《太阳神的宴会》命题报告

福建省福州第一中学 陈雨昕

摘 要

《太阳神的宴会》[1]是我为 IOI2021 候选队员自主互测活动[2]命制的一道试题，本题要求选手探究一种等价关系下的字符串匹配。以往基于后缀数组的算法具有运行时间略高、不够直观的局限性。本文介绍了本题在不同的特殊条件下的解决思路与算法选择，其中满分算法通过引入辅助序列描述等价类，并将转移不同的等价类分开，将后缀自动机的应用范围扩展到了等价关系上，作者称它为子形状自动机；将本题所要求的相似关系和拼接过程，对应到子形状自动机上，转化为自动机所对应的有向无环图的带权路径计数；在此基础上用合并相同转移的手段进一步优化，从而解决了本题。进一步地，本文指出，针对一类字符串上的等价关系，子形状序列自动机可用于处理字符串的本质不等价子串相关问题。

记号的约定

本文讨论的字符串与序列均为有限长的。

字符集为 Σ 的全体字符串记作 Σ^*，空串记为 ϵ。对于字符串 s，记其长度为 $|s|$，从第一个字符到最后一个字符依次为 $s_1, s_2, \ldots, s_{|s|}$，记作 $s = s_1 s_2 \cdots s_{|s|}$。字符串 s 和 t 的连接记作 st。如果 $x, y, z \in \Sigma^*$ 使得 $s = xyz$，则称 x 为 s 的前缀，y 为 s 的子串，z 为 s 的后缀。s 的子串 $s[l, r]$ 是指字符串 $s_l s_{l+1} \cdots s_r$（$1 \le l \le r \le |s|$），特别地，对于不在上述范围内的 l, r，$s[l, r] = \epsilon$。两个字符串 s, t 的最长公共前缀长度简称 LCP，记为 $\mathrm{LCP}(s, t)$。字符串集合 S 的全体元素的最长公共前缀记为 $\mathrm{LCP}\, S$。

空序列也记作 ϵ。对于序列 a，记其长度为 $|a|$，第一项到最后一项依次为 $a_1, a_2, \ldots, a_{|a|}$，记作 $a = (a_1, a_2, \ldots, a_{|a|})$。序列 a 和 b 的连接记作 $a \cdot b$，如无歧义也可记作 ab。如果序列 x, y, z 使得 $a = xyz$，则称 x 为 a 的前缀，y 为 a 的连续子序列，z 为 a 的后缀。a 的连续子序列 $a[l, r]$ 是指序列 $(a_l, a_{l+1}, \ldots, a_r)$（$1 \le l \le r \le |s|$），特别地，对于不在上述范围内的 l, r，$a[l, r] = \epsilon$。两个序列 a, b 的 LCP 记为 $\mathrm{LCP}(a, b)$。序列集合 A 的全体元素的最长公共前缀记为 $\mathrm{LCP}\, A$。

[1] https://uoj.ac/problem/595

[2] https://uoj.ac/contest/58

一个部分确定有限状态自动机简称部分 DFA，用五元组 $(Q,\Sigma,\delta,\bar{q},F)$ 表示。其中，Q 是状态集合，Σ 是字符集，δ 是转移函数，$\bar{q}$ 是初始状态，F 是终态集合。在部分 DFA 中，状态 q 的 α 转移 $\delta(q,\alpha)$ 可以存在，也可以不存在，如果不存在，记 $\delta(q,\alpha)=0$。

对于命题 P，记艾弗森括号 $[P]=1$ 当且仅当 P 为真，$[P]=0$ 当且仅当 P 为假。

1 试题

1.1 题目描述

设字符集 Σ 为前 k 个小写英文字母的集合，题中所有字符串均由 Σ 中的字符构成。

对于 Σ 上的置换 f 和 $s\in\Sigma^*$，令 s 作用 f 后的结果为 $f(s)=f(s_1)f(s_2)\cdots f(s_{|s|})$。

我们称字符串 s 与字符串 t **相似**，当且仅当存在一个 Σ 上的置换 f，使得 $f(s)=t$，记作 $s\sim t$。例如，字符串 aab 与 bba 相似，与 abb 不相似。

现给出 n 个字符串 $S_1,S_2,\ldots,S_n$. 记 C_i 为全体与 S_i 的至少一个子串相似的字符串的集合。

称字符串 T 是**好**的，当且仅当存在字符串 $T_1,T_2,\ldots,T_n$，使得 $T_i\in C_i$，且 $T=T_1T_2\cdots T_n$。

求有多少个好的字符串。

1.2 输入格式

第一行两个正整数 n,k。

接下来 n 行，每行一个字符串，依次表示 $S_1,S_2,\ldots,S_n$。

1.3 输出格式

输出好的字符串的数量，对 998244353 取模。

1.4 样例 1

1.4.1 样例输入

```
2 2
aa
a
```

1.4.2 样例输出

11

1.4.3 样例解释

S_1 的子串有 ϵ, a, aa，与它们之一相似的串有 ϵ, a, b, aa, bb。

S_2 的子串有 ϵ, a，与它相似的串有 ϵ, a, b。

连接起来，好的串有：ϵ, a, b, aa, ab, ba, bb, aaa, aab, bba, bbb。

1.5 数据规模与约定

简记 $L = \sum_{i=1}^{n} |S_i|$。

保证 $2 \le k \le 26$，S_i 非空，$L \le 10^6$。

子任务一（2 分）：$n = 1$，$k = 2$，$L \le 1000$。

子任务二（7 分）：$n = 1$，$L \le 1000$。

子任务三（11 分）：$n = 1$，$L \le 10^5$。

子任务四（13 分）：$k = 2$，$L \le 1000$。

子任务五（17 分）：$L \le 1000$。

子任务六（31 分）：$k \le 5$。

子任务七（19 分）：无特殊限制。

1.6 时空限制

时间限制：3s

空间限制：2GB

2 做题情况

本次互测活动共 3 题，本题为其中第一道。

活动面向全体 Universal OJ 用户，共 122 位选手参与了本次互测活动。

共 28 位选手在本题上得分：2 分 6 位、9 分 13 位、15 分 1 位、22 分 1 位、39 分 6 位，最高分为 50 分，1 位。

3 初步分析

首先我们说明：

性质 3.1. $\sim$ 是一个等价关系。

证明. 自反性：对于字符串 s，取置换 $f = \mathrm{id}_\Sigma$，得 $s \sim s$。

对称性：设 $s \sim t$。则 $|s| = |t|$，且存在置换 f 满足 $\forall 1 \le i \le |s|$，$f(s_i) = t_i$。由于 f 是置换，可取其逆置换 f^{-1}，则 $f^{-1}(t_i) = s_i$。所以 $t \sim s$。

传递性：设 $r \sim s$，$s \sim t$。则 $|r| = |s| = |t|$，且存在置换 f, g 满足 $\forall 1 \le i \le |r|$，$f(r_i) = s_i$，$g(s_i) = t_i$。那么取置换的复合 $h = g \circ f$，得 h 也是置换，且 $\forall 1 \le i \le |r|$，$h(r_i) = t_i$。所以 $r \sim t$。 □

