关于回文/反向互补重复纠错码

孙宇博和葛根年

arXiv:2602.01151v1 [cs.IT] 1 Feb 2026

摘要

受体内 DNA 存储应用的启发，我们研究了用于纠正重复的码。长度为 k 的反向互补重复，是将长度为 k 的子串反转并取互补后，插入其原始位置旁；而回文复制则只插入反转后的副本，不进行互补。首先，我们构造了一种显式码，仅需一个冗余符号，即可纠正任意数量的反向互补重复（或回文重复），前提是所有重复的长度均为 $k \geqslant 3\lceil \log_q n \rceil$，且彼此不相交。接下来，我们推导出能够纠正任意长度反向互补重复（或回文复制）的码的吉尔伯特–瓦尔沙莫夫界，表明最优冗余上界为 $2\log_q n + \log_q \log_q n + O(1)$。最后，对于 $q \geqslant 4$，我们给出两种能够纠正 t 长度为一的反向互补重复的显式码构造。第一种构造的冗余为 $2t \log_q n + O(\log_q \log_q n)$，编码复杂度为 $O(n)$，解码复杂度为 $O\big(n(\log_2 n)^4\big)$。第二种构造将冗余改进至 $(2t-1)\log_q n + O(\log_q \log_q n)$，但编码和解码复杂度为 $O\big(n \cdot \mathrm{poly}(\log_2 n)\big)$。

I. 引言

在当前大数据时代，全球数字数据量正呈指数级增长。然而，在可预见的未来，现有技术的存储容量不太可能跟上不断增长的需求。这种差距要求存储技术取得突破，以提供创新、高密度、高效且耐用的解决方案。在这些方案中，DNA 存储系统凭借超高密度、卓越的耐久性和低维护成本，成为传统存储介质的有力替代方案。在 [5], [9]（用于体外 DNA 存储）和 [27]（用于体内 DNA 存储）中，研究者已证明将信息存储于 DNA 中的可行性。与体外 DNA 存储相比，体内 DNA 存储还具备其他功能，例如为转基因生物（GMOs）添加水印以验证其真实性并追踪未经授权的使用 [2], [12], [23], ，以及在生物学研究中标记生物体 [37]。然而，体内 DNA 存储也带来了独特的挑战：在 DNA 复制过程中出现错误时，可能会产生重复错误。这促使我们研究能够纠正此类错误的码。

通常，长度为 k 的重复错误涉及从 DNA 字符串中取出一个长度为 k 的子串，并将其副本（可能经过修改）插入到原位置相邻处。重复错误主要有三种类型：

- 串联重复，即副本保持不变；
- 回文复制（也称为反向重复），即副本经过反向处理；
- 反向互补重复，即副本先反向，然后再取互补。

尽管串联重复已在文献中得到广泛研究， [6], [7], [8], [10], [13], [14], [16], [17], [18], [34], [35], [36], [40], 反向互补重复和回文重复由于其复杂的内在结构，仍相对缺乏研究。

本文聚焦于反向互补重复和回文复制。鉴于重复可视为一种特殊类型的突发插入，我们可以直接使用突发插入纠错码来纠正重复。然而，这种方法并非最高效，因为与一般的突发插入相比，重复本身蕴含了额外信息。因此，一个自然的问题是：我们能否利用重复的内在性质，设计出冗余度低于一般突发插入纠错码 [11], [20], [21], [22], [28], [29], [30], [31], [32], [33], [38] 的重复纠错码？在 [3], Ben-Tolila 和 Schwartz 证明，当重复长度 k 为奇数且字母表大小为 $q \geqslant 2$时，可通过保持互补映射，利用任何能够纠正长度为 k 的突发插入的二进制码，构造能够纠正长度为 k的反向互补重复的 q元码。值得注意的是，当 $q \geqslant 4$时，他们得到了冗余度低于一般单突发插入纠错码的构造。此外，Yohananov 和 Schwartz [39] 构造了码率分别为 $\log_q(q-2)$ 和 $\log_q(q-1)$的最优码，能够分别纠正任意数量的长度为 $k=1$的反向互补重复和回文复制。在 [19], Lenz、Wachter-Zeh 和 Yaakobi 证明，若一个码 $\mathcal{C}$ 不包含长度至少为二的回文重复串，即不存在两个相邻的子串，其长度为 $k \geqslant 2$，且后一个子串是前一个子串的反向，则 $\mathcal{C}$ 能够纠正任意长度 $k \geqslant 2$的回文复制。此外，他们还证明，这类码的码率至少为 $\log_q(q-1)$。Liu 的近期工作，

本研究得到中国国家重点研发计划项目（资助编号：2025YFC3409900）、中国国家自然科学基金项目（资助编号：12231014）和北京市学者计划的资助。Y. Sun（2200502135@cnu.edu.cn）和 G. Ge（gnge@zju.edu.cn）任职于首都师范大学数学科学学院，中国北京 100048。

Tang、Fan 和 Sagar [24] 将这一结论推广到了反向互补重复。鉴于能够纠正任意数量的长度为 $k=1$ 的突发插入，或纠正任意长度 $k\geqslant 2$ 的突发插入的码，其码率为零， [19], [24], [39] 中引用的研究也得出了在纠正重复方面优于一般突发插入纠错码的码。

本文其余部分组织如下。在第二节中，我们介绍全文所使用的相关记号。在第三节中，我们考虑用于纠正长长度回文重复/反向互补重复的码。我们证明，若码 $\mathcal{C}$ 避免长度为 m的反向互补重复串（相应地，回文重复串），则 $\mathcal{C}$ 能够纠正任意 $t\geqslant 1$个反向互补重复（相应地，回文重复），前提是所有重复的长度均为 $k\geqslant 3m-3$且互不相交。对于 $m=2$ 和 $t=1$，该码可加以推广，以纠正长度至少为二的反向互补重复（相应地，回文复制），从而恢复 [19]和 [24] 中已建立的结果。对于 $m=\lfloor \log_q n\rfloor+1$，我们为该码提出了仅使用一个冗余符号的高效编码和解码算法。在第四节中，我们考虑用于纠正任意长度的反向互补重复（相应地，回文复制）的码。结合第三节中提出的码，我们推导了此类码的吉尔伯特–瓦尔沙莫夫界，表明存在一种构造，其冗余符号数为$2\log_q n+\log_q\log_q n+O(1)$，即渐近码率为$1$。这改进了 [19] 和 [24]中建立的码率界$\log_q(q-1)$。在第五节中，我们考虑用于纠正多个长度为一的反向互补复制的码。通过采用 [39], 中的技术，我们将长度为一的反向互补复制转化为替换。此外，通过施加游程长度受限约束，并将 [25] 中的替换纠错码与 [22], 中的插入删除纠错码相结合，我们设计了两族 q元码（其中 $q\geqslant 4$），能够纠正 t 个长度为一的反向互补复制。第一种构造实现的冗余为 $2t\log_q n+O(\log_q\log_q n)$，编码复杂度为 $O(n)$，解码复杂度为 $O(n(\log_2 n)^4)$。第二种构造实现了改进后的冗余 $(2t-1)\log_q n+O(\log_q\log_q n)$，但编码和解码复杂度为 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$。这些构造优于直接使用插入删除纠错码 [22], [29], [30], [31], [38], 的方法；当分别为 $t=2$ 和 $t\geqslant 3$时，后者需要$5\log_q n+O(\log_q\log_q n)$ 和 $(4t-1)\log_q n+O(\log_q\log_q n)$个冗余符号。最后，在第六节中，我们对本文进行总结。

II. 预备知识

A. 记号

给定两个整数 i,j，定义整数区间 $[i,j]$为 $\{i,i+1,\ldots,j\}$（若 $i\leqslant j$），否则定义为空集 $\varnothing$。对于任意整数 $q\geqslant 2$，定义 $\Sigma_q=[0,q-1]$ 为 q元字母表。令 Σ_q^n表示由Σ_q中的符号构成的所有长度为 n 的字符串集合。对于任意字符串 $\boldsymbol{x}\in\Sigma_q^n$，我们将其写作 $\boldsymbol{x}=x_1x_2\cdots x_n$或 $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$，其中 x_s表示$\boldsymbol{x}$ 的第 s个符号， $s\in[1,n]$。 $\boldsymbol{x}$的长度记为 $|\boldsymbol{x}|$。对于任意整数区间 $[i,j]\subseteq[1,n]$，我们称$\boldsymbol{x}_{[i,j]}=x_i\cdots x_j$为 $\boldsymbol{x}$的子串，其中当 $i>j$时， $x_i\cdots x_j$表示空字符串 $\boldsymbol{\epsilon}$。令 $\boldsymbol{y}\in\Sigma_q^m$。$\boldsymbol{x}$与 $\boldsymbol{y}$的串接记作 $\boldsymbol{xy}$或 $(\boldsymbol{x},\boldsymbol{y})$，定义为 $x_1x_2\cdots x_ny_1y_2\cdots y_m$。对于集合 V，其基数记为 $\#V$。对于非负整数 a，令$Rep_{q,m}(a)$ 为 a 的长度为 m的 q进制表示。则其逆映射 $Rep_q^{-1},_m(\cdot)$ 由 $Rep_q^{-1},_m\big(Rep_{q,m}(a)\big)=a$ 定义，是良定义的。

B. 互补操作和反转操作

一个互补操作[1] 被定义为一个双射，将字母表 Σ_q 中的每个符号 a 映射到 Σ_q中的唯一符号 $\overline{a}$，并满足以下性质：

$$\overline{a}\neq a \quad\text{and}\quad \overline{\overline{a}}=a \quad\text{for each } a\in\Sigma_q.$$

这一设定中的一个隐含假设是 q 为偶数。因此，在考虑互补操作时，我们始终假设 q为偶数。该互补操作可以自然地扩展到整个字符串，即对于 $\boldsymbol{x}=x_1x_2\cdots x_n$，我们定义

$$\overline{\boldsymbol{x}}=\overline{x_1}\,\overline{x_2}\cdots\overline{x_n}.$$

$\boldsymbol{x}$的反转记为 $\boldsymbol{x}^R$，其构造方式是反转 $\boldsymbol{x}$中符号的顺序，即我们定义

$$\boldsymbol{x}^R=x_nx_{n-1}\cdots x_1.$$

此外， $\boldsymbol{x}$的反向互补记为 $\boldsymbol{x}^{RC}$，其构造方式是先反转 $\boldsymbol{x}$，再对每个符号取互补，即我们定义

$$\boldsymbol{x}^{RC}=\overline{x_n}\,\overline{x_{n-1}}\cdots\overline{x_1}.$$

R引理 II.1。 互补操作和反转操作可交换。具体而言，有

$$\boldsymbol{x}^{RC}=\overline{\boldsymbol{x}^R}=\overline{\boldsymbol{x}}^R.$$

[1]互补ent 操作可视为一个置换 $\pi:\ \Sigma_q\to\Sigma_q$，其中每个 $a\in\Sigma_q$都有 $\pi(a)\neq a$ 和 $\pi\big(\pi(a)\big)=a$

C. 重复纠错码

现在我们形式化定义反向互补重复和回文复制。设 $\boldsymbol{x}=\boldsymbol{uvw}$ 为一个长度为 n的字符串，其中 $\boldsymbol{u}$、 $\boldsymbol{v}$和 $\boldsymbol{w}$ 分别是长度为 $|\boldsymbol{u}|=i-1$、 $|\boldsymbol{v}|=k$和 $|\boldsymbol{w}|=n-i-k+1$的子串。在 $\boldsymbol{x}$ 中位置 $i\in[1,\ n-k+1]$ 处的错误称为 k-反向互补重复，如果所得字符串为

$$RC_{k,i}(\boldsymbol{x})=\boldsymbol{uvv}^{RC}\boldsymbol{w}.$$

类似地，在 $\boldsymbol{x}$ 中位置 $i\in[1,n-k+1]$ 发生的错误称为 k-回文复制，如果所得字符串为

$$R_{k,i}(\boldsymbol{x})=\boldsymbol{uvv}^{R}\boldsymbol{w}.$$

以 $\boldsymbol{x}$ 为中心、半径为 t 的反向互补重复球 k定义为

$$RC_k^t(\boldsymbol{x})=\left\{RC_{k,i_t}\big(RC_{k,i_{t-1}}(\ldots(RC_{k,i_1}(\boldsymbol{x})\ldots)\big)\colon\ i_j\in[1,n+(j-1)k-k+1]\text{对于 }\ j\in[1,t]\right\}.$$

若码 $\mathcal{C}\subseteq\Sigma_q^n$ 能够纠正 t k次反向互补重复，即满足以下条件，则称其为 t k-反向互补重复纠错码：

$$RC_k^t(\boldsymbol{x})\cap RC_k^t(\boldsymbol{y})=\varnothing\quad\text{for each }\boldsymbol{x}\neq\boldsymbol{y}\in\mathcal{C}.$$

t k-回文复制纠错码 的定义可以类似地给出。为评估码 $\mathcal{C}\subseteq\Sigma_q^n$，我们考虑其冗余或码率：冗余定义为 $n-\log_q|\mathcal{C}|$，码率定义为 $\log_n^q|\mathcal{C}|$。

III. 长回文复制/反向互补重复的纠正

对于串联重复，Jain、Farnoud、Schwartz 和 Bruck [14] 以及 Goshkoder、Polyanskii、Vorobyev [10] 证明了，包含不含长度为 k 的串联重复序列的字符串的码 $\mathcal{C}$，本身具备纠正该长度串联重复的能力。然而，尽管对于长度为 $k=2$的重复，这一结果已在 [19] 和 [24]中得到证明，但它并不能普遍推广到反向互补重复和回文重复。为说明这一局限性，我们考虑码 $\mathcal{C}=\{000111000,\ 000111110\}$，该码专门用于排除任何长度为四的反向互补重复串。尽管如此， $\mathcal{C}$ 仍无法纠正长度为四的反向互补重复。具体而言：

- 字符串 000111000 可以演化为 0001110001110，其中带下划线的子串表示被复制的部分。
- 此外 ，字符串 000111110 也可以演变为 0001110001110 .

