回文自动机

2021-06-05 | 2022-10-09 | OI & ACM | 6
1.3k | 1 分钟

简介

和 AC 自动机长得很像的东西

在学习回文自动机之前，我们首先要知道一个定理

一个长度为 n 的字符串的本质不同的回文子串最多只有 $O(n)$ 个。

证明：

考虑以每个点为结尾的若干个回文串，能够发现只有最长的那个有贡献

其他的回文串一定作为这个最长的回文串的一个子串出现且结尾一定不是当前点

所以实际上其他的点已经被统计过了

有了这个定理之后，我们稍微对回文自动机做一些解释

回文自动机

回文自动机上每个节点代表原串中一个回文子串，每个节点所代表的子串不同

回文自动机有两部分组成：长度为奇数的回文串和长度为偶数的回文串分别为于两个回文自动机上，这两部分没有联系

回文自动机上的点之间有两种边，形如 AC 自动机，一种是转移边另一种是 $fail$ 边，串 S 有一条到 T 且字符为 c 的转移边表示 $S = cTc$，串 S 有一条到 T 的 $fail$ 边，表示 T 是 S 的最常回文后缀，当然它也是 S 的最长 $border$

性质

1. 构建回文自动机

时间复杂度 $O(n)$

```
struct PAM {
    int l, nxt[26], fail;
} T[maxn]; int top, last;
void init_PAM() {
    top = 1; last = 0;
    T[0].l = 0; T[0].fail = 1;
    T[1].l = -1; T[1].fail = -1;
}

inline int get(int i, int u) {
    while (s[i - T[u].l - 1] != s[i]) u = T[u].fail;
    return u;
}

int ans;
void insert(int i, int ch) {
    int p = get(i, last);
    if (!T[p].nxt[ch]) {
        int q = ++top;
        T[q].l = T[p].l + 2;
        T[q].fail = T[get(i, T[p].fail)].nxt[ch];
        T[p].nxt[ch] = q;
    } last = T[p].nxt[ch];
}
```

应用

例题

1. 简要题意：给定一个长度为 n 的串 S，定义一个长度为 k 区间序列 $[l_1, r_1], \cdots, [l_k, r_k]$ 为一个合法区间序列，当且仅当满足于 $l_1 < l_2 < \cdots < l_k \leq r_k < r_{m-1} < \cdots < r_1$，求有多少区间套序列满足，$\forall i \in [1, m], S[l_i .. r_i]$ 是回文串

   $n \leq 10^6$

我们考虑对于所有右端点来计数，那么为了求所有以 r 结尾的回文串，我们可以想到回文自动机，我们定义 $f_{u,0/1/2}$ 分别表示最后一个选择的区间是 u 这个区间；最后一个选择的区间是 u 的 $border$；最后一个选择的区间是 u 的子串，能够发现我们可以用通过字符边连向 u 的节点 v 和 u 的 $fail$ 边这两个节点的状态来进行转移

时间复杂度 $O(n)$

校内赛 T3 数学分析

---------- 本文结束 感谢阅读 ----------

Tech # 回文自动机

CF 1511E Colorings and Dominoes

CF 952C Big Secret