# USACO Notebook

Ben Qi

July 17, 2018

## Contents

# 1 Contest

## 1.1 C++ Template

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

using namespace std;
using namespace __gnu_pbds;

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 sz(x) (int)(x).size()
#define mp make_pair
#define pb push_back
#define f first
#define s second
#define lb lower_bound
#define ub upper_bound
#define all(x) x.begin(), x.end()

const int MOD = 1000000007;
const ll INF = 1e18;
const int MX = 100001;

int main() {
   ios_base::sync_with_stdio(0); cin.tie(0);

}

/* Look for:
* the exact constraints (multiple sets are too slow
    for n=10^6 :( )
* special cases (n=1?)
* overflow (ll vs int?)
* array bounds
*/
```

## 1.2 FastScanner

```java
/**
* Source: Matt Fontaine
*/

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();
    }
}
```

## 1.3 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
* Verification:
    https://open.kattis.com/problems/intervalcover
   * actually, you need to account for A=B and add
       epsilons but w/e
*/

double A,B; // interval to be covered, assuming A<B
vector<pair<pd,int>> in; // intervals
int N; // # of intervals

vi solve() {
    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
*/

int main() {
    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
* Verification: POI 12 - The Bus
*/

void compress(vector<array<int,3>>& 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 Customization

```
/**
* Source: StackOverflow
* Description: Define your own comparator / hash
    function
*/

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

struct hsh {
    size_t operator()(const pi& k) const {
        return k.f^k.s; // bad, but you get the point
    }
};

set<int,cmp> s;
map<int,int,cmp> m;
unordered_map<pi,int,hsh> u;
```

# 4 Graphs Easy (2)

## 4.1 Traversal

### 4.1.1 BFS on Grid

```
/**
* Note: Use xdir and ydir
*/

int xdir[4] = {0,1,0,-1}, ydir[4] = {1,0,-1,0};
int dist[21][21];
queue<pi> todo;

void process(pi x) {
        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
                }
        }
}

int main() {
        FOR(i,21) FOR(j,21) dist[i][j] = MOD;
        dist[10][10] = 0; todo.push({10,10}); //
            initialize queue, distances
        while (todo.size()) {
            process(todo.front());
            todo.pop(); // pop point from queue
        }
        cout << dist[4][5]; // 11
}
```

### 4.1.2 DFS

```
int n, 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

}

int main() {
        cin >> n;
        FOR(i,n-1) {
            int a,b; cin >> a >> b;
            adj[a].pb(b), adj[b].pb(a);
        }
        dfs(1);
}
```

## 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
* Verification:
    https://open.kattis.com/problems/shortestpath3
*/

const ll INF = 1e18;

int n,m,q,s,bad[1000];
vector<pair<pi,int>> edge;
ll dist[1000];

void solve() {
    edge.clear();
    FOR(i,n) dist[i] = INF, bad[i] = 0;
    dist[s] = 0;
    FOR(i,m) {
        int u,v,w; cin >> u >> v >> w;
        edge.pb({{u,v},w});
    }
    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;

    FOR(i,q) {
        int x; cin >> x;
        if (bad[x]) cout << "-Infinity\n";
        else if (dist[x] == INF) cout <<
            "Impossible\n";
        else cout << dist[x] << "\n";
    }
    cout << "\n";
}
```

### 4.2.2 Dijkstra

```
/**
* Description: shortest path!
* Works with negative edge weights (aka SPFA?)
*/

template<int SZ> struct Dijkstra {
    int dist[SZ];
    vpi adj[SZ];
    priority_queue<pi,vpi,greater<pi>> q;

    void gen() {
        fill_n(dist,SZ,MOD); dist[0] = 0;

        q.push({0,0});
        while (sz(q)) {
                pi x = q.top(); q.pop();
                if (dist[x.s] < x.f) continue;
                for (pi y: adj[x.s]) if (x.f+y.s <
                     dist[y.f]) {
                        dist[y.f] = x.f+y.s;
                        q.push({dist[y.f],y.f});
                }
        }
    }
};
```

### 4.2.3 Floyd-Warshall

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

int n,m,q; // vertices, edges, queries
ll dist[150][150], bad[150][150];

