arXiv:2304.08860v1 [cs.DM] 18 Apr 2023

重新审视商环上的快速傅里叶乘法算法

RamiroMart´ınez *Paz Morillo 加泰
罗尼亚理工大学`ecnica deCatalunya

2023年4月

摘要

本文对多项式商环（如 $\mathbb{Z}_m[x]/\langle x^n - a\rangle$）中多项式的高效快速傅里叶变换乘法算法进行了形式化，其中 n为2的幂， m 为不一定是素数的整数。我们还深入研究了这些乘法算法适用所需的必要条件和／或充分条件。本文统一了在这些商环中高效计算两个多项式乘积的不同方法。

关键词：快速傅里叶变换、多项式商环、乘法算法、理想格

1 引言

为系数取自环 R 的多项式构造高效乘法算法一直是一个广泛的研究领域。给定两个次数分别不超过 n 的多项式 $g(x),h(x) \in R[x]$，以及 $g(x) = \sum_{i=0}^{n-1} g_i x^i$ 和 $h(x) = \sum_{i=0}^{n-1} h_i x^i$，以朴素方式计算其乘积（即教科书式乘法算法），

$$(g \cdot h)(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (g_i \cdot h_j)\, x^{i+j},$$

需要进行二次数量的运算，即 n^2 次环 R 中元素的乘法。

*电子邮件： ramiro.martinez@upc.edu

$$
\begin{array}{rrrrrrrr}
 & & & & g_3x^3+ & g_2x^2+ & g_1x+ & g_0 \\
 & & & \times & h_3x^3+ & h_2x^2+ & h_1x+ & h_0 \\
\hline
 & & & & g_3\cdot h_0x^3+ & g_2\cdot h_0x^2+ & g_1\cdot h_0x+ & g_0\cdot h_0 \\
 & & & g_3\cdot h_1x^4+ & g_2\cdot h_1x^3+ & g_1\cdot h_1x^2+ & g_0\cdot h_1x & \\
 & & g_3\cdot h_2x^5+ & g_2\cdot h_2x^4+ & g_1\cdot h_2x^3+ & g_0\cdot h_2x^2 & & \\
 & g_3\cdot h_3x^6+ & g_2\cdot h_3x^5+ & g_1\cdot h_3x^4+ & g_0\cdot h_3x^3 & & & \\
\hline
\sum_{i+j=6} g_ih_jx^6+ & \sum_{i+j=5} g_ih_jx^5+ & \sum_{i+j=4} g_ih_jx^4+ & \sum_{i+j=3} g_ih_jx^3+ & \sum_{i+j=2} g_ih_jx^2+ & \sum_{i+j=1} g_ih_jx+ & g_0h_0 &
\end{array}
$$

如果现在要在环 $R[x]/\langle f(x)\rangle$ 中进行运算，教科书式乘法算法所执行的运算次数相同，不过要考虑到最后还需要进行约化。

对于特定情形 $f(x) = x^n \pm 1$，显然该算法执行的运算次数相同，因为我们只需将 ∓ 1 all$i$ 从 n 到 $2n-2$ 的系数相乘，并将它们加到相应的 $i-n$ 列中。

本工作的目标是形式化并统一一些用于构建更高效乘法算法的概念，重点研究环 $R[x] = \mathbb{Z}_m[x]$，继而研究其商环 $\mathbb{Z}_m[x]/\langle x^n+1\rangle$；其中 n 是 2 的幂，且 m 不一定是素数（尽管有时我们也会考虑 x^n-1 或一般情形 x^n-a）。

我们选择研究这一特定环的优化，因为密码学构造广泛使用它，并以理想格 [10], 的安全性为基础；这些理想格可以与 $\mathbb{Z}_m[x]/\langle x^n+1\rangle$ 中的理想对应起来。

大多数文献通常通过给出一套可用于某些特定环的方案来处理这一问题（考察 m 是否为素数，或 x^n-a 如何分解），却没有说明这些方案中施加的条件是必要条件还是仅仅充分条件。这种细节上的缺失可能会增加应用这些方案的难度。本文则从全面且数学化的角度分析哪些基本性质能够带来计算加速，使读者能够理解这些技术，并区分内在性质与多余的惯例。

因此，本文所提出的分析将帮助读者在使用商环中的多项式乘法算法时避免混淆 $\mathbb{Z}_m[x]/\langle x^n+1\rangle$。

在其他环上工作时，本文提出的许多想法都是约定俗成知识，例如 $\mathbb{C}[x]$ 或 $\mathbb{Z}_q[x]$，其中 q 是满足某些条件的素数。不过，进行全面分析或许能有助于弄清这些想法何时能够完全或部分推广到我们关注的环上，以及对底层环提出的条件究竟是必要条件还是仅仅充分条件。

1.1 相关工作

1.1.1 Karatsuba 乘法算法

第一个次二次乘法算法由 Karatsuba 于 [8], 提出，其代价为 $\mathcal{O}\left(n^{\log 3}\right)$，该算法源自一种巧妙的分而治之策略。我们在此介绍这一算法，因为我们的方法会将 Karatsuba 算法用作子程序。

设 $g(x)$ 和 $h(x)$ 是两个次数严格小于 n （2 的幂）的多项式。我们可以将这两个多项式拆分为高次部分和低次部分： $g(x) = g_U(x)x^{n/2} + g_L(x)$ 和 $h(x) = h_U(x)x^{n/2} + h_L(x)$，其中 g_L、 g_U、 h_L、 h_U 的次数均小于 $n/2$。

一种朴素的计算方法是

$$\begin{aligned} g(x) \cdot h(x) &= (g_U(x)x^{n/2} + g_L(x))(h_U(x)x^{n/2} + h_L(x)) \\ &= (g_U(x) \cdot h_U(x))\, x^n \\ &\quad + (g_U(x) \cdot h_L(x) + g_L(x) \cdot h_U(x))\, x^{n/2} \\ &\quad + g_L(x) \cdot h_L(x). \end{aligned}$$

这样，我们就把整个乘法拆分为四次规模为 $n/2$ 的多项式乘法。然而可以看出，这并未提高计算效率。设 $T(n)$ 为使用此方法计算乘积所需的运算次数。所得递推关系 $T(n) = 4T(n/2) + \mathcal{O}(n)$ 表明（根据分治递推关系的主定理 [3]），$T(n) = \mathcal{O}\left(n^2\right)$。

Karatsuba 的巧妙之处在于，他注意到交叉项可以由其他项以及一次 $n/2$-多项式乘法得到。也就是说，我们可以将项 $(g_U(x) \cdot h_L(x) + g_L(x) \cdot h_U(x))$ 写成

$$(g_L(x) + g_U(x))(h_L(x) + h_U(x)) - (g_U(x) \cdot h_U(x)) - (g_L(x) \cdot h_L(x))\,.$$

算法 1： KARATSUBA

输入：两个次数不超过 n 的多项式 $g(x)$ 和 $h(x)$

结果： $g(x) \cdot h(x)$ 的乘积

1 如果 n $= 1$，则返回 $g \cdot h$;

将 $g(x)$ 和 $h(x)$ 拆分为 $g_L(x)$ $g_U(x)$ $h_L(x)$ $h_U(x)$。

2 $a(x) :=$ KARATSUBA($g_U(x)$ $h_U(x)$)

3 $b(x) :=$ KARATSUBA($g_L(x)$ $h_L(x)$)

4 $c(x) :=$ KARATSUBA($g_L(x) + g_U(x)$ $h_L(x) + h_U(x)$)

5 返回 $a(x)x^n + (c(x) - a(x) - b(x))x^{n/2} + b(x)$

注意，现在这个递推式只计算 3 个规模为一半的乘积，并且只需要线性数量的运算（$T(n) = 3T(n/2) + \mathcal{O}(n)$），从而实现了所需的次线性运行时间 $T(n) = \mathcal{O}\left(n^{\log 3}\right)$（通过主定理求解分治递推式 [3]）。

1.1.2 模 $f(x)$ 的 Karatsuba 乘法算法

不过请注意，在模 $x^n \pm 1$ 下进行运算时，这并不是书写这种递归最自然的方式，因为除最后一次递归调用需要进行约化外，其余递归调用的处理方式都相同。

或者，我们可以将 $g(x)$ 表示为 $= g_1(x^2)x + g_0(x^2)$，其中 g_0 包含偶数次项的系数， g_1 包含奇数次项的系数。这样，我们可以将次数小于 n 的 $g(x)$ 视为 $g(x, y) = g_0(y) + g_1(y)$，其 x 次数为 1，且 y 次数小于 $n/2$。这称为双重 Karatsuba，其思路保持不变：

$$\begin{aligned} g(x) \cdot h(x) &= (g_1(x^2)x + g_0(x^2))(h_1(x^2)x + h_0(x^2)) \\ &= \left(g_1(x^2) \cdot h_1(x^2)\right) x^2 \\ &\quad + \left(g_1(x^2) \cdot h_0(x^2) + g_0(x^2) \cdot h_1(x^2)\right) x \\ &\quad + g_0(x^2) \cdot h_0(x^2) \end{aligned}$$

同样地，正如我们之前所做的，我们可以只增加一次乘法来表示第二项。$\left(g_1(x^2) \cdot h_0(x^2) + g_0(x^2) \cdot h_1(x^2)\right)$ 是：

$$\left(\left(g_1(x^2) + g_0(x^2)\right) \cdot \left(h_1(x^2) + h_0(x^2)\right) - g_1(x^2) \cdot h_1(x^2) - g_0(x^2) \cdot h_0(x^2)\right).$$

算法 2： KARATSUBA （模 $x^n \pm 1$）

输入：两个次数不超过 n 的多项式 $g(x)$ 和 $h(x)$

结果： $g(x) \cdot h(x)$ 的乘积

如果 n $= 1$ 则返回 $g \cdot h$;

将 $g(x)$ 和 $h(x)$ 拆分为 $g_0(x)$ $g_1(x)$ $h_0(x)$ $h_1(x)$。

$a(y) :=$ KARATSUBA$(g_1(y)$ $h_1(y))$

$b(y) :=$ KARATSUBA$(g_0(y)$ $h_0(y))$

$c(y) :=$ KARATSUBA$(g_0(y) + g_1(y)$ $h_0(y) + h_1(y))$

返回 $a(x^2)x^2 + (c(x^2) - a(x^2) - b(x^2))x + b(x^2)$（模 $x^n \pm 1$）

现在，在递归算法的每一层都可以对 $x^n \pm 1$ 进行模约简，因为再次进行变量替换后， $\widehat{g}(y) \cdot \widehat{h}(y)$（模 $y^{n/2} \pm 1$）等价于 $\widehat{g}(x^2) \cdot \widehat{h}(x^2)$（模 $x^n \pm 1$）。这并未带来计算优势，但有助于我们从不同角度理解它。

请注意， $g_0(y) + g_1(y) \equiv g(x, y)$ 模 $x - 1$ 以及 $g_0(y) \equiv g(x, y)$ 模 x。文献 [4] 探讨了这些思想，以便将这些工具都视为同一框架的一部分。

1.1.3 更快的乘法算法

通过更巧妙的递推关系，可以得到速度更快的算法：以递归方式将多项式映射到另一个域（递归地计算两个规模减半的变换 $T(n) = 2T(n/2) + \mathcal{O}(n)$），在该域中，多项式可以

高效地在线性时间内相乘，因此最终的计算复杂度为 $\mathcal{O}(n \log n)$。本文将重点系统地探讨基于快速傅里叶变换（快速傅里叶变换）范式的算法，并将在后续章节中详细介绍这一范式。在有限域上使用时，这种方法通常称为数论变换（数论变换）。

快速傅里叶变换递推的主要思想归功于高斯，并由 Cooley 和 Tukey 在其开创性著作中充分发展 [6] ，该著作研究了复数环。

此后，人们提出了许多变体，将 [6] 推广到非 2 的幂次的有界次数多项式，或提供了其他技巧和解读方式，让我们从中受益，例如 [7]。一篇内容详尽、文档记录极为出色的综述见 [4]。

大多数研究都聚焦于这样一种特定情形：在$\mathbb{Z}_q[x]/\langle x^n+1\rangle$ 中进行乘法，其中 q 是素数，且 x^n+1 完全分裂为一次因子。对此，人们已从软件 [9] 和硬件 [14] 两个角度研究了一段时间。

我们尤其关注 [12], 中提出的思想，因为该文专门讨论了在 $\mathbb{Z}_q[x]/\langle x^n+1\rangle$ 中进行多项式乘法的情形：此时 x^n+1 不能完全分裂为一次因子（某些基于格的密码方案会出现这种情况，例如某些承诺方案 [2]），因而只能部分应用标准快速傅里叶变换。然而， [12] 只考虑了底数为素数的情况 q ，并简要介绍了相关步骤。