其次，题中字符串的相似关系具有以下性质：

性质 3.2. 若 $s \sim t$，则 $\forall 1 \le l \le r \le |s|$，$s[l, r] \sim t[l, r]$。

证明. 因为 $s \sim t$，所以存在置换 f，$\forall 1 \le i \le |s| = |t|$，$f(s_i) = t_i$。$|s[l, r]| = |t[l, r]| = r - l + 1$，且 $\forall l \le i \le r$，$f(s_i) = t_i$。故 $s[l, r] \sim t[l, r]$。 □

4 单字符串情形下的经典算法

对于只有一个字符串的情形，只需求出 $|C_1|$。

4.1 算法一：$n = 1$，$k = 2$，$L \le 1000$

在 $k = 2$ 的情形下，Σ 上只有两种置换，S_1 只有 $O(L^2)$ 种子串。因此 C_1 中所有可能出现的字符串可以直接枚举。考虑这些字符串的去重问题，容易想到用 Trie 来解决。

因此得出一个简单的算法：枚举 Σ 上的置换 f，将 S_1 的所有后缀作用 f 后插入 Trie。最终 Trie 的结点数即为所求。时空复杂度 $O(L^2)$，期望得分 2 分。

4.2 形状序列

当 k 更大时，枚举置换将带来巨大的时空复杂度。因此，我们要考虑一种更高效的方法来描述相似关系的等价类。

为了判定相似性，我们试图将字符串对应到一种辅助序列上，使得它记录字符间的相等关系，而丢弃字符的具体取值。

4.2.1 形状序列的引入

定义 4.1. 对于字符串 s，定义其形状序列为一个长度为 $|s|$ 的非负整数序列 a：

对于 $1 \le i \le |s|$，

- 若字符 s_i 在 s 中首次出现，令 $a_i = 0$；
- 若字符 s_i 在 s 中并非首次出现，那么在 $s[1, i-1]$ 中，找到所有字符的最后一次出现位置，从大到小记作 $j_1 > j_2 > \cdots > j_z$。如果 $s_{j_x} = s_i$，令 $a_i = x$。

由定义可得，字符串的形状序列有以下性质：

性质 4.1. 对于字符串 s 和整数 $0 \le r \le |s|$，如果 s 的形状序列为 a，那么 $s[1, r]$ 的形状序列为 $a[1, r]$。

性质 4.2. 一个字符串的不同字符个数，等于其形状序列中 0 的个数。

推论 4.1. 一个字符串的形状序列中 0 的个数不超过 k。

以下记 $n_0(a)$ 表示非负整数序列 a 中 0 的个数。

性质 4.3. 设一个字符串的形状序列为 a，则 $\forall 1 \le i \le |a|$，有 $a_i \le n_0(a[1, i-1])$。

接下来我们说明形状序列在相似关系方面的显著作用。

引理 4.1. 如果两字符串 s, t 相似，那么 $\forall 1 \le i < j \le |s|$，有 $s_i = s_j \iff t_i = t_j$。

证明. 由于 $s \sim t$，可得 $|s| = |t|$，且存在置换 f，使得 $\forall 1 \le i \le |s|$ 有 $f(s_i) = t_i$。对于 $1 \le i < j \le |s|$，$s_i = s_j \Longrightarrow t_i = f(s_i) = f(s_j) = t_j$。

由于 $t \sim s$，同理可得 $t_i = t_j \Longrightarrow s_i = s_j$。因此 $s_i = s_j \iff t_i = t_j$。 □

定理 4.1. 两字符串 s, t 相似，当且仅当 s 和 t 的形状序列相同。

证明. 先证必要性。对于 $1 \le i \le |s|$，

- 若 s_i 是首次出现的，则由引理 4.1，可得 t_i 也是首次出现的，即 $a_i = b_i = 0$；
- 若 s_i 不是首次出现的，那么在 $s[1, i-1]$ 中，找到所有字符的最后一次出现位置，从大到小记作 $j_1 > j_2 > \cdots > j_z$。设 $s_i = s_{j_x}$，由引理 4.1，可得 $t_{j_x} = t_i$，且 j_x 也是 $t[1, i-1]$ 中 t_{j_x} 的最后一次出现位置。所以 $a_i = b_i = x$。

再证充分性。设 s, t 具有相同的形状序列 a，则 $|s| = |a| = |t|$。

设 a 所有为 0 的位置为 $i_1, i_2, \ldots, i_z$。根据形状序列的构造方式可知，s_{i_j} 互不相同，t_{i_j} 也互不相同。因此存在置换 f，使得 $f(s_{i_j}) = t_{i_j}$。

以下利用数学归纳法证明：对于任意的 $1 \le i \le |s|$，上述 f 均满足 $f(s_i) = t_i$。

对于 $1 \le i \le |s|$，假设对于任意的 $1 \le j < i$，均有 $f(s_j) = t_j$。

- 若 $a_i = 0$，$f(s_i) = t_i$。

- 若 $a_i = x > 0$，那么在 $s[1, i-1]$ 中，找到所有字符的最后一次出现位置，从大到小记作 $j_1 > j_2 > \cdots > j_z$。应用归纳假设，$f(s_i) = f(s_{j_x}) = t_{j_x} = t_i$。

综上所述，$\forall 1 \le i \le |s|$，$f(s_i) = t_i$。所以 $s \sim t$。

□

为了求一个字符串 s 的形状序列 a，我们可以依次枚举 $i = 1, 2, \ldots, |s|$，同时维护 $last(\alpha)$ 表示字符 α 在 $s[1, i-1]$ 中最后一次出现的位置（不存在设为 0）。若 $last(s_i) = 0$，则 $a_i = 0$；否则 $last(s_i)$ 是所有 $last(\alpha)$（$\alpha \in \Sigma$）中第 a_i 大的。因此，一个字符串 s 的形状序列可以在 $O(k|s|)$ 的时间复杂度、$O(k)$ 的空间复杂度内求出。

4.2.2 形状序列对应的字符串数

接下来我们讨论，怎样的非负整数序列是一个字符串的形状序列：

定义 4.2. 对于非负整数序列 a，称 a 是一个合法形状序列，当且仅当 $n_0(a) \le k$，且 $\forall 1 \le i \le |a|$，均有 $a_i \le n_0(a[1, i-1])$。