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;
    }

    FOR(i,q) {
        int u,v; cin >> u >> v;
        if (bad[u][v]) cout << "-Infinity\n";
        else if (dist[u][v] == INF) cout <<
            "Impossible\n";
        else cout << dist[u][v] << "\n";
    }
    cout << "\n";
}
```

## 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 DSU

```
/**
* Description: Disjoint Set Union
* 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;
    }
};
```

### 4.4.2 Kruskal

```
/**
* Description: computes the minimum spanning tree in
    O(ElogE) time
* Verification: USACO superbull
*/

ll kruskal(vector<array<int,3>> edge) {
    DSU<MX> D;
    sort(all(edge));
    ll ans = 0;
    for (auto a: edge) if (D.unite(a[1],a[2])) ans +=
        a[0]; // edge is in MST
    return ans;
}
```

# 5 Algorithm Design (2)

## 5.1 Minimum Deque (3)

```
/**
* Source: own
* Verification: Jan 18 Lifeguards
*/

struct MinDeque {
    int lo = 0, hi = -1;
    deque<pi> d;

    void ins(int 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();
    }

    int get() {
        return sz(d) ? d.front().f : MOD;
    }
};
```

## 5.2 Ternary Search (4)

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

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
 * Verification: POI 16 Ticket Inspector
 */

template<class T, int SZ> struct sums {
    T sum[SZ][SZ];
    sums () { 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.
* Verification: Problem Tournament from IOI 2012:
    http://wcipeg.com/problem/ioi1223
* Source code: 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 Wavelet Tree (6)

```
/**
* Description: Segment tree on values instead of
    indices
* Verification: http://www.spoj.com/problems/MKTHNUM/
*/

int N,Q, A[100000];
map<int,int> m;
vi revm;

void input() {
        cin >> N >> Q;
        FOR(i,N) cin >> A[i];
}

void compress() {
    FOR(i,N) m[A[i]] = 0;
    int nex = 0;
    for (auto& a: m) {
        a.s = nex++;
        revm.pb(a.f);
    }
    FOR(i,N) A[i] = m[A[i]];
}

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) {
            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);
    }
};

wavelet<1<<17> w;

int main() {
    input();
    compress();
    w.build();

    FOR(i,Q) {
        int l,r,k; cin >> l >> r >> k;
        cout << revm[w.query(l-1,r-1,k)] << "\n";
    }
}
```

## 6.2 1D Range Queries (3)

### 6.2.1 Binary Indexed Tree

```
/**
* Description: 1D range sum query with point update
* Verification: SPOJ Fenwick
*/

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

    BIT() { memset(bit,0,sizeof bit); }

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

    T query(int k) {
        T temp = 0;
        for (;k > 0;k -= (k&-k)) temp += bit[k];
        return temp;
    }
    T query(int l, int r) { return
         query(r)-query(l-1); } // range query [l,r]
};
```

### 6.2.2 SegTree

```
/*
* 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 BIT with Range Update (4)

```
/**
* Source: GeeksForGeeks?
* Description: 1D range update, range query
* Alternative to lazy segment tree
*/

// BIT template

template<class T, int SZ> struct BITrange {
    BIT<T,SZ> bit[2]; // sums 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 query(int x) { return
         bit[1].query(x)*x+bit[0].query(x); }
    T query(int x, int y) { return
         query(y)-query(x-1); }
};
```

### 6.2.4 Lazy SegTree (4)

```
/**
* Description: 1D range update, range query
* 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);
    }
};
```

### 6.2.5 Sparse SegTree (4)

```
/**
* Source: Own
*/

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;
    }
};
```

### 6.2.6 SegTree Beats (6)

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

const int MX = 1<<20;

int N,M, a[MX];

struct Seg {
    ll sum[2*MX];
    int mx1[2*MX], mx2[2*MX], maxCnt[2*MX];

    void pull(int ind) {
        mx1[ind] = max(mx1[2*ind],mx1[2*ind+1]);
        mx2[ind] = max(mx2[2*ind],mx2[2*ind+1]);
        maxCnt[ind] = 0;

        if (mx1[2*ind] == mx1[ind]) maxCnt[ind] +=
             maxCnt[2*ind];
        else mx2[ind] = max(mx2[ind],mx1[2*ind]);

        if (mx1[2*ind+1] == mx1[ind]) maxCnt[ind] +=
             maxCnt[2*ind+1];
        else mx2[ind] = max(mx2[ind],mx1[2*ind+1]);