这种部分应用快速傅里叶变换的技术有时称为不完全数论变换 [5] ，通常被解释为中国剩余变换（CRT），如 [11]进行快速中国剩余分解 [17]时那样。

然而，大多数文献只给出一些充分条件，用以支持某种特定实现或快速乘法算法的特定抽象，而这些条件无法直接推广。尽管如此，我们仍提出了一个更通用的乘法框架 $\mathbb{Z}_m[x]/\langle x^n+1\rangle$ ，让读者能够从数学上严谨、同时又便于不熟悉代数构造的读者理解的角度，明白为什么一些约定俗成的假设确实必要，而另一些则并非如此。

1.2 记号与约定

由于我们的目标是在 $R[x]/\langle f(x)\rangle$ 中运算，其中 f 是一个首一多项式，因此我们选择 $g(x) \in R[x]/\langle f(x)\rangle$ 除以 $f(x)$ 所得的余式作为其代表元，记作 $g(x) \bmod f(x)$。

我们用小写粗体罗马字母表示向量，并用 log 表示二进制对数。

我们的记号大多借鉴自 [4], ，并在第 3 节和第 4 节中提出了一些新的定义，我们认为这些定义本身也具有独立的研究价值。

2 逐点乘积

任何快速傅里叶变换乘法技术的核心思想，都是通过计算环 R 中某些点处的取值的逐点乘积，来求两个多项式的乘积。

根据多项式乘积的定义，显然，对于给定的点 $x_0 \in R$，乘积的取值等于各因子的取值之积（$g \cdot h$）（x_0）$= g$（x_0）$\cdot h$（x_0）。

这样，如果我们计算足够多的取值（将此变换记为 T），就可以对这些取值进行逐点乘积运算（记为 $\odot$），然后再将多项式插值还原（T^{-1}）。

$$\begin{array}{ccccc} (R[x])^2 & \xrightarrow{\odot \circ T} & R^k & \xrightarrow{T^{-1}} & R[x] \\ (g(x), h(x)) & \mapsto & \begin{pmatrix} g(x_0) \\ g(x_1) \\ \vdots \\ g(x_{k-1}) \end{pmatrix} \odot \begin{pmatrix} h(x_0) \\ h(x_1) \\ \vdots \\ h(x_{k-1}) \end{pmatrix} & \mapsto & (g \cdot h)(x) \end{array}$$

在接下来的章节中，我们将探讨 R、k 和 x_0、x_1、.....、x_{k-1} 有哪些可能的取值，使得 T^{-1} 有明确定义，且 T 和 T^{-1} 都能高效计算。我们将刻画求值点为能够执行这些运算所必须满足的充要条件。

2.1 T 的一般可逆性

为处理变换 T 在次数不超过 n 的多项式上的可逆性问题，我们注意到它是一个线性映射，并通过其对应矩阵对其进行刻画

$$V = \begin{pmatrix} 1 & x_0 & {x_0}^2 & \cdots & {x_0}^{n-1} \\ 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k-1} & {x_{k-1}}^2 & \cdots & {x_{k-1}}^{n-1} \end{pmatrix}.$$

这种特殊矩阵称为范德蒙德矩阵。如果我们选择 $k = n$，就会得到一个方形范德蒙德矩阵，并可以讨论它的可逆性。

由于我们处理的是一般的含幺交换环 R，矩阵可逆当且仅当其行列式可逆 [16]。范德蒙德矩阵的行列式很容易计算，其形式为

$$\det(V) = \prod_{0 \le i < j < n} (x_j - x_i).$$

现在，如果 R 是域，只需选择 n 个不同的求值点 x_0，.....， x_{n-1}。

如果 R 只是一个交换环，那么选择点使其差在 R 中可逆，是一个充要条件。

条件 1（差可逆的点）。如果一组点中任意两点之差在 R 中都可逆，我们就说这组点满足条件 1。

备注 1。在 R 中选择差可逆的 n 求值点，是变换 T 可逆的充要条件。

请注意，这里讨论的是对某个次数的多项式进行变换和逆变换。给定两个多项式 $g(x)$ 和 $h(x)$，其次数分别为 n 和 n'。由于我们要恢复乘积 $g(x) \cdot h(x)$，因此应将它们视为次数小于或等于 $n+n'$ 的多项式，并使用 $(n+n'+1) \times (n+n'+1)$ 范德蒙德矩阵。

2.2 模 $f(x)$ 的取值逐点乘积

给定 $a \in R$[1]，我们总可以将 $R[x]/\langle x^n - a\rangle$ 中的多项式视为 $R[x]$ 中次数严格小于 n 的多项式（使用典范代表元 rem）。如前所述，通过 $2n$ 变换计算它们在 $R[x]$ 中的乘积，该乘积是次数严格小于 $2n$ 的多项式；然后再次应用 rem $x^n - a$，得到次数小于 n 的代表元。

使用逐点取值乘积技术计算两个多项式模 $f(x)$ 的乘积时，我们面临的主要问题是，取值通常并非良定义，因为它取决于我们从模 $f(x)$ 的等价类中选择的代表元。

一般而言，对于任意的 x_0，计算 $(g(x) \text{ rem } f(x))(x_0) \cdot (h(x) \text{ rem } f(x))(x_0)$，与计算 $(g(x) \cdot h(x) \text{ rem } f(x))(x_0)$ 并不相同。

因此，我们必须选择一些特定点，使取值与同余类相容。如果 α 满足这样的条件，即任意两个等价的多项式

[1]可以看出，在特殊情形 $a = \pm 1$ 下， $R[x]/\langle x^n - 1\rangle$ 和 $R[x]/\langle x^n + 1\rangle$ 中的乘积在文献中通常称为循环/反循环卷积（记作 $*$）。

$$
\begin{aligned}
(g \cdot h)(x) \pmod{x^n - 1} &= g(x) \cdot h(x) \pmod{x^n - 1} \\
&= \left(\sum_{i=0}^{n-1} g_i x^i\right)\left(\sum_{j=0}^{n-1} h_j x^j\right) \pmod{x^n - 1} \\
&= \sum_{k=0}^{n-1} \left(\sum_{\substack{i,j \\ i+j \equiv k \\ \bmod n}} g_i \cdot h_j \right) x^k \\
&= (g * h)(x)
\end{aligned}
$$

这种直觉本身也可能值得关注，因为常规域中的卷积乘积在变换域中就是逐点乘积。

多项式 $g(x) \equiv \hat{g}(x)$（模 $f(x)$）可得 $g(\alpha) = \hat{g}(\alpha)$，于是必有$f(\alpha) = 0$，且 α必须是 $f(x)$ 的一个根。

令 α 为 $f(x)$ 的一个根。则 $g(x) \equiv \hat{g}(x)$（模 $f(x)$）意味着 $g(\alpha) = \hat{g}(\alpha)$，因此

$$(g(x) \text{ rem } f(x))(\alpha) \cdot (h(x) \text{ rem } f(x))(\alpha) = (g(x) \cdot h(x) \text{ rem } f(x))(\alpha).$$

总之，对于 $f(x) = x^n - a$ 这一特定情形，其中 $a \in R, \alpha$ 必须是 a 的 n 次根。通过选取 $\alpha_0, \ldots, \alpha_{n-1}$ 个互不相同的 a 的 n 次根，且它们的差可逆，我们便能够直接从这些根在 $\alpha_0, \ldots, \alpha_{n-1}$ 中的取值的逐点乘积，恢复 $g(x) \cdot h(x)$ 模 $x^n - a$ 的结果。

这减少了所需的计算次数，因为只需要 n 个求值点，且无需进一步约化即可得到典范代表元。

条件 2（作为点的 $f(x)$ 的根）。如果一组点是 $R[x]/\langle f(x)\rangle$ 中某个多项式的根，则称该组点满足条件 2。

注 2. 选择 n 次根作为求值点，是确保 $R[x]/\langle x^n - a\rangle$ 中多项式的取值定义良好的必要且充分条件。

目前，条件 1 和 2 是将取值的逐点乘积用作计算 $R[x]/\langle x^n - a\rangle$ 中两个多项式乘积的技术的必要且充分条件。下一步是研究如何高效地完成这一计算。

3 高效变换

本节将从理论和渐近的角度定义高效变换协议和逆变换协议。其他实现技巧或方法（例如，以迭代还是递归的方式求解递推关系）可能会显著节省空间或计算量，但不在本文讨论范围之内。

3.1 高效计算 T 的取值

了解了在各种相关情形下逐点乘积取值能够成立的条件后，我们必须讨论如何高效地应用这些条件。

逐个计算每个 n 取值需要 $\mathcal{O}(n)$次运算，总计为$\mathcal{O}\left(n^2\right)$。快速傅里叶方法通过分治递归策略，同时计算 n 取值，效率更高。

如果我们称 x_i 为其中一个求值点，并像双重 Karatsuba 那样，将多项式$g(x)$分解为两个规模减半、分别由偶数次和奇数次系数组成的多项式 $g_0(x)$ 和 $g_1(x)$，那么可以写成

$$g(x_i) = g_0(x_i^2) + x_i \cdot g_1(x_i^2).$$

通过这种递归，我们将一次次数不超过 n 的多项式求值，化为两次次数不超过 $n/2$的多项式求值、一次在 R 中的乘法以及一次在 R中的加法。直接这样做并不能节省任何开销，因为每次求值仍需 $\mathcal{O}(n)$。

核心思路是选择求值点 $\{x_0, x_1, \ldots, x_{n-1}\}$，使得包含其平方的集合 $\{y \mid y = {x_i}^2\}$ 只包含 $n/2$ 个元素（因此，我们可以重复使用 g_0 和 g_1 在这些平方上的取值）。我们希望这一性质能够递归成立，因此引入以下定义，该定义已满足条件 2。

定义 1（ n 次根的二重集合）。元素 $a \in R$ 的 n 次根的一个带索引集合 $\alpha_0, \ldots, \alpha_{n-1} \in R$（必要时可适当重新编号），其中 n 是 2 的幂，称为 n 次根的二重集合，若 $i \equiv j \pmod{2^{\log(n)-k}}$ 蕴含：对于从 0 到 $\log(n)$ 的 k，均有 ${\alpha_i}^{2^k} = {\alpha_j}^{2^k}$。

我们可以像图 1 那样用一棵满二叉树直观地表示它。请注意，树的叶子（即求值点）按位反转顺序排列。

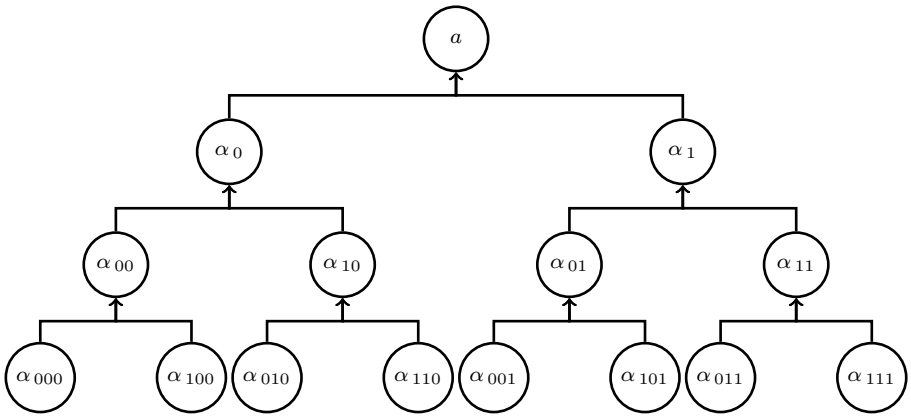

图 1：8 次根的二重集合

注 3. 给定 $\alpha_0, \ldots, \alpha_{n-1}$，它们是 $a \in R$ 的 n 次根的二重集合，则集合 $\{{\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2\}$ 是 a 的 $n/2$ 次根的二重集合。

选取根 $\alpha_0, \ldots, \alpha_{n-1}$ 的二重集合作为求值点，对于每个 $0 \le i < n/2$，等式 ${\alpha_i}^2 = {\alpha_{i+n/2}}^2$ 成立，我们可以写成

$$\begin{aligned} g(\alpha_i) &= g_0({\alpha_i}^2) + \alpha_i \cdot g_1({\alpha_i}^2) \\ g(\alpha_{i+n/2}) &= g_0({\alpha_i}^2) + \alpha_{i+n/2} \cdot g_1({\alpha_i}^2). \end{aligned}$$

我们可以用它来给出对快速傅里叶变

形式（快速傅里叶变换）算法 3。

算法 3： FFT

输入：次数不超过 n 的多项式 $g(x)$，以及一个二重集合 $\alpha_0, \ldots, \alpha_{n-1}$，由 a 的 n 次根组成

$g(x)$ 在 $\alpha_0, \ldots, \alpha_{n-1}$ 处的取值

若 $n = 1$，则返回 g;