由推论 4.1、性质 4.3 可得以下结论：

引理 4.2. 一个字符串的形状序列一定是合法形状序列。

接下来我们关注，如何从形状序列还原回字符串。形状序列丢失了信息，为了将这些信息补回，我们引入以下概念：

定义 4.3. 对于字符串 s，将其所含有的字符，按照第一次出现位置从左到右的顺序写下来，形成一个字符串 t，称为 s 的提示串。

由定义可得以下结论：

引理 4.3. 字符串 s 的提示串长度等于其不同字符个数。

引理 4.4. 字符串 s 的提示串的字符两两不同。

现在我们可以根据形状序列和提示串还原一个字符串了。

引理 4.5. 设 a 是合法形状序列，字符串 t 的字符两两不同，且 $|t| = n_0(a)$。则存在唯一字符串 s 使得其形状序列为 a，提示串为 t。

证明. 以下设 $|t| = n_0(a) = z$。我们利用数学归纳法证明。

当 $|a| = 0$ 时，可得 $|t| = 0$。有且只有 ϵ 满足条件，引理显然成立。

对于正整数 N，假设 $|a| = N - 1$ 的情况下引理成立。当 $|a| = N$ 时：

首先可得字符串 s 的长度为 N。根据性质 4.1，可得 $s[1, N-1]$ 的形状序列为 $a[1, N-1]$。

- 若 $a_N = 0$，则 s_N 在 s 中首次出现，可得 $s_N = t_z$，且 $s[1, N-1]$ 的提示串为 $t[1, z-1]$。对于 $a[1, N-1]$ 和 $t[1, z-1]$ 应用归纳假设，可得存在唯一字符串 $s[1, N-1]$ 使得其形状序列为 $a[1, N-1]$，提示串为 $t[1, z-1]$。因此引理成立。

- 若 $a_N = x \neq 0$，那么 s_N 在 s 中非首次出现，$s[1, N-1]$ 的提示串为 t。对于 $a[1, N-1]$ 和 t 应用归纳假设，可得存在唯一字符串 $s[1, N-1]$ 使得其形状序列为 $a[1, N-1]$，提示串为 t。

  根据性质 4.2，在 $s[1, N-1]$ 中，不同字符有 z 种。找到所有字符的最后一次出现位置，从大到小记作 $j_1 > j_2 > \cdots > j_z$。因为 a 是合法形状序列，可得 $x \leq z$。那么 $s_i = s_{j_x}$。因此引理成立。

综上所述，引理成立。 □

最后我们得出以下定理：

定理 4.2. 对于合法形状序列 a，恰有 $A_k^{n_0(a)}$ 个字符串的形状序列为 a。其中 A_n^m 表示排列数 $\frac{n!}{(n-m)!}$。

证明. 满足 $n_0(a) = |t|$ 的合法形状序列 a 与字符两两不同的字符串 t 所构成的全体二元组 (a, t) 集合记作 P。

根据性质 4.2、引理 4.2、引理 4.3、引理 4.4，可构造映射 $f : \Sigma^* \to P$，对于字符串 s，设其形状序列为 a，提示串为 t，令 $f(s) = (a, t)$。根据引理 4.5，f 是可逆映射。

现在已知合法形状序列 a，可得长度为 $n_0(a)$ 的、字符两两不同的字符串 t 恰有 $A_k^{n_0(a)}$ 个。因此恰有 $A_k^{n_0(a)}$ 个二元组 $(a, t) \in P$，所以恰有 $A_k^{n_0(a)}$ 个字符串使得其形状序列为 a。 □

4.2.3 形状序列与子串

前缀的形状序列与原串的形状序列的关系，已经在性质 4.1 中说明。

现在我们分析后缀的形状序列的性质。对于字符串 s 和整数 $1 \leq l \leq |s| + 1$，设 s 的形状序列为 a，$s[l, |s|]$ 的形状序列为 b。

依次考虑 $i = 1, 2, \ldots, |s| - l + 1$：

- 若字符 s_{l+i-1} 在 $s[1, |s|]$ 中首次出现，那么显然它也在 $s[l, |s|]$ 中首次出现，因此 $a_{l+i-1} = b_i = 0$；

- 若字符 s_{l+i-1} 在 $s[1, |s|]$ 中并非首次出现，那么在 $s[1, l+i-2]$ 中，找到所有字符的最后一次出现位置，从大到小记作 $j_1 > j_2 > \cdots > j_z$。设 $a_{l+i-1} = x$，即 s_{l+i-1} 上次出现位置为 j_x。

设 $n_0(b[1,i-1])=z'$，由性质 4.1 和性质 4.2，$s[l,l+i-2]$ 中有 z' 种不同字符。那么在 $s[l,l+i-2]$ 中，所有字符的最后一次出现位置从大到小为 $j_1>j_2>\cdots>j_{z'}$。所以若 $x>z'$，则 s_{l+i-1} 在 $s[l,|s|]$ 中首次出现，那么 $b_i=0$；否则 s_{l+i-1} 在 $s[l,|s|]$ 中并非首次出现，所以 $b_i=x$。

因此我们得到结论：

性质 4.4. 对于字符串 s 和整数 $1\le l\le |s|+1$，设 s 的形状序列为 a，$s[l,|s|]$ 的形状序列为 b。

对于 $1\le i\le |s|-l+1$，$b_i=\begin{cases}a_{l+i-1}, & a_{l+i-1}\le n_0(b[1,i-1]);\\ 0, & a_{l+i-1}>n_0(b[1,i-1]).\end{cases}$

根据该性质，我们可以遍历 $i=1,2,\ldots,|s|-l+1$ 递推确定 b_i，它只和形状序列 a 而不和原字符串 s 有关。我们把子串看作后缀的前缀，可作如下定义：

定义 4.4. 对于合法形状序列 a 和整数 $1\le l\le r\le |a|$，令 a 关于区间 $[l,r]$ 的子形状序列为一个长度为 $r-l+1$ 的序列 b，满足：

对于 $1\le i\le r-l+1$，$b_i=\begin{cases}a_{l+i-1}, & a_{l+i-1}\le n_0(b[1,i-1]);\\ 0, & a_{l+i-1}>n_0(b[1,i-1]).\end{cases}$

a 关于区间 $[l,r]$ 的子形状序列记作 $\text{subshape}(a,l,r)$。

特殊地，对于不在上述范围内的 l,r，$\text{subshape}(a,l,r)=\epsilon$。

对于整数 $1\le l\le |a|+1$，$\text{subshape}(a,l,|a|)$ 统称为 a 的后缀形状序列。

需要指出的是，子形状序列不同于子序列和连续子序列，它不仅要求截取连续的一段，还需要对越界的项进行处理。

由性质 4.1、性质 4.4 立得：

性质 4.5. 设字符串 s 的形状序列为 a，则子串 $s[l,r]$ 的形状序列为 $\text{subshape}(a,l,r)$。

现在讨论合法形状序列和其子形状序列的 0 的个数的关系。

性质 4.6. 对于合法形状序列 a 和整数 l,r，$n_0(\text{subshape}(a,l,r))\le n_0(a)$。

证明. 设 a 是字符串 s 的形状序列，则 $\text{subshape}(a,l,r)$ 是 $s[l,r]$ 的形状序列。

所以 $n_0(\text{subshape}(a,l,r))$ 为 $s[l,r]$ 中不同字符数，自然不超过 s 中不同字符数，即 $n_0(a)$。

□

4.3 算法二：$n=1$，$L\le 1000$

将 S_1 的所有后缀的形状序列插入 Trie。对最终 Trie 的每个结点，统计其代表的形状序列对应的字符串数。

时间复杂度 $O(L^2)$，空间复杂度 $O(kL^2)$，期望得分 9 分。

4.4 算法三：$n = 1$，$L \le 10^5$

算法三基于后缀数组 [3]。

沿用后缀数组的思想，我们试图求出任意两个后缀的形状序列的 LCP，并比较字典序大小。

记 S_1 的形状序列为 A，非空后缀 $S_1[l, L]$ 的形状序列为 $A^{(l)}$。设 $A^{(l)}$ 中所有为 0 的位置集合为 P_l，那么对于 $1 \le i \le L - l + 1$，$A_i^{(l)} = \begin{cases} A_{l+i-1} & i \notin P_l; \\ 0 & i \in P_l. \end{cases}$

