“回文匹配”命题报告

张景行

2022 年 10 月 30 日

1 题目

1.1 题面

小 Δ 的字符串水平终于达到了普及组水平！他学会了求一个字符串在另一个字符串中的出现次数，但他意识到了这是为人熟知的原题，不能出到互测里。过了几天，小 Δ 开始学习提高组字符串算法，比如求一个字符串的最长回文子串，但这也是为人熟知的原题。但小 Δ 觉得只需要融合一下这两道题，就不算为人熟知的原题了，他出的题目如下。

对于一个字符串 $S = a_1a_2a_3\cdots a_k$，定义字符串的长度 $|S| = k$，定义 S 的子串 $[l, r]$ 为 $S[l, r] = a_la_{l+1}\cdots a_r$，并定义其反转为 $S^R = a_ka_{k-1}\cdots a_1$

定义一个字符串 S 的回文集合是其中所有回文串的位置的集合，也就是 $P(S) = \{(l, r) \in \mathbb{Z}^2 \mid 1 \le l \le r \le |S|, S[l, r] = (S[l, r])^R\}$

定义两个字符串 S, T 回文匹配，记为 $S \approx T$，当且仅当 $P(S) = P(T)$

给定 n 个字符串 S_i，令 $f(i, j)$ 为 S_j 有多少个子串与 S_i 回文匹配，也就是 $f(i, j) = |\{(l, r) \in \mathbb{Z}^2 \mid 1 \le l \le r \le |S_j|, S_j[l, r] \approx S_i\}|$

给定 q 次询问，每次给定 i, j，求 $f(i, j)$

本题有两种输入方式，第一种是直接给出 S_i，第二种是把 S_i 给定为其前面的某一个 S_{f_i} 后面接上一个字符 c_i，见输入格式。

1.2 输入格式

第一行，三个正整数 T, n, q

若 $T = 0$，则接下来 n 行，第 i 行一个字符串 S_i

若 $T = 1$，则接下来 n 行，第 i 行一个整数 f_i，和一个字符 c_i，若 $f_i = 0$，则代表 $S_i = c_i$，否则代表 $S_i = S_{f_i}c_i$

接下来 q 行，每行两个正整数 i, j，代表一组询问。

1.3 输出格式

q 行，每行一个整数，代表一组询问的答案。

1.4 数据范围与限制

时间限制 1.5s，空间限制 1GB

$T \in \{0,1\}, 1 \le n,q \le 5 \times 10^5$，$S_i$ 非空且只包含小写英文字符，$1 \le i,j \le n$

当 $T=0$ 时，令 $L=\sum_{i=1}^{n}|S_i|$，保证 $L \le 5 \times 10^5$

当 $T=1$ 时，保证 $0 \le f_i \le i-1$

- 子任务 1（5 分）$T=0, L \le 1000$
- 子任务 2（15 分）$T=0$，所有 $|S_i|$ 全部相同。
- 子任务 3（20 分）$T=0, q=1$
- 子任务 4（20 分）$T=0$
- 子任务 5（40 分）无特殊限制

1.5 样例

1.5.1 样例 1

输入	输出
0 5 4 aba acaba acaca aa zzzzz 1 2 1 3 2 3 4 5	2 3 0 4

1.5.2 样例 2

输入	输出
0 2 1 abca jintianshikendejifengkuangxingqisivwowushi 1 2	35

1.5.3 样例 3

输入	输出
1 7 4 0 a 1 b 2 a 1 a 4 b 5 a 0 a 7 6 3 6 2 5 4 6	4 1 1 1

2 题解

2.1 算法一

考虑 $S \approx T$ 当且仅当 S, T 中每一个中心的回文串的最大长度相同，故可以先对于每一个字符串 $O(|S_i|^2)$ 地暴力求出每一个中心的最长回文串，然后注意到一个字符串的子串的最大回文中心可以简单地由原串的长度和子串的边界位置取最小值得到，故 $f(i,j)$ 可以 $O(|S_i||S_j|)$ 求得，预处理后可以 $O(1)$ 回答询问。

