Stringology

郭暁旭 (@ftiasch)

2019 年 5 月 25 日

ICPCCamp 2015. Sequence Sorting

给 n 个串 $S_1, S_2, \ldots, S_n$, 按照字典序排序输出。

- $n \leq 10^6$
- $\sum_{i=1}^{n} |S_i| \leq 5 \times 10^6$
- 对于任意 i, 满足 $1 \leq S_{i,1} \leq S_{i,2} \leq \cdots \leq S_{i,|S_i|} \leq 5 \times 10^6$

我们同时把 n 个串插入 Trie，按照字符从小到大的顺序。因为串是递增的，所以这样是一致的。

对于每个 Trie 的节点，记录上一次经过的边，每次经过节点时，要么使用上一条要么新开一条。

复杂度关于串长度和字符集大小线性。

ICPCCamp 2016. Aho–Corasick Automaton

给字典树 T, 对于节点 $u \in T$, 定义 $s(u)$ 表示根到 u 路径上的串。对于每个 u, 求 $\mathrm{fail}(u)$ 表示最长的 $s(v)$ 是 $s(u)$ 的后缀。

$|T|, |\Sigma| \le 2 \times 10^5$

算法 0

直接调用 Aho-Corasick 自动机的构造算法，复杂度是 $O(n^2)$，或者 $O(|\Sigma|n)$.

最坏情况：考虑 $0^{10^5}1$，$0^{10^5}2$，$0^{10^5}3$，... 组成的 Trie 树，共有 2×10^5 个点。

算法 1

我们注意到构造后缀数组的倍增算法也适用于 Trie 树，所以可以 $O(n \log n)$ 对 $s(u)$ 排序。

假设节点 $v_1, v_2, \ldots, v_n$ 满足 $s(v_1) \leq s(v_2) \leq \cdots \leq s(v_n)$，我们需要对于每个 $s(v_i)$ 求出最大的 $j < i$ 满足 $\mathrm{LCP}(s(v_j), s(v_i)) = |s(v_j)|$（还有一个对称的情况，在此略去）。

我们从小到大枚举 i，维护集合 S 是所有满足 $\mathrm{LCP}(s(v_j), s(v_i)) = |s(v_j)|$ 的 j. 当 i 变化到 $(i+1)$ 时，集合 S 中所有不满足 $s(v_j) \leq \mathrm{LCP}(s(v_i), s(v_{i+1}))$ 的 j 需要被删除。

算法 2

我们设 $\mathrm{go}(u, c)$ 是节点 u 走字符 c 后到达的节点。假设点 u 的儿子是 $v_1, v_2, \ldots, v_k$, 对应的字符是 $c_1, c_2, \ldots, c_k$. 那么 $\mathrm{fail}(v_i) = \mathrm{go}(u, c_i)$.

$$\mathrm{go}(u, c) = \begin{cases} v_j & \exists c_j = c \\ \mathrm{go}(\mathrm{fail}(u), c) & \text{otherwise} \end{cases}$$

如果用持久化数组（线段树）维护 $\mathrm{go}(u, *)$, 那么 $\mathrm{go}(u, *)$ 可以通过 $\mathrm{go}(\mathrm{fail}(u), *)$ 修改 $O(n)$ 次得到。

Chengdu 2013. GRE Words Revenge [1]

维护一个串集合 P, 支持两种操作:

- 在 P 中插入串 p
- 询问 T 有多少子串在 P 中。

$|P| \leq 10^5, |T| \leq 5 \times 10^6$

[1] http://acm.hdu.edu.cn/showproblem.php?pid=4787

算法 1

维护 2 个 AC 自动机 SMALL 和 BIG. 每次把串 p 插入到 SMALL 中，如果 $\mathrm{SMALL} > \sqrt{|P|}$，则把 SMALL 合并进 BIG 中。
显然，插入 SMALL 的单次代价是 $O(\sqrt{|P|})$，合并 BIG 的次数是 $O(\sqrt{P})$. 总复杂度是 $O(|P|\sqrt{|P|} + |T|)$.

