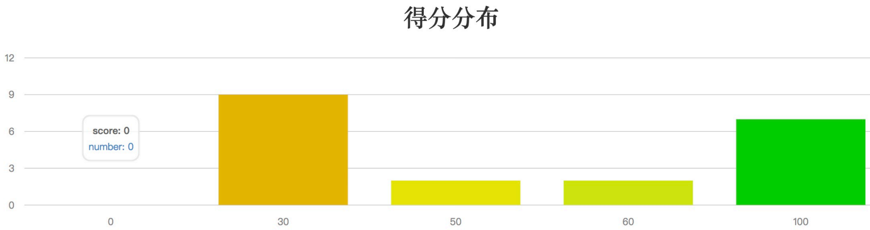
Day1 讲题 + 字符串

Scape

July 9, 2018

得分情况



Solution

先进行一些化简。
令 $f(i) = \sum_{j=1}^{i} \sum_{k=1}^{i} \gcd(i, j, k)$
那么

$$f(i) = \sum_{d|i} \sum_{j=1}^{i} \sum_{k=1}^{i} d[gcd(i,j,k) == d]$$

$$= \sum_{d|i} \sum_{j=1}^{\frac{i}{d}} \sum_{k=1}^{\frac{i}{d}} d[gcd(\frac{i}{d}, j, k) == 1]$$

Solution

$$= \sum_{d|i} \sum_{j=1}^{\frac{i}{d}} \sum_{k=1}^{\frac{i}{d}} d \sum_{e|gcd(\frac{i}{d},j,k)} \mu(e)$$

令 $T = de$
那么

$$f(i) = \sum_{T|i} \sum_{d|T} d\mu(\frac{T}{d})(\frac{i}{T})^2$$

$$= \sum_{T|i} \phi(T)(\frac{i}{T})^2 = \sum_{T|i} \phi(\frac{i}{T}) T^2$$

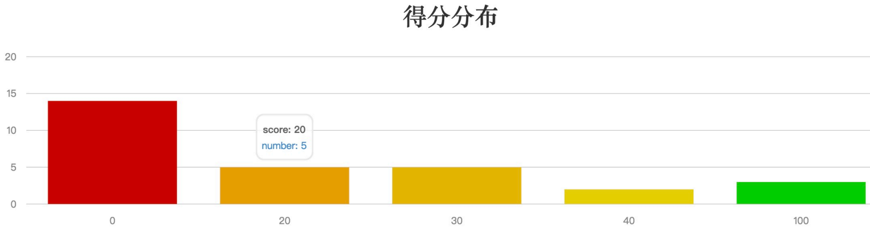
可以发现 $f(i)$ **是积性函数，可以线性筛。**

Solution

$$\sum_{i=1}^{n} f(i) = \sum_{i=1}^{n} \sum_{T|i} \phi(\frac{i}{T}) T^2 = \sum_{T=1}^{n} T^2 \sum_{i=1}^{\frac{n}{T}} \phi(i)$$

杜教筛一波就好了。

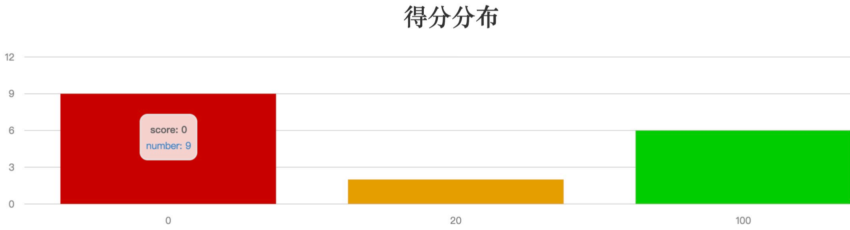
得分情况



Solution

考虑一下 a 到 b 必须经过点 c 这个条件他代表的意义，实际上就是限制了以 c 点为根的树里，a 和 b 不能在同一个子树内。
所以以 1 为根, 令 $v_{i,j}$ 表示礼物 i 是否在子树 j 内, 那么每个条件都可以变成一个 $2-SAT$ 里的限制。
考虑一下，这是一个 nm 个点,n^2m+qn 条边的 2-SAT，故时间复杂度为 $O(n^2m+qn)$。