总复杂度 $O(L^2+q)$，期望通过子任务 1，得分 5 分。

2.2 算法二

若所有字符串的长度均相同，则 $f(i,j)=[S_i \approx S_j]$，显然地，只需要使用 Manacher 算法求出所有 S_i 每个中心最长的回文串长度，就可以预处理出 S_i 在 $\approx$ 下的的等价类，每次询问只需判断两个串是否等价即可。

使用哈希维护等价类，很容易做到 $O(L+q)$，期望通过子任务 2，得分 15 分。

2.3 算法三

考虑 $\approx$ 是否可以进行类似的 KMP 算法。

首先观察到若 $aS \approx bT$，则 $S \approx T$，若 $Sa \approx Tb$，则 $S \approx T$，也就是说，两个等价的串对应位置的子串也等价。

接下来考虑 border，若 $T=S[1,i] \approx S[|S|-i+1,|S|], T[1,j] \approx T[i-j+1,i]$，那么，$S[1,j]=T[1,j] \approx T[i-j+1,i]=S[i-j+1,i] \approx S[|S|-j+1,|S|]$，也就是 border 的 border 是 border，故 border 的结构也和普通的字符串一样。

并且，若 $Sa[1,i] \approx Sa[|S|-i+2,|S|+1]$，则 $S[1,i-1]=Sa[1,i-1] \approx Sa[|S|-i+2,|S|]=S[|S|-i+1,|S|]$，也就是 Sa 的 border 一定是从 S 的 border 上拓展一个字符得来的。

有了这两条性质，容易发现 KMP 算法所需的跳 border，以及扩展 border 都是对的，所以就可以直接套用了，而唯一需要解决的问题就是在知道 $S \approx T$ 的前提下，如何判定是否有 $Sa \approx Tb$，实际上有很多均摊 $O(1)$ 或 $O(\log)$ 的算法均可以通过，这里介绍一种可以扩展到后续子任务的方法。

定义一个字符串 S 的后缀回文长度是每一个前缀的最长回文后缀的长度 $p(S)_i = \max\{i-l+1 \mid S[l,i] = (S[l,i])^R\}$，可以证明，$p(S) = p(T) \iff S \approx T$。$S \approx T \implies p(S) = p(T)$ 是显然的，而若 $p(S) = p(T)$，从左往右加入 S, T 中的每一个字符，由于加了新的字符所形成的最长回文后缀相同，而类似 Manacher 算法，更短的回文后缀可以通过最长的那一个翻转到回文结构已知的区域，故是唯一确定的，所以都是相同的。

所以，只需要求模式串 S 的某一个前缀的最长回文后缀长度，和 S, T 的某些子串的最长回文后缀长度。S 的前缀的最长回文后缀长度可以简单的通过 Manacher 预处理，而注意到，KMP 过程中，要求最长后缀长度的子串的左右端点一定单调不降。而容易证明，在左端点或右端点往右移动的时候，最长回文后缀的中心一定不会向左移动，所以只需要从上一个最长回文后缀的位置暴力寻找最长的，并通过 Manacher 预处理每个中心的最长回文子串来判定是否是回文后缀，总共就是线性的了。

注意到 $q = 1$ 就是在做一次某一个模式串和一个文本串的匹配，直接套用这个修改的 KMP 算法即可做到 $O(L)$，期望通过子任务 3，得分 20 分。

2.4 算法四

考虑 $T = 0$，注意到如果等价关系是正常的字符串等价关系，这是一个经典的 AC 自动机问题，只需要对所有字符串建出 AC 自动机后，求 S_i 对应节点的 fail 子树中，有多少 S_j 的前缀即可，可以将询问离线，然后使用树状数组维护单点加子树求和，或者使用虚树等任何其它做法，简单地做到 $O((n+q)\log n)$，所以问题就是如何建这样一个 AC 自动机。