算法 2

维护 $O(\log|P|)$ 个 AC 自动机 $A_0, A_1, \ldots$，满足 $|A_i| < 2^{i+1}$.

假设 k 满足 $2^k \le |p| < 2^{k+1}$，将串 p 插入到 A_k 中，并重建 A_k。重建的复杂度是 $2^{k+1} + |p| \le 3|p|$. 之后 A_k 的大小可能不满足条件，此时有 $2^{k+1} \le |A_k| < 2^{k+2}$.

我们设势能 $\Phi = -\sum_{i=0}^{\infty} 4i|A_i|$. 如果 A_k 大小不满足条件，就把 A_k 的所有串加入 A_{k+1}，重建 A_{k+1}. 这里需要 $|A_k| + |A_{k+1}| \le 2^{k+3}$ 次操作，同时势能的减小是 $4|A_k| \ge 4 \cdot 2^{k+1} = 2^{k+3}$，即操作次数不超过势能的减小量。

而最终势能 $\Phi_{\text{final}} = -O(|P|\log|P|)$，总的复杂度是 $O((|P| + |T|)\log|P|)$.

Beijing 2014. GRE Words Once More! [2]

给定 n 个点 m 条边的有向无环自动机，q 个 k_i 询问被接受的串中字典序第 k_i 小的串的长度。

$n, m, q \leq 10^5, q \leq 10^8$

[2] http://acm.hdu.edu.cn/showproblem.php?pid=5118

设 $\text{ways}(u)$ 表示从点 u 出发被接受的串的总数，我们可以递推求出。不妨设 $\max\{\text{ways}(u)\} \le 10^8 = M$.

对于点 u，设 $\text{prefer}(u)$ 是点 u 的所有后继中，$\text{ways}(v)$ 最大的后继 v. 如果我们连边 $u \to \text{prefer}(u)$, 那么这些边形成了一个有向森林 $\mathcal{F}$。我们称呼这为有向无环图的轻重链剖分。

假设点 u 在 $\mathcal{F}$ 中到根的路径是 $u_0, u_1, \ldots, u_k$. 对于点 u_i 的某个后继 $v \neq u_{i+1}$, 我们有 $2\text{ways}(v) \le \text{ways}(u_i) \le \text{ways}(u_0) = \text{ways}(u)$. 即，类似于树的情况，如果走一条非 $\mathcal{F}$ 的边，ways 至少对折。

如果我们对于每个 u，能够 $O(\log n)$ 确定偏离它到根路径的非 $\mathcal{F}$ 的边，我们可以 $O(\log n \log M)$ 回答询问。

实际上，因为 $\mathcal{F}$ 是森林，在树上倍增即可。需要注意的细节是重链把后继分成两个部分，需要分别处理。

Substrings[3]

令 $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. 串 $u, v \in \Sigma^k$ 同构，当且仅当存在单射 $\phi : \Sigma \to \Sigma$ 满足 $\phi(u_i) = v_i$.

给出串 $s \in \Sigma^n$, 问 s 的**不**同构的子串数量。

$n \leq 5 \times 10^4$

[3] https://ac.nowcoder.com/acm/contest/139/I

解法 1

对于串 u, 我们取字典序最小的 $\phi(u)$ 作为 u 的代表。

我们注意到，对于 u 的前缀 v, $\phi(v)$ 也是 v 的代表。

对于每个后缀，取代表。问题转化为：给定 n 个串，求它们不同前缀的数量。这是经典问题，只需要按照字典序排序，答案是总长度减去相邻两串的 LCP.

实现上，因为 ϕ 只有 6 种，我们把 s 的 6 种变换结果连接，求后缀数组。

概念上，很容易得到每个后缀的代表，也很容易计算 LCP.

实际上，排序和求解 RMQ (i.e., LCP) 是不必要的，因为遍历后缀数组时已经排序好了。

解法 2

类似解法 1，我们把 s 的 6 种变换结果连接，求**不同的子串数量**，记为 A.

我们注意到：

- 对于字符集为 1 的串，A 中贡献为 3;
- 对于字符集为 2 的串，A 中贡献为 6;
- 对于字符集为 3 的串，A 中贡献为 6.

