USACO Notebook

Ben Qi

March 6, 2019

Contents

1 Contest

1.1 C++ Input and Output

```
/**
 * Description: convenient functions for input / output
 * Source: https://codeforces.com/blog/entry/65311,
     misc others
 * Verification:
     http://codeforces.com/contest/1045/problem/D
 */

namespace input {
    template<class T> void re(complex<T>& x);
    template<class T1, class T2> void re(pair<T1,T2>&
        p);
    template<class T> void re(vector<T>& a);
    template<class T, size_t SZ> void re(array<T,SZ>&
        a);

    template<class T> void re(T& x) { cin >> x; }
    void re(double& x) { string t; re(t); x = stod(t);
        }
    void re(ld& x) { string t; re(t); x = stold(t); }
    template<class Arg, class... Args> void re(Arg&
        first, Args&... rest) {
        re(first); re(rest...);
    }

    template<class T> void re(complex<T>& x) { T a,b;
        re(a,b); x = cd(a,b); }
    template<class T1, class T2> void re(pair<T1,T2>&
        p) { re(p.f,p.s); }
    template<class T> void re(vector<T>& a) {
        FOR(i,sz(a)) re(a[i]); }
    template<class T, size_t SZ> void re(array<T,SZ>&
        a) { FOR(i,SZ) re(a[i]); }
}

using namespace input;

namespace output {
    template<class T1, class T2> void pr(const
        pair<T1,T2>& x);
    template<class T, size_t SZ> void pr(const
        array<T,SZ>& x);
    template<class T> void pr(const vector<T>& x);
    template<class T> void pr(const set<T>& x);
    template<class T1, class T2> void pr(const
        map<T1,T2>& x);

    template<class T> void pr(const T& x) { cout << x;
        }
    template<class Arg, class... Args> void pr(const
        Arg& first, const Args&... rest) {
        pr(first); pr(rest...);
    }

    template<class T1, class T2> void pr(const
        pair<T1,T2>& x) {
        pr("{",x.f,", ",x.s,"}");
```

```
    }
    template<class T> void prContain(const T& x) {
        pr("{");
        bool fst = 1; trav(a,x) pr(!fst?", ":"",a),
            fst = 0;
        pr("}");
    }
    template<class T, size_t SZ> void pr(const
        array<T,SZ>& x) { prContain(x); }
    template<class T> void pr(const vector<T>& x) {
        prContain(x); }
    template<class T> void pr(const set<T>& x) {
        prContain(x); }
    template<class T1, class T2> void pr(const
        map<T1,T2>& x) { prContain(x); }

    void ps() { pr("\n"); }
    template<class Arg, class... Args> void ps(const
        Arg& first, const Args&... rest) {
        pr(first," "); ps(rest...); // print w/ spaces
    }
}

using namespace output;

namespace io {
    void setIn(string s) {
        freopen(s.c_str(),"r",stdin); }
    void setOut(string s) {
        freopen(s.c_str(),"w",stdout); }
    void setIO(string s = "") {
        ios_base::sync_with_stdio(0); cin.tie(0); //
            fast I/O
        if (sz(s)) { setIn(s+".in"), setOut(s+".out");
            } // for USACO
    }
}

using namespace io;
```

1.2 C++ Template

```
#pragma GCC optimize ("O3")
#pragma GCC target ("sse4")

#include <bits/stdc++.h>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/rope>

using namespace std;
using namespace __gnu_pbds;
using namespace __gnu_cxx;

typedef long long ll;
typedef long double ld;
typedef complex<ld> cd;

typedef pair<int, int> pi;
typedef pair<ll,ll> pl;
typedef pair<ld,ld> pd;

typedef vector<int> vi;
typedef vector<ld> vd;
typedef vector<ll> vl;
typedef vector<pi> vpi;
typedef vector<pl> vpl;
typedef vector<cd> vcd;

template <class T> using Tree = tree<T, null_type,
    less<T>,
    rb_tree_tag,tree_order_statistics_node_update>;

#define FOR(i, a, b) for (int i = (a); i < (b); i++)
#define F0R(i, a) for (int i = 0; i < (a); i++)
#define FORd(i,a,b) for (int i = (b)-1; i >= (a); i--)
#define F0Rd(i,a) for (int i = (a)-1; i >= 0; i--)
#define trav(a, x) for (auto& a : x)

#define mp make_pair
#define pb push_back
#define f first
#define s second
#define lb lower_bound
#define ub upper_bound

#define sz(x) (int)x.size()
#define beg(x) x.begin()
#define en(x) x.end()
#define all(x) beg(x), en(x)
#define resz resize

const int MOD = 1000000007;
const ll INF = 1e18;
const int MX = 100001;
const ld PI = 4*atan((ld)1);

template<class T> void ckmin(T &a, T b) { a = min(a,
    b); }
template<class T> void ckmax(T &a, T b) { a = max(a,
    b); }

int main() {
    // you should actually read the stuff at the bottom
    setIO();

    // you should actually read the stuff at the bottom
}

/* stuff you should look for
    * int overflow, array bounds
    * special cases (n=1?), set tle
    * do smth instead of nothing and stay organized
*/
```

1.3 Java FastScanner

```
/**
* Description: Faster input / output for java
* Source: Matt Fontaine
```

```
* Verification: ?
*/

class FastScanner {
    private InputStream stream;
    private byte[] buf = new byte[1024];
    private int curChar;
    private int numChars;

    public FastScanner(InputStream stream) {
        this.stream = stream;
    }
    int read() {
        if (numChars == -1)
            throw new InputMismatchException();
        if (curChar >= numChars) {
            curChar = 0;
            try {
                numChars = stream.read(buf);
            } catch (IOException e) {
                throw new InputMismatchException();
            }
            if (numChars <= 0) return -1;
        }
        return buf[curChar++];
    }

    boolean isSpaceChar(int c) {
        return c == ' ' || c == '\n' || c == '\r' || c
            == '\t' || c == -1;
    }

    boolean isEndline(int c) {
        return c == '\n' || c == '\r' || c == -1;
    }

    public int nextInt() {
        return Integer.parseInt(next());
    }

    public long nextLong() {
        return Long.parseLong(next());
    }

    public double nextDouble() {
        return Double.parseDouble(next());
    }

    public String next() {
        int c = read();
        while (isSpaceChar(c)) c = read();
        StringBuilder res = new StringBuilder();
        do {
            res.appendCodePoint(c);
            c = read();
        } while (!isSpaceChar(c));
        return res.toString();
    }

    public String nextLine() {
        int c = read();
        while (isEndline(c))
            c = read();
        StringBuilder res = new StringBuilder();
        do {
            res.appendCodePoint(c);
            c = read();
        } while (!isEndline(c));
        return res.toString();
    }
}
```

1.4 Troubleshooting

Source: KACTL

Pre-submit:

- Write a few simple test cases, if sample is not enough.
- Are time limits close? If so, generate max cases.
- Is the memory usage fine?
- Could anything overflow?
- Make sure to submit the right file.

Wrong answer:

- Print your solution! Print debug output, as well.
- Are you clearing all datastructures between test cases?
- Can your algorithm handle the whole range of input?
- Read the full problem statement again.
- Do you handle all corner cases correctly?
- Have you understood the problem correctly?
- Any uninitialized variables?
- Any overflows?
- Confusing N and M, i and j, etc.?
- Are you sure your algorithm works?
- What special cases have you not thought of?
- Are you sure the STL functions you use work as you think?
- Add some assertions, maybe resubmit.
- Create some testcases to run your algorithm on.
- Go through the algorithm for a simple case.
- Go through this list again.
- Explain your algorithm to a team mate.
- Ask the team mate to look at your code.

- Go for a small walk, e.g. to the toilet.
- Is your output format correct? (including whitespace)
- Rewrite your solution from the start or let a team mate do it.

Runtime error:

- Have you tested all corner cases locally?
- Any uninitialized variables?
- Are you reading or writing outside the range of any vector?
- Any assertions that might fail?
- Any possible division by 0? (mod 0 for example)
- Any possible infinite recursion?
- Invalidated pointers or iterators?
- Are you using too much memory?
- Debug with resubmits (e.g. remapped signals, see Various).

Time limit exceeded:

- Do you have any possible infinite loops?
- What is the complexity of your algorithm?
- Are you copying a lot of unnecessary data? (References)
- How big is the input and output? (consider scanf)
- Avoid vector, map. (use arrays/unordered map)
- What do your team mates think about your algorithm?

Memory limit exceeded:

- What is the max amount of memory your algorithm should need?
- Are you clearing all data structures between test cases?

2 Sorting And Searching (2)

2.1 Interval Cover

```
/**
* Description: Example of greedy algorithm
* Source: Own
* Verification:
    https://open.kattis.com/problems/intervalcover
   * actually, you need to account for A=B and add
       epsilons but w/e
*/
vi solve(double A, double B, vector<pair<pd,int>> in)
    { // cover [A,B] with intervals from in
    pair<double,int> mx = {A,-1};
    vi ans;
    int nex = 0;

    sort(all(in));
    while (mx.f < B) {
        double cur = mx.f;
        while (nex < sz(in) && in[nex].f.f <= cur)
            mx = max(mx,{in[nex].f.s,in[nex].s}), nex++;
        if (mx.f == cur) return {};
        ans.pb(mx.s);
    }

    return ans;
}
```

2.2 Binary Search

```
/**
* Description: Basic example of binary search
* Guess the Number
* https://open.kattis.com/problems/guess
*/

void binarySearch() {
    int lo = 1, hi = 1000;
    while (1) {
        int mid = (lo+hi)/2;
        cout << mid << endl;
        string res; cin >> res;
        if (res == "correct") return 0;
        else if (res == "lower") hi = mid-1;
        else lo = mid+1;
    }
}
```

3 Data Structures (2)

3.1 Set

3.1.1 Coordinate Compression

```
/**
* Description: Demonstrates use of map
* Source: Own
* Verification: POI 12 - The Bus
*/

void compress(vector<vi>& x, int ind) {
    map<int,int> m;
    for (auto& a: x) m[a[ind]] = 0;
    int co = 0; for (auto& a: m) a.s = co++;
    for (auto& a: x) a[ind] = m[a[ind]];
}
```

3.1.2 Map Comparator

```
/**
* Description: Custom comparator for map / set
* Source: StackOverflow
* Verification: ?
*/

struct cmp {
    bool operator()(const int& l, const int& r) const {
        return l > r;
    }
};

set<int,cmp> s;
map<int,int,cmp> m;
```

3.1.3 Unordered Map

```
/**
* Description: faster than standard unordered map
* Source: http://codeforces.com/blog/entry/62393
* Verification:
    http://codeforces.com/contest/966/problem/E
    * normal unordered map gets TLE
*/

struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
            chrono::steady_clock::now()
            .time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

template<class T> using um = unordered_map<ll, T,
    custom_hash>;
template<class T> using ht = gp_hash_table<ll, T,
    custom_hash>;

template<class T> T get(ht<T>& u, ll x) {
  if (u.find(x) == u.end()) return 0;
  return u[x];
}
```

4 Graphs Easy (2)

4.1 Traversal

4.1.1 BFS on Grid

```
/**
* Description: BFS through grid with fixed xdir and
    ydir arrays
* Source: Own
*/

const int xdir[4] = {0,1,0,-1}, ydir[4] = {1,0,-1,0};
int dist[21][21];

void bfs() {
       FOR(i,21) FOR(j,21) dist[i][j] = MOD;
       dist[10][10] = 0;

       queue<pi> todo; todo.push({10,10}); //
           initialize queue, distances
       while (sz(todo)) {
              pi x = todo.front(); todo.pop(); // pop
                  point from queue
              FOR(i,4) {
                     pi y = {x.f+xdir[i],x.s+ydir[i]};
                     if (y.f < 0 || y.f > 20 || y.s <
                         0 || y.s > 20) continue; //
                         ignore this point if it's
                         outside of grid
                     if (dist[y.f][y.s] == MOD) { //
                         test whether point has been
                         visited or not
                        dist[y.f][y.s] =
                            dist[x.f][x.s]+1;
                        todo.push(y); // push point
                            to queue
                     }
              }
       }

    assert(dist[4][5] == 11);
}
```

4.1.2 DFS

```
/**
* Description: print nodes of graph in depth-first
    order
* Source: Own
*/

bool visit[MX];
vi adj[MX];

void dfs(int node) {
    if (visit[node]) return;
    visit[node] = 1;
    for (int i: adj[node]) dfs(i);
```

```
    cout << node << "\n";
        // do stuff
}
```

4.2 Shortest Path (3)

4.2.1 Bellman-Ford

```
/**
* Description: Shortest Path w/ negative edge weights
    * Can be useful with linear programming
    * Constraints of the form x_i-x_j<k
* Source: Own
* Verification:
    https://open.kattis.com/problems/shortestpath3
*/

template<int SZ> struct BellmanFord {
    int n;
    bool bad[SZ];
    vector<pair<pi,int>> edge;
    ll dist[SZ];

    ll query(int x) {
        if (bad[x]) return -INF;
        return dist[x];
    }

    void init(int s) {
        FOR(i,n) dist[i] = INF, bad[i] = 0;
        dist[s] = 0;

        FOR(i,n) for (auto a: edge)
            if (dist[a.f.f] < INF) dist[a.f.s] =
                min(dist[a.f.s], dist[a.f.f]+a.s);

        for (auto a: edge) if (dist[a.f.f] < INF)
            if (dist[a.f.s] > dist[a.f.f]+a.s)
                bad[a.f.s] = 1;

        FOR(i,n) for (auto a: edge)
            if (bad[a.f.f]) bad[a.f.s] = 1;
    }
};
```

4.2.2 Dijkstra

```
/**
 * Description: shortest path
 * Source: ?
 * Verification: ?
 */

template<class T> using pqg =
    priority_queue<T,vector<T>,greater<T>>;

template<class T> T poll(pqg<T>& x) {
    T y = x.top(); x.pop();
    return y;
}

template<int SZ> struct Dijkstra {
    ll dist[SZ];
    vpi adj[SZ];
    pqg<pl> q;

    void addEdge(int A, int B, int C) {
        adj[A].pb({B,C}), adj[B].pb({A,C});
    }

    void init(int st) {
        fill_n(dist,SZ,INF);
        q = pqg<pl>(); q.push({dist[st] = 0,st});
        while (sz(q)) {
                auto x = poll(q);
                if (dist[x.s] < x.f) continue;
                for (auto y: adj[x.s]) if (x.f+y.s <
                    dist[y.f])
                    q.push({dist[y.f] =
                        x.f+y.s,y.f});
        }
    }
};
```

4.2.3 Floyd-Warshall

```
/**
* Description: All-Pairs Shortest Path
* Source: Own
* Verification:
    https://open.kattis.com/problems/allpairspath
*/

template<int SZ> struct FloydWarshall {
    int n; // vertices, edges, queries
    ll dist[SZ][SZ];
    bool bad[SZ][SZ];

    ll query(int x, int y) {
        if (bad[x][y]) return -INF;
        return dist[x][y];
    }

    void solve() {
        FOR(i,n) FOR(j,n) dist[i][j] = INF, bad[i][j]
            = 0;
        FOR(i,n) dist[i][i] = 0;
        FOR(i,m) {
            int u,v,w; cin >> u >> v >> w;
            dist[u][v] = min(dist[u][v],(ll)w);
        }

        FOR(k,n) FOR(i,n) FOR(j,n) if (dist[i][k] !=
            INF && dist[k][j] != INF)
            dist[i][j] =
                min(dist[i][j],dist[i][k]+dist[k][j]);
```

```
        FOR(k,n) FOR(i,n) FOR(j,n) if (dist[i][k] !=
            INF && dist[k][j] != INF)
            if (dist[i][j] > dist[i][k]+dist[k][j])
                bad[i][j] = 1;

        FOR(k,n) FOR(i,n) FOR(j,n) {
            if (dist[i][k] < INF && bad[k][j])
                bad[i][j] = 1;
            if (bad[i][k] && dist[k][j] < INF)
                bad[i][j] = 1;
        }
    }
};
```

4.3 Topological Sort (3)

```
/**
* Description: sorts vertices such that if there
    exists an edge x->y, then x goes before y
*/

template<int SZ> struct Topo {
    int N, in[SZ];
    vi res, adj[SZ];

    void addEdge(int x, int y) {
        adj[x].pb(y), in[y] ++;
    }

    void sort() {
        queue<int> todo;
        FOR(i,1,N+1) if (in[i] == 0) todo.push(i);
        while (sz(todo)) {
            int x = todo.front(); todo.pop();
            res.pb(x);
            for (int i: adj[x]) {
                in[i] --;
                if (!in[i]) todo.push(i);
            }
        }
    }
};
```

4.4 MST (3)

4.4.1 Connectivity Queries

```
/**
 * Description: For each pair of points, calculates
     the first time when they are connected
 * Source: Own
 * Verification:
     https://oj.uz/problem/view/COCI18_pictionary
 */

template<int SZ> struct ConnectQuery {
    int n,q; // vertices, edges, # queries
    vpi ed; // edges
    pi p[SZ]; // connectivity queries
    int l[SZ],r[SZ]; // left and right bounds for
        answer
    vi tri[SZ];

    bool left() {
        FOR(i,sz(ed)+1) tri[i].clear();
        bool ok = 0;
        FOR(i,q) if (l[i] != r[i]) {
            tri[(l[i]+r[i])/2].pb(i);
            ok = 1;
        }
        return ok;
    }

    void test() {
        DSU<SZ> D = DSU<SZ>();
        FOR(i,sz(ed)+1) {
            if (i) D.unite(ed[i-1].f,ed[i-1].s);
            for (int x: tri[i]) {
                if (D.get(p[x].f) == D.get(p[x].s))
                    r[x] = i;
                else l[x] = i+1;
            }
        }
    }

    void solve() {
        FOR(i,q) l[i] = 0, r[i] = sz(ed)+1;
        while (left()) test();
    }
};
```

4.4.2 DSU

```
/**
* Description: Disjoint Set Union
* Source: CSAcademy
* Verification: USACO superbull
*/

template<int SZ> struct DSU {
    int par[SZ], sz[SZ];
    DSU() {
        FOR(i,SZ) par[i] = i, sz[i] = 1;
    }

    int get(int x) { // path compression
        if (par[x] != x) par[x] = get(par[x]);
        return par[x];
    }

    bool unite(int x, int y) { // union-by-rank
        x = get(x), y = get(y);
        if (x == y) return 0;
        if (sz[x] < sz[y]) swap(x,y);
        sz[x] += sz[y], par[y] = x;
        return 1;
    }
```

```
};

// computes the minimum spanning tree in O(ElogE) time

template<class T> T kruskal(vector<pair<T,pi>> edge) {
    DSU<MX> D;
    sort(all(edge));
    T ans = 0;
    for (auto a: edge) if (D.unite(a.s.f,a.s.s)) ans
        += a.f; // edge is in MST
    return ans;
}
```

4.4.3 Manhattan MST (4)

```
/**
 * Description: Compute MST of points where edges are
    manhattan distances
 * Source: https://open.kattis.com/problems/gridmst
 * Verification:
    https://open.kattis.com/problems/gridmst, CSA 84
    The Sprawl
 */

struct {
    map<int,pi> m;

    void upd(int a, pi b) {
        auto it = m.lb(a);
        if (it != m.end() && it->s <= b) return;
        m[a] = b; it = m.find(a);
        while (it != m.begin() && prev(it)->s >= b)
            m.erase(prev(it));
    }

    pi query(int y) { // for all a > y find min
        possible value of b
        auto it = m.ub(y);
        if (it == m.end()) return {2*MOD,2*MOD};
        return it->s;
    }
} S;

void solve() {
    sort(all(ind),[](int a, int b) { return cur[a][0]
        > cur[b][0]; });
    S.m.clear();
    int nex = 0;
    trav(x,ind) { // cur[x][0] <= ?, cur[x][1] < ?
        while (nex < N && cur[ind[nex]][0] >=
            cur[x][0]) {
            int b = ind[nex++];
            S.upd(cur[b][1],{cur[b][2],b});
        }
        pi t = S.query(cur[x][1]);
        if (t.s != 2*MOD)
            ed.pb({(ll)t.f-cur[x][2],{x,t.s}});
    }
}

ll mst(vpi v) {
    N = sz(v); cur.resz(N); FOR(i,N) ind.pb(i);
    sort(all(ind),[&v](int a, int b) { return v[a] <
        v[b]; });
    FOR(i,N-1) if (v[ind[i]] == v[ind[i+1]])
        ed.pb({0,{ind[i],ind[i+1]}});

    FOR(i,2) { // it's probably ok to consider just
        two quadrants?
        FOR(i,N) {
            auto a = v[i];
            cur[i][2] = a.f+a.s;
        }
        FOR(i,N) { // first octant
            auto a = v[i];
            cur[i][0] = a.f-a.s;
            cur[i][1] = a.s;
        }
        solve();
        FOR(i,N) { // second octant
            auto a = v[i];
            cur[i][0] = a.f;
            cur[i][1] = a.s-a.f;
        }
        solve();
        trav(a,v) a = {a.s,-a.f}; // rotate 90
            degrees, repeat
    }

    return kruskal(ed);
}
```