这一观察凸显了在处理反向互补重复和回文复制时采用更细致方法的必要性。

接下来，我们仅关注反向互补重复，因为本文中确立的所有结果都可以无缝扩展到处理回文复制。我们首先证明，如果一个码 $\mathcal{C}$ 能够确保其码字中不存在两个相邻的、长度为 m 的子串，且后一个子串是前一个子串的反向互补，那么 $\mathcal{C}$ 就能够纠正长度至少为 $3m-3$的反向互补重复。为此，我们引入以下概念。

定义 III.1. 设 $m\geqslant 2$ 为一个整数。如果字符串 $\boldsymbol{x}\in\Sigma_q^n$ 满足：对于任意两个长度为 m的相邻子串，后一个都不是前一个的反向互补，则称其为 m-反向互补重复根 （简称 m-RCD 根）。换言之，对于每个 $i\in[1,n-2m+1]$，我们要求

$$\boldsymbol{x}_{[i+m,i+2m-1]}\neq\boldsymbol{x}_{[i,i+m-1]}^{RC},$$

或者等价地

$$\boldsymbol{x}_{[i,i+m-1]}\neq\boldsymbol{x}_{[i+m,i+2m-1]}^{RC}.$$

以下结论对我们的码设计至关重要：

引理 III.2. 设 $\boldsymbol{x}\in\Sigma_q^n$是一个 m-*RCD* 根，其中 $m\geqslant 2$ 为整数。若对于某个整数 $k\geqslant 3m-3$ 和位置 $i\in[1,\ n-k+1]$，则对于 $j<i$，有 $\boldsymbol{y}=RC_{k,i}(\boldsymbol{x})$，有 $\boldsymbol{y}_{[i+k,i+2k-1]}=\boldsymbol{y}_{[i^{RC},i+k-1]}^{RC}$，且 $\boldsymbol{y}_{[j+k,j+2k-1]}\neq\boldsymbol{y}_{[j^{RC},j+k-1]}^{RC}$。

证明。由于 $\boldsymbol{y}=RC_{k,i}(\boldsymbol{x})$，根据定义可知 $\boldsymbol{y}_{[i+k,i+2k-1]}=\boldsymbol{y}_{[i^{RC},i+k-1]}^{RC}$，且 $\boldsymbol{y}_{[1,i+k-1]}=\boldsymbol{x}_{[1,i+k-1]}$。鉴于 $\boldsymbol{x}$ 是一个 m-RCD 根，我们可知前缀 $\boldsymbol{y}_{[1,i+k-1]}$ 也保持 m-RCD 根的性质。反设存在某个 $j<i$，使得 $\boldsymbol{y}_{[j+k,j+2k-1]}=\boldsymbol{y}_{[j^{RC},j+k-1]}^{RC}$。我们分两种情况进行分析：

- 如果 $j\leqslant i-m$，那么 $j+k+m-1\leqslant i+k-1$。由于 $\boldsymbol{y}_{[j+k,j+2k-1]}=\boldsymbol{y}_{[j^{RC},j+k-1]}^{RC}$，可得 $\boldsymbol{y}_{[j+k,j+k+m-1]}=\boldsymbol{y}_{[j+k-m,j+k-1]}^{RC}$，这与 $\boldsymbol{y}_{[1,i+k-1]}$ 是一个 m-RCD 根这一事实矛盾。
- 如果 $i-m<j<i$，令 $\ell\triangleq|\boldsymbol{x}_{[j,i-1]}|=i-j$ 和 $s=\lfloor\frac{k+\ell}{2\ell}\rfloor$，则 $\ell\in[1,m-1]$，且 $2s\ell\in[k-\ell+1,k+\ell]$ 是一个偶整数。注意到 $k-\ell+1\geqslant(3m-3)-(m-1)+1=2m-1$，可得 $2s\ell\geqslant 2m$，即 $s\ell\geqslant m$。我们定义 $\boldsymbol{u}^{(t)}=\boldsymbol{y}_{[i+k-t\ell,i+k-(t-1)\ell-1]}$，其中 $t\in[1,2s]$；并定义 $\boldsymbol{v}^{(t)}=\boldsymbol{y}_{[i+k+(t-1)\ell,i+k+t\ell-1]}$，其中 $t\in[1,2s-1]$，定义如图 III.1 所示。由于 $\boldsymbol{y}_{[i+k,i+2k-1]}=\boldsymbol{y}_{[i^{RC},i+k-1]}^{RC}$，对于 $t\in[1,2s-1]$，有 $\boldsymbol{v}^{(t)}=\left(\boldsymbol{u}^{(t)}\right)^{RC}$。此外，

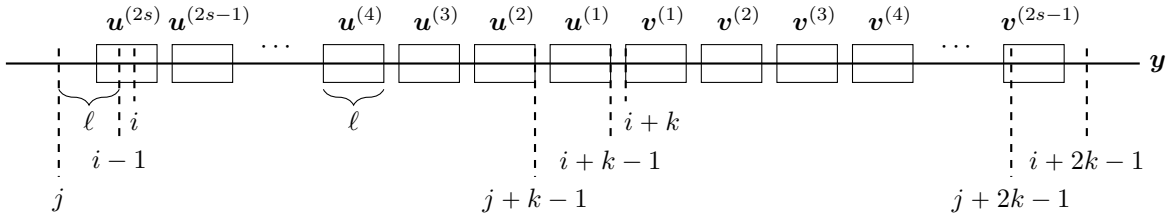

图 III.1： $\boldsymbol{u}^{(t)}$ 对于 $t \in [1,\ 2s]$ 以及 $\boldsymbol{v}^{(t)}$ 对于 $t \in [1,\ 2s$ –的定义示意图 $1]$.

由于 $\boldsymbol{y}_{[j+k,\ j+2k-1]} = \boldsymbol{y}^{RC}_{[j\ ,\ j+k-1]}$，对于 $t \in [1,\ 2s-2]$，有 $\boldsymbol{u}^{(1)} = \left(\boldsymbol{u}^{(2)}\right)^{RC}$ 和 $\boldsymbol{v}^{(t)} = \left(\boldsymbol{u}^{(t+2)}\right)^{RC}$。因此，$\boldsymbol{u}^{(1)} = \boldsymbol{u}^{(3)} = \cdots = \boldsymbol{u}^{(2s-1)} = \left(\boldsymbol{u}^{(2)}\right)^{RC}$ 和 $\boldsymbol{u}^{(2)} = \boldsymbol{u}^{(4)} = \cdots = \boldsymbol{u}^{(2s)}$。于是可得

$$\begin{aligned}\boldsymbol{y}_{[i+k-s\ell,i+k-1]} &= \boldsymbol{u}^{(s)}\boldsymbol{u}^{(s-1)}\cdots\boldsymbol{u}^{(1)}\\ &= \left(\boldsymbol{u}^{(s+1)}\right)^{RC}\left(\boldsymbol{u}^{(s+2)}\right)^{RC}\cdots\left(\boldsymbol{u}^{(2s)}\right)^{RC}\\ &= \left(\boldsymbol{u}^{(2s)}\boldsymbol{u}^{(2s-1)}\cdots\boldsymbol{u}^{(s+1)}\right)^{RC}\\ &= \boldsymbol{y}^{RC}_{[i+k-2s\ell,i+k-s\ell-1]}.\end{aligned}$$

注意，$\boldsymbol{y}_{[i+k-s\ell,\ i+k-1]}$ 的长度为 $s\ell \geqslant m$，这与 $\boldsymbol{y}_{[1,\ i+k-1]}$ 是一个 m-RCD 根这一事实相矛盾。

在这两种情况下，假设存在这样的 $j < i$ 都会导致矛盾，因此不存在这样的 j。证明至此完成。

□

假设一个 m-RCD 根 $\boldsymbol{x} \in \Sigma_q^n$遭受长度为 $k \geqslant 3m-3$ 的反向互补重复，并产生 $\boldsymbol{y}$。令 i 为满足 $\boldsymbol{y}_{[i+k,\ i+2k-1]} = \boldsymbol{y}^{RC}_{[i\ ,\ i+k-1]}$的最小索引，则根据引理 III.2，有 $\boldsymbol{x} = \boldsymbol{y}_{[1,\ i+k-1]}\boldsymbol{y}_{[i+2k,\ n+k]}$。由于比较两个长度为 k 的序列需要 $O(k)$ 时间，且最坏情况下可能需要检查多达 n 对，因此解码时间为 $O(kn)$。如果重复长度 k 最大可达 $O(n)$，则最坏情况下的时间为 $O(n^2)$。下面的引理将提出一种更高效的方法。

引理 III.3. 设 $\boldsymbol{x} \in \Sigma_q^n$是一个 m-*RCD* 根。若 $\boldsymbol{x}$ 在位置 i 处发生长度为 $k \geqslant 3m-3$ 的反向互补重复并得到 $\boldsymbol{y}$，令 $p \in [1、n-3m+4]$ 为满足 $\boldsymbol{y}_{[p+3m-3,\ p+6m-7]} = \boldsymbol{y}^{RC}_{[p\ ,\ p+3m-4]}$的最小下标，则 $i = p+3m-3-k$ 且 $\boldsymbol{x} = \boldsymbol{y}_{[1,\ p+3m-4]}\boldsymbol{y}_{[p+3m-3+k,\ n+k]}$。该解码过程在 $O(mn)$ 时间内完成。

证明。由于 $\boldsymbol{y} = RC_{k、i}(\boldsymbol{x})$，我们有 $\boldsymbol{y}_{[1、i+k-1]} = \boldsymbol{x}_{[1、i+k-1]}$ 以及 $\boldsymbol{y}_{[i+k、i+2k-1]} = \boldsymbol{y}^{RC}_{[i\ 、i+k-1]}$。鉴于 $k \geqslant 3m-3$，我们可以进一步推出 $\boldsymbol{y}_{[i+k、i+k+3m-4]} = \boldsymbol{y}^{RC}_{[i+k-3m-3、i+k-1]}$。考虑字符串 $\boldsymbol{y}_{[1、i+k-1]}$ 与 $\boldsymbol{y}_{[1、i+k+3m-4]}$，我们有 $\boldsymbol{y}_{[1、i+k+3m-4]} = RC_{3m-3、i+k-3m+3}(\boldsymbol{y}_{[1、i+k-1]})$。由于 $\boldsymbol{x}$ 是一个 m-RCD 根，可见前缀 $\boldsymbol{y}_{[1、i+k-1]}$ 同样保持 m-RCD 根性质。由引理 III.2，令 $p \in [1、n-3m+4]$ 为满足 $\boldsymbol{y}_{[p+3m-3、p+6m-7]} = \boldsymbol{y}^{RC}_{[p\ 、p+3m-4]}$的最小下标，我们有 $p = i+k-3m+3$。因此，我们可以确定 $i = p+3m-3-k$，进而确定 $\boldsymbol{x} = \boldsymbol{y}_{[1、p+3m-4]}\boldsymbol{y}_{[p+3m-3+k、n+k]}$。由于比较两个长度为 $3m-3$ 的序列需要 $O(m)$ 时间，且在最坏情况下最多可能检查 n 对，由此得到解码时间为 $O(mn)$。证明完毕。 □

注 III.4. 当 $m = 2$ 时，引理 *III.3* 为 m-*RCD* 根提供了一种解码策略，该策略能够纠正长度为 k、满足 $k \geqslant 3m-3 = 3$ 的反向互补重复。事实上，正如下文所述，对于 $k = 2$，这一策略仍然有效。

设 $\boldsymbol{x} \in \Sigma_q^n$ 是一个2-*RCD* 根。假设对于某些 $i \in [1、n-1]$，$\boldsymbol{y} = RC_{2,i}(\boldsymbol{x})$。那么 $\boldsymbol{x} = \boldsymbol{y}_{[1,i+1]}\boldsymbol{y}_{[i+4,n+2]}$ 且 $y_{i+3} = y_i$。设 $p \in [1、n-1]$ 为满足 $\boldsymbol{y}_{[p+2,p+3]} = \boldsymbol{y}^{RC}_{[p\ ,p+1]}$ 的最小指标。我们有 $p \in \{i-1、i\}$；否则，对于某个 $p \leqslant i-2$，有 $\boldsymbol{x}_{[p+2,p+3]} = \boldsymbol{x}^{RC}_{[p\ ,p+1]}$，这与 2-*RCD* 根的性质矛盾。

- 若 $p = i$，则 $\boldsymbol{x} = \boldsymbol{y}_{[1,\ p+1]}\boldsymbol{y}_{[p+4,\ n+2]}$。
- 若 $p = i-1$，则 $\boldsymbol{y}_{[i+1,\ i+2]} = \boldsymbol{y}^{RC}_{[i-1,\ i]}$。这意味着 $y_{i+1} = y_i$。由于 $y_{i+3} = y_i$，我们得到 $y_{i+3} = y_{i+1}$ 和$\boldsymbol{y}_{[1,\ p+1]}\boldsymbol{y}_{[p+4,\ n+2]} = \boldsymbol{y}_{[1,\ \overline{i+1}]}\boldsymbol{y}_{[i+4,\ n+2]}$。

在这两种情况下，解码策略都会输出正确的 $\boldsymbol{x} = \boldsymbol{y}_{[1,\ i+1]}\boldsymbol{y}_{[i+4,\ n+2]}$。因此，2-*RCD* 根能够纠正长度至少为二的反向互补重复，从而重新得到 *[24]* 中确立的主要结果。此外，与 *[24]*, 相比，我们提出了一种高效的解码策略。

A. 扩展到纠正不相交重复

现在，我们将引理 III.3 推广到纠正不相交定长反向互补重复。首先介绍相关定义。

定义 III.5. 假设 t、k 和 i_1、…、i_t 为正整数，且

$$1 \leqslant i_1 < \cdots < i_t \leqslant n-k+1 \quad \textit{and} \quad i_j - i_{j-1} \geqslant k \quad \textit{for } j \in [1,t].$$

Let

$$\boldsymbol{x} = \boldsymbol{u}^{(1)}\boldsymbol{v}^{(1)}\boldsymbol{u}^{(2)}\boldsymbol{v}^{(2)}\cdots\boldsymbol{u}^{(t)}\boldsymbol{v}^{(t)}\boldsymbol{w} \in \Sigma_q^n,$$

其中 $|\boldsymbol{w}| \geqslant 0$ 和 $|\boldsymbol{u}^{(j)}| \geqslant 0$, $|\boldsymbol{v}^{(j)}| = k$，以及 $|\boldsymbol{u}^{(j)}| + \sum_{p=1}^{j-1}(|\boldsymbol{u}^{(p)}| + |\boldsymbol{v}^{(p)}|) = i_j - 1$ 对于 $j \in [1,t]$。我们称 $\boldsymbol{x}$ 在位置 $i_1, \ldots, i_t$ 遭受 t不相交 k-反向互补重复，如果所得字符串为

$$RC_{k,(i_1,i_2,\ldots,i_t)}(\boldsymbol{x}) = \boldsymbol{u}^{(1)}\boldsymbol{v}^{(1)}\left(\boldsymbol{v}^{(1)}\right)^{RC}\boldsymbol{u}^{(2)}\boldsymbol{v}^{(2)}\left(\boldsymbol{v}^{(2)}\right)^{RC}\cdots\boldsymbol{u}^{(t)}\boldsymbol{v}^{(t)}\left(\boldsymbol{v}^{(t)}\right)^{RC}\boldsymbol{w}.$$