因此，我们只需要求出最长的字符集为 1 的串的长度 l，$\frac{A+3l}{6}$ 即是答案。

解法 3

下面讨论 $O(|\Sigma| n \log n)$ 的算法。

我们换用另外的方法 $\mathrm{enc}(u)$ 来表示串 u.

$\mathrm{enc}(u)_i$ 是最小的 $j > 0$ 满足 $u_i = u_{i-j}$. 即 $\mathrm{enc}(u)_i$ 是 i 与上一个等于 u_i 的字符的距离。容易验证，$\mathrm{enc}(u) = \mathrm{enc}(v)$ 等价于 u, v 同构。

$\mathrm{enc}(s)[i:]$ 和 $\mathrm{enc}(s[i:])$ 会有 $|\Sigma|$ 个位置不同。

如果预处理 $\mathrm{enc}(s)$ 的后缀数组，我们可以 $O(|\Sigma|)$ 求 $\mathrm{enc}(s[i:])$ 和 $\mathrm{enc}(s[j:])$ 的 `LCP`.

`std::sort` 排序得到 $O(|\Sigma| n \log n)$.

解法 4

下面讨论 $|\Sigma| = O(n)$ 的情形，会得到 $O(n\log^3 n)$ 的算法。

我们维护一个可持久化数组（线段树）aux. 从后往前考虑字符 u_i, 每次找到最小的 $j > 0$ 满足 $u_i = u_{i+j}$, 把 $\mathrm{aux}[i+j]$ 修改为 j.

那么对于后缀 $s[i:]$，假设考虑 u_i 后的数组版本是 aux_i, 那么

$$\mathrm{enc}(s[i:]) = \mathrm{aux}_i[i:].$$

因为 $\mathrm{enc}(s[i:])$ 对应于可持久化线段树某个版本的某个区间，如果在线段树上维护节点的 Rabin Karp Hash 值，我们可以 $O(\log n)$ 得到 $\mathrm{enc}(s[i:])$ 某个前缀的 RK Hash.

如果我们二分 `LCP`，可以在 $O(\log^2 n)$ 比较两个串的字典序。

`std::sort` 排序得到 $O(n\log^3 n)$.

Codeforces 530F. Zhe-function

对于串 u, 定义 $\mathrm{Zhe}(u) = \sum_{i=1}^{|u|} \mathrm{LCP}(u, u[i:])$.

给出串 s, q 个 $[l_i, r_i]$ 询问 $\mathrm{Zhe}(s[l_i..r_i])$ 的值。

$|s|, q \leq 2 \times 10^5$

$[l, r]$ 要询问

$$\sum_{i=l}^{r} \min\{\mathrm{LCP}(s[l:], s[i:]), r-i+1\}.$$

建立 s 的后缀树，并做轻重链剖分。我们枚举 l 到根路径上的点 p，尝试计算和 l 的 `LCA` 恰好是 p 的点 i 的值。分为两种情况：

1. 点 l 在 p 的轻儿子中
2. 点 l 在 p 的重儿子中

只有 $O(\log n)$ 个 p 满足情况 1。所有满足 $\mathrm{LCA}(l, i) = p$ 的点 i 落在点 p 不包含点 l 的子树中，且有 $\mathrm{LCP}(s[l:], s[i:])$ 是定值，记为 l_p.

这时相当于要求子树中满足 $l \le i \le r - l_p + 1$ 的点 i 数量，以及满足 $r - l_p + 1 < i$ 的 $(r - i + 1)$ 的和。只要在 `DFS` 序上建立主席树，即可 $O(\log n)$ 完成查询。

情况 1 的总复杂度是 $O(q \log^2 n)$.

对于情况 2，点 i 来自点 p 的轻儿子。对于某个 p 和某条重链，满足情况 2 的点 p 是重链的一个前缀，总共要询问 $O(q \log n)$ 个前缀。

考虑某条重链，按照前缀长度从小到大处理所有询问。

我们维护一个集合 S，每次加入点 p 时，对于点 p 的所有轻儿子子树中的点 i，把 (i, l_i) 加入 S，其中 $l_i = \mathrm{LCP}(s[l:], s[i:]) = l_p$. 因为任意点 v 只属于 $O(\log n)$ 个轻子树，所以对于所有重链，S 的总大小是 $O(n \log n)$.