5 Algorithm Design Topics (2)

5.1 Minimum Deque (3)

```
/**
* Description: maintains minimum of deque while adding
    elements to back or deleting elements from front
* Source: own
* Verification: Jan 18 Lifeguards
*/

template<class T> struct MinDeque {
    int lo = 0, hi = -1;
    deque<pair<T,int>> d;

    void ins(T x) { // add to back
        while (sz(d) && d.back().f >= x) d.pop_back();
        d.pb({x,++hi});
    }

    void del() { // delete from front
        if (d.front().s == lo++) d.pop_front();
    }

    T get() {
        return sz(d) ? d.front().f : MOD; // change
            based on T
```

```
    }
};
```

5.2 Ternary Search (4)

```
/**
* Description: use on functions which are strictly
    decreasing then strictly increasing
* Source: Own
*/

double eval(double x) {
    return (x-5)*(x-5);
}

double ternary(double l, double r) {
    if (abs(r-l) <= 1e-9) return (l+r)/2;
    double l1 = (2*l+r)/3, r1 = (l+2*r)/3;
    return eval(l1) < eval(r1) ? ternary(l,r1) :
        ternary(l1,r);
} // ternary(-100,100) = 5
```

6 Range Queries (2)

6.1 Static Array Queries

6.1.1 Prefix Sums

```
/**
 * Description: Calculates rectangle sums in constant
     time
 * Source: Own
 * Verification: POI 16 Ticket Inspector
 */

template<class T, int SZ> struct PrefixSums {
    T sum[SZ][SZ];
    PrefixSums () { memset(sum,0,sizeof sum); }
    void init() {
        FOR(i,1,SZ) FOR(j,1,SZ)
            sum[i][j] += sum[i][j-1]
            +sum[i-1][j]-sum[i-1][j-1];
    }
    T get(int X1, int X2, int Y1, int Y2) {
        return sum[X2][Y2]-sum[X1-1][Y2]
                -sum[X2][Y1-1]+sum[X1-1][Y1-1];
    }
};
```

6.1.2 Range Minimum Query (3)

```
/**
* Description: Supports 1D range minimum query in
    constant time.
* Source: Own
* Verification: http://wcipeg.com/problem/ioi1223
    * https://pastebin.com/ChpniVZL
*/

template<class T, int SZ> struct RMQ {
    T stor[SZ][32-__builtin_clz(SZ)];

    T comb(T a, T b) {
        return min(a,b);
    }

    void build(vector<T>& x) {
        FOR(i,sz(x)) stor[i][0] = x[i];
        FOR(j,1,32-__builtin_clz(SZ))
            FOR(i,SZ-(1<<(j-1)))
            stor[i][j] = comb(stor[i][j-1],
                        stor[i+(1<<(j-1))][j-1]);
    }

    T query(int l, int r) {
        int x = 31-__builtin_clz(r-l+1);
        return comb(stor[l][x],stor[r-(1<<x)+1][x]);
    }
};
```

6.1.3 Range Query (3)

```
/**
 * Description: constructs in O(nlogn) and queries a
     range in O(1)
 * Source: own
 * Verification:
     https://www.codechef.com/problems/SEGPROD
 */

template<class T, int SZ> struct RangeQuery {
    int n;
    T stor[SZ][32-__builtin_clz(SZ)], id = 1;
    vector<T> a;

    T comb (T a, T b) { return mul(a,b); } //
        associative operation

    void fill(int l, int r, int ind) {
        if (ind < 0) return;
        int m = (l+r)/2;
        T prod = id; FORd(i,l,m) stor[i][ind] = prod =
            comb(a[i],prod);
        prod = id; FOR(i,m,r) stor[i][ind] = prod =
            comb(prod,a[i]);
        fill(l,m,ind-1); fill(m,r,ind-1);
    }

    void init() {
        n = 1; while ((1<<n) < sz(a)) n ++;
        a.resz(1<<n); fill(0,(1<<n),n-1);
    }

    T query(int l, int r) {
        if (l == r) return a[l];
```

```
        int t = 31-__builtin_clz(r^l);
        return comb(stor[l][t],stor[r][t]);
    }
};
```

6.1.4 Sqrt Tree (6)

```
/**
 * Description: constructs in O(nloglogn) and queries
     a range in O(1)
   * O(nlogn) construction almost always suffices
 * Source:
     https://cp-algorithms.com/data_structures/sqrt-tree.html
 * Verification: ?
 */

template<int SZ> struct SqrtTree {
    int n, precomp[32-__builtin_clz(SZ)];
    vi suf[SZ], pre[SZ], block[SZ], a;
    vi levels;

    void build(int ind, int lev) {
        int lev2 = (lev+1)/2;
        FOR(i,1<<(lev-lev2)) {
            int ind2 = ind+(i<<lev2);
            int prod = 1;
            FOR(j,1<<lev2) {
                int cur = ind2+j; MUL(prod,a[cur]);
                suf[cur].pb(prod);
            }
            prod = 1;
            FORd(j,1<<lev2) {
                int cur = ind2+j; MUL(prod,a[cur]);
                pre[cur].pb(prod);
            }
        }
        FOR(i,1<<lev) block[ind+i].pb(1);
        FOR(i,1<<(lev-lev2)) {
            int prod = 1;
            FOR(j,i,1<<(lev-lev2)) {
                MUL(prod,pre[ind+(j<<lev2)].back());
                block[ind+(i<<lev2)+j].back() = prod;
            }
        }
    }

    void buildLevel(int lev) {
        levels.pb(lev);
        if (lev == 1) { levels.pb(0); return; }
        for (int i = 0; i < sz(a); i += (1<<lev))
            build(i,lev);
        buildLevel((lev+1)/2);
    }

    int queryBad(int l, int r) {
        int prod = 1; FOR(i,l,r+1) MUL(prod,a[i]);
        return prod;
    }

    int query(int l, int r) {
        int t = 31-__builtin_clz(l^r);
        if (t <= 0) return queryBad(l,r);
        int ind = precomp[t];
        int b0 = (l>>levels[ind])+1, b1 =
            (r>>levels[ind])-1;
        int ans = mul(pre[l][ind-1],suf[r][ind-1]);
        if (b0 <= b1)
            MUL(ans,block[(b0<<levels[ind])+b1%(1<<(levels[ind
        return ans;
    }

    void init() {
        FOR(i,SZ) suf[i].clear(), pre[i].clear(),
            block[i].clear();
        levels.clear();

        n = 1; while ((1<<n) < sz(a)) n ++;
        a.resz(1<<n);

        buildLevel(n);
        FORd(i,n) {
            if (i != n-1) precomp[i] = precomp[i+1];
            else precomp[i] = 0;
            while (levels[precomp[i]] > i) precomp[i]
                ++;
        }
    }
};
```

6.1.5 Wavelet Tree (6)

```
/**
* Description: Segment tree on values instead of
    indices
* Source: http://rachitiitr.blogspot.com/2017/06
   /wavelet-trees-wavelet-trees-editorial.html
* Verification: http://www.spoj.com/problems/MKTHNUM/
*/

template<int SZ> struct Wavelet {
    vi mapl[2*SZ], mapr[2*SZ], val[2*SZ];

    void build(int ind = 1, int L = 0, int R = SZ-1) {
        // build a wavelet tree
        if (ind == 1) { FOR(i,N) val[ind].pb(i); }
        if (L == R) return;
        int M = (L+R)/2;
        for (int i: val[ind]) {
            val[2*ind+(A[i] > M)].pb(i);
            mapl[ind].pb(sz(val[2*ind])-1);
            mapr[ind].pb(sz(val[2*ind+1])-1);
        }
        build(2*ind,L,M);
        build(2*ind+1,M+1,R);
    }

    int getl(int ind, int x) { return x < 0 ? -1 :
        mapl[ind][x]; }
```

```
    int getr(int ind, int x) { return x < 0 ? -1 :
        mapr[ind][x]; }

    int query(int lind, int rind, int k, int ind = 1,
        int L = 0, int R = SZ-1) { // how many <= mid
        with index <= r
        if (L == R) return L;

        int M = (L+R)/2;
        int t = getl(ind,rind)-getl(ind,lind-1);
        if (t >= k) return query(getl(ind,lind-1)+1,
                                 getl(ind,rind),k,2*ind,L,M);
        return query(getr(ind,lind-1)+1,
                     getr(ind,rind),k-t,2*ind+1,M+1,R);
    }
};
```

6.2 1D Range Queries (3)

6.2.1 Binary Indexed Tree

```
/**
 * Description: N-D range sum query with point update
 * Source: https://codeforces.com/blog/entry/64914
 * Verification: SPOJ matsum
 */

template <class T, int ...Ns> struct BIT {
    T val = 0;
    void upd(T v) { val += v; }
    T query() { return val; }
};

template <class T, int N, int... Ns> struct BIT<T, N,
    Ns...> {
    BIT<T,Ns...> bit[N + 1];
    template<typename... Args> void upd(int pos,
        Args... args) {
        for (; pos <= N; pos += (pos&-pos))
            bit[pos].upd(args...);
    }
    template<typename... Args> T sum(int r, Args...
        args) {
        T res = 0; for (; r; r -= (r&-r)) res +=
            bit[r].query(args...);
        return res;
    }
    template<typename... Args> T query(int l, int r,
        Args... args) {
        return sum(r,args...)-sum(l-1,args...);
    }
}; // BIT<int,10,10> gives a 2D BIT

template<class T, int SZ> struct BITrange {
    BIT<T,SZ> bit[2]; // piecewise linear functions
    void upd(int hi, T val) {
        bit[1].upd(1,val), bit[1].upd(hi+1,-val);
        bit[0].upd(hi+1,hi*val);
    }
    void upd(int lo, int hi, T val) { upd(lo-1,-val),
        upd(hi,val); }
    T sum(int x) { return
        bit[1].sum(x)*x+bit[0].sum(x); }
    T query(int x, int y) { return sum(y)-sum(x-1); }
}; // equivalent to 1D lazy segment tree for sum
```

6.2.2 Segment Tree

```
/*
* Source: http://codeforces.com/blog/entry/18051
* Description: 1D point update, range query
* Verification: SPOJ Fenwick
*/

template<class T, int SZ> struct Seg {
    T seg[2*SZ], MN = 0;

    Seg() {
        memset(seg,0,sizeof seg);
    }

    T comb(T a, T b) { return a+b; } // easily change
        this to min or max

    void upd(int p, T value) { // set value at
        position p
        for (seg[p += SZ] = value; p > 1; p >>= 1)
            seg[p>>1] = comb(seg[(p|1)^1],seg[p|1]); //
                non-commutative operations
    }

    void build() {
        FORd(i,SZ) seg[i] = comb(seg[2*i],seg[2*i+1]);
    }

    T query(int l, int r) { // sum on interval [l, r]
        T res1 = MN, res2 = MN; r++;
        for (l += SZ, r += SZ; l < r; l >>= 1, r >>=
            1) {
            if (l&1) res1 = comb(res1,seg[l++]);
            if (r&1) res2 = comb(seg[--r],res2);
        }
        return comb(res1,res2);
    }
};
```

6.2.3 Lazy Segment Tree (4)

```
/**
* Description: 1D range update, range query
* Source: USACO Counting Haybales
* Verification: SPOJ Horrible
*/

template<class T, int SZ> struct LazySegTree {
    T sum[2*SZ], mn[2*SZ], lazy[2*SZ]; // set SZ to a
        power of 2
```

```
    LazySegTree() {
        memset (sum,0,sizeof sum);
        memset (mn,0,sizeof mn);
        memset (lazy,0,sizeof lazy);
    }

    void push(int ind, int L, int R) {
        sum[ind] += (R-L+1)*lazy[ind];
        mn[ind] += lazy[ind];
        if (L != R) lazy[2*ind] += lazy[ind],
            lazy[2*ind+1] += lazy[ind];
        lazy[ind] = 0;
    }

    void pull(int ind) {
        sum[ind] = sum[2*ind]+sum[2*ind+1];
        mn[ind] = min(mn[2*ind],mn[2*ind+1]);
    }

    void build() {
        FORd(i,SZ) pull(i);
    }

    T qsum(int lo, int hi, int ind = 1, int L = 0, int
        R = SZ-1) {
        push(ind,L,R);
        if (lo > R || L > hi) return 0;
        if (lo <= L && R <= hi) return sum[ind];

        int M = (L+R)/2;
        return qsum(lo,hi,2*ind,L,M) +
            qsum(lo,hi,2*ind+1,M+1,R);
    }

    T qmin(int lo, int hi, int ind = 1, int L = 0, int
        R = SZ-1) {
        push(ind,L,R);
        if (lo > R || L > hi) return INF;
        if (lo <= L && R <= hi) return mn[ind];

        int M = (L+R)/2;
        return min(qmin(lo,hi,2*ind,L,M),
            qmin(lo,hi,2*ind+1,M+1,R));
    }

    void upd(int lo, int hi, ll inc, int ind = 1, int
        L = 0, int R = SZ-1) {
        push(ind,L,R);
        if (hi < L || R < lo) return;
        if (lo <= L && R <= hi) {
            lazy[ind] = inc;
            push(ind,L,R);
            return;
        }

        int M = (L+R)/2;
        upd(lo,hi,inc,2*ind,L,M);
            upd(lo,hi,inc,2*ind+1,M+1,R);
        pull(ind);
    }
};
```

6.2.4 Sparse Segment Tree (4)

```
/**
* Description: Does not allocate storage for nodes
    with no data
* Source: Own
* Verification: USACO Mowing the Field?
*/

const int SZ = 1<<20;

template<class T> struct node {
    T val;
    node<T>* c[2];

    node() {
        val = 0;
        c[0] = c[1] = NULL;
    }

    void upd(int ind, T v, int L = 0, int R = SZ-1) {
        // add v
        if (L == ind && R == ind) { val += v; return; }

        int M = (L+R)/2;
        if (ind <= M) {
            if (!c[0]) c[0] = new node();
            c[0]->upd(ind,v,L,M);
        } else {
            if (!c[1]) c[1] = new node();
            c[1]->upd(ind,v,M+1,R);
        }

        val = 0;
        if (c[0]) val += c[0]->val;
        if (c[1]) val += c[1]->val;
    }

    T query(int low, int high, int L = 0, int R =
        SZ-1) { // query sum of segment
        if (low <= L && R <= high) return val;
        if (high < L || R < low) return 0;

        int M = (L+R)/2;
        T t = 0;
        if (c[0]) t += c[0]->query(low,high,L,M);
        if (c[1]) t += c[1]->query(low,high,M+1,R);
        return t;
    }

    void UPD(int ind, node* c0, node* c1, int L = 0,
        int R = SZ-1) { // for 2D segtree
        if (L != R) {
            int M = (L+R)/2;
            if (ind <= M) {
                if (!c[0]) c[0] = new node();
                c[0]->UPD(ind,c0 ? c0->c[0] : NULL,c1 ?
                    c1->c[0] : NULL,L,M);
```

```
        } else {
            if (!c[1]) c[1] = new node();
            c[1]->UPD(ind,c0 ? c0->c[1] : NULL,c1 ?
                c1->c[1] : NULL,M+1,R);
        }
    }
    val = 0;
    if (c0) val += c0->val;
    if (c1) val += c1->val;
  }
};
```

6.2.5 Lazy Persistent Segment Tree (5)

```
/**
* Description: persistent segtree with lazy updates
* Sources: CF, Franklyn Wang
* Verification:
    https://codeforces.com/contest/1090/problem/G
   * Note: This implementation assumes that lazy[cur]
       is included in val[cur] before propagating cur.
   * If lazy[cur] is not included in val[cur], you
       must propagate children before pulling.
*/

template<class T, int SZ> struct pseg {
  static const int LIMIT = 10000000; // adjust
  int l[LIMIT], r[LIMIT], nex = 0;
  T val[LIMIT], lazy[LIMIT];

  //// HELPER
  int copy(int cur) {
    int x = nex++;
    val[x] = val[cur], l[x] = l[cur], r[x] =
        r[cur], lazy[x] = lazy[cur];
    return x;
  }
  T comb(T a, T b) { return min(a,b); }
  void pull(int x) { val[x] =
      comb(val[l[x]],val[r[x]]); }
  void push(int cur, int L, int R) {
    if (!lazy[cur]) return;
    if (L != R) {
      l[cur] = copy(l[cur]);
      val[l[cur]] += lazy[cur];
      lazy[l[cur]] += lazy[cur];

      r[cur] = copy(r[cur]);
      val[r[cur]] += lazy[cur];
      lazy[r[cur]] += lazy[cur];
    }
    lazy[cur] = 0;
  }

  //// IMPORTANT
  T query(int cur, int lo, int hi, int L, int R) {
    if (lo <= L && R <= hi) return val[cur];
    if (R < lo || hi < L) return INF;
    int M = (L+R)/2;
    return lazy[cur]+comb(query(l[cur],lo,hi,L,M),
        query(r[cur],lo,hi,M+1,R));
  }
  int upd(int cur, int lo, int hi, T v, int L, int
      R) {
    if (R < lo || hi < L) return cur;

    int x = copy(cur);
    if (lo <= L && R <= hi) { val[x] += v, lazy[x]
        += v; return x; }
    push(x,L,R);

    int M = (L+R)/2;
    l[x] = upd(l[x],lo,hi,v,L,M), r[x] =
        upd(r[x],lo,hi,v,M+1,R);
    pull(x); return x;
  }
  int build(vector<T>& arr, int L, int R) {
    int cur = nex++;
    if (L == R) {
      if (L < sz(arr)) val[cur] = arr[L];
      return cur;
    }

    int M = (L+R)/2;
    l[cur] = build(arr,L,M), r[cur] =
        build(arr,M+1,R);
    pull(cur); return cur;
  }

  //// PUBLIC
  vi loc;
  void upd(int lo, int hi, T v) {
    loc.pb(upd(loc.back(),lo,hi,v,0,SZ-1)); }
  T query(int ti, int lo, int hi) { return
      query(loc[ti],lo,hi,0,SZ-1); }
  void build(vector<T>& arr) {
    loc.pb(build(arr,0,SZ-1)); }
};
```

6.2.6 Segment Tree Beats (6)

```
/**
 * Description: Interval min modifications
 * Source: CF tutorial?
 * Verification:
     http://acm.hdu.edu.cn/showproblem.php?pid=5306
 */

template<int SZ> struct SegTreeBeats {
  int N;
  ll sum[2*SZ];
  int mx[2][2*SZ], maxCnt[2*SZ];

  void pull(int ind) {
    mx[0][ind] = max(mx[0][2*ind],mx[0][2*ind+1]);
    mx[1][ind] = max(mx[1][2*ind],mx[1][2*ind+1]);
    maxCnt[ind] = 0;

    FOR(i,2) {
```

```
        if (mx[0][2*ind^i] == mx[0][ind])
            maxCnt[ind] += maxCnt[2*ind^i];
        else mx[1][ind] =
            max(mx[1][ind],mx[0][2*ind^i]);
    }

    sum[ind] = sum[2*ind]+sum[2*ind+1];
}

void build(vi& a, int ind = 1, int L = 0, int R =
    -1) {
    if (R == -1) R += N;
    if (L == R) {
        mx[0][ind] = sum[ind] = a[L];
        maxCnt[ind] = 1; mx[1][ind] = -1;
        return;
    }

    int M = (L+R)/2;
    build(a,2*ind,L,M); build(a,2*ind+1,M+1,R);
        pull(ind);
}

void push(int ind, int L, int R) {
    if (L == R) return;
    FOR(i,2)
        if (mx[0][2*ind^i] > mx[0][ind]) {
            sum[2*ind^i] -= (ll)maxCnt[2*ind^i]*
                            (mx[0][2*ind^i]-mx[0][ind]);
            mx[0][2*ind^i] = mx[0][ind];
        }
}

void upd(int x, int y, int t, int ind = 1, int L =
    0, int R = -1) { // set a_i = min(a_i,t)
    if (R == -1) R += N;
    if (R < x || y < L || mx[0][ind] <= t) return;
    push(ind,L,R);
    if (x <= L && R <= y && mx[1][ind] < t) {
        sum[ind] -= (ll)maxCnt[ind]*(mx[0][ind]-t);
        mx[0][ind] = t;
        return;
    }
    if (L == R) return;
    int M = (L+R)/2;
    upd(x,y,t,2*ind,L,M);
        upd(x,y,t,2*ind+1,M+1,R); pull(ind);
}

ll qsum(int x, int y, int ind = 1, int L = 0, int
    R = -1) {
    if (R == -1) R += N;
    if (R < x || y < L) return 0;
    push(ind,L,R);
    if (x <= L && R <= y) return sum[ind];

    int M = (L+R)/2;
    return
        qsum(x,y,2*ind,L,M)+qsum(x,y,2*ind+1,M+1,R);
}
```

```
    int qmax(int x, int y, int ind = 1, int L = 0, int
        R = -1) {
        if (R == -1) R += N;
        if (R < x || y < L) return -1;
        push(ind,L,R);
        if (x <= L && R <= y) return mx[0][ind];

        int M = (L+R)/2;
        return max(qmax(x,y,2*ind,L,M),
            qmax(x,y,2*ind+1,M+1,R));
    }
};
```

6.3 2D Range Queries (4)

