

Contents

1	Data Structures	2
1.1	Disjoint Set Union	2
1.2	Segment Tree (with Lazy propagation)	2
1.3	2D Segment Tree	3
1.4	Merge Sort Tree	4
2	Dynamic Programming	4
2.1	Li Chao Tree	4
3	Mathematics	6
3.1	Fast Fourier Transform	6
3.2	Berlekamp Massey Algorithm	6
4	Geometry	8
4.1	Point header	8
4.2	Line / Line segment	8
4.3	Polygons	9
5	Graph	10
5.1	Network Flow	10
5.1.1	Dinic's Maxflow	10
5.1.2	Min cost Max Flow	11
6	String	12
6.1	KMP	12
6.2	Manacher	12
7	Misc	12
7.1	GCC Extensions : OST	12

1 Data Structures

1.1 Disjoint Set Union

struct DSU

```
{
    int par[V], sz[V];
    DSU(){init(V);}
    void init(int n)
    {
        for (int i = 0; i<n; i++)
            par[i] = i, sz[i] = 1;
    }
    int find(int x)
    {
        return x == par[x] ? x : (par[x] = find(par[x]));
    }
    int getSize(int k){return sz[find(k)];}
    void unite(int x, int y)
    {
        int u=find(x), v=find(y);
        if(u==v) return;
        if(sz[u]>sz[v]) swap(u, v);
        sz[v]+=sz[u];
        sz[u] = 0;
        par[u] = par[v];
    }
};
```

1.2 Segment Tree (with Lazy propagation)

Incremental modifications with lazy propagation. All intervals are $[l, r)$.

const int N = 1001010;

struct Segtree

```
{
    int n, h;
    int T[2*N];
    int Lazy[N];
    int32_t Len[2*N];
    void apply(int pos, int val)
    {
        T[pos] += val * Len[pos];
```

```
        if (pos < n) Lazy[pos] += val;
    }
    void build(vector<int> &v)
    {
        n = v.size();
        h = sizeof(int) * 8 - __builtin_clz(n);
        for (int i = 0; i<n; i++)
        {
            T[n+i] = v[i];
            Len[n+i] = 1;
        }
        for (int i = n-1; i>0; i--)
        {
            T[i] = T[i<<1] + T[(i<<1)|1];
            Len[i] = Len[i<<1] + Len[(i<<1)|1];
        }
    }
    void pupd(int p)
    {
        while(p > 1)
        {
            p>>=1;
            T[p] = (T[p<<1] + T[(p<<1)|1] + (Lazy[p] * Len[p]));
        }
    }
    void propagate(int p)
    {
        for (int s = h; s > 0; s--)
        {
            int i = p >> s;
            if (!i) continue;
            if (Lazy[i] != 0)
            {
                apply(i << 1, Lazy[i]);
                apply((i << 1)|1, Lazy[i]);
                Lazy[i] = 0;
            }
        }
    }
    void modify(int pos, int val)
    {
        for (T[pos += n] = val; pos > 1; pos >>= 1)
```

```
        T[pos >> 1] = T[pos] + T[pos^1];
    }
    void modifyRange(int l, int r, int val)
    {
        l += n, r += n;
        int l0 = l, r0 = r;
        for (; l < r; l >>= 1, r >>= 1)
        {
            if (l & 1) apply(l++, val);
            if (r & 1) apply(--r, val);
        }
        pupd(l0), pupd(r0-1);
    }
    // query is on [l, r)!!
    int query(int l, int r)
    {
        l += n, r += n;
        propagate(l); propagate(r-1);
        int res = 0;
        for (; l < r; l >>= 1, r >>= 1)
        {
            if (l & 1)
                res += T[l++];
            if (r & 1)
                res += T[--r];
        }
        return res;
    }
} S;
```

1.3 2D Segment Tree

Point update, rectangle query. All intervals are $[a, b] \times [c, d]$.

```
auto gif = [](int a, int b){return a+b;};
class SEG2D
{
public:
    int n;
    int m;
    vector <vector <int>>> tree;
    SEG2D(int n = 0, int m = 0)
    {
```

```

    tree.resize(2*n);
    for (int i = 0; i<2*n; i++) tree[i].resize(2*m);
    this->n = n;
    this->m = m;
}
SEG2D(int n, int m, vector<vector<int>> &data)
{
    tree.resize(2*n);
    for (int i = 0; i<2*n; i++) tree[i].resize(2*m);
    this->n = n;
    this->m = m;
    init(data);
}
void init(vector <vector <int>> & data)
{
    n = data.size();
    m = data.front().size();
    tree = vector<vector<int>>(2*n, vector<int>(2*m, 0));
    for (int i = 0; i<n; i++)
        for (int j = 0; j<m; j++)
            tree[i+n][j+m] = data[i][j];
    for (int i = n; i<2*n; i++)
        for (int j = m-1; j>0; j--)
            tree[i][j] = gif(tree[i][j*2], tree[i][j*2+1]);
    for (int i = n-1; i>0; i--)
        for (int j = 1; j<2*m; j++)
            tree[i][j] = gif(tree[i*2][j], tree[i*2+1][j]);
}
void update(int x, int y, int val)
{
    tree[x+n][y+m] = val;
    for(int i = y+m; i > 1; i /= 2)
        tree[x+n][i/2] = gif(tree[x+n][i] , tree[x+n][i+1]);
    for (int i = x+n; i>1; i/=2)
        for (int j = y+m; j>=1; j/=2)
            tree[i/2][j] = gif(tree[i][j] , tree[i+1][j]);
}
int query_1D(int x, int yl, int yr)
{
    int res = 0;
    int u = yl+m, v = yr+m+1;
    for(; u<v; u/=2, v/=2)
    {
        if (u & 1)
            int k = query_1D(u++, yl, yr);
            res = gif(res, k);
        if (v & 1)
            int k = query_1D(--v, yl, yr);
            res = gif(res, k);
    }
    return res;
}
int query_2D(int xl, int xr, int yl, int yr)
{
    int res = 0;
    int u = xl+n, v = xr+n+1;
    for(; u<v; u/=2, v/=2)
    {
        if (u & 1)
        {
            int k = query_1D(u++, yl, yr);
            res = gif(res, k);
        }
        if (v & 1)
        {
            int k = query_1D(--v, yl, yr);
            res = gif(res, k);
        }
    }
    return res;
}
};

