Some Additions For 2021 School Competation

CandyOre

HIT

December 29, 2021

Contents

一些补充

整体 template

```cpp
#include <cstdio>
#include <cmath>
#include <iostream>
#include <chrono>
#include <algorithm>
#include <vector>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <map>
#include <set>
#include <queue>
#include <stack>
using namespace std;

// Define abbr. here.
#define IL inline
#define fi first
#define se second
#define mk make_pair
#define pb push_back
#define SZ(x) (int)(x).size()
#define ALL(x) (x).begin(), (x).end()
#define dbg1(x) cout << #x << " = " << x << ", "
#define dbg2(x) cout << #x << " = " << x << endl
#define rep(i,n) for (int i=0;i<(int)(n);++i)
#define rep1(i,n) for (int i=1;i<=(int)(n);++i)
using ll = long long;
using ld = long double;
using ull = unsigned long long;
using pii = pair<int,int>;
using umii = unordered_map<int,int>;
using v1i = vector<int>;
using v1l = vector<ll>;
using v1b = vector<bool>;

// Define const. here.
// #define INT_MAX 0x7fffffff
// #define LLONG_MAX 0x7fffffffffffffff
const double pi = acos(-1);
const ll mod = 998244353;
const double eps = 1e-10;

// Timer
class TimerClock {
public:
    TimerClock() {
        update();
    }

    ~TimerClock() {}

    void update() {
        _start = chrono::high_resolution_clock::now();
    }

    double getTimerSecond() {
        return getTimerMicroSec()*0.000001;
    }

    void printTimerSecond() {
        cout << getTimerSecond() << "s\n";
    }

    double getTimerMilliSec() {
        return getTimerMicroSec()*0.001;
    }
```

```
68
69      void printTimerMillSec() {
70          cout << getTimerMilliSec() << "ms\n";
71      }
72
73      long long getTimerMicroSec() {
74          return chrono::duration_cast<chrono::microseconds>
75              (chrono::high_resolution_clock::now() - _start).count();
76      }
77
78  private:
79      chrono::time_point<chrono::high_resolution_clock> _start;
80  };
81
82  // TimerClock TC;
83
84
85  // Montgomery
86  template <uint32_t base>
87  struct Montgomery {
88      using i32 = int32_t;
89      using u32 = uint32_t;
90      using u64 = uint64_t;
91
92      static constexpr u32 mod() {
93          return base;
94      }
95
96      static constexpr u32 np = []() {
97          u32 x = base;
98          for (int i = 0; i < 4; ++i) {
99              x *= 2u - base * x;
100         }
101         return -x;
102     }();
103     static constexpr u32 r2 = -(u64)(base) % base;
104
105     static_assert(base < (1u << 30));
106     static_assert(base * np + 1 == 0);
107
108     static u32 reduce(u64 x) {
109         return (x + (u64)((u32)x * np) * base) >> 32;
110     }
111
112     u32 x;
113     Montgomery(): x(0) {}
114     constexpr Montgomery(long long y): x(y ? reduce((u64)(y % base + base) * r2) : 0) {}
115
116     Montgomery& operator +=(const Montgomery& ot) {
117         if ((i32)(x += ot.x - 2 * base) < 0) {
118             x += 2 * base;
119         }
120         return *this;
121     }
122
123     Montgomery& operator -=(const Montgomery& ot) {
124         if ((i32)(x -= ot.x) < 0) {
125             x += 2 * base;
126         }
127         return *this;
128     }
129
130     Montgomery& operator *=(const Montgomery& ot) {
131         x = reduce((u64)x * ot.x);
132         return *this;
133     }
134
135     Montgomery& operator /=(const Montgomery& ot) {
136         return *this *= ot.inverse();
137     }
138
```

```
    friend Montgomery operator +(Montgomery a, const Montgomery& b) {
        a += b;
        return a;
    }

    friend Montgomery operator -(Montgomery a, const Montgomery& b) {
        a -= b;
        return a;
    }

    friend Montgomery operator *(Montgomery a, const Montgomery& b) {
        a *= b;
        return a;
    }

    friend Montgomery operator /(Montgomery a, const Montgomery& b) {
        a /= b;
        return a;
    }

    Montgomery operator -() const {
        return Montgomery() - *this;
    }

    u32 get() const {
        u32 res = reduce(x);
        return res < base ? res : res - base;
    }

    u32 operator ()() const {
        return get();
    }

    Montgomery inverse() const {
        return pow(base - 2);
    }

    Montgomery pow(int64_t p) const {
        if (p < 0) {
            return pow(-p).inverse();
        }
        Montgomery res = 1;
        Montgomery a = *this;
        while (p) {
            if (p & 1) {
                res *= a;
            }
            p >>= 1;
            a *= a;
        }
        return res;
    }

    friend istream& operator >>(istream& istr, Montgomery& m) {
        long long x;
        istr >> x;
        m = Montgomery(x);
        return istr;
    }

    friend ostream& operator <<(ostream& ostr, const Montgomery& m) {
        return ostr << m.get();
    }

    bool operator ==(const Montgomery& ot) const {
        return (x >= base ? x - base : x) == (ot.x >= base ? ot.x - base : ot.x);
    }

    bool operator !=(const Montgomery& ot) const {
        return (x >= base ? x - base : x) != (ot.x >= base ? ot.x - base : ot.x);
    }
```

```
    explicit operator int64_t() const {
        return x;
    }

    explicit operator bool() const {
        return x;
    }
};

using mt = Montgomery<mod>;


// Read Functions

// inline ll read(){
//     ll x = 0, f = 1;char ch = getchar();
//     while(ch > '9' || ch < '0'){if(ch == '-') f = -1;ch = getchar();}
//     while(ch >= '0' && ch <= '9'){x = x * 10 + ch -'0';ch = getchar();}
//     return x * f;
// }

template <typename T> vector<T> readVector(int n) {
    vector<T> res(n);
    for (int i = 0 ; i < n ; i++) cin >> res[i];
    return res;
}


// Define struct and reload " >> " here.
struct edge {
    int v1,v2,len;
};

istream& operator >>(istream& is, edge &e) {
    is >> e.v1 >> e.v2 >> e.len; return is;
}

struct pnt {
    int x,y;
};

istream& operator >>(istream& is, pnt &p) {
    is >> p.x >> p.y;
    return is;
}


#ifdef ONLINE_JUDGE
const bool OJ = 1;
#else
const bool OJ = 0;
#endif


// Define variables and functions here
int n;


void init(){

}

void solve(int case_no){

}

const bool MulCase = 0;

signed main(){
    std::ios::sync_with_stdio(false);
```

```
    std::cin.tie(0);std::cout.tie(0);

    if(!OJ){
        freopen("try.in", "r", stdin);
        freopen("try.out", "w", stdout);
    }

    int T = 1;
    if(MulCase) cin >> T;
    init();
    rep(i,T) solve(i);

    return 0;
}
```