注意到，$i \in P_l$ 当且仅当 $S_{1,l+i-1}$ 在 $S_1[l, L]$ 中第一次出现，可以通过简单的倒序递推求出。

因此，为求出 $\mathrm{LCP}\left(A^{(l)}, A^{(l')}\right)$ 并比较 $A^{(l)}$ 和 $A^{(l')}$ 的字典序大小，我们可以按照如下步骤进行：

1. 利用后缀数组预处理 A 的任意两个后缀的 LCP；
2. 预处理 $P_1, P_2, \ldots, P_L$；
3. 设 $P_l \cup P_{l'}$ 含有元素 $i_1 < i_2 < \cdots < i_z$，称它们为特殊点；
4. 对特殊点直接比较，对相邻两个特殊点间的连续子序列，利用后缀数组求它们的 LCP 后比较。

如果采用 $O(L)$ 时间预处理、$O(1)$ 时间查询的后缀数组算法，以上算法预处理时间复杂度 $O(Lk)$，单次查询时间复杂度 $O(k)$。

将 $A^{(1)}, A^{(2)}, \ldots, A^{(L)}$ 按照字典序从小到大排序，设排在第 i 位的形状序列是 $A^{(sa_i)}$，$A^{(i)}$ 排在第 $rank_i$ 位。记 $h_i = \mathrm{LCP}\left(A^{(sa_i)}, A^{(sa_{i+1})}\right)$（$1 \le i < L$）。根据后缀数组的相关结论，对于 S_1 的两个非空子串 $S_1[i, i + l - 1]$ 和 $S_1[j, j + l - 1]$（$i \ne j$），若不妨设 $rank_i < rank_j$，则它们相似当且仅当 $l \le \min\{h_x \mid rank_i \le x < rank_j\}$。

如果采用快速排序，由于单次比较时间为 $O(k)$，总时间为 $O(kL \log L)$，这是本算法的时间瓶颈。

现在我们可以开始统计了。先考虑怎么快速求一个合法形状序列的所有非空前缀对应的字符串数量。

对于合法形状序列 a，设它的所有 0 的位置从小到大为 $i_1 < i_2 < \cdots < i_z$。如果设 $i_{z+1} = |a| + 1$，那么 a 的所有长度位于区间 $[i_x, i_{x+1})$ 的前缀均含有 x 个 0，也就是对应 A_k^x 个字符串。因此它的所有非空前缀对应的字符串数量为：

$$\sum_{x=1}^{z} (i_{x+1} - i_x) A_k^x$$

记 $G(l,r)=|\{s\mid \exists r'\in[l,r], s\sim S_1[l,r']\}|$，它可以对 subshape$(A,l,r)$ 做上述算法求得，单次求算时间复杂度 $O(k)$。

现在我们着手解决原问题。

设 $T(i,j)$ 表示全体满足 $l\in\{sa_i,sa_{i+1},\ldots,sa_j\}$，$l\le r\le L$ 的子串 $s[l,r]$ 的形状序列集合，$F(i,j)$ 表示 $T(i,j)$ 总共对应的字符串数量。

当 $i=j$ 时 $F(i,j)=G(sa_i,L)$。

当 $i<j$ 时，设 $h_i,h_{i+1},\ldots,h_{j-1}$ 中最小的是 h_m（若最小值有多个，任取一个）。那么，$h_m=\mathrm{LCP}\{A^{(sa_x)}\mid i\le x\le j\}$。所以 $u\in T(i,m)\cap T(m+1,j)$ 当且仅当 u 是 $A^{(sa_i)}$ 的长度不超过 h_m 的非空前缀。换句话说，两部分形状序列的交集是 $T(sa_i,sa_i+h_m-1)$。因此，将两部分的贡献相加，再扣除重复贡献，我们可以写出如下的式子：

$$F(i,j)=F(i,m)+F(m+1,j)-G(sa_i,sa_i+h_m-1)$$

按照这个式子递归求解 $F(1,L)$ 即为答案。由于 $1\le m<L$ 作为最小值点只能出现在一次递归中，总共递归到的状态数是 $O(L)$ 级别的。

综上所述，这个算法时间复杂度 $O(kL\log L)$，空间复杂度 $O(kL)$，期望得分 20 分。

5 设计新算法解决一般情况下的问题

当 $n\ge 2$ 时，字符串相互影响较大，并且难以化归到单字符串的情况下求解，我们必须另寻出路。

我们考虑怎样的字符串是好的。

定义 5.1. 字符串 T 是 i–好的，当且仅当存在字符串 T 的一个划分 $T_iT_{i+1}\cdots T_n$，使得 $\forall i\le j\le n$，$T_j\in C_j$。特别地，字符串 T 是 $(n+1)$–好的，当且仅当 $T=\epsilon$。

引理 5.1. 对于 $1\le i\le n$ 和字符串 T，设 $T[1,r]$ 是 T 的最长前缀，使得 $T[1,r]\in C_i$。则 T 是 i–好的，当且仅当 $T[r+1,|T|]$ 是 $(i+1)$–好的。

证明. 充分性显然，只需证必要性。

假设 T 是 i–好的，相应划分为 $T_iT_{i+1}\cdots T_n$。

由 $T[1,r]$ 的最长性，得 $|T_i|\le r$。

对于子串 T_j（$i<j\le n$），若 $T_j=\epsilon$，则令 $T'_j=\epsilon$；否则若 T_j 对应的位置区间是 $[a,b]$，则令 $T'_j=T[\max\{a,r+1\},b]$，它为 T_j 的子串，由性质 3.2 及 $T_j\in C_j$ 自然可得 $T'_j\in C_j$。又 $T[r+1,|T|]=T'_{i+1}T'_{i+2}\ldots T'_n$，故 $T[r+1,|T|]$ 是 $(i+1)$–好的。 □

反复应用引理 5.1，可得以下定理：

定理 5.1. 对于字符串 T，设 $R_1 = T$。

对于 $1 \le i \le n$，取 R_i 的最长前缀 $R_i[1,r]$，使 $R_i[1,r] \in C_i$。令 $R_{i+1} = R_i[r+1, |R_i|]$。