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

    void build(int ind = 1, int L = 0, int R = N-1) {
        if (L == R) {
            mx1[ind] = sum[ind] = a[L];
            maxCnt[ind] = 1;
            mx2[ind] = -1;
            return;
        }

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

    void push(int ind, int L, int R) {
        if (L == R) return;
        if (mx1[2*ind] > mx1[ind]) {
            sum[2*ind] -=
                 (ll)maxCnt[2*ind]*(mx1[2*ind]-mx1[ind]);
            mx1[2*ind] = mx1[ind];
        }
        if (mx1[2*ind+1] > mx1[ind]) {
            sum[2*ind+1] -=
                 (ll)maxCnt[2*ind+1]*(mx1[2*ind+1]-mx1[ind]);
            mx1[2*ind+1] = mx1[ind];
        }
    }

    void modify(int x, int y, int t, int ind = 1, int
         L = 0, int R = N-1) {
        if (R < x || y < L || mx1[ind] <= t) return;
        push(ind,L,R);
        if (x <= L && R <= y && mx2[ind] < t) {
            sum[ind] -= (ll)maxCnt[ind]*(mx1[ind]-t);
            mx1[ind] = t;
            return;
        }
        if (L == R) return;
        int M = (L+R)/2;
        modify(x,y,t,2*ind,L,M);
        modify(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 = N-1) {
        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 = N-1) {
        if (R < x || y < L) return -1;
        push(ind,L,R);
        if (x <= L && R <= y) return mx1[ind];

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

Seg S = Seg();

void solve() {
        cin >> N >> M;
        FOR(i,N) cin >> a[i];
        S.build();

        FOR(i,M) {
            int t; cin >> t;
            if (t == 0) {
                int x,y,z; cin >> x >> y >> z;
                S.modify(x-1,y-1,z);
            } else if (t == 1) {
                int x,y; cin >> x >> y;
                cout << S.qmax(x-1,y-1) << "\n";
            } else {
                int x,y; cin >> x >> y;
                cout << S.qsum(x-1,y-1) << "\n";
            }
        }
}
```

## 6.3 2D Range Queries (4)

### 6.3.1 2D BIT

```
/**
```

```
* Description: Supports point update & range query,
    can be extended to range update
* Verification: SPOJ matsum
* Dependency: Binary indexed tree
*/

template<class T, int SZ> struct BIT2D {
    BIT<T,SZ> bit[SZ+1];
    void upd(int X, int Y, T val) {
        for (; X <= SZ; X += (X&-X)) bit[X].upd(Y,val);
    }
    T query(int X, int Y) {
        T ans = 0;
        for (; X > 0; X -= (X&-X)) ans +=
            bit[X].query(Y);
        return ans;
    }
    T query(int X1, int X2, int Y1, int Y2) {
        return query(X2,Y2)-query(X1-1,Y2)
            -query(X2,Y1-1)+query(X1-1,Y1-1);
    }
};

int main() {
        int T; cin >> T;
        FOR(i,T) {
            int N; cin >> N;
            BIT2D<ll,1024> B = BIT2D<ll,1024>();
            while (1) {
                string c; cin >> c;
                if (c == "SET") {
                    int x, y,num; cin >> x >> y >> num;
                    x++, y++;
                    B.upd(x,y,num-B.query(x,x,y,y));
                } else if (c == "SUM") {
                    int x1, y1, x2, y2; cin >> x1 >> y1
                        >> x2 >> y2;
                    x1 ++, y1 ++, x2 ++, y2++;
                    cout << B.query(x1,x2,y1,y2) << "\n";
                } else break;
            }
        }
}
```

### 6.3.2 2D SegBIT

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

const int SZ = 1<<17;

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);
    }
};
```