得分情况



Solution

考虑按高度从小到大分步计算答案。每次把高度相同的放在一起做
每次高度相同的可能会有若干行若干列。
注意到删点是一行行一列列删的。所以当前行还剩的点数是每一行的总点数-删去的列数。列亦然
所以我们要求的就是，现在这个高度下，对所有没有被覆盖的点给一个高度，使得这若干行若干列都至少有一个点达到了现在这个点的高度的方案数。

Solution

那么考虑容斥。我们算 i 行 j 列炸了。那么贡献就是。

$$(h+1)^{cntA} \times h^{cntB} \times (-1)^{i+j}\binom{i+j}{i}$$

cntA 表示未炸点数,cntB 表示已炸点数。
已炸点数和未炸点数可以通过上面的方法轻易地算出来。
那么再把每一个高度的总贡献乘起来就好啦。

字符串基础

1.SA
2.SAM
3. 后缀树
4. 回文树
5.SAM/后缀数组 on Trie
相信大家都会

[TJOI&HEOI2016] 字符串

字符串 S
Q 次询问，每次给出 $[a, b][c, d]$，询问 $s[a, b]$ 的所有子串和 $s[c, d]$ 的最长公共前缀的最大值。
$|S| 10^5$

[TJOI&HEOI2016] 字符串

把 S 取反建 SAM 可二分每次二分后缀长度找到后缀问 S[a-b] 是否存在该后缀主席树维护 Right 集合

NOI2016 失恋测 D5T2

2 可持久化字符串(string.c/cpp/pas)

2.1 题目描述

一个串T是S的循环节，当且仅当存在正整数k，使得S是T^k(即T重复k次)的前缀，比如abcd是abcdabcdab的循环节。

维护一个字符串S，一开始是空串，进行m次操作，每次操作包含两个整数x_i, c_i，表示这次操作的字符串为在第x_i次操作之后的字符串末尾添加一个字符c_i所形成的字符串。

请在每次操作完毕之后，求出该次操作得到的字符串最短的循环节的长度。

强制在线。

NOI2016

如果一个字符串可以被拆分为 $AABB$ 的形式，其中 A 和 B 是任意非空字符串，则我们称该字符串的这种拆分是优秀的。

例如，对于字符串 aabaabaa，如果令 $A = \text{aab}$，$B = \text{a}$，我们就找到了这个字符串拆分成 $AABB$ 的一种方式。

一个字符串可能没有优秀的拆分，也可能存在不止一种优秀的拆分。比如我们令 $A = \text{a}$，$B = \text{baa}$，也可以用 $AABB$ 表示出上述字符串；但是，字

现在给出一个长度为 n 的字符串 S，我们需要求出，在它所有子串的所有拆分方式中，优秀拆分的总个数。这里的子串是指字符串中连续的一段。

以下事项需要注意:

1. 出现在不同位置的相同子串，我们认为是不同的子串，它们的优秀拆分均会被记入答案。
2. 在一个拆分中，允许出现 $A = B$。例如 cccc 存在拆分 $A = B = \text{c}$。
3. 字符串本身也是它的一个子串。

NOI2016

切关键点算一算。

BZOJ4310

给定一个长度为 n 的字符串 S，请将 S 分成不超过 k 段，使得每一段的所有子串中，字典序最大的子串最小。
10^5

Fim1 Seal

给定一个字符串 S，维护一个字符串序列 T。
T.append(T[x]+c)
T.append(c+T[x])
询问 T[x] 在 S[l:r] 中的出现次数。
$|S|, M \leq 10^5$

BZOJ4231

给定一个边带字母的树，每次询问一段路径上顺次连接形成的字符串中给定的串 s 出现了多少次。
$n, m <= 10^5$
$\sum |s| <= 3 * 10^5$

