《复读程度》解题报告

中国人民大学附属中学 许庭强

2022 年 10 月

1 题目

1.1 题目大意

给定字符串 $s[1,n]$，第 i 个字符被赋予两个权值：**左权值** wl_i 和**右权值** wr_i。

定义一个子串 $s[l,r]$ 的**左权值** $vl(s[l,r])$ 为：其在原串中各个匹配的左端点的**左权值** wl 和；**右权值** $vr(s[l,r])$ 为：其在原串中各个匹配的右端点的**右权值** wr 和。这里 t 在 s 中所有的匹配是 $\forall 1 \le i \le j \le n, s[i,j] = t$，我们把这样的 i 和 j 分别叫做一个匹配的左右端点。

定义一个子串 $s[l,r]$ 的复读程度是它的**左权值**与**右权值**的乘积，即 $w(s[l,r]) = vl(s[l,r]) \cdot vr(s[l,r])$。

q 次查询 l_1, r_1, l_2, r_2，令 $S = \{(l,r) \mid r - l \ge \max\{r_1 - l_1, r_2 - l_2\}, s[l, l + r_1 - l_1] = s[l_1, r_1], s[r - r_2 + l_2, r] = s[l_2, r_2]\}$，即以 S 为所有满足 $s[l_1,r_1]$ 为 $s[l,r]$ 的前缀，$s[l_2,r_2]$ 为 $s[l,r]$ 的后缀的 (l,r) 的集合。求 $\sum\limits_{(l,r)\in S} w(s[l,r])$。

由于答案很大，你只需要输出答案对 2^{64} 取模后的结果。

1.2 输入格式

第一行两个正整数 n, q，表示字符串长度与查询次数。

第二行一个由小写字母构成的字符串 s。

第三行 n 个整数 wl_i 表示左权值。

第四行 n 个整数 wr_i 表示右权值。

接下来 q 行，每行给出四个整数 l_1, r_1, l_2, r_2，表示一次查询。

1.3 输出格式

q 行，第 i 行输出第 i 次查询的答案对 2^{64} 取模后的结果。

1.4 数据范围

保证 $1 \le n, q \le 10^5, 0 \le wl_i, wr_i < 2^{64}, 1 \le l_1 \le r_1 \le n, 1 \le l_2 \le r_2 \le n$，$s$ 仅有小写字母构成。

subtask1(7pts)：保证 $n, q \le 500$。

subtask2(15pts)：保证 $n, q \le 5000$。

subtask3(12pts)：保证 $n \le 5000$。

subtask4(6pts)：保证 $l_1 = 1$，且对任意 $2 \le i \le n$，有 $s_1 \ne s_i$。

subtask5(16pts)：保证 $s[l_1, r_1]$ 在 s 中最多出现 5 次。

subtask6(21pts)：保证 $n, q \le 5 \times 10^4$。

subtask7(23pts)：无特殊限制。

1.5 时空限制

时间限制：5s

空间限制：1GB

2 解题过程

2.1 算法 1

暴力预处理出所有区间 $[l, r]$ 对应的复读程度 $w(s[l, r])$。然后对每次查询，枚举所有可能的 $[l, r]$，用哈希判断是否有 $[l, r] \in S$，统计答案。

时间复杂度 $O(n^3 + qn^2)$，可以通过子任务 1，期望得分 7 分。

2.2 算法 2

首先考虑如何快速求出 $w(s[l, r])$。考虑将正反串的 SAM 建出，我们规定如下记号：

- T_0, T_1 分别表示 s 的正串/反串 parent 树。
- $s_0(u), s_1(u)$ 分别表示正串/反串 parent 树上 u 节点包含的字符串集合。
- $T_0(l, r), T_1(l, r)$ 分别表示 $s[l, r]$ 所在的正串/反串 SAM 节点。
- $\text{subtree}_0(u), \text{subtree}_1(u)$ 分别表示正串/反串 parent 树上以 u 为根的子树。
- $\text{occ}(t)$ 表示字符串 t 在 s 中的出现次数。