```

1.4 Merge Sort Tree

greater(s, e, k, 1, 0, n) : How many elements in range [s, e) are strictly greater than k?

```

#define MAXN (1<<18)
#define ST (1<<17)
struct merge_sort_tree
{
    vector <int> tree[MAXN];
    int n;
    void construct (vector <int> data)
    {
        n = 1;

```

```

        while(n < data.size()) n <= 1;
        for (int i = 0; i<data.size(); i++)
            tree[i+n] = {data[i]};
        for (int i = data.size(); i<n; i++)
            tree[i+n] = {};
        for (int i = n-1; i>0; i--)
        {
            tree[i].resize(tree[i*2].size()+tree[i*2+1].size());
            for (int p = 0, q = 0, j = 0; j < tree[i].size(); j++)
            {
                if (p == tree[i*2].size() ||
                    (q<tree[i*2+1].size() && tree[i*2+1][q] > tree[i*2][p]))
                    tree[i][j] = tree[i*2+1][q++];
                else tree[i][j] = tree[i*2][p++];
            }
        }
    }
    //greater(s,e,k,1,0,n)
    int greater(int s, int e, int k, int node, int ns, int ne)
    {
        if (ne <= s || ns >= e)
            return 0;
        if(s <= ns && ne <= e)
            return tree[node].end() - upper_bound(all(tree[node]), k);
        int mid = (ns+ne)>>1;
        return greater(s,e,k,node*2,ns,mid) +
            greater(s,e,k,node*2+1,mid,ne);
    }
};

```

2 Dynamic Programming

2.1 Li Chao Tree

Objective

- (1) Line insert query (ax + b)
- (2) Max / Min on x = t query

Current Implementation : Max query

```

struct LiChao
{
    struct Line // Linear function ax + b
    {

```

```

    int a, b;
    int eval(int x)
    {
        return a*x + b;
    }
};

struct Node // [start, end] has line f
{
    int left, right;
    int start, end;
    Line f;
};

Node new_node(int a, int b)
{
    return {-1,-1,a,b,{0,-INF}};
    // for min, change -INF to INF
}

vector <Node> nodes;

void init(int min_x, int max_x)
{
    nodes.push_back(new_node(min_x, max_x));
}

void insert(int n, Line new_line)
{
    int xl = nodes[n].start, xr = nodes[n].end;
    int xm = (xl + xr)/2;
    Line llo, lhi;
    llo = nodes[n].f, lhi = new_line;
    if (llo.eval(xl) >= lhi.eval(xl))
        swap(llo, lhi);
    if (llo.eval(xr) <= lhi.eval(xr))
    {
        nodes[n].f = lhi;
        // for min, lhi -> llo
        return;
    }
    else if (llo.eval(xm) > lhi.eval(xm))
    {

```

```

        nodes[n].f = llo;
        // for min, llo -> lhi
        if (nodes[n].left == -1)
        {
            nodes[n].left = nodes.size();
            nodes.push_back(new_node(xl,xm));
        }
        insert(nodes[n].left, lhi);
        // for min, lhi -> llo
    }
    else
    {
        nodes[n].f = lhi;
        // for min, lhi -> llo
        if (nodes[n].right == -1)
        {
            nodes[n].right = nodes.size();
            nodes.push_back(new_node(xm+1,xr));
        }
        insert(nodes[n].right,llo);
        // for min, llo -> lhi
    }
}

void insert(Line f)
{
    insert(0, f);
}

int get(int n, int q)
{
    // for min, max -> min, -INF -> INF
    if (n == -1) return -INF;
    int xl = nodes[n].start, xr = nodes[n].end;
    int xm = (xl + xr)/2;
    if (q > xm)
        return max(nodes[n].f.eval(q), get(nodes[n].right, q));
    else return max(nodes[n].f.eval(q), get(nodes[n].left, q));
}

int get(int pt)
{
    return get(0, pt);
}

};

```

3 Mathematics

3.1 Fast Fourier Transform

Multiply two polynomials in $O(n \log n)$. For

```

#define _USE_MATH_DEFINES

#define sz(v) ((int)(v).size())
#define all(v) (v).begin(),(v).end()
typedef vector<int> vi;
typedef complex<double> base;

void fft(vector <base> &a, bool invert)
{
    int n = sz(a);
    for (int i=1,j=0;i<n;i++){
        int bit = n >> 1;
        for (;j>=bit;bit>>=1) j -= bit;
        j += bit;
        if (i < j) swap(a[i],a[j]);
    }
    for (int len=2;len<=n;len<=<=1){
        double ang = 2*M_PI/len*(invert?-1:1);
        base wlen(cos(ang),sin(ang));
        for (int i=0;i<n;i+=len){
            base w(1);
            for (int j=0;j<len/2;j++){
                base u = a[i+j], v = a[i+j+len/2]*w;
                a[i+j] = u+v;
                a[i+j+len/2] = u-v;
                w *= wlen;
            }
        }
    }
    if (invert){
        for (int i=0;i<n;i++) a[i] /= n;
    }
}

void multiply(const vi &a,const vi &b,vi &res)
{
    vector <base> fa(all(a)), fb(all(b));
    int n = 1;

```

```

while (n < max(sz(a),sz(b))) n <= 1;
fa.resize(n); fb.resize(n);
fft(fa,false); fft(fb,false);
for (int i=0;i<n;i++) fa[i] *= fb[i];
fft(fa,true);
res.resize(n);
for (int i=0;i<n;i++)
    res[i] = int(fa[i].real()+(fa[i].real()>0?0.5:-0.5));
}
//multiply(a, b, res)
//for n = b.size
//sum(a[i+j], b[n-1-j]) -> res[n-1+i]

```

3.2 Berlekamp Massey Algorithm

Find minimum linear recurrence from $3n$ first terms.

```

struct Berlekamp_Massey
{

```

```

    const int mod = 1000000007;
    using lint = long long;
    lint ipow(lint x, lint p){
        lint ret = 1, piv = x;
        while(p){
            if(p & 1) ret = ret * piv % mod;
            piv = piv * piv % mod;
            p >>= 1;
        }
        return ret;
    }

```

```

vector<int> berlekamp_massey(vector<int> x){
    vector<int> ls, cur;
    int lf, ld;
    for(int i=0; i<x.size(); i++){
        lint t = 0;
        for(int j=0; j<cur.size(); j++){
            t = (t + 1ll * x[i-j-1] * cur[j]) % mod;
        }
        if((t - x[i]) % mod == 0) continue;
        if(cur.empty()){
            cur.resize(i+1);
            lf = i;
            ld = (t - x[i]) % mod;