6.3.1 2D SegBIT

```
/**
* Description: Binary Indexed Tree of Segment Trees
* Source: USACO Mowing the Field
* Verification: ~
*/

const int SZ = 1<<17;

// struct Node

template<class T> struct SegBit {
    node<T> seg[SZ+1];

    SegBit() {
        FOR(i,SZ+1) seg[i] = node<T>();
    }

    void upd(int x, int y, int v) { // add v
        for (x++;x <= SZ; x += (x&-x)) seg[x].upd(y,v);
    }

    T query(int x, int y1, int y2) {
        T ret = 0;
        for (;x > 0; x -= (x&-x)) ret +=
            seg[x].query(y1,y2);
        return ret;
    }

    T query(int x1, int x2, int y1, int y2) { // query
        sum of rectangle
        return query(x2+1,y1,y2)-query(x1,y1,y2);
    }
};
```

6.3.2 2D Segment Tree

```
/**
* Source: USACO Mowing the Field
* Dependency: Sparse SegTree
*/
```

```
const int SZ = 1<<17;

template<class T> struct Node {
    node<T> seg;
    Node* c[2];

    void upd(int x, int y, T v, int L = 0, int R =
        SZ-1) { // add v
        if (L == x && R == x) {
            seg.upd(y,v);
            return;
        }

        int M = (L+R)/2;
        if (x <= M) {
            if (!c[0]) c[0] = new Node();
            c[0]->upd(x,y,v,L,M);
        } else {
            if (!c[1]) c[1] = new Node();
            c[1]->upd(x,y,v,M+1,R);
        }

        seg.UPD(y,c[0] ? &c[0]->seg : NULL,c[1] ?
            &c[1]->seg : NULL);
    }

    T query(int x1, int x2, int y1, int y2, int L = 0,
        int R = SZ-1) { // query sum of rectangle
        if (x1 <= L && R <= x2) return
            seg.query(y1,y2);
        if (x2 < L || R < x1) return 0;

        int M = (L+R)/2;
        T t = 0;
        if (c[0]) t += c[0]->query(x1,x2,y1,y2,L,M);
        if (c[1]) t += c[1]->query(x1,x2,y1,y2,M+1,R);
        return t;
    }
};
```

6.3.3 Merge-Sort Tree Offline

```
/**
 * Description: merge-sort tree with offline updates
    * O(nlogn) memory, O(nlog^2n) time with better
        constant
 * Source: own
 * Verification:
     https://codeforces.com/contest/1093/problem/E
 */

template<class T> struct BIT { // offline build
    vector<T> bit, vals;

    void build() {
        vals.pb(0);
        sort(all(vals));
            vals.erase(unique(all(vals)),vals.end());
        bit.resize(sz(vals));
    }

    int getInd(T k) {
        return ub(all(vals),k)-vals.begin()-1;
    }

    void upd(int k, T val) { // add val to index k
        k = getInd(k);
        for ( ;k < sz(vals); k += (k&-k)) bit[k] +=
            val;
    }

    T query(int k) {
        k = getInd(k);
        T temp = 0;
        for (;k;k -= (k&-k)) temp += bit[k];
        return temp;
    }

    T query(int l, int r) { return
        query(r)-query(l-1); } // range query [l,r]
};

template<class T, int SZ> struct mstree {
    BIT<T> val[SZ+1];

    void updPre(int x, int y) {
        for (int X = x; X <= SZ; X += X&-X)
            val[X].vals.pb(y);
    }

    void build() {
        FOR(i,SZ+1) val[i].build();
    }

    void upd(int x, int y, int t = 1) { //
        x-coordinate between 1 and SZ inclusive
        for (int X = x; X <= SZ; X += X&-X)
            val[X].upd(y,t);
    }

    int query(int x, int y) {
        int t = 0;
        for (;x > 0; x -= x&-x) t += val[x].query(y);
        return t;
    }

    int query(int lox, int hix, int loy, int hiy) { //
        query number of elements within a rectangle
        return query(hix,hiy)-query(lox-1,hiy)
            -query(hix,loy-1)+query(lox-1,loy-1);
    }
};
```

6.3.4 Merge-Sort Tree

```
/**
* Description: Similar to 2D segtree, less memory
* For more complex queries use a customized treap
* Verification:
    http://codeforces.com/contest/785/submission/33953058
```

```
*/

template<int SZ> struct mstree {
    Tree<pi> val[SZ+1]; // for offline queries use
        vector with binary search instead

    void upd(int x, int y, int t = 1) { //
        x-coordinate between 1 and SZ inclusive
        for (int X = x; X <= SZ; X += X&-X) {
            if (t == 1) val[X].insert({y,x});
            else val[X].erase({y,x});
        }
    }

    int query(int x, int y) {
        int t = 0;
        for (;x > 0; x -= x&-x) t +=
            val[x].order_of_key({y,MOD});
        return t;
    }

    int query(int lox, int hix, int loy, int hiy) { //
        query number of elements within a rectangle
        return query(hix,hiy)-query(lox-1,hiy)
            -query(hix,loy-1)+query(lox-1,loy-1);
    }
};
```

6.4 BBST (4)

6.4.1 Lazy Treap

```
/*
* Source:
    https://cp-algorithms.com/data_structures/treap.html
    + others
* Description: Easiest BBST
* Verification: http://www.spoj.com/problems/ORDERSET/
*/

namespace treap {
    typedef struct tnode* pt;

    struct tnode {
        int pri, val; pt c[2]; // essential
        int sz; ll sum; // for range queries
        bool flip; // lazy update

        tnode (int _val) {
            pri = rand()+(rand()<<15); val = _val; c[0]
                = c[1] = NULL;
            sz = 1; sum = val;
            flip = 0;
        }
    };

    int getsz(pt x) { return x?x->sz:0; }
    ll getsum(pt x) { return x?x->sum:0; }

    pt prop(pt x) {
        if (!x || !x->flip) return x;
        swap(x->c[0],x->c[1]);
        x->flip = 0;
        FOR(i,2) if (x->c[i]) x->c[i]->flip ^= 1;
        return x;
    }

    void tour(pt x, vi& v) {
        if (!x) return;
        prop(x);
        tour(x->c[0],v); v.pb(x->val); tour(x->c[1],v);
    }

    pt recalc(pt x) {
        assert(!x->flip);
        prop(x->c[0]), prop(x->c[1]);
        x->sz = 1+getsz(x->c[0])+getsz(x->c[1]);
        x->sum =
            x->val+getsum(x->c[0])+getsum(x->c[1]);
        return x;
    }

    pair<pt,pt> split(pt t, int v) { // >= v goes to
        the right
        if (!t) return {t,t};
        prop(t);
        if (t->val >= v) {
            auto p = split(t->c[0], v); t->c[0] = p.s;
            return {p.f, recalc(t)};
        } else {
            auto p = split(t->c[1], v); t->c[1] = p.f;
            return {recalc(t), p.s};
        }
    }

    pair<pt,pt> splitsz(pt t, int sz) {
        if (!t) return {t,t};
        prop(t);
        if (getsz(t->c[0]) >= sz) {
            auto p = splitsz(t->c[0], sz); t->c[0] =
                p.s;
            return {p.f, recalc(t)};
        } else {
            auto p = splitsz(t->c[1],
                sz-getsz(t->c[0])-1); t->c[1] = p.f;
            return {recalc(t), p.s};
        }
    }

    pt merge(pt l, pt r) {
        if (!l || !r) return l ? l : r;
        prop(l), prop(r);
        pt t;
        if (l->pri > r->pri) l->c[1] =
            merge(l->c[1],r), t = l;
        else r->c[0] = merge(l,r->c[0]), t = r;
        return recalc(t);
    }

    pt ins(pt x, int v) { // insert v
        auto a = split(x,v), b = split(a.s,v+1);
        return merge(a.f,merge(new tnode(v),b.s));
```

```
    }

    pt del(pt x, int v) { // delete v
        auto a = split(x,v), b = split(a.s,v+1);
        return merge(a.f,b.s);
    }
}

using namespace treap;
```

6.4.2 Link-Cut Tree (5)

```
/**
 * Sources: Dhruv Rohatgi,
 *          https://sites.google.com/site/kc97ble
 *          /container/splay-tree/splaytree-cpp-3
 * Verification: SPOJ DYNACON1, DYNALCA
 */

using namespace splayTree;

template<int SZ> struct LCT {
    ps S[SZ];
    LCT () { FOR(i,SZ) S[i] = new snode(i); }

    // disconnect x from d-th child
    void dis(ps x, int d) {
        ps y = x->c[d];
        if (x) x->c[d] = NULL, recalc(x);
        if (y) { y->p = NULL; if (d) y->pp = x; }
    }
    // set x to be child of pp
    void makeChild(ps x) { setLink(x->pp,x,1); x->pp =
        NULL; }
    // unlink x->pp from its preferred child, then set
        x to be preferred child
    void setPref(ps x) { splay(x->pp), dis(x->pp,1),
        makeChild(x), splay(x); }
    // x is brought to the root of auxiliary tree
    ps access(ps x) { dis(splay(x),1); while (x->pp)
        setPref(x); return x; }

    //////// UPDATES

    ps makeRoot(ps v) { access(v)->flip = 1; return
        access(v); }
    // make y the parent of x
    void link(ps x, ps y) { makeRoot(x)->pp = y; }
    // cut link between x and its parent
    void cut(ps x) { dis(access(x),0); }

    //////// QUERIES

    int getDepth(ps x) { access(x); return
        getsz(x->c[0]); }
    ps getRoot(ps x) { return farthest(access(x),0); }
    ps lca(ps x, ps y) {
        ps root = getRoot(y);
        if (farthest(splay(x),0) == root) return x;
        while (splay(x)->pp) {
            if (farthest(splay(x->pp),0) == root)
                return x->pp;
            setPref(splay(x));
        }
        return NULL;
    }
};
```

6.4.3 Splay Tree (5)

```
/**
 * Description: Treap alternative
 * Source: see LCT
 * Verification: ~
 */

namespace splayTree {
    typedef struct snode* ps;

    struct snode {
        int val, sz; // value, # nodes in subtree
        ps p, pp, c[2]; // parent, path-parent (for
            LCT), children
        bool flip; // for range flip

        snode (int val) : val(val) {
            sz = 1;
            p = pp = c[0] = c[1] = NULL;
            flip = 0;
        }
    };

    int getsz(ps x) { return x ? x->sz : 0; }
    int dir(ps x, ps y) { return x ? (x->c[1] == y) :
        -1; }
    ps recalc(ps x) {
        x->sz = 1+getsz(x->c[0])+getsz(x->c[1]);
        return x;
    }
    void prop(ps x) {
        if (!x || !x->flip) return;
        swap(x->c[0],x->c[1]);
        if (x->c[0]) x->c[0]->flip ^= 1;
        if (x->c[1]) x->c[1]->flip ^= 1;
        x->flip = 0;
    }
    void propAnc(ps x) { // propagate ancestors
        if (!x) return;
        if (x->p) propAnc(x->p);
        prop(x);
    }
    void tour(ps x, vi& v) {
        if (!x) return;
        tour(x->c[0],v); v.pb(x->val); tour(x->c[1],v);
    }

    void setLink(ps x, ps y, int d) { // x propagated
        if (x) x->c[d] = y, recalc(x);
        if (y) y->p = x;
    }
```

```
    void rot(ps x, int d) { // precondition: x &
        parents propagated
        ps y = x->c[d], z = x->p; prop(y);
        setLink(x, y->c[d^1], d);
        setLink(y, x, d^1);
        setLink(z, y, dir(z, x));
        y->pp = x->pp; x->pp = NULL; // set y to be
            parent of x
    }
    ps splay(ps x) {
        propAnc(x);
        while (x && x->p) {
            ps y = x->p, z = y->p;
            int dy = dir(y, x), dz = dir(z, y);
            if (!z) rot(y, dy);
            else if (dy == dz) rot(z, dz), rot(y, dy);
            else rot(y, dy), rot(z, dz);
        }
        return x;
    }
    ps farthest(ps x, int d) { // get leftmost or
        rightmost node
        prop(x); return
            x->c[d]?farthest(x->c[d],d):splay(x);
    }
}

using namespace splayTree;
```

7 DP (3)

7.1 Examples

7.1.1 Distinct Subsequences

```
/**
* Description: DP eliminates overcounting
* Verification: https://cses.fi/problemset/task/1149/
*/

using namespace modOp;

int distinct(string S) {
    vi tot(26);
    int ans = 1;
    for (char c: S) {
        int t = sub(ans,tot[c-'a']);
        AD(tot[c-'a'],t), AD(ans,t);
    }
    return ans;
}
```

7.1.2 Knapsack

```
/**
* Description: solves knapsack in pseudo-polynomial
    time
* Verification:
    https://open.kattis.com/problems/knapsack
*/

const int MX = 2001;

vi solve(int cap, vi v, vi w) {
        int dp[MX][MX]; FOR(i,cap+1) dp[0][i] = 0;

    FOR(i,sz(v)) {
        FOR(j,cap+1) dp[i+1][j] = dp[i][j];
        FOR(j,cap+1) if (w[i]+j <= cap)
            dp[i+1][w[i]+j] =
            max(dp[i+1][w[i]+j],dp[i][j]+v[i]);
    }

    vi ans;
    FORd(i,sz(v)) if (dp[i][cap] != dp[i+1][cap]) cap
        -= w[i], ans.pb(i);
    return ans;
}
```

7.1.3 Longest Common Subsequence

```
/**
* Description: Classic DP example
* Verification:
    https://pcs.cs.cloud.vt.edu/problems/224
*/

int lcs(string a, string b) {
        vi dp(sz(b)+1);
        for (char c: a) {
            FORd(i,sz(b)) if (b[i] == c) dp[i+1] =
                dp[i]+1;
            FOR(i,sz(b)) dp[i+1] = max(dp[i+1],dp[i]);
        }
        return dp[sz(b)];
}
```

7.1.4 Longest Increasing Subsequence

```
/**
* Description: DP with Binary Search
*/

vi bes = {INT_MIN}; // last term of increasing
    sequence with i terms

void ad(int x) { // add terms of sequence one by one
    int lo = lb(all(bes),x)-bes.begin();
    if (lo == sz(bes)) bes.pb(0);
    bes[lo] = x; // sz(bes)-1 is your current answer
}
```

7.1.5 Traveling Salesman (4)

```
/**
* Description: Bitset DP example
* Solves TSP for small N
*/

const int MX = 15;

int N, dp[MX][1<<MX], dist[MX][MX];

int solve() {
    FOR(i,N) FOR(j,1<<N) dp[i][j] = MOD;

    dp[0][1] = 0;
    FOR(j,1<<N) FOR(i,N) if (j&(1<<i))
        FOR(k,N) if (!(j&(1<<k)))
            dp[k][j^(1<<k)] = min(dp[k][j^(1<<k)],
                                  dp[i][j]+dist[i][k]);

    int ans = MOD;
    FOR(j,1,N) ans =
        min(ans,dp[j][(1<<N)-1]+dist[j][0]);
    return ans;
}
```

7.2 Divide And Conquer (4)

```
/**
* Source: Own
* Verification: CEOI 2004 Two Sawmills
*/

void divide(int lo, int hi, int L, int R) {
    if (lo > hi) return;

    int mid = (lo+hi)/2;
    pair<ll,int> tmp = {1e18,-1};
    FOR(i,max(mid+1,L),R+1)
        tmp = min(tmp,{calc(0,mid)+calc(mid+1,i)
               +calc(i+1,n),i});
    ans = min(ans,tmp.f);

    divide(lo,mid-1,L,tmp.s);
    divide(mid+1,hi,tmp.s,R);
}
```

7.3 SOS DP (5)

```
/**
 * Description: if you add one to dp[i]
   * it adds one to dp[j] for all j such that j&i = j
 * Source: Own
 * Verification: CF?
 */

void sos (vi& dp, int x = 1) { // x = -1 reverses
    int SZ = 31-__builtin_clz(sz(dp));
    FOR(i,SZ) FOR(j,1<<SZ) if (j&(1<<i))
        dp[j^(1<<i)] += x*dp[j];
}

vi andConv(vi a, vi b) {
    sos(a), sos(b);
    FOR(i,sz(a)) a[i] *= b[i];
    sos(a,-1);
    return a;
}
```

8 Trees (3)

8.1 Queries (3)

8.1.1 LCA with Binary Jumps

```
/**
* Description: calculates least common ancestor in
    tree with binary jumping
* Source: USACO Camp
* Verification: Debug the Bugs
*/

template<int SZ> struct LCA {
    const int MAXK = 32-__builtin_clz(SZ);

    int N, R = 1; // vertices from 1 to N, R = root
    vi adj[SZ];
    int par[32-__builtin_clz(SZ)][SZ], depth[SZ];

    void addEdge(int u, int v) {
        adj[u].pb(v), adj[v].pb(u);
    }

    void dfs(int u, int prev){
        par[0][u] = prev;
        depth[u] = depth[prev]+1;
        for (int v: adj[u]) if (v != prev) dfs(v, u);
    }

    void init(int _N) {
        N = _N;
        dfs(R, 0);
        FOR(k,1,MAXK) FOR(i,1,N+1)
            par[k][i] = par[k-1][par[k-1][i]];
    }

    int lca(int u, int v){
        if (depth[u] < depth[v]) swap(u,v);

        FORd(k,MAXK) if (depth[u] >= depth[v]+(1<<k))
            u = par[k][u];
        FORd(k,MAXK) if (par[k][u] != par[k][v]) u =
            par[k][u], v = par[k][v];

        if(u != v) u = par[0][u], v = par[0][v];
        return u;
    }
```

```
    int dist(int u, int v) {
        return depth[u]+depth[v]-2*depth[lca(u,v)];
    }
};
```

8.1.2 LCA with RMQ

```
/**
* Description: Euler Tour LCA w/ O(1) query
* Source: own
* Verification: Debug the Bugs
* Dependency: Range Minimum Query
*/

template<int SZ> struct LCA {
    vi adj[SZ];
    RMQ<pi,2*SZ> r;
    vpi tmp;
    int depth[SZ], pos[SZ];

    int N, R = 1;

    void addEdge(int u, int v) {
        adj[u].pb(v), adj[v].pb(u);
    }

    void dfs(int u, int prev){
        pos[u] = sz(tmp); depth[u] = depth[prev]+1;
        tmp.pb({depth[u],u});
        for (int v: adj[u]) if (v != prev) {
            dfs(v, u);
            tmp.pb({depth[u],u});
        }
    }

    void init(int _N) {
        N = _N;
        dfs(R, 0);
        r.build(tmp);
    }

    int lca(int u, int v){
        u = pos[u], v = pos[v];
        if (u > v) swap(u,v);
        return r.query(u,v).s;
    }

    int dist(int u, int v) {
        return depth[u]+depth[v]-2*depth[lca(u,v)];
    }
};
```

8.1.3 Small To Large Merging (4)

```
/**
* Description: offline subtree queries in O(Nlog^2N)
* To verify: January Easy 2018 - Shubham & Tree 1
*/

struct SmallToLarge {
    int val[MX];
    vi child[MX];
    map<int,int> dat[MX];

    void comb(int a, int b) {
        bool swa = 0;
        if (sz(dat[a]) < sz(dat[b])) swap(a,b), swa =
            1;
        for (auto& x: dat[b]) dat[a][x.f] += x.s;
        dat[b].clear();
        if (swa) swap(dat[a],dat[b]);
    }

    void process(int ind) {
        dat[ind][val[ind]] ++;
        for (int i: child[ind]) {
            process(i);
            comb(ind,i);
        }
        // now do stuff with values
    }
};
```

8.2 Tree Diameter (4)

```
/**
* Description: Calculates longest path in tree
* Source: Own
* Verification: http://www.spoj.com/problems/PT07Z/
*/

template<int SZ> struct TreeDiameter {
    int n, dist[SZ], pre[SZ];
    vi adj[SZ];

    void addEdge(int a, int b) {
        adj[a].pb(b), adj[b].pb(a);
    }

    void dfs(int cur) {
        for (int i: adj[cur]) if (i != pre[cur]) {
            pre[i] = cur;
            dist[i] = dist[cur]+1;
            dfs(i);
        }
    }

    void genDist(int cur) {
        memset(dist,0,sizeof dist);
        pre[cur] = -1;
        dfs(cur);
    }

    int diameterLength() {
        genDist(1);
        int bes = 0; FOR(i,1,n+1) if (dist[i] >
            dist[bes]) bes = i;
```

```
        genDist(bes); FOR(i,1,n+1) if (dist[i] >
            dist[bes]) bes = i;
        return dist[bes];
    }

    vi genCenter() {
        int t = diameterLength();
        int bes = 0; FOR(i,1,n+1) if (dist[i] >
            dist[bes]) bes = i;

        FOR(i,t/2) bes = pre[bes];
        if (t&1) return {bes,pre[bes]};
        return {bes};
    }
};
```

