IOI2020 中国国家集训队第一阶段作业 解题报告

长沙市长郡中学 周书予

1 New Language

1.1 试题来源

Codeforces Round #315 (Div. 1) C

1.2 试题大意

给出一个长度为 n 字符集大小为 l 的字符串 S，要求构造一个字典序不小于 S 且最小的合法串 T。一个串 T 合法当且仅当满足给出的 m 组形如 (pos_1, t_1, pos_2, t_2) 的限制：当 $T_{pos_1} \in t_1$ 时，$T_{pos_2} \in t_2$，其中 $t_1, t_2 \in \{V, C\}$，V, C 是给定的字符集合，满足 $V \wedge C = \emptyset, V \vee C$ 为前 l 个小写字母组成的集合。

1.3 数据范围

$1 \le n \le 200$
$1 \le m \le 4n(n-1)$
$1 \le l \le 26$

1.4 时空限制

2 seconds, 256 megabytes

1.5 算法介绍

形如 $p \to q$ 的限制很容易让人联想到使用 2-sat 模型解决此问题。对于构造出的答案串 T 的每个位置 i，建立两个点表示两个命题 $T_i \in V$ 和 $T_i \in C$，对于一组给出的限制 (pos_1, t_1, pos_2, t_2)，建边 $(T_{pos_1} \in t_1) \to (T_{pos_2} \in t_2)$ 以及 $(T_{pos_2} \notin t_2) \to (T_{pos_1} \notin t_1)$（逆否命题），使用经典的 2-sat 问题解法即可实现判定是否有解，时间复杂度为 $O(n+m)$。

这里有别于经典 2-sat 问题的方面在于需要构造一组字典序不小于 S 且最小的合法解 T。可以先从大到小枚举 T 与 S 的最长公共前缀长度，记之为 len，在 T_{len+1} 位上依次填入大于 S_{len+1} 的字符后判断是否有解，一旦发现有解则说明确定了 T 与 S 的最长公共前缀 len 以及 T_{len+1} 位的取值，此时大于 $len+1$ 部分的 T_i 均可以任意取，从前往后枚举每一位上填入什么字符并判断有解即可。

上述做法的时间复杂度是 $O(nml)$，注意到可以只枚举满足限制条件的最小 V 集合元素和最小 C 集合元素，即可将时间复杂度优化到 $O(nm)$。

注意当 V, C 中一者为空集时需要对命题进行特殊处理。

2 Forensic Examination

2.1 试题来源

Codeforces Round #349 (Div. 1) E

2.2 试题大意

给出一个模式串 S 以及 m 个匹配串 T_i, q 组询问, 每组询问给出 l, r, p_l, p_r, 询问 S 的子串 $S_{p_l \dots p_r}$ 在串 $T_l, T_{l+1}, ..., T_r$ 中的哪一个串里的出现次数最多，求出出现次数最多的串的下标以及最多的出现次数。

2.3 数据范围

$1 \le |S| \le 5 \times 10^5$
$1 \le m \le 5 \times 10^4$
$1 \le \sum |T_i| \le 5 \times 10^4$
$1 \le q \le 5 \times 10^5$

2.4 时空限制

6 seconds, 768 megabytes

2.5 算法介绍

使用广义后缀自动机（基于多串匹配的后缀自动机）来解决此问题。

对给出的 m 个匹配串建立广义后缀自动机，并对于后缀自动机上的每一个节点（对应若干长度连续的子串），求出每个匹配串的 $endpos$ 集合大小。此时问题已经可以转化成如下两个独立的部分：

- 对于给出 S 的子串 $S_{p_l \dots p_r}$，求出其在后缀自动机上对应的节点编号。
- 对于给定的后缀自动机节点，求串 $T_l, T_{l+1}, ..., T_r$ 中的哪一个串的 $endpos$ 集合最大。

对于第一部分，可以先求出串 $S_{1 \dots p_r}$ 在后缀自动机上出现过的最长后缀对应的节点，此时若 $S_{p_l \dots p_r}$ 严格长于此最长后缀则说明 $S_{p_l \dots p_r}$ 不对应节点，否则说明 $S_{p_l \dots p_r}$ 对应的节点应是最长后缀对应节点在 $fail$ 树上的深度最浅且 len 值不小于 $p_r - p_l + 1$ 的祖先，通过倍增的方式定位到节点即可。

对于第二部分，可以对后缀自动机每个节点预先建立一棵线段树以维护每个匹配串的出现次数，这样对于一组询问就可以直接使用经典的线段树区间查询最大值的方式解决。维护这棵线段树也只需要使用线段树合并。

记 $\sum |T_i| = n$，该算法的时间复杂度为 $O((n+q)\log n)$，空间复杂度为 $O(n \log n)$。

3 Walking on a Tree

3.1 试题来源

AtCoder Grand Contest 025 E

3.2 试题大意

给出一棵 n 个节点的树以及 m 条路径 (u_i, v_i)，需要将所有路径定向（$u_i \to v_i$ 或 $v_i \to u_i$），定向后记 $x_i, y_i \in \{0,1\}$ 分别表示第 i 条树边 (a_i, b_i) 是否在 m 条路径中被沿 $a_i \to b_i$ 方向经过以及是否被沿 $b_i \to a_i$ 方向经过，最大化目标函数

$$T = \sum_{i=1}^{n-1} x_i + y_i$$

求出最大的 T 并构造出一组使得 T 取得最大值的定向方案。

3.3 数据范围

$1 \le n, m \le 2000$

3.4 时空限制

2 seconds, 1024 megabytes

3.5 算法介绍

记 c_i 表示在 m 条路径中，第 i 条树边 (a_i, b_i) 被覆盖了多少次。显然答案存在上界 $\sum_{i=1}^{n-1} \min\{2, c_i\}$，接下来将给出一种可以取到该上界的构造方案。

取出树上的任意一个叶子节点 v，记 u 为节点 v 的父亲节点，e 为树边 (u, v) 的编号，并考虑此时覆盖树边 e 的路径数量 c_e。

- 若 $c_e = 0$，那么树边 e 对答案的贡献为 0，可以直接将节点 v 以及树边 e 移除。
- 若 $c_e = 1$，假设覆盖 e 的唯一路径为 (v, x)，可以将 (v, x) 替换成 (u, x) 后删除节点 v 以及树边 e，这样最终路径 (v, x) 的定向将完全取决于 (u, x) 的定向，且始终对答案的贡献为 1。
- 若 $c_e \ge 2$，任取两条覆盖 e 的路径 $(v, x), (v, y)$，设两条路径的交为 (v, z)。对限制进行如下加强：若选择定向 $v \to x$ 则必须选择定向 $y \to v$，若选择定向 $x \to v$ 则必须选择定向 $v \to y$。可以发现此时路径 (v, z) 上的每一条树边的贡献都已经到达了 2，可以不再考虑。因此，选择 $v \to x, y \to v$ 等价于选择 $y \to x$，选择 $x \to v, v \to y$ 等价于选择 $x \to y$，直接删除路径 $(v, x), (v, y)$ 后加入路径 (x, y)，可以通过后续 (x, y) 的定向来确定 $(v, x), (v, y)$ 的定向。不断地选取两条覆盖 e 的路径直至 $c_e < 2$，便可以转化到前述的两种情况。

在具体实现中，上述的每一步都可在 $O(n+m)$ 的时间复杂度内完成，因此总复杂度为 $O(n(n+m))$。