则 T 是好的，当且仅当 $R_{n+1} = \epsilon$。

获取了一个字符串为好字符串的充要条件后，我们需要建立模型来描述上述定理，从而进行计数。

5.1 算法四：$k = 2$，$L \le 1000$

不难想到一种思路，即构造一个自动机使得其恰好接受好的字符串，随后利用递推对其接受的字符串数计数。

对于每个字符串 S_i，令部分 DFA $M_i = (Q_i, \Sigma, \delta_i, \bar{q}_i, Q_i)$ 为一棵 Trie。枚举 f，将 S_i 的每个后缀作用 f 后插入到 M_i 上。则 M_i 接受的字符串集合恰为 C_i。

我们现在考虑构造部分 DFA $M'_i = (Q'_i, \Sigma, \delta'_i, \bar{q}'_i, F'_i)$，使得其恰好接受所有 i–好的字符串。

首先构造 M'_{n+1}。这是容易的：$Q'_{n+1} = F'_{n+1} = \{\bar{q}'_{n+1}\}$，无转移。

对于 $i = n, n-1, \ldots, 1$，我们递推地构造 M'_i。根据引理 5.1，我们应该在自动机 M_i 中接受尽量长的前缀，并将剩余部分输入自动机 M'_{i+1} 中继续判定。

因此，构造如下：

- $Q'_i = F'_i = Q_i \cup Q'_{i+1}$
- $\bar{q}'_i = \bar{q}_i$
- 对于 $q \in Q'_{i+1}$ 及 $\alpha \in \Sigma$，$\delta'_i(q,\alpha) = \delta'_{i+1}(q,\alpha)$。
- 对于 $q \in Q_i$ 及 $\alpha \in \Sigma$，$\delta'_i(q,\alpha) = \begin{cases} \delta_i(q,\alpha), & \delta_i(q,\alpha) \neq 0; \\ \delta'_{i+1}(\bar{q}'_{i+1},\alpha), & \delta_i(q,\alpha) = 0. \end{cases}$

那么，自动机 M'_1 接受的就是所有好的字符串。

记 $f(u)$ 表示在 M'_1 中状态 u 接受的字符串数，进行递推。$\sum_{u \in Q'_1} f(u)$ 即为所求。

这个算法，总时空复杂度 $O(L^2)$，期望得分 15 分。

5.2 算法五：$L \le 1000$

根据之前的分析，形状序列是我们判定字符串相似的有力工具。我们试图从形状序列入手。

首先，我们将定义 5.1 与引理 5.1 改写到形状序列上。

定义 5.2. 合法形状序列 A 是 i–好的，当且仅当存在整数 $0 = p_{i-1} \le p_i \le \cdots \le p_n = |A|$，使得 $\forall i \le j \le n$，$\text{subshape}(A, p_{j-1}+1, p_j)$ 是 S_j 的至少一个子串的形状序列。特别地，合法形状序列 A 是 $(n+1)$–好的，当且仅当 $A = \epsilon$。

一个合法形状序列是好的，当且仅当它是 1–好的。

对照定义 5.1 与定义 5.2，可得：

引理 5.2. 一个字符串是 i–好的，当且仅当其合法形状序列是 i–好的。

引理 5.3. 对于 $1 \le i \le n$ 和合法形状序列 A，设 $A[1,r]$ 是 A 的最长前缀，使得 $A[1,r]$ 是 S_i 的至少一个子串的形状序列。则 A 是 i–好的，当且仅当 $\text{subshape}(A, r+1, |A|)$ 是 $(i+1)$–好的。

我们沿用算法四的思路，对于各字符串的所有子串的形状序列建立自动机。以下我们讨论的字符集均为 $\Sigma' = \{0, 1, \ldots, k\}$。

对于每个字符串 S_i，令部分 DFA $M_i = (Q_i, \Sigma', \delta_i, \bar{q}_i, Q_i)$ 为一棵 Trie，将字符串 S_i 的所有后缀的形状序列插入到 M_i 上。则 M_i 能接受所有 S_i 的子串的形状序列。记 z_q 表示状态 q 接受的合法形状序列中 0 的个数。

对于 $a > z_q$，令 $\delta_i(q,a) = \delta_i(q,0)$。那么根据定义 4.4，合法形状序列 A 的子形状序列 $\text{subshape}(A,l,r)$ 被自动机 M_i 接受，当且仅当其连续子序列 $A[l,r]$ 被自动机 M_i 接受。

构造部分 DFA $M'_i = (Q'_i, \Sigma', \delta'_i, \bar{q}'_i, F'_i)$ 以接受所有 i–好的合法形状序列，构造同算法四。那么一个合法形状序列是好的，当且仅当它被 M'_1 接受。但是，M'_1 接受的序列并不一定是合法形状序列。

现在我们着手统计答案。记 $f(q)$ 表示形状序列被自动机 M'_1 的状态 q 接受的字符串数。考虑 q 接受的所有合法形状序列，设为 $A_1, A_2, \ldots, A_m$。由性质 4.6，有 $z_q \le n_0(A_i)$。设 $n_0(A_i) = z$，对每个整数 $0 \le a \le z$，考虑接受合法形状序列 $A_i \cdot (a)$ 的状态及这一转移贡献的字符串数：

- $a = 0$，那么 $A_i \cdot (a)$ 被状态 $\delta'_1(q,0)$ 接受，字符串数：$A_k^{z+1} = (k-z)A_k^z$。
- $1 \le a \le z_q$，那么 $A_i \cdot (a)$ 被状态 $\delta'_1(q,a)$ 接受，字符串数：A_k^z。
- $z_q < a \le z$，那么 $A_i \cdot (a)$ 被状态 $\delta'_1(q,a) = \delta'_1(q,0)$ 接受，字符串数：A_k^z。

那么 A_i 对 $f(\delta'_1(q,0))$ 贡献了 $(k-z_q)A_k^{n_0(A_i)}$，对 $f(\delta'_1(q,a))$（$1 \le a \le z_q$）贡献了 $A_k^{n_0(A_i)}$。结合 $f(q) = \sum_{i=1}^m A_k^{n_0(A_i)}$，可得一个状态 q 对 $f(\delta'_1(q,0))$ 贡献了 $(k-z_q)f(q)$，对 $f(\delta'_1(q,a))$（$1 \le a \le z_q$）贡献了 $f(q)$。对于 $a > z_q$，转移 $\delta'_1(q,a)$ 的贡献已经在 $\delta'_1(q,0)$ 中计算过，可以删去。至此，我们可将自动机 M'_1 对应为一张加权有向无环图，对其进行带权路径计数。$\sum_{q \in Q'_1} f(q)$ 即为所求。

值得一提的是，在删去多余的转移后，M'_i 事实上等同于：

- $Q'_i = F'_i = Q_i \cup Q'_{i+1}$
- $\bar{q}'_i = \bar{q}_i$
- 对于 $q \in Q'_{i+1}$ 及 $0 \le a \le z_q$，$\delta'_i(q,a) = \delta'_{i+1}(q,a)$。
- 对于 $q \in Q_i$ 及 $0 \le a \le z_q$，$\delta'_i(q,a) = \begin{cases} \delta_i(q,a), & \delta_i(q,a) \neq 0; \\ \delta'_{i+1}(\bar{q}'_{i+1}, 0), & \delta_i(q,a) = 0. \end{cases}$

