Standard Code Library

ONGLU

North Eastern University

August 2021

Contents

初始化

数据结构

轻重链剖分

```cpp
void dfs1(int x, int pre) {
    siz[x] = 1; mson[x] = 0;
    dth[x] = dth[pre] + 1;
    fa[x] = pre;
    for(auto y : son[x]) if(y != pre) {
        dfs1(y, x);
        siz[x] += siz[y];
        if(!mson[x] || siz[y] > siz[mson[x]])
            mson[x] = y;
    }
}
void dfs2(int x, int pre, int ntp) {
    id[x] = ++idcnt;
    ltp[x] = ntp;
    if(mson[x]) dfs2(mson[x], x, ntp);
    for(auto y : son[x]) {
        if(y == mson[x] || y == pre) continue;
        dfs2(y, x, y);
    }
}
void link_modify(int x, int y, int z) {
    z %= mod;
    while(ltp[x] != ltp[y]) {
        dth[ltp[x]] < dth[ltp[y]] && (x ^= y ^= x ^= y);
        modify(1, n, id[ltp[x]], id[x], 1, z);
        x = fa[ltp[x]];

    }
    dth[x] < dth[y] && (x ^= y ^= x ^= y);
    modify(1, n, id[y], id[x], 1, z);
}
int link_query(int x, int y) {
    int ans = 0;
    while(ltp[x] != ltp[y]) {
        dth[ltp[x]] < dth[ltp[y]] && (x ^= y ^= x ^= y);
        ans = (1ll * ans + query(1, n, id[ltp[x]], id[x], 1)) % mod;
        x = fa[ltp[x]];
    }
    dth[x] < dth[y] && (x ^= y ^= x ^= y);
    ans = (1ll * ans + query(1, n, id[y], id[x], 1)) % mod;
    return ans;
}
```

数学

图论

计算几何

字符串

字串哈希

```cpp
namespace String {
    const int x = 135;
    const int p1 = 1e9 + 7, p2 = 1e9 + 9;
    ull xp1[N], xp2[N], xp[N];
    void init_xp() {
        xp1[0] = xp2[0] = xp[0] = 1;
        for(int i = 1; i < N; i++) {
```

```cpp
            xp1[i] = xp1[i - 1] * x % p1;
            xp2[i] = xp2[i - 1] * x % p2;
            xp[i] = xp[i - 1] * x;
        }
    }
    struct HashString {
        char s[N];
        int length, subsize;
        bool sorted;
        ull h[N], hl[N];
        ull init(const char *t) {
            if(xp[0] != 1) init_xp();
            length = strlen(t);
            strcpy(s, t);
            ull res1 = 0, res2 = 0;
            h[length] = 0;
            for(int j = length - 1; j >= 0; j--) {
            #ifdef ENABLE_DOUBLE_HASH
                res1 = (res1 * x + s[j]) % p1;
                res2 = (res2 * x + s[j]) % p2;
                h[j] = (res1 << 32) | res2;
            #else
                res1 = res1 * x + s[j];
                h[j] = res1;
            #endif
            }
            return h[0];
        }
        //获取子串哈希，左闭右开
        ull get_substring_hash(int left, int right)  {
            int len = right - left;
        #ifdef ENABLE_DOUBLE_HASH
            unsigned int mask32 = ~(0u);
            ull left1 = h[left] >> 32, right1 = h[right] >> 32;
            ull left2 = h[left] & mask32, right2 = h[right] & mask32;
            return ((((left1 - right1 * xp1[len] % p1 + p1) % p1) << 32) |
                    (((left2 - right2 * xp2[len] % p2 + p2) % p2)));
        #else
            return h[left] - h[right] * xp[len];
        #endif
        }
        void get_all_subs_hash(int sublen) {
            subsize = length - sublen + 1;
            for (int i = 0; i < subsize; ++i)
                hl[i] = get_substring_hash(i, i + sublen);
            sorted = 0;
        }

        void sort_substring_hash() {
            sort(hl, hl + subsize);
            sorted = 1;
        }

        bool match(ull key) const {
            if (!sorted) assert (0);
            if (!subsize) return false;
            return binary_search(hl, hl + subsize, key);
        }
    };
}
```

Trie

```cpp
namespace trie {
    int t[N][26], sz, ed[N];
    int _new() {
        sz++;
        memset(t[sz], 0, sizeof(t[sz]));
        return sz;
    }
    void init() {
```

```
9          sz = 0;
10         _new();
11         memset(ed, 0, sizeof(ed));
12     }
13     void Insert(char *s, int n) {
14         int u = 1;
15         for(int i = 0; i < n; i++) {
16             int c = s[i] - 'a';
17             if(!t[u][c]) t[u][c] = _new();
18             u = t[u][c];
19         }
20         ed[u]++;
21     }
22     int find(char *s, int n) {
23         int u = 1;
24         for(int i = 0; i < n; i++) {
25             int c = s[i] - 'a';
26             if(!t[u][c]) return -1;
27             u = t[u][c];
28         }
29         return u;
30     }
31 }
```

