解题报告

陈立言

October 24, 2019

1 Duff is Mad

1.1 题目大意

给出 n 个字符串，第 i 个为 s_i。q 次询问，每次给出三个整数 l, r, k，询问 $\sum_{i=l}^{r} \operatorname{occur}(s_i, s_k)$，其中 $\operatorname{occur}(t, s)$ 为串 t 在串 s 中的出现次数。

1.2 数据范围

$1 \le n, q, \sum |s_i| \le 100000$

1.3 约定

$f(l, r, k) = \sum_{i=l}^{r} \operatorname{occur}(s_i, s_k)$。

$g(r, k) = f(1, r, k)$，特别地，$g(0, k) = 0$。

树 S 为储存所有 s_i 的 trie 树。S 的根的编号为 1。$P(\text{str})$ 为串 str 在 S 中对应的结点，若不存在，则 $P(\text{str}) = 0$。$P_i = P(s_i)$。

fail_i 为对树 S 运行 Aho–Corasick algorithm 后，结点 i 的失配指针指向的结点。

fail 树为对于所有 $i \in S, i \neq 1$，在 fail_i 和 i 连边后得到的无向图。可以证明它是树，且以 1 为根时，结点 i $(i > 1)$ 的父亲为 fail_i。

1.4 解题过程

先考虑如何计算 $\operatorname{occur}(s_i, s_k)$。我们可以枚举 s_k 的每个前缀，判断 s_i 是否是这个前缀的后缀。

因此我们建出 trie 树 S 和 fail 树。

那么，枚举 s_k 的每个前缀就是枚举 P_k 在 S 上到根的路径上的每个点。

fail 树满足如下性质：如果 $P(s) \neq 0, P(t) \neq 0$，那么串 s 是串 t 的后缀当且仅当在 fail 树上 $P(s)$ 是 $P(t)$ 的祖先（包括 $P(t)$）。证明在附录中给出。

于是，我们可以得到 $\mathrm{occur}(s_i, s_k)$ 的一个计算方式：

1. 令答案为 0。
2. 枚举 P_k 在 S 上到根的路径上的所有点，当枚举到 k 时，若 k 在 fail 树上在 P_i 的子树里，那么答案加 1。

类似地，我们得到 $g(r, k)$ 的一个计算方式：

1. 对 S 上每个结点 i 设置变量 $v_i = 0$。
2. 枚举所有 $1 \leq i \leq r$，把所有在 fail 树中 P_i 的子树里的结点 j 的 v_j 加 1。
3. 答案等于 S 中 P_k 到根的路径上的所有结点的权值 v 之和。

考虑 $f(l, r, k) = g(r, k) - g(l-1, k)$。因此，我们可以离线处理询问，按 r 从小到大计算 $g(r, k)$。

那么问题实际上就相当于：

> 给出两棵有根树 T_1，T_2。每个编号 i 分别对应 T_1 和 T_2 中的一个结点，和一个初始为 0 的变量 v。需要支持以下两种操作：
>
> - 给定 i，把 T_2 中 i 子树内的点对应的 v 加 1。
> - 给定 i，计算 T_1 中 i 到根的路径上的点对应的 v 之和。

考虑使用分块来解决这个问题。因为本题中 $n, q, \sum |s_i|$ 同阶，方便起见都认为它们是 $O(n)$ 的。

我们令块的大小 $B = \lfloor \sqrt{n} \rfloor$，那么 $B = O(\sqrt{n})$，且 $\frac{n}{B} = O(\sqrt{n})$。

我们把子树加操作看成在 dfs 序上做区间加，并对 T_2 的 dfs 序分块。这样每个区间加操作相当于 $O(\sqrt{n})$ 个整块加和 $O(\sqrt{n})$ 个单点加。询问时应分开考虑这两部分的贡献。

对于整块加，我们直接在这个块上打标记。查询时，我们只要知道询问点到根的路径上有多少个这个块里的点，就容易计算出整块加对答案的贡献。这是可以预处理的。

对于单点加，需要注意修改次数是 $O(n\sqrt{n})$ 的，然而询问只有 $O(n)$ 次。我们可以对这个问题作进一步转化：单点加点到根求和，相当于区间加单点求和，又相当于单点加区间求和。而通过使用分块，容易得到单点加区间求和问题的 $O(1)$ 修改 $O(\sqrt{n})$ 询问的算法。这里我们直接套用。

