Luogu P6292 区间本质不同子串个数

2022-08-20 | OI & ACM | 5
4.6k | 4 分钟

题目描述

https://www.luogu.com.cn/problem/P6292

简要题意：给定一个长度为 n 的字符串 S，现在有 m 次询问，每次询问给定区间 $[l,r]$，求 $S[l..r]$ 有多少本质不同的子串

$n \leq 10^5, m \leq 2 \times 10^5$

Solution

我们首先考虑一个弱化的问题，每次求结束位置在 $[l,r]$ 的本质不同的子串，我们对 S 建立后缀自动机，然后我们找到 1 到 n 所对应的 n 个 np 节点，那么区间 $[l,r]$ 的查询相当于是查询 $[l,r]$ 的 np 节点在 $parent$ 树上树链的并，对于这个问题我们还是考虑离线，我们考虑做扫描线，将树上每个节点的值挂在子树内最大 np 节点上，那么每次查询相当于是区间查询，每次修改相当于是将一个点到根的路径上的所有节点重新染色，注意到对于同一个颜色段的操作相同，而且这个操作即是 LCT 的 $access$ 操作，所以我们可以用 LCT 维护同色链，用树状数组维护权值，总时间复杂度 $O(n \log^2 n + m \log n)$

然后我们回到这个问题，我们现在的限制不只是结束位置在 $[l,r]$，还要保证起始位置也在 $[l,r]$，但其实操作类似，对于 $parent$ 树上的点，不妨设这个点所表示的串长在 $[a,b]$ 以及当前扫描线扫到 r，我们只需要将这个点的每个串的贡献挂在 $[r-a+1, r-b+1]$ 即可，同时在 $parent$ 树上，树链所表示的串长是一个连续区间，可以维护，那么我们现在每次修改相当于是区间加等差数列单点查询，我们可以将其转换成区间加区间查询，这样我们仍然可以使用树状数组来维护，时间复杂度 $O(n \log^2 n + m \log n)$

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
#define maxn 100010
#define Maxn 200010
#define maxm 200010
#define ll long long
#define lowbit(i) ((i) & (-i))
using namespace std;

int n, m, a[maxn];
char s[maxn];

namespace Bit {
    ll B1[maxn], B2[maxn];

    void add(int x, int v) {
        for (int i = x; i <= n; i += lowbit(i))
            B1[i] += v, B2[i] += x * v;
    }
    ll get_sum(int x) {
        ll s = 0;
        for (int i = x; i; i -= lowbit(i))
            s += (x + 1) * B1[i], s -= B2[i];
        return s;
    }
    void update(int l, int r, int v) { add(l, v); add(r + 1, -v); }
    ll query(int l, int r) { return get_sum(r) - get_sum(l - 1); }
}

namespace SAM {
    struct node {
        int sz, L, l, nxt[26];
    } T[Maxn]; int f[Maxn], top, last, rt;
    void init() {
        for (int i = 1; i <= top; ++i) {
            T[i].sz = T[i].L = T[i].l = f[i] = 0;
            fill(T[i].nxt, T[i].nxt + 26, 0);
        }
        rt = last = top = 1;
        T[rt].L = T[rt].l = f[rt] = 0;
    }
    void extend(int ch, int k) {
        int np = ++top, p = last; last = np;
        T[np].L = T[p].L + 1; a[k] = np;
```

```
        while (p && !T[p].nxt[ch]) T[p].nxt[ch] = np, p = f[p];
        if (!p) return f[np] = rt, void();
        int q = T[p].nxt[ch];
        if (T[q].L - 1 == T[p].L) f[np] = q;
        else {
            int nq = ++top; T[nq].L = T[p].L + 1; f[nq] = f[q];
            for (int i = 0; i < 26; ++i) T[nq].nxt[i] = T[q].nxt[i];
            while (p && T[p].nxt[ch] == q) T[p].nxt[ch] = nq, p = f[p];
            f[np] = f[q] = nq;
        }
    }
    int tax[maxn], tp[Maxn];
    void rsort(int n) {
        for (int i = 1; i <= n; ++i) tax[i] = 0;
        for (int i = 1; i <= top; ++i) ++tax[T[i].L];
        for (int i = 1; i <= n; ++i) tax[i] += tax[i - 1];
        for (int i = 1; i <= top; ++i) tp[tax[T[i].L]--] = i;
        for (int i = top, u = tp[i]; i > 1; u = tp[--i]) T[u].l = T[f[u]].L + 1;
    }
}