- $\text{dfn}(u)$ 表示 u 在对应 parent 树上的 dfs 序。

容易发现，我们有

$$vl(s[l,r]) = \sum_{T_1(i,n)\in \text{subtree}_1(T_1(l,r))} wl_i$$

$$vr(s[l,r]) = \sum_{T_0(1,i)\in \text{subtree}_0(T_0(l,r))} wr_i$$

同时，$(l,r)\in S$ 成立当且仅当

1. $T_0(l,r)\in \text{subtree}_0(T_0(l_2,r_2))\setminus\{T_0(l_2,r_2)\}$ 或 $T_0(l,r)=T_0(l_2,r_2), r-l\geq r_2-l_2$
2. $T_1(l,r)\in \text{subtree}_1(T_1(l_1,r_1))\setminus\{T_1(l_1,r_1)\}$ 或 $T_1(l,r)=T_1(l_1,r_1), r-l\geq r_1-l_1$

同时成立。我们可以令 $S=S_1\setminus(S_2\cup S_3)$，其中

$$S_1=\{(l,r)\mid T_0(l,r)\in \text{subtree}_0(T_0(l_2,r_2)), T_1(l,r)\in \text{subtree}_1(T_1(l_1,r_1))\}$$

$$S_2=\{(l,r)\mid (l,r)\in S_1, T_0(l,r)=T_0(l_2,r_2), r-l<r_2-l_2\}$$

$$S_3=\{(l,r)\mid (l,r)\in S_1, T_1(l,r)=T_1(l_1,r_1), r-l<r_1-l_1\}$$

接下来考虑如何分别求出 $\sum\limits_{(l,r)\in S_1} w(s[l,r])$ 与 $\sum\limits_{(l,r)\in S_2\cup S_3} w(s[l,r])$。

对于 S_1，首先求出 T_0,T_1 的 dfs 序，此时 S_1 对应的答案就变为了一个矩形求和。矩形中每个点的权值为一个 $s_0(u)\cap s_1(v)(u\in T_0,v\in T_1)$ 中所有串对应的 w 之和。我们可以 $O(n^2)$ 预处理出所有 $w(s[l,r])$，然后找到其对应的节点 $T_0(l,r)$ 与 $T_1(l,r)$，将其统计到矩形中。这样就求出了矩形中每个节点的权值。用 $O(n^2)$ 暴力预处理出前缀和，即可对每次查询 $O(1)$ 得到答案。考虑到定位 $T_0(l_2,r_2)$ 与 $T_1(l_1,r_1)$ 所需的代价，该部分时间复杂度为 $O(n^2+q\log n)$。

对于 $S_2\cup S_3$，由于数据范围较小，可以枚举 $s_0(T_0(l_2,r_2))$ 与 $s_1(T_1(l_1,r_1))$ 中的所有字符串。该部分时间复杂度为 $O(qn)$。

总复杂度为 $O(n^2+qn)$，可以通过子任务 1,2，期望得分 22 分。

2.3 算法 3

考虑如何加速 $\sum_{(l,r)\in S_2\cup S_3} w(s[l,r])$ 的求解过程。可以暴力与处理出 $s_0(T_0(l_2,r_2))$ 中所有字符串在 T_1 上的哪个点，那么求出 $\sum_{(l,r)\in S_2} w(s[l,r])$ 就是容易的。对于 S_3 同理。对于 $S_2\cap S_3$，可以证明（将在后面具体给出）$s_0(T_0(l_2,r_2))\cap s_1(T_1(l_1,r_1))$ 最多包含一种串，可以用多种方法在 $O(\log n)$ 内找出。

结合算法 2 中对 S_1 部分的求解，总时间复杂度为 $O(n^2+q\log n)$，可以通过子任务 1,2,3，期望得分 34 分。

2.4 算法 4

如果 $l_1=1$ 且对任意 $2\le i\le n$，有 $s_1\neq s_i$，那么显然所有 S 中的 (l,r) 必须有 $l=1$，而且仅出现这一次。因此，$w(s[l,r])=wl_1wr_r$，只需求所有满足条件的 r 的 wr_r 之和即可。线段树合并后容易做到每次查询 $O(\log n)$。

时间复杂度 $O((n+q)\log n)$，可以通过子任务 4，与算法 3 结合后期望得分 40 分。

2.5 算法 5

如果 $s[l_1,r_1]$ 在 s 中最多出现 5 次，那么 S 中的任何 (l,r) 都有 $s[l,r]$ 在 s 中最多出现 5 次。于是可以枚举 S 中的所有可能 l，并找到所有出现次数对应的 r 的区间，求出 r 在对应区间中的所有串对答案的贡献。

时间复杂度 $O(n\log n+25q\log n)$，可以通过子任务 4,5，与算法 3 结合后期望得分 56 分。

2.6 算法 6

我们进一步剖析所有子串的结构。以往的后缀数据结构多为单向的联系，即某一个 s 的子串是另一个的后缀（或前缀）。然而本题所求的串需要满足两个方向的内容：$s[l_1,r_1]$ 为其前缀且 $s[l_2,r_2]$ 为其后缀。这时我们就需要将不同方向联系起来。