以上两部分的复杂度都是 $O(n\sqrt{n})$。于是，我们得到了本问题的一个 $O(n\sqrt{n})$ 的算法，可以通过测试数据。

1.5 附录

定理 1：fail 树满足如下性质：如果 $P(s) \neq 0, P(t) \neq 0$，那么串 s 是串 t 的后缀当且仅当在 fail 树上 $P(s)$ 是 $P(t)$ 的祖先（包括 $P(t)$）。

定理 1 证明：

由 fail 指针的定义，定理 1 的必要性是显然的。下面我们来证明充分性。

对 s 和 t 的长度差使用归纳法。

首先当 $|s| = |t|$，那么 $s = t$，$P(s) = P(t)$。

现在设 $|t| - |s| = a > 0$。考虑 t 在 fail 树上的父亲的长度 $|t| - b$，有三种可能：

- $a = b$。那么 s 就是 t 在 fail 树上的父亲。
- $a > b$。由于长度差 $a - b < a$，这说明 $P(s)$ 是 $P(t)$ 的父亲的祖先。因此 $P(s)$ 是 $P(t)$ 的祖先。
- $a < b$。这说明 s 是比 $\text{fail}_{P(t)}$ 对应的串更长，且是 t 的后缀，且不为 t 的串。这不符合 fail 指针的定义。

由此我们证明了定理 1 的充分性。 □

2 Optimal Point

2.1 题目大意

给出三维空间中的 n 个整点。求一个整点，要求最小化到 n 个点的最大曼哈顿距离。

2.2 数据范围

$n \le 10^5$，坐标的绝对值在 10^{18} 以内。

2.3 解题过程

先二分答案。设当前二分的答案为 d。那么就得到 n 个限制，都为 $|x-x_0|+|y-y_0|+|z-z_0| \le d$ 的形式。

不失一般性地考虑 $|x|+|y|+|z| \le d$ 的特殊情况。由于 $|a| \ge a, |a| \ge -a$，所以我们可以枚举各个绝对值取正或取负，得到 8 个不等式。

$$\begin{cases} x+y+z \le d \\ -x+y+z \le d \\ x-y+z \le d \\ x+y-z \le d \\ \cdots \\ -x-y-z \le d \end{cases}$$

我们断言，这个不等式组和原来的不等式是等价的。实际上，在原不等式中枚举 x, y, z 的正负，能得到类似的不等式组，但每一条都只用在 x, y, z 满足限制的情况下成立。(例如能得到 $-x+y+z \le d$，当 $x \le 0, y \ge 0, z \ge 0$)

因此，只要对 x, y, z 作加常数的调整，每个形如 $|x-x_0|+|y-y_0|+|z-z_0| \le d$ 的不等式都能以上述形式表示。合并 n 个限制后，我们能得到这样一个不等式组 C：

$$\begin{cases} l_0 \le x+y+z \le r_0 \\ l_1 \le -x+y+z \le r_1 \\ l_2 \le x-y+z \le r_2 \\ l_3 \le x+y-z \le r_3 \end{cases}$$

接下来就要尝试求出它的一个整数解。

设 $a=-x+y+z, b=x-y+z, c=x+y-z$，那么 $x=\frac{b+c}{2}, y=\frac{a+c}{2}, z=\frac{a+b}{2}$，且 $x+y+z=a+b+c$，因此上述不等式组能表示为

$$\begin{cases} l_0 \le a+b+c \le r_0 \\ l_1 \le a \le r_1 \\ l_2 \le b \le r_2 \\ l_3 \le c \le r_3 \end{cases}$$

且要求 a, b, c 奇偶性相同。

于是我们再令 $a = 2a' + r, b = 2b' + r, c = 2c' + r$，其中 $r \in \{0, 1\}$。代入得到不等式组 C'

$$\begin{cases} l_0 \le 2(a'+b'+c')+3r \le r_0 \\ l_1 \le 2a'+r \le r_1 \\ l_2 \le 2b'+r \le r_2 \\ l_3 \le 2c'+r \le r_3 \end{cases}$$

定理 1：不等式组 C 有整数解当且仅当不等式组 C' 有满足 $r \in \{0, 1\}$ 的整数解。

定理 1 证明：

充分性：设不等式组 C 有整数解 (x, y, z)。那么，可以令 $r = (x+y+z) \bmod 2$，于是可以得到 $a' = \frac{x+y+z-r}{2} - x, b' = \frac{x+y+z-r}{2} - y, c' = \frac{x+y+z-r}{2} - z$。这是不等式组 C' 的一个合法解。