对于每个转移 $\delta_i(q,a) \neq 0$，若 $a = 0$，其权值为 $k - z_q$，否则其权值为 1。

这个算法，时空复杂度为 $O(kL^2)$，期望得分 39 分。

5.3 后缀自动机的扩展——子形状自动机

不难发现，算法五的劣势在于自动机的状态数过多。很多状态彼此等效，因此有巨大的优化空间。事实上，对于合法形状序列 A，本节将构造一种状态数与转移数均为 $O(k|A|)$ 的部分 DFA，使得其恰好接受 A 的所有子形状序列。

既然我们可以建立后缀自动机 [1, 4] 来接受一个字符串的其所有子串，联想一下，能否建立**子形状自动机**，使得其接受一个合法形状序列的所有子形状序列呢？

5.3.1 子形状自动机的定义及其大小

现在设 A 为某合法形状序列。对于 A 的子形状序列 B，设 B 在 A 中的右端点集合为 $R_A(B) = \{r | \text{subshape}(A, r - |B| + 1, r) = B\}$。特别地，令 $R_A(\epsilon) = \{0, 1, \ldots, |A|\}$。

若 A 的子形状序列 B, B' 满足 $|B| \le |B'|$，且 $R_A(B) \cap R_A(B') \neq \emptyset$，则 B 是 B' 的后缀形状序列，$R_A(B) \subseteq R_A(B')$。因此，所有子形状序列的右端点集合，若相同的只保留一个，则包含关系可构成一棵树。一个结点要么有至少两个子结点，要么是一个元素出现的最深的集合。因此，可得不同的右端点集合只有不超过 $2|A| + 1$ 种。

对于 A 的子形状序列 B, B'，我们说 $B \equiv_A B'$，当且仅当 $R_A(B) = R_A(B')$。显然 $\equiv_A$ 是一个等价关系。在 $\equiv_A$ 下，子形状序列 B 所在的等价类记为 $[B]_A$。它有以下性质：

引理 5.4. 若 B, C, B' 是非负整数序列，$BC, B'C$ 都是 A 的子形状序列，$B \equiv_A B'$，则 $BC \equiv_A B'C$。

推论 5.1. 若 B, C, B' 是非负整数序列，BC 是 A 的子形状序列，$B \equiv_A B'$ 且 $n_0(B) = n_0(B')$，则 $B'C$ 也是 A 的子形状序列，$BC \equiv_A B'C$ 且 $n_0(BC) = n_0(B'C)$。

引理 5.5. 若 A 的子形状序列 B, B' 满足 $B \equiv_A B'$，则 B 是 B' 的后缀形状序列，或者反之。

因此，存在部分 DFA $D_A = (Q, \Sigma', \delta, \bar{q}, Q)$，使得其每个状态均形如 $q = (E, z)$，其中 E 为 $\equiv_A$ 下的一个等价类，z 为一个非负整数，表示该状态接受 A 的子形状序列 B 当其仅当 $[B]_A = E$ 且 $n_0(B) = z$。对于 $q = (E, z)$，记 $E_q = E, z_q = z$。构造如下 [2, 引理 2]：

- 初始状态 $\bar{q} = ([\epsilon]_A, 0)$。
- 对于状态 q 和状态 q'，若存在序列 B 和非负整数 a 使得 B 被 q 接受，$B \cdot (a)$ 被 q' 接受，则 $\delta(q, a) = q'$。

这个部分 DFA D_A 称作形状序列 A 的子形状自动机。

由于不同的右端点集合数为 $O(|A|)$ 级别，0 的个数不超过 k，所以 D_A 的状态数为 $O(k|A|)$。这个级别是容易达到的，只需构造 $(0)^k(1)^{|A|-k}$。

以初始状态为根，取 D_A 对应的有向无环图的一棵外向树。对于图的每条非树边 $\langle u, v\rangle$，从根沿树边到 u、u 到 v、v 沿树边到任意一个叶结点的路径对应了 A 的至少一个后缀形状序列，且一个后缀形状序列与其转移路径的第一条非树边（如果存在）对应。所以非树边至多 $|A|$ 条。因此，D_A 的转移数与状态数同阶。

5.3.2 子形状自动机的构造算法

现在我们给出 D_A 的构造算法。

对于每个状态，定义其**代表元**为其接受的序列中最长的那一个。若 B 是 q 的代表元，则 q 接受的所有字符串均为 B 的后缀形状序列。

因此对于一个状态 q，设它的代表元为 B，则令 $len_q = |B|$。对于 B 的后缀形状序列 $B_l = \text{subshape}(B, l, |B|)$，$|R(B_l)|$ 随 l 非严格递增（由性质 4.5），$n_0(B_l)$ 随 l 非严格递减（由性质 4.6）。于是可得：存在非负整数 L 使得 q 接受了 B 的长度不小于 L 的所有后缀形状序列。对于 $q \neq \bar{q}$，$L \neq 0$，则定义状态 q 的**后缀连接** $parent_q$ 是接受 B 的长度为 $L - 1$ 的后缀形状序列的状态。特别地，记 $parent_{\bar{q}} = 0$。

定义状态 q 的**后缀链** $SC(q)$ 是指从 q 开始沿后缀连接跳转所经过的状态序列。形式化地，可递归定义：$SC(\bar{q}) = (\bar{q})$，$SC(q) = (q) \cdot SC(parent_q)$。那么，$SC(q)$ 接受了 q 的代表元的所有后缀形状序列。

考虑从空串开始，每次在末尾添加一个字符，并维护每个状态 q 对应的 $z_q, len_q, parent_q$。D_ϵ 是平凡的。

设序列 B、非负整数 a，使得 $B' = B \cdot (a)$ 为合法形状序列，假设现在已知 D_B，考虑在其基础上构造 $D_{B'}$。则新加入的字符串为 B' 的所有后缀形状序列。

对于整数 $1 \le l \le |B'|$ 有：

$$\text{subshape}(B', l, |B'|) = \begin{cases} B_l \cdot (a), & a \le n_0(B_l); \\ B_l \cdot (0), & a > n_0(B_l). \end{cases}$$

以下记 $tr(q,a)=\begin{cases}a, & a\le z_q;\\ 0, & a>z_q.\end{cases}$

于是，若 q 接受 B_l，则 subshape$(B',l,|B'|)$ 应被 $\delta(q,tr(q,a))$ 接受。

以下设在自动机 D_B 中，B 由状态 $sink_B=([B]_B,n_0(B))$ 接受。

增加一个字符的过程与一般的后缀自动机类似，不同点为：

- 对于 $sink_B$ 的后缀链上的状态 q，其对应转移的项为 $tr(q,a)$ 而不是 a。
- 对于 $sink_B$ 的后缀链上的状态 q、当前增加的状态 p，若 $z_q+[tr(q,a)=0]<z_p$，则需要按照后缀自动机的拆分算法来拆分 p。