```

```

        continue;
    }
    lint k = -(x[i] - t) * ipow(ld, mod - 2) % mod;
    vector<int> c(i-lf-1);
    c.push_back(k);
    for(auto &j : ls) c.push_back(-j * k % mod);
    lint ret = 0;
    if(c.size() < cur.size()) c.resize(cur.size());
    for(int j=0; j<cur.size(); j++){
        c[j] = (c[j] + cur[j]) % mod;
    }
    if(i-lf+(int)ls.size()>=(int)cur.size()){
        tie(ls, lf, ld) = make_tuple(cur, i, (t - x[i]) % mod);
    }
    cur = c;
}
for(auto &i : cur) i = (i % mod + mod) % mod;
return cur;
}

int get_nth(vector<int> rec, vector<int> dp, lint n){
    int m = rec.size();
    vector<int> s(m), t(m);
    s[0] = 1;
    if(m != 1) t[1] = 1;
    else t[0] = rec[0];
    auto mul = [&rec](vector<int> v, vector<int> w){
        int m = v.size();
        vector<int> t(2 * m);
        for(int j=0; j<m; j++){
            for(int k=0; k<m; k++){
                t[j+k] += 1ll * v[j] * w[k] % mod;
                if(t[j+k] >= mod) t[j+k] -= mod;
            }
        }
        for(int j=2*m-1; j>=m; j--){
            for(int k=1; k<=m; k++){
                t[j-k] += 1ll * t[j] * rec[k-1] % mod;
                if(t[j-k] >= mod) t[j-k] -= mod;
            }
        }
        t.resize(m);
        return t;
    };
}

```

```

while(n){
    if(n & 1) s = mul(s, t);
    t = mul(t, t);
    n >>= 1;
}
return ret % mod;
}

int guess_nth_term(vector<int> x, lint n){
    if(n < x.size()) return x[n];
    if(x.size() < 1000) return berlekamp_massey(x);
    if(v.empty()) return 0;
    return get_nth(v, x, n);
}

struct elem{int x, y, v;};
vector<int> get_min_poly(int n, vector<elem> M)
{
    vector<int> rnd1, rnd2;
    mt19937 rng(0x14004);
    auto randint = [&rng](int lb, int ub){
        return uniform_int_distribution<int>(lb, ub);
    };
    for(int i=0; i<n; i++){
        rnd1.push_back(randint(1, mod - 1));
        rnd2.push_back(randint(1, mod - 1));
    }
    vector<int> gobs;
    for(int i=0; i<2*n+2; i++){
        int tmp = 0;
        for(int j=0; j<n; j++){
            tmp += 1ll * rnd2[j] * rnd1[j] % mod;
            if(tmp >= mod) tmp -= mod;
        }
        gobs.push_back(tmp);
        vector<int> nxt(n);
        for(auto &i : M){
            nxt[i.x] += 1ll * i.v * rnd1[i.y] % mod;
            if(nxt[i.x] >= mod) nxt[i.x] -= mod;
        }
        rnd1 = nxt;
    }
}

```

```

    auto sol = berlekamp_massey(gobs);
    reverse(sol.begin(), sol.end());
    return sol;
}
// Usage : guess_nth_term(first_values, n);
};

```

4 Geometry

4.1 Point header

Use as typedef Point<double> P;

```

template <class T> int sgn(T x) { return (x > 0) ? 1 : (x < 0) ? -1 : 0; }
template <class T>
struct Point {
    typedef Point P_;
    T x, y;
    explicit Point(T x=0, T y=0) : x(x), y(y) {}
    bool operator<(P_ p) const { return tie(x,y) < tie(p.x,p.y); }
    bool operator==(P_ p) const { return tie(x,y)==tie(p.x,p.y); }
    P_ operator+(P_ p) const { return P_(x+p.x, y+p.y); }
    P_ operator-(P_ p) const { return P_(x-p.x, y-p.y); }
    P_ operator*(T d) const { return P_(x*d, y*d); }
    P_ operator/(T d) const { return P_(x/d, y/d); }
    T dot(P_ p) const { return x*p.x + y*p.y; }
    T cross(P_ p) const { return x*p.y - y*p.x; }
    T cross(P_ a, P_ b) const { return (a-*this).cross(b-*this); }
    T dist2() const { return x*x + y*y; }
    double dist() const { return sqrt((double)dist2()); }
    // angle to x-axis in interval [-pi, pi]
    double angle() const { return atan2(y, x); }
    P_ unit() const { return *this/dist(); } // makes unit vector
    P_ perp() const { return P_(-y, x); } // rotates +90 degrees
    P_ normal() const { return perp().unit(); }
    // returns point rotated 'a' radians ccw around the origin
    P_ rotate(double a) const {
        return P_(x*cos(a)-y*sin(a),x*sin(a)+y*cos(a));
    }
    friend ostream& operator<<(ostream& os, P_ p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};

```

4.2 Line / Line segment

lineDist : Returns the signed distance between point p and the line containing points a and b. Positive value on left side and negative on right as seen from a towards b. a==b gives nan.

lineInter, segInter : Unique intersection : (1, point), No intersection : (0, (0, 0)), Infinitely many : (-1, (0, 0)). segDist : Return shortest distance between p and segment [s,e].

```

double lineDist(const P& a, const P& b, const P& p) {
    return (double)(b-a).cross(p-a)/(b-a).dist();
}

pair<int, P> lineInter(P s1, P e1, P s2, P e2) {
    auto d = (e1 - s1).cross(e2 - s2);
    if (d == 0) { // if parallel
        if (s1.cross(e1, s2) == 0), P(0, 0);
        else p = s2.cross(e1, e2), q = s2.cross(e2, s1);
        return {1, (s1 * p + e1 * q) / d};
    }
    double segDist(P& s, P& e, P& p) {
        if (s==e) return (p-s).dist();
        auto d = (e-s).dist2(), t = min(d,max(.0,(p-s).dot(e-s)/d),d);
        return ((p-s)*d-(e-s)*t).dist()/d;
    }

    pair<int, P> segInter(P a, P b, P c, P d) {
        auto oa = c.cross(d, a), ob = c.cross(d, b),
            oc = a.cross(b, c), od = a.cross(b, d);
        // Checks if intersection is single non-endpoint point
        if (sgn(oa) * sgn(ob) < 0 && sgn(oc) * sgn(od) < 0)
            return {(a * ob - b * oa) / (ob - oa)};
        set<P> s;
        if (onSegment(c, d, a)) s.insert(a);
        if (onSegment(c, d, b)) s.insert(b);
        if (onSegment(a, b, c)) s.insert(c);
        if (onSegment(a, b, d)) s.insert(d);
    }
}