namespace LCT {
#define lc T[i].ch[0]
#define rc T[i].ch[1]
    struct LinkCutTree {
        int vmn, vmx, valmn, valmx, ch[2];
        bool rev; int c;
    } T[Maxn]; int f[Maxn];
    void init(int n) {
        T[0].vmn = 1e9; T[0].vmx = 0;
        for (int i = 1; i <= n; ++i) {
            T[i].valmn = T[i].vmn = SAM::T[i].l;
            T[i].valmx = T[i].vmx = SAM::T[i].L;
            f[i] = SAM::f[i];
        }
        T[1].valmn = T[1].vmn = 1e9;
        T[1].valmx = T[1].vmx = 0;
    }
    inline int get(int i) {
        if (T[f[i]].ch[0] == i) return 0;
        if (T[f[i]].ch[1] == i) return 1;
        return -1;
    }
    inline void maintain(int i) {
        T[i].vmn = min({ T[i].valmn, T[lc].vmn, T[rc].vmn });
        T[i].vmx = max({ T[i].valmx, T[lc].vmx, T[rc].vmx });
```

```
93      }
94      inline void setc(int i, int c) {
95          if (!i) return ;
96          T[i].c = c;
97      }
98      inline void setr(int i) {
99          if (!i) return ;
100         T[i].rev ^= 1; swap(lc, rc);
101     }
102     inline void push(int i) {
103         bool &rev = T[i].rev; int &c = T[i].c;
104         if (rev) setr(lc), setr(rc);
105         setc(lc, c), setc(rc, c);
106         rev = 0;
107     }
108     inline void rotate(int x) {
109         int fa = f[x], ffa = f[f[x]], wx = get(x);
110         if (~get(fa)) T[ffa].ch[T[ffa].ch[1] == fa] = x;
111         f[x] = ffa; f[fa] = x; f[T[x].ch[wx ^ 1]] = fa;
112         T[fa].ch[wx] = T[x].ch[wx ^ 1]; T[x].ch[wx ^ 1] = fa;
113         maintain(fa); maintain(x);
114     }
115     void clt(int i) {
116         static int st[maxn], top;
117         st[top = 1] = i;
118         while (~get(i)) st[++top] = i = f[i];
119         while (top) push(st[top--]);
120     }
121     void Splay(int i) {
122         clt(i);
123         for (int fa = f[i]; ~get(i); rotate(i), fa = f[i])
124             if (~get(fa)) rotate(get(i) == get(fa) ? fa : i);
125     }
126     void access(int i, int c) {
127         for (int p = 0; i; i = f[p = i]) {
128             Splay(i);
129             rc = 0, maintain(i);
130             if (T[i].c && i != 1) Bit::update(T[i].c - T[i].vmx + 1, T[i].c - T[i
131             T[i].c = c, rc = p, maintain(i);
132         }
133     }
134 }
135
136 vector<pair<int, int>> A[maxn];
137 ll ans[maxm];
138
```

```cpp
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr); cout.tie(nullptr);

    cin >> s + 1 >> m; n = strlen(s + 1); SAM::init();
    for (int i = 1; i <= n; ++i) SAM::extend(s[i] - 'a', i);
    SAM::rsort(n); LCT::init(SAM::top);
    for (int i = 1; i <= m; ++i) {
        int x, y; cin >> x >> y;
        A[y].emplace_back(x, i);
    }
    for (int i = 1; i <= n; ++i) {
        LCT::access(a[i], i); Bit::update(1, i, 1);
        for (auto [l, id] : A[i]) ans[id] = Bit::query(l, i);
    }
    for (int i = 1; i <= m; ++i) cout << ans[i] << "\n";
    return 0;
}

```

---------- 本文结束 感谢阅读 ----------

后缀自动机 # LCT

Luogu P7880 [Ynoi2006] rldcot

bzoj 3569 DZY Loves Chinese II