表示同构或双射的变量；参见其中。特别是问题 2.3.7 后引入的记号 $\iota_{\vec{r}}$
ι_i	8	双射 $\mathbb{Z}/s\mathbb{Z} \to C_i$, $x \mapsto \omega^{i+dx}$
$\kappa_{i\,p}$	99	$\nu^{(\kappa_p}$的简写$)_p$ $(\overline{\beta}_i)$
κ_p	99	简写表示$\nu_p(s)$
$\kappa_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'}, J)$	25	技术参数, used in the definitionof $\sigma_{\mathcal{P},A}(\vec{\nu},$ $\vec{\nu'})$
$\lambda(\mathcal{P}, A)$	47	如 $\mathfrak{P}'(\mathcal{P}, A)$，删除最后一个生成同余式
$\lambda_i^t(\mathcal{P})$	48	在 $i < d$ 是 $\overline{f}$-周期的、 $t \in \mathbb{N}_0$ 且 $\mathcal{P}$ 是一个时定义 C_i的算术划分；这是算术划分的 C_{i_t} 通过 $\lambda_i^0(\mathcal{P})$ 递归定义：$= \mathcal{P}$ 以及 $\lambda_i^{t+1}(\mathcal{P}) := \lambda(\lambda_i^t(\mathcal{P})\ A_{i_t})$
Λ	28	表示提升函数 $\mathbb{Z}/m_1\mathbb{Z} \to \mathbb{Z}/m_2\mathbb{Z}$ 的变量，其中 m_1、 $m_2 \in \mathbb{N}^+$，其中 $m_1 \mid m_2$（即对于所有 x， $\Lambda(x) \equiv x \pmod{m_1}$ $x \in \mathbb{Z}/m_1\mathbb{Z}$);其定义的具体细节因上下文而异

μ_a	14	群 $\mathbb{Z}/m\mathbb{Z}$ 的端同态 $x \mapsto ax$； m 必须根据上下文明确其含义。
ν	10	表示逻辑符号的变量（$\emptyset$ 或 $\neg$）
$\nu_{l,l'}$	167	（用于小小节 5.3.3）逻辑符号的记号； 它被定义为 $\emptyset$（正逻辑符号），如果 $l \mid l'$，并且 如 $\neg$ 否则
$\vec{\nu}_{i,L,l}$	167	逻辑符号元组 $(\nu_{l,l'})_{l' \in \mathfrak{C}_{i,L}}$；如果所有 f-周期点 在 C_i 中具有 f-周期长度至多为 L 且 $l \in \mathfrak{C}_{i,L}$，则 $\mathcal{B}(\mathcal{V}_{i,L}, \vec{\nu}_{i,L,l})$ 恰好由那些 f-周期的 $x \in C_i$ 组成 其 f-周期长度恰好为 l
$\vec{\nu}^{+}$	124	第 5.2.3 小小节中使用的特殊记号
$\vec{\nu'}^{-}$	124	第 5.2.3 小小节中使用的一种特殊记号； 记号 $\vec{\nu}^{+}$的逆命题
$\nu_p(n)$	14	n 的 p-进赋值
$\nu_p^{(v)}(n)$	14	定义为 $\min\{v, \nu_p(n)\}$
$\vec{\xi}_{i,k}$	47	当 $i \in \{0, 1, \ldots, d-1\}$ 是 $\overline{f}$-周期的时定义，且 $k \in \{0, 1, \ldots, H_i\}$；它是长度为 H_i 的逻辑符号元组 其中恰好前 k 项等于正数 逻辑符号；块 $\mathcal{B}(\mathcal{U}_i, \vec{\xi}_{i,k})$ 仅由那些 $x \in C_i$ 其 $\mathfrak{h}$ 值为 k的
$\pi(m)$	36	m 的素因子集合
ρ_{l}	20	群 G 的左正则表示（显然来自 作用于自身的上下文），即函数 $G \to \mathrm{Sym}(G)$, $x \mapsto (y \mapsto xy)$
ρ_{r}	20	群 G 的右正则表示（显然由 context) 在其自身上，即函数 $G \to$ 对称群(G), $x \mapsto (y \mapsto yx)$
σ	11	约数和函数
$\sigma_{\mathcal{P},A}(\vec{\nu}, \vec{\nu'})$	24	一个技术参数，其意义请参见 关于 $\mathfrak{P}'(\mathcal{P}, A)$ 的注释
$\tau(m)$	132	$m \in \mathbb{N}^+$ 的（正）因数个数
ϕ	29	欧拉 φ 函数
φ	20	表示群自同态的变量
φ_i	40	定义于 $i \in \{0, 1, \ldots, d-1\}$，满足 $a_i \neq 0$；它是 μ_{α_i}, 与 A_i 相关联的 $\mathbb{Z}/s\mathbb{Z}$ 群自同态
χ	77	表示特征函数的变量
ψ	3	X的一个置换
ω	4	$\mathbb{F}_q$的一个本原元

表 11：本文所用记号和术语的表格概览。