解题报告

张哲宇

November 19, 2019

1 Cutting the Line

1.1 题目大意

给字符串 S ，限制段数 k 。

要求将 S 划分成不超过 k 段子串，任意翻转其中任意个子串，再按顺序拼接。

最小化得到的字符串的字典序。

1.2 数据范围

$|S| \leq 5 \times 10^6$

1.3 解题过程

首先问题转换，将原问题转换成：给定字符串 S ，字符串 T 初始为空，进行不超过 k 次操作，每次截取 S^r 的一个后缀，然后选择将其翻转或不翻转后拼接至 T 后，使得 S^r 最后为空。求 T 的最小字典序。

下文均针对转换后的问题。

先解决当 $k = |S|$ 时的问题。

首先当 $k = |S|$ 时，一定存在一个最优方案全部选择不翻转。即先考虑没有翻转的情况。

引理 1：令 tmp 的字典序最小的后缀为 A ，则 A 在 tmp 中的出现位置不交。

证明：假设有交，则 A 有周期，而最小后缀没有周期，矛盾。 $\square$

定义对字符串 tmp 作最小后缀分解的为：令 A 为其最小后缀，将其表示为 $B_1 + c_1 \times A + B_2 + c_2 \times A + ... + c_m \times A$ 的形式。

B 为字符串数组，B_1 可能为空，$\forall i > 1 \quad \text{len}(B_i) \geq 1$ 。

C 为正整数数组。

定理 1：当 $k = |S|$ 时，每次截取字典序最小的后缀一定是一种最优方案。

证明：由 **引理 1** 可将 S^r 作最小后缀分解。为了最大化之后加入 T 的字符串的前缀中 A 的连续出现次数，必然取最后的若干个 A 均可。 □

截至目前，已解决 $k = |S|$ 时的问题。

不严谨地，考虑有 k 的限制，那么就需要尽量压缩操作次数，使 T 与 $k = |S|$ 时的答案尽量接近。方法如下：

(1) 如果多次截取的字符串相同，可以压缩为一次。

(2) 如果多次截取的字符串都为回文串，可以压缩为一次。

这两个方法是不冲突的，因为一个回文串重复若干次依旧是一个回文串。

定理 2：当 $k \geq 3$ 时，对 S^r 作最小后缀分解，若 $\text{len}(A) \neq 1$，则一定存在一种最优方案，第一次截取 $c_m \times A$ 。

证明：因为 A 是字典序最小的后缀，所以 $\text{len}(A) \neq 1$ 等价于 A 不是回文串。

若 $m > 1$ ，因为 $k \geq 3$ ，所以能够取到 $(\max(c_1, c_2, ..., c_{m-1}) + c_m) \times A$。

若 $m = 1$ ，显然。 □

至此，我们已经可以对此题给出一个时间复杂度 $\mathcal{O}(|S|)$ 的做法 **算法 1** 了：

步骤 (1) 当 $k \geq 3$ 的时候，截取 $c_m \times A$ 。如果 A 和截取后的 S^r 的最小后缀都是回文串，则不消耗操作数。

步骤 (2) 当 $k \leq 2$ 的时候，暴力枚举所有操作可能。

步骤 1 通过 Lyndon 分解实现。

步骤 2 可以通过字符串数据结构 $\mathcal{O}(|S|)$ 预处理，$\mathcal{O}(1)$ 查询 $S + S^r$ 中任意两个后缀的最长公共前缀完成比较。

显然对于大部分选手而言，这个做法的 **步骤 2** 的实现令人难以接受，需要发掘性质寻找更优美的做法。

当 $k = 2$ 时，对于所有可能的操作，并没有一概而论的性质，需要讨论截取之后是否翻转分别解决。

定义字符串数组 D, E 满足：

条件 (1) $\text{len}(D) = \text{len}(E)$

条件 (2) $D_1 + D_2 + D_3 + ... + D_{\text{len}(D)} = S^r$

条件 (3) $\forall i \quad \exists j \in \mathbf{Z}^* \quad D_i = j \times E_i$

条件 (4) $\forall i \quad E_i$ 是 $D_1 + D_2 + D_3 + ... + D_{i-1} + D_i$ 的最小后缀。

条件 (5) $\forall i \quad E_i$ 是 $D_i + D_{i+1} + ... + D_{\text{len}(D)}$ 的最长的满足是其本身最小后缀的前缀。