如果码 $\mathcal{C} \subseteq \Sigma_q^n$ 能够纠正 t 不相交的 k-反向互补重复，则称其为 t-不相交 k-反向互补重复纠错码。

算法 1 去重算法

输入： $\boldsymbol{y} \in RC_{k,(i_1,i_2,\ldots,i_t)}(\boldsymbol{x})$，其中 $\boldsymbol{x} \in \Sigma_q^n$是一个 m-RCD 根， $k \geqslant 3m-3$, $t \geqslant 1$, $1 \leqslant i_1 < \cdots i_t \leqslant n-k+1$，且 $i_j - i_{j-1} \geqslant k$ 对于 $j \in [2,t]$

输出： $\boldsymbol{x}$

1: **初始化：** 令 $p = 1$ 和 $n' \triangleq |\boldsymbol{y}| = n + tk$2:
当 $t > 0$ **时**3: **如果** $\boldsymbol{y}_{[p+3m-3,\ p+6m-7]} = \boldsymbol{y}^{RC}_{[p,\ p+3m-4]}$ **，则**4: $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,\ p+3m-4]}\boldsymbol{y}_{[p+3m-3+k,\ n']}$
5: $n' \leftarrow n' - k$6: $t \leftarrow t-1$7: $p \leftarrow p+k$8: **否则**
9: $p \leftarrow p+1$10: **结束如果**11: **结束循环**12:
$\boldsymbol{x} \leftarrow \boldsymbol{y}$13: **返回** $\boldsymbol{x}$

定理 III.6. 设 $\mathcal{C} \subseteq \Sigma_q^n$ 为所有 m-*RCD* 根的集合，则 $\mathcal{C}$ 是一个 t-不相交的 k-反向互补重复纠错码，适用于 $t \geqslant 1$ 和 $k \geqslant 3m-3$。此外，算法 *1* 正确，运行时间为 $O(mn)$。

证明。设 $\boldsymbol{x} \in \mathcal{C}$ and $\boldsymbol{y} \in RC_{k,(i_1,i_2,\ldots,i_t)}(\boldsymbol{x})$，其中 $1 \leqslant i_1 < \cdots i_t \leqslant n-k+1$ and $i_j - i_{j-1} \geqslant k$ for $j \in [2,t]$。我们将证明算法 1 输出正确的 $\boldsymbol{x}$。在从步骤 2 到步骤 10 的循环中，我们首先找到满足 $\boldsymbol{y}_{[p_1+3m-3,p_1+6m-7]} = \boldsymbol{y}^{RC}_{[p_1,p_1+3m-4]}$ 的最小索引 p_1。根据引理 III.3，我们有 $i_1 = p_1 + 3m - 3 - k$。在步骤 4 中，我们通过移除插入的子串 $\boldsymbol{y}_{[i_1+k,i_1+2k-1]} = \boldsymbol{y}_{[p_1+3m-3,p_1+3m-3+k-1]}$ 执行去重操作，并更新 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,p+3m-4]}\boldsymbol{y}_{[p+3m-3+k,n']}$。回想一下， n' 表示 $\boldsymbol{y}$ 的长度；在步骤 5 中，我们更新 $n' \leftarrow n' - k$。由于我们已纠正最左侧的重复错误，步骤 6 将 t 减一。现在， i_2 表示第一个重复错误的位置，而 $i_2 - i_1 \geqslant k$，我们要寻找索引 $p_2 = i_2 + k - 3m + 3 \geqslant i_1 + 2k - 3m + 3 = p_1 + k$，使得$\boldsymbol{y}_{[p_2+3m-3,p_2+6m-7]} = \boldsymbol{y}^{RC}_{[p_2,p_2+3m-4]}$，因此在步骤 7 中我们跳过 k 个位置。循环结束后， t 的值减为零，这意味着所有重复错误均已纠正。因此，算法输出正确的 $\boldsymbol{x}$。

接下来，我们分析该算法的复杂度。由于 $i_t \leqslant n-k+1$，在循环（步骤 2–10）的最后一次执行中，我们有 $p = i_t + k - 3m + 3 \leqslant n$。因此，该循环最多执行 n 次。每次迭代中，我们比较两个长度为 $3m-3$ 的子串。因此，总体时间复杂度为 $O(mn)$。证明至此完成。 □

B. RCD 根

为了评估我们的码，我们需要计算 m-RCD 根的总数。

引理 III.7. 设 $A(n,\ m)$ 是 Σ_q^n 中 m-*RCD* 根的数量。若 $m \geqslant \lceil \log_q n \rceil + 1$，则 $A(n,m) \geqslant (q-1)q^{n-1}$。

证明。 我们通过概率分析证明该引理。均匀随机地选取一个字符串 $\boldsymbol{x} \in \Sigma_q^n$，令 $\mathbb{P}$ 为 $\boldsymbol{x}$ 不满足 m-RCD 根性质的概率，即存在某个 $i \in [1,\ n-2m+1]$，使得 $\boldsymbol{x}_{[i+m,\ i+2m-1]} = \boldsymbol{x}^{RC}_{[i,\ i+m-1]}$。我们将证明 $\mathbb{P} \leqslant \frac{1}{q}$。

对于 $i \in [1,\ n-2m+1]$, $\boldsymbol{x}_{[i+m,\ i+2m-1]} = \boldsymbol{x}^{RC}_{[i,\ i+m-1]}$ 的概率为 $\frac{1}{q^m} \leqslant \frac{1}{nq}$。根据并集界，我们可以计算出 $\mathbb{P} \leqslant (n-2m+1)\cdot\frac{1}{q^m} \leqslant \frac{1}{q}$。然后得到 $A(n,\ m) \geqslant q^n(1-\frac{1}{q}) = (q-1)q^{n-1}$。至此，证明完成。 □

一个自然的问题是，如何高效地将字符串编码为 m-RCD 根。接下来，我们采用 [15, 构造 2]中提出的一般框架，在满足 $m \geqslant \lceil \log_q n \rceil + 1$ 的条件下，将字符串编码为仅含一个冗余符号的 m-RCD 根。

算法2 通用迭代编码器 ENC

输入： $\boldsymbol{x} \in \Sigma_q^{n-1}$ **输出：** $\boldsymbol{y} = ENC(\boldsymbol{x}) \in \mathcal{C} \subseteq \Sigma_q^n$

1: $\boldsymbol{y} \leftarrow \boldsymbol{x}$

2: **当** $\mathbb{1}_{\mathcal{C}}(\boldsymbol{y}) = 0$ **时**

3: $\boldsymbol{y} \leftarrow \xi(\boldsymbol{y})$

4: **结束循环**

5: **返回** $\boldsymbol{y}$

算法3 通用迭代译码器 DEC DEC

输入： $\boldsymbol{y} = ENC(\boldsymbol{x}) \in \mathcal{C}$ 对于某个 $\boldsymbol{x} \in \Sigma_q^{n-1}$ **输出：** $\boldsymbol{x} = DEC(\boldsymbol{y})$

1: **当** $y_n = 0$ **时**

2: $\boldsymbol{y} \leftarrow \xi^{-1}(\boldsymbol{y}_{[1,\ n-1]})$ ▷ ξ^{-1} 表示 ξ的“逆”

3: **结束循环**

4: $\boldsymbol{x} \leftarrow \boldsymbol{y}_{[1,\ n-1]}$

5: **返回** $\boldsymbol{x}$

引理III.8。 *[15, 构造 2]* 设 $\mathcal{C} \subseteq \Sigma_q^n$是给定的约束，满足 $|\mathcal{C}| \geqslant q^{n-1}$。给定单射函数 $\xi\colon \Sigma_q^n \backslash \mathcal{C} \to \Sigma_q^{n-1}$和指示函数 $\mathbb{1}_{\mathcal{C}}\colon \Sigma_q^n \to \{0,1\}$，则算法 *2* 和 *3* 给出了一种高效的信道构造，仅需一个冗余符号，且 $O\big(T(n)\big)$ 平均时间复杂度为，其中 $T(n)$ 表示 $\xi, \xi^{-1}, \mathbb{1}_{\mathcal{C}}$ 中的最大时间复杂度。

定理III.9。 令 $m = \lceil \log_q n \rceil + 1$。设 $\mathcal{C} \subseteq \Sigma_q^n$为 *m-RCD* 根的集合。我们定义函数 $\xi\colon \Sigma_q^n \backslash \mathcal{C} \to \Sigma_q^{n-1}$, $\xi^{-1}\ \ \Sigma_q^{n-1} \to \Sigma_q^n \qquad \mathbb{1}_{\mathcal{C}}\ \ \Sigma_q^n \to \{0\ 1\}$:和 :，如下所示：

• 对于任意 $\boldsymbol{x} \notin \mathcal{C}$, $\xi(\boldsymbol{x}) = \boldsymbol{x}_{[1,\ i+m-1]}\boldsymbol{x}_{[i+2m\ \ \ \

备注 III.11. 回想一下，互补操作可以视为一个置换 $\pi:\ \Sigma_q \to \Sigma_q$，其中对每个 $a \in \Sigma_q$，都有 $\pi(a) \neq a$ 和 $\pi\big(\pi(a)\big) = a$。由于本文确立的所有结果均不要求条件 $\pi(a) \neq a$，因此将互补操作替换为恒等置换后，这些结论依然成立。因此，只需将“反向互补”的实例替换为“回文”，本文确立的所有结果便可直接扩展至处理回文重复。

IV. 纠正任意长度的回文/反向互补重复

Liu、Tang、Fan 和 Sagar [24]，以及 Lenz、Wachter-Zeh 和 Yaakobi [19] 分别证明，存在能够纠正任意长度反向互补重复和回文复制的码，其码率为 $\log_q(q-1)$。本节将为这类码推导 Gilbert-Varshamov 界，证明最优冗余至多为 $2\log_q n + \log_q \log_q n + O(1)$，即渐近最优码率为 1。下文仅讨论反向互补重复，因为只需将“反向互补”替换为“回文”，本文的所有结果便可直接扩展至回文复制。在继续讨论之前，我们先介绍一些必要的图论工具。

定义 IV.1. 设 $G = (V, E)$ 是一个图，其中 V 是顶点集，E 是边集。对于顶点 $\boldsymbol{v} \in V$，令 $d(\boldsymbol{v})$ 表示其度，即与 $\boldsymbol{v}$ 相邻的顶点数。G 中的独立集是 V 的一个子集，其中任意两个不同顶点之间都没有边相连。G 的独立数 记为 $\alpha(G)$，等于 G 中最大独立集的基数。

引理 IV. 2. *[1, 第 100 页，定理 1]* 对于任意图 $G = (V, E)$，独立数满足

$$\alpha(G) \geqslant \sum_{\boldsymbol{v} \in V} \frac{1}{d(\boldsymbol{v}) + 1}.$$

现在考虑图 $G^* = (V,\ E)$，其中 V 是 Σ_q^n 的子集。两个不同的顶点 $\boldsymbol{x},\ \boldsymbol{z} \in V$ 相邻，即 $(\boldsymbol{x},\ \boldsymbol{z}) \in E$，当且仅当存在某个 $k \geqslant 1$ 使得 $RC_k^1(\boldsymbol{x}) \cap RC_k^1(\boldsymbol{z}) \neq \varnothing$。在此背景下，码 $\mathcal{C} \subseteq V$ 能够纠正任意长度的反向互补重复，当且仅当 $\mathcal{C}$ 构成 G^* 中的一个独立集。因此，此类码 $\mathcal{C}$ 的最大大小等于 $\alpha(G^*)$。

为了确立 $\alpha(G^*)$ 的下界，我们应用引理 IV.2。这需要推导每个顶点 $\boldsymbol{x} \in V$ 的度 $d(\boldsymbol{x})$ 的上界。为此，我们定义

$$N_k(\boldsymbol{x}; V) = \left\{\boldsymbol{z} \in V : \boldsymbol{z} \neq \boldsymbol{x} \text{ and } RC_k^1(\boldsymbol{z}) \cap RC_k^1(\boldsymbol{x}) \neq \varnothing\right\}.$$

引理 IV.3. 对于任意 $V \subseteq \Sigma_q^n$ 和 $\boldsymbol{x} \in V$，有

$$\#N_k(\boldsymbol{x}; V) \leqslant n(n-k+1) - 1.$$

证明. 每个顶点 $\boldsymbol{z} \in N_k(\boldsymbol{x}; V) \cup \{\boldsymbol{x}\}$ 都可以通过以下两种操作从 $\boldsymbol{x}$ 得到：
1）进行长度为 k 的反向互补重复，得到长度为 $n+k$ 的字符串 $\boldsymbol{y}$。选择重复起始位置的方式至多有 $n-k+1$ 种，因此此类 $\boldsymbol{y}$ 的数量至多为 $n-k+1$。2）进行长度为 k 的反向互补去重复，得到长度为 n 的字符串 $\boldsymbol{z}$。选择去重复起始位置的方式至多有 n 种，因此此类 $\boldsymbol{z}$ 的数量至多为 $n(n-k+1)$。

因此，$\#N_k(\boldsymbol{x}; V) \leqslant n(n-k+1) - 1$。证明至此完成。 □

利用引理 IV.3，我们计算得

$$d(\boldsymbol{x}) \leqslant \sum_{k=1}^{n} \#N_k(\boldsymbol{x}; V) \leqslant \sum_{k=1}^{n} \big(n(n-k+1) - 1\big) \leqslant n^3 - 1.$$

由引理 IV.2，可得

$$\alpha(G^*) \geqslant \frac{\#V}{n^3}.$$

令 $V = \Sigma_q^n$（即所有可能的字符串），我们得到以下结果。

引理 IV.4. 存在一种码，能够纠正任意长度的反向互补重复，且最多只需$3\log_q n$ 个冗余符号。

这　　以下定理通过选取所有 m-RCD 根作为集合 V，给出了更优的界　　.