将 $g(x)$ 拆分为 $g_0(x)$ 和 $g_1(x)$。

$\boldsymbol{y}_0 \coloneqq \mathrm{FFT}(g_0(x)\ {\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2)$

$\boldsymbol{y}_1 \coloneqq \mathrm{FFT}(g_1(x)\, {\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2)$

返回 $\boldsymbol{y}_0 + \boldsymbol{y}_1 \odot (\alpha_0, \ldots, {}^{\alpha_{n/2-1}}) || \boldsymbol{y}_0 + \boldsymbol{y}_1 \odot ({}^{\alpha_{n/2}}, \ldots, \alpha_{n-1})$

现在分析其计算成本，我们发现，计算 n-快速傅里叶变换所需的时间，与计算两次 $n/2$-快速傅里叶变换加上在 $R(T(n) = 2T(n/2) + \mathcal{O}(n))$. 中进行线性数量的乘积和加法所需的时间相同。再次使用 [3]，最终得到总成本 $\mathcal{O}(n \log n)$。请注意，这里指的是变换的总成本，而不是像之前那样每次取值的成本。

条件 3（根的二重集合）。如果一个点集是二重集合，我们就说它满足条件 3。

备注 4。将求值点选为 n 次根的二重集合，是存在高效快速傅里叶变换的充分条件。

注意，根据定义，条件 3 蕴含条件 2。不过，我们倾向于将其单独处理，因为它足以支持高效实现，但对于一般的逐点乘积方法而言并非必需。

我们将使用以下记号。对于任意 $i \in \{0, 1, \ldots, n-1\}$，令 $\overline{\imath} = i + n/2$ 除以 n所得的余数为 $\overline{\imath} = i + n/2 \operatorname{rem} n$。对于 j 和 $\overline{\jmath}$，记法类似。

3.2 T^{-1} 的高效求值

为了获得高效乘法算法，我们不仅需要高效的变换算法，还需要高效的逆变换算法。

请预先注意：正如我们在注 3 中看到的，条件 3 在对求值点平方后仍然成立，我们也应检查条件 1 是否同样如此。为此，我们首先需要以下引理 1。

引理 1. 设 α_0，…，α_{n-1} 是 a 的 n 阶根组成的二重集合，且它们的差均可逆。

那么 $\alpha_{\overline{\imath}} = -\alpha_i$。

证明。由于该集合满足条件 3，我们知道 ${\alpha_i}^2 = {\alpha_{\overline{\imath}}}^2$。也就是说

$$0 = {\alpha_i}^2 - {\alpha_{\overline{\imath}}}^2 = (\alpha_i - \alpha_{\overline{\imath}})(\alpha_i + \alpha_{\overline{\imath}}).$$

利用这些元素之间的差可逆，我们得到 $\alpha_i + \alpha_{\overline{\imath}} = 0$。 □

命题 2（满足条件 1 至 3 的集合的平方也满足条件 1 至 3）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 a 的 n 次根的二重集合，且其差可逆。

那么，集合 ${\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2$ 是 a 的 $n/2$ 次根构成的二重集合，且其差可逆。

证明。根据二重集合的定义，平方构成的集合 ${\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2$ 是 a 的 $n/2$ 次根构成的二重集合。

我们只需检查这些平方之间的差是否仍然可逆。

$$ {\alpha_i}^2 - {\alpha_j}^2 = (\alpha_i - \alpha_j)(\alpha_i + \alpha_j) = (\alpha_i - \alpha_j)(\alpha_i - \alpha_{\overline{j}}). $$

根据引理 1，我们已知这些平方之间的差是原始元素之间差的乘积；根据假设，这些差可逆，因此这些平方也满足相应条件。

□

完成这些准备后，可以从构造的角度，通过反向执行变换算法或显式求出变换矩阵的逆元来定义逆变换。不过，为了更深入地理解，我们将用拉格朗日插值的语言来描述它。

给定一个差可逆的带索引的点集 $\{\alpha_i\}_{i=0}^{n-1}$，以及多项式在这些点上的取值 $\{g(\alpha_i)\}_{i=0}^{n-1}$，我们可以使用拉格朗日多项式 $l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x)$ 恢复原多项式 $g(x)$，如下所示：

$$ g(x) = \sum_{i=0}^{n-1} g(\alpha_i) l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x), \quad l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n-1} \frac{x - \alpha_j}{\alpha_i - \alpha_j}. $$

关键在于注意到，由于我们的求值点集合具有特殊要求，即条件 1 至 3，我们的拉格朗日多项式以一种特殊方式进行因式分解。

利用引理 1，我们可以看出拉格朗日多项式是如何分解的。

$$ \begin{aligned} l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) &= \prod_{\substack{j=0 \\ j \neq i}}^{n-1} \frac{x - \alpha_j}{\alpha_i - \alpha_j} \\ &= \frac{x - \alpha_{\overline{i}}}{\alpha_i - \alpha_{\overline{i}}} \prod_{\substack{j=0 \\ j \neq i \\ (\mathrm{mod}\ n/2)}}^{n/2-1} \left(\frac{x - \alpha_j}{\alpha_i - \alpha_j}\right)\left(\frac{x - \alpha_{\overline{j}}}{\alpha_i - \alpha_{\overline{j}}}\right) \\ &= \frac{x - \alpha_{\overline{i}}}{\alpha_i - \alpha_{\overline{i}}} \prod_{\substack{j=0 \\ j \neq i \\ (\mathrm{mod}\ n/2)}}^{n/2-1} \frac{x^2 - {\alpha_j}^2}{{\alpha_i}^2 - {\alpha_j}^2} \\ &= l_i^{\{\alpha_i, \alpha_{\overline{i}}\}}(x) l_{i \bmod n/2}^{\{{\alpha_j}^2\}_{j=0}^{n/2-1}}(x^2). \end{aligned} $$

需要特别注意，对于二重集合 $l_{\overline{i}}^{\{\alpha_j{}^2\}_{j=0}^{n/2-1}}{}_{\text{余 } n/2}(x) = l_i^{\{\alpha_j{}^2\}_{j=0}^{n/2-1}}{}_{\text{余 } n/2}(x)$。

然后我们可以写成

$$\begin{aligned} g(x) &= \sum_{i=0}^{n-1} g(\alpha_i) l_i^{\{\alpha_j\}_{j=0}^{n-1}}(x) \\ &= \sum_{i=0}^{n-1} g(\alpha_i) l_i^{\{\alpha_i,\alpha_{\overline{i}}\}}(x) l_{i \bmod n/2}^{\{\alpha_j{}^2\}_{j=0}^{n/2-1}}(x) \\ &= \sum_{i=0}^{n/2-1} \left(g(\alpha_i) l_i^{\{\alpha_i,\alpha_{\overline{i}}\}}(x) + g(\alpha_{\overline{i}}) l_{\overline{i}}^{\{\alpha_i,\alpha_{\overline{i}}\}}(x) \right) l_i^{\{\alpha_j{}^2\}_{j=0}^{n/2-1}}(x^2). \end{aligned}$$

一旦得到这种分解，这种表述方式的优势就在于它能让我们对其进行解释。我们将 $g(x)$ 视为 $g_0(x^2) + xg_1(x^2)$。多项式 $l_i^{\{\alpha_j{}^2\}_{j=0}^{n/2-1}}(x)$ 可以帮助我们对 $g_0(x)$ 和 $g_1(x)$ 进行插值，前提是我们知道它们在 $\{g_0({\alpha_i}^2)\}$ 和 $\{g_1({\alpha_i}^2)\}$处的取值。然而，我们已知的取值是 $\{g(\alpha_i)\}$。但此时，$l_i^{\{\alpha_i,\alpha_{\overline{i}}\}}(x)$ 和 $l_{\overline{i}}^{\{\alpha_i,\alpha_{\overline{i}}\}}(x)$ 正是对 $g_0({\alpha_i}^2) + xg_1({\alpha_i}^2)$ 进行插值的多项式（一个次数为 1、其系数为所需取值的多项式），这些取值来自 $g(\alpha_i)$ 和 $g(\alpha_{\overline{i}})$。

由于我们稍后会用到它，下面明确说明 ate

$$g(\alpha_i) l_i^{\{\alpha_i,\alpha_{\overline{i}}\}}(x) + g(\alpha_{\overline{i}}) l_{\overline{i}}^{\{\alpha_i,\alpha_{\overline{i}}\}}(x) = \alpha_i \left(\frac{g(\alpha_i) + g(\alpha_{\overline{i}})}{\alpha_i - \alpha_{\overline{i}}} \right) + x \left(\frac{g(\alpha_i) - g(\alpha_{\overline{i}})}{\alpha_i - \alpha_{\overline{i}}} \right).$$

这可用于构造一种高效插值算法。根据多项式 n 在 $\{\alpha_i\}_{i=0}^{n-1}$ 处的取值 g，其中其次数至多为 n，我们可以求出多项式 g_0 和 g_1 在 $\{{\alpha_i}^2\}_{i=0}^{n/2-1}$ 处的取值（这是通过对次数至多为 2 的多项式进行插值来完成的，因此每次只需常数时间，总共需要 $\mathcal{O}(n)$ 次运算）。接着，我们利用这些取值对 g_0 和 g_1 进行插值，它们都是次数至多为 $n/2$、属于 $R[x]/\langle x^{n/2} - a \rangle$ 的多项式。也就是说，$T(n) = 2T(n/2) + \mathcal{O}(n)$，最终我们得到

再次达到 $T(n) = \mathcal{O}(n \log n)$。

算法4: IFFT

输入: 大小为 n 的求值向量 $\boldsymbol{y}$ 和一个二重集合

$\alpha_0, \ldots, \alpha_{n-1}$，由具有可逆差的 n 次单位根组成

结果: 在 $g, \ldots$ 处插值的多项式 $x(\boldsymbol{y})$ 的系数

$\alpha_0, \ldots, \alpha_{n-1}$

若 $n = 1$ 则返回 y;

对于 $i \leftarrow 0$ 到 $n/2 - 1$，执行

$\boldsymbol{y}_0[i] := \alpha_i \left(\frac{\boldsymbol{y}[i] + \boldsymbol{y}[i+n/2]}{\alpha_i - \alpha_{i+n/2}} \right)$

$\boldsymbol{y}_1[i] := \left(\frac{\boldsymbol{y}[i] - \boldsymbol{y}[i+n/2]}{\alpha_i - \alpha_{i+n/2}} \right)$

end

$g_0(x) := \mathrm{IFFT}(\boldsymbol{y}_0, {\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2)$

$g_1(x) := \mathrm{IFFT}(\boldsymbol{y}_1, {\alpha_0}^2, \ldots, {\alpha_{n/2-1}}^2)$

返回 $g_0(x^2) + x g_1(x^2)$

3.3 在 $R[x]/\langle x^n - a \rangle$ 中的高效乘法算法

结合算法 3 和算法 4，我们在算法 5 中描述了一种在 $R[x]/\langle x^n - a \rangle$ 上的高效乘法算法。

算法 5: Efficient FFT Multiplication

输入: 两个次数不超过 n 的多项式 $g(x)$、$h(x)$

辅助: $\alpha_0, \ldots, \alpha_{n-1}$ 中由 a 的 n 次根构成的二重集合，且具有可逆差

结果: $R[x]/\langle x^n - a \rangle$ 中 $g(x)$ 与 $h(x)$ 的乘积 $(g \cdot h)(x)$

$\boldsymbol{g} := \mathrm{FFT}(g(x)\ \alpha_0, \ldots, \alpha_{n-1})$

$\boldsymbol{h} := \mathrm{FFT}(h(x)\ \alpha_0, \ldots, \alpha_{n-1})$

$\boldsymbol{f} := \boldsymbol{g} \odot \boldsymbol{h}$

$f(x) := \mathrm{IFFT}(\boldsymbol{f}, \alpha_0, \ldots, \alpha_{n-1})$

返回 $f(x)$

因此，我们的工作是研究满足条件 1 至 3 的求值点集合是否存在，以及如何在我们期望的 R中找到它们。

4 $\mathbb{Z}_m[x]/\langle x^n - a\rangle$中的合适求值点集的刻画

本节聚焦于 $\mathbb{Z}_m$，并研究给定条件之间的关系，以确认这些条件恰好是所需的概念，并给出存在适当求值集的必要充分条件。

我们可以先证明，条件 1 至 3 通常确实彼此独立。

命题 3（若 m 不是素数幂，则在 $\mathbb{Z}_m$ 中条件 3 不蕴含条件 1）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_m$ 中 a 的 n 次根构成的二重集合，其中 m 不是素数幂。则存在一个由 $\mathbb{Z}_m$ 中 a 的 n 次根构成的二重集合，其元素之差均不可逆。