条件 (6) $\forall i \quad E_i > E_{i+1}$

上述结构就是重新定义了下 Lyndon 分解的结果。

令 $\text{SD}(i) = D_i + D_{i+1} + ... + D_{\text{len}(D)}$ 。

定理 3：若 $k = 2$ 时，第一段不翻转，则一定存在一种最优方案满足第一段截取的字符串 tmp 满足 $\exists i \quad \text{tmp} = \text{SD}(i)$ 。

证明：假设存在断点 p ，使得截取字符串 $S^r[p : \text{len}(S)]$ 优于截取所有 $\text{SD}(i)$ 。

若 p 是 E 的断点，则其左右两侧是相同的字符串 E_j ，则截取 $\text{SD}(j)$ 和 $\text{SD}(j+1)$ 至少有一种不会更劣。

否则，若 p 在 D_j 中，则截取 $\text{SD}(j)$ 一定更优。 □

引理 2：若字符串 t_1 满足其本身是其字典序最小的后缀，且 $t_1 > t_2 > t_3$ ，则 $t_1 > t_2 + t_2 > t_2 + t_3$。

证明：分类讨论。

若 t_2 不是 t_1 的前缀，则 t_1 与 t_2 比较时直接产生不同。

若 t_2 是 t_1 的前缀，则 $t_3 < t_2 < t_1 < t_1$ 非其本身的任意后缀，固成立。 □

定理 4：$\forall i \quad \text{SD}(i) > \text{SD}(i+1)$ 。

证明：将 $\text{SD}(i)$ 和 $\text{SD}(i+1)$ 都表示为若干个 E 中元素拼接，然后使用 **引理 2** 的结论。 □

定理 5：若 $\text{SD}(i+1)$ 为 $\text{SD}(i)$ 的前缀，则 $\text{len}(D_i) > \text{len}(\text{SD}(i+1))$ 。

证明：假设该命题不成立，则 $\text{SD}(i)$ 有一个长度为 $\text{len}(D_i)$ 且不足总长一半的周期，矛盾。 □

令 p 为最大的正整数满足 $\text{SD}(p+1)$ 不为 $\text{SD}(p)$ 的前缀。

则根据 **定理 3** 和 **定理 4** 可得若 $k = 2$ 时，第一段不翻转，则一定存在一种最优方案满足第一段截取的字符串 tmp 满足 $\exists i > p \quad \text{tmp} = \text{SD}(i)$ 。

并且由 **定理 5** 得，满足条件的 i 的数量 $\leq \log_2^{|S|}$ 。

至此，我们可以给出一个较为优美的算法 **算法 2** 了：

步骤 (1) 当 $k \geq 3$ 的时候，截取 $c_m \times A$ 。如果 A 和截取后的 S^r 的最小后缀都是回文串，则不消耗操作数。

步骤 (2) 当 $k \leq 2$ 的时候，特判 $k = 1$ 以及 $\text{len}(D) = 1$ 后，解决如下四种情况：

情况 (1) 第一段翻转，第二段翻转。得到结果均为 S 。

情况 (2) 第一段翻转，第二段不翻转。因为查询的 lcp 都是关于 S 的，使用 $z - function(exkmp)$ 预处理。

情况 (3) 第一段不翻转，枚举至多 $\mathcal{O}(\log(|S|))$ 种可能，使用二分哈希比较大小。

继续分析性质。

接下来着重分析 $k = 2$，第一段不翻转，第二段翻转的特殊情况。

已知结论 (1) 只可能先截取满足 $i > p$ 的 $\text{SD}(i)$。

已知结论 (2) $\forall p < i < j \quad \text{SD}(j)$ 是 $\text{SD}(i)$ 的长度小于一半的前缀。

令 $\text{AD}(i)$ 为第一段截取 $\text{SD}(i)$ 不翻转，第二段翻转的答案。

定理 6：$\forall i > p$ 若 $\text{AD}(i+1) < \text{AD}(i)$ ，则 $\forall j \in (p, i] \quad \text{AD}(i+1) < \text{AD}(j)$ 。