定理 IV.5. 存在一种码，能够纠正任意长度的反向互补重复，且至多需要$2\log_q n + \log_q \log_q n + O(1)$ 个冗余符号，即渐近码率为1。

证明。令 $G^* = (V, E)$ 如上定义。令 V为满足 $m = \lceil \log_q n \rceil + 1$ 的 m-RCD 根的集合。由引理 III.2，得 $\#V \geqslant (q-1)q^{n-1}$。此外，由定理 III.6，得 $\#N_k(\boldsymbol{x};\ V) = 0$ 当 $k \geqslant 3\lceil \log_q n \rceil$ 时。因此，由引理 IV.3，我们可以计算

$$d(\boldsymbol{x}) \leqslant \sum_{k=1}^{3\lceil \log_q n \rceil - 1} \#N_k(\boldsymbol{x}; V) \leqslant 3n^2 \lceil \log_q n \rceil - 1.$$

最后，根据引理 IV.2，我们得到

$$\alpha(G^*) \geqslant \frac{(q-1)q^{n-1}}{3n^2\lceil \log_q n \rceil} = O\left(\frac{q^n}{n^2 \log_q n}\right).$$

于是结论得证。 □

注 IV.6。在本注中，我们概述一种可能的方法，用于构造能够纠正任意长度的反向互补重复的码，其冗余优于定理 *IV.5* 中推导出的 *Gilbert-Varshamov*界。

基于 *Bitar*、*Hanna*、*Polyanskii* 和 *Vorobyev* 的工作 *[4],*——他们构造了一种能够纠正长度至多为 $k = O\left(\frac{n}{\log_2 n)^2}\right)$ 的突发插入的二进制码 $\mathcal{C}_1 \subseteq \Sigma_2^n$，其冗余为 $\overline{\log_2 n} + O\left(k(\log_2(k \log_2 n))^2\right)$——我们提出如下构造。

令 $\mathcal{C}_2 \subseteq \Sigma_2^n$ 为满足 $m \leqslant \binom{k+4}{3}$ 的 m-*RCD* 根的集合。根据定理 *III.6*， $\mathcal{C}_2$ 能够纠正长度至少为 $k+1$ 的反向互补重复。于是，码 $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$ 能够纠正任意长度的反向互补重复。通过恰当地选择参数 k，可以使 $\mathcal{C}$ 的冗余至多为 $\log_2 n + o(\log_2 n)$。该构造可以推广到 q-元字母表，因为 q-元符号可以视为长度为 $\lceil \log_2 q \rceil$ 的二进制字符串。

V. Correcting Length-One Reverse-Complement Duplications

本节研究用于纠正短反向互补重复的码。Ben-Tolila 和 Schwartz [3] 证明，当重复长度 k 为奇数时，任何能够纠正长度为 k 的突发插入的二进制码，都可以通过保持互补映射构造出能够纠正长度为 k的反向互补重复的 q元码。

定义 V.1。设 $\beta\colon\ \Sigma_q \to \Sigma_2$ 是一个映射。若 β 满足互补保持 条件，即 $\beta(a) = \beta(\overline{a})$ 对 $a \in \Sigma_q$成立，也就是说，β与互补操作可交换，则称该映射为互补保持映射。此互补保持映射自然地扩展到整个字符串，即对于任意$\boldsymbol{x} = x_1x_2\cdots x_n \in \Sigma_q^n$，定义 $\boldsymbol{\beta}(\boldsymbol{x}) = \beta(x_1)\beta(x_2)\cdots\beta(x_n)$。

引理 V.2。 *[3, 定理 17]* 设 $\beta\colon\ \Sigma_q \to \Sigma_2$ 是一个互补保持映射，且 k 是一个奇整数。若 $\mathcal{C}' \subseteq \Sigma_2^n$ 能够纠正长度为 k的突发插入，则码 $\mathcal{C} \triangleq \left\{\boldsymbol{x} \in \Sigma_q^n\colon\ \boldsymbol{\beta}(\boldsymbol{x}) \in \mathcal{C}'\right\}$ 能够纠正长度为 k的反向互补重复。

一个自然的问题是，这一结论是否也适用于 k，以及涉及多个反向互补重复的情形。遗憾的是，答案是否定的。

例 V.3. 设 $n = 6$、 $q = 4$、 $0 \triangleq \overline{1}$和 $2 = \overline{3}$。令 $\beta\colon\ \Sigma_q \to \Sigma_2$ 为一个互补保持映射，使得 $\beta(0) = \beta(2) = 0$ 且 $\beta(1) = \beta(3) = 1$。定义 $\mathcal{C}' = \{010101\} \subseteq \Sigma_2^n$ 和 $\mathcal{C} = \{232301、\ 230101\} \subseteq \Sigma_q^n$。由于 $\mathcal{C}'$ 仅包含一个码字，因此它显然能够纠正 $t\ k$ 个反向互补重复，其中 t、$k \geqslant 1$。此外，由于 $\boldsymbol{\beta}(232301) = \boldsymbol{\beta}(230101) = 010101$，我们有 $\mathcal{C} \subseteq \left\{\boldsymbol{x} \in \Sigma_q^n\colon\ \boldsymbol{\beta}(\boldsymbol{x}) \in \mathcal{C}'\right\}$。我们将证明， $\mathcal{C}$ 无法纠正长度为二的反向互补重复，也无法纠正两个长度为一的反向互补重复。

- 若 $k = 2$ 且 $t = 1$，则有 $232301\underline{\ } \to 23230101$ 和 $23\underline{01}01 \to 23230101$，其中带下划线的子串表示重复的部分。这意味着，232301 和230101 都可以通过一次长度为二的反向互补复制演化为23230101 。因此， $\mathcal{C}$ 无法纠正一次长度为二的反向互补复制。
- 若 $k = 1$ 且 $t = 2$，则有 $232301 \to 2323010 \to 23230101$ 和 $230101 \to 2320101 \to 23230101$。这意味着，$23\overline{2}301$ 和230101 都可以通过两次长度为一的反向互补复制演化为 23230101 。因此， $\mathcal{C}$ 无法纠正两次长度为一的反向互补复制。

这促使我们开发新的工具，以处理偶数长度的反向互补复制或多次反向互补复制。下文中，我们将重点研究多次反向互补复制，并限制每次复制的长度均为一。我们首先考察长度为一的反向互补复制与插入之间的相似之处和区别。设 $\boldsymbol{x} \in \Sigma_q^n$。回顾一下， $RC_1^1(\boldsymbol{x})$ 表示以 $\boldsymbol{x}$ 为中心、半径为一的 1-反向互补球。设$I_1(\boldsymbol{x})$ 表示以 $\boldsymbol{x}$为中心的 1-插入球，即通过恰好一次插入从 $\boldsymbol{x}$ 得到的字符串集合。

引理 V.4. 对于任意 $\boldsymbol{x} \in \Sigma_q^n$，都有

$$RC_1^1(\boldsymbol{x}) = \{\boldsymbol{x}_{[1,i]}\overline{x}_i\boldsymbol{x}_{[i+1,n]} : i \in [1,n]\},$$
$$I_1(\boldsymbol{x}) = \{a\boldsymbol{x} : a \in \Sigma_q\} \cup \{\boldsymbol{x}_{[1,i]}a\boldsymbol{x}_{[i+1,n]} : i \in [1,n], a \in \Sigma_q, a \neq x_i\}.$$

证明。根据定义，我们有 $RC_1^1(\boldsymbol{x}) = \{\boldsymbol{x}_{[1,\ i]}x_i\boldsymbol{x}_{[i+1,\ n]}:\ i \in [1,\ n]\}$ 以及 $I_1(\boldsymbol{x}) = \{a\boldsymbol{x}:\ a \in \Sigma_q\} \cup \{\boldsymbol{x}_{[1,\ i]}a\boldsymbol{x}_{[i+1,\ n]}:\ i \in [1,\ n],\ a \in \Sigma_q\}$。考虑序列 $\boldsymbol{x}_{[1,\ i]}x_i\boldsymbol{x}_{[i+1,\ n]}$，其中 $i \in [1,\ n]$，

- 若 $x_j = x_i$ 对 $j \in [1,\ i]$，则 $\boldsymbol{x}_{[1,\ i]}x_i\boldsymbol{x}_{[i+1,\ n]} = x_i\boldsymbol{x} \in \{a\boldsymbol{x}:\ a \in \Sigma_q\}$；
- 若 $x_j \neq x_i$ 对某个 $j \in [1,\ i]$，令 $p \in [1,\ i-1]$ 为满足 $x_p \neq x_i$的最大索引，则 $\boldsymbol{x}_{[1,\ i]}x_i\boldsymbol{x}_{[i+1,\ n]} = \boldsymbol{x}_{[1,\ p]}x_i\boldsymbol{x}_{[p+1,\ n]} \in \{\boldsymbol{x}_{[1,\ p]}a\boldsymbol{x}_{[p+1,\ n]}:\ a \in \Sigma_q,\ a \neq x_p\}$。

因此，我们有 $I_1(\boldsymbol{x}) = \{a\boldsymbol{x}:\ a \in \Sigma_q\} \cup \{\boldsymbol{x}_{[1,\ i]}a\boldsymbol{x}_{[i+1,\ n]}:\ i \in [1,\ n],\ a \in \Sigma_q,\ a \neq x_i\}$。证明至此完成。 □

引理 V.4 表明，在二元情形下，长度为一的反向互补重复与插入几乎没有区别，而在非二元情形下，这种区别则变得显著。因此，很难设计出在纠正长度为一的反向互补重复方面优于一般插入纠错码的二进制码 [11], [20], [22], [28], [31], [32]，接下来我们转向非二元字母表。在本节余下部分，我们假设 $q \geqslant 4$ 是一个固定的偶整数，并将 $a \in \Sigma_q^n$ 的补定义为

$$\overline{a} = \begin{cases} 2\lfloor a/2 \rfloor + 1, & \text{if } 2|a; \\ 2\lfloor a/2 \rfloor, & \text{if } 2 \nmid a. \end{cases}$$

换言之，令 $i = \lfloor a/2 \rfloor$，则 $\{\overline{a},\ a\} = \{2i,\ 2i+1\}$。显然，这个互补操作定义良好 d.

定义 V.5。对于任意 $a \in \Sigma_q$，令

$$a^{\oplus} = a\{a, \overline{a}\}^*$$

表示所有以 a 开头、其后跟随仅由 a 和 $\overline{a}$ 组成的任意有限长度字符串的集合。在字符串 $\boldsymbol{x} \in \Sigma_q^n$ 中，若子串 $\boldsymbol{x}_{[i,\ j]}$ 满足 $1 \leqslant i \leqslant j \leqslant n$，则称其为一个游程[2]，如果 $\boldsymbol{x}_{[i,\ j]} \in x_i^{\oplus}$ 且其长度达到最大，即 $x_{i-1} \notin \{x_i, \overline{x_i}\}$（当 $i > 1$ 时）以及 $x_{j+1} \notin \{x_i, \overline{x_i}\}$（当 $j < n$ 时）。

定义 V.6。对于任意 $\boldsymbol{x} \in \Sigma_q^n$，令 $r(\boldsymbol{x})$ 表示 $\boldsymbol{x}$ 中游程的数量。对于 $i \in [1, r(\boldsymbol{x})]$，令 $\boldsymbol{\sigma}(\boldsymbol{x}, i)$ 表示 $\boldsymbol{x}$ 的第 i 个游程，并令 $\sigma(\boldsymbol{x})_i = \sigma(\boldsymbol{x}, i)_1$ 表示 $\boldsymbol{\sigma}(\boldsymbol{x}, i)$ 的第一个元素，随后我们将 $\boldsymbol{x}$ 的特征 定义为

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \big(\sigma(\boldsymbol{x})_1, \sigma(\boldsymbol{x})_2, \cdots, \sigma(\boldsymbol{x})_{r(\boldsymbol{x})}\big).$$

例 V.7. 设 $q = 4$，则 $0 = 1$ 且 $2 = 3$。考虑 $\boldsymbol{x} = 01123221001$，则

- $r(\boldsymbol{x}) = 3$；
- $\boldsymbol{\sigma}(\boldsymbol{x}, 1) = 011$，$\boldsymbol{\sigma}(\boldsymbol{x}, 2) = 2322$，以及 $\boldsymbol{\sigma}(\boldsymbol{x}, 3) = 1001$；
- $\sigma(\boldsymbol{x})_1 = 0$，$\sigma(\boldsymbol{x})_2 = 2$，以及 $\sigma(\boldsymbol{x})_3 = 1$；
- $\boldsymbol{\sigma}(\boldsymbol{x}) = (0,\ 2,\ 1)$。

下面的引理是 [39, 引理 4] 的一个变体。为完整起见，我们在此给出完整证明。

引理 V.8. 设 $t \geqslant 1$ 为整数。对任意 $\boldsymbol{x} \in \Sigma_q^n$ 与 $\boldsymbol{y} \in RC_1^t(\boldsymbol{x})$，则

- $\boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{\sigma}(\boldsymbol{y})$，即对 $i \in [1, r(\boldsymbol{x})]$ 有 $r(\boldsymbol{x}) = r(\boldsymbol{y})$ 且 $\sigma(\boldsymbol{x})_i = \sigma(\boldsymbol{y})_i$；
- 存在某些非负整数 $t_1, t_2, \ldots\ldots, t_{r(\boldsymbol{x})}$，满足 $t_1 + t_2 + \cdots + t_{r(\boldsymbol{x})} = t$，使得对 $i \in [1, r(\boldsymbol{x})]$ 有 $\boldsymbol{\sigma}(\boldsymbol{y}, i) \in RC_1^{t_i}\big(\boldsymbol{\sigma}(\boldsymbol{x}, i)\big)$。

证明。设 $\boldsymbol{x} = \boldsymbol{y}^{(0)}$ 和 $\boldsymbol{y}^{(t)} = \boldsymbol{y}$。由于 $\boldsymbol{y} \in RC_1^t(\boldsymbol{x})$，存在 $t-1$ 个中间序列 $\boldsymbol{y}^{(1)}, \ldots\ldots, \boldsymbol{y}^{(t-1)}$，使得$\boldsymbol{y}^{(i)} \in RC_1^1\big(\boldsymbol{y}^{(i-1)}\big)$ 对 $i \in [1,\ t]$成立。假设 $\boldsymbol{y}^{(i)}$ 是通过在 $\boldsymbol{y}^{(i-1)}$的第 i_j个游程中的某个位置进行一次重复而从 $\boldsymbol{y}^{(i-1)}$ 得到的，那么这次重复的效果只是将 $\boldsymbol{y}^{(i-1)}$ 的第 i_j个游程延长一个符号，同时保持其首符号不变，且不影响其他游程。这意味着

$$\boldsymbol{\sigma}\big(\boldsymbol{y}^{(i)}\big) = \boldsymbol{\sigma}\big(\boldsymbol{y}^{(i-1)}\big), \quad \boldsymbol{\sigma}\big(\boldsymbol{y}^{(i)}, i_j\big) \in RC_1^1\big(\boldsymbol{\sigma}(\boldsymbol{y}^{(i-1)}, i_j)\big), \quad \text{and} \quad \boldsymbol{\sigma}\big(\boldsymbol{y}^{(i)}, s\big) = \boldsymbol{\sigma}\big(\boldsymbol{y}^{(i-1)}, s\big) \quad \text{for } s \neq i_j.$$

于是结论得证。 □

若将每个游程记作一个特殊符号并应用引理 V.8，则长度为一的反向互补重复可视为替换。因此，我们可以利用汉明度量中的工具来纠正反向互补重复。

A. 游程长度受限约束

由于某些游程的长度可能为 $O(n)$，将每个游程视为一个符号会得到一个定义在较大字母表上的向量，因此需要大量冗余才能纠正其中的替换。为了减少冗余，最好限制游程的最大长度，使该向量定义在较小的字母表上。为此，我们引入游程长度受限字符串的概念。