CF 700E

一个字符串对另一个字符串是好的当且仅当这个字符串在另一个字符串中出现了两次 eg:"abc"->"abcabc" "ddd"->"dddd"
给定一个字符串 S_1, 最大化 k 使得字符串序列 $S_1, \cdots, S_k$, 并且满足 S_2 对 S_1,S_{i+1} 对 S_i 都是好的。
$|S| 2 \times 10^5$

CF 700E

S_i 显然是 S_1 的子串先建 SAM
对于 $i \geq 2, S_{i+1}$ 一定是 S_i 的一个后缀（否则我们把 S_i 的最后一位切掉，仍然符合条件。
那么 S_{i+1} 一定是 S_i parent 树上的一个祖先。
令 $fa[i]$ 表示 i 最近的祖先并且使得 i 在 $fa[i]$ 内出现了两次那么 $dp[i] = dp[fa[i]] + 1$

CF 700E

怎么计算 fa[i]？
显然是可以二分的
二分后如何判断一个节点在另外一个节点中的出现次数？
在 i 的 Right 集里随便挑一个 x 后询问 $fa[i]$ 的 Right 集里在 $[x-len[i]+1,x]$ 的个数
可持久化权值线段树
时间复杂度 $O(nlog^2n)$

某忘记出处的模拟赛题

某忘记出处的模拟赛题

- 给定字符串 S

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i,j)$ 为最大的数 x, 使得

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i,j)$ 为最大的数 x, 使得
- $S(i-x+1,i) = S(j-x+1,j)$ 且 $S(i-x+1,i)$ 回文

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i, j)$ 为最大的数 x, 使得
- $S(i-x+1, i) = S(j-x+1, j)$ 且 $S(i-x+1, i)$ 回文
- $F(x) = \sum_{1 \le i \le x} \sum_{1 \le j \le x} plcp(i, j)$

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i, j)$ 为最大的数 x, 使得
- $S(i-x+1, i) = S(j-x+1, j)$ 且 $S(i-x+1, i)$ 回文
- $F(x) = \sum_{1 \le i \le x} \sum_{1 \le j \le x} plcp(i, j)$
- 每次在 S 后添加一个字母 c, 输出 $F_{|S|}$

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i,j)$ 为最大的数 x, 使得
- $S(i-x+1,i)=S(j-x+1,j)$ 且 $S(i-x+1,i)$ 回文
- $F(x)=\sum_{1\le i\le x}\sum_{1\le j\le x} plcp(i,j)$
- 每次在 S 后添加一个字母 c, 输出 $F_{|S|}$
- 强制在线

某忘记出处的模拟赛题

- 给定字符串 S
- 定义 $plcp(i,j)$ 为最大的数 x, 使得
- $S(i-x+1,i) = S(j-x+1,j)$ 且 $S(i-x+1,i)$ 回文
- $F(x) = \sum_{1\le i\le x}\sum_{1\le j\le x} plcp(i,j)$
- 每次在 S 后添加一个字母 c, 输出 $F_{|S|}$
- 强制在线
- $|S| \le 10^5$

某忘记出处的模拟赛题

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$
- 显然就是回文树上第 |S| 个字母代表的节点与其它所有节点的 LCA 的长度和

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$
- 显然就是回文树上第 |S| 个字母代表的节点与其它所有节点的 LCA 的长度和
- 考虑它每个祖先的贡献显然是 $size(x) \times (v[fa[x]] - v[x])$

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$
- 显然就是回文树上第 |S| 个字母代表的节点与其它所有节点的 LCA 的长度和
- 考虑它每个祖先的贡献显然是 $size(x) \times (v[fa[x]] - v[x])$
- 用个 lct 维护一波就好

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$
- 显然就是回文树上第 |S| 个字母代表的节点与其它所有节点的 LCA 的长度和
- 考虑它每个祖先的贡献显然是 $size(x) \times (v[fa[x]] - v[x])$
- 用个 lct 维护一波就好
- 时间复杂度 $O(nlogn)$

