Ayudamemoria

My room is random Sorted

29 de septiembre de 2024

Índice

1. Estructuras de datos **1**
- 1.1. Fenwick Tree 1
- 1.2. Tabla Aditiva 1

2. Geometria **2**
- 2.1. Lower Envelope 3

3. Template **4**

4. Otros **4**
- 4.1. Fijar el numero de decimales 4
- 4.2. Criba Lineal 4

1. Estructuras de datos

1.1. Fenwick Tree

```
struct Fenwick {
    static const int sz = 1<<11;
    ll t[sz];
    void adjust(ll v, int p) {
        for (int i=p; i<sz; i+=(i&-i)) t[i]+=v; }
    ll sum(int p){
        ll s=0;
        for(int i=p; i; i-=(i&-i)) s+=t[i];
        return s;
    }
    ll sum(int a, int b) {return sum(b)-sum(a-1);}
};

struct RangeFenwick {
    Fenwick add;
    Fenwick sub;
    void adjust(ll v, int a, int b) {
        add.adjust(v,a);
        add.adjust(-v,b+1);
        sub.adjust((a-1)*v,a);
        sub.adjust(-b*v,b+1);
    }
    ll sum(int p) {
        return 1ll * p * add.sum(p) - sub.sum(p);
    }
    ll sum(int a, int b) {
        return sum(b) - sum(a-1);
    }
};

struct Fenwick2D {
    static const int sz=(1<<10);
    Fenwick t[sz];
    void adjust(int x, int y, ll v) {
        for (int i=x; i<sz; i+=(i&-i)) t[i].adjust(y,v);}
    ll sum(int x, int y) {
        ll s=0;
        for (int i=x; i; i-=(i&-i)) s += t[i].sum(y);
        return s;
    }
    ll sum(int x1, int y1, int x2, int y2) {
        ll s = sum(x2,y2)
             + ((x1>1) ? -sum(x1-1,y2) : 0)
             + ((y1>1) ? -sum(x2,y1-1) : 0)
             + ((x1>1&&y1>1) ? sum(x1-1,y1-1) : 0);
        return s;
    }
};
```

1.2. Tabla Aditiva

```
// Tablita aditiva 2D
forn (dim, 2) {
    forn (i, N) {
        forn (j, M) {
            int pi = i-(dim==0), pj = j-(dim==1);
            if (pi >= 0 && pj >= 0) {
                dp[i][j] += dp[pi][pj];
            }
        }
    }
}
// Generalizacion a 32 dimensiones para mascaras de bits
forn (i, 32) {
    forn (mask, 1<<32) {
        if ((mask>>i)&1) {
            dp[mask] += dp[mask - (1<<i)];
        }
    }
}
```

2. Geometria

```
struct Point
{
    double x, y;
    double Point::operator*(const Point &o) const {
        return x * o.x + y * o.y; }
    double Point::operator^(const Point &o) const {
        return x * o.y - y * o.x; }
    Point Point::operator-(const Point &o) const {
        return {x - o.x, y - o.y}; }
    Point Point::operator+(const Point &o) const {
        return {x + o.x, y + o.y}; }
    Point Point::operator*(const double &u) const {
        return {x * u, y * u}; }
    Point Point::operator/(const double &u) const {
        return {x / u, y / u}; }
    double Point::norm_sq() const {
        return x * x + y * y; }
    double Point::norm() const {
        return sqrt(x * x + y * y); }
};

struct Comp {
    Vector o, v;
    Comp(Vector _o, Vector _v) : o(_o), v(_v) {}
    bool half(Vector p) {
        assert(!(p.x == 0 && p.y == 0));
        return (v ^ p) < 0 || ((v ^ p) == 0 && (v * p) < 0);
    }
    bool operator()(Vector a, Vector b) {
        return mp(half(a - o), 0ll) < mp(half(b - o), ((a - o) ^ (b
            - o)));
    }
};

struct Segment {
    Vector a, b;
    long double eval() const
    { // funcion auxiliar para ordenar segmentos
        assert(a.x != b.x || a.y != b.y);
        Vector a1 = a, b1 = b;
        if (a1.x > b1.x)
            swap(a1, b1);
        assert(x >= a1.x && x <= b1.x);
        if (x == a1.x)
            return a1.y;
        if (x == b1.x)
            return b1.y;
        Vector ab = b1 - a1;
        return a1.y + (x - a1.x) * (ab.y / ab.x);
    }
    bool operator<(Segment o) const
    { // orden de segmentos en un punto (x=cte)
        return (eval() - o.eval()) < -1e-13;
    }
};

bool ccw(const Point &a, const Point &m, const Point &b) {
    return ((a - m) ^ (b - m)) > EPS; }
```

```
58
59 bool collinear(const Point &a, const Point &b, const Point &c) {
60     return fabs((b - a) ^ (c - a)) < EPS; }
61
62 double dist_sq(const Point &a, const Point &b) {
63     return (a - b).norm_sq(); }
64
65 double dist(const Point &a, const Point &b) {
66     return (a - b).norm(); }
67
68 bool in_segment(const Point &p, const Point &b, const Point &c) {
69     return fabs(dist_sq(p, b) + dist_sq(p, c) - dist_sq(b, c)) <
           EPS; }
70
71 double angle(const Point &a, const Point &m, const Point &b) {
72     Point ma = a - m, mb = b - m;
73     return atan2(ma ^ mb, ma * mb); }
74
75 void sweep_space() {
76     vector<Event> eventos; // puntos, segmentos, ...
77     sort(eventos);       // sort por x, y, ...
78     set<Info> estado;   // mantener la informacion ordenada
79     forn(i, sz(eventos)) {
80         Event &e = eventos[i];
81         process(e, estado); // procesar un evento cambia el estado
82         ans = actualizar(ans);
83 } }
84
85 vector<pt> minkowski_sum(vector<pt> p, vector<pt> q){
86     int n=SZ(p),m=SZ(q),x=0,y=0;
87     fore(i,0,n) if(p[i]<p[x]) x=i;
88     fore(i,0,m) if(q[i]<q[y]) y=i;
89     vector<pt> ans={p[x]+q[y]};
90     fore(it,1,n+m){
91         pt a=p[(x+1)%n]+q[y];
92         pt b=p[x]+q[(y+1)%m];
93         if(b.left(ans.back(),a)) ans.pb(b), y=(y+1)%m;
94         else ans.pb(a), x=(x+1)%n;
95     }
96     return ans; }
```