[2]在传统定义中（例如 [26] 中），游程是由单个符号构成且长度最大的子串。

定义 V.9。 若字符串 $\boldsymbol{x} \in \Sigma_q^n$ 中每个游程的长度都不超过 m，则称其为 m-游程长度受限字符串 （简称 m-RLL 字符串），即

$$|\boldsymbol{\sigma}(\boldsymbol{x}, i)| \leqslant m \quad \textit{for } i \in [1, r(\boldsymbol{x})].$$

引理 V.10。 令 $B(n,m)$ 为 Σ_q^n 中 m-*RLL* 字符串的数量。若 $q \geqslant 4$ 为偶数且 $m \geqslant \lfloor \log_{q/2} n \rceil + 1,$，则 $B(n, m) \geqslant q^n - 2q^{n-1} \geqslant \frac{q^n}{2}$。—

证明。 我们通过概率分析证明该引理。均匀随机地选取一个字符串 $\boldsymbol{x} \in \Sigma_q^n$，令 $\mathbb{P}$ 为 $\boldsymbol{x}$ 不满足 m-RLL 约束的概率，即存在某个 $i \in [1,\ n-m]$，使得 $\boldsymbol{x}_{[i,\ i+m]}$ 是一个游程。我们将证明 $\mathbb{P} \leqslant \frac{1}{2}$。

对于 $i \in [1,\ n-m]$，$\boldsymbol{x}_{[i,\ i+m]}$ 是一个游程的概率可计算为 $\frac{q \cdot 2^m}{q^{m+1}} = \frac{1}{(q/2)^m} \leqslant \frac{2}{qn^{\circ}}$根据并集界，我们可以计算出 $\mathbb{P} \leqslant (n-m) \cdot \frac{2}{qn} \leqslant \frac{2}{q}$。于是得到 $B(n,\ m) \geqslant q^n(1 - \frac{2}{q}) = q^n - 2q^{n-1}$。证明完毕。 □

接下来，我们使用序列替换技术设计一个高效编码器，在满足 $m = \lfloor \log_q n \rceil + \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil + 1$ 的条件下，将给定字符串映射为一个带有一个冗余符号的 m-RLL 字符串。需要注意的是，当 $\log_q n$ 和 $(\log_q n + 1) \cdot \log_{q/2} 2$ 为整数时，我们有

$$\begin{aligned} \lceil \log_q n \rceil + \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil + 1 &= (1 + \log_{q/2} 2) \cdot \log_q n + \log_{q/2} 2 + 1 \\ &= \log_{q/2} n + \log_{q/2} 2 + 1. \end{aligned}$$

因此，这一 m 的选择接近引理 V.10 中推导出的下界。对 m 作此选择的理由如下：当存在长度超过 m 的游程时，我们首先从中移除一个长度为 m 的子串。为保持整体长度不变并使这一过程可逆，接着用 $\lfloor \log_q n \rceil$ 个符号记录被移除子串的位置，用 $\left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil$ 个符号记录被移除子串的信息（其有效性将在下文证明），并用一个标记符号指示是否进行过移除。

引理 V.11. 设 $m = \lceil \log_q n \rceil + m' + 1$。当且仅当 $q^{m'} \geqslant 2^m$ 时，$m' \geqslant \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2$。证明。注意到

$$\begin{aligned} q^{m'} \geqslant 2^m &\Leftrightarrow (q/2)^{m'} \geqslant 2^{\lceil \log_q n \rceil + 1} \\ &\Leftrightarrow m' \cdot \log_2(q/2) \geqslant \lceil \log_q n \rceil + 1 \\ &\Leftrightarrow m' \geqslant \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2. \end{aligned}$$

证明至此完成。 □

以下函数将用于 RLL 编码器的设计。

定义 V.12. 对于任意满足 $q \geqslant 4$ 为偶数的 $a \in \Sigma_q$，定义

$$\begin{aligned} h_1(a) &= 2\lfloor a/2 \rfloor + 3 \pmod q, \\ h_2(a) &= 2\lfloor a/2 \rfloor + 2 \pmod q. \end{aligned}$$

引理 V.13. 设 $h_1(\cdot)$ 和 $h_2(\cdot)$ 如上所定义。对于任意满足 $q \geqslant 4$ 为偶数的 $a \in \Sigma_q$，$h_1(a) \notin \{a$ 为奇数，且 $h_2(a) \notin \{a,\ a\}$ 为偶数。

证明。 设 $i = \lfloor a/2 \rfloor,$，则 $\{a, {}^{-}a\} = \{2i,\ 2i+1\}$。由此可得，$h_1(a) = 2i+3$（模 $q) \notin \{2i,\ 2i+1\}$ 为奇数，而 $h_2(a) = 2i+2$（模 $q) \notin \{2i,\ 2i+1\}$ 为偶数。证明至此完成。 □

定义 V.14. 对于任意满足 $a_1 \in \Sigma_q$ 的游程字符串 $a_1 a_2 \cdots a_m \in a_1^{\oplus}$，我们将其指示向量 定义为

$$\mathbb{1}(a_1 a_2 \cdots a_m) = \big(\mathbb{1}(a_1), \mathbb{1}(a_2), \ldots, \mathbb{1}(a_m)\big),$$

其中，对于 $i \in [1,$ 若 $a_i = a_1$ 则 $\mathbb{1}(a_i) = 1$，否则为 $\mathbb{1}(a_i) = 0$；此外，我们将$\mathbb{1}(a_1 a_2 \cdots a_m)$ 的整数表示 定义为

$$\varphi\big(a_1 a_2 \cdots a_m\big) = \sum_{j=1}^{m} 2^{j-1} \cdot \mathbb{1}(a_{m+1-j}).$$

注意，$\mathbb{1}(a_1) = 1$ 和 $\varphi(a_1 a_2 \cdots a_m) \in [2^{m-1},\ 2^m - 1]$。因此，游程 $a_1 a_2 \cdots a_m$ 的长度由$\varphi(a_1 a_2 \cdots a_m)$决定。由此，我们可以从 $\varphi(a_1 a_2 \cdots a_m)$恢复 $\mathbb{1}(a_1 a_2 \cdots a_m)$。如果 a_1 是先验已知的，[3] 我们还可以进一步恢复 $a_1 a_2 \cdots a_m$。这意味着，当 a_1 是先验已知的时，由下式定义的逆映射 φ^{-1}

$$\varphi^{-1}\big(\varphi(a_1 a_2 \cdots a_m), a_1\big) = a_1 a_2 \cdots a_m \tag{1}$$

[3]这是一个 na 然的假设，因为长度为一的反向互补重复不会改变游程的起始符号。

定义良好。

例 V.15. 接着例 *V.7*，我们考虑字符串 $\boldsymbol{x} = 01123221001$，然后

- $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x},1)\big) = 100$, $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x},2)\big) = 1011$，以及 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x},3)\big) = 1001$;
- $\varphi\big(\boldsymbol{\sigma}(\boldsymbol{x},1)\big) = 4$, $\varphi\big(\boldsymbol{\sigma}(\boldsymbol{x},2)\big) = 11$，以及 $\varphi\big(\boldsymbol{\sigma}(\boldsymbol{x},3)\big) = 9$。

算法 4 m-RLL 编码器，适用于 $m = m_1 + m_2 + 1$，其中 $m_1 = \lceil \log_q n \rceil$ 和 $m_2 = \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil$

输入： $\boldsymbol{x} \in \Sigma_q^{n-1}$ **输出：** $\boldsymbol{y} = ENC_m^{RLL}(\boldsymbol{x}) \in \Sigma_q^n$ 是一个 m-RLL 字符串1:
初始化： 令 $\boldsymbol{y} = \boldsymbol{x} \circ h_1(x_{n-1})$, $i = 1$，以及 $i_{end} \triangleq n - 1 - m$2: **当** $i \leqslant i_{end}$ **时**3: 令 $j > i$ 为满足条件的最小索引 $y_j \notin \{x_i, x_i\}$4: **当** $j \geqslant i + m + 1$ **时**5: $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,i]} \circ \boldsymbol{y}_{[i+m+1,n]} \circ Rep_{q,m_2}\big(\varphi(\boldsymbol{y}_{[i,i+m]}) - 2^m\big) \circ Rep_{q,m_1}(i)$

▷ $Rep_{q,\ m_1}(i)$ 表示 q元表示 i，长度为 $m_1$6:

$\boldsymbol{y} \leftarrow \boldsymbol{y} \circ h_2(y_{n-1})$7: $j \leftarrow j - m$8:
$i_{end} \leftarrow i_{end} - m$9: **结束循环**10:
$i \leftarrow j$11: **结束循环**12: **返回** $\boldsymbol{y}$

---

5 m-RLL 解码器，用于 $m = m_1 + m_2 + 1$，其中 $m_1 = \lceil \log_q n \rceil$ 和 $m_2 = \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil$

输入： $\boldsymbol{y} = ENC_m^{RLL}(\boldsymbol{x}) \in \Sigma_q^n$ 对于某个 $\boldsymbol{x} \in \Sigma_q^{n-1}$ **输出：** $\boldsymbol{x} = DEC_m^{RLL}(\boldsymbol{y})$1: **当** $2 | y_n$ **执行**2 $i \leftarrow Rep_{q\ m_1}^{-1}(\boldsymbol{y}_{[n-m_1\ n-1]})$: ，，：
3 $a \leftarrow Rep_{q\ m_2}^{-1}(\boldsymbol{y}_{[n-m_1-m_2\ n-m_1-1]}) + 2^m$，，：
4 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,\ i-1]} \circ \varphi^{-1}(a,\ y_i) \circ \boldsymbol{y}_{[i+1,\ n-m]}$5: **结束循环**6: $\boldsymbol{x} \leftarrow \boldsymbol{y}_{[1,\ n-1]}$7: **返回** $\boldsymbol{x}$

现在我们可以介绍我们的编码器和解码器了。

定理 V.16. 设 $q \geqslant 4$ 和 $m = m_1 + m_2 + 1$ 满足 $m_1 = \lceil \log_q n \rceil$ 且 $m_2 = \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil$。算法 *4* 和 *5* 是正确的，即对于任意 $\boldsymbol{x} \in \Sigma_q^{n-1}$，有

- $ENC_m^{RLL}(\boldsymbol{x}) \in \Sigma_q^n$是一个 *m-RLL* 字符串；
- $DEC_m^{RLL}\big(ENC_m^{RLL}(\boldsymbol{x})\big) = \boldsymbol{x}$。

此外，算法 *4* 和 *5* 的运行时间为 $O(n)$。

证明。我们首先考察算法 4 的正确性。在步骤 1 中，将一个奇符号 $h_1(x_{n-1}) \notin \{x_{n-1}, x_{n-1}\}$ 追加到$\boldsymbol{x}$的末尾，得到长度为 n 的字符串 $\boldsymbol{y}$。在步骤 2 至 11 的循环中，我们找出长度超过 m的最左侧游程。令 i 和 $j-1$ 分别表示该游程的起始位置和结束位置。在内层循环（步骤 4 至 9）中，我们将选定游程缩短 m，同时在末尾追加长度为 m 的字符串，以保持整体长度不变；此过程持续进行，直到游程长度不超过 m。具体而言，我们首先移除子串 $\boldsymbol{y}_{[i+1,i+m]}$（注意，$y_i$ 会保留），并在末尾追加一个长度为 m_1的字符串，用于编码 $\boldsymbol{y}_{[i,i+m]}$的信息；追加一个长度为 m_2 的字符串，用于编码 i的位置；再追加一个标记符号，用于指示循环迭代并防止产生长度超过 m的新游程。由于 $i < n \leqslant q^{\lceil \log_q n \rceil} = q^{m_1}$，$i$的位置可以用 $Rep_{q,m_1}(i)$编码。此外，由于 $\boldsymbol{y}_{[i,\ i+m]} \in y_i^{\oplus}$的长度为 $m+1$且 y_i会保留，根据定义 V.14，我们可以使用双射 $\varphi(\cdot)$将其映射为 $\varphi(\boldsymbol{y}_{[i,i+m]}) \in [2^m, 2^{m+1}-1]$。由于 m的值是先验已知的，因此只需记录 $\varphi(\boldsymbol{y}_{[i,i+m]}) - 2^m \in [0, 2^m - 1]$。根据引理 V.11，鉴于 $m_2 = \left\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \right\rceil$，，我们可以用 $Rep_{q,m_1}\big(\varphi(\boldsymbol{y}_{[i,i+m]}) - 2^m\big)$编码 $\boldsymbol{y}_{[i,i+m]}$的信息。在步骤 5 中，我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,i]} \circ \boldsymbol{y}_{[i+m+1,n]} \circ Rep_{q,m_2}\big(\varphi(\boldsymbol{y}_{[i,i+m]}) - 2^m\big) \circ Rep_{q,m_1}(i)$。最后，标记由偶符号 $h_2(y_{n-1}) \notin \{y_{n-1}, y_{n-1}\}$定义，并且在步骤 6 中我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y} \circ h_2(y_{n-1})$。由于选定游程的长度和原字符串的长度都减少了 m，因此在步骤 7 中我们更新 $j \leftarrow j - m$，在步骤 8 中我们更新 $i_{end} \leftarrow i_{end} - m$。当选定游程的长度缩短至不超过 m时，我们退出内层循环（步骤 4–9），定位

下一个长度超过 m的游程，并重复此过程，直到不存在这样的游程。编码器完成时，字符串 $\boldsymbol{y} = ENC_m^{RLL}(\boldsymbol{x}) \in \Sigma_q^n$ 的长度为 n，且对于 $i \leqslant i_{end}$，子串 $\boldsymbol{y}_{[i,\ i+m]}$ 不是一个游程。此外，对于 $i > i_{end}$，由于子串 $\boldsymbol{y}_{[i,\ i+m]}$ 包含一个标记符号（ $h_1(\cdot)$ 或 $h_2(\cdot)$）以及其左侧相邻的符号，根据引理 V.13，我们可以得出 $\boldsymbol{y}_{[i,\ i+m]}$ 仍然不是一个游程。因此，最终输出是一个 m-RLL 字符串。由 y_n 的奇偶性可知，算法 4 中的循环（步骤 4 至 9）是否执行。由于算法 5 只是编码器的逆，我们有 $DEC_m^{RLL}\big(ENC_m^{RLL}(\boldsymbol{x})\big) = \boldsymbol{x}$。