证明。将 $m = pq$ 分解为互素的适当因子 p 和 q 后，我们总能构造另一个二重集合，使其模 p 下定义 $\alpha_i' \equiv \alpha_i$，但模 q 下不定义 $\alpha_i' \equiv \alpha_0$。它仍是一个二重集合，但其中任意两个元素之差都不可逆。 □

当模数是素数幂时，p^e 可逆性取决于模 p 下不相同，而模 p^e 下不相同通常并不能推出这一点。为处理这种特殊情况，我们首先需要进一步刻画 $\mathbb{Z}_{p^e}$ 中的 n 次根，其中 p 为素数。

定理 4（[13] 中定理 2.23 所述的亨泽尔引理）。设 $f(x)$ 是一个具有整数系数的多项式。若 $f(x_0) \equiv 0$（模 p^e），且 $f'(x_0) \not\equiv 0$（模 p），则存在唯一的 t（模 p），使得 $f(x_0 + tp^e) \equiv 0$（模 p^{e+1}）。

在我们的情形中，我们尤其关注 $f(x) = x^n - a$，其中 n 是 2 的幂，因此 $f'(x) = nx^{n-1}$。由于我们的解是 a 的 n 次根（按模 p^e 考虑时，因此按模 p 考虑时也如此），只要 $p \neq 2$ 和 $a \neq 0$，就有 $n{x_0}^{n-1} \not\equiv 0$（模 p）。

推论 5. p 为奇素数时，$\mathbb{Z}_{p^e}$ 与$\mathbb{Z}_p$中的 n 次根之间存在一一对应。

证明. 一方面，通过应用（模 p），可以看出 $\mathbb{Z}_{p^e}$ 中的每个根都是 $\mathbb{Z}_p$ 中的一个根。

另一方面，只需从 $\mathbb{Z}_p$ 到 $\mathbb{Z}_{p^e}$迭代应用定理 4 共 $e-1$ 次即可。

□

特别地，这意味着作为群元素的阶保持不变，因为根的幂可以唯一地提升为其提升后的根的幂。

注 5. 这里略去 $p = 2$ 的情形，因为在接下来的几节中我们将看到，其他条件会排除这一特殊情形。

命题 6（在 $\mathbb{Z}_m$ 中，若 m 是奇素数幂，则条件 2 蕴含条件 1）。设 $m = p^e$，其中 p 是奇素数，并设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 n 个不同的 n 次根，它们属于 $\mathbb{Z}_{p^e}$中的 a。那么对于所有 $i \neq j$，$\alpha_i - \alpha_j$ 在 $\mathbb{Z}_{p^e}$ 中可逆。

证明。每个 α_i 在模 p意义下都是 $f(x) = x^n - a$ 的根。根据前一个推论，我们有 $\alpha_i \not\equiv \alpha_j \pmod{p^e}$ 蕴含 $\alpha_i \not\equiv \alpha_j \pmod{p}$。因此，最大公因数 $(\alpha_i - \alpha_j,\ p) = 1$，正如我们所希望的。 □

命题 7（条件 3 蕴含条件 1，若 m 是奇素数幂）。设 $m = p^e$，其中 p 是奇素数，并设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_{p^e}$ 中 a 的 n 次根构成的二重集合。那么对于所有 $i \neq j$，$\alpha_i - \alpha_j$ 在 $\mathbb{Z}_{p^e}$ 中可逆。

证明。条件 3 蕴含条件 2，而条件 2 蕴含条件 1。 □

对于任意模数 m，其素数分解为 $m = {p_1}^{e_1} \ldots {p_k}^{e_k}$，我们可以通过中国剩余定理（中国剩余定理），根据 $d_{(i)}$ 完全确定任意 $d \in \mathbb{Z}_m$，使得

$$\begin{aligned} d_{(1)} &\equiv d \pmod{{p_1}^{e_1}} \\ d_{(2)} &\equiv d \pmod{{p_2}^{e_2}} \\ &\vdots \\ d_{(k)} &\equiv d \pmod{{p_k}^{e_k}}. \end{aligned}$$

利用这一表示，我们可以证明以下定理。

定理 8（当且仅当条件 2 模每个 ${p_j}^{e_j}$ 成立时，条件 1 和 2 成立）。设 $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{Z}_m$ 是 $a \in \mathbb{Z}_m$ 的不同 n 次根构成的集合，并设 $m = {p_1}^{e_1} \ldots {p_k}^{e_k}$ 是奇模数 m 的素数分解。

当且仅当将这些元素分别模任意一个 ${p_j}^{e_j}$ 考察时，它们仍两两不同时，这些元素之间的差在模 m 下可逆。

证明。证明思路与前面的命题相同，因为一个元素可逆，当且仅当它模其模数的因式分解中的所有互素因子都可逆；并且我们已经看到（推论 5），这类根模 ${p_j}^{e_j}$ 两两不同时，当且仅当它们模 p_j 两两不同，因此它们的差可逆。

□

同理，我们也可以看到，其他一些蕴含关系一般也不成立。

命题 9（当 $\mathbb{Z}_m$ 中的 m 不是素数幂时，条件 1 和 2 不蕴含条件 3）。并非每个由具有可逆差的 $a \in \mathbb{Z}_m$ 的 n 次根构成的 n 集合，都是由 n 次根构成的二重集合。一般而言，这一结论并不成立，甚至对单位根也不成立。

证明。考虑 $\mathbb{Z}_{65}$ 中以下 4 次单位根：

$$\begin{aligned} 12^4 &\equiv 1 \pmod{65} & 12^2 &\equiv 14 \pmod{65} \\ 14^4 &\equiv 1 \pmod{65} & 14^2 &\equiv 1 \pmod{65} \\ 18^4 &\equiv 1 \pmod{65} & 18^2 &\equiv 64 \pmod{65} \\ 21^4 &\equiv 1 \pmod{65} & 21^2 &\equiv 51 \pmod{65} \end{aligned}$$

它们的差集为 $\{2,3,4,6,7,9\}$，其中所有元素都是模 65 下的可逆元素。 □

注意，例子中的取值点确实经过重新排列后，构成了 n次单位根的一个二重集合，分别位于 $\mathbb{Z}_5$ 和 $\mathbb{Z}_{13}$中，但两种排列方式不同。

$12^2 \equiv 18^2$	（模 5）	$12^2 \equiv 14^2$	（模 13）
$14^2 \equiv 21^2$	（模 5）	$18^2 \equiv 21^2$	（模 13）

再一次，如果我们在奇素数幂模下工作，该命题确实成立。为证明这一点，我们需要几个引理，它们揭示了单位根的重要作用，并使我们能够聚焦于单位根，以便更好地理解这些排列。

引理 10（条件 1 和 2 意味着求值点可逆）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_m$ 中互不相同的 n 次根，且对于所有 $i \neq j$，$\alpha_i - \alpha_j$ 在 $\mathbb{Z}_m$ 中可逆。那么对于所有 i，α_i 在 $\mathbb{Z}_m$ 中可逆。

证明。选取两个指标 i 和 j，并令 d 为 α_i 和 m 的一个无平方因子公因数。由于 ${\alpha_i}^n \equiv a$（模 m）且 ${\alpha_j}^n \equiv a$（模 m），我们有 $m | {\alpha_i}^n - {\alpha_j}^n$，这意味着 $d | \alpha_j$（这里需要用到 d 是无平方因子的这一点）。最终可得 $d | \alpha_i - \alpha_j$，但我们知道最大公因数$(\alpha_i - \alpha_j, m) = 1$，因此 $d = 1$，从而证明对所有 i，最大公因数$(\alpha_i, m) = 1$。 □

注 6。如果 a 的根可逆，则可直接推出 a 本身也必须可逆。我们会在需要时加以要求，因为这一论证表明，这是逆变换 T^{-1} 存在的必要条件。

引理 11（ a 的根和 1 的根）。设 $a \in \mathbb{Z}_m$。以下两个陈述等价：

(i) 集合 $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{Z}_m$ 满足条件 1 和 2。

(ii) 集合 $\alpha_0, \ldots, \alpha_{n-1} \in \mathbb{Z}_m$ 可以由 a 的一个可逆 n 次根构造而成，记作 α，以及 $\mathbb{Z}_m$ 中 n 个互不相同的 n 次单位根 $\omega_0, \ldots, \omega_{n-1}$ 构造而成，且这些单位根的差可逆，使得 $\alpha_i = \alpha\omega_i$。

证明。我们来证明两个方向：

- (i) $\Longrightarrow$ (ii)

  设 $\alpha_0, \ldots, \alpha_{n-1}$ 是满足这些条件的根。

  我们可以定义 $\alpha = \alpha_0$（由引理 10 可知其可逆）和 $\omega_i = \alpha_i \cdot {\alpha_0}^{-1}$。我们可以验证 ω_i 是单位根，且它们的差可逆

$$(\omega_i - \omega_j)^{-1} = \left(\frac{\alpha_i}{\alpha_0} - \frac{\alpha_j}{\alpha_0}\right)^{-1} = \alpha_0 \cdot (\alpha_i - \alpha_j)^{-1}.$$

- (ii) $\Longrightarrow$ (i)

  设 $\alpha, \omega_0, \ldots, \omega_{n-1}$ 是满足这些条件的一组根。

  现在定义 $\alpha_i = \alpha \cdot \omega_i$。根据构造，所有 α_i 都是 n 次 a 的根，且它们的差可逆，因为

$$(\alpha_i - \alpha_j)^{-1} = (\alpha \cdot \omega_i - \alpha \cdot \omega_j)^{-1} = \alpha^{-1}(\omega_i - \omega_j)^{-1}.$$

□

这促使我们定义一个充分条件；正如我们将要看到的，当 m 是奇素数幂时，该条件也是必要的。

定义 2（(α,ω)-集）。设 α 是一个可逆的 $a \in R$ 的 n 次根，ω 是 R 中一个阶为 n 的 n 次单位根，且其各次幂之差均可逆。则称定义为 $\alpha_i = \alpha\omega^i$ 的集合为一个 (α,ω)-集。

条件 4（(α,ω)-集）。若一组求值点经适当重排后，是某些 $\alpha, \omega \in R$ 对应的 (α,ω)-集，则称其满足条件 4。

注释 7。设 $m = {p_1}^{e_1} \ldots {p_k}^{e_k}$ 是模 m 的素数分解。一个 n 次单位根 $\omega \in \mathbb{Z}_m$ 可由 $\omega_{(i)} \equiv \omega \pmod{p_i^{e_i}}$ 确定，而 ω 的阶恰为 $\omega_{(i)}$ 在 $\mathbb{Z}_{p_i^{e_i}}$ 中各自阶的最小公倍数。由于我们选择 n 为 2 的幂，且每个 $\omega_{(i)}$ 的阶都整除 n，因此这个最小公倍数就是这些阶中的最大值。于是，要使 ω 成为一个阶为 n 的 n 次单位根，只需其中一个 $\omega_{(j)}$ 的阶为 n。然而，由于还需要保证其各次幂之差可逆，因此每个 $\omega_{(i)}$ 的阶都必须为 n。

命题 12（条件 4 蕴含条件 1 至 3）。 $\mathbb{Z}_m$中的一个 (α,ω)-集合是由 a 的 n次根构成的二重集合，且其差可逆。

证明。设 $\alpha_i = \alpha\omega^i$，并设 $m = {p_1}^{e_1}{p_2}^{e_2} \ldots {p_k}^{e_k}$为 m的素数分解。

我们可以先检验条件 1。由引理 11 的证明可知， n的一个 (α,ω)-次根集合满足条件 1，当且仅当 α 可逆且 $\omega^i - \omega^j$ 也可逆。

由于 a 在 $\mathbb{Z}_m$ 中可逆，因此 α 作为 a 的一个 n次根也必定可逆。否则，若对某个 j 有 $\alpha \equiv 0 \pmod{p_j}$，则对同一个$j$ 也有 $a \equiv 0 \pmod{p_j}$，而它也不会可逆，这与命题矛盾。

根据定义，第二个条件得以满足。因此，每个差在模 m下都可逆。

条件 2 可由 (α,ω)-集的构造推出。

那么，对于条件 3，令 $i \equiv j \pmod{2^{\log(n)-k}}$，并且不失一般性，假设 $i \geq j$，因此对于某个非负整数 c，有 $i = j + c \cdot 2^{\log(n)-k}$。那么

$$\alpha_i^{2^k} = \left(\alpha_0\omega^i\right)^{2^k} = \alpha_0^{2^k}\omega^{(j+c\cdot 2^{\log(n)-k})\cdot 2^k} = \alpha_0^{2^k}\omega^{j\cdot 2^k + c\cdot n} = \left(\alpha_0\omega^j\right)^{2^k} = \alpha_j^{2^k}.$$