2.1. Lower Envelope

```
1  const ll is_query = -(1LL<<62);
2  struct Line {
3      ll m, b;
4      mutable multiset<Line>::iterator it;
5      const Line *succ(multiset<Line>::iterator it) const;
6      bool operator<(const Line & rhs) const {
7          if (rhs.b != is_query) return m < rhs.m;
8          const Line *s = succ(it);
9          if (!s) return 0;
10         ll x = rhs.m;
11         return b - s->b > (s->m - m) * x;
12     }
13 };
14 struct HullDynamic : public multiset<Line> {
15     bool bad(iterator y) {
16         iterator z = next(y);
17         if (y == begin()) {
18             if (z == end()) return 0;
19             return y->m == z->m && y->b >= z->b;
20         }
21         iterator x = prev(y);
22         if (z == end()) return y->m == x->m && y->b >= x->b;
23         return (x->m-z->m)*(z->b-y->b) >= (z->b-x->b)*(y->m-z->m);
24     }
25     iterator next(iterator y) {return ++y;}
26     iterator prev(iterator y) {return --y;}
27     void insert_line(ll m, ll b) {
28         iterator y = insert((Line) {m, b});
29         y->it = y;
30         if (bad(y)) {erase(y); return;}
31         while (next(y) != end() && bad(next(y))) erase(next(y));
32         while (y != begin() && bad(prev(y))) erase(prev(y));
33     }
34     ll eval(ll x) {
35         Line l = *lower_bound((Line) {x, is_query});
36         return l.m * x + l.b;
37     }
38 } h;
```

```
const Line *Line::succ(multiset<Line>::iterator it) const {
    return (++it==h.end() ? NULL : &*it); }
```

3. Template

```
#include <bits/stdc++.h>
using namespace std;
#define dprint(v) cout << #v "=" << v << endl //;)
#define forr(i, a, b) for (int i = (a); i < (b); i++)
#define forn(i, n) forr(i, 0, n)
#define dforn(i, n) for (int i = n - 1; i >= 0; i--)
#define forall(it, v) for (auto it = v.begin(); it != v.end();
    ++it)
#define sz(c) ((int)c.size())
#define zero(v) memset(v, 0, sizeof(v))
#define pb push_back
#define fst first
#define snd second
#define mid(a,b) ((a+b)>>1)
#define mp make_pair
#define all(v) begin(v),end(v)
#define endl '\n'
typedef long long ll;
typedef pair<int, int> ii;
typedef vector<int> vi;

int main(int argc, char **argv){
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    if(argc == 2) freopen(argv[1], "r", stdin);

}
```

4. Otros

4.1. Fijar el numero de decimales

```
cout.precision(7); fixed(cout);
cout << x << " " << y;
// otra forma
cout.precision(7);
cout << fixed << x << " " << fixed << y;
```

4.2. Criba Lineal

```
const int N = 10000000;
vector<int> lp(N+1);
vector<int> pr;

for (int i=2; i <= N; ++i) {
    if (lp[i] == 0) {
        lp[i] = i;
        pr.push_back(i);
    }
    for (int j = 0; i * pr[j] <= N; ++j) {
        lp[i * pr[j]] = pr[j];
        if (pr[j] == lp[i]) {
            break;
        }
    }
}
```