证明：因为 $\text{AD}(i+1)\,[\,\text{len}(\text{SD}(i)) + 1 :\,] < \text{AD}(i)\,[\,\text{len}(\text{SD}(i)) + 1 :\,]$，

所以 $\text{AD}(i+1)\,[\,:\text{len}(\text{SD}(i))\,] < \text{AD}(i)\,[\,:\text{len}(\text{SD}(i))\,]$。

又因为 **已知结论 (2)** 即得结论。 □

令 q 为满足 $\text{AD}(q) < \text{AD}(q-1)$ 且 $q > p+1$ 的最大的数，若没有满足的则 $q = p+1$ 。

则 $\text{AD}(\text{len}(D)) \geq \text{AD}(\text{len}(D) - 1) \geq ... \geq \text{AD}(q)$。

又因为 **定理 6** 得 $\text{AD}(q)$ 为 $k = 2$ 第一段不翻转第二段翻转的最优解。

至此，我们可以给出一个极其优美的算法 **算法 3** 了：

步骤 (1) 当 $k \geq 3$ 的时候，截取 $c_m \times A$ 。如果 A 和截取后的 S^r 的最小后缀都是回文串，则不消耗操作数。

步骤 (2) 当 $k \leq 2$ 的时候，特判 $k = 1$ 以及 $\text{len}(D) = 1$ 后，解决如下四种情况：

情况 (1) 第一段翻转，第二段翻转。得到结果均为 S 。

情况 (2) 第一段翻转，第二段不翻转。因为查询的 lcp 都是关于 S 的，使用 $z-function(exkmp)$ 预处理。

情况 (3) 第一段不翻转，第二段不翻转，则求出 S^r 的最小循环表示即可。

情况 (4) 第一段不翻转，第二段翻转，暴力比较求出 q ，第一段截取 SD(q) 即可。

时间复杂度 $\mathcal{O}(|S|)$ 。空间复杂度 $\mathcal{O}(|S|)$ 。

2 RNG and XOR

2.1 题目大意

给定 N 和长为 2^N 的数组 A 。

有一个初始为 0 的变量 X 。用一个概率分布为 A 的随机数生成器每次随一个数 v ，将 X 变成 X 异或 v 。

对于每个 i ，求 X 第一次变成 i 的期望次数。

2.2 数据范围

$1 \le N \le 18$

2.3 解题过程

规定 p_i 为一次随机到 i 的概率。

本文中集合幂级数的乘法为异或卷积，$\oplus$ 表示异或，& 表示按位与。

2.3.1 算法一

因为运算为异或，所以 X 初始为 0 ，第一次变成 i 的期望次数等于 X 初始为 i ，第一次变成 0 的期望次数。

令 f_i 为若 X 初始为 i ，第一次变成 0 的期望次数。

根据其定义可得：

$$f_i = \begin{cases} 0 & i = 0 \\ \left(\sum_{j=0}^{2^N} f_{i\oplus j} \times p_j\right) + 1 & i \neq 0 \end{cases}$$

令 F 为 f 的集合幂级数，P 为 p 的集合幂级数。

令 I 为集合幂级数，满足 $\forall i \quad I_i = 1$ 。

则 $F \times P + I = F + c$ ，其中 c 为一个数，用于修复 $F_0 = 0$ 。

令 $S(F)$ 表示集合幂级数 F 中所有元素的和。

则 $S(F) \times S(P) + S(I) = S(F) + c$ 。

因为 $S(P) = 1$ ，所以 $c = S(I) = 2^N$。

代入得：

$$
\begin{gathered}
F \times P + I = F + 2^N \\
F \times (P - 1) = 2^N - I
\end{gathered}
$$