对每个 s 的子串 t，我们定义其扩展串 $\text{ext}(t)$ 为最长的 s 的子串 t' 使得其包含 t 且满足 $\text{occ}(t)=\text{occ}(t')$。

我们接下来证明这是良定的。首先 $\text{ext}(t)$ 一定存在，因为 t 与它本身 occ 相等。其次，如果有两个不同的串 t',t'' 均满足条件，我们有 $\text{occ}(t)=\text{occ}(t')=\text{occ}(t'')$，所以 t' 与 t'' 中一定恰好只包含一次 t。因此，我们有 t' 的所有出现位置与 t 的所有出现位置的一一对应。对 t'' 同理。这样，我们考虑一次 t 在 s 中的出现，设为 $s[l,r]$。找到其对应的 t' 与 t'' 的出现位置，设为 $s[l',r']$ 与 $s[l'',r'']$。我们有 $l',l''\le l\le r\le r',r''$ 且 $s[l,r]=t,s[l',r']=t',s[l'',r'']=t''$。

那么设 $L=\min(l',l''),R=\max(r',r'')$，$s[L,R]$ 被 t,t',t'' 唯一确定。因此，对于每一个不同的 (l,r)，$s[L,R]$ 均会出现一次，所以 $\text{occ}(s[L,R])\geq\text{occ}(t)$。又由于 $s[L,R]$ 包含 $s[l,r]=t$，$\text{occ}(s[L,R])=\text{occ}(t)$。由 $\text{ext}(t)$ 的最长性，一定有 $R-L=r'-l'=r''-l''$，该式成立当且仅当 $l'=l'',r'=r''$，即 $t'=t''$。

从该证明过程中也可以注意到，如果 l',r',l'',r'' 满足 $l',l''\leq l\leq r\leq r',r'',\text{occ}(s[l',r'])=\text{occ}(s[l'',r''])=\text{occ}(s[l,r])$，那么也一定有 $\text{occ}(s[\min(l',l''),\max(r',r'')])=\text{occ}(s[l,r])$。进一步的，对任意在 l',l'' 之间的 i，和在 r',r'' 之间的 j，均有 $\text{occ}(s[i,j])=\text{occ}(s[l,r])$。

接下来我们将所有串按照 ext 分成若干等价类，即任意两个不同的串 x 与 y，它们在同一类当且仅当 $\text{ext}(x)=\text{ext}(y)$。容易验证，$\text{ext}(\text{ext}(x))=\text{ext}(x)$。我们称 $\text{ext}(x)$ 为 x 所在等价类的代表元。

定义 t 在 s 中第一次出现的位置为 $s[\text{posl}(t),\text{posr}(t)]$。那么有 $\text{posl}(\text{ext}(t))\leq\text{posl}(t)\leq\text{posr}(t)\leq\text{posr}(\text{ext}(t))$。根据前面的分析，能够发现，同一个类中的所有 $(\text{posl}(t),\text{posr}(t))$ 在二维平面上构成了一个阶梯状网格图。我们又称一个阶梯状网格图为一块。

例如，若 $s=\texttt{abaab}$，则所有类的代表元分别为 $\texttt{abaab},\texttt{ab},\texttt{a}$。$(\text{posl},\text{posr})$ 构成的阶梯状网格分别为 $(1\sim2,3),(1\sim3,4),(1\sim3,5)$，$(1\sim2,2)$ 与 $(1,1)$。

另外，对于网格图的边缘，添加/删除一个字符得到的串 occ 一定不相同，这也说明了，在网格图中，每一行均对应一个正串 SAM 节点上的所有串，每一列均对应一个反串 SAM 节点上的所有串。这样，通过这个结构，我们很好的将正串与反串的 SAM 联系在了一起。同时我们也证明了所有块的周长之和不超过 $O(n)$。我们称该结构为基本子串结构。

回到题目，可以发现，在基本子串结构上，每一个串的复读程度即为其在块内对应的两个 SAM 节点权值乘积。即，对每一行有一个权值 a_i，对每一列有一个权值 b_j，点 (i,j) 对应的串的复读程度为 a_ib_j。我们沿用算法 2 的思路。首先对于 $S_2\cup S_3$ 这一部分，我们可以分两类讨论：