### 6.3.3 2D SegTree

```
/**
* 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;
    }
};
```

### 6.3.4 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 Treap

```
/*
* Sources: various
* Description: Easiest BBST
* Verification: http://www.spoj.com/problems/ORDERSET/
*/

struct tnode {
    int val, pri, sz;
    tnode *c[2];

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

    void inOrder(bool f = 0) {
        if (c[0]) c[0]->inOrder();
        cout << val << " ";
        if (c[1]) c[1]->inOrder();
        if (f) cout << "\n------------\n";
    }

    void recalc() {
        sz = 1+(c[0]?c[0]->sz:0)+(c[1]?c[1]->sz:0);
    }
};

pair<tnode*,tnode*> split(tnode* t, int v) { // >= v
     goes to the right
    if (!t) return {t,t};

    if (v <= t->val) {
        auto p = split(t->c[0], v);
        t->c[0] = p.s; t->recalc();
        return {p.f, t};
    } else {
        auto p = split(t->c[1], v);
        t->c[1] = p.f; t->recalc();
        return {t, p.s};
    }
}

pair<tnode*,tnode*> split_by_order(tnode* t, int v) {
    if (!t) return {t,t};
    int tmp = t->c[0]?t->c[0]->sz:0;
    if (v <= tmp) {
        auto p = split_by_order(t->c[0], v);
        t->c[0] = p.s; t->recalc();
        return {p.f, t};
    } else {
        auto p = split_by_order(t->c[1], v-tmp-1);
        t->c[1] = p.f; t->recalc();
        return {t, p.s};
    }
}

tnode* merge(tnode* l, tnode* r) {
    if (!l) return r;
    if (!r) return l;

    if (l->pri > r->pri) {
        l->c[1] = merge(l->c[1],r);
        l->recalc();
        return l;
    } else {
        r->c[0] = merge(l,r->c[0]);
        r->recalc();
        return r;
    }
}

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

```
tnode* del(tnode* x, int v) { // delete all values
     equal to v
    auto a = split(x,v), b = split(a.s,v+1);
    return merge(a.f,b.s);
}

tnode *root;

int order_of_key(int x) {
    auto a = split(root,x);
    int t = a.f?a.f->sz:0;
    root = merge(a.f,a.s);
    return t;
}

int find_by_order(int x) {
    auto a = split_by_order(root,x);
    auto b = split_by_order(a.f,x-1);
    int t = b.s->val;
    root = merge(merge(b.f,b.s),a.s);
    return t;
}
```

### 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
*/

template<int SZ> struct LCT {
    // [splay tree template]

    snode* S[SZ];
    LCT () { FOR(i,SZ) S[i] = new snode(i); }

    void dis(snode* x, int d) {
        snode* y = x->c[d];
        if (x) x->c[d] = NULL, x->recalc();
        if (y) y->p = NULL, y->pp = x;
    }

    void con(snode* x, int d) { setLink(x->pp,x,d);
         x->pp = NULL; }

    snode* getExtreme(snode* x, int d) {
        prop(x);
        if (x->c[d]) return getExtreme(x->c[d],d);
        return splay(x);
    }

    void setPref(snode* x) { splay(x->pp),
         dis(x->pp,1), con(x,1); splay(x); }

    snode* access(snode* x) { // x is brought to the
         root of auxiliary tree
        dis(splay(x),1);
        while (x->pp) setPref(x);
        return x;
    }

    //////// UPDATES

    snode* makeRoot(snode* v) { access(v)->flip = 1;
         return access(v); }

    void link(snode* v, snode* w) {
        access(w)->pp = makeRoot(v);
        con(w,0);
    }

    void cut(snode* x) { // cut link between x and its
         parent
        snode* y = access(x)->c[0];
        dis(x,0); y->pp = NULL;
    }

    //////// QUERIES

    int getDepth(snode* v) { access(v); return
         getNum(v->c[0]); }

    int getRoot(snode* v) { return
         getExtreme(access(v),0)->id; }

    int lca(snode* x, snode* y) {
        snode* root = getExtreme(access(y),0);

        dis(splay(x),1);
        auto z = getExtreme(x,0);
        if (z == root) return x->id;
        splay(x);

        while (x->pp) {
            auto z = getExtreme(splay(x->pp),0);
            if (z == root) return x->pp->id;
            setPref(x);
        }

        return -1;
    }
};
```