线段树

```
const int N = 2e5 + 10;

#define lson rt << 1      // == rt * 2   左儿子
#define rson rt << 1 | 1  // == rt * 2 + 1 右儿子
#define int_mid int mid = tree[rt].l + tree[rt].r >> 1

int A[N]; // 初始值

struct node {
    int l, r;
    ll val, lazy;
} tree[N * 4];

void push_up(int rt) {
    // tree[rt].val = min(tree[lson].val, tree[rson].val);
    tree[rt].val = max(tree[lson].val, tree[rson].val);
    // tree[rt].val = tree[lson].val + tree[rson].val;
}

void push_down(int rt) {
    if (tree[rt].lazy) {
        tree[lson].lazy += tree[rt].lazy;
        tree[rson].lazy += tree[rt].lazy;
        { // 维护最大最小值
            tree[lson].val += tree[rt].lazy;
            tree[rson].val += tree[rt].lazy;
        }
        // { // 维护和
        //  int l = tree[rt].l, r = tree[rt].r;
        //  int mid = l + r >> 1;
        //  tree[lson].val += 1ll * (mid - l + 1) * tree[rt].lazy;
        //  tree[rson].val += 1ll * (r - mid) * tree[rt].lazy;
        // }
        tree[rt].lazy = 0;
    }
}

void build(int rt, int l, int r) { // 建树
    tree[rt].l = l, tree[rt].r = r;
    tree[rt].lazy = 0;
    if (l == r) {
        tree[rt].val = A[l]; // 给定一个初始值
        return;
    } else {
        int mid = l + r >> 1; // (l + r) / 2
        build(lson, l, mid);
        build(rson, mid + 1, r);
        push_up(rt);
    }
}

void update_point(int rt, int pos, ll val) { // 单点更新
    if (tree[rt].l == tree[rt].r && pos == tree[rt].l) {
        tree[rt].val += val;
```

```
        return;
    }
    int_mid;
    if (pos <= mid) update_point(lson, pos, val);
    else update_point(rson, pos, val);
    push_up(rt);
}

void update_interval(int rt, int l, int r, ll val) { // 区间更新
    if (l <= tree[rt].l && r >= tree[rt].r) {
        tree[rt].lazy += val;
        tree[rt].val += val; // 维护最大最小值
        // tree[rt].val += 1ll * (tree[rt].r - tree[rt].l + 1) * val; // 维护和
        return;
    }
    push_down(rt);
    int_mid;
    if (l <= mid) update_interval(lson, l, r, val);
    if (r > mid) update_interval(rson, l, r, val);
    push_up(rt);
}

ll query_point(int rt, int pos) { // 单点查询
    if (tree[rt].l == tree[rt].r && tree[rt].l == pos)
        return tree[rt].val;
    push_down(rt);
    int_mid;
    if (pos <= mid) query_point(lson, pos);
    else query_point(rson, pos);
}

ll query_interval(int rt, int l, int r) { // 区间查询
    if (l <= tree[rt].l && r >= tree[rt].r)
        return tree[rt].val;
    push_down(rt);
    int_mid;
    if (r <= mid) return query_interval(lson, l, r);
    else if (l > mid) return query_interval(rson, l, r);
    else {
        //return min(query_interval(lson, l, mid), query_interval(rson, mid + 1, r));
        return max(query_interval(lson, l, mid), query_interval(rson, mid + 1, r));
        // return query_interval(lson, l, mid) + query_interval(rson, mid + 1, r);
    }
}
```