8.3 Advanced (4)

8.3.1 Centroid Decomposition

```
/**
* Source: own
* Verification:
    https://codeforces.com/contest/342/problem/E
* Description: can support tree path queries and
    updates
*/

template<int SZ> struct CentroidDecomp {
    bool done[SZ];
    int sub[SZ], par[SZ], ans[SZ];
    vi dist[SZ], adj[SZ], ANS[SZ];
    pi cen[SZ];

    void addEdge(int a, int b) { adj[a].pb(b),
        adj[b].pb(a); }

    void dfs (int no) {
        sub[no] = 1;
        for (int i: adj[no]) if (!done[i] && i !=
            par[no]) {
            par[i] = no;
            dfs(i);
            sub[no] += sub[i];
        }
    }

    void genDist(int par, int no) {
        for (int i: adj[no]) if (!done[i] && i != par)
            {
            cen[i] = cen[no];
            dist[i].pb(dist[no].back()+1);
            genDist(no,i);
        }
    }

    int getCentroid(int x) {
        par[x] = 0; dfs(x);
        int sz = sub[x];
        while (1) {
            pi mx = {0,0};
            for (int i: adj[x]) if (!done[i] && i !=
                par[x]) mx = max(mx,{sub[i],i});
            if (mx.f*2 > sz) x = mx.s;
            else return x;
        }
    }

    void solve (int x) { // call solve(1) to initialize
        x = getCentroid(x); done[x] = 1;
        dist[x].pb(0);
        for (int i: adj[x]) if (!done[i]) {
            cen[i] = {x,sz(ANS[x])};
            dist[i].pb(1);
            genDist(x,i);
            ANS[x].pb(0);
        }
        for (int i: adj[x]) if (!done[i]) solve(i);
    }

    void upd(int v) {
        pi V = {v,-1};
        for (int ind = sz(dist[v])-1; V.f; V =
            cen[V.f], ind --) {
            ans[V.f] ++;
            if (V.s != -1) ANS[V.f][V.s] ++;
        }
    }

    int query(int v) {
        pi V = {v,-1}; int ret = 0;
        for (int ind = sz(dist[v])-1; V.f; V =
            cen[V.f], ind --) {
            ret += ans[V.f];
            if (V.s != -1) ret -= ANS[V.f][V.s];
        }
        return ret;
    }
};
```

8.3.2 Heavy-Light Decomposition

```
/**
* Description: Heavy Light Decomposition
* Source: http://codeforces.com/blog/entry/22072
* Verification: USACO Grass Planting
*/

vector<vi> graph;

template<int SZ> struct HLD { // sum queries, sum
    updates
    int parent[SZ], heavy[SZ], depth[SZ];
    int root[SZ], treePos[SZ];
    LazySegTree<int,SZ> tree;

    void init() {
        int n = sz(graph)-1;
        FOR(i,1,n+1) heavy[i] = -1;
        parent[1] = -1, depth[1] = 0;
```

```
        dfs(1);
        for (int i = 1, currentPos = 0; i <= n; ++i)
            if (parent[i] == -1 || heavy[parent[i]]
                != i)
                for (int j = i; j != -1; j =
                    heavy[j]) {
                    root[j] = i;
                    treePos[j] = currentPos++;
                }
    }

    int dfs(int v) {
        int size = 1, maxSubtree = 0;
        for (auto u : graph[v]) if (u != parent[v]) {
            parent[u] = v;
            depth[u] = depth[v] + 1;
            int subtree = dfs(u);
            if (subtree > maxSubtree) heavy[v] = u,
                maxSubtree = subtree;
            size += subtree;
        }
        return size;
    }

    template <class BinaryOperation>
    void processPath(int u, int v, BinaryOperation op)
        {
        for (; root[u] != root[v]; v =
            parent[root[v]]) {
            if (depth[root[u]] > depth[root[v]])
                swap(u, v);
            op(treePos[root[v]], treePos[v]);
        }
        if (depth[u] > depth[v]) swap(u, v);
        op(treePos[u]+1, treePos[v]); // assumes
            values are stored in edges, not vertices
    }

    void modifyPath(int u, int v, int value) { // add
        one to vertices along path
        processPath(u, v, [this, &value](int l, int r)
            { tree.upd(l, r, value); });
    }

    ll queryPath(int u, int v) { // query sum of path
        ll res = 0;
        processPath(u, v, [this, &res](int l, int r) {
            res += tree.qsum(l, r); });
        return res;
    }
};
```

9 Strings (3)

9.1 Hashing

```
/**
 * Source: own
 * Description: Pairs reduce frequency of collision
 * Verification: ?Dec 17 Plat 1, CF Check Transcription
 */

using namespace pairOp;

const int tmp =
    chrono::high_resolution_clock::now().time_since_epoch().c

struct hsh {
    string S;
    vpmi pows, ipows, cum;
    pmi base = mp(948392576,tmp+1), invbase; //
        probably want to randomize base

    hsh() {}
    hsh(string s) { init(s); }

    void init(string _S) {
        invbase = {mi(1)/base.f,mi(1)/base.s};
        S = _S; pows.resz(sz(S)), ipows.resz(sz(S)),
            cum.resz(sz(S)+1);
        pows[0] = ipows[0] = {1,1};
        FOR(i,1,sz(S)) pows[i] = pows[i-1]*base,
            ipows[i] = ipows[i-1]*invbase;
        FOR(i,sz(S)) cum[i+1] =
            cum[i]+pows[i]*(S[i]-'a'+1);
    }

    pmi get(int l, int r) { return
        ipows[l]*(cum[r+1]-cum[l]); }

    int lcp(hsh& b) {
        int lo = 0, hi = min(sz(S),sz(b.S));
        while (lo < hi) {
            int mid = (lo+hi+1)/2;
            if (cum[mid] == b.cum[mid]) lo = mid;
            else hi = mid-1;
        }
        return lo;
    }
};
```

9.2 Z

```
/**
* Source: http://codeforces.com/blog/entry/3107
* Description: similar to KMP
* Verification: POI 12 Template
*/

vi z(string s) {
    int N = sz(s); s += '#';
    vi ans(N); ans[0] = N;
    while (s[1+ans[1]] == s[ans[1]]) ans[1] ++;

    int L = 1, R = ans[1];
    FOR(i,2,N) {
        if (i <= R) ans[i] = min(R-i+1,ans[i-L]);
        while (s[i+ans[i]] == s[ans[i]]) ans[i] ++;
        if (i+ans[i]-1 > R) L = i, R = i+ans[i]-1;
```

```
    }
    return ans;
}

vi getPrefix(string a, string b) { // find prefixes of
    a in b
    string s = a+"@"+b;
    vi t = z(s);
    return vi(t.begin()+sz(a)+1,t.end());
}

//
    pr(z("abcababcabcaba"),getPrefix("abcab","uwetrabcerabcab"));
```

9.3 Suffix Array

```
/**
* Description:
* Sources: SuprDewd, KACTL, majk, ekzhang
* Verification:
    *
        http://usaco.org/index.php?page=viewproblem2&cpid=768
        * https://pastebin.com/y2Z9FYr6
    * https://open.kattis.com/problems/suffixsorting
    * https://codeforces.com/contest/1090/problem/J
*/

struct LCP {
    string S; int N;
    vi sa, inv, lcp;
    RMQ<int,MX> R;

    void suffixArray() { // http://ekzlib.herokuapp.com
        sa.resz(N);
        vi classes(N);
        FOR(i,N) {
            sa[i] = N - 1 - i;
            classes[i] = S[i];
        }
        stable_sort(all(sa), [this](int i, int j) {
            return S[i] < S[j]; });
        for (int len = 1; len < N; len *= 2) {
            vi c(classes);
            FOR(i,N) {
                bool same = i && sa[i - 1] + len < N
                            && c[sa[i]] == c[sa[i - 1]]
                            && c[sa[i] + len / 2] ==
                                c[sa[i - 1] + len / 2];
                classes[sa[i]] = same ? classes[sa[i -
                    1]] : i;
            }
            vi cnt(N), s(sa);
            FOR(i,N) cnt[i] = i;
            FOR(i,N) {
                int s1 = s[i] - len;
                if (s1 >= 0) sa[cnt[classes[s1]]++] =
                    s1; // order pairs w/ same first
                    element by second element
            }
        }
    }

    void lcpArray() { // KACTL
        int h = 0;
        inv.resz(N), lcp.resz(N);
        FOR(i,N) inv[sa[i]] = i;
        FOR(i,N) if (inv[i]) {
            int p0 = sa[inv[i] - 1];
            while (max(i,p0)+h < N && S[i+h] ==
                S[p0+h]) h++;
            lcp[inv[i]] = h; // lcp of suffixes
                starting at p0 and i
            if (h) h--;
        }
    }

    void init(string _S) {
        S = _S; N = sz(S);
        suffixArray(); lcpArray();
        R.build(lcp);
    }

    int getLCP(int a, int b) {
        if (max(a,b) >= N) return 0;
        if (a == b) return N-a;
        int t0 = inv[a], t1 = inv[b];
        if (t0 > t1) swap(t0,t1);
        return R.query(t0+1,t1);
    }
};
```

9.4 Aho-Corasick

```
/**
 * Source: https://ideone.com/0cMjZJ
 * Verification: Kattis stringmultimatching
*/

template<int SZ> struct AhoCorasick {
    int link[SZ], dict[SZ], sz = 1, num = 0;
    vpi ind[SZ];
    map<char,int> to[SZ];
    vi oc[SZ];
    queue<int> q;

    AhoCorasick() {
        memset(link,0,sizeof link);
        memset(dict,0,sizeof dict);
    }

    void add(string s) {
        int v = 0;
        for(auto c: s) {
            if (!to[v].count(c)) to[v][c] = sz++;
            v = to[v][c];
        }
        dict[v] = v; ind[v].pb({++num,sz(s)});
    }

    void pushLinks() {
```

```
        link[0] = -1; q.push(0);
        while (sz(q)) {
            int v = q.front(); q.pop();
            for (auto it: to[v]) {
                char c = it.f; int u = it.s, j =
                    link[v];
                while (j != -1 && !to[j].count(c)) j =
                    link[j];
                if (j != -1) {
                    link[u] = to[j][c];
                    if (!dict[u]) dict[u] =
                        dict[link[u]];
                }
                q.push(u);
            }
        }
    }

    void process(int pos, int cur) { // process matches
        cur = dict[cur];
        while (cur) {
            for (auto a: ind[cur])
                oc[a.f].pb(pos-a.s+1);
            cur = dict[link[cur]];
        }
    }

    int nex(int pos, int cur, char c) {
        // get position after adding character
        // speed up with memoization
        while (cur != -1 && !to[cur].count(c)) cur =
            link[cur];
        if (cur == -1) cur = 0;
        else cur = to[cur][c];
        process(pos, cur);
        return cur;
    }
};
```

9.5 Manacher

```
/**
* Source: http://codeforces.com/blog/entry/12143
* Description: Calculates length of largest palindrome
    centered at each character of string
* Verification: http://www.spoj.com/problems/MSUBSTR/
*/

vi manacher(string s) {
    string s1 = "@";
    for (char c: s) s1 += c, s1 += "#";
    s1[s1.length()-1] = '&';

    vi ans(s1.length()-1);
    int lo = 0, hi = 0;
    FOR(i,1,s1.length()-1) {
        if (i != 1) ans[i] = min(hi-i,ans[hi-i+lo]);
        while (s1[i-ans[i]-1] == s1[i+ans[i]+1])
            ans[i] ++;
        if (i+ans[i] > hi) lo = i-ans[i], hi =
            i+ans[i];
    }

    ans.erase(ans.begin());
    FOR(i,sz(ans)) if ((i&1) == (ans[i]&1)) ans[i] ++;
        // adjust lengths
    return ans;
}

// vi v = manacher("abacaba"); cout << v;
```

9.6 Minimum Rotation

```
/**
* Description: Calculates min rotation of string in
    O(N)
* Source: KACTL
* Verification: ?
*/

int minRotation(string s) {
        int a = 0, N = sz(s); s += s;
        FOR(b,N) FOR(i,N) {
                if (a+i == b || s[a+i] < s[b+i]) {b +=
                    max(0, i-1); break;}
                if (s[a+i] > s[b+i]) { a = b; break; }
        }
        return a;
}
```

9.7 Palindromic Tree

```
/**
* Description: maintains tree of palindromes
* Source: http://codeforces.com/blog/entry/13959
* Verification:
    https://oj.uz/problem/view/APIO14_palindrome
*/

template<int SZ> struct palTree {
    static const int sigma = 26;

    int s[SZ], len[SZ], link[SZ], to[SZ][sigma],
        oc[SZ];
    int n, last, sz;

    palTree() {
        s[n++] = -1;
        link[0] = 1;
        len[1] = -1;
        sz = 2;
    }

    int getLink(int v) {
        while (s[n-len[v]-2] != s[n-1]) v = link[v];
        return v;
```

```
    }

    void addChar(int c) {
        s[n++] = c;
        last = getLink(last);
        if (!to[last][c]) {
            len[sz] = len[last]+2;
            link[sz] = to[getLink(link[last])][c];
            to[last][c] = sz++;
        }
        last = to[last][c];
        oc[last] ++;
    }

    void prop() { // number of occurrences of each
        palindrome
        vpi v;
        FOR(i,2,sz) v.pb({len[i],i});
        sort(all(v)); reverse(all(v));
        for (auto a: v) oc[link[a.s]] += oc[a.s];
    }
};
```

9.8 Bitset Trie (4)

```
/**
* Source: Algorithms Gym
* Verification: January Easy 2018 - Shubham and
    Subarray Xor
*/

template<int MX> struct tri {
    static const int MXBIT = 60;
    int trie[MX][2], nex = 0; // easily changed to
        character
    int sz[MX];

    tri() {
        memset(trie,0,sizeof trie);
    }

    void ins(ll x, int a = 1) { // insert or delete
        int cur = 0; sz[cur] += a;
        FORd(i,MXBIT) {
            int t = (x&(1LL<<i))>>i;
            if (!trie[cur][t]) trie[cur][t] = ++nex;
            cur = trie[cur][t];
            sz[cur] += a;
        }
    }

    ll test(ll x) { // compute max xor
        if (sz[0] == 0) return -INF;
        int cur = 0;
        FORd(i,MXBIT) {
            int t = ((x&(1LL<<i))>>i) ^ 1;
            if (!trie[cur][t] || !sz[trie[cur][t]]) t
                ^= 1;
            cur = trie[cur][t];
            if (t) x ^= (1LL<<i);
        }
        return x;
    }
};
```

9.9 Reverse Burrows-Wheeler (6)

```
/**
* Description: Reverse Burrows-Wheeler
* Verification: https://cses.fi/problemset/task/1113/
*/

string reverseBW(string s) {
        vector<pair<char,int>> v;
        int nex[sz(s)];

        FOR(i,sz(s)) v.pb({s[i],i});
        sort(all(v));
        FOR(i,sz(v)) nex[i] = v[i].s;

        int cur = nex[0];
        string ret;
        while (cur != 0) {
                ret += v[cur].f;
                cur = nex[cur];
        }
        return ret;
}
```

9.10 Lyndon Factorization (6)

```
/**
 * Description: Compute Lyndon Factorization in O(n)
 * Source:
     https://cp-algorithms.com/string/lyndon_factorization.ht
 * Verification:
     https://uva.onlinejudge.org/index.php?option=onlinejudge
 */

vector<string> duval(string const& s) {
    int n = s.size();
    vector<string> factorization;
    for (int i = 0; i < n; ) {
        int j = i + 1, k = i;
        for (;j < n && s[k] <= s[j]; ++j) {
            if (s[k] < s[j]) k = i;
            else k++;
        }
        while (i <= k) {
            factorization.pb(s.substr(i, j - k));
            i += j - k;
        }
    }
    return factorization;
}
```

9.11 String Repetitions (6)

```
/**
 * Description: Main-Lorentz Algorithm
   * Finds all (x,y) such that s.substr(x,y-1) ==
       s.substr(x+y,y-1)
   *
       https://cp-algorithms.com/string/main_lorentz.html
 * Source: own
 * Verification:
     http://codeforces.com/contest/1043/problem/G
 */

template<int SZ> struct stringRepeat {
    string S;
    vector<array<int,3>> al;

    vector<array<int,3>> solveLeft(string s, int m) {
        vector<array<int,3>> v;

        vi v2 =
            get(string(s.begin()+m+1,s.end()),string(s.begin(),s.begin()+m+1));
        string V = string(s.begin(),s.begin()+m+2);
            reverse(all(V)); vi v1 = z(V);
            reverse(all(v1));

        FOR(i,m+1) if (v1[i]+v2[i] >= m+2-i) {
            int lo = max(1,m+2-i-v2[i]), hi =
                min(v1[i],m+1-i);
            lo = i-lo+1, hi = i-hi+1; swap(lo,hi);
            v.pb({2*(m+1-i),lo,hi});
        }

        return v;
    }

    void divi(int l, int r) {
        if (l == r) return;
        int m = (l+r)/2; divi(l,m); divi(m+1,r);

        string t = string(S.begin()+l,S.begin()+r+1);
        m = (sz(t)-1)/2;
        auto a = solveLeft(t,m);
        reverse(all(t));
        auto b = solveLeft(t,sz(t)-2-m);

        for (auto x: a) al.pb({x[0],x[1]+l,x[2]+l});
        for (auto x: b) {
            int ad = r-x[0]+1;
            al.pb({x[0],ad-x[2],ad-x[1]});
        }
    }

    void init(string _S) {
        S = _S;
        divi(0,sz(S)-1);
    }

    vi genLen() { // for each index
        priority_queue<pi,vpi,greater<pi>> m;
            m.push({MOD,MOD});
        vpi ins[SZ]; for (auto a: al)
            ins[a[1]].pb({a[0],a[2]});
        vi len(SZ);
        FOR(i,sz(S)) {
            for (auto j: ins[i]) m.push(j);
            while (m.top().s < i) m.pop();
            len[i] = m.top().f;
        }
        return len;
    }
};
```

9.12 Suffix Automaton (6)

```
/**
 * Description: Suffix Automaton
 * Source:
     https://cp-algorithms.com/string/suffix-automaton.html
 * Verification: https://www.spoj.com/problems/SUBLEX/
 */
template<int SZ> struct sa {
    struct state {
        int len, link;
        map<char, int> next;
    };

    state st[SZ * 2];
    int sz, last;
    ll ans[SZ * 2]; // number of distinct substrings
        from current pos

    void extend(char c) {
        int cur = sz++;
        st[cur].len = st[last].len + 1;
        int p = last;
        while (p != -1 && !st[p].next.count(c)) {
            st[p].next[c] = cur;
            p = st[p].link;
        }
        if (p == -1) {
            st[cur].link = 0;
        } else {
            int q = st[p].next[c];
            if (st[p].len + 1 == st[q].len) {
                st[cur].link = q;
            } else {
                int clone = sz++;
                st[clone] = st[q]; st[clone].len =
                    st[p].len + 1;
                while (p != -1 && st[p].next[c] == q) {
                    st[p].next[c] = clone;
                    p = st[p].link;
                }
                st[q].link = st[cur].link = clone;
            }
        }
        last = cur;
    }
```

```
    void init(string s) {
        st[0].len = 0, st[0].link = -1; sz ++; last =
            0;
        trav(x,s) extend(x);
    }

    ll gen(int x) {
        if (ans[x]) return ans[x];
        ans[x] = 1; trav(y,st[x].next) ans[x] +=
            gen(y.s);
        return ans[x];
    }
};
```

10 Math (4)

10.1 Number Theory

10.1.1 Basic Factoring

```
/**
* Description: factors N in O(sqrtN) time
* Source: Own
* Verification: ?
*/

namespace factor1 {
    vpl factor(ll x) { // x <= 10^{14} is fine
        vpl pri;

        for (ll i = 2; i*i <= x; ++i) if (x % i == 0) {
            int t = 0;
            while (x % i == 0) x /= i, t ++;
            pri.pb({i,t});
        }

        if (x > 1) pri.pb({x,1});
        return pri;
    }

    /* Note:
     * number of operations needed s.t.
     *              phi(phi(...phi(n)...))=1
     * is O(log n).
     * Euler's theorem: a^{\phi(p)}\equiv 1 (mod p),
         gcd(a,p)=1
     */

    ll phi(ll x) {
        for (auto a: factor(x)) x /= a.f, x *= a.f-1;
        return x;
    }

    void tour(vpl& v, vl& V, int ind, ll cur) {
        if (ind == sz(v)) V.pb(cur);
        else {
            ll mul = 1;
            FOR(i,v[ind].s+1) {
                tour(v,V,ind+1,cur*mul);
                mul *= v[ind].f;
            }
        }
    }

    vl getDivi(ll x) {
        vpl v = factor(x);
        vl V; tour(v,V,0,1); sort(all(V));
        return V;
    }
}

using namespace factor1;
```