我们的 S 需要支持以下两种询问:

1. 统计满足 $l \le i \le r$ 的 l_i 的和
2. 统计满足 $l \le i \le r$ 且 $i + l_i > r$ 的 $(r - i + 1 - l_i)$ 的和.

第 1 种询问只需维护 i 的线段树。对于第 2 种询问，需要 $O(\log^2 n)$ 的复杂度。

但是，我们注意到，如果我们可以通过减小询问前缀的长度，从而只需要 $l_i \le r - l + 1$ 的 i. 那么 $i \le r < i + l_i$ 蕴含了 $i \le l$. 因此，我们可以每次插入区间 $[i, i + l_i)$，询问 r，也是 $O(\log n)$.

Petrozavodsk Winter 2016. Deep Purple [4]

给定串 s, q 个 $[l_i, r_i]$ 询问最大的 $0 \le x \le r_i - l_i$ 满足 $s[l_i..l_i + x - 1] = s[r_i - x + 1..r_i]$.

$n, q \le 2 \times 10^5$

[4] https://codeforces.com/gym/100962

$[l, r]$ 要询问最小的 i 满足 $i > l$ 且 $i + \mathrm{LCP}(s[l:], s[i:]) > r$. 注意如果对应的 $i > r$, 那么答案为 0.

类似于上题，仍然考虑 l 后缀树上的祖先 p. 仍然假设 $l_p \le r - l + 1$, $i + \mathrm{LCP}(s[l:], s[i:]) > r$ 蕴含了 $i > l$.

如果 l 在 p 的轻儿子中，那么要在 p 的子树中询问最小的 $i > r - l_p$. 把询问按照 $(r - l_p)$ 离线。处理到 i 时，把 i 到根的路径都设上 i，询问时单点查询。

如果 l 在 p 的重儿子中，把 i 按照 $(i + i_p)$ 离线，询问时区间询问。

Atcoder Regular Round 058 F. 文字列大好きいろはちゃん

给定 n 个串 $s_1, s_2, \dots, s_n$，**顺次**选择若干个串拼接，问总长为 k 的串中，字典序最小的串。

$n \leq 2000, k \leq 10^4, \sum_{i=1}^{n} |s_i| \leq 10^6$

算法 1

预处理 $\text{possible}(i, j)$ 表示 $s_i, s_{i+1}, \ldots, s_n$ 是否能够拼出长度为 j 的串。
从左到右构造答案，设 $F(i, j)$ 表示 $s_1, s_2, \ldots, s_i$ 拼出的长度为 j 的字典序最小的串。假设其中最长存在的串长度是 j^*，那么对于 $j < j^*$，$F(i, j)$ 是 $F(i, j^*)$ 的前缀。所以只需要记录 $F(i, j^*)$ 的串，和其他合法的串的长度即可。
从 $(i-1)$ 转移到 i 时，如果 $F(i-1, j)$ 存在，有以下两种转移：

- 如果 $\text{possible}(i+1, k-j)$，可以用 $F(i-1, j)$ 更新 $F(i, j)$
- 如果 $\text{possible}(i+1, k-j-|s_i|)$，可以用 $F(i-1, j) + s_i$ 更新 $F(i, j+|s_i|)$.

下面考虑如何从 $F(i-1, *)$ 更新到 $F(i, *)$. 假设 $F(i-1, *)$ 中最长的串是 T，那么参与更新的串是 $T[1..j]$ 或者 $T[1..j] + s_i$. 为了快速比较字典序，扩展 KMP 预处理 T、s_i 的任意后缀和 s_i 的 LCP.
总复杂度是 $O(nk + \sum_{i=1}^{n} |s_i|)$.