□

这个条件非常方便，例如，正如我们之前提到的，它使我们能够通过范德蒙德矩阵，以紧凑的形式明确描述该变换：$(V)_{ij} = (\alpha\omega^i)^j$ 以及 $(V^{-1})_{ij} = \frac{1}{n}(\alpha^{-1}\omega^{-j})^i$。请注意，逆矩阵看起来是某个 $(\alpha^{-1}, \omega^{-1})$-集合的求值矩阵的转置乘以 $1/n$。如果我们在 $\mathbb{Z}_m[x]/\langle x^n - 1\rangle$ 中使用 $\alpha = 1$，转置就无关紧要，因为该矩阵是对称矩阵，而且直接变换所用的高效递归求值技术也可直接用于其逆变换。

然而，即使条件 4 蕴含条件 1 至 3，反过来一般也不成立。此外，我们可以看到，当 m 是素数幂时，在某些特殊情况下结论成立。

命题 13（若 m 是奇素数幂，则条件 1 和 2 在 $\mathbb{Z}_m$ 中蕴含条件 4）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_{p^e}$ 中一组具有可逆差的 n 次 a 根，其中 p 是奇素数。除重新排序外，这是一组位于 $\mathbb{Z}_{p^e}$ 中的 n 次 a 根的 (α, ω) 集。

证明。由引理 11 可推出，存在一个可逆的 α 和 $\omega_0, \ldots, \omega_{n-1}$，它们具有可逆差，且满足 $\alpha_i = \alpha \cdot \omega_i$。

由于 $\mathbb{Z}_p$ 是一个域，其乘法群是一个阶为 $p-1$ 的循环群。因此，$\mathbb{Z}_p$ 中的 n 次单位根的集合也是一个群；作为循环群的子群，它也必定是循环的。

众所周知，$x^n - 1$ 模 p 至多有 n 个解（参见 [13]中的定理2.6），因此，n 个根 ω_i（模 p）（由于 $\alpha\omega_i$（模 p）彼此不同，它们在 $\mathbb{Z}_{p^e}$中提升为不同的点）构成了单位根的整个循环群，因此由其中一个根生成。

由于 $\mathbb{Z}_p$ 中的 n次根与 $\mathbb{Z}_{p^e}$中的 n次根一一对应，原来的 $\omega_i \in \mathbb{Z}_{p^e}$ 也由一个阶为 n的 ω 生成。也就是说，存在一个置换 π 使得 $\omega_{\pi(i)} = \omega^i$。

经过由 π定义的重新排序后，集合$\alpha_{\pi(0)}, \alpha_{\pi(1)}, \ldots, \alpha_{\pi(n-1)}$ 是一个(α, ω)-集。□

命题14（条件 1 和 2 在 $\mathbb{Z}_m$ 中确实推出条件 3，若 m 是奇素数幂）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_{p^e}$ 中一组 n 次 a 根，且差可逆，其中 p 为奇素数。除重新排序外，这是一组 $\mathbb{Z}_{p^e}$中的 a 的 n次根二重集。

证明。由命题 13 可知，当 m 是奇素数幂时，条件 1 和 2 蕴含条件 4；而如命题12所示，条件 4 总是蕴含条件 3，这一点是直接的。□

这个新条件似乎是恰当的选择，而且快速傅里叶变换通常是从与此定义等价的构造引入的；但我们应先研究，当 m 不是奇素数幂时，将求值点限制为这个特定集合是否会减少计算 FFT 乘法的可选方案。

同样，条件 1 和 2 对每个 ${p_j}^{e_j}$ 模都成立，应当意味着条件 4 对每个 ${p_j}^{e_j}$ 模都成立，但置换可能不同。

在某些情况下，这些排列可以使条件 3 成立，而条件 4 不成立。

定理 15（条件 1 至 3 一般并不能推出条件 4，但能推出满足条件 4 的集合存在）。设 $\alpha_0, \ldots, \alpha_{n-1}$ 是 $\mathbb{Z}_m$ 中一个由 n 次单位根组成的二重集合，且其差可逆，其中 m 为奇数且不是素数幂，n 是大于 2^2 的 2 的幂。那么 a 在 $\mathbb{Z}_m$ 中有逆元；此外，存在一个由 n 次 a 的根组成、位于 $\mathbb{Z}_m$ 中且差可逆但不满足条件 4 的二重集合 $\alpha'_0, \ldots, \alpha'_{n-1}$，以及一个集合 $\alpha''_i = \alpha\omega^i$，其中 α 是 a 的 n 次根，ω 是阶为 n 的单位根，且其所有幂的差均可逆，也就是说，它是一个 (α, ω)-集合。

证明。根据引理 10，我们得到 a 可逆。

设 $m = {p_1}^{e_1} {p_2}^{e_2} \ldots {p_k}^{e_k}$ 是 m 的素数分解。

在 $\mathbb{Z}_{{p_j}^{e_j}}$ 中，a 的 n 次根（允许重新排序）构成唯一一个具有可逆差的二重集合。这是因为每个 a 的根模 ${p_j}^{e_j}$ 的值都完全确定：它们是从 $\mathbb{Z}_{p_j}$ 中 a 的唯一 n 个根提升到 $\mathbb{Z}_{{p_j}^{e_j}}$ 后得到的唯一元素。根据命题 13，它满足条件 4。我们唯一可以选择的是它们各自的排列顺序。

这个顺序在 $\mathbb{Z}_{{p_j}^{e_j}}$ 中无关紧要，但在考虑 $\mathbb{Z}_m$ 时变得重要，因为一旦我们固定模 ${p_1}^{e_1}$ 的顺序，其余 ${p_j}^{e_j}$ 各自不同的排列顺序就会在 $\mathbb{Z}_m$ 中产生不同的元素。

根据二重集合的定义，我们得到了图 1 中的树结构，以及这些点的特定记号（索引的位分解）。只有树结构保留了这种二重结构，因此唯一可能的重新排序，是交换某个节点的左右子节点。也就是说，选取 $b_{i-1} \ldots b_0$（一个内部节点），并将所有 $b_{\log(n)-1}$ 的 $\alpha_{b_{\log(n)-1} \ldots b_i \ldots b_0}$ 映射为 $\alpha_{b_{\log(n)-1} \ldots \overline{b_i} \ldots b_0}$，....., b_{i+1}，仍能保留这种结构。

例如，选择节点 α_1 和 α_{10}，我们会得到不同的排列，如图 2 所示。

这意味着，给定一个二重集合，恰好有 2^{n-1} 种可能的重排（因为树中恰好有 $n-1$ 个内部节点，而且每个节点都可以交换或不交换）。

当奇数 m 不是素数幂时，一旦我们固定模 ${p_1}^{e_1}$ 的根的阶，模 ${p_j}^{e_j}$ 的根的每种可能重排都会为其余的 j 生成一个新的、具有可逆差的二重集合，总共有 $2^{(k-1)(n-1)}$种可能。

另一方面，我们可以计算 (α, ω)-集合的数量。为此，重要的是要注意，(α, ω) 和另一对不同的 (β, ξ) 可以生成同一个集合（只是顺序不同）。若是如此，令 $\pi \in \mathfrak{S}_n$ 为满足 $\beta_i = \alpha_{\pi(i)}$的置换。令 $c = \pi(0)$，则 $\beta = \beta_0 = \alpha_{\pi(0)} = \alpha\omega^c$。令 $d = \pi(1) - \pi(0)$，则 $\xi = \beta_1/\beta_0 = \alpha_{\pi(1)}/\alpha_{\pi(0)} = (\alpha\omega^{\pi(1)})/(\alpha\omega^{\pi(0)}) = \omega^{\pi(1)-\pi(0)} = \omega^d$。

这使我们能够将置换 π 完全刻画为

$$\beta_i = \beta\xi^i = \alpha\omega^c(\omega^d)^i = \alpha\omega^{c+di} = \alpha_{c+di}.$$

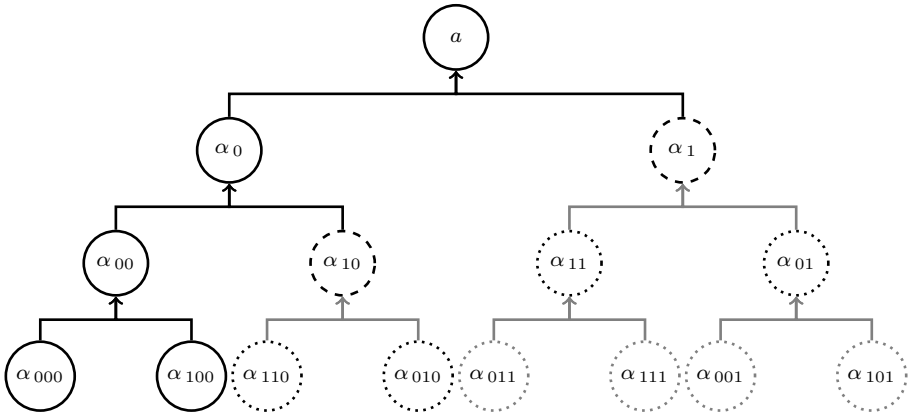

图 2：8 次单位根的二重集合，交换 α_1 和 α_{10} 的左、右后代

要使 π 成为置换， d 必须模 n 可逆，因此它必须是奇数。

这就是所需的全部内容，因为任意一对 $c \in \mathbb{Z}_n$ 和奇数 $d \in \mathbb{Z}_n$ 都会为同一集合生成一组新的生成元（$\alpha\omega^c$， ω^d）。也就是说，每个具有可逆差的（α， ω）集合，都可以由 $n^2/2$ 对 a 和 1 的不同根构造而成（ c 有 n 种选择， d 有 $n/2$ 种选择）。

因此，由具有可逆差的二重集合的存在性以及前述命题可知，在 $\mathbb{Z}_{p_j{}^{e_j}}$ 中有 n 个可能的 a 根，在 $\mathbb{Z}_{p_j{}^{e_j}}$ 中有 $n/2$ 个可能的 n 阶单位根；将它们通过中国剩余定理组合后，就能定义一个具有可逆差的（α''， ω''）集合。总共有 $n^{2k}/2^k$ 对生成元，而每个集合由 $n^2/2$ 对生成元定义，因此共有 $n^{2(k-1)}/2^{(k-1)}$ 个不同的集合。

然而，如果 $n > 2^2$，那么 $2^{(k-1)(n-1)} > n^{2(k-1)}/2^{(k-1)}$，因此，满足条件 1 至 3 的某些 $2^{(k-1)(n-1)}$ 集 $\{\alpha'_i\}_i$ 将不满足条件 4。

□

这一重要定理确保，即使条件 4 并非设计高效快速傅里叶变换乘法算法所必需的（文献中有时会间接地将其视为理所当然），在处理 $\mathbb{Z}_m[x]$ 时，我们也可以放心地假设条件 4 成立，因为相较于必要的条件 1 至 3，它不会对 m 增加任何限制。只有完成这项分析后，我们才能在方便时放心地使用 (α,ω)-集，而不会损失任何一般性。

4.1 $\mathbb{Z}_m$ 中合适的根的存在性与构造

一旦我们确立了利用该变换高效计算两个多项式模 $x^n - a$ 的乘积所需的条件，接下来就要研究我们所需的环中是否存在这样的点，以及如何找到它们。

正如我们在定理 15 中所见，存在满足条件 1 至 3 的合适点，意味着存在满足条件 4 的 (α,ω)-集。为方便起见，我们将刻画这类点集何时存在，以及如何找到它。

注 8. 当 $R=\mathbb{C}$ 时，我们只需选择 $\omega=e^{\frac{i2\pi}{n}}$ 和 $\alpha=\sqrt[n]{a}$（例如，当 $a=1$ 时选择 $\alpha=1$，或当 $a=-1$ 时选择 $\alpha=e^{\frac{i\pi}{n}}$）。这是标准的傅里叶变换，文献中通常会直接介绍它。

现在我们可以考虑 $R=\mathbb{Z}_m$ 的情况。

一方面，在 $\mathbb{Z}_m$ 中寻找一个本原 n 次单位根 ω，也就是说，寻找一个阶为 n 的 n 次单位根，使其所有幂之间的差都可逆；这意味着要在每个 $\mathbb{Z}_{{p_i}^{e_i}}$ 中寻找一个本原 n 次单位根 $\omega_{(i)}$，然后利用中国剩余定理重构 ω。

这样，我们就把在任意 $\mathbb{Z}_m$ 中寻找本原 n 次单位根的问题，化简为在 $\mathbb{Z}_{p^e}$ 中寻找本原 n 次单位根的问题，其中 m 任意。