10.1.2 Modular Int

```
/**
 * Description: operations with modular arithmetic
 * Source: https://codeforces.com/blog/entry/63903
 * Verification: ? see NTT
 */

int invGeneral(int a, int b) {
    a %= b; if (a == 0) return b == 1 ? 0 : -1;
    int x = invGeneral(b,a); return x == -1 ? -1 :
        ((1-(ll)b*x)/a+b)%b;
}

struct mi {
    int val, mod = -1;
    // if mod is determined at compile time, use
        static const
    // -1 means mod has not been determined

    void setMod(int m) {
        if (m == -1 || mod == m) return;
        mod = m; val %= mod; if (val < 0) val += mod;
    }
    mi(int v = 0, int m = -1) : val(v) { setMod(m); }

    operator int() const { return val; }
    friend ostream& operator<<(ostream& os, const mi&
        a) { return os << a.val; }
    friend bool operator==(const mi& a, const mi& b) {
        return a.val == b.val; }
    friend bool operator!=(const mi& a, const mi& b) {
        return !(a == b); }

    friend void check(mi& a, mi& b) { // make sure all
        operations are valid
        assert min(a.mod,b.mod) == -1 || a.mod !=
            b.mod;
        int mod = max(a.mod,b.mod); if (mod == -1) mod
            = MOD;
        a.setMod(mod), b.setMod(mod);
    }
    mi& operator+=(mi b) { check(b); val += b; if (val
        >= mod) val -= mod; return *this; }
    mi& operator-=(mi b) { check(b); val -= b; if (val
        < 0) val += mod; return *this; }
```

```
    mi& operator*=(mi b) { check(b); val =
        (ll)val*b%mod; return *this; }
    friend mi exp(mi b, ll p) {
        mi ans = mi(1,b.mod);
        for (; p; p /= 2, b *= b) if (p&1) ans *= b;
        return ans;
    }
    friend mi inv(const mi& b) { return
        exp(b,b.mod-2); }
    mi& operator/=(int b) { return *this *=
        inv(mi(b,mod)); }

    friend mi operator+(mi a, const mi& b) { return a
        += b; }
    friend mi operator-(mi a, int b) { return a -= b; }
    friend mi operator-(const mi& a) { return
        mi(0,a.mod)-a; }
    friend mi operator*(mi a, int b) { return a *= b; }
    friend mi operator/(mi a, int b) { return a /= b; }
};

typedef pair<mi,mi> pmi;
typedef vector<mi> vmi;
typedef vector<pmi> vpmi;

vmi toVmi(vi v) { vmi V(sz(v)); FOR(i,sz(V)) V[i] =
    mi(v[i],MOD); return V; }
vi toVi(vmi V) { vi v(sz(V)); FOR(i,sz(v)) v[i] =
    V[i].val; return v; }
```

10.1.3 Russian Peasant Multiplication

```
/**
 * Description: Russian Peasant Multiplication
   * multiply two 64-bit integers mod another if
       128-bit is not available
 * Source: KACTL
 * Verification: ?
 */

typedef unsigned long long ul;

namespace rpm {
    const int bits = 14; // if all numbers are less
        than 2^k, set bits = 64-k
    const ul po = (ul)1<<bits;

    ul mod_mul(ul a, ul b, ul &c) { // return
        (__int128(a)*b) % c;
        ul x = 0;
        for (; b; b >>= bits, a = (a << bits) % c)
            x = (x + (a * (b & (po - 1))) % c) % c;
        return x;
    }

    ul mod_pow(ul a, ul b, ul mod) {
        if (b == 0) return 1;
        ul res = mod_pow(a, b / 2, mod);
        res = mod_mul(res, res, mod);
        if (b & 1) return mod_mul(res, a, mod);
        return res;
    }
}

using namespace rpm;
```

10.1.4 Sieve of Eratosthenes

```
/**
 * Description: Tests primality up to n in
     O(nlog(logn))
 * Source: KACTL
 * Verification:
     https://open.kattis.com/problems/primesieve
 */

template<int SZ> struct Sieve {
        bitset<SZ> comp;
        vi pr;
        // int sp[SZ];

        Sieve() {
                for (int i = 2; i*i < SZ; ++i) if
                    (!comp[i])
                        for (int j = i*i; j < SZ; j +=
                            i) comp[j] = 1;
                FOR(i,2,SZ) if (!comp[i]) pr.pb(i);

                /*FOR(i,2,SZ) { // O(N) sieve
                        if (sp[i] == 0) { sp[i] = i;
                            pr.pb(i); }
                        for (int p : pr) {
                                if (p > sp[i] || i*p >=
                                    SZ) break;
                                sp[i*p] = p;
                        }
                }*/
        }
};
```

10.1.5 Combinations (5)

```
/**
 * Description: extends Combo to all natural numbers
 * Source: Own
 * Verification: ? https://dmoj.ca/problem/tle17c4p5
 */

using namespace modOp;
using namespace factor1;

template<int SZ> struct Combo {
    int MOD, fac[SZ+1], ifac[SZ+1];
    vpl factors;
    vi cnt[SZ+1];

    void init(ll _MOD) {
        MOD = _MOD; factors = factor(MOD);
```

```
        cnt[0].resize(sz(factors));

        fac[0] = ifac[0] = 1;
        FOR(i,1,SZ+1) {
            cnt[i] = cnt[i-1];

            int I = i;
            FOR(j,sz(factors))
                while (I % factors[j].f == 0)
                    I /= factors[j].f, cnt[i][j] ++;

            fac[i] = mul(I,fac[i-1],MOD), ifac[i] =
                invGeneral(fac[i],MOD);
        }
    }

    ll comb(ll a, ll b) {
        if (a < b || b < 0) return 0;
        ll tmp =
            mul(mul(fac[a],ifac[b],MOD),ifac[a-b],MOD);
        FOR(i,sz(factors)) {
            int t = cnt[a][i]-cnt[a-b][i]-cnt[b][i];
            tmp = mul(tmp,po(factors[i].f,t),MOD);
        }
        return tmp;
    }
};
```

10.1.6 Discrete Logarithm (5)

```
/**
 * Description: find k such that root^k%mod=x
        * mod is prime, root is primitive
        * meet in the middle: O(sqrt(mod))
 * Source: Own
 * Verification: ?
 */

// dependency: mi

struct DiscreteLog {
        int mod, root, block;
        vmi invy;
        unordered_map<int,int> u;

        int query(int x) {
                FOR(i,block) {
                        int X = invy[i]*x;
                        if (u.count(X)) return
                            i*block+u[X];
                }
                return -1;
        }

        void init(int m, int r) : mod(m), root(r) {
                u.clear(); block = sqrt(mod)+1;

                mi cur(1,mod); FOR(i,block) u[cur] = i,
                    cur *= root;
                cur = inv(cur,mod);
                invy.resz(block); invy[0] = 1;
                    FOR(i,1,block) invy[i] =
                    cur*invy[i-1];
        }
};
```

10.1.7 Chinese Remainder Theorem (6)

```
/**
 * Description: Chinese Remainder Theorem
 * Source: Own
 * Verification: ? Kattis generalchineseremainder
 */

using namespace modOp;

pl solve(pl a, pl b) {
    ll g = __gcd(a.s,b.s), l = a.s*b.s/g;
    if ((b.f-a.f) % g != 0) return {-1,-1};
    ll A = a.s/g, B = b.s/g;
    ll mul = (b.f-a.f)/g*invGeneral(A%B,B) % B;
    return {((mul*a.s+a.f)%l+l)%l,l};
}
```

10.1.8 Faster Factoring (6)

```
/**
 * Source: KACTL
 * Description: Factors integers up to 2^{60}
 * Verification: https://www.spoj.com/problems/FACT0/
 *
     https://codeforces.com/contest/1033/submission/44009089
       * is probably faster
 */

using namespace rpm;

namespace factor2 {
        Sieve<1<<20> S = Sieve<1<<20>(); // should
            take care of all primes up to n^(1/3)

        bool prime(ll p) { // miller-rabin
                if (p == 2) return true;
                if (p == 1 || p % 2 == 0) return false;
                ll s = p - 1;
                while (s % 2 == 0) s /= 2;
                FOR(i,15) {
                        ll a = rand() % (p - 1) + 1, tmp
                            = s;
                        ll mod = mod_pow(a, tmp, p);
                        while (tmp != p - 1 && mod != 1
                            && mod != p - 1) {
                                mod = mod_mul(mod, mod,
                                    p);
                                tmp *= 2;
                        }
```

```
                if (mod != p - 1 && tmp % 2 ==
                    0) return false;
            }
            return true;
        }

        ll f(ll a, ll n, ll &has) { return (mod_mul(a,
            a, n) + has) % n; }

        vpl factor2(ll d) {
            vpl res;

            vi& pr = S.pr;
            for (int i = 0; i < sz(pr) &&
                pr[i]*pr[i] <= d; i++) if (d %
                pr[i] == 0) {
                int co = 0;
                while (d % pr[i] == 0) d /=
                    pr[i], co ++;
                res.pb({pr[i],co});
            }

            if (d > 1) { // d is now a product of
                at most 2 primes.
                if (prime(d)) res.pb({d,1});
                else while (1) {
                    ll has = rand() % 2321 +
                        47;
                    ll x = 2, y = 2, c = 1;
                    for (; c == 1; c =
                        __gcd((y > x ? y - x
                        : x - y), d)) {
                        x = f(x, d, has);
                        y = f(f(y, d,
                            has), d, has);
                    }
                    if (c != d) {
                        d /= c; if (d > c)
                            swap(d,c);
                        if (c == d)
                            res.pb({c,2});
                        else res.pb({c,1}),
                            res.pb({d,1});
                        break;
                    }
                }
            }

            return res;
        }
}
```

10.1.9 Modular Sqrt (6)

```
/**
 * Description: find sqrt of integer via a prime
 * Source:
     http://www.math.vt.edu/people/brown/class_homepages/shanks_tonelli.pdf
 * Verification: ?
     https://www.spoj.com/problems/CRYPTO1
 */

int modsqrt(int a) {
    int p = exp(a,(MOD-1)/2); if (p != 1) return p ==
        0 ? 0 : -1;
    int s = MOD-1, e = 0; while (s % 2 == 0) s /= 2, e
        ++;
    int n = 1; while (exp(n,(MOD-1)/2) == 1) n ++; //
        find non-square residue

    mi x = exp(a,(s+1)/2), b = exp(a,s), g = exp(n,s),
        r = e;
    while (1) {
        mi B = b; int m = 0; while (B != 1) B *= B, m
            ++;
        if (m == 0) break;
        FOR(i,r-m-1) g *= g;
        x *= g; g *= g; b *= g; r = m;

        /* Explanation:
         * Initially, x^2=ab, ord(b) = 2^m, ord(g) =
             2^r where m<r
         * g = g^{2^{r-m-1}} -> ord(g) = 2^{m+1}
         * if x'=x*g, then b' = b*g^2
               (b')^{2^{m-1}} = (b*g^2)^{2^{m-1}}
                              = b^{2^{m-1}}*g^{2^m}
                              = -1*-1
                              = 1
           -> ord(b')|ord(b)/2
         * m decreases by at least one each iteration
         */
    }
    return min((int)x,MOD-(int)x);
}
```

10.1.10 Order (6)

```
/**
 * Description: Calculates smallest P such that x^P
     equiv 1 (mod p)
 * Source: Own
 * Verification: ?
 */

using namespace rpm;
using namespace factor1;

ll order(ll x, ll p) {
    if (__gcd(x,p) != 1) return 0;
    ll P = phi(p);
    auto a = fac(P);

    for (auto t: a) while (P % t.f == 0 &&
        mod_pow(x,P/t.f,p) == 1) P /= t.f;
    return P;
}
```

10.2 Matrix

10.3 Operators

10.4 Structs

10.4.1 Big Integer

```
/**
 * Description: ?
 * Source: https://github.com/indy256/codelibrary/
           blob/master/cpp/numbertheory/bigint.cpp
 * Verification: ?
 */

namespace bigint {
    // base and base_digits must be consistent
    constexpr int base = 1000000000;
    constexpr int base_digits = 9;

    struct bigint {
        // value == 0 is represented by empty z
        vector<int> z; // digits

        // sign == 1 <==> value >= 0
        // sign == -1 <==> value < 0
        int sign;

        bigint() : sign(1) {
        }

        bigint(long long v) {
            *this = v;
        }

        bigint &operator=(long long v) {
            sign = v < 0 ? -1 : 1;
            v *= sign;
            z.clear();
            for (; v > 0; v = v / base)
                z.push_back((int) (v % base));
            return *this;
        }

        bigint(const string &s) {
            read(s);
        }

        bigint &operator+=(const bigint &other) {
            if (sign == other.sign) {
                for (int i = 0, carry = 0; i <
                     other.z.size() || carry; ++i) {
                    if (i == z.size())
                        z.push_back(0);
                    z[i] += carry + (i < other.z.size()
                         ? other.z[i] : 0);
                    carry = z[i] >= base;
                    if (carry)
                        z[i] -= base;
                }
            } else if (other != 0 /* prevent infinite
                 loop */) {
                *this -= -other;
            }
            return *this;
        }

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

        bigint &operator-=(const bigint &other) {
            if (sign == other.sign) {
                if (sign == 1 && *this >= other || sign
                     == -1 && *this <= other) {
                    for (int i = 0, carry = 0; i <
                         other.z.size() || carry; ++i) {
                        z[i] -= carry + (i <
                             other.z.size() ? other.z[i]
                             : 0);
                        carry = z[i] < 0;
                        if (carry)
                            z[i] += base;
                    }
                    trim();
                } else {
                    *this = other - *this;
                    this->sign = -this->sign;
                }
            } else {
                *this += -other;
            }
            return *this;
        }

        friend bigint

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

        bigint &operator*=(int v) {
            if (v < 0)
                sign = -sign, v = -v;
            for (int i = 0, carry = 0; i < z.size() ||
                 carry; ++i) {
                if (i == z.size())
                    z.push_back(0);
                long long cur = (long long) z[i] * v +
                     carry;
                carry = (int) (cur / base);
                z[i] = (int) (cur % base);
            }
            trim();
            return *this;
        }

        bigint operator*(int v) const {
            return bigint(*this) *= v;
        }

        friend pair<bigint, bigint> divmod(const
             bigint &a1, const bigint &b1) {
```

```
    int norm = base / (b1.z.back() + 1);
    bigint a = a1.abs() * norm;
    bigint b = b1.abs() * norm;
    bigint q, r;
    q.z.resize(a.z.size());

    for (int i = (int) a.z.size() - 1; i >= 0;
        i--) {
        r *= base;
        r += a.z[i];
        int s1 = b.z.size() < r.z.size() ?
            r.z[b.z.size()] : 0;
        int s2 = b.z.size() - 1 < r.z.size() ?
            r.z[b.z.size() - 1] : 0;
        int d = (int) (((long long) s1 * base +
            s2) / b.z.back());
        r -= b * d;
        while (r < 0)
            r += b, --d;
        q.z[i] = d;
    }

    q.sign = a1.sign * b1.sign;
    r.sign = a1.sign;
    q.trim();
    r.trim();
    return {q, r / norm};
}

friend bigint sqrt(const bigint &a1) {
    bigint a = a1;
    while (a.z.empty() || a.z.size() % 2 == 1)
        a.z.push_back(0);

    int n = a.z.size();

    int firstDigit = (int) ::sqrt((double)
        a.z[n - 1] * base + a.z[n - 2]);
    int norm = base / (firstDigit + 1);
    a *= norm;
    a *= norm;
    while (a.z.empty() || a.z.size() % 2 == 1)
        a.z.push_back(0);

    bigint r = (long long) a.z[n - 1] * base +
        a.z[n - 2];
    firstDigit = (int) ::sqrt((double) a.z[n -
        1] * base + a.z[n - 2]);
    int q = firstDigit;
    bigint res;

    for (int j = n / 2 - 1; j >= 0; j--) {
        for (;; --q) {
            bigint r1 = (r - (res * 2 * base +
                q) * q) * base * base +
                    (j > 0 ? (long long)
                        a.z[2 * j - 1] * base
                        + a.z[2 * j - 2] : 0);
            if (r1 >= 0) {
                r = r1;
                break;
            }
        }
        res *= base;
        res += q;

        if (j > 0) {
            int d1 = res.z.size() + 2 <
                r.z.size() ? r.z[res.z.size() +
                2] : 0;
            int d2 = res.z.size() + 1 <
                r.z.size() ? r.z[res.z.size() +
                1] : 0;
            int d3 = res.z.size() < r.z.size() ?
                r.z[res.z.size()] : 0;
            q = (int) (((long long) d1 * base *
                base + (long long) d2 * base +
                d3) / (firstDigit * 2));
        }
    }

    res.trim();
    return res / norm;
}

bigint operator/(const bigint &v) const {
    return divmod(*this, v).first;
}

bigint operator%(const bigint &v) const {
    return divmod(*this, v).second;
}

bigint &operator/=(int v) {
    if (v < 0)
        sign = -sign, v = -v;
    for (int i = (int) z.size() - 1, rem = 0; i
        >= 0; --i) {
        long long cur = z[i] + rem * (long
            long) base;
        z[i] = (int) (cur / v);
        rem = (int) (cur % v);
    }
    trim();
    return *this;
}

bigint operator/(int v) const {
    return bigint(*this) /= v;
}

int operator%(int v) const {
    if (v < 0)
        v = -v;
    int m = 0;
    for (int i = (int) z.size() - 1; i >= 0;
        --i)
        m = (int) ((z[i] + m * (long long)
            base) % v);
    return m * sign;
}

bigint &operator*=(const bigint &v) {
    *this = *this * v;
```

```
    return *this;
}

bigint &operator/=(const bigint &v) {
    *this = *this / v;
    return *this;
}

bool operator<(const bigint &v) const {
    if (sign != v.sign)
        return sign < v.sign;
    if (z.size() != v.z.size())
        return z.size() * sign < v.z.size() *
            v.sign;
    for (int i = (int) z.size() - 1; i >= 0;
        i--)
        if (z[i] != v.z[i])
            return z[i] * sign < v.z[i] * sign;
    return false;
}

bool operator>(const bigint &v) const {
    return v < *this;
}

bool operator<=(const bigint &v) const {
    return !(v < *this);
}

bool operator>=(const bigint &v) const {
    return !(*this < v);
}

bool operator==(const bigint &v) const {
    return !(*this < v) && !(v < *this);
}

bool operator!=(const bigint &v) const {
    return *this < v || v < *this;
}

void trim() {
    while (!z.empty() && z.back() == 0)
        z.pop_back();
    if (z.empty())
        sign = 1;
}

bool isZero() const {
    return z.empty();
}

friend bigint operator-(bigint v) {
    if (!v.z.empty())
        v.sign = -v.sign;
    return v;
}

bigint abs() const {
    return sign == 1 ? *this : -*this;
}

long long longValue() const {
    long long res = 0;
    for (int i = (int) z.size() - 1; i >= 0;
        i--)
        res = res * base + z[i];
    return res * sign;
}

friend bigint gcd(const bigint &a, const
    bigint &b) {
    return b.isZero() ? a : gcd(b, a % b);
}

friend bigint lcm(const bigint &a, const
    bigint &b) {
    return a / gcd(a, b) * b;
}

void read(const string &s) {
    sign = 1;
    z.clear();
    int pos = 0;
    while (pos < s.size() && (s[pos] == '-' ||
        s[pos] == '+')) {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }
    for (int i = (int) s.size() - 1; i >= pos;
        i -= base_digits) {
        int x = 0;
        for (int j = max(pos, i - base_digits +
            1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        z.push_back(x);
    }
    trim();
}

friend istream &operator>>(istream &stream,
    bigint &v) {
    string s;
    stream >> s;
    v.read(s);
    return stream;
}

friend ostream &operator<<(ostream &stream,
    const bigint &v) {
    if (v.sign == -1)
        stream << '-';
    stream << (v.z.empty() ? 0 : v.z.back());
    for (int i = (int) v.z.size() - 2; i >= 0;
        --i)
        stream << setw(base_digits) <<
            setfill('0') << v.z[i];
    return stream;
}

static vector<int> convert_base(const
    vector<int> &a, int old_digits, int
    new_digits) {
```

```
    vector<long long> p(max(old_digits,
        new_digits) + 1);
    p[0] = 1;
    for (int i = 1; i < p.size(); i++)
        p[i] = p[i - 1] * 10;
    vector<int> res;
    long long cur = 0;
    int cur_digits = 0;
    for (int v : a) {
        cur += v * p[cur_digits];
        cur_digits += old_digits;
        while (cur_digits >= new_digits) {
            res.push_back(int(cur %
                p[new_digits]));
            cur /= p[new_digits];
            cur_digits -= new_digits;
        }
    }
    res.push_back((int) cur);
    while (!res.empty() && res.back() == 0)
        res.pop_back();
    return res;
}

typedef vector<long long> vll;

static vll karatsubaMultiply(const vll &a,
    const vll &b) {
    int n = a.size();
    vll res(n + n);
    if (n <= 32) {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                res[i + j] += a[i] * b[j];
        return res;
    }

    int k = n >> 1;
    vll a1(a.begin(), a.begin() + k);
    vll a2(a.begin() + k, a.end());
    vll b1(b.begin(), b.begin() + k);
    vll b2(b.begin() + k, b.end());

    vll a1b1 = karatsubaMultiply(a1, b1);
    vll a2b2 = karatsubaMultiply(a2, b2);

    for (int i = 0; i < k; i++)
        a2[i] += a1[i];
    for (int i = 0; i < k; i++)
        b2[i] += b1[i];

    vll r = karatsubaMultiply(a2, b2);
    for (int i = 0; i < a1b1.size(); i++)
        r[i] -= a1b1[i];
    for (int i = 0; i < a2b2.size(); i++)
        r[i] -= a2b2[i];

    for (int i = 0; i < r.size(); i++)
        res[i + k] += r[i];
    for (int i = 0; i < a1b1.size(); i++)
        res[i] += a1b1[i];
    for (int i = 0; i < a2b2.size(); i++)
        res[i + n] += a2b2[i];
    return res;
}

bigint operator*(const bigint &v) const {
    vector<int> a6 = convert_base(this->z,
        base_digits, 6);
    vector<int> b6 = convert_base(v.z,
        base_digits, 6);
    vll a(a6.begin(), a6.end());
    vll b(b6.begin(), b6.end());
    while (a.size() < b.size())
        a.push_back(0);
    while (b.size() < a.size())
        b.push_back(0);
    while (a.size() & (a.size() - 1))
        a.push_back(0), b.push_back(0);
    vll c = karatsubaMultiply(a, b);
    bigint res;
    res.sign = sign * v.sign;
    for (int i = 0, carry = 0; i < c.size();
        i++) {
        long long cur = c[i] + carry;
        res.z.push_back((int) (cur % 1000000));
        carry = (int) (cur / 1000000);
    }
    res.z = convert_base(res.z, 6, base_digits);
    res.trim();
    return res;
}

};

bigint random_bigint(int n) {
    string s;
    for (int i = 0; i < n; i++) {
        s += rand() % 10 + '0';
    }
    return bigint(s);
}

// random tests
void bigintTest() {
    bigint x = bigint("120");
    bigint y = bigint("5");
    cout << x / y << endl;

    for (int i = 0; i < 1000; i++) {
        int n = rand() % 100 + 1;
        bigint a = random_bigint(n);
        bigint res = sqrt(a);
        bigint xx = res * res;
        bigint yy = (res + 1) * (res + 1);

        if (xx > a || yy <= a) {
            cout << i << endl;
            cout << a << " " << res << endl;
            break;
        }

        int m = rand() % n + 1;
        bigint b = random_bigint(m) + 1;
```

```
            res = a / b;
            xx = res * b;
            yy = b * (res + 1);

            if (xx > a || yy <= a) {
                cout << i << endl;
                cout << a << " " << b << " " << res <<
                    endl;
                break;
            }
        }

        bigint a = random_bigint(10000);
        bigint b = random_bigint(2000);
        clock_t start = clock();
        bigint c = a / b;
        printf("time=%.3lfsec\n", (clock() - start) *
            1. / CLOCKS_PER_SEC);
    }
}

using namespace bigint;
```