明确起见，我们给出伪代码：

Algorithm 1 $update(B,a)$

Require: 当前自动机为 B 的子形状自动机，$B\cdot(a)$ 为合法形状序列

Ensure: 修改后自动机为 $B'=B\cdot(a)$ 的子形状自动机

```
q := sink_B
新建状态 p
sink_{B'} := p
z_p := z_q + [a = 0]
len_p := len_q + 1
while q ≠ 0 and δ(q, tr(q,a)) = 0 do
    if z_q + [tr(q,a) = 0] < z_p then
        p := split_new_node(p, q, tr(q,a))
    end if
    δ(q, tr(q,a)) := p
    q := parent_q
end while
if q = 0 then
    parent_p := q_0
    return
end if
r := δ(q, tr(q,a))
if len_r = len_q + 1 then
    parent_p := r
else
    parent_p := split_old_node(r, q, tr(q,a))
end if
```

在算法 1 中，我们用到了 $split\_new\_node(p,q,c)$ 表示从新状态 p 中拆分出从 $\delta(q,c)$ 转移而来的区间。该算法实现如下：

Algorithm 2 $split\_new\_node(p,q,c)$

新建状态 p'
$z_{p'} := z_q + [c = 0]$
$len_{p'} := len_q + 1$
$parent_p := p'$
return p'

在算法 1 中，我们用到了 $split\_old\_node(r,q,c)$ 表示从旧状态 r 中拆分出从 $\delta(q,c)$ 转移而来的区间。该算法实现如下：

Algorithm 3 $split\_old\_node(r,q,c)$

新建状态 r'
$z_{r'} := z_q + [c = 0]$
$len_{r'} := len_q + 1$
$parent_{r'} := parent_r$
$parent_r := r'$
for all $x \in \Sigma$ **do**
    $\delta(r', x) := \delta(r, x)$
end for
while $q \neq 0$ **and** $\delta(q,c) = r$ **do**
    $\delta(q,c) := r'$
    $q := parent_q$
end while
return r'

现在我们分析该算法的时间复杂度。注意到，如果使用数组保存各转移，空间复杂度为 $O(k^2|A|)$，任意单次操作时间为 $O(1)$。每次增量，至多拆分 k 个新状态，每次拆分需 $O(1)$ 时间；至多拆分一个旧状态，每次拆分中，复制信息和转移需 $O(k)$ 时间。故这些操作的总时间为 $O(k|A|)$。因此只需考虑新增或修改后缀链的转移的时间。

考虑 $|SC(sink_{B'})|$ 相对于 $|SC(sink_B)|$ 的变化。对于状态 $\bar{q} \neq p \in SC(sink_B)$，均存在 $q \in SC(sink_B)$，使得 $\delta(q, tr(q,a)) = p$。对于新增或修改后缀链的转移，其要么指向新状态，要么指向拆分出的旧状态，因此至多有 $k+1$ 种不同的后继状态。因此若新增或修改了后缀链上 t 个状态的转移，则有

$$|SC(sink_{B'})| \leq |SC(sink_B)| - t + k + 1$$

初始 $|SC(\bar{q})| = 1$，且 $|SC(sink_B)| > 0$ 恒成立。所以新增或修改后缀链的转移的总次数也为 $O(k|A|)$ 级别。

所以我们可得结论：

定理 5.2. 对于合法形状序列 A，可以在 $O(k|A|)$ 时间、$O(k^2|A|)$ 空间内，构造一种状态数与转移数均为 $O(k|A|)$ 的部分 *DFA*，使得其恰好接受 A 的所有子形状序列。

5.4 算法六：$k \le 5$

记 S_i 的形状序列为 A_i，M_i 为 A_i 的子形状自动机，其余部分同算法五。

优化后，M_1' 的状态数为 $O(\sum_{i=1}^{n} k|Q_i|) = O(kL)$，每个状态的有效转移数均为 $O(k)$，因此总时空复杂度 $O(k^2L)$，期望得分 70 分。结合算法三，期望得分 81 分。

5.5 合并相同转移

事实上，子形状自动机还有优化的空间。

在合法形状序列 A 的子形状自动机 $D_A = (Q, \Sigma', \delta, \bar{q}, Q)$ 中，若状态 $q \neq q'$ 但 $E_q = E_{q'}$，则对于 $1 \le a \le \min\{z_q, z_{q'}\}$，有 $E_{\delta(q,a)} = E_{\delta(q',a)}$。对于 $1 \le a \le z_q$，若 $\delta(q,a)$ 存在，则有 $z_{\delta(q,a)} = z_q$。

于是，对于等价类 E，若其中最长的序列为 B，我们定义 $\delta(E,a) = [B\cdot(a)]_A$。特别地，若 $B\cdot(a)$ 不是 A 的子形状序列，则 $\delta(E,a) = 0$。于是，对于 $1 \le a \le z_q$，均有 $\delta(q,a) = (\delta(E_q,a), z_q)$。因此，对于每个状态 $q = (E,z)$，都只需要额外记录 $\delta(q,0)$。

由于非空的等价类只有 $O(|A|)$ 个，可得只需要 $O(k|A|)$ 空间来记录转移，因此我们得到更强的结论：

定理 5.3. 对于合法形状序列 A，可以在 $O(k|A|)$ 时间、$O(k|A|)$ 空间内，构造一种状态数与转移数均为 $O(k|A|)$ 的部分 *DFA*，使得其恰好接受 A 的所有子形状序列。

5.6 算法七：无特殊限制

上述优化无法直接用于算法五中构造的 M_1'，考虑进一步优化。仍然设 $f(q)$ 表示在自动机 M_1' 对应的加权有向无环图中，从 $\bar{q}_1'$ 到 q 的所有路径上，各边权值积之和。

现在我们放弃显式建立自动机 M_1'。对于 $\equiv_{A_i}$ 的等价类 E，统一考虑所有状态 (E,z) 在递推中对后继状态的贡献。

在原自动机中，转移只有 $O(kL)$ 条，这部分转移可以直接枚举。对于新加入的转移，设 M_i 的状态 $q = (E,z)$ 和整数 $0 \le a \le z$ 使得 $M_i(q,a) = 0$，对于这样的转移，我们有：

- 若 $a = 0$，则枚举状态直接转移，总共有 $O(kL)$ 条转移。

- 若 $1 \le a \le z$，这意味着 $\delta_i(E_q, a) = 0$，则 q 对 $f(\delta'_{i+1}(\bar{q}'_{i+1}, 0))$ 贡献了 $f(q)$。可以换一个角度看：对于每个等价类 E，记 m_E 和 M_E 分别表示使得状态 (E, z) 存在的 z 的最小值和最大值。所有状态 (E, z) 总共对 $f(\delta'_{i+1}(q'_{i+1}, 0))$ 贡献了 $\sum_{z=\max\{m_E,a\}}^{M_E} f((E, z))$，这可以用后缀和快速计算。所以这部分的时间复杂度优化到了 $O(kL)$。