算法 2（semiexp 野蛮做法）

先考虑一般的情况，求 NFA （带 ϵ 转移的有限状态自动机）可接受的字典序最小的串。如果假设每个状态 s 出发都有可接受的串，方法是维护一个集合 S，表示当前可能的状态，每次选择一个 a 使得 $\{\delta(s,a) : s \in S\}$ 非空进行转移。

回到原问题，为了方便说话，我们先用一个新字符 $*$，把 s_i 连接，得到长度为 $\sum_{i=1}^{n} |s_i| + n$ 的大字符串 S. 我们的 NFA 有 $O(|S| \times k)$ 个状态。有两种转移:

- 对于 $s_i \neq *$ 的 i，有 $(i,j) \to (i+1, j+1)$，标号为 s_i.
- 对于 $s_i = *$ 的 i，有 $(i,j) \to (k,j)$，标号为 ϵ，其中 $k > i$ 且 $s_{k-1} = *$ （即 i 是字符串的结尾，k 是字符串的开头）.

我们注意到，对于 $s_i \neq *$ 的 i，其转移唯一。所以我们不需要对于每个 (i, j) 计算是否存在由它出发的可接受的串。我们只需要对于 $s_{i-1} = *$ 的 i, 不访问 $\mathrm{possible}(i, k-j)$ 不成立的 (i, j) 即可。

我们按照 j 递增构造答案，使用 `std::bitset` S 来存储可达的 $(*, j)$ 状态 (即上文提到的 S)。除了有 $O(nk)$ 个 ϵ 需要手工处理，其他的只需要从小到大枚举 s，设 S_a 表示所有 a 字符出现的下标，只要（`S >> 1`）`& S_a` 非空，就可以把 S 转移到（`S >> 1`）`& S_a`.

这样做的复杂度是 $O(k \cdot \frac{|S|}{64} \cdot |\Sigma|)$. 如果我们把 S_a 组织成线段树，我们可以在线段树上二分，复杂度变成 $O(k \cdot \frac{|S|}{64} \cdot \log |\Sigma|)$.

ICPCCamp 2015. Knuth–Morris–Pratt

对于串 s, 定义 $\mathrm{next}_i = \max\{j : 0 \le j < i \wedge s[1..j] = s[i-j+1..j]\}$.

给出 $\mathrm{next}_1, \mathrm{next}_2, \ldots, \mathrm{next}_n$ 的值，问满足条件的长度为 n，字符集为 Σ 的串 s 的数量，模 (10^9+7).

$n \le 2 \times 10^5, |\Sigma| \le 10^9$

考虑 KMP 算法的过程，实际上只进行了 $O(n)$ 次比较。只要满足这 $O(n)$ 个相等 / 不等关系，next 数组就相同。

连边 $i \to \mathrm{next}_{i-1} + 1$, 可以得到森林 $\mathcal{F}$. 对于 i, 根据 next 的计算过程，我们知道 next_i 也是 i 在 $\mathcal{F}$ 上的祖先。同时，跟 i 的限制有两种:

1. $s_i = s_{\mathrm{next}_i}$
2. 对于在 (i, next_i) 路径上的 j, $s_i \neq s_j$.

如果我们把字符按照第 1 类限制归类，$\mathcal{F}$ 会被划分成 k 个子集 $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$. 可以验证，如果 $\mathcal{T}_i$ 和 $\mathcal{T}_j$ 的 Steiner Tree 在树上有交，跟第 2 类限制是等价的。

换句话说，我们需要给子集 $|\Sigma|$ 染色，保证相交的子集颜色不同。

我们可以按照 LCA 的深度从小到大给子集染色。可以验证，当给子集 $\mathcal{T}_i$ 染色时，所有与它相交的、染过色的子集都彼此相交于 $\mathcal{T}_i$ 的 LCA。所以，这些子集两两不同色，$\mathcal{T}_i$ 可行的色数唯一确定。

更一般地，**子树的相交图**就是 **弦图**，同时 LCA 深度从大到小的顺序就是对应弦图的完美消除序列。上面实际上描述了求弦图 $|\Sigma|$ 染色数的过程。

XV New Year Contest. Automaton [5]

[5] https://contest.yandex.com/newyear2019/contest/11451/enter/

CTSC 2016. 萨菲克斯·阿瑞 [6]

[6] https://loj.ac/problem/2988