在 [13] 中，定理 4 的证明明确说明了如何将模 p^e 的解 x_e 提升为模 p^{e+1} 的解 x_{e+1}。它可以通过递归计算：使用 $x_{e+1}\equiv x_e-f(x_e)f'(x_1)$（模 p^{e+1}），其中 $f'(x_1)$ 是将 $f'(x_1)$ 视为 $\mathbb{Z}_p$ 中的元素时的逆元。回想一下，$f(x)$ 是 x^n-a，因此 $f'(x)=nx^{n-1}$。

这项计算中唯一不显然的一步是计算 $f'(x_1)$。由于我们确定 $f'(x_1)$ 模 p 不为 0，因此可以使用扩展欧几里得算法计算整数 r 和 s，使得 $f'(x_1)$ $r+ps=$ 最大公因数$(f'(x_1)$ ~~$p)=1$~~，而 r 就是所需的 $f'(x_1)$。

最后，我们想分析在 p 和 n 满足哪些条件时，$\mathbb{Z}_p$ 中存在本原 n 次单位根，以及如何求出它们。我们从费马小定理 16 得到必要条件。

定理 16（费马小定理，引自 [13] 中的定理 2.7）。设 p 为素数。若 $p\nmid x_0$，则

$$x_0{}^{p-1}\equiv 1 \pmod{p}.$$

推论 17。若 $\mathbb{Z}_p$ 中包含一个阶为 n 的 n 次单位根，则 $n|p-1$。

证明。设 x_0 是 $\mathbb{Z}_p$ 中一个阶为 n 的 n 次单位根。由定理 16 可得 $x_0{}^{p-1}\equiv 1$ (模 p)，因此其阶整除 $p-1$，即对于某个整数 k，有 $n|p-1$ 或 $p=kn+1$。 □

备注 9. 利用这一条件，在特殊情形 $f(x)=x^n\pm 1$ 下，$f'(x_1)$ 的计算（其中 x_1 是 ∓ 1 模 p 的一个 n 次根）现在可以显式地计算为 $f'(x_1)=\pm x_1(p-1)/n$；通过计算 $f'(x_1)\cdot f'(x_1)$ 并得到 $(\pm x_1(p-1)/n)(n{x_1}^{n-1})\equiv 1$（模 p），即可轻松验证这一点。

备注 10. 这一必要条件排除了我们此前对 $n\not\equiv 0$（模 p_i）所作的所有额外考虑，例如确保 m 必须为奇数。注意，在算法4中，我们必须计算一个以 $\alpha_i-\alpha_{\bar{i}}$ 为除数的商，也就是说，$2\alpha_i$ 位于分母中。要求条件 1 至 3 始终成立，意味着环的两倍单位元必须可逆（由引理1和引理10可知）。

推论18. $\mathbb{Z}_p$ 包含阶为 n 的 n次单位根，当且仅当 $n|p-1$。

证明. 我们已经证明了一个方向，现在考虑满足 $n|p-1$的 $\mathbb{Z}_p$。我们知道乘法群 $\mathbb{Z}_p^*$ 是循环的。取一个生成元 g，它的阶为$p-1$，由于 $n|p-1$，我们可以选择 $\omega = g^{(p-1)/n}$，由构造可知，它将是一个阶为 n的 n次单位根。 □

备注 11。注意，根据狄利克雷算术级数定理，这种形式的素数有无穷多个。

现在我们可以开始计算 α （模 p_j）和 ω （模 p_j）。在 $\mathbb{Z}_{p_j}$ 中选取 u_j 一个二次非剩余，我们可以令 $\omega_{(j)} \equiv {u^{(p_j-1)}}_j{}^{/n} \in \mathbb{Z}_{p_j}$，然后再次利用定理 4 的构造性证明将其提升到 $\mathbb{Z}_{{p_j}^{e_j}}$。

首先请注意，目前尚无已知的确定性多项式时间算法能够找到二次非剩余（见 [1]）；然而，通过计算勒让德符号 $\left(\frac{u_j}{p_j}\right) \equiv {u_j}^{(p_j-1)/2}$，可以判断从 $\mathbb{Z}_{p_j}^*$ 中均匀随机选取的元素是否为二次非剩余，因为当且仅当它模 p 是二次非剩余时，该符号为 -1 （见 [13] 中的定理3.1），且成功概率接近二分之一。

另一方面，我们知道 $n|p_j - 1$ 是存在适当根的一个条件，因此 $\omega_{(j)} = {u_j}^{(p_j-1)/n}$ 是良定义的。其 n次幂为 ${\omega_{(j)}}^n = {u_j}^{p_j-1} = 1$ （再次由定理 16 得出）。

剩下唯一要做的就是检查它的所有幂之间的差都可逆。如果情况并非如此，那么 $\omega_{(j)}$ 的阶将为 k，且 $k < n$。这不可能成立，因为我们已经知道 ${\omega_{(j)}}^n = 1$，于是便会推出$k|n$；而如果 k 是严格小于 n 的2的幂，就会得到矛盾，因为根据构造，$-1 = {u^{(p_j-1)}}_j{}^{/2} = {\omega_{(j)}}^{n/2} = \left({\omega_{(j)}}^k\right)^{n/(2k)} = 1^{n/(2k)} = 1$。

这确保最终的 ω 满足所需条件。

为了计算 α，我们可能有不同的方法。如果 $a = 1$，平凡解 $\alpha = 1$ 完全满足我们的需求。如果 $a = -1$，我们可以采用类似的步骤。注意，由于 n 次根 -1 是 1 的 $2n$ 次根，现在的条件是 $2n|p_j - 1$，并令 $\alpha_{(j)} \equiv {u_j}^{(p_j-1)/2n}$ （模 p_j），再次将它们提升，并从其中国剩余定理表示中计算出最终的 α。

如果 $a \neq \pm 1$，那么我们的另一种选择是使用 Tonelli–Shanks

算法 [15]。

算法 6： TONELLI–SHANKS([15])

输入：一个奇素数 p、一个二次剩余 $a \in \mathbb{Z}_p$ 和一个非剩余 $u \in \mathbb{Z}_p$

结果：元素 $\alpha \in \mathbb{Z}_p$，即 a 的平方根

设 v 和 s，使得 $p-1 = v2^s$ 且 v 为奇数。

设

$k \leftarrow s$

$c \leftarrow u^v$

$t \leftarrow a^v$

$r \leftarrow a^{\frac{v+1}{2}}$

当 $t \neq 0 \wedge t \neq 1$ 时

求最小的 $i, 0 < i < k$, 使得 $t^{2^i} = 1$

令 $d \leftarrow c^{2^{k-i-1}}$ 并设

$k \leftarrow i$

$c \leftarrow d^2$

$t \leftarrow td^2$

$r \leftarrow rd$

end

如果 $t = 0$，那么

返回 $\alpha = 0$

end

如果 $t = 1$ 那么

返回 $\alpha = r$

end

该算法使我们能够在 $\mathbb{Z}_p$ 中计算二次剩余的一个平方根。请注意，该算法再次使用一个二次非剩余作为辅助元素。通过迭代应用该算法，我们可以利用 ${\alpha_{(j)}}^{n/2^i} \pmod{p_j}$ 中的 a 的 2^i-次根来计算其平方根 ${\alpha_{(j)}}^{n/2^{i+1}}$，直到最终得到 $\alpha_{(j)}$，再由此恢复 α。

再次回到 $x^n + 1$ 的情形，我们将详细描述计算的每一步

（α, ω）见算法 7。

算法 7: COMPUTATION OF α AND ω

输入: 2 的幂 n 和模数 m （已知其因式分解）

结果: -1 和 1 的适当（α, ω）根

令 $m = {p_1}^{e_1} \dots {p_k}^{e_k}$ 为 m的素数分解。

1 确保每个素数都满足 $2n | p_j - 1$, 否则终止。

对于 $j \leftarrow 1$ 到 k 执行

2 /* 在 $\mathbb{Z}_{p_j}$ 中取得一个二次非剩余 */

$tests$ =False; // 判断候选数是否为非剩余

当 $tests$ 不成立时执行

$u_j \xleftarrow{R} \mathbb{Z}_{p_j}$; // 选择一个非剩余候选数

如果 $u^{(p_j-1)/2} \equiv -1$ (模 p_j) 则

$tests$ = True; // 它是一个非剩余

end

end

3 /* 计算 α 和 ω (模 ${p_j}^{e_j}$) */

$\alpha_{(j)} = u_j^{(p-1)/2n}$; // 计算 α （模 p_j）

$\omega_{(j)} = u_j^{(p-1)/n}$; // 计算 ω （模 p_j）

$c, aux = \text{EXTEUCLIDES}\ (n{\alpha^{2n-1}}_j)\ p_j$; // 计算 $f'\ (\overline{\alpha_{(j)}})$

$d, aux = \text{EXTEUCLIDES}\ (n{\omega^{n-1}}_j)\ p_j$; // 计算 $f'(\overline{\omega_{(j)}})$

对于 $e \leftarrow 2$ 到 e_j, 执行 // 应用亨泽尔引理

$\alpha_{(j)} \leftarrow \alpha_{(j)} - (\alpha_{(j)}^n + 1)c$ 对 ${p_j}^e$ 取余; // 将 ${p_j}^{e-1}$ 提升到 ${p_j}^e$

$\omega_{(j)} \leftarrow \omega_{(j)} - (\omega_{(j)}^n - 1)d$ 对 ${p_j}^e$ 取余; // 将 ${p_j}^{e-1}$ 提升到 ${p_j}^e$

end

end

4 从 $(\alpha_{(1)}, \dots, \alpha_{(k)})$ 重构 α; // 通过中国剩余定理

5 从 $(\omega_{(1)}, \dots, \omega_{(k)})$ 重构 ω; // 通过中国剩余定理

返回（α, ω）

5 快速傅里叶变换推广

如第 4 节所示，只有当 $m = {p_1}^{e_1} \dots {p_k}^{e_k}$ 满足每个 $p_i \equiv 1$ （模 $2n$）时，我们才能在环$\mathbb{Z}_m[x]/\langle x^n + 1\rangle$ 中找到合适的求值点。

这些同余关系与将 $x^n + 1$ （在 $\mathbb{Z}[x]$ 中不可约）模 m 考虑时的因式分解密切相关。相关内容已在 [12] 中阐述

其中给出了定理 19。

定理 19（ [12]中的推论 1.2）。设 $n \geq d > 1$ 为 2 的幂，且 $p \equiv 2d{+}1$（模 $4d$）为素数。那么多项式 $x^n + 1$ 分解为

$$x^n + 1 \equiv \prod_{j=0}^{d-1} \left(x^{n/d} - \alpha_j \right) \pmod{p}$$

对于互不相同的 $\alpha_j \in \mathbb{Z}_p^*$，其中 $x^{n/d} - \alpha_j$ 在 $\mathbb{Z}_p[x]$ 中不可约。

利用前几节讨论的结果，定理 19 也可以推广到模数 m 不一定为素数的情形。

定理 20（将定理 19 推广到模数 m 不一定为素数的情形）。设 $n \geq d > 1$ 为 2 的幂，$m = {p_1}^{e_1} \ldots {p_k}^{e_k}$ 是 m 的素数分解，且 $p_i \equiv 2d + 1$（模 $4d$）。那么多项式 $x^n + 1$ 分解为

$$x^n + 1 \equiv \prod_{j=0}^{d-1} \left(x^{n/d} - \alpha_j \right) \pmod{m}$$

对于互不相同的 $\alpha_j \in \mathbb{Z}_m^*$。

证明。首先注意到，$p_i \equiv 2d + 1$（模 $4d$）蕴含 $p_i \equiv 1$（模 $2d$）；因此，根据第 4.1 节讨论的结果，我们知道存在一个由 d 次单位根组成的二重集合 -1，其元素之间的差 α_0，...，α_{d-1} 均可逆。根据引理 10，我们知道每个 $\alpha_i \in \mathbb{Z}_m^*$。我们还知道，根据引理 1，

$$\prod_{j=0}^{d-1} \left(x^{n/d} - \alpha_j \right) \equiv \prod_{j=0}^{d/2-1} \left(x^{n/d} - \alpha_j \right) \left(x^{n/d} - \alpha_{\overline{j}} \right) \equiv \prod_{j=0}^{d/2-1} \left(x^{2n/d} - {\alpha_j}^2 \right).$$

鉴于平方运算保留初始性质（如命题 2 所示）$_{d-1}$，迭代应用相同的思路，最终得到所需的 $\prod \left(x^{n/d} - \alpha_j \right) \equiv x^n + 1_{j=0}$（模 m）。

□

这意味着，在满足快速傅里叶变换乘法算法所需条件时，$x^n + 1$ 模 m 时完全分裂为一次因子。然而，有时情况并非如此，我们不得不使用特定的模数，要求 $x^n + 1$ 分裂为较少的因子。