接下来，我们分析这些算法的时间复杂度。假设有 r 个长度超过 m 的游程：$\boldsymbol{x}_{[i_1,\ j_1-1]}$、$\boldsymbol{x}_{[i_2,\ j_2-1]}$，……，$\boldsymbol{x}_{[i_r,\ j_r-1]}$。首先，我们可以在 $O(n)$ 时间内定位这些游程。对于每个 $s \in [1,\ r]$，算法 4 的内层循环（步骤 4–9）会将游程 $\boldsymbol{x}_{[i_s,\ j_s-1]}$ 转换为一个长度至多为 m 的新游程。每次执行都会使所选游程的长度缩短 m，因此针对第 s 个游程的循环会执行 $\lfloor \frac{j_s - i_s}{m} \rfloor$ 次。由于这些游程互不相交，该内层循环总共执行

$$\sum_{s=1}^{r} \left\lfloor \frac{j_s - i_s}{m} \right\rfloor \leqslant \frac{1}{m} \sum_{s=1}^{r} (j_s - i_s) \leqslant \frac{n}{m}$$

次。此外，在每次迭代中，我们都会计算函数 $Rep_{q,\ m_2}\big(\varphi(\cdot) - 2^m\big)$ 和 $Rep_{q,\ m_1}(\cdot)$，这可以在 $O(m)$ 时间内完成。因此，算法 4 的总复杂度为 $O(n) + O(\frac{n}{m}) \cdot O(m) = O(n)$。由于算法 5 只是编码器的逆，其复杂度也为 $O(n)$。证明完毕。 □

现在我们给出一个示例，以说明我们的编码和解码算法。

示例 V.17。设 $n = 20$ 和 $q = 4$，其中 $0 = 1$ 和 $2 = 3$。则 $m_1 = \lceil \log_q n \rceil = 3$、$m_2 = \big\lceil \lceil \log_q n + 1 \rceil \cdot \log_{q/2} 2 \big\rceil = 4$，以及 $m = m_1 + m_2 + 1 = 8$。假设 $\boldsymbol{x} = 01222222222333333333 \in \Sigma_q^{n-1}$。我们应用算法 4 将其编码为 m-RLL 字符串。

• 首先，我们计算 $h_1(x_{n-1}) = 2 \cdot \lfloor \frac{x_{n-1}}{2} \rfloor + 3$ (模 q) $= 2 \cdot \lfloor \frac{3}{2} \rfloor + 3$ (模 4) $= 1$。然后令 $\boldsymbol{y} = \boldsymbol{x} \circ h_1(x_{n-1}) =$*01222222222333333331*、$i = 1$和 $i_{end} = n - 1 - m = 20 - 1 - 8 = 11$。现在，我们进入循环（步骤 *2–10*）。• 由于 $i < i_{end}$，我们令 $j = 3$，它表示满足 $y_j \notin \{y_i,\ y_i\} = \{0,\ 1\}$的最小索引。现在，$i + m + 1 =$1 + 8 + 1 = 10 > j。在步骤 *10* 中，我们更新 $i \leftarrow j$，即重置 $i = 3$。• 由于 $i < i_{end}$，我们重置 $j = 20$。现在，$i + m + 1 = 3 + 8 + 1 = 12 \leqslant j$ 并进入内层循环（步骤 *4–9*）。– 考虑子串 $\boldsymbol{y}_{[i,\ i+m]} =$ 222222222，我们计算 $\mathbb{1}(\boldsymbol{y}_{[i,\ i+m]}) =$ 111111111、$\varphi(\boldsymbol{y}_{[i,\ i+m]}) - 2^m = 127 - 64 = 63$、$Rep_{q\ m_2}\big(\varphi(\boldsymbol{y}_{[i\ i+m]}) - 2^m\big) = Rep_{4\ 4}(63) =$ 0333 $Rep_{q\ m_1}(i) = Rep_{4\ 3}(3) =$ 003 *5,* 、、、、、以及、、。在步骤 中，我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,\ i]} \circ \boldsymbol{y}_{[i+m+1,\ n]} \circ Rep_{q,\ m_2}\big(\varphi(\boldsymbol{y}_{[i,\ i+m]}) - 2^m\big) \circ Rep_{q,\ m_1}(i)$，即重置 $\boldsymbol{y} =$ 0123333333310333003。然后我们计算 $h_2(y_{n-1}) = 2 \cdot \lfloor \frac{y_{n-1}}{2} \rfloor + 2$ (模 q) $= 2 \cdot \lfloor \frac{3}{2} \rfloor + 2$ (模 4) $= 0$。在步骤 *6* 中，我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y} \circ h_2(y_{n-1})$，即重置 $\boldsymbol{y} =$ 01233333333103330030。在步骤 *7* 中，我们更新 $j \leftarrow j - m$，即重置 $j = 20 - 8 = 12$。在步骤 *8* 中，我们更新 $i_{end} \leftarrow i_{end} - m$，即重置 $i_{end} = 11 - 8 = 3$。– 由于 $i + m + 1 = 3 + 8 + 1 = 12 \leqslant j$ $\boldsymbol{y}_{[i\ i+m]} =$ 233333333 $\mathbb{1}(\boldsymbol{y}_{[i\ i+m]}) =$，我们考虑子串、。我们计算 100000000、、$\varphi(\boldsymbol{y}_{[i\ i+m]}) - 2^m = 64 - 64 = 0$ $Rep_{q\ m_2}\big(\varphi(\boldsymbol{y}_{[i\ i+m]}) - 2^m\big) = Rep_{4\ 4}(0) =$ 0000 $Rep_{q\ m_1}(i) =$、、、、、、$Rep_{4\ 3}(3) =$ 003 *5,* $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1\ i]} \circ \boldsymbol{y}_{[i+m+1\ n]} \circ Rep_{q\ m_2}\big(\varphi(\boldsymbol{y}_{[i\ i+m]}) - 2^m\big) \circ Rep_{q\ m_1}(i)$、以及、、。在步骤 中，我们更新、、、、、、，即重置 $\boldsymbol{y} =$ 0121033300300000003。然后我们计算 $h_2(y_{n-1}) = 2 \cdot \lfloor \frac{y_{n-1}}{2} \rfloor + 2$（模 q) $= 2 \cdot \lfloor \frac{3}{2} \rfloor + 2$ (模 4) $= 0$。在步骤 *6* 中，我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y} \circ h_2(y_{n-1})$，即重置 $\boldsymbol{y} =$ 01210333003000000030。在步骤 *7* 中，我们更新 $j \leftarrow j - m$，即重置 $j = 12 - 8 = 4$。在步骤 *8* 中，我们更新 $i_{end} \leftarrow i_{end} - m$，即重置 $i_{end} = 3 - 8 = -5$。– 由于 $i + m + 1 = 3 + 8 + 1 = 12 > j$，我们退出内层循环（步骤 *4–9*）。

• 由于 $i > i_{end}$，我们退出循环（步骤 *2–10*），算法输出 $\boldsymbol{y} =$ 01210333003000000030。现在，我们使用算法 *5* 从 $\boldsymbol{y} =$ 01210333003000000030中解码 $\boldsymbol{x}$。• 由于 $y_n = 0$ $i = Rep_{q\ m_1}^{-1}(\boldsymbol{y}_{[n-m_1\ n-1]}) = Rep_{4\ 3}^{-1}(\boldsymbol{y}_{[17\ 19]}) = Rep_{4\ 3}^{-1}(003) = 3$ $a = Rep_{4\ 4}^{-1}(\boldsymbol{y}_{[13\ 16]}) +$ 是偶数，我们计算 ,,,,,,,,, $2^4 = Rep_{4\ 4}^{-1}(0000) + 2^8 = 0 + 64 = 64$ $\varphi^{-1}(a\ y_i) = \varphi^{-1}(64\ 2) =$ 233333333 *4,* $\boldsymbol{y} \leftarrow$ 和，，。在步骤中，我们更新 $\boldsymbol{y}_{[1\ i-1]} \circ \varphi^{-1}(a\ y_i) \circ \boldsymbol{y}_{[i+1\ n-m]}$ $\boldsymbol{y} =$ 01233333333103330030 ,,, 即重置 。• 由于 $y_n = 0$ $i = Rep_{q\ m_1}^{-1}(\boldsymbol{y}_{[n-m_1\ n-1]}) = Rep_{4\ 3}^{-1}(\boldsymbol{y}_{[17\ 19]}) = Rep_{4\ 3}^{-1}(003) = 3$ $a = Rep_{4\ 4}^{-1}(\boldsymbol{y}_{[13\ 16]}) +$ 是偶数，我们计算 ,,,,,,,,, $2^4 = Rep_{4\ 4}^{-1}(0333) + 2^8 = 63 + 64 = 127$ $\varphi^{-1}(a\ y_i) = \varphi^{-1}(127\ 2) =$ 222222222 *4,* 和，，。在步骤中，我们更新 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{[1,i-1]} \circ \varphi^{-1}(a, y_i) \circ \boldsymbol{y}_{[i+1,n-m]}$，即重置 $\boldsymbol{y} =$ 01222222222333333331。• 由于 $y_n = 1$ 是奇数，我们退出循环，算法输出 $\boldsymbol{x} = \boldsymbol{y}_{[1,n-1]} =$ 01222222222333333333。

B. 码构造

在本小节中，我们给出两种能够纠正 t 长度为一的反向互补重复的码构造。第一种构造的冗余为 $2t\log_q n + O(\log_q \log_q n)$，编码复杂度为 $O(n)$，解码复杂度为 $O\big(n(\log_2 n)^4\big)$。第二种构造将冗余改进为 $(2t-1)\log_q n + O(\log_q \log_q n)$，但编码和解码复杂度为 $O\big(n \cdot \mathrm{poly}(\log_2 n)\big)$。这两种构造都结合了来自 [22] 的插入删除纠错码和来自 [25] 的替换纠错码。

引理 V. **18** *[22, 2]* t $\boldsymbol{\eta}_{n\,q\,t}$ $\Sigma_q^n \to$. 推论 设 为一个固定 $\Sigma_q^{4t\log_q n+o(\ \ _q n)}$ (n) $\boldsymbol{\eta}_{n\,q\,t}(\boldsymbol{c})$ $\boldsymbol{c}'$ 的正整数。存在一个显式哈希函数 ,,: log，可在多项式时间内计算，即 poly时间，使得给定 ,,以及一个由 $\boldsymbol{c}\in\Sigma_q^n$ 经过 t 插入和删除得到的任意字符串，都可以在多项式时间内解码 $\boldsymbol{c}$。当长度 n 和字母表大小 q 由上下文明确时，我们将 $\boldsymbol{\eta}_{n,q,t}$ 简写为 $\boldsymbol{\eta}_t$，以突出纠错能力。

引理 V.19. *[25, 第 III 节和第 IV 节]* 设 t 为一个固定整数，ℓ 为满足 $\ell\geqslant\max\{n, 2t(q-1)+1\}$ 的最小素数。[4] 那么存在一个显式哈希函数 $\boldsymbol{\zeta}_{n,q,t}\colon \Sigma_q^n \to \Sigma_q^{\lceil\log_q(2t(q-1)+1)\ell^{2t-1}\rceil}$，可在 $O(n)$ 时间内计算，使得给定 $\boldsymbol{\zeta}_{n,q,t}(\boldsymbol{c})$ 以及任意一个由 $\boldsymbol{c}\in\Sigma_q^n$ 经过至多 t 次替换得到的字符串 $\boldsymbol{c}'$，可以在 $O\big(n(\log_2 n)^4\big)$ 时间内解码出 $\boldsymbol{c}$。当长度 n 和字母表大小 q 在上下文中明确时，我们将 $\boldsymbol{\zeta}_{n,q,t}$ 简写为 $\boldsymbol{\zeta}_t$，以突出纠错能力。

1) 第一种构造：在第一种构造中，我们只需使用函数 $\varphi(\cdot)$（定义 V.14）记录每个游程，并将游程内部发生的重复视为一次替换。我们首先介绍所需的定义。

定义 V.20。设 $\varphi(\cdot)$ 在定义 *V.14* 中定义。对于任意字符串 $\boldsymbol{x}\in\Sigma_q^n$，我们定义其关联向量 $\boldsymbol{a}$ s

$$\boldsymbol{\phi}(\boldsymbol{x}) = \big(\phi(\boldsymbol{x})_1, \phi(\boldsymbol{x})_2, \ldots, \phi(\boldsymbol{x})_{r(\boldsymbol{x})}\big),$$

其中 $\phi(\boldsymbol{x})_i = \varphi\big(\boldsymbol{\sigma}(\boldsymbol{x}、\ i)\big)$ 和 $\boldsymbol{\sigma}(\boldsymbol{x}、\ i)$ 分别表示在 $i\in[1、\ r(\boldsymbol{x})]$条件下，$\boldsymbol{x}$ 的第 i 个游程。

引理 V.21。对于任意字符串 $\boldsymbol{x}\in\Sigma_q^n$，给定其特征 $\boldsymbol{\sigma}(\boldsymbol{x})$，由下式定义的 $\boldsymbol{\phi}$ 的逆映射

$$\boldsymbol{\phi}^{-1}\big(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})\big) = \big(\varphi^{-1}(\phi(\boldsymbol{x})_1, \sigma(\boldsymbol{x})_1), \varphi^{-1}(\phi(\boldsymbol{x})_2, \sigma(\boldsymbol{x})_2), \ldots, \varphi^{-1}(\phi(\boldsymbol{x})_{r(\boldsymbol{x})}, \sigma(\boldsymbol{x})_{r(\boldsymbol{x})})\big)$$

是良定义的，即 $\boldsymbol{\phi}^{-1}\big(\boldsymbol{\phi}(\boldsymbol{x})、\ \boldsymbol{\sigma}(\boldsymbol{x})\big) = \boldsymbol{x}$。此外，函数 $\boldsymbol{\phi}^{-1}(\cdot、\ \cdot)$ 可在 $O(n)$ 时间内计算。

证明。令 $n_i = |\boldsymbol{\sigma}(\boldsymbol{x}, i)|$，其中 $i\in[1, r(\boldsymbol{x})]$。于是 $\sum_{i=1}^{r(\boldsymbol{x})} n_i = n$。根据公式 **(1)**，我们可以在 $O(n_i)$ 时间内计算出 $\varphi^{-1}(\phi(\boldsymbol{x})_1, \sigma(\boldsymbol{x})_1) = \boldsymbol{\sigma}(\boldsymbol{x}, i)$。因此，我们有 $\boldsymbol{\phi}^{-1}\big(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})\big) = \boldsymbol{x}$，且函数 $\boldsymbol{\phi}^{-1}(\cdot, \cdot)$ 可在 $\sum_{i=1}^{r(\boldsymbol{x})} O(n_i) = O(n)$ 时间内计算完成。证明完毕。 □