```

```
return {all(s)};
```

```
}
```

```

bool onSegment(P s, P e, P p) {
    return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
}

```

4.3 Polygons

```

vector<P> convexHull(vector<P> pts) {
    if (sz(pts) <= 1) return pts;
    sort(all(pts));
    vector<P> h(sz(pts)+1);
    int s = 0, t = 0;
    for (int it = 2; it--; s = --t, reverse(all(pts)))
        for (P p : pts) {
            while (t >= s + 2 && h[t-2].cross(h[t-1], p) >= 0)
                h[t++] = p;
        }
    return {h.begin(), h.begin() + t - (t == 2 && h[0] == h[1])};
}

//Returns two points with max distance on a convex hull
array<P, 2> hullDiameter(vector<P> S) {
    int n = sz(S), j = n < 2 ? 0 : 1;
    pair<ll, array<P, 2>> res({0, {S[0], S[0]}});
    rep(i,0,j) {
        for (j = (j + 1) % n) {
            res = max(res, {(S[i] - S[j]).dist2(), {S[i], S[j]}});
            if ((S[(j + 1) % n] - S[j]).cross(S[i + 1] - S[j]) >= 0)
                break;
        }
    }
    return res.second;
}

bool inPolygon(vector<P> &p, P a, bool strict = true) {
    int i = 0, j = 1;
    if (i == j) return false;
    if (i == 0) j = 1;
    if (j == 0) i = 1;
    if (i == 1) j = 2;
    if (j == 1) i = 0;
    if (i == 2) j = 3;
    if (j == 2) i = 1;
    if (i == 3) j = 4;
    if (j == 3) i = 2;
    if (i == 4) j = 5;
    if (j == 4) i = 3;
    if (i == 5) j = 6;
    if (j == 5) i = 4;
    if (i == 6) j = 7;
    if (j == 6) i = 5;
    if (i == 7) j = 8;
    if (j == 7) i = 6;
    if (i == 8) j = 9;
    if (j == 8) i = 7;
    if (i == 9) j = 10;
    if (j == 9) i = 8;
    if (i == 10) j = 11;
    if (j == 10) i = 9;
    if (i == 11) j = 12;
    if (j == 11) i = 10;
    if (i == 12) j = 13;
    if (j == 12) i = 11;
    if (i == 13) j = 14;
    if (j == 13) i = 12;
    if (i == 14) j = 15;
    if (j == 14) i = 13;
    if (i == 15) j = 16;
    if (j == 15) i = 14;
    if (i == 16) j = 17;
    if (j == 16) i = 15;
    if (i == 17) j = 18;
    if (j == 17) i = 16;
    if (i == 18) j = 19;
    if (j == 18) i = 17;
    if (i == 19) j = 20;
    if (j == 19) i = 18;
    if (i == 20) j = 21;
    if (j == 20) i = 19;
    if (i == 21) j = 22;
    if (j == 21) i = 20;
    if (i == 22) j = 23;
    if (j == 22) i = 21;
    if (i == 23) j = 24;
    if (j == 23) i = 22;
    if (i == 24) j = 25;
    if (j == 24) i = 23;
    if (i == 25) j = 26;
    if (j == 25) i = 24;
    if (i == 26) j = 27;
    if (j == 26) i = 25;
    if (i == 27) j = 28;
    if (j == 27) i = 26;
    if (i == 28) j = 29;
    if (j == 28) i = 27;
    if (i == 29) j = 30;
    if (j == 29) i = 28;
    if (i == 30) j = 31;
    if (j == 30) i = 29;
    if (i == 31) j = 32;
    if (j == 31) i = 30;
    if (i == 32) j = 33;
    if (j == 32) i = 31;
    if (i == 33) j = 34;
    if (j == 33) i = 32;
    if (i == 34) j = 35;
    if (j == 34) i = 33;
    if (i == 35) j = 36;
    if (j == 35) i = 34;
    if (i == 36) j = 37;
    if (j == 36) i = 35;
    if (i == 37) j = 38;
    if (j == 37) i = 36;
    if (i == 38) j = 39;
    if (j == 38) i = 37;
    if (i == 39) j = 40;
    if (j == 39) i = 38;
    if (i == 40) j = 41;
    if (j == 40) i = 39;
    if (i == 41) j = 42;
    if (j == 41) i = 40;
    if (i == 42) j = 43;
    if (j == 42) i = 41;
    if (i == 43) j = 44;
    if (j == 43) i = 42;
    if (i == 44) j = 45;
    if (j == 44) i = 43;
    if (i == 45) j = 46;
    if (j == 45) i = 44;
    if (i == 46) j = 47;
    if (j == 46) i = 45;
    if (i == 47) j = 48;
    if (j == 47) i = 46;
    if (i == 48) j = 49;
    if (j == 48) i = 47;
    if (i == 49) j = 50;
    if (j == 49) i = 48;
    if (i == 50) j = 51;
    if (j == 50) i = 49;
    if (i == 51) j = 52;
    if (j == 51) i = 50;
    if (i == 52) j = 53;
    if (j == 52) i = 51;
    if (i == 53) j = 54;
    if (j == 53) i = 52;
    if (i == 54) j = 55;
    if (j == 54) i = 53;
    if (i == 55) j = 56;
    if (j == 55) i = 54;
    if (i == 56) j = 57;
    if (j == 56) i = 55;
    if (i == 57) j = 58;
    if (j == 57) i = 56;
    if (i == 58) j = 59;
    if (j == 58) i = 57;
    if (i == 59) j = 60;
    if (j == 59) i = 58;
    if (i == 60) j = 61;
    if (j == 60) i = 59;
    if (i == 61) j = 62;
    if (j == 61) i = 60;
    if (i == 62) j = 63;
    if (j == 62) i = 61;
    if (i == 63) j = 64;
    if (j == 63) i = 62;
    if (i == 64) j = 65;
    if (j == 64) i = 63;
    if (i == 65) j = 66;
    if (j == 65) i = 64;
    if (i == 66) j = 67;
    if (j == 66) i = 65;
    if (i == 67) j = 68;
    if (j == 67) i = 66;
    if (i == 68) j = 69;
    if (j == 68) i = 67;
    if (i == 69) j = 70;
    if (j == 69) i = 68;
    if (i == 70) j = 71;
    if (j == 70) i = 69;
    if (i == 71) j = 72;
    if (j == 71) i = 70;
    if (i == 72) j = 73;
    if (j == 72) i = 71;
    if (i == 73) j = 74;
    if (j == 73) i = 72;
    if (i == 74) j = 75;
    if (j == 74) i = 73;
    if (i == 75) j = 76;
    if (j == 75) i = 74;
    if (i == 76) j = 77;
    if (j == 76) i = 75;
    if (i == 77) j = 78;
    if (j == 77) i = 76;
    if (i == 78) j = 79;
    if (j == 78) i = 77;
    if (i == 79) j = 80;
    if (j == 79) i = 78;
    if (i == 80) j = 81;
    if (j == 80) i = 79;
    if (i == 81) j = 82;
    if (j == 81) i = 80;
    if (i == 82) j = 83;
    if (j == 82) i = 81;
    if (i == 83) j = 84;
    if (j == 83) i = 82;
    if (i == 84) j = 85;
    if (j == 84) i = 83;
    if (i == 85) j = 86;
    if (j == 85) i = 84;
    if (i == 86) j = 87;
    if (j == 86) i = 85;
    if (i == 87) j = 88;
    if (j == 87) i = 86;
    if (i == 88) j = 89;
    if (j == 88) i = 87;
    if (i == 89) j = 90;
    if (j == 89) i = 88;
    if (i == 90) j = 91;