考虑算法三中我们发现的 $\approx$ 的等价条件 $p(S) = p(T)$，自然可以想到，直接将字符串 S 当成 $p(S)$，然后对所有 $p(S)$ 建 AC 自动机。但是需要注意，虽然 $p(Sa)[1, |S|] = p(S)$，但是 $p(aS)[2, |S|+1] \neq p(S)$，也就是说 S 的后缀的 p 并不是 $p(S)$ 的后缀。我们现在开始把一个 $p(S)$ 对应的 S 的后缀的 p 称之为 $p(S)$ 的“后缀”，注意到若 $T \approx S[i, |S|], U \approx T[j, |T|]$，那么 $U \approx T[j, |T|] \approx S[j, |S|]$，也就是后缀的后缀还是后缀，所以在这种“后缀”关系下，fail 还是可以定义的，但是不能像普通 AC 自动机一样求。

并且也容易发现，在 $\approx$ 下，若 Ta 是 Sa 的后缀，那么 T 是 S 的后缀。所以和正常 AC 自动机一样，一个节点的儿子的 fail，一定是其父亲的 fail 链上的一个点的儿子。

所以说，首先我们可以先对所有字符串做 Manacher，然后求出它们的

$p(S)$，然后把所有 $p(S)$ 建一个 Trie，注意 $p(S)$ 的值域是 L，所以需要使用 STL map，或哈希表维护儿子。接下来是 BFS 这个 Trie 求出 fail，注意由于字符集很大，而且求后缀还不是正常的后缀，实际上并不可以建出真正意义上的 AC 自动机，但是模仿 KMP 算法，fail 还是可以直接求的，只需要暴力地遍历其父亲的 fail，然后检查其儿子中是否有可以拓展的节点即可。判断是否可以扩展，可以找到 Trie 上的这个节点对应的原字符串，然后套用算法三的做法。

注意对于插入的每一个字符串，其过程和运行 KMP 是完全一致的，只不过是在整棵字典树上找 fail，所以无论是求最长回文后缀的子串的左右端点的单调性，还是跳 fail 的均摊复杂度分析都和 KMP 是一样的。由于需要 map 来查询 Trie 的儿子，所以是 $O(L \log L)$ 的。

总复杂度 $O(L \log L + (n+q) \log n)$，期望通过子任务 1,2,3,4，得分 60 分。

2.5 算法五

最后，对于 $T = 1$ 的情况。若我们还是通过了某些手段得到了这个 Trie 的 AC 自动机，询问仍然是好处理的，只需要按照 Trie 树的 dfs 序枚举所有的 j，然后使用树状数组维护 fail 树的单点加子树求和做到 $O((n+q) \log n)$。

若是一个 Trie，则所有串的 p 就不能使用 Manacher 来求了，考虑回文自动机，注意回文自动机天然的求的就是最长回文后缀，只需要 dfs 这个 Trie 的同时建立回文自动机，然后回溯的时候回滚修改就好。但有一个小问题就是常见的回文自动机的复杂度是均摊的，这很容易解决，因为均摊的部分就是求一个点的 fail 链上的回文后缀中，第一个左边的字符是 c 的点。注意到除了最长的回文后缀，剩下的所有回文后缀左边的字符是能直接确定的，所以只需要维护一个 $t_{x,c}$ 代表 x 的所有 fail（不包含自己）之中，最长的左边一个字符是 c 的节点。因为每次只有挂儿子的操作，t 可以直接从父亲继承。这样复杂度就是不均摊的 $O(n|\Sigma|)$ 了，所以可以直接在 Trie 树上用。另外注意，虽然操作回滚掉了孩子数组，但是每一个位置的 fail 树等在过程中都是可以完整的保留的，这些信息接下来会用到。

有一个重要的性质，就是对于一个序列 l，若存在若干个 S，$p(S) = l$，那么所有可能的 S 的 $\{p(Sx)\}$ 集合相同。也就是说，我们任取所有 $p(S) = l$ 作为代表元，其加一个字符后体现在 l 的字符的集合相同。这是因为给定一个 l，字符串的回文结构是确定的，考虑字符串的两个不同的回文后缀，

