解题报告

vincent.163

November 2019

1 Roads in Yusland

首先注意到该题具有树的结构，并且以 v_1 作为根节点建立有根树时，各棵子树之间路径覆盖的情况关系不大，因此考虑树形 dp。但怎么设计状态呢？一种很直观的想法是设 dp_i 表示修完以 v_i 为根的子树上所有道路需要的总代价。但这样的状态不足以转移，因为一个工人覆盖的是一整段路径，需要跨多次转移保留下来。在哪里考虑每个工人呢？如果把工人放在靠近根的节点 v_i 上，此时需要从根处考虑整个子树，状态过于复杂；如果把工人放在 u_i 上，则状态就简单很多，只需要考虑在某棵子树内所有路径全部修好的条件下，向 v_1 “延伸”若干条边所需要的代价。

因此 dp 状态需要对每个子树记录多个数，表示向 v_1 延伸一条、两条、三条边需要的代价。令 $f_i(k)$ 表示**仅**利用 v_i 子树内的工人修好 v_i 子树内所有边以及 v_i 开始向祖先数 k 条边需要的最小代价。

考虑转移：设 v_i 的孩子集合是 C_i，所有工人的三元组 (u, v, c) 组成的集合为 W（不妨设 $u \neq v$），v_i 的深度为 d_i，则

$$f_i(0) = \sum_{j \in C_i} f_j(1)$$

$$f_i(k) = f_i(0) + \min \begin{cases} \min\limits_{j \in C_i} (f_j(k+1) - f_j(1)) \\ \min\limits_{(u,i,c) \in W, d_u - d_i \geq k} c \end{cases}$$

将 $f_i(k)$ 视作数组直接维护的话，在树的深度是 $O(n)$ 的情况下需要的复杂度是 $O(n^2)$，需要用数据结构进行优化。

为了方便使用数据结构，我们对 $f_i(k)$ 的定义作出调整，表示修好 v_i 子树内所有边以及深度在 k 以下的所有祖先边所需要的代价，这样的转移便类似于合并操作，易于维护。

观察转移方程，该数据结构维护的是由深度和代价组成的二元组，需要支持以下操作：

- 合并：合并两个儿子的 f 函数。
- 取最小值：用来计算 $f_j(1)$.
- 集合减法：整个集合的代价减去 $f_j(1)$.

- 删除所有深度大于 d_i 的元素。

其中能够同时实现操作 $1, 2, 3$ 的集合只有可并堆，可以在堆上打标记实现集合减法操作。而第 4 个操作可以以懒惰的方式实现：取最小值操作时，不断删除深度不符合要求的点直到符合要求为止，从而在均摊意义上得到 $O(n \log n)$ 的复杂度。

用数据结构（例如左偏树和斜堆）实现上述 dp 的总复杂度为 $O(n \log n)$，能够通过本题。

2 Mike and Friends

该题是一道字符串计数题。我们首先给出简化题意，并给出较明显的暴力解法，然后分析题目性质，给出一种高效的解法。

2.1 简化题意

给定 n 个由英语小写字母组成的字符串，其总长度不超过 2×10^5，并编号为 $1 \ldots n$. 给出 q 个问题，每个问题有如下形式：有一个区间 $l, r (1 \leq l \leq r \leq n)$ 和一个编号 $k (1 \leq k \leq n)$，求区间 $l \ldots r$ 内的字符串中，出现编号为 k 的字符串的总个数。

2.2 暴力算法

记 $L = \sum_{i=1}^{n} len(s_i)$，则 $L \leq 2 \times 10^5$.

根据题意模拟，每次询问需要将区间 $l \ldots r$ 的所有字符串与编号为 k 的字符串进行匹配。假设使用 KMP 算法进行匹配，预处理每个模板字符串的 pre 数组，由于区间 $l \ldots r$ 的字符串总长度不超过 L，所以每次询问耗时 L，总耗时 qL.