这样，这个算法的总时空复杂度降低到 $O(kL)$，期望得分 100 分。

6 讨论与扩展

6.1 子形状自动机的优点

子形状自动机在处理子串的相似关系时，较基于后缀数组的算法（见算法三），构建的时间复杂度更低，空间复杂度不变。由于自动机的结构更加直观，在面对一些问题如本题时，容易设计出优秀的算法；而本题特意抓住后缀数组算法的弱点来命制，暴露出了后缀数组或后缀树在处理字符串匹配问题的场景中信息大量压缩、不够直观的问题。

6.2 子形状自动机的局限性

子形状自动机的各项复杂度仍然与字符集大小有关，难以处理字符集较大的情况。不过，基于后缀数组或后缀树的算法同样无法处理字符集较大的情况。

6.3 子形状自动机在其它一类关系下的应用

本题中，从字符串对应到形状序列，可以用自动机变换 [2][3] $M : \Sigma^* \to (\Sigma')^*$ 描述。

$M : \Sigma^* \to (\Sigma')^*$ 是**自动机变换**，当且仅当存在四元组 $(Q, M_Q, M_{\Sigma'}, \bar{q})$，其中，$Q$ 是一个有限的状态集合，$M_Q : Q \times \Sigma \to Q$ 是转移函数，$M_{\Sigma'} : Q \times \Sigma \to \Sigma'$ 是变换函数，$\bar{q} \in Q$ 是初始状态，满足 $\forall s \in \Sigma^*$，存在状态序列 $(q_0, \ldots, q_{|s|})$，使得 $q_0 = \bar{q}$，$q_i = M_Q(q_{i-1}, s_i)$，$M(s)_i = M_{\Sigma'}(q_{i-1}, s_i)$，且 $|M(s)| = |s|$。其中使得 $|Q|$ 最小的四元组 $(Q, M_Q, M_{\Sigma'}, \bar{q})$ 称为 M 的**最小自动机** [2, 引理 2]。

本题主要利用了等价关系 $\sim$ 的下列性质：

- 存在自动机变换 $M : \Sigma^* \to (\Sigma')^*$，使得 $\forall s, t \in \Sigma^*$，$s \sim t \iff M(s) = M(t)$。仍然称 $M(s)$ 为 s 的形状序列，如果 $a \in \mathrm{Im}M$，称 a 为合法形状序列。

- 一个合法形状序列的后缀，可以按照长度分成 K 个区间，每个区间均被 M 的最小自动机的同一个状态接受。

[3] 这里与原文的定义略有区别。

对于满足这些性质的等价关系，利用本文的算法，均能在 $O(K|a|)$ 时间、$O(K|a||\Sigma'|)$ 空间内建立合法形状序列 a 对应的子形状自动机，其状态数、转移数均为 $O(K|a|)$ 级别，从而解决这类子串等价关系问题。如果 $|\Sigma'|$ 较大，还可以利用可持久化线段树保存转移，时空复杂度均为 $O(K|a|\log|\Sigma'|)$。

这类性质不但包括本题中的相似，还包括例如：

例 6.1. 在字符集 Σ 上定义一个全序 $\leq$，对于 $s,t\in\Sigma^*$，$s\sim t$ 当且仅当 $|s|=|t|$ 且 $\forall 1\leq i,j\leq|s|$，$[s_i\leq s_j]=[t_i\leq t_j]$。

解. 对于字符串 s，其形状序列为非负整数序列 a，使得 $|a|=|s|$。设在 $s[1,i]$ 中出现的各字母中，s_i 为第 x 小的。若 s_i 在 $s[1,i-1]$ 中出现过，则 $a_i=2x$，否则 $a_i=2x-1$。$\Sigma'=\{1,2,\ldots,2k-1\}$，$K=|\Sigma|+1$。

例 6.2. Σ 是 M 以内全体正整数集，对于 $s,t\in\Sigma^*$，$s\sim t$ 当且仅当 $|s|=|t|$ 且 $\forall 1\leq i\leq|s|$，$s_i/\gcd(s_1,s_2,\ldots,s_i)=t_i/\gcd(t_1,t_2,\ldots,t_i)$。

解. 对于序列 $s\in\Sigma^*$，其形状序列为非负整数序列 a，使得 $|a|=|s|$，$a_i=s_i/\gcd(s_1,s_2,\ldots,s_i)$，$\Sigma'=\{1,2,\ldots,M\}$，$K=\lfloor\log_2 M\rfloor+1$。

7 总结

7.1 命题思路

在 2019 年由日本经济新闻社与 AtCoder 共同主办的第二届全日统一编程之王决赛中，出现了一道新颖的试题。该题的研究对象是一类与其反串相似（定义同本题）的字符串，参考解法通过巧妙地扩展 Manacher 算法，并对这种情况下的 Manacher 算法的时间复杂度进行仔细的分析，最终得到了一种高效而简洁的解法。作者学习该题时很受启发，进一步地对类似问题进行了扩展研究。

于是，同一个字符串的两个子串的相似关系首先进入了作者的视线。针对此类问题，以往有一种基于后缀数组的算法。而作者对此类问题下后缀自动机的扩展应用展开了研究，通过引入辅助序列描述等价类、将转移不同的等价类分开，发现了在该类问题上相较于后缀数组更高效、更直观、更简便的数据结构——子形状自动机。

为了进一步体现子形状自动机建图直观的优势，作者将 2017 年山东一轮集训《字符串》[4]与子形状自动机结合，命制了《太阳神的宴会》一题。

本题在部分分的设计上，覆盖了单串情形的经典算法，选手思考单串情形，可以发现形状序列这一重要工具。在多串情形下建立自动机，对选手的知识和模型积累提出了一定要求，作者对不同层次的算法给出了多样的部分分，引导选手往正解的方向思考。

[4]https://loj.ac/p/6071

7.2 算法扩展

本文探究了一类基于自动机变换的等价关系，针对这类等价关系提出了子形状自动机及其构造算法，直观、快速、简便地处理了本质不等价子串的有关问题。在题中相似关系的特殊情形下，还通过合并相同转移节省了空间，在这一问题上取得了优于原有算法的进展。我希望《太阳神的宴会》这道题能够带给大家更多关于子形状自动机以及字符串的等价类问题的思考。

致谢

感谢父母的养育之恩。

感谢中国计算机学会给予的交流机会。

感谢福州一中陈颖老师的关心与指导。

感谢清华大学陈俊锟学长、钟知闲学长、吴作同学长的关心与指导。

感谢清华大学陈俊锟学长、福州一中林昊翰同学和福州一中信息学竞赛组其他同学为本文审稿。

感谢 Universal OJ 为本次自主互测提供平台。

感谢互测的参赛选手参与本题的做题情况调查。

感谢各位的阅读。

参考文献

[1] A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M.T. Chen, and J. Seiferas. The smallest automation recognizing the subwords of a text. Theoretical Computer Science, pages 31–55, 1985.

[2] A. Nerode. Linear automaton transformations. Proceedings of The American Mathematical Society, pages 541–541, 1958.

[3] 刘汝佳、陈锋. 算法竞赛入门经典, pages 219–221. 清华大学出版社, 2012.

[4] 张天扬. 后缀自动机及其应用. In 2015 年信息学奥林匹克中国国家队候选队员论文集, 2015.