设为 $S[a,|S|],S[b,|S|]$，设 $a<b$，将短的回文后缀在长的上面翻转，得 $S[b-1,|S|]=S[a,a+|S|-b+1]^R$，而注意到 $S[a,a+|S|-b]=S[b,|S|]$ 是一个回文子串，由于回文结构确定，我们可以知道这个回文子串是否可以扩展，也就是 $S[a-1]$ 与 $S[a+|S|-b+1]=S[b-1]$ 是否相等，考虑把这样一个回文后缀前面那个字符称之为前驱字符。那么 S 后面加一个字符后最长回文后缀的所有取值，就是 S 的所有在与其前驱字符相同的回文后缀中长度最长的回文后缀的长度 $+2$，注意刚才说明了前驱字符的等价关系已知，所以哪些回文后缀是与其前驱相同中的最长的也已知。所以说 l 就唯一确定了 $p(Sa)$ 的集合，也就是说，对于某一个给定的 l，我们可以任选一个可能的 S，其往后加字符之后表现出的取值范围没有区别。

仔细分析一下上述求 Trie 上每一个点的最长回文后缀的方法，我们维护了一个 $t_{x,c}$，代表 x 这个点不包括自己的 fail 中，所有的前面一个字符是 c 的 fail 当中最长的一个。注意到对于一个 S，若 S 的最长回文后缀是 z，那么 Sa 的最长回文后缀只能是 $|t_{z,c}|+2$，或者 $|z|+2$，这最多 $O(|\Sigma|)$ 种取值，也就是说，对于所有 $p(S)=l$ 的 S，$p(Sa)=lx$，那么这个 x 的取值范围只有至多 $O(|\Sigma|)$，且这个取值范围可以直接从 $t_{z,c}$ 中得到。而之前已经说明了后继集合与选择的字符串无关，所以，我们使用 p 数组建立的这样一个 Trie，可以说每个节点的有效字符集大小至多是 $O(|\Sigma|)$，且我们是可以预处理出来的。

那么既然每个点有效字符集大小是 $O(|\Sigma|)$，这就意味着实际上完整的 AC 自动机是有希望可以建出来的。考虑对于一个序列 x，若我们确定了它的 fail 是“后缀”y，那么对于所有可能接在 x 后面的 c，考虑 xc 的长度为 $|y|+1$ 的后缀 r，z 的前面 $|y|$ 位显然与 y 相等，而 z 的最后一位实际上是好求的，若 $p(S)=x,p(Sd)=xc$，那么 z 的最后一位实际上就是 Sd 的最长的，长度不超过 $|y|+1$ 的回文后缀，其一定是 S 的回文后缀拓展一位得到的，所以可以在回文自动机上，从 S 的最长的回文后缀倍增到最后一个长度 $>|y|-1$ 的 fail，然后直接利用这个点的 $t_{x,c}$ 就可以求得所有 c 对应的答案。

求得了上面的东西，实际上也就说明了我们对于每一个点往下挂一个字符得到的序列，如果我们取一个长度为其 fail 第长度 $+1$ 的后缀，其相当于其 fail 后面挂上的字符是已知的。这实际上就是在把这个点的字符集对应到它的 fail 上。那么剩下的就和正常 AC 自动机一样了。

预处理字符集直接排序可以做到 $O(n|\Sigma|\log|\Sigma|)$，倍增回文自动机需要

$O(n \log n)$，总共复杂度 $O(n|\Sigma| \log |\Sigma| + (n+q) \log n)$，期望通过所有子任务，得分 100 分。

3 参考资料与后记

部分的算法来源：

Tomohiro I, Shunsuke Inenaga, Masayuki Takeda, Palindrome pattern matching, https://doi.org/10.1016/j.tcs.2012.01.047.

实际上，该论文还介绍了对这种等价关系求后缀树的方式。

OI 中已经出现过一些研究非严格相等的等价关系的字符串题，例如有一些将等价关系定义为两个字符串的字符经过置换后相等，并已经出现了研究的很深入的题目，我找了一种我没见过在 OI 中出现过的等价关系，应和着其特有的性质，结合了阿狸的打字机这一道经典的 AC 自动机题目，得到了这道题目。