某些密码学构造便是如此：它们利用 [12] 中的定理 19（或类似版本）来保证 $\mathbb{Z}_m[x]/\langle x^n + 1 \rangle$ 中特定元素子集的可逆性。

条件 $p_i \equiv 2d + 1$ (模 $4d$) 表明，在 $\mathbb{Z}_m$ 中存在 -1 的 d 次根，但不存在 -1 的 $2d$ 次根。因此，如果 $d < n$，就不存在合适的求值点。

然而，仅有 -1 的 d 次根并不妨碍我们找到一种相当高效的乘法算法。我们无法完成递归策略，但仍然可以部分采用它。

为此，我们只需应用一种称为 n/d 次数步进的技术（如 [4] 中所述），将多项式映射到一个更便于处理的环，并定义一个新的辅助变量 $y = x^{n/d}$。我们始终可以将 $R[x]/\langle x^n + 1\rangle$ 视为 $R[x,y]/\langle x^{n/d} - y, x^n + 1\rangle$ 的子环，并注意到这个第二个环也可以表示为 $R[x][y]/\langle x^{n/d} - y, y^d + 1\rangle$。

通过这一简单的变量替换，我们现在可以将原始多项式表示为关于 y 的多项式，其 y 次数不超过 d，且系数是关于 $R[x]$ 的多项式，其 x 次数不超过 n/d。作为关于 y 的多项式，它满足所有必需条件，因为我们只考虑模数 $y^d + 1$，而新环 $R' = R[x]$ 确实包含 d 个求值点 $\alpha_0, \ldots, \alpha_{d-1}$，满足条件 1 至 3。

有了这些 d 次单位根，我们可以高效地使用 $g(x,y)$ 和 $h(x,y)$ 来计算取值 $\{g(x,\alpha_i)\}$ 和 $\{h(x,\alpha_i)\}$，在 $R[x]$ 中进行逐点乘积（使用辅助乘法算法，例如（双重）Karatsuba 算法），然后对变换进行逆变换，以恢复乘积 $(g \cdot h)(x,y)$，它能直接给出我们所需的解：再次将 y 替换为 $x^{n/d}$。

请注意，正如我们必须选择一个 α，使其为 -1 的 d 次根，从而使取值与商环 $\langle y^d + 1\rangle$ 兼容一样，另一个商环 $\langle x^{n/d} - y\rangle$ 意味着，在计算变量y在 α 处的取值时，得到的结果是 $R[x]/\langle x^{n/d} - \alpha\rangle$ 中的一个多项式，因此只在模 $x^{n/d} - \alpha$ 意义下定义。这是该技术带来的一个副作用：引入的额外变量在过程中没有实际作用，却有助于我们在计算这些 x-多项式的乘积时控制其规模。

现在，变换和逆变换的运行时间均为$\mathcal{O}(n\log\ d)$，而次数为 n/d 的多项式的 d 乘积需要每次$\mathcal{O}\left((n/d)^{\log 3}\right)$，总计为 $\mathcal{O}\left(n\log\ d + d(n/d)^{\log 3}\right)$。

从抽象的角度来看，我们可以直接应用算法 5，因为它是针对一般的环 R 描述的，因此只需考虑每个元素所属的环即可使用。不过，为了便于阅读，

我们在算法 8 中明确给出了这一推广。

算法 8: GENERALIZED EFFICIENT FFT MULTIPLICATION

输入：两个次数不超过 n 的多项式 $g(x)$、$h(x)$

辅助：$\alpha_0, \ldots, \alpha_{d-1}$，由 -1 的 d 次根构成的二重集合，且满足可逆差

结果：$\mathbb{Z}_m[x]/\langle x^n+1\rangle$ 中 $g(x)$ 与 $h(x)$ 的乘积 $(g \cdot h)(x)$

1 $\widehat{g}(y) \coloneqq g(x)$ 余 $x^{n/d} - y$；// $(\mathbb{Z}_m[x])[y]$中的多项式

2 $\widehat{h}(y) \coloneqq h(x)$ 余数 $x^{n/d} - y$；// $(\mathbb{Z}_m[x])[y]$中的多项式

3 $\boldsymbol{g} \coloneqq \mathrm{FFT}(\widehat{g}(y)\ \alpha_0, \ldots, \alpha_{d-1})$；// $\mathbb{Z}_m[x]$ 中的多项式向量

4 $\boldsymbol{h} \coloneqq \mathrm{FFT}(\widehat{h}(y)\ \alpha_0, \ldots, \alpha_{d-1})$；// $\mathbb{Z}_m[x]$中的多项式向量

定义 $\boldsymbol{f}$：$\mathbb{Z}_m[x]$ 中的一个多项式向量，大小为 d。

5 对于 $i \leftarrow 0$ 到 $d-1$，执行

  $\boldsymbol{f}[i] \coloneqq \mathrm{KARATSUBA}(\boldsymbol{g}[i], \boldsymbol{h}[i])$ 对 $x^{n/d} - \alpha_i$ 取余

end

6 $\widehat{f}(y) \coloneqq \mathrm{IFFT}(\boldsymbol{f}, \alpha_0, \ldots, \alpha_{d-1})$

7 $f(x) \coloneqq \widehat{f}(y)$ 余式 $y - x^{n/d}$

返回 $f(x)$

5.1 快速中国剩余定理

正如我们在引言中提到的，关于部分分裂环这一特定问题——我们无法将原始的完整快速傅里叶变换直接应用于初始多项式——已有研究从不同角度对模数为素数的环进行了探讨 [12]，研究如何利用类似快速傅里叶变换的算法高效应用中国剩余定理 [11, 17]。

这些中国剩余定理方法的核心思想是，将在 α_i 处的取值视为 $g(x)$（模 $x - \alpha_i$）的代表元；由于 $x^n + 1 \equiv \prod_{j=0}^{n-1}(x - \alpha_j)$（模 m），这些取值足以通过中国剩余定理确定 $g(x)$（正如定理 19 和 20 所示）。同样的递归也适用，因为既可以从 $g(x)$（模 $x^2 - \alpha_i{}^2$）计算 $g(x)$（模 $x - \alpha_i$），也可以计算$g(x)$（模 $x - \alpha_7$）（方式与第 3 节所述相同）。

给定一个由 d 次 -1 的根组成的二重集合 $\alpha_0, \ldots, \alpha_{d-1}$，算法 8 所做的正是计算余式（余 $x^{n/d} - \alpha_i$），并通过中国剩余定理确定原多项式。

然而，我们认为我们的表述仍然更加直接且信息丰富，因为将结果推广到部分分裂环中不一定为素数的模数，在 [12], 中完全不费吹灰之力；而若采用中国剩余定理来解释，当环不是主理想整环、甚至不是唯一因子分解整环时，技术上就需要进行复杂得多的分析，因为要验证该定理的假设，

必须检查某些理想是否互素，以确保每个映射确实都是同构；而且某些条件（例如要求 m 为奇数）只会作为构造过程中的副产物出现。

回想一下，求任意多项式在 α 中的取值 $g(\alpha)$，等价于计算该多项式除以 $x-\alpha$ 后的余式（这称为多项式余式定理）。

在每个 α_i 处的所有取值就是 $g(\alpha_i) = g(x)$（模 $x-\alpha_i$）。我们可以将由这些取值构成的向量视为该多项式的中国剩余定理表示。对于两个多项式 $g(x)$ 和 $h(x)$，对其取值向量进行逐点乘法，就是在中国剩余定理域中进行乘法；而插值则是从中国剩余定理表示中恢复多项式系数。

这似乎只是对同一思路的另一种解释。即使 x^n-a 没有完全分裂，也可以进行同样的分析，因为可以对一组因子计算中国剩余定理表示。

重要的是，即使 $d \neq n$，这一论证仍然成立；我们可以用它表示任意多项式的 d 个余式（模 $x^{n/d}-\alpha_i$），计算次数不超过 n/d 的多项式之间的逐点乘积，然后恢复出模 x^n-a的乘积多项式。

定理 21（$\mathbb{Z}_m[x]/\langle x^n-a\rangle$ 的中国剩余定理）。设 $x^n-a=\prod_{i=0}^{d-1}(x^{n/d}-\alpha_i)$，其中$\{\alpha_i\}_i$是 d 次根的一个二重集合，且这些根之差在 $\mathbb{Z}_m$ 中可逆，则 $\mathbb{Z}_m[x]/\langle x^n-a\rangle \cong \mathbb{Z}_m[x]/\left\langle x^{n/d}-\alpha_0\right\rangle \times \cdots \times \mathbb{Z}_m[x]/\left\langle x^{n/d}-\alpha_{d-1}\right\rangle$。

证明。为方便起见，我们将 d 根标记为 $\alpha_{b_{\log(d)-1}\ldots b_0}$，如我们在 $Figure\ 1$ 中所做的那样。

根据与定理 20 类似的论证，我们知道可以写成$x^{n/2^k}-\alpha_{b_{k-1}\ldots b_0}=(x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0})(x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0})$。

我们必须证明 $\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^k}-\alpha_{b_{k-1}\ldots b_0}\right\rangle$ 与.....同构

$$\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0}\right\rangle \times \mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0}\right\rangle.$$

我们可以定义一个映射，将 $\mathbb{Z}_m[x]$ 映射到.....

$$\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0}\right\rangle \times \mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0}\right\rangle$$

通过计算余数 rem，核将是.....

$$\left\langle x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0}\right\rangle \bigcap \left\langle x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0}\right\rangle.$$

也就是说，该映射从.....出发也是良定义的

$$\mathbb{Z}_m[x]\Big/\left\langle x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0}\right\rangle \bigcap \left\langle x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0}\right\rangle.$$

目前为止，这一讨论完全是一般性的。然而，对于这些特定的多项式，我们有

$$\left\langle x^{n/2^{k+1}}-\alpha_{0b_{k-1}\ldots b_0}\right\rangle + \left\langle x^{n/2^{k+1}}-\alpha_{1b_{k-1}\ldots b_0}\right\rangle \cong \mathbb{Z}_m[x].$$

这是因为

$$(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0}) - (x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0}) = \alpha_{1b_{k-1}\ldots b_0} - \alpha_{0b_{k-1}\ldots b_0}$$

它在 $\mathbb{Z}_m$ 中可逆（因为根之间的每个差都可逆），这意味着

$$1 \in \left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0} \right\rangle + \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0} \right\rangle.$$

我们可以明确地写出：存在两个多项式 $g(x)$ 和 $h(x)$，使得

$$(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0})g(x) + (x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0})h(x) = 1.$$

一方面，这意味着该映射是满射。根据前一个恒等式，我们知道任意一对多项式 $(a(x)\ b(x))$ 都有一个原像 $b(x)g(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0}) + a(x)h(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0})$。

另一方面，这意味着

$$\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0} \right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0} \right\rangle = \left\langle x^{n/2^k} - \alpha_{b_{k-1}\ldots b_0} \right\rangle.$$

右侧显然是左侧的子集。为证明另一个包含关系，我们可以验证对于任意

$$a(x) \in \left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0} \right\rangle \bigcap \left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0} \right\rangle,$$

也就是说，存在多项式 $b(x)$ 和 $c(x)$，使得

$$a(x) = b(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0}) = c(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0}),$$

同样可以证明

$$\begin{aligned} a(x) &= a(x) \cdot 1 \\ &= a(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0})g(x) + a(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0})h(x) \\ &= c(x)(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0})(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0})g(x) \\ &\quad + b(x)(x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0})(x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0})h(x) \\ &= (c(x)g(x) + b(x)h(x))(x^{n/2^k} - \alpha_{b_{k-1}\ldots b_0}) \end{aligned}$$

因此 $a(x) \in \left\langle x^{n/2^k} - \alpha_{b_{k-1}\ldots b_0} \right\rangle$。

综上所述，所需的映射是一个同构。

这里的主要问题是，中国剩余定理通常是在主理想整环，或至少是在唯一分解整环上定义的。如果情况并非如此，例如我们的构造，就必须专门检查这些额外性质，例如

$\left\langle x^{n/2^{k+1}} - \alpha_{0b_{k-1}\ldots b_0} \right\rangle$ and $\left\langle x^{n/2^{k+1}} - \alpha_{1b_{k-1}\ldots b_0} \right\rangle$ being comaximal. □

我们的任务定义如下。给定一个多项式 $g(x) \in \mathbb{Z}_m[x]/\langle x^n - a\rangle$，以及一个由 a 的 d次根组成、元素间的差可逆的二重集合 $\{\alpha_{b_{\log(d)-1}b_{\log(d)-2}\ldots b_0}\}$。

我们的目标是计算所有 $g_{b_{\log(d)-1}\ldots b_0}(x) = g(x)$（模 $x^{n/d} - \alpha_{b_{\log(d)-1}\ldots b_0}$）。