令 $G = P - 1$ ，如果求得 G 的逆元，只需要做一遍卷积就可以得到 F 。

很遗憾的是，G 并没有逆元，因为 $\sum_i G_i = 0$，即 $\hat{G}_0 = 0$ 。

庆幸的是，因为 $\sum_{i=1}^{2^N-1} G_i = -G_0$ ，所以 $\forall i \neq 0 \quad \hat{G}_i \neq 0$ 。

即，可以求出 $\forall i \neq 0 \quad \hat{F}_i$ 的值。

因为我们知道 $F_0 = 0$，只需要对 $\hat{F}_0$ 待定系数即可。

时间复杂度：$\mathcal{O}\left(2^N \times N\right)$ 。空间复杂度：$\mathcal{O}\left(2^N\right)$ 。

2.3.2 算法二

令 F_i 为一个序列，第 j 位为 j 次操作后 $X = i$ 的概率。

令 $G_i(x)$ 为 F_i 的生成函数。

令 $H_i(x) = \frac{G_i(x)}{G_0(x)}$ ，则 $[x^j]H_i(x)$ 为 j 次操作后后第一次 $X = i$ 的概率。

根据期望的定义，第一次变成 i 的答案 $\mathrm{Ans}_i = H_i'(1)$ 。

联立得：$\mathrm{Ans}_i = \left(\frac{G_i}{G_0}\right)'(1) = \frac{G_i'(1) \times G_0(1) - G_i(1) \times G_0'(1)}{G_0(1) \times G_0(1)}$

所以只需要对于每个 i ，求出 $G_i(1)$ 和 $G_i'(1)$ 即可。

令 B 为长度 2^N 的集合幂级数，$B_i(x) = p_i x$ 。

令 C 为长度 2^N 的集合幂级数，$C_i(x) = G_i(x)$ 。

则 $C = B^0 + B^1 + B^2 + \dots$

即 $\hat{C} = \hat{B}^0 + \hat{B} + \hat{B}^2 + \dots$

不难发现 $\hat{B}$ 非常容易计算且 $\forall i \in [0, 2^N)$ ，$\hat{B}_i(x)$ 只有一次项的系数可能不为 0 。

那么 $\hat{C}_i(x) = \frac{1}{1 - \hat{B}_i(x)} = \frac{1}{1 - \left([x]\hat{B}_i(x)\right)x}$

言归正传，我们要求 $G_i(1)$ 和 $G_i'(1)$

$G_i(1) = C_i(1) = \sum_{j=0}^{2^N-1} (-1)^{\mathrm{bitcount}(i \& j)} \times \hat{C}_j(1)$

$G_i'(1) = C_i'(1) = \sum_{j=0}^{2^N-1} (-1)^{\mathrm{bitcount}(i \& j)} \times \hat{C}_j'(1)$

需要注意的一点是，虽然看上去本题已经做完了，但是我们忽略了一个重要问题：

$\hat{C}_0(x)$ 在 $x=1$ 处没有定义。

但这只是一个小问题，因为不难证明 $\forall i \in (0, 2^N)$ ，$[x]\hat{B}_i(x) \neq 1$，即 $\hat{C}_i(x)$ 在 $x=1$ 处有良定义。

运用常见技巧修一下即可。

令 D 为长度 2^N 的集合幂级数，$D_i(x) = C_i(x) \times (1-x)$ 。

则 $\hat{D}_i(x) = \hat{C}(x) \times (1-x) = \frac{1-x}{1-\left([x]\hat{B}_i(x)\right)x}$

令 $\mathrm{GG}_i(x) = G_i(x) \times (1-x)$

$\mathrm{GG}_i(1) = D_i(1) = \sum_{j=0}^{2^N-1} (-1)^{\mathrm{bitcount}(i\&j)} \times \hat{D}_j(1)$

$\mathrm{GG}'_i(1) = D'_i(1) = \sum_{j=0}^{2^N-1} (-1)^{\mathrm{bitcount}(i\&j)} \times \hat{D}'_j(1)$