    if (j == 90) i = 89;
    if (i == 91) j = 92;
    if (j == 91) i = 90;
    if (i == 92) j = 93;
    if (j == 92) i = 91;
    if (i == 93) j = 94;
    if (j == 93) i = 92;
    if (i == 94) j = 95;
    if (j == 94) i = 93;
    if (i == 95) j = 96;
    if (j == 95) i = 94;
    if (i == 96) j = 97;
    if (j == 96) i = 95;
    if (i == 97) j = 98;
    if (j == 97) i = 96;
    if (i == 98) j = 99;
    if (j == 98) i = 97;
    if (i == 99) j = 100;
    if (j == 99) i = 98;
    if (i == 100) j = 101;
    if (j == 100) i = 99;
    if (i == 101) j = 102;
    if (j == 101) i = 100;
    if (i == 102) j = 103;
    if (j == 102) i = 101;
    if (i == 103) j = 104;
    if (j == 103) i = 102;
    if (i == 104) j = 105;
    if (j == 104) i = 103;
    if (i == 105) j = 106;
    if (j == 105) i = 104;
    if (i == 106) j = 107;
    if (j == 106) i = 105;
    if (i == 107) j = 108;
    if (j == 107) i = 106;
    if (i == 108) j = 109;
    if (j == 108) i = 107;
    if (i == 109) j = 110;
    if (j == 109) i = 108;
    if (i == 110) j = 111;
    if (j == 110) i = 109;
    if (i == 111) j = 112;
    if (j == 111) i = 110;
    if (i == 112) j = 113;
    if (j == 112) i = 111;
    if (i == 113) j = 114;
    if (j == 113) i = 112;
    if (i == 114) j = 115;
    if (j == 114) i = 113;
    if (i == 115) j = 116;
    if (j == 115) i = 114;
    if (i == 116) j = 117;
    if (j == 116) i = 115;
    if (i == 117) j = 118;
    if (j == 117) i = 116;
    if (i == 118) j = 119;
    if (j == 118) i = 117;
    if (i == 119) j = 120;
    if (j == 119) i = 118;
    if (i == 120) j = 121;
    if (j == 120) i = 119;
    if (i == 121) j = 122;
    if (j == 121) i = 120;
    if (i == 122) j = 123;
    if (j == 122) i = 121;
    if (i == 123) j = 124;
    if (j == 123) i = 122;
    if (i == 124) j = 125;
    if (j == 124) i = 123;
    if (i == 125) j = 126;
    if (j == 125) i = 124;
    if (i == 126) j = 127;
    if (j == 126) i = 125;
    if (i == 127) j = 128;
    if (j == 127) i = 126;
    if (i == 128) j = 129;
    if (j == 128) i = 127;
    if (i == 129) j = 130;
    if (j == 129) i = 128;
    if (i == 130) j = 131;
    if (j == 130) i = 129;
    if (i == 131) j = 132;
    if (j == 131) i = 130;
    if (i == 132) j = 133;
    if (j == 132) i = 131;
    if (i == 133) j = 134;
    if (j == 133) i = 132;
    if (i == 134) j = 135;
    if (j == 134) i = 133;
    if (i == 135) j = 136;
    if (j == 135) i = 134;
    if (i == 136) j = 137;
    if (j == 136) i = 135;
    if (i == 137) j = 138;
    if (j == 137) i = 136;
    if (i == 138) j = 139;
    if (j == 138) i = 137;
    if (i == 139) j = 140;
    if (j == 139) i = 138;
    if (i == 140) j = 141;
    if (j == 140) i = 139;
    if (i == 141) j = 142;
    if (j == 141) i = 140;
    if (i == 142) j = 143;
    if (j == 142) i = 141;
    if (i == 143) j = 144;
    if (j == 143) i = 142;
    if (i == 144) j = 145;
    if (j == 144) i = 143;
    if (i == 145) j = 146;
    if (j == 145) i = 144;
    if (i == 146) j = 147;
    if (j == 146) i = 145;
    if (i == 147) j = 148;
    if (j == 147) i = 146;
    if (i == 148) j = 149;
    if (j == 148) i = 147;
    if (i == 149) j = 150;
    if (j == 149) i = 148;
    if (i == 150) j = 151;
    if (j == 150) i = 149;
    if (i == 151) j = 152;
    if (j == 151) i = 150;
    if (i == 152) j = 153;
    if (j == 152) i = 151;
    if (i == 153) j = 154;
    if (j == 153) i = 152;
    if (i == 154) j = 155;
    if (j == 154) i = 153;
    if (i == 155) j = 156;
    if (j == 155) i = 154;
    if (i == 156) j = 157;
    if (j == 156) i = 155;
    if (i == 157) j = 158;
    if (j == 157) i = 156;
    if (i == 158) j = 159;
    if (j == 158) i = 157;
    if (i == 159) j = 160;
    if (j == 159) i = 158;
    if (i == 160) j = 161;
    if (j == 160) i = 159;
    if (i == 161) j = 162;
    if (j == 161) i = 160;
    if (i == 162) j = 163;
    if (j == 162) i = 161;
    if (i == 163) j = 164;
    if (j == 163) i = 162;
    if (i == 164) j = 165;
    if (j == 164) i = 163;
    if (i == 165) j = 166;
    if (j == 165) i = 164;
    if (i == 166) j = 167;
    if (j == 166) i = 165;
    if (i == 167) j = 168;
    if (j == 167) i = 166;
    if (i == 168) j = 169;
    if (j == 168) i = 167;
    if (i == 169) j = 170;
    if (j == 169) i = 168;
    if (i == 170) j = 171;
    if (j == 170) i = 169;
    if (i == 171) j = 172;
    if (j == 171) i = 170;
    if (i == 172) j = 173;
    if (j == 172) i = 171;
    if (i == 173) j = 174;
    if (j == 173) i = 172;
    if (i == 174) j = 175;
    if (j == 174) i = 173;
    if (i == 175) j = 176;
    if (j == 175) i = 174;
    if (i == 176) j = 177;
    if (j == 176) i = 175;
    if (i == 177) j = 178;
    if (j == 177) i = 176;
    if (i == 178) j = 179;
    if (j == 178) i = 177;
    if (i == 179) j = 180;
    if (j == 179) i = 178;
    if (i == 180) j = 181;
    if (j == 180) i = 179;
    if (i == 181) j = 182;
    if (j == 181) i = 180;
    if (i == 182) j = 183;
    if (j == 182) i = 181;
    if (i == 183) j = 184;
    if (j == 183) i = 182;
    if (i == 184) j = 185;
    if (j == 184) i = 183;
    if (i == 185) j = 186;
    if (j == 185) i = 184;
    if (i == 186) j = 187;
    if (j == 186) i = 185;
    if (i == 187) j = 188;
    if (j == 187) i = 186;
    if (i == 188) j = 189;
    if (j == 188) i = 187;
    if (i == 189) j = 190;
    if (j == 189) i = 188;
    if (i == 190) j = 191;
    if (j == 190) i = 189;
    if (i == 191) j = 192;
    if (j == 191) i = 190;
    if (i == 192) j = 193;
    if (j == 192) i = 191;
    if (i == 193) j = 194;
    if (j == 193) i = 192;
    if (i == 194) j = 195;
    if (j == 194) i = 193;