某忘记出处的模拟赛题

- 考虑 $F_{|S|} - F_{|S|-1}$
- 显然就是回文树上第 |S| 个字母代表的节点与其它所有节点的 LCA 的长度和
- 考虑它每个祖先的贡献显然是 $size(x) \times (v[fa[x]] - v[x])$
- 用个 lct 维护一波就好
- 时间复杂度 $O(nlogn)$
- 出题人写了紫荆花之恋不知道是什么想法。。

2017 集训队互测 +2018 湖南省队集训

区间本质不同回文子串个数。

2017 集训队互测 +2018 湖南省队集训

离线 $n = 10^5$

2017 集训队互测 +2018 湖南省队集训

在线 $n = 10^5$

2017 集训队互测 +2018 湖南省队集训

离线 $n = 5 \times 10^5$

某同样忘记出处的模拟赛题

某同样忘记出处的模拟赛题

- 求一个字符串的所有子串中，任选 2 个（可以相同），计算 $\sum(|A| + |B| - 2LCP(A, B) \leq L)$。

某同样忘记出处的模拟赛题

- 求一个字符串的所有子串中，任选 2 个（可以相同），计算 $\sum(|A| + |B| - 2LCP(A,B) \leq L)$。
- $N \leq 10^5$，串伪随机（其实不伪随机也可以做）

某同样忘记出处的模拟赛题

某同样忘记出处的模拟赛题

- 问题也就是询问该串的后缀树上，长度不超过 L 的路径条数。

某同样忘记出处的模拟赛题

- 问题也就是询问该串的后缀树上，长度不超过 L 的路径条数。
- 那么树分治。

某同样忘记出处的模拟赛题

- 问题也就是询问该串的后缀树上，长度不超过 L 的路径条数。
- 那么树分治。
- 伪随机是给边分治用的。

某同样忘记出处的模拟赛题

- 问题也就是询问该串的后缀树上，长度不超过 L 的路径条数。
- 那么树分治。
- 伪随机是给边分治用的。
- 点分治也是可以上的！写起来略麻烦。

某中二少女出的鬼畜模拟题

给出一棵 N 个点树，点有黑白，根节点（1 号节点）为白色，fsf 在这棵人间的树上行走，她首先选择一个起点，每次先进行判定，若她还留在人间，则继续向父节点行走。
判定：
1. 若当前在根节点上，她有 $1/2$ 的概率升上天堂，她有 $1/4$ 的概率坠落地狱，1/4 的概率重新选择起点进行行走。
2. 否则若当前在白点上，她有 $1/2$ 的概率升上天堂，1/2 的概率留在人间。
3. 否则当前在黑点上，她有 $1/4$ 的概率坠落地狱，1/4 的概率重新选择起点进行行走，1/2 的概率留在人间。
她最多只能选择 K 次起点（选择的起点不能重复），第 $K+1$ 次他选择起点时将坠落地狱。起点的选择是他可以任意选择的，求最优策略下他升上天堂的概率。
$N \leq 5000000$
保证输出大小不超过 1MB。保证树的深度不超过 50000。

某中二少女出的鬼畜模拟题

考虑令 p_x 为从点 x 出发到根升上天堂的概率。
显然应该按照 p_x 从大到小选起点。
将概率表示成二进制小数的形式，可以发现我们应该构建出 Trie 上的后缀数组。
在此之后，令 f_i 为第 i 次选起点的升入天堂的概率，有 $f_i = p + \frac{1-p}{2} f_{i+1}$。

某中二少女出的鬼畜模拟题

考虑合并信息，注意到若 $f_l = a + bf_m$ $f_m = c + df_r$，则 $f_l = a + bc + bdf_r$。
令 $len(l, r)$ 为合并 l 到 r 的信息之后，得到 $f_l = a + bf_r$，a, b 的二进制表示的长度的较大值。
则按照 $len(l, m) + len(m + 1, r)$ 为代价每次合并代价最小的。
为了复杂度后缀数组需要在线性时间内建出。