### 6.4.3 Splay Tree (5)

```
/**
* Description: Treap alternative
* Sources: see LCT
*/

struct snode {
    int id, num = 1;
    bool flip = 0;
    snode *p, *pp, *c[2];

    snode (int _id) {
        id = _id;
        c[0] = c[1] = p = pp = NULL;
    }

    void inOrder(bool f = 0) {
        if (c[0]) c[0]->inOrder();
        cout << id << " ";
        if (c[1]) c[1]->inOrder();
        if (f) cout << "\n------------\n";
    }

    void recalc() {
        num = 1+(c[0]?c[0]->num:0)+(c[1]?c[1]->num:0);
    }
};

int getNum(snode* x) { return x?x->num:0; }
int getDir(snode* x, snode* y) { return x?(x->c[1] ==
    y):-1; }

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

void pushDown(snode* x) {
    if (!x) return;
    if (x->p) pushDown(x->p);
    prop(x);
}

void rot(snode* x, int d) { // precondition: x &
    parents propagated
    snode *y = x->c[d], *z = x->p;
    prop(y);
    setLink(x, y->c[d^1], d);
    setLink(y, x, d^1);
    setLink(z, y, getDir(z, x));
    y->pp = x->pp; x->pp = NULL;
}

snode* splay(snode* x) {
    pushDown(x);
    while (x && x->p) {
        snode* y = x->p, *z = y->p;
        int dy = getDir(y, x), dz = getDir(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;
}
```

## 6.5 Lazy PST (5)

```
/**
* Description: persistent segtree without lazy updates
* Sources: CF, Franklyn Wang
*/

template<class T, int SZ> struct pseg {
    static const int LIMIT = 10000000;
    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
    }
    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)); }
};
```

# 7 DP (3)

## 7.1 Examples

### 7.1.1 Knapsack

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

double C;
int n,v[2000],w[2000],dp[2001][2001];

void solve() {
    FOR(i,n) cin >> v[i] >> w[i];
    FOR(i,n) {
        FOR(j,C+1) dp[i+1][j] = dp[i][j];
        FOR(j,C+1) if (w[i]+j <= C) dp[i+1][w[i]+j] =
            max(dp[i+1][w[i]+j],dp[i][j]+v[i]);
    }

    vi ans;
    int x = C;
    FORd(i,n) if (dp[i][x] != dp[i+1][x]) x -= w[i],
        ans.pb(i);
}
```

### 7.1.2 Longest Common Subsequence

```
/**
* Description: Classic DP example
*/

int dp[1001][1001];
```

```
string a,b;

int main() {
        cin >> a >> b;
        FOR(i,sz(a)) FOR(j,b.sz(b)) {
            dp[i+1][j+1] = max(dp[i+1][j],dp[i][j+1]);
            if (a[i] == b[j]) dp[i+1][j+1] =
                max(dp[i+1][j+1],dp[i][j]+1);
        }
        cout << dp[sz(a)][sz(b)];
}
```

### 7.1.3 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
}
```

### 7.1.4 String Removals

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

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

### 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;
}

int main() {
        int T; cin >> T;
        FOR(i,T) {
            cin >> N; N++;
            FOR(j,N) FOR(k,N) if (j != k) cin >>
                 dist[j][k];
            cout << solve() << "\n";
        }
}
```

## 7.2 Divide And Conquer (4)

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

void divi(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);

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

# 8 Strings (3)

## 8.1 Hashing

```
/**
* Source: own
* Description: Pairs reduce frequency of collision
* Verification: Dec 17 Plat 1
*/

template<class T> pair<T,T> operator+(const pair<T,T>&
     l, const pair<T,T>& r) {
    return {(l.f+r.f)%MOD,(l.s+r.s)%MOD};
}

template<class T> pair<T,T> operator-(const pair<T,T>&
     l, const pair<T,T>& r) {
    return {(l.f-r.f+MOD)%MOD,(l.s-r.s+MOD)%MOD};
}

template<class T> pair<T,T> operator*(const pair<T,T>&
     l, const T& r) {
    return {l.f*r%MOD,l.s*r%MOD};
}

template<class T> pair<T,T> operator*(const pair<T,T>&
     l, const pair<T,T>& r) {
    return {l.f*r.f%MOD,l.s*r.s%MOD};
}