    if (i == 195) j = 196;
    if (j == 195) i = 194;
    if (i == 196) j = 197;
    if (j == 196) i = 195;
    if (i == 197) j = 198;
    if (j == 197) i = 196;
    if (i == 198) j = 199;
    if (j == 198) i = 197;
    if (i == 199) j = 200;
    if (j == 199) i = 198;
    if (i == 200) j = 201;
    if (j == 200) i = 199;
    if (i == 201) j = 202;
    if (j == 201) i = 200;
    if (i == 202) j = 203;
    if (j == 202) i = 201;
    if (i == 203) j = 204;
    if (j == 203) i = 202;
    if (i == 204) j = 205;
    if (j == 204) i = 203;
    if (i == 205) j = 206;
    if (j == 205) i = 204;
    if (i == 206) j = 207;
    if (j == 206) i = 205;
    if (i == 207) j = 208;
    if (j == 207) i = 206;
    if (i == 208) j = 209;
    if (j == 208) i = 207;
    if (i == 209) j = 210;
    if (j == 209) i = 208;
    if (i == 210) j = 211;
    if (j == 210) i = 209;
    if (i == 211) j = 212;
    if (j == 211) i = 210;
    if (i == 212) j = 213;
    if (j == 212) i = 211;
    if (i == 213) j = 214;
    if (j == 213) i = 212;
    if (i == 214) j = 215;
    if (j == 214) i = 213;
    if (i == 215) j = 216;
    if (j == 215) i = 214;
    if (i == 216) j = 217;
    if (j == 216) i = 215;
    if (i == 217) j = 218;
    if (j == 217) i = 216;
    if (i == 218) j = 219;
    if (j == 218) i = 217;
    if (i == 219) j = 220;
    if (j == 219) i = 218;
    if (i == 220) j = 221;
    if (j == 220) i = 219;
    if (i == 221) j = 222;
    if (j == 221) i = 220;
    if (i == 222) j = 223;
    if (j == 222) i = 221;
    if (i == 223) j = 224;
    if (j == 223) i = 222;
    if (i == 224) j = 225;
    if (j == 224) i = 223;
    if (i == 225) j = 226;
    if (j == 225) i = 224;
    if (i == 226) j = 227;
    if (j == 226) i = 225;
    if (i == 227) j = 228;
    if (j == 227) i = 226;
    if (i == 228) j = 229;
    if (j == 228) i = 227;
    if (i == 229) j = 230;
    if (j == 229) i = 228;
    if (i == 230) j = 231;
    if (j == 230) i = 229;
    if (i == 231) j = 232;
    if (j == 231) i = 230;
    if (i == 232) j = 233;
    if (j == 232) i = 231;
    if (i == 233) j = 234;
    if (j == 233) i = 232;
    if (i == 234) j = 235;
    if (j == 234) i = 233;
    if (i == 235) j = 236;
    if (j == 235) i = 234;
    if (i == 236) j = 237;
    if (j == 236) i = 235;
    if (i == 237) j = 238;
    if (j == 237) i = 236;
    if (i == 238) j = 239;
    if (j == 238) i = 237;
    if (i == 239) j = 240;
    if (j == 239) i = 238;
    if (i == 240) j = 241;
    if (j == 240) i = 239;
    if (i == 241) j = 242;
    if (j == 241) i = 240;
    if (i == 242) j = 243;
    if (j == 242) i = 241;
    if (i == 243) j = 244;
    if (j == 243) i = 242;
    if (i == 244) j = 245;
    if (j == 244) i = 243;
    if (i == 245) j = 246;
    if (j == 245) i = 244;
    if (i == 246) j = 247;
    if (j == 246) i = 245;
    if (i == 247) j = 248;
    if (j == 247) i = 246;
    if (i == 248) j = 249;
    if (j == 248) i = 247;
    if (i == 249) j = 250;
    if (j == 249) i = 248;
    if (i == 250) j = 251;
    if (j == 250) i = 249;
    if (i == 251) j = 252;
    if (j == 251) i = 250;
    if (i == 252) j = 253;
    if (j == 252) i = 251;
    if (i == 253) j = 254;
    if (j == 253) i = 252;
    if (i == 254) j = 255;
    if (j == 254) i = 253;
    if (i == 255) j = 256;
    if (j == 255) i = 254;
    if (i == 256) j = 257;
    if (j == 256) i = 255;
    if (i == 257) j = 258;
    if (j == 257) i = 256;
    if (i == 258) j = 259;
    if (j == 258) i = 257;
    if (i == 259) j = 260;
    if (j == 259) i = 258;
    if (i == 260) j = 261;
    if (j == 260) i = 259;
    if (i == 261) j = 262;
    if (j == 261) i = 260;
    if (i == 262) j = 263;
    if (j == 262) i = 261;
    if (i == 263) j = 264;
    if (j == 263) i = 262;
    if (i == 264) j = 265;
    if (j == 264) i = 263;
    if (i == 265) j = 266;
    if (j == 265) i = 264;
    if (i == 266) j = 267;
    if (j == 266) i = 265;
    if (i == 267) j = 268;
    if (j == 267) i = 266;
    if (i == 268) j = 269;
    if (j == 268) i = 267;
    if (i == 269) j = 270;
    if (j == 269) i = 268;
    if (i == 270) j = 271;
    if (j == 270) i = 269;
    if (i == 271) j = 272;
    if (j == 271) i = 270;
    if (i == 272) j = 273;
    if (j == 272) i = 271;
    if (i == 273) j = 274;
    if (j == 273) i = 272;
    if (i == 274) j = 275;
    if (j == 274) i = 273;
    if (i == 275) j = 276;
    if (j == 275) i = 274;
    if (i == 276) j = 277;
    if (j == 276) i = 275;
    if (i == 277) j = 278;
    if (j == 277) i = 276;
    if (i == 278) j = 279;
    if (j == 278) i = 277;
    if (i == 279) j = 280;
    if (j == 279) i = 278;
    if (i == 280) j = 281;
    if (j == 280) i = 279;
    if (i == 281) j = 282;
    if (j == 281) i = 280;
    if (i == 282) j = 283;
    if (j == 282) i = 281;
    if (i == 283) j = 284;
    if (j == 283) i = 282;
    if (i == 284) j = 285;
    if (j == 284) i = 283;
    if (i == 285) j = 286;
    if (j == 285) i = 284;
    if (i == 286) j = 287;
    if (j == 286) i = 285;
    if (i == 287) j = 288;
    if (j == 287) i = 286;
    if (i == 288) j = 289;
    if (j == 288) i = 287;
    if (i == 289) j = 290;
    if (j == 289) i = 288;
    if (i == 290) j = 291;
    if (j == 290) i = 289;
    if (i == 291) j = 292;
    if (j == 291) i = 290;
    if (i == 292) j = 293;
    if (j == 292) i = 291;
    if (i == 293) j = 294;
    if (j == 293) i = 292
```

```

int cnt = 0, n = sz(p);
rep(i,0,n) {
    P q = p[(i + 1) % n];
    if (onSegment(p[i], q, a)) return !strict;
    //or: if (segDist(p[i], q, a) <= eps) return !strict;
    cnt ^= ((a.y<p[i].y) - (a.y<q.y)) * a.cross(p[i], q);
}
return cnt;
}