现在我们给出第一个构造。令 $m = \lceil\log_q n\rceil + \big\lceil\lceil\log_q n + 1\rceil\cdot\log_{q/2} 2\big\rceil + 1$。对于任意 $\boldsymbol{x}\in\Sigma_q^{n-1}$，我们首先使用算法 4 m $\boldsymbol{x}' \triangleq ENC_m^{RLL}(\boldsymbol{x}) \in \Sigma_q^n$ $\boldsymbol{\phi}(\boldsymbol{x}') =$ 将其编码为一个 -RLL 字符串。 $\big(\phi(\boldsymbol{x}')_1\ \phi(\boldsymbol{x}')_2\ \ldots\ \phi(\boldsymbol{x}')_{r(\boldsymbol{x}')}\big)$ $\phi(\boldsymbol{x}')_i = \varphi\big(\boldsymbol{\sigma}(\boldsymbol{x}'\ i)\big) \in [2^{m-1}\ 2^m - 1]$ $i \in [1\ r(\boldsymbol{x}')]$ 然后，其关联向量 ,,,，其中 ,, 对于 ,, 可视为一个长度为 $r(\boldsymbol{x}')$ 、取自大小为 2^m 的字母表的字符串。由于 $r(\boldsymbol{x}') \leqslant n$ 取决于特定 $\boldsymbol{x}'$ 的选择，我们定义

$$\boldsymbol{\phi}'(\boldsymbol{x}') \triangleq \left(\underbrace{0, \ldots, 0}_{n-r(\boldsymbol{x}')}, \boldsymbol{\phi}(\boldsymbol{x}')\right) \in \Sigma_{2^m}^n$$

作为通过在 $\boldsymbol{\phi}(\boldsymbol{x}')$ 前置 $n - r(\boldsymbol{x}')$ 个 0 所得到的字符串。由于 $q\geqslant 4$，我们有 $2^m \leqslant 2^{2\log_4 n+4} = 16n$。

定理V.22。设 $t\geqslant 1$ 为一个固定整数。令 $m = \lceil\log_q n\rceil + \big\lceil\lceil\log_q n + 1\rceil\cdot\log_{q/2} 2\big\rceil + 1$。设 $\boldsymbol{R}_{t+1}(\cdot)$ 为 $(t+1)$ 重重复码的编码函数。对于任意 $\boldsymbol{x}\in\Sigma_q^{n-1}$，令 $\boldsymbol{x}' = ENC_m^{RLL}(\boldsymbol{x})$，我们定义

$$\mathcal{E}(\boldsymbol{x}) = \left(\boldsymbol{x}', \underbrace{\overline{x'_n}, \ldots, \overline{x'_n}}_{t}, \boldsymbol{\zeta}_t\big(\boldsymbol{\phi}'(\boldsymbol{x}')\big), \boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}')))\big)\right).$$

则该码

$$\mathcal{C} = \big\{\mathcal{E}(\boldsymbol{x}) : \boldsymbol{x}\in\Sigma_q^{n-1}\big\}$$

可以使用 $2t\log_q n + O(\log_q \log_q n)$ 个冗余符号纠正 t 次长度为一的反向互补重复。此外，编码复杂度为 $O(n)$，解码复杂度为 $O\big(n(\log_2 n)^4\big)$。

证明。对于任意序列 $\boldsymbol{x}\in\Sigma_2^{n-1}$ $n_1 \triangleq \big|\boldsymbol{\zeta}_t\big(\boldsymbol{\phi}'(\boldsymbol{x}')\big)\big| = 2t\ _q n + O(1)$ $n_2 \triangleq \big|\boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}')))\big)\big| =$，我们有 log 和 $O(\ \ _q\ _q n)$ $\mathcal{C}$ $n_1+n_2+t+1 = 2t\ _q n + O(\ \ _q\ _q n)$ $ENC_m^{RLL}(\cdot)$ log log。因此，的冗余为 log log log。注意到函数 、$\boldsymbol{\phi}'(\cdot)$、$\boldsymbol{\zeta}_t(\cdot)$、$\boldsymbol{\eta}_t(\cdot)$ 和 $\boldsymbol{R}_{t+1}(\cdot)$ 均可在 $O(n)$ 时间内计算，因此总体编码复杂度为 $O(n)$。

假设 $\mathcal{E}(\boldsymbol{x})$ 经历了 t次长度为一的反向互补重复，并在 $\boldsymbol{y}$中得到结果。 n

- $\boldsymbol{y}_{[1,\ n+t]}$ 是由 $\boldsymbol{x}'$ 经过 t 次反向互补重复得到的，正如 $i\in[n+1,\ n+t]$的 $\overline{\mathcal{E}(x)}_i = x'_n$ 所示。由引理 V.8 和 $\boldsymbol{\phi}'(\cdot)$ 的定义可得$r(\boldsymbol{y}_{[1,\ n+t]}) = r(\boldsymbol{x}')$、$\boldsymbol{\sigma}(\boldsymbol{y}_{[1,\ n+t]}) = \boldsymbol{\sigma}(\boldsymbol{x}')$，并且$\boldsymbol{\phi}'(\boldsymbol{y}_{[1,\ n+t]})$ 可由 $\boldsymbol{\phi}'(\boldsymbol{x}')$经过至多 t 次替换得到。这里，若$\boldsymbol{y}_{[1,\ n+t]}$ 中的某个游程长度超过 m，则可以断定该游程处发生了重复，并将$\boldsymbol{\phi}'(\boldsymbol{y}_{[1,\ n+t]})$ 中对应的项设为零。

[4]根据著名的伯特兰-切比雪夫定理，我们有 $\ell\leqslant 2\cdot\max\{n, 2t(q-1)+1\}$。

• $\boldsymbol{y}_{[n+t+1,n+2t+n']}$ 通过 t 次插入从 $\boldsymbol{\zeta}_t\big(\boldsymbol{\phi}'(\boldsymbol{x}')\big)$ 得到；• $\boldsymbol{y}_{[n+t+n_1+1,\ n+2t+n_1+n_2]}$ 通过 t 次插入从 $\boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}')))\big)$ 得到。

首先，由于 $\boldsymbol{R}_{t+1}(\cdot)$ 是 $(t+1)$ 重重复码的编码函数，我们可以在 $O(n_2)=o(n)$ 时间内从 $\boldsymbol{y}_{[n+t+n_1+1,\ n+2t+n_1+n_2]}$ 恢复 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}'))\big)$。接下来，根据引理 V.18，我们可以在 $\mathrm{poly}(n_1)=o(n)$ 时间内从 $\boldsymbol{y}_{[n+t+1,\ n+2t+n']}$ 和 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}'))\big)$ 恢复 $\boldsymbol{\zeta}_t\big(\boldsymbol{\phi}'(\boldsymbol{x}')\big)$。注意，我们可以在 $O(n)$ 时间内计算 $r(\boldsymbol{y}_{[1,\ n+t]})$、$\boldsymbol{\sigma}(\boldsymbol{y}_{[1,\ n+t]})$、$\boldsymbol{\phi}'(\boldsymbol{y}_{[1,\ n+t]})$。根据引理 V.19，我们可以在 $O\big(n(\log_2 n)^4\big)$ 时间内从 $\boldsymbol{\phi}'(\boldsymbol{y}_{[1,\ n+t]})$ 和 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{\phi}'(\boldsymbol{x}'))\big)$ 恢复 $\boldsymbol{\phi}'(\boldsymbol{x}')$。然后，我们可以在 $O(n)$ 时间内从 $\boldsymbol{\phi}'(\boldsymbol{x}')$ 和 $r(\boldsymbol{y}_{[1,\ n+t]})$ 恢复$\boldsymbol{\phi}(\boldsymbol{x}')$。因此，根据引理 V.21，我们可以在 $O(n)$ 时间内使用 $\boldsymbol{\phi}(\cdot)$ 的逆来确定 $\boldsymbol{x}'=\boldsymbol{\phi}^{-1}\big(\boldsymbol{\phi}(\boldsymbol{x}'),\ \boldsymbol{\sigma}(\boldsymbol{x}')\big)$。最后，我们使用算法 5 在 $O(n)$ 时间内从 $\boldsymbol{x}'$ 解码 $\boldsymbol{x}$。因此，$\mathcal{C}$ 可以纠正 t 个长度为一的反向互补重复，且总解码复杂度为 $O\big(n(\log_2 n)^4\big)$。证明完毕。 □

2) 第二种构造：在第二种构造中，我们不再记录整个游程，而只存储该游程的哈希值，以辅助纠正其中的重复。这将字母表大小从 $O(n)$ 降至 $O(\log_2 n)$。因此，当使用引理 V.19 中的替换纠错码来纠正 t 次替换时，冗余变为 $(2t-1)\log_q n+O(\log_q\log_q n)$。接下来，我们将详细说明该方法。

令 $m=\lceil\log_q n\rceil+\big\lceil\lceil\log_q n+1\rceil\cdot\log_{q/2}2\big\rceil+1$。对于任意 $\boldsymbol{x}\in\Sigma_q^{n-1}$，我们首先使用算法 4 将其编码为 m-RLL 字符串 $\boldsymbol{x}'\triangleq ENC_m^{RLL}(\boldsymbol{x})\in\Sigma_q^n$。回想一下，$\boldsymbol{\sigma}(\boldsymbol{x}',i)$ 表示 $\boldsymbol{x}'$ 的第 i 个游程，而 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',i)\big)$ 表示$\boldsymbol{\sigma}(\boldsymbol{x}',i)$的指示向量；我们首先在 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',i)\big)$ 前面添加足够多的零，使其长度恰好增至 $m+t$，然后使用哈希函数$\boldsymbol{\eta}_{m+t,2,2t}(\cdot)$（定义于引理 V.18）对其进行编码，最后记录该哈希值的整数表示，记为$f\big(\boldsymbol{\sigma}(\boldsymbol{x}',i)\big)\in\Sigma_{2^{8t\log_2 m+o(\log_2 m)}}$。注意，$2^{8t\log_2 m+o(\log^2 m)}=m^{8t+o(1)}$。

引理 V.23。令 $f(\cdot)$ 如上定义。给定 $f\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$ 以及任意字符串 $\boldsymbol{z}$，该字符串由 $\boldsymbol{\sigma}(\boldsymbol{x}'$经过至多 t次长度为一的反向互补重复得到，可以在 $O\big($多项式$(m)\big)$ 时间内恢复 $\boldsymbol{\sigma}(\boldsymbol{x}',\ i)$。

证明。令 $\boldsymbol{c}$ 由 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$ 在前面添加 $m+t-\big|\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)\big|$ 个零得到，令 $\boldsymbol{c}'$ 由 $\mathbb{1}(\boldsymbol{z})$ 在前面添加 $m+t-|\mathbb{1}(\boldsymbol{z})|$ 个零得到。由于 $\boldsymbol{z}$ 由 $\boldsymbol{\sigma}(\boldsymbol{x}',\ i)$ 经过 $t'\leqslant t$ 次长度为一的反向互补重复得到，我们可以断定，$\boldsymbol{c}'$ 由 $\boldsymbol{c}$ 先经过 t' 次插入、再在最左侧删除 t' 个零得到。因此，我们可以通过 $2t$ 次插入和删除，从 $\boldsymbol{c}$得到 $\boldsymbol{c}'$。给定 $f\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$，即 $\boldsymbol{\eta}_{m+t,\ 2,\ 2t}(\boldsymbol{c})$的整数表示，我们可以在 $O(m)$ 时间内计算 $\boldsymbol{\eta}_{m+t,\ 2,\ 2t}(\boldsymbol{c})$。然后根据引理 V.18，我们可以在 $O\big($多项式$(m)\big)$ 时间内从 $\boldsymbol{c}'$和 $\boldsymbol{\eta}_{m+t,\ 2,\ 2t}(\boldsymbol{c})$ 解码出 $\boldsymbol{c}$。根据定义 V.14，$\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$ 的第一项为 1。因此，从 $\boldsymbol{c}$中删除最左侧的所有零，即可得到正确的 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$。由于重复不会改变 $\boldsymbol{\sigma}(\boldsymbol{x}',\ i)$的第一项，因此有 $\sigma(\boldsymbol{x}',\ i)_1=z_1$。最后，根据定义 V.14，我们可以在 $O(m)$ 时间内从 $\mathbb{1}\big(\boldsymbol{\sigma}(\boldsymbol{x}',\ i)\big)$ 和 z_1 恢复 $\boldsymbol{\sigma}(\boldsymbol{x}',\ i)$。证明完毕。 □

现在，我们定义向量

$$\boldsymbol{F}(\boldsymbol{x}')\triangleq\left(\underbrace{0,\ldots,0}_{n-r(\boldsymbol{x}')},f\big(\boldsymbol{\sigma}(\boldsymbol{x}',1)\big),\ldots,f\big(\boldsymbol{\sigma}(\boldsymbol{x}',r(\boldsymbol{x}'))\big)\right)\in\Sigma_{m^{8t+o(1)}}^n.$$

定理 V.24. 设 t 为一个固定的正整数。令 $m=\lceil\log_q n\rceil+\big\lceil\lceil\log_q n+1\rceil\cdot\log_{q/2}2\big\rceil+1$。设 $\boldsymbol{R}_{t+1}(\cdot)$ 为 $(t+1)$重重复码的编码函数。对任意 $\boldsymbol{x}\in\Sigma_q^n$，令 $\boldsymbol{x}'=ENC_m^{RLL}(\boldsymbol{x})$，我们定义

$$\mathcal{E}(\boldsymbol{x})=\left(\boldsymbol{x}',\underbrace{\overline{x'_n},\ldots,\overline{x'_n}}_{t},\boldsymbol{\zeta}_t\big(\boldsymbol{F}(\boldsymbol{x}')\big),\boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}')))\big)\right).$$

则该码

$$\mathcal{C}=\big\{\mathcal{E}(\boldsymbol{x}):\boldsymbol{x}\in\Sigma_q^{n-1}\big\}$$

能够纠正 t 个长度为一的反向互补复制，且使用 $(2t-1)\log_q n+O(\log_q\log_q n)$ 个冗余符号。此外，编码和解码复杂度为 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$。

证明。对任意序列 $\boldsymbol{x}\in\Sigma_2^{n-1}$，我们有$n_1\triangleq\big|\boldsymbol{\zeta}_t\big(\boldsymbol{F}(\boldsymbol{x}')\big)\big|=(2t-1)\log_q n+O(\log_q\log_q n)$ 以及 $n_2\triangleq\big|\boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}')))\big)\big|=O(\log_q\log_q n)$。因此，$\mathcal{C}$的冗余为 $n_1+n_2+t+1=(2t-1)\log_q n+O(\log_q\log_q n)$。注意到函数 $f(\cdot)$可在 $O\big($poly$(m)\big)=O\big(\mathrm{poly}(\log_2 n)\big)$ 时间内计算，我们可以在 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$ 时间内计算函数 $\boldsymbol{F}(\cdot)$。此外，由于函数 $ENC_m^{RLL}(\cdot)$、$\boldsymbol{\zeta}_t(\cdot)$、$\boldsymbol{\eta}_t(\cdot)$和 $\boldsymbol{R}_{t+1}(\cdot)$可在 $O(n)$时间内计算，总体编码复杂度为 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$。