某中二少女出的鬼畜模拟题

考虑合并信息，注意到若 $f_l = a + bf_m$ $f_m = c + df_r$，则 $f_l = a + bc + bdf_r$。
令 $len(l, r)$ 为合并 l 到 r 的信息之后，得到 $f_l = a + bf_r$，a, b 的二进制表示的长度的较大值。
则按照 $len(l, m) + len(m + 1, r)$ 为代价每次合并代价最小的。
为了复杂度后缀数组需要在线性时间内建出。

某中二少女出的鬼畜模拟题

考虑线性时间构造 Trie 上后缀数组的方法。
构造出 Trie 上的后缀自动机，考虑它的一棵 dfs 树，每条边 (u, v) 均在后缀自动机中满足 $len(u) + 1 = len(v)$。
在 Trie 上的后缀树层面考虑，令 pop( ) 为在节点 代表的字符串前去掉一个字符对应的后缀树节点。既然 被建出，则必有代表后缀的节点 ，使得 lca(,)= 。则显然 lca(pop( ),pop( ))=pop( )，证毕。
合并信息的时候需要 FFT 优化高精度，最后需要将二进制转换成十进制，同样用 FFT 优化。
复杂度 $O(N + K\log^2 L)$。

某中二少女出的鬼畜模拟题

考虑线性时间构造 Trie 上后缀数组的方法。
构造出 Trie 上的后缀自动机，考虑它的一棵 dfs 树，每条边 (u, v) 均在后缀自动机中满足 $len(u) + 1 = len(v)$。
在 Trie 上的后缀树层面考虑，令 pop(x) 为在节点 x 代表的字符串前去掉一个字符对应的后缀树节点。既然 x 被建出，则必有代表后缀的节点 y,z 使得 lca(y,z)=x。则显然 lca(pop(y),pop(z))=pop(x)，证毕。
合并信息的时候需要 FFT 优化高精度，最后需要将二进制转换成十进制，同样用 FFT 优化。
复杂度 $O(N + K\log^2\ L)$。

Deep Purple

给定一个长度为 n 的字符串 S，q 次询问。
每次询问 $s[l:r]$ 的 LBorder（即最大的 $i \leq r-l+1$ 使得 $S[l:l+i-1] = S[r-i+1:r]$）。
$n, q \leq 10^5$

Deep Purple

考虑后缀树，我们要求找到一个最小的 i 使得 $l < i \le r \le i + LCP(l, i) - 1$。
树剖，然后从左往右扫，依次加入 l 和 i，注意到对于 l 我们只要求最小的 i，因此我们可以在加入 i 的时候暴力找 l 并将其删除。
考虑在 LCP(l, i) 节点上处理 i 对 l 的贡献，回忆树剖求 lca 的过程中的最后一步中，当 l 和 i 来到同一条重链时的情形，我们就在这条重链上处理。
若此时 $dep_l < dep_i$，则有 $r - dep_l + 1 \le i$，可以用线段树维护 $r - dep_l + 1$ 的最小值。
否则有 $r \le i + depi - 1$，可以用线段树维护 r 的最小值。
时间复杂度 $O(n \log^2 n)$。

Fim4

给定 n 个字符串 s_i 和 m 个 $1 \sim n$ 之间的整数 a_i，令母串为 $s_{a_1} + s_{a_2} + \cdots + s_{a_m}$，回答 q 次询问，每次给出一个字符串 t_i，询问这个串在母串中的出现次数。
保证 s_i 和 t_i 都只由字母 a,b 组成。
$1 \leq n, m, q \leq 5 \times 10^4$
$\sum_{1 \leq i \leq n} |s_i|, \sum_{1 \leq i \leq q} |t_i| \leq 10^5$。