必要性：设不等式组 C' 有解 (a', b', c', r)。于是有 $x = b' + c' + r, y = a' + c' + r, z = a' + b' + r$。这是不等式组 C 的一个整数解。 $\square$

接下来问题就变成求解不等式组 C' 。首先枚举 r，然后整理不等式，可以得到如下形式：

$$\begin{cases} l'_0 \le a'+b'+c' \le r'_0 \\ l'_1 \le a' \le r'_1 \\ l'_2 \le b' \le r'_2 \\ l'_3 \le c' \le r'_3 \end{cases}$$

那么 $a' + b' + c'$ 可以是属于区间 $[l'_1 + l'_2 + l'_3, r'_1 + r'_2 + r'_3]$ 中的任意一个整数，且我们可以构造出相应的 a', b', c'。一个构造方法是贪心，先让 a', b', c' 分别取 l'_1, l'_2, l'_3，然后调大 a', b', c'。

因此，我们得到了判断不等组 C' 是否有解及求出一组解的 $O(1)$ 算法。

设坐标的权值范围为 V，那么本算法时间复杂度为 $O(n \log V)$。

3 Everything on It

3.1 题目大意

有 n 个物品，生成了 2^n 个集合。求有多少种方案，选取任意数量个集合，使得每个物品至少被两个所选的集合包含。对 m 取模。

3.2 数据范围

$n \le 3000$ 且 $10^8 \le m \le 10^9+9$

3.3 约定

记 $2^J = \{S \mid S \subseteq J\}$，其中 J 是一个有限集合。

记 $U = 2^{2^{\{1,\ldots,n\}}}$。

3.4 解题过程

使用容斥原理来解决这个问题。

设 P_i 表示满足物品 i 出现不超过两次的 U 的元素构成的集合。那么，根据容斥原理（内容可参考附录），答案就是

$$|U| - |\bigcup_{i=1}^{n} P_i| = |U| + \sum_{\emptyset \ne J \subseteq \{1,\ldots,n\}} (-1)^{|J|} |\bigcap_{j\in J} P_j|$$

因为每个物品都是相同的，那么对于大小相同的集合 J ($\emptyset \ne J \subseteq \{1,\ldots,n\}$)，$|\bigcap_{j\in J} P_j|$ 都应是相等的。我们记 $|J| = k$ 时，$|\bigcap_{j\in J} P_j| = f_k$。特别地，令 $f_0 = |U|$。那么要求的就是

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} f_k$$

考虑计算 f_k。这里，有 k 个物品必须属于不超过 1 个集合，$n-k$ 个物品没有限制。首先考虑不包含那 k 个有限制物品的集合，它们是可以任意选取的，于是有 $2^{2^{n-k}}$ 种方案。而 k 个特殊物品中的每一个要么只属于一个集合，要么不属于任何一个集合，并且它们也能跟那 $n-k$ 个物品在同一个集合中。记 $g_{k,j}$ 表示把 k 个不相同的物品选择性地分到 j 个无序非空集合的方案数（选择性是指一个物品可以不被分到任何集合中）。那么，如果最后有 j 个集合包含了 k 个特殊物品中的至少一个，那么方案数就是 $g_{k,j} \times (2^{n-k})^j$。

于是我们有 $f_k = 2^{2^{n-k}} \sum_j 2^{(n-k)j} g_{k,j}$。问题就变成计算 $g_{k,j}$。考虑递推式。

首先，显然有 $g_{k,0}=1$。

然后考虑 $g_{k,j}$。我们可以讨论第 k 个物品的情况。

- 单独在一个大小为 1 的集合中。剩下的方案数就是 $g_{k-1,j-1}$。
- 在一个大小大于 1 的集合中。那么方案数就是先让前 $k-1$ 个物品形成 j 个集合，再选一个集合加入第 k 个物品。方案数是 $j\times g_{k-1,j}$。
- 不在任何一个集合中。方案数是 $g_{k-1,j}$。

于是，我们能得到 $g_{k,j}=g_{k-1,j-1}+(j+1)g_{k-1,j}$。

那么，我们就能 $O(n^2)$ 预处理出所有 $g_{k,j}$。

在上述等式中，快速幂的运算次数可以只有 $O(n)$ 次，故本算法的复杂度为 $O(n^2)$。

3.5 附录

容斥原理设有 n 个有限集合 $A_1,A_2,\ldots,A_n$，那么有

$$\left|\bigcup_{i=1}^{n}A_i\right|=\sum_{\emptyset\neq J\subseteq\{1,\ldots,n\}}(-1)^{|J|+1}\left|\bigcap_{j\in J}A_j\right|$$