假设 $\mathcal{E}(\boldsymbol{x})$遭受 t长度为一的反向互补重复，并在 $\boldsymbol{y}$中产生结果。该 n

• $\boldsymbol{y}_{[1,\ n+t]}$ 通过 t 次反向互补重复，由 $\boldsymbol{x}'$ 得到；如 $\mathcal{E}(x)_i=x'_n$ 对于 $i\in[\overline{n+1},\ n+t]$。根据引理 V.8 和 $\boldsymbol{F}(\cdot)$ 的定义，我们有 $r(\boldsymbol{y}_{[1,\ n+t]})=r(\boldsymbol{x}')$、$\boldsymbol{\sigma}(\boldsymbol{y}_{[1,\ n+t]})=\boldsymbol{\sigma}(\boldsymbol{x}')$，并且 $\boldsymbol{F}(\boldsymbol{y}_{[1,\ n+t]})$ 通过至多 t 次替换，由 $\boldsymbol{F}(\boldsymbol{x}')$ 得到。

• $\boldsymbol{y}_{[n+t+1,\ n+2t+n']}$ 通过 t 次插入，由 $\boldsymbol{\zeta}_t\big(\boldsymbol{F}(\boldsymbol{x}')\big)$ 得到；

- $\boldsymbol{y}_{[n+t+n_1+1,\ n+2t+n_1+n_2]}$ 通过 t 次插入，由 $\boldsymbol{R}_{t+1}\big(\boldsymbol{\eta}_t(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}')))\big)$ 得到。

首先，由于 $\boldsymbol{R}_{t+1}(\cdot)$是 $(t+1)$重复码的编码函数，我们可以在 $O(n_2)=o(n)$ 时间内根据 $\boldsymbol{y}_{[n+t+n_1+1、\ n+2t+n_1+n_2]}$ 恢复 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}'))\big)$。接下来，根据引理 V.18，我们可以在 $\mathrm{poly}(n_1)=o(n)$ 时间内根据 $\boldsymbol{y}_{[n+t+1、\ n+2t+n']}$ 和 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}'))\big)$ 恢复 $\boldsymbol{\zeta}_t\big(\boldsymbol{F}(\boldsymbol{x}')\big)$。注意，我们可以在 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$ 时间内计算 $r(\boldsymbol{y}_{[1、\ n+t]})$、$\boldsymbol{\sigma}(\boldsymbol{y}_{[1、\ n+t]})$、$\boldsymbol{F}(\boldsymbol{y}_{[1、\ n+t]})$。根据引理 V.19，我们可以在 $O\big(n(\log_2 n)^4\big)$ 时间内根据 $\boldsymbol{F}(\boldsymbol{y}_{[1、\ n+t]})$ 和 $\boldsymbol{\eta}_t\big(\boldsymbol{\zeta}_t(\boldsymbol{F}(\boldsymbol{x}'))\big)$ 恢复 $\boldsymbol{F}(\boldsymbol{x}')$。然后，我们可以在 $O(n)$ 时间内根据 $\boldsymbol{F}(\boldsymbol{x}')$、$r(\boldsymbol{y}_{[1}$和 $_{n+t]})$ 恢复 $f\big(\boldsymbol{\sigma}(\boldsymbol{x}'、\ i)\big)$、$\ldots$、$f\big(\boldsymbol{\sigma}(\boldsymbol{x}'、\ r(\boldsymbol{x}'))\big)$。此外，对于 $i\in[1、\ r(\boldsymbol{x}')]$，根据引理 V.23，我们可以在 $O\big(\mathrm{poly}(\log_2 n)\big)$ 时间内根据 $\boldsymbol{\sigma}(\boldsymbol{y}_{[1、\ n+t]}、\ i)$ 和 $f\big(\boldsymbol{\sigma}(\boldsymbol{x}'、\ i)\big)$ 解码 $\boldsymbol{\sigma}(\boldsymbol{x}'、\ i)$。由此得到正确的 $\boldsymbol{x}'$，即 $\boldsymbol{x}'=\big(\boldsymbol{\sigma}(\boldsymbol{x}'、\ 1)、\ldots、\boldsymbol{\sigma}(\boldsymbol{x}'、\ r(\boldsymbol{x}'))\big)$。最后，我们使用算法 5 在 $O(n)$ 时间内根据 $\boldsymbol{x}'$ 解码 $\boldsymbol{x}$。因此，$\mathcal{C}$可以纠正 t 个长度为一的反向互补重复，且总体解码复杂度为 $O\big(n\cdot\mathrm{poly}(\log_2 n)\big)$。证明完毕。 □

注 V.25. 定理 *V.22* 和 *V.24* 中提出的构造优于直接使用插入删除纠错码的方法 *[22], [29], [30], [31], [38], ,* 后者需要$5\log_q n+O(\log_q\log_q n)$ 和 $(4t-1)\log_q n+O(\log_q\log_q n)$ 个冗余符号，分别对应 $t=2$ 和 $t\geqslant 3$。

VI. 结论

本文研究了用于纠正反向互补重复或回文重复的码。我们构造了一种仅含一个冗余符号的显式码，在所有重复长度为 $k\geqslant 3\lceil\log_q n\rceil$ 且彼此不重叠的条件下，该码能够纠正任意数量的反向互补重复（或回文重复）。此外，对于 $q\geqslant 4$，我们提出了两种显式码构造，能够纠正 t 个长度为一的反向互补重复，分别使用$2t\log_q n+O(\log_q\log_q n)$和 $(2t-1)\log_q n+O(\log_q\log_q n)$ 个冗余符号。在纠正反向互补重复或回文重复方面，这些码优于通用突发插入纠错码。对于其他参数范围，能够纠正回文重复或反向互补重复的码是否能够比直接使用通用突发插入纠错码显著降低冗余，仍有待研究。

参考文献

[1] N. Alon 和 J. H. Spencer，《概率方法》，第 4 版，Wiley 离散数学与优化丛书。Wiley，2016 年。[2] M. Arita 和 Y. Ohashi，“基因组 DNA 中的秘密特征”，《生物技术进展》，第 20 卷，第 5 期，第 1605–7 页，2004 年。[3] E. Ben-Tolila 和 M. Schwartz，“论反向互补串重复系统”，IEEE信息论汇刊，第 68 卷，第 11 期，第 7184–7197 页，2022 年。[4] R. Bitar、S. K. Hanna、N. Polyanskii 和 I. Vorobyev，“局部删除纠错的最优码”，载于 2021 年 IEEE 信息论国际研讨会（ISIT），澳大利亚墨尔本，2021 年，第 1991–1996 页。[5] G. M. Church、Y. Gao 和 S. Kosuri，“DNA 中的新一代数字信息存储”，《科学》，第 337 卷，第 6102 期，第 1628–1628 页，2012 年。[6] O. Elishco，“突变系统中 k元组频率的长期行为”，IEEE信息论汇刊，第 70 卷，第 12 期，第 8524–8545 页，2024 年。[7] O. Elishco、F. Farnoud、M. Schwartz 和 J. Bruck，“某些 P´olya 字符串模型的熵率”，IEEE信息论汇刊，第 65 卷，第 12 期，第 8180–8193 页，2019 年。[8] F. Farnoud、M. Schwartz 和 J. Bruck，“串重复系统的容量”，IEEE信息论汇刊，第 62 卷，第 2 期，第 811–824 页，2016 年。[9] N. Goldman、P. Bertone、S. Chen、C. Dessimoz、E. M. LeProust、B. Sipos 和 E. Birney，“朝着在合成 DNA 中实现实用、高容量、低维护的信息存储迈进”，《自然》，第 494 卷，第 77–80 页，2013 年。[10] D. Goshkoder、N. Polyanskii 和 I. Vorobyev，“长重复错误纠错码”，《IEEE 分子、生物与多尺度通信汇刊》，第 10 卷，第 2 期，第 272–288 页，2024 年。[11] V. Guruswami 和 J. H˚astad，“冗余度达到存在性界的显式双删除码”，IEEE信息论汇刊，第 67 卷，第 10 期，第 6384–6394 页，2021 年。[12] D. Heider 和 B. Angelika，“使用 DNA-Crypt 算法的基于 DNA 的水印”，《BMC 生物信息学》，第 8 卷，第 176 页，2007 年。[13] S. Jain、F. Farnoud 和 J. Bruck，“基因组串联重复的容量与表达能力”，IEEE信息论汇刊，第 63 卷，第 10 期，第 6129–6138 页，2017 年。[14] S. Jain、F. Farnoud、M. Schwartz 和 J. Bruck，“用于生物体 DNA 数据存储的重复纠错码”，IEEE信息论汇刊，第 63 卷，第 8 期，第 4996–5010 页，2017 年。[15] A. Kobovich、O. Leitersdorf、D. Bar-Lev 和 E. Yaakobi，“参数化约束编码的通用框架”，载于 2024 年 IEEE 国际信息论研讨会（ISIT），希腊雅典，2024 年，第 1023–1028 页。[16] M. Kova˘cevi´c，“重复信道的零错误容量”，《IEEE 通信汇刊》，第 67 卷，第 10 期，第 6735–6742 页，2019 年。[17] ——，“短串联重复演化下不可混淆字符串的最大数量”，《信息传输问题》，第 58 卷，第 2 期，第 111–121 页，2022 年。[18] M. Kova˘cevi´c 和 V. Y. F. Tan，“DNA 存储系统中纠正定长重复错误的渐近最优码”，《IEEE 通信快报》，第 22 卷，第 11 期，第 2194–2197 页，2018 年。[19] A. Lenz、A. Wachter-Zeh 和 E. Yaakobi，“重复纠错码”，《设计、码与密码学》，第 87 卷，第 2–3 期，第 277–298 页，2019 年。[20] V. Levenshtein，“能够纠正删除、插入和反转的二进制码”，《苏联物理学文献》，第 10 卷，第 8 期，第 707–710 页，1966 年。[21] ——，“能够纠正一个或两个相邻比特丢失的渐近最优二进制码”，《系统理论研究》，第 19 卷，304，1970 4，298第。页 – 。[22] Y. Li 和 F. Farnoud，“Linial 算法与系统性删除纠错码”，载于 2023 年 IEEE 国际信息论研讨会（ISIT），台湾台北，2023 年，第 2703–2707 页。[23] M. Liss、D. Daubert、K. Brunner、K. Kliche、U. Hammes、A. Leiherer 和 R. Wagner，“在合成基因中嵌入永久水印”，《公共科学图书馆·综合》，第 7 卷，第 8 期，第 e42465 页，2012 年。[24] H. Liu、C. Tang、C. Fan 和 V. Sagar，“显式构造可纠正任意长度单次反向互补重复的码”，《通信数学进展》，提前在线发表，2025 年。 [在线]。网址：
https://www.aimsciences.org/article/id/693fd867f7c950317f042e11

[25] S. Liu 和 C. Xing，“低冗余非线性码”，载于*2024*年*IEEE*国际信息论研讨会（*ISIT*），希腊雅典，2024 年，第 705–710 页。[26] C. Schoeny、A. Wachter-Zeh、R. Gabrys 和 E. Yaakobi，“突发删除或插入纠错码”，*IEEE*信息论汇刊，第 63 卷，第 4 期，第 1971–1985 页，2017 年。[27] S. L. Shipman、J. Nivala、J. D. Macklis 和 G. M. Church，“利用 CRISPR-Cas 将数字电影编码进活细菌种群的基因组”，《自然》，第 547 卷，第 345–349 页，2017 年。[28] J. Sima 、 R. Gabrys 和 J. Bruck，“ 最优系统性 t 删除纠错码 ”，载于 *2020* 年 *IEEE* 国际信息论研讨会（*ISIT*），美国加利福尼亚州洛杉矶，2020 年，第 769–774 页。[29] ——，“最优冗余码的伴随式压缩”，载于*2020*年*IEEE*国际信息论研讨会（*ISIT*），美国加利福尼亚州洛杉矶，2020 年，第 751–756 页。[30] W. Song 和 K. Cai，“非二元双删除纠错码与突发删除纠错码”，*IEEE*信息论汇刊，第 69 卷，第 10 期，第 6470–6484 页，2023 年。[31] W. Song、N. Polyanskii、K. Cai 和 X. He，“纠正多重删除与多重替换错误的系统码”，*IEEE*信息论汇刊，第 68 卷，第 10 期，第 6402–6416 页，2022 年。[32] Y. Sun 和 G. Ge，“纠正两次编辑的二进制码”，*IEEE*信息论汇刊，第 70 卷，第 10 期，第 6877–6898 页，2024 年。[33] Y. Sun、Z. Lu、Y. Zhang 和 G. Ge，“用于纠正 $(t,\ s)$突发错误的渐近最优码”，*IEEE*信息论汇刊，第 71 卷，第 3 期，第 1570–1584 页，2025 年。[34] Y. Tang 和 F. Farnoud，“用于短串联重复和编辑错误的纠错码”，*IEEE*信息论汇刊，第 68 卷，第 2 期，第 871–880 页，2022 年。[35] Y. Tang、S. Wang、H. Lou、R. Gabrys 和 F. Farnoud，“用于纠正多个短重复和编辑错误的低冗余码”，*IEEE*信息论汇刊，第 69 卷，第 5 期，第 2940–2954 页，2023 年。[36] Y. Tang、Y. Yehezkeally、M. Schwartz 和 F. Farnoud，“重复信道与替换信道的单错误检测与纠正”，*IEEE*信息论汇刊，第 66 卷，第 11 期，第 6908–6919 页，2020 年。[37] P. C. Wong、K.-k. Wong 和 H. Foote，“采用 DNA 方法的有机数据存储器”，《美国计算机协会通讯》，第 46 卷，第 1 期，第 95–98 页，2003 年。[38] Z. Ye、Y. Sun、W. Yu、G. Ge 和 O. Elishco，“纠正两个恰含 b 次删除突发的码”，*IEEE*信息论汇刊，提前在线发表，

2025 [ ] 11153993 。在线。网址：

[39] https://ieeexplore.ieee.org/abstract/document/ L. Yohananov 和 M. Schwartz，“论反向互补与回文重复纠错码的编码容量”，《设计、码与密码学》，第 93 卷，第 8 期，第 3283–3302 页，2025 年。[40] W. Yu 和 M. Schwartz，“用于不相交或等长错误的无重复码”，《设计、码与密码学》，第 92 卷，2861，2024 10，2845 第。页 – 。