Fim4

询问串在母串中的一次匹配只有两种情况：要么是询问串在单个字典串内出现，要么询问串横跨了若干个个字典串，我们对两部分分别处理。

Part 1

我们每次给出一个询问串 t，要计算 $\sum_{i=1}^{n} c(t, s_{a_i})$ 。这是一个比较常见的问题。我们对字典建 trie，再对这个 trie 建出它的 SAM。
类似解法一，我们对于 SAM 上的每个节点，预处理出它所代表的字符串在 S 中的出现次数。
每次询问时找到询问串 t 在 parent 树上所对应的节点，就能直接得到答案了。
这一部分的复杂度是 $O(\sum |s_i| + \sum |t_i|)$。

Part 2

我们设置一个 B，对于询问串长度小于 B 的我们都用 Part2 的算法结局。
形式化地说，计算跨越多个字典串的出现次数也就是计算 $\sum_{i=1}^{n-1} c(t, S[\max(lp_i, rp_i - |t| + 2) : rp_i + |t| - 1])$ 。
我们可以把所有 $S[\max(lp_i, rp_i - L + 2) : rp_i + L - 1]$ 拿出来建 Trie ，并对该 Trie 建出 SAM。对于前缀 $S[\max(lp_i, rp_i - L + 2) : rp_i + k - 1]$ 我们将其标号为 k 。

Part 2

统计 $\sum_{i=1}^{n-1} c(t, S[\max(lp_i, rp_i - |t| + 2) : rp_i + |t| - 1])$ 也就是相当于在统计 parent 树上 t 所代表的节点的子树中有多少节点的标号在 $[2, |t| - 1]$ 中，对 parent 树求 dfs 序后可以转化为二维数点问题。
我们注意到这个二维数点的第二维是 $O(L)$ 级别的。并且总点数是 $O(mL)$ 级别的，询问是 $O(q)$ 级别的。我们可以轻松地通过离线做到加入 $O(1)$ 询问 $O(L)$，因此这一部分的总复杂度是 $O(mL)$ 的。

综上所述，我们可以在 $O(\sum |s_i| + \sum |t_i| + mL)$ 的复杂度内解决这个部分。

Part 3

我们考虑对于询问串 $> B$ 的询问。
我们先介绍两个引理:

引理

如果 p, q 都是字符串 S 的周期，且 $p+q \leq |S|$, 那么 $\gcd(p, q)$ 也是字符串 S 的周期。

Proof.

令 $d = q - p(p < q)$，则由 $i - p > 0$ 或 $i + q \leq |S|$ 均可推出 $S_i = S_{i+d}$。

□

Part 3

引理

令 l 是周期长度，对于 $i > 2l, next_i = i - l$。

Proof.

这等价于对于 $i > 2l$ 的前缀，它的最短周期是 l。我们使用反证法：如果存在比 l 更短的周期 T，那么 $T + l < i$, 所以由引理 3.1 可得 $\gcd(T, l)$ 也是这个串的周期，注意到 $\gcd(T, l) | l$，这与 l 是整个串的最短周期矛盾，故得证。

□

Part 3

考虑优化 KMP 算法。我们考虑把询问串放在母串上跑 KMP。我们注意到我们只需要计算串之间的匹配就可以了。
我们令 i 表示当前 KMP 算法指向字典串的指针，j 表示当前 KMP 算法指向询问串的指针,L 表示询问串的长度。
如果存在某个 k，使得 $i \in [lp_k, rp_k]$ 且 $i - j + 1 \in [lp_k, rp_k]$, 那么说明当前正在匹配的是串内的，是不需要被计算的。所以我们直接令 j 等于最长的询问串前缀的长度，并且这个询问串的前缀是 s_{a_k} 的一个后缀，再把 i 设为那个后缀开始的地方再继续 KMP 即可。

Part 3

由于我们可以快速求出两个后缀的 LCP ，因此我们可以很容易地计算下一次失配的位置。因此我们只需要优化跳 next 的过程即可。
在失配之后如果 $next_j < \frac{j}{2}$ 我们就直接执行 $j := next_j$，否则：
如果执行 $j := next_j$ 后还会失配，根据引理，我们可以直接跳到 $l + j \bmod l$ 的位置（l 是周期长度）(Case (1))。