```

// Twice the signed area of polygon

template<class T>

```

T polygonArea2(vector<Point<T>>& v) {
    T a = v.back().cross(v[0]);
    rep(i,0,sz(v)-1) a += v[i].cross(v[i+1]);
    return a;
}

```

```

#define cmp(i,j) sgn(dir.perp().cross(poly[(i)%n]-poly[(j)%n]))

```

```

#define extr(i) cmp(i + 1, i) >= 0 && cmp(i, i - 1) >= 0

```

```

template <class P> int extrVertex(vector<P>& poly, P dir) {

```

```

    int n = sz(poly), lo = 0, hi = n;
    if (extr(0)) return 0;
    while (lo + 1 < hi) {
        int m = (lo + hi) / 2;
        if (extr(m)) return m;
        int ls = cmp(lo + 1, lo), ms = cmp(m + 1, m);
        (ls < ms || (ls == ms && ls == cmp(lo, m))) ? hi = m : lo = m;
    }
    return lo;
}

```

```

#define cmpL(i) sgn(a.cross(poly[i], b))

```

template <class P>

```

array<int, 2> lineHull(P a, P b, vector<P> poly) {
    int endA = extrVertex(poly, (a - b).perp());
    int endB = extrVertex(poly, (b - a).perp());
    if (cmpL(endA) < 0 || cmpL(endB) > 0)
        return {-1, -1};
    array<int, 2> res;
    rep(i,0,2) {

```

```

        int lo = endB, hi = endA, n = sz(poly);
        while ((lo + 1) % n != hi) {
            int m = ((lo + hi + (lo < hi ? 0 : n)) / 2) % n;
            (cmpL(m) == cmpL(endB) ? lo : hi) = m;
        }
        res[i] = (lo + !cmpL(hi)) % n;
    }
    if (res[0] == res[1]) return {res[0], -1};
    if (!cmpL(res[0]) && !cmpL(res[1]))
        switch ((res[0] - res[1] + sz(poly) + 1) % sz(poly)) {
            case 0: return {res[0], res[0]};
            case 2: return {res[1], res[1]};
        }
    return res;
}

```

5 Graph

5.1 Network Flow

5.1.1 Dinic's Maxflow

Fast max flow algorithm. Use maxflow and AddEdge.

```

struct Edge
{
    int u, v;
    ll cap, flow;
    Edge() {}
    Edge(int u, int v, ll cap): u(u), v(v), cap(cap), flow(0) {}
};

struct Dinic
{
    int N;
    vector<Edge> E;
    vector<vector<int>> g;
    vector<int> d, pt;

    Dinic(int N): N(N), E(0), g(N), d(N), pt(N) {}

    void AddEdge(int u, int v, ll cap)
    {
        if (u != v)