struct hsh {
    string S;
    vector<pl> po, ipo, cum;
    pl base = mp(948392576,573928192), invbase =
         mp(499499562,829828935);

    ll modpow(ll b, ll p) {
        return !p?1:modpow(b*b%MOD,p/2)*(p&1?b:1)%MOD;
    }

    ll inv(ll x) {
        return modpow(x,MOD-2);
    }

    void gen(string _S) {
        S = _S;
        po.resize(sz(S)), ipo.resize(sz(S)),
             cum.resize(sz(S)+1);
        po[0] = ipo[0] = {1,1};
        FOR(i,1,sz(S)) {
            po[i] = po[i-1]*base;
            ipo[i] = ipo[i-1]*invbase;
        }
        FOR(i,sz(S)) cum[i+1] =
             cum[i]+po[i]*(ll)(S[i]-'a'+1);
    }

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

int lcp(hsh& a, hsh& b) { // can be used to generate a
     suffix array
    int lo = 0, hi = min(sz(a.S),sz(b.S));
    while (lo < hi) {
        int mid = (lo+hi+1)/2;
        if (a.get(0,mid-1) == b.get(0,mid-1)) lo = mid;
        else hi = mid-1;
    }
    return lo;
}
```

```
int main() {
    string _S = "abacaba";
    hsh h; h.gen(_S);
    FOR(i,sz(_S)) FOR(j,i,sz(_S)) cout << i << " " <<
        j << " " << h.get(i,j).f << " " <<
        h.get(i,j).s << "\n";

    hsh H; H.gen("abadaba");
    cout << lcp(h,H);
}
```

## 8.2 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) {
        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;
    }
};
```

## 8.3 String Searching (4)

### 8.3.1 Aho-Corasick

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

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

    Aho() {
        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 push_links() {
        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
        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;
    }
};

Aho<MX> A;

int n;

void solve() {
    A = Aho<MX>();
    cin >> n;
    FOR(i,n) {
        string pat; getline(cin,pat); if (!i)
            getline(cin,pat);
        A.add(pat);
    }
    A.push_links();

    string t; getline(cin,t);
    int cur = 0;
    FOR(i,sz(t)) cur = A.nex(i,cur,t[i]);
    FOR(i,1,n+1) {
        for (int j: A.oc[i]) cout << j << " ";
        cout << "\n";
    }
}
```

### 8.3.2 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;
}

int main() {
    vi v = manacher("abacaba");
    for (int i: v) cout << i << " ";
}
```

### 8.3.3 Minimum Rotation

```
/**
* Source: KACTL
* Unused
*/

int min_rotation(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;
}
```

### 8.3.4 Palindromic Tree

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

    void add_letter(int c) {
        s[n++] = c;
        last = get_link(last);
        if (!to[last][c]) {
            len[sz] = len[last]+2;
            link[sz] = to[get_link(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];
    }
};
```

### 8.3.5 Z

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

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

int main() {
        vi x = z("abcababcabcaba");
        for (int i: x) cout << i << " ";
        cout << "\n";

        x = get("abcab","uwetrabcerabcab");
        for (int i: x) cout << i << " ";
}
```