最长上升子序列

```
// vector<int> vec
vector<int> stk;
for (auto e : vec) {
    // x < y use lower_bound
    int pos = lower_bound(stk.begin(), stk.end(), e) - stk.begin();
    // x <= y use upper_bound
    int pos = upper_bound(stk.begin(), stk.end(), e) - stk.begin();
    if (pos == stk.size()) stk.push_back(e);
    else stk[pos] = e;
}
return stk.size();
```

欧拉路径

```
vector<vector<int>> path;

vector<vector<int>> Hierholzer (
    unordered_map<int,vector<int>> out_edge,
    unordered_map<int,int> id,
    unordered_map<int,int> od) {

    int s = out_edge.begin()->first;
    for(auto [p, oc] : od) {
```

```
        if (oc == id[p] + 1) {
            s = p;
            break;
        }
    }

    vector<vector<int>> ans;
    function<void(int)> dfs = [&](int u) {
        while(!out_edge[u].empty()) {
            int v = out_edge[u].back();
            out_edge[u].pop_back();
            dfs(v);
            path.push_back({u,v});
        }
    };

    dfs(s);
    reverse(ALL(ans));

    return ans;
}
```

倍增 LCA

```
vector<vector<int>> g;

const int N = 500005, BIN = 31;
int dep[N], pa[N][BIN];

void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    for(int i = 1; i < BIN; i++) pa[u][i] = pa[pa[u][i-1]][i-1];
    for(int v : g[u]) {
        if(v == fa) continue;
        dfs(v,u);
    }
}

int lca(int u, int v) {
    if(dep[u] < dep[v]) swap(u,v);
    int t = dep[u] - dep[v];
    for(int i = 0; i < BIN; i++) if (t & (1 << i)) u = pa[u][i];
    for(int i = BIN - 1; i > -1; i--) {
        int uu = pa[u][i], vv = pa[v][i];
        if(uu != vv) u = uu, v = vv;
    }
    return u == v ? u : pa[u][0];
}
```

快速幂和逆

```
const int BIN = 31;

ll qpow(ll x, int n, ll m) {
    ll r = 1; x %= m;
    for(; n; x = x * x % m, n >>= 1) if(n & 1) r = r * x % m;
    return r;
}

ll inv(ll x, ll p) { return qpow(x, p - 2, p); }
```

组合数

```
const int prodN = 5001;

ll prod[prodN] = {1};

ll power(ll x, int n) {
  ll ret = 1;
```

```
  while (n) {
    if (n & 1) ret = ret * x % mod;
    x = x * x % mod;
    n /= 2;
  }
  return ret;
}

ll inv(const ll& x) { return power(x, mod - 2); }

ll c(const int& n, int m) {
  m = min(m, n - m);
  if (m < 0) return 0;
  return prod[n] * inv(prod[m]) % mod * inv(prod[n - m]) % mod;
}

// in init()
for (int i = 1; i < prodN; i++) prod[i] = prod[i - 1] * i % mod;

// with Montgomery
const int prod_N = 5005;
mt prod[prod_N] = {1};
mt comb(const int& n, int m) {
    m = min(m, n - m);
    if (m < 0) return 0;
    return prod[n] * prod[m].inverse() * prod[n - m].inverse();
}
```

马拉车

```
string str;
vector<int> d1, d2;
void unified_manacher() {
    int n = str.size(), nn = 2 * n + 1;
    string ss(nn, '#');
    for(int i = 0; i < n; i++) ss[2 * i + 1] = str[i]; cout << ss << endl;
    vector<int> d(nn);
    for(int i = 0, l = 0, r = -1; i < nn; i++) {
        int k = (i > r) ? 1 : min(r - i + 1, d[l + r - i]);
        while(-1 < i - k && i + k < nn && ss[i - k] == ss[i + k]) {
            k++;
        }
        d[i] = k--;
        if(i + k > r) {
            r = i + k;
            l = i - k;
        }
    }
    for(int i = 0; i < n; i++) {
        d1[i] = d[2 * i + 1] / 2;
        d2[i] = d[2 * i] / 2;
    }
}
```