10.4.2 Expression Parser

```
/**
 * Description: Evaluates mod expression with
     parentheses, or returns -1 if it is invalid
 * Source: Own
 * Verification: IPSC 2018 I1
 */

namespace exprParse {
    string expr;
    int ind; // reset to 0 every time

    int eval(char c, int x, int y) {
        switch(c) {
            case '+': return (x+y)%MOD;
            case '-': return (x-y+MOD)%MOD;
            case '*': return (ll)x*y%MOD;
            default: exit(5);
        }
    }

    int pri(char c) {
        switch (c) {
            case '+': return 0;
            case '-': return 0;
            case '*': return 1;
            default: exit(5);
        }
    }

    int process(vi& num, vector<char>& op) {
        if (sz(num) < 2 || sz(op) < 1) return -1;
        int y = num.back(); num.pop_back();
        int x = num.back(); num.pop_back();
        char c = op.back(); op.pop_back();
        num.pb(eval(c,x,y));
        return 0;
    }

    int finish(int& state, vi& num, vector<char>& op) {
        if (state != 0) return -1;
        while (sz(op)) {
            int t = process(num,op);
            if (t == -1) return -1;
        }
        if (sz(num) != 1) return -1;
        return num[0];
    }

    int eval(int cur = 0) {
        vi num;
        vector<char> op;

        int state = -1;
        while (ind < sz(expr)) {
            if (expr[ind] == ')') {
                ind ++;
                if (cur == 0) return -1;
                return finish(state,num,op);
            } else if (expr[ind] == '(') {
                ind ++;
                num.pb(eval(1)); if (num.back() == -1)
                    return -1;
                if (state == 0) return -1;
                state = 0;
            } else if (isdigit(expr[ind])) {
                int val = 0;
                while (ind < sz(expr) &&
                    isdigit(expr[ind])) {
                    val = (10LL*val+(expr[ind]-'0')) %
                        MOD;
                    ind ++;
                }
                num.pb(val);
                if (state == 0) return -1;
                state = 0;
            } else {
                while (sz(op) && pri(op.back()) >=
                    pri(expr[ind])) {
                    int t = process(num,op);
                    if (t == -1) return -1;
                }
                op.pb(expr[ind]);
                if (state != 0) return -1;
                state = 1;
                ind ++;
            }
        }

        if (cur == 1) return -1; // parentheses don't
            match up
        if (ind != sz(expr)) return -1;
        return finish(state,num,op);
    }
}

using namespace exprParse;
```

10.4.3 Fraction

```
/**
 * Description: Operations with fractions
 * Source: https://martin-thoma.com/fractions-in-cpp/
 * Verification: ? TopCoder
     MinimizeAbsoluteDifferenceDiv1
 */

struct frac {
    ll n,d;
    frac() { n = 0, d = 1; }
    frac(ll _n, ll _d) {
        n = _n, d = _d;
        ll g = __gcd(n,d);
        n /= g, d /= g;
        if (d < 0) n *= -1, d *= -1;
    }

    friend frac abs(frac F) { return
        frac(abs(F.n),F.d); }

    friend bool operator<(const frac& l, const frac&
        r) { return l.n*r.d < r.n*l.d; }
    friend bool operator==(const frac& l, const frac&
        r) { return l.n == r.n && l.d == r.d; }
    friend bool operator!=(const frac& l, const frac&
        r) { return !(l == r); }

    friend frac operator+(const frac& l, const frac&
        r) { return frac(l.n*r.d+r.n*l.d,l.d*r.d); }
    friend frac operator-(const frac& l, const frac&
        r) { return frac(l.n*r.d-r.n*l.d,l.d*r.d); }
    friend frac operator*(const frac& l, const frac&
        r) { return frac(l.n*r.n,l.d*r.d); }
    friend frac operator*(const frac& l, int r) {
        return l*frac(r,1); }
    friend frac operator*(int r, const frac& l) {
        return l*r; }
    friend frac operator/(const frac& l, const frac&
        r) { return l*frac(r.d,r.n); }
    friend frac operator/(const frac& l, const int& r)
        { return l/frac(r,1); }
    friend frac operator/(const int& l, const frac& r)
        { return frac(l,1)/r; }

    friend frac operator+=(frac& l, const frac& r) {
        return l = l+r; }
    friend frac operator-=(frac& l, const frac& r) {
        return l = l-r; }
    friend template<class T> frac operator*=(frac& l,
        const T& r) { return l = l*r; }
    friend template<class T> frac operator/=(frac& l,
        const T& r) { return l = l/r; }

    friend ostream& operator<<(ostream &strm, const
        frac &a) {
        strm << a.n;
        if (a.d != 1) strm << "/" << a.d;
        return strm;
    }
};
```

10.4.4 Matrix

```
/**
 * Description: 2D matrix operations
 * Source: KACTL
 * Verification: ? https://dmoj.ca/problem/si17c1p5,
     SPOJ MIFF
 */

template<class T> struct Mat {
    T** d;
    int a, b;

    Mat() { a = b = 0; }

    Mat(int _a, int _b) {
        a = _a, b = _b;
        d = new T*[a];
        FOR(i,a) {
            d[i] = new T[b];
            FOR(j,b) d[i][j] = 0;
        }
    }

    Mat (const vector<vector<T>>& v) :
        Mat(sz(v),sz(v[0])) {
        FOR(i,a) FOR(j,b) d[i][j] = v[i][j];
    }

    operator vector<vector<T>> () {
        auto ret = vector<vector<T>>(a,vector<T>(b));
        FOR(i,a) FOR(j,b) ret[i][j] = d[i][j];
        return ret;
    }

    Mat operator+(const Mat& m) {
        Mat r(a,b);
        FOR(i,a) FOR(j,b) r.d[i][j] =
            d[i][j]+m.d[i][j];
        return r;
    }
    Mat operator-(const Mat& m) {
        Mat r(a,b);
        FOR(i,a) FOR(j,b) r.d[i][j] =
            d[i][j]-m.d[i][j];
        return r;
    }
    Mat operator*(const Mat& m) {
        Mat r(a,m.b);
        FOR(i,a) FOR(j,b) FOR(k,m.b) r.d[i][k] +=
            d[i][j]*m.d[j][k];
        return r;
    }

    friend Mat exp(const Mat& m, ll p) {
        assert(a == b);
        Mat r(a,a), base(*this);
        FOR(i,a) r.d[i][i] = 1;
```

```
        for (; p; p /= 2, base *= base) if (p&1) r *=
            base;
        return r;
    }
};

namespace matInv {
    template<class T> void elim(Mat<T>& m, int col,
        int a, int b) { // eliminate row a from row b
        auto x = m.d[b][col];
        FOR(i,col,m.b) m.d[b][i] -= x*m.d[a][i];
    }

    template<class T> T gauss(Mat<T>& m) { //
        determinant of 1000x1000 Matrix in ~1s
        T prod = 1; int nex = 0;

        FOR(i,m.a) {
            int row = -1;
            FOR(j,nex,m.a) if (m.d[j][i] != 0) { row =
                j; break; }
            if (row == -1) { prod = 0; continue; }
            if (row != nex) prod *= -1,
                swap(m.d[row],m.d[nex]);

            prod *= m.d[nex][i];
            auto x = inv(m.d[nex][i]);
            FOR(k,i,m.b) m.d[nex][k] *= x;

            FOR(k,m.a) if (k != nex) elim(m,i,nex,k);
            nex ++;
        }

        return prod;
    }

    int numSpan(Mat<int> m) { // Kirchhoff's theorem
        Mat<int> res(m.a-1,m.a-1);
        FOR(i,m.a) FOR(j,i+1,m.a) {
            if (i) {
                AD(res.d[i-1][i-1],m.d[i][j]);
                SUB(res.d[i-1][j-1],m.d[i][j]);
                SUB(res.d[j-1][i-1],m.d[i][j]);
            }
            AD(res.d[j-1][j-1],m.d[i][j]);
        }
        return gauss(res);
    }

    template<class T> Mat<T> inv(Mat<T> m) {
        Mat<T> x(m.a,2*m.a);
        FOR(i,m.a) FOR(j,m.a) x.d[i][j] = m.d[i][j];
        FOR(i,m.a) x.d[i][i+m.a] = 1;

        if (gauss(x) == 0) return Mat<T>(0,0);

        Mat<T> r(m.a,m.a);
        FOR(i,m.a) FOR(j,m.a) r.d[i][j] =
            x.d[i][j+m.a];
        return r;
    }
}

using namespace matInv;
```

10.4.5 Pair Operators

```
/**
 * Description: modular arithmetic with pairs
       * use for hashing
 * Source: Own
 * Verification: see hashing
 */

namespace pairOp {
    template<class A, class B> A operator+=(A& l,
        const B& r) { return l = l+r; }
    template<class A, class B> A operator-=(A& l,
        const B& r) { return l = l-r; }
    template<class A, class B> A operator*=(A& l,
        const B& r) { return l = l*r; }

        template<class A, class B> pair<A,B>
            operator+(const pair<A,B>& l, const
            pair<A,B>& r) {
                return {l.f+r.f,l.s+r.s};
        }
        template<class A, class B> pair<A,B>
            operator-(const pair<A,B>& l, const
            pair<A,B>& r) {
                return {l.f-r.f,l.s-r.s};
        }
        template<class A, class B> pair<A,B>
            operator*(const pair<A,B>& l, const
            pair<A,B>& r) {
                return {l.f*r.f,l.s*r.s};
        }
        template<class A, class B, class C> pair<A,B>
            operator*(const pair<A,B>& l, const C& r) {
                return {l.f*r,l.s*r};
        }
}

using namespace pairOp;
```

10.4.6 Vector Operators

```
/**
 * Description: modular arithmetic with vectors
    * use for NTT
 * Source: Own
 * Verification: ?
 */

namespace vecOp {
    template<class T> vector<T> rev(vector<T> v) {
        reverse(all(v)); return v; }
    template<class T> vector<T> operator+(vector<T> l,
        const vector<T>& r) {
```

```
        l.resz(max(sz(l),sz(r))); FOR(i,sz(r)) l[i] +=
            r[i];
        return l;
    }
    template<class T> vector<T> operator-(vector<T> l,
        const vector<T>& r) {
        l.resz(max(sz(l),sz(r))); FOR(i,sz(r)) l[i] -=
            r[i];
        return l;
    }
    template<class T> vector<T> operator*(const
        vector<T>& l, const vector<T>& r) {
        if (min(sz(l),sz(r)) == 0) return {};
        vector<T> x(sz(l)+sz(r)-1); FOR(i,sz(l))
            FOR(j,sz(r)) x[i+j] += l[i]*r[j];
        return x;
    }
    template<class T> vector<T> operator*(const
        vector<T>& l, const int& r) {
        vector<T> v = {r}; return l*v;
    }

    template<class T> vector<T> rem(vector<T>
        a,vector<T> b) {
        while (sz(b) && b.back() == 0) b.pop_back();
        assert(sz(b)); b *= inv(b.back());
        while (sz(a) >= sz(b)) {
            auto k = a.back();
            FOR(i,sz(b)) a[sz(a)-sz(b)+i] -= k*b[i];
            while (sz(a) && a.back() == 0) a.pop_back();
        }
        return a;
    }
    template<class T> vector<T>
        interpolate(vector<pair<T,T>> v) {
        vector<T> ret;
        FOR(i,sz(v)) {
            vector<T> prod = {1};
            T todiv = 1;
            FOR(j,sz(v)) if (i != j) {
                todiv *= v[i].f-v[j].f;
                vector<T> tmp = {-v[j].f,1}; prod *=
                    tmp;
            }
            ret += prod*(v[i].s/todiv);
        }
        return ret;
    }
}
```

10.5 Polynomials (6)

10.5.1 FFT

```
/**
 * Description: multiply two polynomials
 * Source: https://cp-algorithms.com/algebra/fft.html
 * Verification: ? SPOJ polymul, CSA manhattan, CF
     Perfect Encoding
 */
typedef complex<double> cd;

namespace FFT {
    int get(int s) { return s > 1 ? 32 -
        __builtin_clz(s - 1) : 0; }

    vcd roots;

    void fft(vcd& a) {
        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)
            for (int i = 0; i < n; i += len)
                FOR(j,len/2) {
                    cd u = a[i+j], v = a[i+j+len/2] *
                        roots[n/len*j];
                    a[i+j] = u+v, a[i+j+len/2] = u-v;
                } // for XOR, let v = a[i+j+len/2]
                    instead
    }

    template<class T> T brute(const T& a, const T& b) {
        T c(sz(a)+sz(b)-1);
        FOR(i,sz(a)) FOR(j,sz(b)) c[i+j] += a[i]*b[j];
        return c;
    }

    vcd conv(vcd a, vcd b) {
        int s = sz(a)+sz(b)-1, L = get(s), n = 1<<L;
        if (s <= 0) return {};
        if (s <= 200) return brute(a,b);

        roots.resz(n); ld ang = 2*PI/n;
        FOR(i,n) roots[i] = cd(cos(ang*i),sin(ang*i));
            // is there a way to do this more quickly?

        a.resz(n), fft(a); b.resz(n), fft(b);
        FOR(i,n) a[i] *= b[i];

        reverse(beg(roots)+1,en(roots));
        fft(a); trav(x,a) x /= n;

        a.resz(s); return a;
    }

    vl conv(const vl& a, const vl& b) {
        vcd X = conv(vcd(all(a)),vcd(all(b)));
        vl x(sz(X)); FOR(i,sz(X)) x[i] =
            round(X[i].real());
        return x;
    } // ~0.55s when sz(a)=sz(b)=1<<19

    vl conv(const vl& a, const vl& b, ll mod) { //
        http://codeforces.com/contest/960/submission/37085144
        int s = sz(a)+sz(b)-1, L = get(s), n = 1<<L;
```

```
        vcd v1(n), v2(n), r1(n), r2(n);
        FOR(i,sz(a)) v1[i] = cd(a[i] >> 15, a[i] &
            32767);
        FOR(i,sz(b)) v2[i] = cd(b[i] >> 15, b[i] &
            32767);

        roots.resz(n); ld ang = 2*PI/n;
        FOR(i,n) roots[i] = cd(cos(ang*i),sin(ang*i));
            // is there a way to do this more quickly?

        fft(v1), fft(v2);

        FOR(i,n) {
            int j = (i ? (n - i) : i);
            cd ans1 = (v1[i] + conj(v1[j])) * cd(0.5,
                0);
            cd ans2 = (v1[i] - conj(v1[j])) * cd(0,
                -0.5);
            cd ans3 = (v2[i] + conj(v2[j])) * cd(0.5,
                0);
            cd ans4 = (v2[i] - conj(v2[j])) * cd(0,
                -0.5);
            r1[i] = (ans1 * ans3) + (ans1 * ans4) *
                cd(0, 1);
            r2[i] = (ans2 * ans3) + (ans2 * ans4) *
                cd(0, 1);
        }

        reverse(beg(roots)+1,en(roots));
        fft(r1), fft(r2); FOR(i,n) r1[i] /= n, r2[i]
            /= n;

        vl ret(n);
        FOR(i,n) {
            ll av = (ll)round(r1[i].real());
            ll bv = (ll)round(r1[i].imag()) +
                (ll)round(r2[i].real());
            ll cv = (ll)round(r2[i].imag());
            av %= mod, bv %= mod, cv %= mod;
            ret[i] = (av << 30) + (bv << 15) + cv;
            ret[i] = (ret[i]%mod+mod)%mod;
        }

        ret.resz(s); return ret;
    } // ~0.8s when sz(a)=sz(b)=1<<19
}

using namespace FFT;
```

10.5.2 Linear Recurrence

```
/**
* Description: Berlekamp-Massey Algo
* Source: http://codeforces.com/blog/entry/61306
* Verification:
    http://codeforces.com/contest/506/problem/E
*/

using namespace vecOp;

struct linRec {
    vector<vi> seq;
    vi x, fail, delta, des;

    linRec (vi _x) {
        x = _x; seq.pb({}); int best = 0;

        FOR(i,sz(x)) {
            delta.pb(mul(-1,x[i]));
            FOR(j,sz(seq.back()))
                AD(delta[i],mul(x[i-j-1],seq.back()[j]));
            if (delta[i] == 0) continue;

            fail.pb(i); if (sz(seq) == 1) {
                seq.pb(vi(i+1)); continue; }

            int k =
                mul(mul(-1,delta[i]),inv(delta[fail[best]]));
            vi cur(i-fail[best]-1); cur.pb(mul(-1,k));
            for (auto a: seq[best]) cur.pb(mul(a,k));

            cur += seq.back();
            if (i-fail[best]+sz(seq[best]) >=
                sz(seq.back())) best = sz(seq)-1;
               // take fail vector with smallest size

            seq.pb(cur);
        }

        FORd(i,sz(seq.back()))
            des.pb(mul(-1,seq.back()[i]));
        des.pb(1);
    }

    vi getPo(int n) {
        if (n == 0) return {1};
        vi x = getPo(n/2); x = rem(x*x,des);
        if (n&1) {
            vi v = {0,1};
            x = rem(x*v,des);
        }
        return x;
    }

    int get(int n) {
        vi t = getPo(n);
        int ANS = 0;
        FOR(i,sz(t)) AD(ANS,mul(t[i],x[i]));
        return ANS;
    }
};
```

10.5.3 NTT Extended

```
/**
 * Description: Evaluate degree n polynomial at n
     points in O(nlog^2n) time
 * Source: CF,
     http://people.csail.mit.edu/madhu/ST12/scribe/lect06.pdf
```

```
 * Verification: ?
     https://codeforces.com/contest/438/problem/E
 */

using namespace vecOp;
using namespace NTT;

namespace NTTextend {
    vi inv(vi v, int p) { // compute inverse of v mod
        x^p, where v[0] = 1
        v.resize(p); if (p == 1) return {1};
        assert(v[0] == 1);
        if (p&1) { // naive
            int cur = 0; auto V = inv(v,p-1);
            FOR(i,p-1) if (p-1-i < sz(v))
                SUB(cur,mul(V[i],v[p-1-i]));
            V.pb(cur); return V;
        }
        vi a = inv(v,p/2);
        vi h0 = vi(v.begin(),v.begin()+p/2);
        vi h1 = vi(v.begin()+p/2,v.end());
        vi c = conv(a,h0); c =
            vi(c.begin()+p/2,c.end());
        assert(c[0] == 1);

        vi b = conv(-1*a,conv(h1,a)+c); b.resz(p/2);
        a.insert(a.end(),all(b)); return a;
    }

    pair<vi,vi> divi(vi f, vi g) { // f = q*g+r
        if (sz(f) < sz(g)) return {{},f};
        vi q = conv(inv(rev(g),sz(f)-sz(g)+1),rev(f));
        q.resz(sz(f)-sz(g)+1); q = rev(q);
        vi r = f-conv(q,g); r.resize(sz(g)-1);
        return {q,r};
    }

    vi sqrt(vi v, int p) { // S*S = v mod x^p, p is
        power of 2
        v.resize(p);
        if (p == 1) return {1};
        vi S = sqrt(v,p/2);
        vi ans = S+conv(v,inv(S,p));
        while (sz(ans) > p) ans.pop_back();
        ans *= inv(2);
        return ans;
    }

    vi poly[1<<19];

    void precompute(int ind, vi vals) {
        if (sz(vals) == 1) { poly[ind] =
            {sub(MOD,vals[0]),1}; return; }
        int m = sz(vals)/2;
        precompute(2*ind,vi(vals.begin(),vals.begin()+m));
        precompute(2*ind+1,vi(vals.begin()+m,vals.end()));
        poly[ind] = conv(poly[2*ind],poly[2*ind+1]);
    }

    void eval(int ind, vi p, vi& ans) { // evaluate p
        over all elements of val
        p = divi(p,poly[ind]).s;
        if (sz(poly[ind]) == 2) { ans.pb(p[0]);
            return; }
        eval(2*ind,p,ans); eval(2*ind+1,p,ans);
    }
}

using namespace NTTextend;
```