## 8.4 Suffix Array (4)

### 8.4.1 Suffix Array

```
/**
* Source: SuprDewd CP Course
* Task: https://open.kattis.com/problems/suffixsorting
* KACTL version is slightly faster
* Verification: USACO December 2017: Standing out from
    the herd:
    http://usaco.org/index.php?page=viewproblem2&cpid=768
* Code to Verify: https://pastebin.com/y2Z9FYr6
*/

struct suffix_array {
    int N;
    vector<vi> P;
    vector<array<int,3>> L;
    vi idx;
    string str;

    /*void bucket(int ind) {
        int mn = MOD, mx = -MOD;

        for (auto a: L) mn = min(mn,a[ind]), mx =
            max(mx,a[ind]);
        vector<array<int,3>> tmp[mx-mn+1];
        FORd(i,sz(L)) tmp[L[i][ind]-mn].pb(L[i]);

        int nex = 0;
        FOR(i,mx-mn+1) for (auto a: tmp[i]) L[nex++] =
            a;
    }

    void bucket_sort() {
        bucket(1), bucket(0);
    }*/

    suffix_array(string _str) {
        str = _str; N = sz(str);
        P.pb(vi(N)); L.resize(N);
        FOR(i,N) P[0][i] = str[i];

        for (int stp = 1, cnt = 1; cnt < N; stp ++,
            cnt *= 2) {
            P.pb(vi(N));
            FOR(i,N) L[i] = {P[stp-1][i],i+cnt < N ?
                P[stp-1][i+cnt] : -1,i};
            sort(all(L)); // bucket_sort();
            FOR(i,N) {
                if (i && mp(L[i][0],L[i][1]) ==
                    mp(L[i-1][0],L[i-1][1]))
                    P[stp][L[i][2]] = P[stp][L[i-1][2]];
                else P[stp][L[i][2]] = i;
            }
        }

        idx.resize(N);
        FOR(i,sz(P.back())) idx[P.back()[i]] = i;
    }

    int lcp(int x, int y) {
        int res = 0;
        if (x == y) return N-x;
        for (int k = sz(P) - 1; k >= 0 && x < N && y <
            N; k--) {
            if (P[k][x] == P[k][y]) {
                x += 1 << k;
                y += 1 << k;
                res += 1 << k;
            }
        }
        return res;
    }
};
```

### 8.4.2 Reverse Burrows-Wheeler (6)

```
/**
* Verification: https://cses.fi/problemset/task/1113/
*/

string transform(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 Trees (4)

## 9.1 Tree Diameter

```
/**
* Might not be obvious why this works!
* Verification: http://www.spoj.com/problems/PT07Z/
*/

int n, dist[MX], pre[MX];
vi adj[MX];

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 treeDiameter() {
    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 = treeDiameter();
    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};
}

int main() {
    cin >> n;
    FOR(i,n-1) {
        int a, b; cin >> a >> b;
        adj[a].pb(b), adj[b].pb(a);
    }
    vi x = genCenter();
    for (int i: x) cout << i << " ";
}
```

## 9.2 Queries

### 9.2.1 Heavy-Light Set

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

struct HeavyLightSet {
    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
    }
};
```

### 9.2.2 LCA Demo

```
/**
* Debug the Bugs
* Description: Use for both LCA's
*/

LCA L;

int Q;

int main() {
    cin >> L.V >> Q >> L.R;
    FOR(i,L.V-1) {
        int u,v; cin >> u >> v;
        L.addEdge(u,v);
    }
    L.construct();

    FOR(i,Q) {
        int u,v; cin >> u >> v;
        cout << L.lca(u,v) << "\n";
    }
}
```

### 9.2.3 LCA with Binary Jumps

```
/**
* 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 edges[SZ];
    int parK[32-__builtin_clz(SZ)][SZ], depth[SZ];

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

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

    void construct() {
        dfs(R, 0);
        FOR(k,1,MAXK) FOR(i,1,N+1)
            parK[k][i] = parK[k-1][parK[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 = parK[k][u];
        FORd(k,MAXK) if (parK[k][u] != parK[k][v]) u =
            parK[k][u], v = parK[k][v];

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

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

### 9.2.4 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 edges[SZ];
    RMQ<pi,2*SZ> r;
    vpi tmp;
    int depth[SZ], pos[SZ];

    int V, R;

    void addEdge(int u, int v) {
        edges[u].pb(v), edges[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: edges[u]) if (v != prev) {
            dfs(v, u);
            tmp.pb({depth[u],u});
        }
    }

    void construct() {
        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)];
    }
};
```

## 9.3 Advanced

### 9.3.1 Centroid Decomposition

```
/**
* Source: own
* Verification:
    https://codeforces.com/contest/342/problem/E
* Description: supports the following operations on a
    tree
 * making node red
 * querying distance to closest red node
*/

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

    // INITIALIZE

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

    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) {
        x = getCentroid(x); done[x] = 1;
        genDist(0,x,x,0);
        for (int i: adj[x]) if (!done[i]) solve(i);
    }

    void init() {
        FOR(i,1,N+1) ans[i] = MOD;
        solve(1);
    }

    // QUERY

    void upd(int v) {
        for (int V = v, ind = sz(dist[v])-1; V; V =
            cen[V], ind --)
            ans[V] = min(ans[V],dist[v][ind]);
    }

    int query(int v) {
        int ret = MOD;
        for (int V = v, ind = sz(dist[v])-1; V; V =
            cen[V], ind --)
            ret = min(ret,ans[V]+dist[v][ind]);
        return ret;
    }
};
```