该算法会超时，平方算法已无法继续优化，因此我们考虑使用字符串后缀数据结构来寻找接近线性的算法。

2.3 正解

对于每个询问，记 $f(x) = call(x, k)$. 注意到每次询问查询一个区间和 $\sum_{i=l}^{r} f(i)$，可以拆成两个前缀和之差 $\sum_{i=1}^{r} f(i) - \sum_{i=1}^{l-1} f(i)$。我们把每组询问拆成两个前缀和询问，每个询问形如 $\sum_{i=1}^{x} call(x, k)$，考虑如何高效地回答这样的询问。

解决字符串问题时，我们常用的方法有以下几种：

- KMP、AC 自动机：适合同时用多个模板串匹配一个字符串的情景。
- 后缀数组：适合注重字符串的字典序、求字符串 LCP 等情景。
- 后缀树、后缀自动机：适合字符串之间有结构、层次关系的情景，例如树结构。

本题中每次询问仅有一个模板串，需要同时匹配多个字符串，因此不适用第一类解法；没有字典序，不适用第二类解法；由于子串的子串还是子串，有一定的层次结构，因此考虑第三类解法。

关于后缀自动机的更多资料可以在 [1] 中找到。下面的定义并非后缀自动机的正式定义，仅列出用于解题必要的性质：

定义 2.1. 后缀自动机是一种易于实现的数据结构，包含了一些字符串的后缀信息，且同时具有有限自动机的结构和树形结构。其有限自动机结构满足：能且仅能接受其包含的字符串的某个子串。其树形结构由通常的后缀自动机实现中的后缀链接组成，除了根节点以外，每个节点的父亲为其后缀链接指向的节点。

尽管在 OI 中该数据结构被称为后缀自动机，但实际使用时较常用的反而是其后缀链接组成的树形结构（常称为 Parent 树）。在后缀自动机的证明中，该树形结构体现了 Right 集合的层次关系。

实际上，这棵树对应的是一棵反向的压缩 Trie 树，包含了用于构建后缀自动机的字符串的每个前缀，且对于任意两个节点，若 A 节点是 B 节点的祖先，则 A 节点对应的字符串是 B 节点对应的字符串的一个后缀，反之亦然。

由于题目给出的字符串之间没有顺序关系，考虑直接建立广义后缀自动机 [2]，这样对于题目给出的每个字符串的每个前缀，都可以在后缀自动机上找到对应的点，且字符串 A 是 B 的当且仅当在 Parent 树上 A 是 B 的祖先。

回到原题，考虑如何用上述性质解决题目。首先，子串即是某个后缀的某个前缀，我们将 s_i 是 s_j 的子串这一条件改写为：s_i 是 s_j 的一个前缀的后缀。改写后每次询问的形式如下：计算 $s_{i\ldots j}$ 每个字符串的每一个前缀中满足后缀是 s_k 的前缀的个数。我们将询问放到 Parent 树上考虑，则询问的形式如下：计算 $s_{i\ldots j}$ 每个字符串的每一个前缀对应的树上节点中位于 s_k 对应的树上节点的子树中的个数.

该询问具有子树求和的结构，通常的做法是对树求一个 DFS 序，这样每棵子树就对应 DFS 序的一个区间，容易利用线段树等数据结构解决。

但本题对一个前缀节点集合进行求和。此时一般考虑两种解法：一种是离线解法。将询问排序，将节点按顺序加入树上，同时在合适的时间进行查询回答询问，用树状数组即可维护。另一种是在线解法，同样将节点按顺序加入树上，但使用可持久化数据结构维护，这样便可以在线回答询问，查询前缀对应的树即可。

3 Monochrome Cat

该题为一道性质很明显的贪心题。我们将会一步步给出题中需要用到的性质并加以证明，然后给出能在规定时间内通过的算法。

首先对题目大意进行整理：我们要按照一定的规则在树上游走，最终将所有点染为黑色。为了方便讨论，我们首先特判掉整棵树所有节点初始时便为黑色的情况，此时答案为 0；以及只有一个白色节点的情况，此时答案为 1. 接下来，我们假设树上至少有两个白色节点。

定义 3.1. 一组解由猫的起始节点以及每次选择的操作组成。定义所有操作的目标节点组成集合为**操作树**。

注意操作树不一定包含起始节点。前 1 次操作只有一个节点，一定是一个联通块；假设前 i 次操作的顶点形成的集合代表了树上的一个联通块，由于第 $i+1$ 次操作与第 i 次操作的顶点要么是同一个点，要么相邻，因此第 $i+1$ 次操作的顶点一定和前 i 次操作的顶点形成的联通块相邻，从而前 $i+1$ 次操作一定形成一个联通块。因此，操作树一定是树上的一个联通块，从而一定是一棵树。

这组解的秒数等于操作序列的长度。

引理 3.1. *存在一组秒数最少的最优解，使得操作树上不存在初始为黑色的叶子节点。*

证明. 假设这样的最优解不存在，我们考虑任一组最优解，其操作次数为 k，某次操作的目标节点为 v_i，且该节点初始时为黑色的叶子节点。我们设该操作是第 b 次操作，并前后寻找操作目标同样是 v_i 的操作的区间端点，设 $a \dots c$ 次操作的目标均为 v_i.

