

Ayudamemoria

My room is random Sorted

14 de septiembre de 2024

Índice

1. Estructuras de datos

- 1.1. Fenwick Tree
- 1.2. Tabla Aditiva

2. Geometria

3. Template

4. Otros

- 4.1. Fijar el numero de decimales

1. Estructuras de datos

1.1. Fenwick Tree

```
1 struct Fenwick {
2     static const int sz = 1<<11;
3     ll t[sz];
4     void adjust(ll v, int p) {
5         for (int i=p; i<sz; i+=(i&-i)) t[i]+=v; }
6     ll sum(int p){
7         ll s=0;
8         for(int i=p; i; i-=(i&-i)) s+=t[i];
9         return s;
10    }
11    ll sum(int a, int b) {return sum(b)-sum(a-1);}
12 };
13
14 struct RangeFenwick {
```

```
15     Fenwick add;
16     Fenwick sub;
17     void adjust(ll v, int a, int b) {
18         add.adjust(v,a);
19         add.adjust(-v,b+1);
20         sub.adjust((a-1)*v,a);
21         sub.adjust(-b*v,b+1);
22     }
23     ll sum(int p) {
24         return 1ll * p * add.sum(p) - sub.sum(p);
25     }
26     ll sum(int a, int b) {
27         return sum(b) - sum(a-1);
28     }
29 };
30
31 struct Fenwick2D {
32     static const int sz=(1<<10);
33     Fenwick t[sz];
34     void adjust(int x, int y, ll v) {
35         for (int i=x; i<sz; i+=(i&-i)) t[i].adjust(y,v);}
36     ll sum(int x, int y) {
37         ll s=0;
38         for (int i=x; i; i-=(i&-i)) s += t[i].sum(y);
39         return s;
40     }
41     ll sum(int x1, int y1, int x2, int y2) {
42         ll s = sum(x2,y2)
43             + ((x1>1) ? -sum(x1-1,y2) : 0)
44             + ((y1>1) ? -sum(x2,y1-1) : 0)
45             + ((x1>1&&y1>1) ? sum(x1-1,y1-1) : 0);
46         return s;
47     }
48 };

```

1.2. Tabla Aditiva

```
1 // Tablita aditiva 2D
2 forn (dim, 2) {
3     forn (i, N) {
```

```

4     forn (j, M) {
5         int pi = i-(dim==0), pj = j-(dim==1);
6         if (pi >= 0 && pj >= 0) {
7             dp[i][j] += dp[pi][pj];
8         }
9     }
10 }
11 }
12 // Generalizacion a 32 dimensiones para mascarar de bits
13 forn (i, 32) {
14     forn (mask, 1<<32) {
15         if ((mask>>i)&1) {
16             dp[mask] += dp[mask - (1<<i)];
17         }
18     }
19 }

```

2. Geometria

```

1 struct Point
2 {
3     double x, y;
4     double Point::operator*(const Point &o) const {
5         return x * o.x + y * o.y; }
6     double Point::operator^(const Point &o) const {
7         return x * o.y - y * o.x; }
8     Point Point::operator-(const Point &o) const {
9         return {x - o.x, y - o.y}; }
10    Point Point::operator+(const Point &o) const {
11        return {x + o.x, y + o.y}; }
12    Point Point::operator*(const double &u) const {
13        return {x * u, y * u}; }
14    Point Point::operator/(const double &u) const {
15        return {x / u, y / u}; }
16    double Point::norm_sq() const {
17        return x * x + y * y; }
18    double Point::norm() const {
19        return sqrt(x * x + y * y); }
20 };
21

```

```

22 struct Comp {
23     Vector o, v;
24     Comp(Vector _o, Vector _v) : o(_o), v(_v) {}
25     bool half(Vector p) {
26         assert(!(p.x == 0 && p.y == 0));
27         return (v ^ p) < 0 || ((v ^ p) == 0 && (v * p) < 0);
28     }
29     bool operator()(Vector a, Vector b) {
30         return mp(half(a - o), 011) < mp(half(b - o), ((a - o) ^ (b
31             - o)));
32     };
33
34 struct Segment {
35     Vector a, b;
36     long double eval() const
37     { // funcion auxiliar para ordenar segmentos
38         assert(a.x != b.x || a.y != b.y);
39         Vector a1 = a, b1 = b;
40         if (a1.x > b1.x)
41             swap(a1, b1);
42         assert(x >= a1.x && x <= b1.x);
43         if (x == a1.x)
44             return a1.y;
45         if (x == b1.x)
46             return b1.y;
47         Vector ab = b1 - a1;
48         return a1.y + (x - a1.x) * (ab.y / ab.x);
49     }
50     bool operator<(Segment o) const
51     { // orden de segmentos en un punto (x=cte)
52         return (eval() - o.eval()) < -1e-13;
53     }
54 };
55
56 bool ccw(const Point &a, const Point &m, const Point &b) {
57     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
72 double angle(const Point &a, const Point &m, const Point &b) {
73     Point ma = a - m, mb = b - m;
74     return atan2(ma ^ mb, ma * mb);
75 }
76
77 void sweep_space() {
78     vector<Event> eventos; // puntos, segmentos, ...
79     sort(eventos); // sort por x, y, ...
80     set<Info> estado; // mantener la informacion ordenada
81     forn(i, sz(eventos)) {
82         Event &e = eventos[i];
83         process(e, estado); // procesar un evento cambia el estado
84         ans = actualizar(ans);
85     }
86 }

```

3. Template

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define dprint(v) cout << #v "=" << v << endl //;
4 #define forr(i, a, b) for (int i = (a); i < (b); i++)
5 #define forn(i, n) forr(i, 0, n)
6 #define dforn(i, n) for (int i = n - 1; i >= 0; i--)
7 #define forall(it, v) for (auto it = v.begin(); it != v.end();
    ++it)
8 #define sz(c) ((int)c.size())
9 #define zero(v) memset(v, 0, sizeof(v))

```

```

10 #define pb push_back
11 #define fst first
12 #define snd second
13 #define mid(a,b) ((a+b)>>1)
14 #define mp make_pair
15 #define all(v) begin(v),end(v)
16 #define endl '\n'
17 typedef long long ll;
18 typedef pair<int, int> ii;
19 typedef vector<int> vi;
20
21 int main(int argc, char **argv){
22     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
23     if(argc == 2) freopen(argv[1], "r", stdin);
24
25 }

```

4. Otros

4.1. Fijar el numero de decimales

```

1 cout.precision(7); fixed(cout);
2 cout << x << " " << y;
3 // otra forma
4 cout.precision(7);
5 cout << fixed << x << " " << fixed << y;

```

4.2. Criba Lineal

```

1 const int N = 10000000;
2 vector<int> lp(N+1);
3 vector<int> pr;
4
5 for (int i=2; i <= N; ++i) {
6     if (lp[i] == 0) {
7         lp[i] = i;
8         pr.push_back(i);
9     }
10    for (int j = 0; i * pr[j] <= N; ++j) {
11        lp[i * pr[j]] = pr[j];

```

```
12         if (pr[j] == lp[i]) {  
13             break;  
14         }  
15     }  
16 }
```