KMP 算法

```
1  namespace KMP {
2      void get_next(char *t, int m, int *nxt) {
3          int j = nxt[0] = 0;
4          for(int i = 1; i < m; i++) {
5              while(j && t[i] != t[j]) j = nxt[j - 1];
6              nxt[i] = j += (t[i] == t[j]);
7          }
8      }
9      vector<int> find(char *t, int m, int *nxt, char *s, int n) {
10         vector<int> ans;
11         int j = 0;
12         for(int i = 0; i < n; i++) {
13             while(j && s[i] != t[j]) j = nxt[j - 1];
14             j += s[i] == t[j];
15             if(j == m) {
16                 ans.push_back(i - m + 1);
17                 j = nxt[j - 1];
18             }
19         }
20         return ans;
21     }
22 }
```

manacher 算法

```
1  namespace manacher {
2      char s[N];
3      int p[N], len;
4      void getp(string tmp) {
5          len = 0;
6          for(auto x : tmp) {
7              s[len++] = '#';
8              s[len++] = x;
9          }
10         s[len++] = '#';
11         memset(p, 0, sizeof(int) * (len + 10));
12         int c = 0, r = 0;
13         for(int i = 0; i < len; i++) {
14             if(i <= r) p[i] = min(p[2 * c - i], r - i);
15             else p[i] = 1;
16             while(i - p[i] >= 0 && i + p[i] < len && s[i - p[i]] == s[i + p[i]])
17                 p[i]++;
18             if(i + p[i] - 1 > r) {
19                 r = i + p[i] - 1;
```

```
20                    c = i;
21                }
22            }
23            for(int i = 0; i < len; i++) p[i]--;
24        }
25        void getp(char *tmp, int n) {
26            len = 0;
27            for(int i = 0; i < n; i++) {
28                s[len++] = '#';
29                s[len++] = tmp[i];
30            }
31            s[len++] = '#';
32            memset(p, 0, sizeof(int) * (len + 10));
33            int c = 0, r = 0;
34            for(int i = 0; i < len; i++) {
35                if(i <= r) p[i] = min(p[2 * c - i], r - i);
36                else p[i] = 1;
37                while(i - p[i] >= 0 && i + p[i] < len && s[i - p[i]] == s[i + p[i]])
38                    p[i]++;
39                if(i + p[i] - 1 > r) {
40                    r = i + p[i] - 1;
41                    c = i;
42                }
43            }
44            for(int i = 0; i < len; i++) p[i]--;
45        }
46        int getlen() {
47            return *max_element(p, p + len);
48        }
49        int getlen(string s) {
50            getp(s);
51            return getlen();
52        }
53    }
```

AC 自动机

```
1    struct ac_automaton {
2        int t[N][26], danger[N], tot, fail[N];
3        int dp[N][N];
4        void init() {
5            tot = -1;
6            _new();
7        }
8        int _new() {
9            tot++;
10            memset(t[tot], 0, sizeof(t[tot]));
11            danger[tot] = 0;
12            fail[tot] = 0;
13            return tot;
14        }
15        void Insert(const char *s) {
16            int u = 0;
17            for(int i = 0; s[i]; i++) {
18                if(!t[u][mp[s[i]]]) t[u][s[i] - 'a'] = _new();
19                u = t[u][mp[s[i]]];
20            }
21            danger[u] = 1;
22        }
23        void build() {
24            queue<int> q;
25            for(int i = 0; i < 26; i++) {
26                if(t[0][i]) {
27                    fail[i] = 0;
28                    q.push(t[0][i]);
29                }
30            }
31            while(q.size()) {
32                int u = q.front(); q.pop();
33                danger[u] |= danger[fail[u]];
34                for(int i = 0; i < 26; i++) {
```

```
35          if(t[u][i]) {
36              fail[t[u][i]] = t[fail[u]][i];
37              q.push(t[u][i]);
38          } else t[u][i] = t[fail[u]][i];
39      }
40  }
41  }
42  int query(const char *s) {
43      memset(dp, 0x3f, sizeof(dp));
44      int n = strlen(s);
45      dp[0][0] = 0;
46      for(int i = 0; i < n; i++) {
47          for(int j = 0; j <= tot; j++) if(!danger[j]) {
48              for(int k = 0; k < 26; k++) if(!danger[t[j][k]]) {
49                  dp[i + 1][t[j][k]] = min(dp[i + 1][t[j][k]], dp[i][j] + (s[i] - 'a' != k));
50              }
51          }
52      }
53      int ans = 0x3f3f3f3f;
54      for(int i = 0; i <= tot; i++) if(!danger[i]) {
55          ans = min(ans, dp[n][i]);
56      }
57      return ans == 0x3f3f3f3f ? -1 : ans;
58  }
59  };
```

杂项

int128

```
1   typedef __uint128_t u128;
2   inline u128 read() {
3       static char buf[100];
4       scanf("%s", buf);
5       // std::cin >> buf;
6       u128 res = 0;
7       for(int i = 0;buf[i];++i) {
8           res = res << 4 | (buf[i] <= '9' ? buf[i] - '0' : buf[i] - 'a' + 10);
9       }
10      return res;
11  }
12  inline void output(u128 res) {
13      if(res >= 16)
14          output(res / 16);
15      putchar(res % 16 >= 10 ? 'a' + res % 16 - 10 : '0' + res % 16);
16      //std::cout.put(res % 16 >= 10 ? 'a' + res % 16 - 10 : '0' + res % 16);
17  }
```

tips:

- 如果使用 sort 比较两个函数，不能出现 $a < b$ 和 $a > b$ 同时为真的情况，否则会运行错误。