1. $T_0(s[l_2,r_2])$ 与 $T_1(s[l_1,r_1])$ 在同一块中，那么 $S_2\cup S_3$ 最多只有一种串，即为在网格图上的交点，可以简单 $O(1)$ 求出。

2. $T_0(s[l_2,r_2])$ 与 $T_1(s[l_1,r_1])$ 在不同块中，此时有 $S_2\cap S_3=\emptyset$，只需分别对 S_2,S_3 统计即可。问题变为在网格图中，一个区间对应的 SAM 节点有多少个点的 dfs 序在某个区间中（即在对应子树中）。可以离线后用二维数点解决。

至此，对于 $S_2\cup S_3$ 的这一部分，我们以 $O(q\log n)$ 的时间复杂度内给出了答案。

接下来讨论 S_1 如何求解。重新审视 S_1 部分的答案，它可以被表示为一个矩形中所有数之和。差分后变为 4 个矩形左下角求和问题。其是一个二维查询，考虑莫队。每一次莫队端点移动相当于给定 x,y，查询 $\sum\limits_{1\leq l\leq r\leq n}[\text{dfn}(T_0(l,r))=x,\text{dfn}(T_1(l,r))\leq y]w(s[l,r])$。这对应的是：在网格图中 $\text{dfn}(u)=x$ 对应的 u 节点所在行，有哪些 $\text{dfn}(v)\leq y$ 的节点对应的列与其

相交，并对相交的位置求权值和。如果对每个块维护一棵主席树，就可以做到 $O(n\sqrt{q}\log n)$ 的时间复杂度。对于另一侧同理。

总时间复杂度 $O((n\sqrt{q}+q)\log n)$，能通过子任务 1,2,3,6，与算法 5 结合后期望得分 77 分。

2.7 算法 7

我们需要想办法去掉算法 6 中的 $\log n$ 因子。一个自然的想法为莫队二次离线。离线后对应的问题为 $n\sqrt{q}$ 次给定 x,y，查询 $\sum\limits_{1\le l\le r\le n}[\text{dfn}(T_0(l,r))=x,\text{dfn}(T_1(l,r))\le y]w(s[l,r])$。离线后，由于阶梯形网格优秀的性质，我们可以按照网格顺序扫描线，每次添加一个位置，查询一个区间。容易做到 $O(\sqrt{n})$ 修改，$O(1)$ 查询。总复杂度优化为 $O(n\sqrt{q}+q\log n)$。对于另一侧同理。

时间复杂度 $O(n\sqrt{q}+q\log n)$，空间复杂度 $O(n\sqrt{q})$，由于空间过大，仍然只能通过子任务 1,2,3,6，与算法 5 结合后期望得分 77 分。

2.8 算法 8

考虑优化二次离线的空间复杂度。将每次一段连续的莫队端点移动合并在一起，这样就仅有 $O(q)$ 段。此时每段对应的查询相当于给定 x_1,x_2,y，查询 $\sum\limits_{1\le l\le r\le n}[x_1\le\text{dfn}(T_0(l,r))\le x_2,\text{dfn}(T_1(l,r))\le y]w(s[l,r])$。对 y 做扫描线，每次添加一个 T_1 上的节点，那么其对 T_0 的影响即为网格图上一段连续的节点。将所有 T_0 的节点重新标号，使得每块中的节点编号连续。记节点 u 的编号为 $\text{relabel}(u)$。

仔细考虑如何对 y 做扫描线。由于块内 (i,j) 对应的串的复读程度可被写为 a_ib_j，我们需要维护两个序列 A,B，其下标为重标号后的下标，且 A_i 固定为标号为 i 的 T_0 上的节点的权值。扫描时，要做的操作即为：对某个编号区间 $[e,f]$，令其中所有的 $B_i+=cA_i$，其中 c 为 dfn 为 y 的点的权值乘上该块对应的 occ。查询即为：遍历 $[x_1,x_2]$，对其中每一个 j，找到 $\text{dfn}(u)=j$ 对应的 u，查询 $B_{\text{relabel}(u)}$ 的值。该问题也可以用一个 $O(\sqrt{n})$ 修改，$O(1)$ 查询的数据结构解决。空间复杂度优化为 $O(n+q)$。对于另一侧同理。

总时间复杂度 $O(n\sqrt{q}+q\log n)$，空间复杂度 $O(n+q)$，可以通过所有子任务，期望得分 100 分。

3 参考资料

[1] 子串统计题解，https://loj.ac/d/3707