Part 3

否则我们统计接下来 $S[i+1:]$ 还能继续匹配多少个 $t[:j]$ 的周期并找到失配时的位置，此时有两种可能的情况，第一种情况是 $S[i+1]=t_j$(Case (2))，此时我们继续向下匹配，第二种情况同之前 Case (1)。
有一个特殊情况是我们匹配到了串末。这时候我们可以计算 $S[i+1:]$ 还能匹配多少个 $t[:j]$ 的周期，能匹配的周期数就是对答案的贡献。之后我们和失配情况一样处理即可。

Part 3

如何证明之前所述的算法的时间复杂度是 $O(m \log n)$ 的？
对于 Case (1)，考虑 $j-(i-lp_k)$ 的值。显然当 $j-(i-lp_k) \le 0$ 时，我们就会重置 i 和 j 的值。
首先，求 LCP 时，$j-(i-lp_k)$ 的值并不会变。
而在跳 next 时，对于 $next_j \le \frac{j}{2}$ 时，执行 $j := next_j$ 会让 j 变小 $\frac{1}{2}$。对于 $next_j > \frac{j}{2}$ 时，当我们执行 $j := l + j \bmod l$ 时，$l + j \bmod l \leqslant \lceil \frac{2l}{3} \rceil$，$j$ 会变小 $\frac{1}{3}$。所以每次 Case（1）操作就会让 $j-(i-lp_k) \le 0$ 变小至少 $\frac{1}{3}$。
所以我们发现 Case (1) 的总复杂度是 $O(m \log n)$ 的。

Part 3

对于 Case (2)，我们先引入一个关于层的定义：
我们令 $b_i = \left[next_i \geqslant \frac{i}{2}\right]$，将 b 序列中从左到右第 i 段连续的 1 称作第 i 层。
我们再证明一个有关层的性质：

Part 3

定理

令第 i 层的前缀的周期为 per_i，那么有 $\frac{3}{2}|per_i| \le |per_{i+1}|$

Proof.

令 pos_i 表示第 i 层最右边的位置，那么显然 per_i 和 per_{i+1} 都是 $S[:pos_i]$ 的周期。
如果 $|per_i| + |per_{i+1}| \le pos_i$，那么根据引理 3.2, 可以知道 $\gcd(|per_i|, |per_{i+1}|)$ 也是 $S[:pos_i]$ 的周期。注意到 $\gcd(|per_i|, |per_{i+1}|)$ 整除 $|per_{i+1}|$，这与 per_{i+1} 是第 $i+1$ 层的最短周期矛盾，故我们得到 $|per_i| + |per_{i+1}| > pos_i$
又有 $pos_i \ge 2|per_i|$，联立两式可得 $\frac{3}{2}|per_i| \le |per_{i+1}|$。

□

Part 3

那么对于 Case (2)，而根据定理 3.3 显然一共只有 $\log$ 层。注意到只有两种操作，而 Case 1 的操作每次只会倒退一层,Case 2 的操作每次只会增加一层。所以 Case (2) 的操作最多只有 $m\log n$ 次。
因此我们的总失配次数是 $O(m\log n)$ 的，而对于求 LCP 部分，我们建出字典串的 SAM on Trie 后直接查询对应他们在 SAM 上节点的 LCA 就做到查询他们的 LCP。注意到 LCA 问题可以转化成欧拉序上的 RMQ 问题，是可以做到 $O(1)$ 回答的。而对于重置 i, j 部分，我们也可以快速地在 $O(\log n)$ 的时间内求两个串的最长公共前后缀，所以我们总的时间复杂度也是 $O(m\log n)$ 的。

Solution

综上所述，我们取 $B = \sqrt{nlogn}$ 时复杂度最低，为 $O(n\sqrt{nlogn})$。