最后一步：$\mathrm{Ans}_i = \left(\frac{G_i}{G_0}\right)'(1) = \left(\frac{\mathrm{GG}_i}{\mathrm{GG}_0}\right)'(1) = \frac{\mathrm{GG}'_i(1)\times\mathrm{GG}_0(1)-\mathrm{GG}_i(1)\times\mathrm{GG}'_0(1)}{\mathrm{GG}_0(1)\times\mathrm{GG}_0(1)}$

时间复杂度：$\mathcal{O}\left(2^N \times N\right)$ 。空间复杂度：$\mathcal{O}\left(2^N\right)$ 。

3 Inversion Sum

3.1 题目大意

给定长度为 N 的序列 A 。依次进行 Q 轮操作。第 i 次操作被描述为两个整数 X_i 和 Y_i ，并且在这次操作中，可以选择交换 A_{X_i} 和 A_{Y_i} 或什么也不做。

因此，存在 2^Q 种不同的方法来执行完所有的操作。每一种方法结束后都可以得到一个最终序列，我们想知道对于所有 2^Q 个最终序列，他们的各自的逆序对数之和是多少。

3.2 数据范围

$1 \leq N \leq 3000$

$0 \leq Q \leq 3000$

3.3 解题过程

使用动态规划，令 $F_{k,i,j}$ 为 k 轮之后，A_i 比 A_j 大的方案数。

则根据定义：

$$F_{k,i,j} = \begin{cases} F_{k-1,i,j} \times 2 & \text{如果 } i,j,X_k,Y_k \text{ 互不相同} \\ F_{k-1,X_k,j} + F_{k-1,Y_k,j} & \text{如果 } i = X_k\text{ , } j \neq Y_k \text{ 或 } i = Y_k\text{ , } j \neq X_k \\ F_{k-1,i,X_k} + F_{k-1,i,Y_k} & \text{如果 } j = X_k\text{ , } i \neq Y_k \text{ 或 } j = Y_k\text{ , } i \neq X_k \\ F_{k-1,X_k,Y_k} + F_{k-1,Y_K,X_k} & \text{如果 } j = X_k\text{ , } i = Y_k \text{ 或 } j = Y_k\text{ , } i = X_k \end{cases}$$

观察到对于某一个 k ，至多只有 $\mathcal{O}(N)$ 对 (i,j) 满足 $F_{k,i,j} \neq F_{k-1,i,j} \times 2$ 。

令 $G_{k,i,j} = F_{k,i,j} \times 2^{-k}$

则:

$$G_{k,i,j} = \begin{cases} G_{k-1,i,j} & \text{如果 } i,j,X_k,Y_k \text{ 互不相同} \\ \frac{1}{2}(G_{k-1,X_k,j} + G_{k-1,Y_k,j}) & \text{如果 } i = X_k\text{ , } j \neq Y_k \text{ 或 } i = Y_k\text{ , } j \neq X_k \\ \frac{1}{2}(G_{k-1,i,X_k} + G_{k-1,i,Y_k}) & \text{如果 } j = X_k\text{ , } i \neq Y_k \text{ 或 } j = Y_k\text{ , } i \neq X_k \\ \frac{1}{2}(G_{k-1,X_k,Y_k} + G_{k-1,Y_K,X_k}) & \text{如果 } j = X_k\text{ , } i = Y_k \text{ 或 } j = Y_k\text{ , } i = X_k \end{cases}$$

由于 $Ans = \sum_{i<j} F_{Q,i,j} = \sum_{i<j} G_{Q,i,j} \times 2^Q$

所以只需要对于每对 (i,j) 求出 $G_{Q,i,j}$ 即可。

因为对于任意 k，至多只有 $\mathcal{O}(N)$ 对 (i,j) 满足 $G_{k,i,j} \neq G_{k-1,i,j}$ 。可以将三维数组滚动 k 这一维，暴力修改可能变动的部分，这部分执行 Q 次的时间复杂度为 $\mathcal{O}(NQ)$ 。

时间复杂度 $\mathcal{O}\left(NQ + N^2\right)$ 。空间复杂度：$\mathcal{O}\left(N^2\right)$ 。