10.5.4 NTT

```
/**
 * Description: Use if you are working with
     non-negative integers
 * Source: KACTL,
     https://cp-algorithms.com/algebra/fft.html
 * Verification: ?
     http://codeforces.com/contest/632/problem/E
 */

// dependency: mi

const int MOD = (119 << 23) + 1, root = 3; // =
    998244353
// For p < 2^30 there is also e.g. (5 << 25, 3), (7 <<
    26, 3),
// (479 << 21, 3) and (483 << 21, 5). The last two are
    > 10^9.

namespace NTT {
    int size(int s) { return s > 1 ? 32 -
        __builtin_clz(s - 1) : 0; }

    vmi roots;

    void ntt(vmi& a) {
        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)
            for (int i = 0; i < n; i += len)
                FOR(j,len/2) {
                    auto u = a[i+j], v =
                        a[i+j+len/2]*roots[n/len*j];
                    a[i+j] = u+v, a[i+j+len/2] = u-v;
                }
    }

    vmi brute(const vmi& a, const vmi& b) {
        vmi c(sz(a)+sz(b)-1);
        FOR(i,sz(a)) FOR(j,sz(b)) c[i+j] += a[i]*b[j];
        return c;
    }

    vmi conv(vmi a, vmi b) {
        int s = sz(a)+sz(b)-1, L = size(s), n = 1<<L;
```

```
        if (s <= 0) return {};
        if (s <= 200) return brute(a,b);

        roots.resz(n); int r = exp(root,(MOD-1)/n);
        roots[0] = 1; FOR(i,1,n) roots[i] =
            roots[i-1]*r;

        a.resz(n), ntt(a); b.resz(n), ntt(b);
        FOR(i,n) a[i] *= b[i];

        reverse(beg(roots)+1,en(roots));
        ntt(a); int in = inv(n); trav(x,a) x *= in;

        a.resz(s); return a;
    } // ~0.22s when sz(a) = sz(b) = 1<<19

    vi conv(vi a, vi b) { return
        toVi(conv(toVmi(a),toVmi(b))); }
}

using namespace NTT;
```

11 Graphs Hard (4)

11.1 SCC

11.1.1 2SAT

```
/*
* Description: Solves 2SAT
* Source: ?
* Verification: https://www.spoj.com/problems/BUGLIFE/
* Also useful: at most one
    http://codeforces.com/contest/1007
    /submission/40284510
*/

// struct scc

template<int SZ> struct twosat {
    scc<2*SZ> S;
    int N;

    void OR(int x, int y) { S.addEdge(x^1,y);
        S.addEdge(y^1,x); }

    int tmp[2*SZ];
    bitset<SZ> ans;

    bool solve() {
        S.N = 2*N; S.genSCC();
        for (int i = 0; i < 2*N; i += 2) if (S.comp[i]
            == S.comp[i^1]) return 0;
        reverse(all(S.allComp));
        for (int i: S.allComp) if (tmp[i] == 0)
            tmp[i] = 1, tmp[S.comp[i^1]] = -1;
            FOR(i,N) if (tmp[S.comp[2*i]] == 1) ans[i]
                = 1;
        return 1;
    }
};
```

11.1.2 Kosaraju

```
/**
* Source: Wikipedia
* Description: generates SCC in topological order
* Verification: POI 8 peaceful commission
*/

template<int SZ> struct scc {
    vi adj[SZ], radj[SZ], todo, allComp;
    int N, comp[SZ];
    bitset<SZ> visit;

    void dfs(int v) {
        visit[v] = 1;
        for (int w: adj[v]) if (!visit[w]) dfs(w);
        todo.pb(v);
    }

    void dfs2(int v, int val) {
        comp[v] = val;
        for (int w: radj[v]) if (comp[w] == -1)
            dfs2(w,val);
    }

    void addEdge(int a, int b) { adj[a].pb(b),
        radj[b].pb(a); }

    void genSCC() {
        FOR(i,N) comp[i] = -1, visit[i] = 0;
        FOR(i,N) if (!visit[i]) dfs(i);
        reverse(all(todo)); // toposort
        for (int i: todo) if (comp[i] == -1)
            dfs2(i,i), allComp.pb(i);
    }
};
```

11.2 Flows

11.2.1 Dinic (5)

```
/**
* Description: Faster Flow, Bipartite Matching
* Source: GeeksForGeeks
* Verification: Problem Fashion (RMI 2017 Day 1)
    * https://pastebin.com/VJxTvEg1
*/

template<int SZ> struct Dinic {
    struct Edge {
        int v;
        ll flow, cap;
        int rev;
    };
```

```
    vector<Edge> adj[SZ];

    void addEdge(int u, int v, ll cap) {
        Edge a{v, 0, cap, sz(adj[v])};
        Edge b{u, 0, 0, sz(adj[u])};
        adj[u].pb(a), adj[v].pb(b);
    }

    int level[SZ], st[SZ];

    bool bfs(int s, int t) {
        FOR(i,SZ) level[i] = -1, st[i] = 0;
        level[s] = 0;

        queue<int> q; q.push(s);
        while (sz(q)) {
            int u = q.front(); q.pop();
            for (auto e: adj[u])
                if (level[e.v] < 0 && e.flow < e.cap) {
                    level[e.v] = level[u] + 1;
                    q.push(e.v);
                }
        }

        return level[t] >= 0;
    }

    ll sendFlow(int s, int t, ll flow) {
        if (s == t) return flow;

        for ( ; st[s] < sz(adj[s]); st[s] ++) {
            Edge &e = adj[s][st[s]];

            if (level[e.v] != level[s]+1 || e.flow ==
                e.cap) continue;
            ll temp_flow = sendFlow(e.v, t, min(flow,
                e.cap - e.flow));

            if (temp_flow > 0) {
                e.flow += temp_flow;
                adj[e.v][e.rev].flow -= temp_flow;
                return temp_flow;
            }
        }

        return 0;
    }

    ll maxFlow(int s, int t) {
        if (s == t) return -1;
        ll total = 0;
        while (bfs(s, t)) while (ll flow = sendFlow(s,
            t, INT_MAX)) total += flow;
        return total;
    }
};
```

11.2.2 Push-Relabel (5)

```
/**
 * Source: http://codeforces.com/blog/entry/14378
 * Verification: SPOJ fastflow
 */

struct Edge {
    int v;
    ll flow, C;
    int rev;
};

template <int SZ> struct PushRelabel {
    vector<Edge> adj[SZ];
    ll excess[SZ];
    int dist[SZ], count[SZ+1], b = 0;
    bool active[SZ];
    vi B[SZ];

    void addEdge(int u, int v, ll C) {
        Edge a{v, 0, C, sz(adj[v])};
        Edge b{u, 0, 0, sz(adj[u])};
        adj[u].pb(a), adj[v].pb(b);
    }

    void enqueue (int v) {
        if (!active[v] && excess[v] > 0 && dist[v] <
            SZ) {
            active[v] = 1;
            B[dist[v]].pb(v);
            b = max(b, dist[v]);
        }
    }

    void push (int v, Edge &e) {
        ll amt = min(excess[v], e.C-e.flow);
        if (dist[v] == dist[e.v]+1 && amt > 0) {
            e.flow += amt, adj[e.v][e.rev].flow -= amt;
            excess[e.v] += amt, excess[v] -= amt;
            enqueue(e.v);
        }
    }

    void gap (int k) {
        FOR(v,SZ) if (dist[v] >= k) {
            count[dist[v]] --;
            dist[v] = SZ;
            count[dist[v]] ++;
            enqueue(v);
        }
    }

    void relabel (int v) {
        count[dist[v]] --; dist[v] = SZ;
        for (auto e: adj[v]) if (e.C > e.flow) dist[v]
            = min(dist[v], dist[e.v] + 1);
        count[dist[v]] ++;
        enqueue(v);
    }

    void discharge(int v) {
        for (auto &e: adj[v]) {
            if (excess[v] > 0) push(v,e);
            else break;
```

```
        }
        if (excess[v] > 0) {
            if (count[dist[v]] == 1) gap(dist[v]);
            else relabel(v);
        }
    }

    ll maxFlow (int s, int t) {
        for (auto &e: adj[s]) excess[s] += e.C;

        count[0] = SZ;
        enqueue(s); active[t] = 1;

        while (b >= 0) {
            if (sz(B[b])) {
                int v = B[b].back(); B[b].pop_back();
                active[v] = 0; discharge(v);
            } else b--;
        }
        return excess[t];
    }
};
```

11.2.3 MinCostFlow (6)

```
/**
* Description:
* Source: GeeksForGeeks
* Verification: CodeForces?
*/

struct Edge {
    int v, flow, C, rev, cost;
};

template<int SZ> struct mcf {
    pi pre[SZ];
    int cost[SZ], num[SZ], SC, SNC;
    ll flo, ans, ccost;
    vector<Edge> adj[SZ];

    void addEdge(int u, int v, int C, int cost) {
        Edge a{v, 0, C, sz(adj[v]), cost};
        Edge b{u, 0, 0, sz(adj[u]), -cost};
        adj[u].pb(a), adj[v].pb(b);
    }

    void reweight() {
        FOR(i,SZ) {
            for (auto& p: adj[i]) p.cost +=
                cost[i]-cost[p.v];
        }
    }

    bool spfa() {
        FOR(i,SZ) cost[i] = MOD, num[i] = 0;
        cost[SC] = 0, num[SC] = MOD;
        priority_queue<pi,vpi,greater<pi>> todo;
            todo.push({0,SC});
        while (todo.size()) {
            pi x = todo.top(); todo.pop();
            if (x.f > cost[x.s]) continue;
            for (auto a: adj[x.s]) if (x.f+a.cost <
                cost[a.v] && a.flow < a.C) {
                pre[a.v] = {x.s,a.rev};
                cost[a.v] = x.f+a.cost;
                num[a.v] = min(a.C-a.flow,num[x.s]);
                todo.push({cost[a.v],a.v});
            }
        }

        ccost += cost[SNC];
        return num[SNC] > 0;
    }

    void backtrack() {
        flo += num[SNC], ans += (ll)num[SNC]*ccost;
        for (int x = SNC; x != SC; x = pre[x].f) {
            adj[x][pre[x].s].flow -= num[SNC];
            int t = adj[x][pre[x].s].rev;
            adj[pre[x].f][t].flow += num[SNC];
        }
    }

    pi mincostflow(int sc, int snc) {
        SC = sc, SNC = snc;
        flo = ans = ccost = 0;

        spfa();
        while (1) {
            reweight();
            if (!spfa()) return {flo,ans};
            backtrack();
        }
    }
};
```

11.3 Tarjan BCC

```
/**
* Description:
* Source: GeeksForGeeks (corrected)
* Verification: USACO December 2017, Push a Box
    * https://pastebin.com/yUWuzTH8
*/

template<int SZ> struct BCC {
    int N;
    vi adj[SZ];
    vector<vpi> fin;

    void addEdge(int u, int v) { adj[u].pb(v),
        adj[v].pb(u); }

    int ti = 0, disc[SZ], low[SZ], comp[SZ], par[SZ];
    vpi st;

    void BCCutil(int u, bool root = 0) {
        disc[u] = low[u] = ti++;
```

```
        int child = 0;

        for (int i: adj[u]) if (i != par[u])
            if (disc[i] == -1) {
                child ++; par[i] = u;
                st.pb({u,i});
                BCCutil(i);
                low[u] = min(low[u],low[i]);

                if ((root && child > 1) || (!root &&
                    disc[u] <= low[i])) { //
                    articulation point!
                    vpi tmp;
                    while (st.back() != mp(u,i))
                        tmp.pb(st.back()),
                        st.pop_back();
                    tmp.pb(st.back()), st.pop_back();
                    fin.pb(tmp);
                }
            } else if (disc[i] < disc[u]) {
                low[u] = min(low[u],disc[i]);
                st.pb({u,i});
            }
    }

    void bcc(int _N) {
        N = _N;
        FOR(i,1,N+1) par[i] = disc[i] = low[i] = -1;
        FOR(i,1,N+1) if (disc[i] == -1) {
            BCCutil(i,1);
            if (sz(st)) fin.pb(st);
            st.clear();
        }
    }
};
```

11.4 Euler Tour (6)

```
/**
 * Description: O(N+M) euler tour for both directed
     and undirected graphs
 * Source: USACO Training
 * Verification:
    * directed:
        https://open.kattis.com/problems/eulerianpath
    * undirected: USACO Training 3.3, Riding the Fences
 */

struct Euler {
    vpi adj[MX], circuit;
    int N, M, nex, out[MX], in[MX], deg[MX];
    bool used[MX], bad;

    void clr() {
        FOR(i,N) adj[i].clear();
        circuit.clear();
        nex = 0;
        FOR(i,N) out[i] = in[i] = deg[i];
        FOR(i,M) used[i] = 0;
        bad = 0;
    }

    void find_circuit(int pre, int cur) {
        while (sz(adj[cur])) {
            pi x = adj[cur].back(); adj[cur].pop_back();
            if (used[x.s]) continue;
            used[x.s] = 1;
            find_circuit(cur,x.f);
        }
        if (sz(circuit) && circuit.back().f != cur)
            bad = 1;
        circuit.pb({pre,cur});
    }

    void addEdge(int a, int b, bool directed) {
        if (directed) {
            adj[a].pb({b,nex});
            out[a] ++, in[b] ++; nex ++;
        } else {
            adj[a].pb({b,nex}), adj[b].pb({a,nex});
            deg[a] ++, deg[b] ++; nex ++;
        }
    }

    vi solve(bool directed) { // edges only involve
        vertices from 0 to N-1
        int start = 0;
        FOR(i,N) if (deg[i] || in[i] || out[i]) start
            = i;

        if (directed) {
            FOR(i,N) if (out[i]-in[i] == 1) start = i;
        } else {
            FOR(i,N) if (deg[i] % 2 == 1) start = i;
        }

        find_circuit(-1,start);
        if (sz(circuit) != M+1 || bad) return {};
        vi ans; FORd(i,sz(circuit))
            ans.pb(circuit[i].s);
        return ans;
    }
};
```

11.5 Edge Coloring (6)

```
/**
 * Description:
    https://en.m.wikipedia.org/wiki/Vizing%27s_theorem
 * Source: Own (not optimized)
 * Verification:
    https://open.kattis.com/problems/gamescheduling
 */

template<int SZ> struct EdgeColor {
    int n, adjVert[SZ][SZ], adjCol[SZ][SZ];
    int deg[SZ], maxDeg;

    EdgeColor(int _n) {
        n = _n; maxDeg = 0;
```

```
    FOR(i,n) {
        deg[i] = 0;
        FOR(j,n) adjVert[i][j] = adjCol[i][j] = -1;
    }
}

void delEdge(int x, int y) {
    if (adjVert[x][y] == -1) return;
    int C = adjVert[x][y];
    adjCol[x][C] = adjCol[y][C] = adjVert[x][y] =
        adjVert[y][x] = -1;
}

void setEdge(int x, int y, int c) { // delete
    previous value if it had one
    delEdge(x,y); assert(adjCol[x][c] == -1 &&
        adjCol[y][c] == -1);
    adjVert[x][y] = adjVert[y][x] = c,
        adjCol[x][c] = y, adjCol[y][c] = x;
}

void shiftPath(int x, vi p) {
    FORd(i,sz(p)) setEdge(x,p[i],notAdj[p[i]]);
}

vi getPath(int st, int c0, int c1) {
    vi res = {st};
    for (int nex = 0; ; nex ^= 1) {
        int c = (nex == 0 ? c0 : c1);
        if (adjCol[res.back()][c] == -1) return res;
        res.pb(adjCol[res.back()][c]);
    }
}

void flipPath(vi p, int c0, int c1) {
    FOR(i,sz(p)-1) delEdge(p[i],p[i+1]);
    FOR(i,sz(p)-1) {
        if (i&1) setEdge(p[i],p[i+1],c0);
        else setEdge(p[i],p[i+1],c1);
    }
}

int notAdj[SZ];

void addEdge(int x, int y) {
    maxDeg = max(maxDeg,max(++deg[x],++deg[y]));

    // generate a color which is not adjacent to
        each vertex
    FOR(i,n) {
        FOR(j,maxDeg+1) if (adjCol[i][j] == -1) {
            notAdj[i] = j;
            break;
        }
    }

    vi nex(n);
    FOR(i,n) if (adjVert[x][i] != -1) nex[i] =
        adjCol[x][notAdj[i]];
    nex[y] = adjCol[x][notAdj[y]];

    // generate sequence of neighbors
    vi vis(n), seq = {y};
    while (seq.back() != -1 && !vis[seq.back()]) {
        vis[seq.back()] = 1;
        seq.pb(nex[seq.back()]);
    }

    // case 1: easy
    if (seq.back() == -1) {
        seq.pop_back(), shiftPath(x,seq);
        return;
    }

    // separate into path and cycle
    int ind = 0; while (seq[ind] != seq.back())
        ind ++;
    seq.pop_back();
    vi path = vi(seq.begin(),seq.begin()+ind);
    vi cyc = vi(seq.begin()+ind,seq.end());
    int c0 = notAdj[x], c1 = notAdj[cyc.back()];

    // case based on a/b path
    vi p = getPath(cyc.back(),c0,c1);
    if (p.back() != path.back()) {
        if (p.back() == x) { p.pop_back(),
            delEdge(x,p.back()); }
        flipPath(p,c0,c1);
        notAdj[seq.back()] = c0; shiftPath(x,seq);
    } else {
        reverse(all(p));
        flipPath(p,c0,c1);
        notAdj[path.back()] = c0; shiftPath(x,path);
    }
  }
};
```

11.6 Unweighted Matching (6)

```
/**
* Description:
    https://www-m9.ma.tum.de/graph-algorithms/matchings-bloss
* Source:
    https://github.com/koosaga/DeobureoMinkyuParty
* Verification:
    https://codeforces.com/contest/1089/problem/B
*/

template<int SZ> struct match {
    int vis[SZ], par[SZ], orig[SZ], match[SZ],
        aux[SZ], t, N; // 1-based index
    vi adj[SZ];
    queue<int> Q;

    void addEdge(int u, int v) {
        adj[u].pb(v); adj[v].pb(u);
    }

    void init(int n) {
        N = n; t = 0;
        FOR(i,N+1) {
            adj[i].clear();
```

```
            match[i] = aux[i] = par[i] = 0;
    }
}

void augment(int u, int v) {
    int pv = v, nv;
    do {
            pv = par[v]; nv = match[pv];
            match[v] = pv; match[pv] = v;
            v = nv;
    } while(u != pv);
}

int lca(int v, int w) {
    ++t;
    while (1) {
            if (v) {
                    if (aux[v] == t) return v;
                        aux[v] = t;
                    v = orig[par[match[v]]];
            }
            swap(v, w);
    }
}

void blossom(int v, int w, int a) {
    while (orig[v] != a) {
            par[v] = w; w = match[v];
            if (vis[w] == 1) Q.push(w), vis[w] = 0;
            orig[v] = orig[w] = a;
            v = par[w];
    }
}

bool bfs(int u) {
    fill(vis+1, vis+1+N, -1); iota(orig + 1, orig
        + N + 1, 1);
    Q = queue<int> (); Q.push(u); vis[u] = 0;
    while (sz(Q)) {
            int v = Q.front(); Q.pop();
            trav(x,adj[v]) {
                    if (vis[x] == -1) {
                            par[x] = v; vis[x] = 1;
                            if (!match[x]) return
                                augment(u, x), true;
                            Q.push(match[x]);
                                vis[match[x]] = 0;
                    } else if (vis[x] == 0 &&
                        orig[v] != orig[x]) {
                            int a = lca(orig[v],
                                orig[x]);
                            blossom(x, v, a);
                                blossom(v, x, a);
                    }
            }
    }
    return false;
}

int Match() {
    int ans = 0;
        // find random matching (not necessary,
            constant improvement)
        vi V(N-1); iota(all(V), 1);
        shuffle(all(V), mt19937(0x94949));
        trav(x,V) if(!match[x])
                trav(y,adj[x]) if (!match[y]) {
                        match[x] = y, match[y] = x;
                        ++ans; break;
                }

        FOR(i,1,N+1) if (!match[i] && bfs(i)) ++ans;
        return ans;
    }
};
```