```

```

        {
            E.push_back(Edge(u, v, cap));
            g[u].push_back(E.size() - 1);
            E.push_back(Edge(v, u, 0));
            g[v].push_back(E.size() - 1);
        }
    }

    bool BFS(int S, int T)
    {
        queue<int> q({S});
        fill(d.begin(), d.end(), N + 1);
        d[S] = 0;
        while(!q.empty())
        {
            int u = q.front();
            q.pop();
            if (u == T) break;
            for (int k: g[u])
            {
                Edge &e = E[k];
                if (e.flow < e.cap && d[e.v] > d[e.u] + 1)
                {
                    d[e.v] = d[e.u] + 1;
                    q.push(e.v);
                }
            }
        }
        return d[T] != N + 1;
    }

    ll DFS(int u, int T, ll flow = -1)
    {
        if (u == T || flow == 0) return flow;
        for (int &i = pt[u]; i < g[u].size(); i++)
        {
            Edge &e = E[g[u][i]];
            Edge &oe = E[g[u][i]^1];
            if (d[e.v] == d[e.u] + 1)
            {
                ll amt = e.cap - e.flow;
                if (flow != -1 && amt > flow) amt = flow;

```

```

        if (ll pushed = DFS(e.v, T, amt))
        {
            e.flow += pushed;
            oe.flow -= pushed;
            return pushed;
        }
    }
    return 0;
}

ll MaxFlow(int S, int T)
{
    ll total = 0;
    while (BFS(S, T))
    {
        fill(pt.begin(), pt.end(), 0);
        while (ll flow = DFS(S, T))
            total += flow;
    }
    return total;
}
};

```

5.1.2 Min cost Max Flow

Fast min cost max flow algorithm. Use `solveMCMF` and `AddEdge`.

`solveMCMF` returns pair of (max flow, cost of such flow).

```

const int MAXN = 1010;
struct MCMF
{
    struct edg { int pos, cap, rev, cost; };
    vector<edg> gph[MAXN];
    void clear(){
        for(int i=0; i<MAXN; i++) gph[i].clear();
    }
    void AddEdge(int s, int e, int cap, int cst)
    // add edge (s to e, capacity = x, cost = c)
    {
        gph[s].push_back({e, cap, (int)gph[e].size(), cst});
        gph[e].push_back({s, 0, (int)gph[s].size()-1, -cst});
    }
}

```

```

int dist[MAXN], pa[MAXN], pe[MAXN];
bool inque[MAXN];
bool spfa(int src, int sink)
{
    memset(dist, 0x3f, sizeof(dist));
    memset(inque, 0, sizeof(inque));
    queue<int> que;
    dist[src] = 0;
    inque[src] = 1;
    que.push(src);
    bool ok = 0;
    while(!que.empty()){
        int x = que.front();
        que.pop();
        if(x == sink) ok = 1;
        inque[x] = 0;
        for(int i=0; i<gph[x].size(); i++)
        {
            edg e = gph[x][i];
            if(e.cap > 0 && dist[e.pos] > dist[x] + e.cost)
            {
                dist[e.pos] = dist[x] + e.cost;
                pa[e.pos] = x;
                pe[e.pos] = i;
                if(!inque[e.pos]){
                    inque[e.pos] = 1;
                    que.push(e.pos);
                }
            }
        }
    }
    return ok;
}

pii solveMCMF(int src, int sink){
    int MCMF_COST = 0;
    int MCMF_FLOW = 0;
    while(spfa(src, sink)){
        int cap = 1e9;
        for(int pos = sink; pos != src; pos = pa[pos])
            cap = min(cap, gph[pa[pos]][pe[pos]].cap);
        MCMF_COST += dist[sink] * cap;
    }
}

```

```

        for(int pos = sink; pos != src; pos = pa[pos])
            int rev = gph[pa[pos]][pe[pos]].rev;
            gph[pa[pos]][pe[pos]].cap -= cap;
            gph[pos][rev].cap += cap;
    }
    MCMF_FLOW += cap;
}
return {MCMF_FLOW, MCMF_COST};
}
};

```

6 String

6.1 KMP

Pi 배열의 정의 : $str[0]$ 부터 $str[i]$ 까지 중 접두사가 접미사와 같은 부분만큼의 길이.

```

vector<int> getPi(string p)
{
    int j = 0;
    int plen = p.length();
    vector<int> pi;
    pi.resize(plen);
    for(int i = 1; i < plen; i++)
    {
        while((j > 0) && (p[i] != p[j]))
            j = pi[j-1];
        if(p[i] == p[j])
        {
            j++;
            pi[i] = j;
        }
    }
    return pi;
}

vector<int> kmp(string s, string p)
{
    vector<int> ans;
    auto pi = getPi(p);
    int slen = s.length(), plen = p.length(), j = 0;
    for(int i = 0; i < slen; i++)
    {
        while(j > 0 && s[i] != p[j])
            j = pi[j-1];
        if(s[i] == p[j])
            j++;
    }
}

```



```

        j = pi[j-1];
        if(s[i] == p[j])
        {
            if(j==plen-1)
            {
                ans.push_back(i-plen+1);
                j = pi[j];
            }
            else
                j++;
        }
    }
    return ans;
}

```

6.2 Manacher

$A[i] = i$ 번을 중심으로 하는 가장 긴 팰린드롬이 되는 반지름.

```

int N,A[MAXN];
char S[MAXN];

void Manachers()
{
    int r = 0, p = 0;
    for (int i=1;i<=N;i++)
    {
        if (i <= r)
            A[i] = min(A[2*p-i],r-i);
        else
            A[i] = 0;
        while (i-A[i]-1 > 0 && i+A[i]+1 <= N
            && S[i-A[i]-1] == S[i+A[i]+1])
            A[i]++;
        if (r < i+A[i])
            r = i+A[i], p = i;
    }
}

```

7 Misc

7.1 GCC Extensions : OST

```
#include <ext/pb_ds/assoc_container.hpp>
```

```

#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/detail/standard_policies.hpp>
typedef tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> ordered_set;

void test()
{
    ordered_set X;
    X.insert(1);
    X.insert(2);
    X.insert(4);
    X.insert(8);
    X.insert(16);

    cout<<*X.find_by_order(1)<<endl; // 2
    cout<<*X.find_by_order(2)<<endl; // 4
    cout<<*X.find_by_order(4)<<endl; // 16
    cout<<(end(X)==X.find_by_order(6))<<endl; // true

    cout<<X.order_of_key(-5)<<endl; // 0
    cout<<X.order_of_key(1)<<endl; // 0
    cout<<X.order_of_key(3)<<endl; // 2
    cout<<X.order_of_key(4)<<endl; // 2
    cout<<X.order_of_key(400)<<endl; // 5
}

```