根据操作区间的位置，考虑三种情况：

- 操作位于开头，即 $a = 1$。此时第 $c+1$ 次操作的目标一定是 v_j，我们删除 $a \dots c$ 区间内所有操作，并将 v_j 作为新的解的起始节点。由于黑色叶子节点仍然是黑色，其它所有节点的颜色不变，这组解的操作序列长度严格小于原有解的长度，与假设矛盾。
- 操作位于结尾，即 $c = k$。此时直接删除第 $a \dots c$ 次操作，基于同样的理由得出矛盾。
- 操作位于中间。此时删去第 $a \dots c$ 次操作后操作序列相邻两点的距离一定不超过 1，仍然满足条件，因此得出矛盾。

无论何种情况假设均不成立，从而原命题成立。 □

接下来，我们只考虑操作树所有叶子节点均为白色节点的情况。显然操作树必须包含所有的白色节点（否则无法变色），因此操作树一定是包含所有白色节点的最小联通块。

引理 3.2. *在上述条件下，存在一组秒数最少的最优解，经过每条边不超过两次。*

证明. 假设有一组解经过一条边至少三次，设该边的两个端点为 v_a 和 v_b。不妨设第一次经过该边时的方向是 $v_a \to v_b$，第二次为 $v_b \to v_a$，第三次为 $v_a \to v_b$。

将第一次经过该边后与第二次经过该边前的操作移动到第三次经过该边后，由于该操作序列的起始位置是 v_b，终止位置也是 v_b，因此不会影响相邻操作之间距离不能超过 1 这个限制。第一次操作和第二次操作的实际效果相当于将 v_a 和 v_b 均进行一次反色，可以在删除这两次操作，并改为在第三次操作时进行两次反色操作。这样秒数不变，但经过每条边的总次数减少了。

对该解进行若干次这样的变换，由于移动操作数不断减少，该过程最终会结束，此时得到的解一定满足经过每条边不超过两次的条件，从而得证。 □

实际上，除了起始节点与终止节点路径上的边恰经过一次以外，每条边都会沿两个方向恰好经过两次。因此，在不进行反色操作的情况下，每个点的反色情况如下：

- 首先对于每个点，若度数为奇数，则会被反色，否则颜色保持不变。
- 对于起始节点和终止节点之间路径上的点（不包括起始节点），由于少进行了一次反色，相当于额外进行了一次反色。

对于进行这样的反色操作后仍然为白色的点，我们至少需要额外进行一次反色操作。我们向操作序列在合适的时间点添加一次反色操作，便得到了最优解。

因此，我们的最优解组成如下：

- 首先取出包含所有黑色节点的最小联通块，设点数为 n'。
- 对于联通块内每个点，若该点在联通块内的度数为奇数，则会被反色，否则颜色保持不变。设此时白色点的个数为 x.
- 在联通块内选择一条路径，但两个端点在原来的树中不能都是叶子节点，这样起始节点就可以选为路径外的一个点。设路径上白色点数为 p，黑色点数为 q，然后将路径反色.
- 答案为 $2n' + x - (p + q) - p + q$，化简得到 $2n' + x - 2p$.

其中 $2n' + x$ 是常数，可以直接计算，而 p 是路径上白色节点的个数，可以用类似于树上带权直径的方法实现，但要注意两个端点不能均为叶子节点。一种比较方便的实现是，将黑色节点赋权值 0，白色叶子节点赋权值 1，其它白色节点赋权值 2，最终计算出的直径除以 2 向上取整。

References

[1] Suffix automation. `https://cp-algorithms.com/string/suffix-automaton.html`.

[2] 刘研绎. 后缀自动机在字典树上的拓展. 国家集训队论文集, pages 1–16, 2015.