12 Geometry (4)

12.1 Techniques

12.1.1 3D Geometry

```
/**
 * Description: Basic 3D Geometry
 * Source: Own
 * Verification: AMPPZ 2011 Cross Spider
 */

typedef vl P;

namespace point3 {
    ld norm(P x) {
        ld sum = 0; FOR(i,sz(x)) sum += (ld)x[i]*x[i];
        return sum;
    }
    ld abs(P x) { return sqrt(norm(x)); }

    P operator+(const P& a, const P& b) {
        P c(sz(a)); FOR(i,sz(a)) c[i] = a[i]+b[i];
        return c;
    }
    P operator+=(P& l, const P& r) { return l = l+r; }
    P operator-(const P& a, const P& b) {
        P c(sz(a)); FOR(i,sz(a)) c[i] = a[i]-b[i];
        return c;
    }
    P operator-=(P& l, const P& r) { return l = l-r; }

    ld dot(P a, P b) {
        ld sum = 0; FOR(i,sz(a)) sum += a[i]*b[i];
        return sum;
    }
    P cross(P a, P b) {
        return {a[1]*b[2]-a[2]*b[1],
                a[2]*b[0]-a[0]*b[2],
                a[0]*b[1]-a[1]*b[0]};
    }

    bool isMult(P a, P b) {
        auto c = cross(a,b);
```

```
        FOR(i,sz(c)) if (c[i] != 0) return 0;
        return 1;
    }
    bool collinear(P a, P b, P c) { return
        isMult(b-a,c-a); }
    bool coplanar(P a, P b, P c, P d) {
        return isMult(cross(b-a,c-a),cross(b-a,d-a));
    }
}

using namespace point3;
```

12.1.2 Closest Pair

```
/**
* Description: O(NlogN) line sweep to find two closest
    points
* Source: Own
* Verification:
    https://open.kattis.com/problems/closestpair2
*/

using namespace point;

pair<P,P> solve(vP v) {
    pair<ld,pair<P,P>> bes; bes.f = INF;
    set<P> S; int ind = 0;

    sort(all(v));
    FOR(i,sz(v)) {
        if (i && v[i] == v[i-1]) return {v[i],v[i]};

        for (; v[i].f-v[ind].f >= bes.f; ind++)
            S.erase({v[ind].s,v[ind].f});

        for (auto it = S.ub({v[i].s-bes.f,INF});
            it != S.end() && it->f < v[i].s+bes.f;
            it = next(it)) {
            P t = {it->s,it->f};
            ckmin(bes,{abs(t-v[i]),{t,v[i]}});
        }

        S.insert({v[i].s,v[i].f});
    }

    return bes.s;
}
```

12.1.3 Point

```
/**
* Description: Easy Geo
* Source: http://codeforces.com/blog/entry/22175, KACTL
* Verification: various
*/

typedef pd P;

namespace point {
    typedef vector<P> vP;

    ld norm(P x) { return x.f*x.f+x.s*x.s; }
    ld abs(P x) { return sqrt(norm(x)); }
    ld angle(P x) { return atan2(x.s,x.f); }
    P conj(P x) { return P(x.f,-x.s); }

    P operator+(const P& l, const P& r) { return
        P(l.f+r.f,l.s+r.s); }
    P operator-(const P& l, const P& r) { return
        P(l.f-r.f,l.s-r.s); }
    P operator*(const P& l, const ld& r) { return
        P(l.f*r,l.s*r); }
    P operator*(const ld& l, const P& r) { return r*l;
        }
    P operator/(const P& l, const ld& r) { return
        P(l.f/r,l.s/r); }

    P operator*(const P& l, const P& r) { return
        P(l.f*r.f-l.s*r.s,l.s*r.f+l.f*r.s); }
    P operator/(const P& l, const P& r) { return
        l*conj(r)/norm(r); }

    template<class T> T operator += (T& l, const T& r)
        { return l = l+r; }
    template<class T> T operator -= (T& l, const T& r)
        { return l = l-r; }
    template<class T> T operator *= (T& l, const T& r)
        { return l = l*r; }
    template<class T> T operator /= (T& l, const T& r)
        { return l = l/r; }

    ld dot(P a, P b) { return (conj(a)*b).f; }
    ld cross(P a, P b) { return (conj(a)*b).s; }
    ld cross(P p, P a, P b) { return cross(a-p,b-p); }
    ld dist(P p, P a, P b) { return
        std::abs(cross(p,a,b))/abs(a-b); }

    P rotate(P a, ld b) { return a*P(cos(b),sin(b)); }
    P reflect(P p, P a, P b) { return
        a+conj((p-a)/(b-a))*(b-a); }
    P foot(P p, P a, P b) { return
        (p+reflect(p,a,b))/(ld)2; }
    P extension(P a, P b, P c, P d) {
        ld x = cross(a,b,c), y = cross(a,b,d);
        return (d*x-c*y)/(x-y);
    }

    // sorts points according to atan2
    // verification: ?
    template<class T> int half(pair<T,T> x) { return
        mp(x.s,x.f) > mp((T)0,(T)0); }
    bool cmp(P a, P b) {
        int A = half(a), B = half(b);
        if (A != B) return A < B;
        return cross(a,b) > 0;
    }

    // computes the center of mass of a polygon with
        constant mass per unit area
```

```
    // verification: kattis polygonarea, VT HSPC 2018
        Holiday Stars
    P centroid(vP v) {
        P cen(0,0); ld area = 0;
        FOR(i,sz(v)) {
            int j = (i+1)%sz(v);
            ld a = cross(v[i],v[j]);
            cen += a*(v[i]+v[j]); area += a;
        }
        area /= (ld)2; // positive if ccw
        return cen/area/(ld)6;
    }

    // tests whether a point is inside, on, or outside
        the perimeter of any polygon
    // verification:
        https://open.kattis.com/problems/pointinpolygon
    string inPoly(vP p, P z) {
        int n = sz(p), ans = 0;
        FOR(i,n) {
            P x = p[i], y = p[(i+1)%n];
            if (cross(x,y,z) == 0 && min(x,y) <= z && z
                <= max(x,y)) return "on";
            if (x.s > y.s) swap(x,y);
            if (x.s <= z.s && y.s > z.s && cross(z,x,y)
                > 0) ans ^= 1;
        }
        return ans ? "in" : "out";
    }
};

using namespace point;
```

12.2 Sweep Line

12.2.1 Convex Hull

```
/**
* Description: Top-bottom convex hull
* Source: Wikibooks
* Verification:
    https://open.kattis.com/problems/convexhull
*/

typedef pl P;

using namespace point;

vP convex_hull(vP P) {
    sort(all(P)); P.erase(unique(all(P)),P.end());
    int n = sz(P);
    if (n == 1) return P;

    vP bot = {P[0]};
    FOR(i,1,n) {
        while (sz(bot) > 1 && cross(bot[sz(bot)-2],
            bot.back(), P[i]) <= 0) bot.pop_back();
        bot.pb(P[i]);
    }
    bot.pop_back();
    vP up = {P[n-1]};
    FORd(i,n-1) {
        while (sz(up) > 1 && cross(up[sz(up)-2],
            up.back(), P[i]) <= 0) up.pop_back();
        up.pb(P[i]);
    }
    up.pop_back();

    bot.insert(bot.end(),all(up));
    return bot;
}
```

12.2.2 LiChao Segment Tree

```
/**
* Description: LiChao Segment Tree
* Source: atatomir, misc
* Verification: CSA Squared Ends
*/

struct Line {
        ll k,m;
        Line(ll _k, ll _m) { k = _k, m = _m; }
        Line() : Line(0,-INF) { }
        ll get(ll x) { return k*x+m; }
        bool majorize(Line X, int L, int R) {
            return get(L) >= X.get(L) && get(R) >=
                X.get(R);
        }
};

struct lc {
    lc* c[2];
    Line S;

    lc() {
        c[0] = c[1] = NULL;
        S = Line();
    }

    void rm() {
        if (c[0]) c[0]->rm();
        if (c[1]) c[1]->rm();
        delete this;
    }

    void mc(int i) {
        if (!c[i]) c[i] = new lc();
    }

    ll query(ll X, ll L, ll R) {
        ll ans = S.get(X), M = (L+R)/2;
        if (X <= M) return max(ans, c[0] ?
            c[0]->query(X,L,M): -INF);
        return max(ans, c[1] ? c[1]->query(X,M+1,R):
            -INF);
    }

    void modify(Line X, ll L, ll R) {
```

```
        if (X.majorize(S,L,R)) swap(X,S);
        if (S.majorize(X,L,R)) return;
        if (S.get(L) < X.get(L)) swap(X,S);

        ll M = (L+R)/2;
        if (X.get(M) >= S.get(M)) swap(X,S), mc(0),
            c[0]->modify(X,L,M);
        else mc(1), c[1]->modify(X,M+1,R);
    }

    void upd(Line X, ll lo, ll hi, ll L, ll R) { //
        untested
        if (R < hi || L < lo) return;
        if (lo <= L && R <= hi) { modify(X,L,R);
            return; }
        ll M = (L+R)/2;
        mc(0), c[0]->upd(X,lo,hi,L,M);
        mc(1), c[1]->upd(X,lo,hi,M+1,R);
    }
};
```

12.2.3 LineContainer

```
/**
* Description: Given set of lines, computes the
    greatest y-coordinate for any x
* Source: KACTL
* Verification: CSA Squared Ends
*/

bool Q;
struct Line {
        mutable ll k, m, p; // slope, y-intercept,
            last optimal x
        bool operator<(const Line& o) const {
                return Q ? p < o.p : k < o.k;
        }
};

struct LineContainer : multiset<Line> {
        const ll inf = LLONG_MAX;
        ll div(ll a, ll b) { // floored division
            if (b < 0) a *= -1, b *= -1;
            if (a >= 0) return a/b;
            return -((-a+b-1)/b);
        }

        // updates x->p, determines if y is unneeded
        bool isect(iterator x, iterator y) {
                if (y == end()) { x->p = inf; return 0;
                    }
                if (x->k == y->k) x->p = x->m > y->m ?
                    inf : -inf;
                else x->p = div(y->m - x->m, x->k -
                    y->k);
                return x->p >= y->p;
        }

        void add(ll k, ll m) {
                auto z = insert({k, m, 0}), y = z++, x
                    = y;
                while (isect(y, z)) z = erase(z);
                if (x != begin() && isect(--x, y))
                    isect(x, y = erase(y));
                while ((y = x) != begin() && (--x)->p
                    >= y->p) isect(x, erase(y));
        }

        ll query(ll x) {
                assert(!empty());
                Q = 1; auto l = *lb({0,0,x}); Q = 0;
                return l.k * x + l.m;
        }
};
```

12.2.4 Maximum Rectangle

```
/**
* Description: Computes size of max rectangle in grid
    w/ obstacles
* Source: Own
* Verification: https://cses.fi/problemset/task/1147/
*/

int n,m,cur[1000];
char g[1000][1000];
ll ans = 0;

void processCol(int x) {
    vi nex[m+1];
    FOR(i,n) nex[cur[i]-x].pb(i);

    DSU<1000> D = DSU<1000>();
    FORd(i,m+1) for (int a: nex[i]) {
        D.par[a] = a;
        if (a > 0 && D.par[a-1] != -1) D.unite(a,a-1);
        if (a < n-1 && D.par[a+1] != -1)
            D.unite(a,a+1);
        ans = max(ans,i*(ll)D.sz[D.get(a)]);
    }
}

int solve() {
    FOR(i,n) cur[i] = m;
    FORd(j,m) {
        FOR(i,n) if (g[i][j] == '*') cur[i] = j; //
            obstacle
        processCol(j);
    }
    return ans;
}
```

12.3 Lattice Point Counter

```
/**
 * Description: Counts the number of lattice points
     (x,y) such that 0<x, 0<y, ax+by <= c
```

```
 * Source: Own
 * Verification:
     https://codeforces.com/contest/1098/problem/E
 */

ll getSum (ll a, ll b) { // sum of a+(a-b)+(a-2b)+...
    if (a <= 0) return 0;
    ll z = a/b;
    return (z+1)*(2*a-z*b)/2;
}

ll getTri(ll c, ll a, ll b) {
    if (a+b > c) return 0;
    if (a > b) swap(a,b);
    ll k = b/a;
    // if ax+kay <= kac/b then we can divide by a and
        count
    // otherwise x > kc/b-ky so we can subtract the
        latter expression from x and continue
    return
        getTri(c-a*(c*k/b),a,b-k*a)+getSum(c*k/b-k,k);
}

ll brute(ll c, ll a, ll b) {
    ll ans = 0;
    for (ll i = c-a; i >= 0; i -= a) ans += i/b;
    return ans;
}
```

12.4 Max Collinear

```
/**
* Description: Compute the maximum number of points
    which lie on the same line in O(n^2logn)
* Source: own
* Verification:
    https://open.kattis.com/problems/maxcolinear
*/

int n, mx, ans;
map<pair<pi,int>,int> m;
pi p[1000];

pair<pi,int> getline(pi a, pi b) {
    pi z = {b.f-a.f,b.s-a.s};
    swap(z.f,z.s); z.f *= -1;
    int g = __gcd(z.f,z.s); z.f /= g, z.s /= g;
    if (z.f < 0 || (z.f == 0 && z.s < 0)) z.f *= -1,
        z.s *= -1;
    return {z,z.f*a.f+z.s*a.s};
}

void solve() {
    mx = ans = 0; m.clear();
    FOR(i,n) cin >> p[i].f >> p[i].s;
    FOR(i,n) FOR(j,i+1,n) m[getline(p[i],p[j])] ++;

    for (auto a: m) mx = max(mx,a.s);
    FOR(i,1,n+1) if (i*(i-1)/2 <= mx) ans = i;
    cout << ans << "\n";
}
```

12.5 Delaunay (6)

```
/*
* Description: Delaunay Triangulation w/ Bowyer-Watson
    O(n^2logn)
* Source: Own
* Verification: ICPC WF Panda Preserve
*/

namespace Delaunay {
    // stay with __int128 for better precision, if
        possible
    ld cross(cd b, cd c) { return (conj(b)*c).imag(); }
    ld cross(cd a, cd b, cd c) { return
        cross(b-a,c-a); }

    bool inCircle (cd a, cd b, cd c, cd d) {
        a -= d, b -= d, c -= d;
        ld x = norm(a)*cross(b,c)+norm(b)*cross(c,a)
               +norm(c)*cross(a,b);
        if (cross(a,b,c) < 0) x *= -1;
        return x > 0;
    }

    vector<array<int,3>> triangulate(vcd v) {
        // works with cyclic quads
        // not when all points are collinear!
        // creates super-triangle, adjusts as necessary

        v.pb(cd(-1e5,-1e5)); v.pb(cd(1e5,0));
            v.pb(cd(0,1e5));

        vector<array<int,3>> ret;
        ret.pb({sz(v)-3,sz(v)-2,sz(v)-1});

        FOR(i,sz(v)-3) {
            map<pi,int> m;
            vector<array<int,3>> tmp;
            for (auto a: ret) {
                if
                    (inCircle(v[a[0]],v[a[1]],v[a[2]],v[i]))
                    m[{a[0],a[1]}] ++, m[{a[1],a[2]}]
                        ++, m[{a[0],a[2]}] ++;
                else tmp.pb(a);
            }
            for (auto a: m) if (a.s == 1) {
                array<int,3> x = {a.f.f,a.f.s,i};
                    sort(all(x));
                tmp.pb(x);
            }
            ret = tmp;
        }
        vector<array<int,3>> tmp;
        for (auto a: ret) if (a[2] < sz(v)-3)
            tmp.pb(a);
        return tmp;
    }
}
```

```
    void print(vcd x) { // produces asymptote code
        cout << "[asy]\n";
        cout << "pair[] A = {";
        bool done = 0;
        for (auto a: x) {
            if (done) cout << ",";
            cout << a; done = 1;
        }
        cout << "};\n";

        cout << "for (int i = 0; i < " << sz(x) << ";
            ++i) {\n\tdot(A[i]);\n}\n";

        for (auto b: triangulate(x)) cout << "draw(A["
            << b[0] << "]--A[" << b[1] << "]--A[" <<
            b[2] << "]--cycle);\n";
        cout << "[/asy]\n";
    }
};
```

12.6 Linear Programming (6)

```
/**
 * Description: Simplex Algorithm for linear
     programming
    * maximize cT x subject to Ax <= b, x >= 0; (oops
       I should try to understand all the code)
 * Usage:
       *
           https://open.kattis.com/contests/fvfhq4/problems/goatropes
       * http://codeforces.com/contest/375/problem/E
 * Source: KACTL, Stanford
 */

typedef double T;
typedef vector<T> vd;
typedef vector<vd> vvd;

const T eps = 1e-8, inf = 1/.0;
#define ltj(X) if(s == -1 || mp(X[j],N[j]) <
    mp(X[s],N[s])) s=j

struct LPSolver {
       int m, n;
       vi N, B;
       vvd D;

       LPSolver(const vvd& A, const vd& b, const vd&
           c) :
              m(sz(b)), n(sz(c)), N(n+1), B(m),
                  D(m+2, vd(n+2)) {
                 FOR(i,m) FOR(j,n) D[i][j] =
                     A[i][j];
                 FOR(i,m) { B[i] = n+i; D[i][n] =
                     -1; D[i][n+1] = b[i];}
                 FOR(j,n) { N[j] = j; D[m][j] =
                     -c[j]; }
                 N[n] = -1; D[m+1][n] = 1;
              }

       void pivot(int r, int s) { // row, column
              T *a = D[r].data(), inv = 1 / a[s]; //
                  eliminate col s from consideration?
              FOR(i,m+2) if (i != r && abs(D[i][s]) >
                  eps) {
                     T *b = D[i].data(), inv2 = b[s]
                         * inv;
                     FOR(j,n+2) b[j] -= a[j] * inv2;
                     b[s] = a[s] * inv2;
              }
              FOR(j,n+2) if (j != s) D[r][j] *= inv;
              FOR(i,m+2) if (i != r) D[i][s] *= -inv;
              D[r][s] = inv;
              swap(B[r], N[s]);
       }

       bool simplex(int phase) {
              int x = m + phase - 1;
              for (;;) {
                     int s = -1;
                     FOR(j,n+1) if (N[j] != -phase)
                         ltj(D[x]); // find most
                         negative col
                     if (D[x][s] >= -eps) return true;
                     int r = -1;
                    FOR(i,m) {
                                 if (D[i][s] <= eps)
                                     continue;
                                 if (r == -1 ||
                                     mp(D[i][n+1] /
                                     D[i][s], B[i])
                                             <
                                                  mp(D[r][n+1]
                                                  /
                                                  D[r][s],
                                                  B[r])) r
                                                  = i; //
                                                  find
                                                  smallest
                                                  positive
                                                  ratio
                     }
                     if (r == -1) return false;
                     pivot(r, s);
              }
       }

       T solve(vd &x) {
              int r = 0;
              FOR(i,1,m) if (D[i][n+1] < D[r][n+1]) r
                  = i;
              if (D[r][n+1] < -eps) {
                     pivot(r, n);
                     if (!simplex(2) || D[m+1][n+1] <
                         -eps) return -inf;
                     FOR(i,m) if (B[i] == -1) {
                            int s = 0;
                            FOR(j,1,n+1) ltj(D[i]);
                            pivot(i, s);
                     }
              }
              bool ok = simplex(1); x = vd(n);
```

```
            FOR(i,m) if (B[i] < n) x[B[i]] =
                D[i][n+1];
            return ok ? D[m][n+1] : inf;
        }
};

/*vvd A; vd B,C;
int co = 0; // # extra variables

void addIneq(vd a, double b) { // a*x <= b
    FOR(i,co) a.pb(0);
    a.pb(1); co ++;
    for (auto& z: A) z.pb(0);
    A.pb(a); B.pb(b); C.pb(0);
}*/
```

13 Additional (4)

13.1 Mo

```
/**
* Description: Answers queries offline in (N+Q)sqrt(N)
    * Also see Mo's on trees
* Source: Codeforces
* Verification: ?
*/

int N, A[MX];
int ans[MX], oc[MX], BLOCK;
vector<array<int,3>> todo; // store left, right, index
    of ans

bool cmp(array<int,3> a, array<int,3> b) { // sort
    queries
    if (a[0]/BLOCK != b[0]/BLOCK) return a[0] < b[0];
    return a[1] < b[1];
}

int l = 0, r = -1, cans = 0;

void ad(int x, int y = 1) {
    x = A[x];
    // if condition: cans --;
    oc[x] += y;
    // if condition: cans ++;
}

int answer(int L, int R) { // adjust interval
    while (l > L) ad(--l);
    while (r < R) ad(++r);
    while (l < L) ad(l++,-1);
    while (r > R) ad(r--,-1);
    return cans;
}

void solve() {
    BLOCK = sqrt(N); sort(all(todo),cmp);
    trav(x,todo) {
        answer(x[0],x[1]);
        ans[x[2]] = cans;
    }
}
```