为此，我们先计算 $g_0(x) = g(x)$（模 $x^{n/2} - \alpha_0$）和 $g_1(x) = g(x)$（模 $x^{n/2} - \alpha_1$）。这需要 $\mathcal{O}(n)$ 次运算。

然后我们注意到 $g_{0b_0}(x) = g(x)$（模 $x^{n/4} - \alpha_{0b_0}$）$= g_{b_0}(x)$（模 $x^{n/4} - \alpha_{0b_0}$），以及 $g_{1b_0}(x) = g(x)$（模 $x^{n/4} - \alpha_{1b_0}$）$= g_{b_0}(x)$（模 $x^{n/4} - \alpha_{1b_0}$）。

一般而言，我们可以递归地计算 $g_{b_i b_{i-1}\ldots b_0}(x) = g(x)$（模 $x^{n/2^{i+1}} - \alpha_{b_i b_{i-1}\ldots b_0}$）$= g_{b_{i-1}\ldots b_0}(x)$（模 $x^{n/2^{i+1}} - \alpha_{b_i b_{i-1}\ldots b_0}$）。因此，在第 i 层，计算每个余式需要 $\mathcal{O}\left(n/2^i\right)$ 次运算，并且需要执行 $\mathcal{O}\left(2^i\right)$ 次，总成本为 $\mathcal{O}(n)$。

注意，计算 $g_{b_{i-1}\ldots b_0}(x)$（模 $x^{n/2^{i+1}} - \alpha_{0b_{i-1}\ldots b_0}$）的方法是取 $g_{b_{i-1}\ldots b_0}(x)$ 的低次系数，并加上高次系数乘以$\alpha_{0b_{i-1}\ldots b_0}$。

同理，由于 $\alpha_{1b_{i-1}\ldots b_0} = -\alpha_{0b_{i-1}\ldots b_0}$，我们可以通过取 $g_{b_{i-1}\ldots b_0}(x)$ 的低次系数，并减去高次系数乘以 $\alpha_{0b_{i-1}\ldots b_0}$，来计算 $g_{b_{i-1}\ldots b_0}(x)$（模 $x^{n/2^{i+1}} - \alpha_{1b_{i-1}\ldots b_0}$）。

注意，这些乘法相同（可以重复使用），唯一的区别是根据具体情况进行加法或减法。

层级数为 $\log(d)$，最终需要 $\mathcal{O}(n\log(d))$ 次运算，才能计算出 $g(x)$ 对 d 个次数为 n/d 的不同多项式取模后的中国剩余定理表示。

算法 9：FFT（中国剩余定理）

输入：次数不超过 n 的多项式 $g(x)$ 和一个二重集合

$$\alpha_0, \ldots, \alpha_{d-1} \text{ of } d\text{-th roots of } a \text{ with invertible differences}$$

结果：$g(x)$ 除以.....所得的余式

$$x^{n/d} - \alpha_0, \ldots, x^{n/d} - \alpha_{d-1}$$

如果 $r = 1$，则返回 g;

r $\{g_{b_{\log(d)-2}\ldots b_0}\} \coloneqq \mathrm{FFT}(g, {\alpha_0}^2, \ldots, {\alpha_{d/2-1}}^2)$

对于 $b_{\log(d)-2}\ldots b_0 \leftarrow \{0,1\}^{\log(d)-1}$ **do**

将 $g_{b_{\log(d)-2}\ldots b_0}$ 拆分 拆分为 $g^L_{b_{\log(d)-2}\ldots b_0}$ and $g^H_{b_{\log(d)-2}\ldots b_0}$

$g_{0b_{\log(d)-2}\ldots b_0} \coloneqq g^L_{b_{\log(d)-2}\ldots b_0} + g^H_{b_{\log(d)-2}\ldots b_0} \cdot \alpha_{0b_{\log(d)-2}\ldots b_0}$

$g_{1b_{\log(d)-2}\ldots b_0} \coloneqq g^L_{b_{\log(d)-2}\ldots b_0} - g^H_{b_{\log(d)-2}\ldots b_0}$

end

返回 $\{g_{b_{\log(d)-1}b_{\log(d)-2}\ldots b_0}\}$

此时，我们可以使用 Karatsuba 算法将它们相乘，代价为 $\mathcal{O}\left(d(n/d)^{\log_2(3)}\right)$。

接下来，我们必须对这些运算求逆。为此，可以沿用

相同的思路，在每个层级上对运算求逆。要恢复 $g_{b_{i-1}\dots b_0}(x)$ 的低位部分，我们将 $g_{0b_{i-1}\dots b_0}(x)$ 与 $+\,g_{1b_{i-1}\dots b_0}(x)$ 相加，再除以二（也就是将每个系数乘以 2^{-1}）。要恢复高位部分，则将它们相减，计算 $g_{0b_{i-1}\dots b_0}(x)$ 与 $-\,g_{1b_{i-1}\dots b_0}(x)$，并将每个系数乘以$2^{-1}(\alpha_{0b_{i-1}\dots b_0})^{-1}$。这些运算的代价仍为 $\mathcal{O}(n\log(d))$。

由于每一层都需要除以 2，因此可以直接跳过这一步，最后再除以 d。

算法 10：IFFT （中国剩余定理）

输入：多项式 $g(x)$ 的中国剩余定理表示 $g_0 \dots g_{d-1}$，其次数不超过 n，以及由 a 的 d 次根组成的二重集合 $\alpha_0, \dots, \alpha_{d-1}$ 且差值可逆

结果：多项式 $g(x)$

如果 $r = 2$，则返回 $2^{-1}\left((g_0 + g_1) + x^{n/2}(g_0 + g_1)\cdot {\alpha_0}^{-1}\right)$;

对于 $b_{\log(d)-2}\dots b_0 \leftarrow \{0,1\}^{\log(d)-1}$ **do**

$$g_{b_{\log(d)-2}\dots b_0} := 2^{-1}(g_{0b_{\log(d)-2}\dots b_0} + g_{1b_{\log(d)-2}\dots b_0}) + x^{n/2}2^{-1}(g_{0b_{\log(d)-2}\dots b_0} - g_{1b_{\log(d)-2}\dots b_0})\cdot {\alpha_{0b_{\log(d)-2}\dots b_0}}^{-1}$$

end

返回 $\mathrm{IFFT}(\{g_{b_{\log(d)-2}\dots b_0}\}, {\alpha_0}^2, \dots, {\alpha_{d/2-1}}^2)$

这种替代性解释具有相同的性质，也可用于设计相同的高效乘法算法。不过，我们认为此前的阐述更有启发性。

6 结果与讨论

主要结果可概括如下：为了在 $\mathbb{Z}_m[x]/\langle x^n - a\rangle$ 中通过快速傅里叶变换设计高效乘法算法，必要且充分条件是存在一组 n 个互不相同的 n 次 a 根（条件 2，使乘法与同余类相容），且这些根之差均可逆（条件 1，使逆变换有定义），并且它们的递归平方两两相等（条件 3，使计算能够高效地递归进行）。

这种刻画与通常基于单位根的刻画（条件 4）相似，但并不等价；后者是充分条件，却非必要条件。不过，我们已证明，正如定理 15 所述，将范围限制在满足条件 4 的集合中，并不会降低这些高效乘法算法的适用性，因为这类 (α,ω) 集合存在，当且仅当满足条件 1 至 3 的必要且充分的集合存在。

作为中间结果，我们还证明了在一般情况下，这些性质确实彼此独立；我们也相信，这项分析或许有助于厘清通常在

约定俗成知识中提及的一些考量和条件，究竟是关于代数结构的基本考量，还是仅仅针对特定情境下某种特定实例化的约定（单位根的情况便是如此）。

这一框架也从严谨的数学角度，对快速傅里叶变换乘法作了通俗易懂的概述，即便读者并不熟悉更高级的代数考量。

例如，正如我们在第5节所见，我们的分析可直接推广到这样的环： $x^n - a$ 并非完全分裂，且 $\mathbb{Z}_m$ 不是域；而其他解读方式则要复杂得多。

致谢。本研究部分得到欧盟PROMETHEUS项目（地平线2020研究与创新计划，资助编号780701）以及西班牙经济与竞争力部通过项目MTM2016-77213-R的资助。

参考文献

[1] Eric Bach 和 Jeffrey Outlaw Shallit。《算法数论》。计算基础丛书。麻省理工学院出版社，马萨诸塞州剑桥，1996年。[2] Carsten Baum、Ivan Damg˚ard、Vadim Lyubashevsky、Sabine Oechsner 和 Chris Peikert。《基于结构化格假设的更高效承诺方案》。载于 Dario Catalano 和 Roberto De Prisco 编，《SCN 18》，《计算机科学讲义》第11035卷，第368–385页，意大利阿马尔菲，2018年9月5日至7日。Springer，德国海德堡。doi:10.1007/978-3-319-98113-0 20。[3] Jon Louis Bentley、Dorothea Haken 和 James B. Saxe。《求解分治递推式的一般方法》。《SIGACT News》，12(3):36—44，1980年9月。doi:10.1145/1008861.1008865。[4] Daniel J Bernstein。《面向数学家的多位数乘法》。https://cr.yp.to/papers/m3.pdf。[5]Chi-Ming Marvin Chung、Vincent Hwang、Matthias J. Kannwischer、Gregor Seiler、Cheng-Jhih Shih 和 Bo-Yin Yang。《面向不适用于数论变换的环的数论变换乘法》。《IACR TCHES》，2021(2):159–188，2021年。doi:10.46586/tches.v2021.i2.159-188。[6] James W. Cooley 和 John W. Tukey。《一种用于机器计算复傅里叶级数的算法》。《计算数学》，19(90):297–301，1965年。doi:10.2307/2003354。[7] Charles M. Fiduccia。《通过除法算法进行多项式求值：重新审视快速傅里叶变换》。载于《第四届 ACM 计算理论年度研讨会论文集》，STOC ’72，第88—93页，美国纽约州纽约市，1972年。计算机协会。doi:10.1145/800152.804900。

[8] Anatolii Karatsuba。《自动机上的多位数乘法》。载于《苏联物理学报告》，第7卷，第595–596页，1963年。[9]Vadim Lyubashevsky、Daniele Micciancio、Chris Peikert 和 Alon Rosen。《SWIFFT：一种用于快速傅里叶变换哈希的适度方案》。载于 Kaisa Nyberg 编，《FSE 2008》，《计算机科学讲义》第5086卷，第54–72页，瑞士洛桑，2008年2月10日至13日。Springer，德国海德堡。doi:10.1007/978-3-540-71039-4 4。[10] Vadim Lyubashevsky、Chris Peikert 和 Oded Regev。《关于理想格与环上的带误差学习》。载于 Henri Gilbert 编，《EUROCRYPT 2010》，《计算机科学讲义》第6110卷，第1–23页，法国里维埃拉，2010年5月30日至6月3日。Springer，德国海德堡。doi:10.1007/978-3-642-13190-5 1。[11]Vadim Lyubashevsky、Chris Peikert 和 Oded Regev。《环上带误差学习密码学工具箱》。载于 Thomas Johansson 和 Phong Q. Nguyen 编，《EUROCRYPT 2013》，《计算机科学讲义》第7881卷，第35–54页，希腊雅典，2013年5月26日至30日。Springer，德国海德堡。doi:10.1007/978-3-642-38348-9 3。[12]Vadim Lyubashevsky 和 Gregor Seiler。《部分分裂圆分环中的短可逆元素及其在基于格的零知识证明中的应用》。载于 Jesper Buus Nielsen 和 Vincent Rijmen 编，《EUROCRYPT 2018》，第一部分，《计算机科学讲义》第10820卷，第204–224页，以色列特拉维夫，2018年4月29日至5月3日。Springer，德国海德堡。doi:10.1007/978-3-319-78381-9 8。[13] Ivan Niven、Herbert S. Zuckerman 和 Hugh L. Montgomery。《数论导论》。John Wiley，第5版，1991年。[14] Thomas P¨oppelmann 和 Tim G¨uneysu。《面向可重构硬件上基于格的密码学的高效算术》。载于 Alejandro Hevia 和 Gregory Neven 编，《LATINCRYPT 2012》，《计算机科学讲义》第7533卷，第139–158页，智利圣地亚哥，2012年10月7日至10日。Springer，德国海德堡。doi:10.1007/978-3-642-33481-8 8。[15] Daniel Shanks。《五种数论算法》。载于《第二届马尼托巴数值数学会议论文集》（温尼伯），第51–70页，1973年。[16] Gilbert Strang。《线性代数导论》。剑桥出版社，韦尔斯利，第5版，2016年。[17]Joachim von zur Gathen 和 J¨urgen Gerhard。《现代计算机代数》。剑桥大学出版社，第3版，2013年。doi:10.1017/CBO9781139856065。