### 9.3.2 Heavy-Light Decomposition

```
/**
* Source: http://codeforces.com/blog/entry/22072
* Dependency: Lazy SegTree
* Verification: USACO Grass Planting
*/

vector<vi> graph;

template <int V> struct HeavyLight { // sum queries,
    sum updates
    int parent[V], heavy[V], depth[V];
    int root[V], treePos[V];
    LazySegTree<V> 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) {
        processPath(u, v, [this, &value](int l, int r)
            { tree.upd(l, r, value); });
    }

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

HeavyLight<1<<17> H;
int N,M;

int main() {
        cin >> N >> M;
        graph.resize(N+1);
        FOR(i,N-1) {
            int a,b; cin >> a >> b;
            graph[a].pb(b), graph[b].pb(a);
        }
        H.init();
        FOR(i,M) {
            char c; int A,B;
            cin >> c >> A >> B;
            if (c == 'P') H.modifyPath(A,B,1);
            else cout << H.queryPath(A,B) << "\n";
        }
}
```

# 10 Math (4)

## 10.1 Number Theory

### 10.1.1 Fraction

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

struct Fraction {
    ll n,d;

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

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

bool operator<(const Fraction& lhs, const Fraction&
    rhs) {
    return lhs.n*rhs.d < rhs.n*lhs.d;
}

bool operator==(const Fraction& lhs, const Fraction&
    rhs) {
    return lhs.n == rhs.n && lhs.d == rhs.d;
}

bool operator!=(const Fraction& lhs, const Fraction&
    rhs) {
    return !(lhs == rhs);
}

Fraction operator+(const Fraction& lhs, const
    Fraction& rhs) {
    return
        Fraction(lhs.n*rhs.d+rhs.n*lhs.d,lhs.d*rhs.d);
}

Fraction operator+=(Fraction& lhs, const Fraction&
    rhs) {
    return lhs =
        Fraction(lhs.n*rhs.d+rhs.n*lhs.d,lhs.d*rhs.d);
}

Fraction operator-(const Fraction& lhs, const
    Fraction& rhs) {
    return
        Fraction(lhs.n*rhs.d-rhs.n*lhs.d,lhs.d*rhs.d);
}

Fraction operator-=(Fraction& lhs, const Fraction&
    rhs) {
    return lhs =
        Fraction(lhs.n*rhs.d-rhs.n*lhs.d,lhs.d*rhs.d);
}

Fraction operator*(const Fraction& lhs, const
    Fraction& rhs) {
    return Fraction(lhs.n*rhs.n,lhs.d*rhs.d);
}

Fraction operator*=(Fraction& lhs, const Fraction&
     rhs) {
    return lhs = Fraction(lhs.n*rhs.n,lhs.d*rhs.d);
}

Fraction operator*(int lhs, const Fraction& rhs) {
    return Fraction(lhs*rhs.n,rhs.d);
}

Fraction operator*(const Fraction& rhs, int lhs) {
    return Fraction(lhs*rhs.n,rhs.d);
}

Fraction operator/(const Fraction& lhs, const
     Fraction& rhs) {
    return Fraction(lhs.n*rhs.d, lhs.d*rhs.n);
}

std::ostream& operator<<(std::ostream &strm, const
     Fraction &a) {
    if (a.d == 1) {
        strm << a.n;
    } else {
        strm << a.n << "/" << a.d;
    }
    return strm;
}
```

### 10.1.2 Mod

```
/*
* Description: Basic operations with modular arithmetic
*/

ll po (ll b, ll p) { return
     !p?1:po(b*b%MOD,p/2)*(p&1?b:1)%MOD; }
ll inv (ll b) { return po(b,MOD-2); }

ll ad(ll a, ll b) { return (a+b)%MOD; }
ll sub(ll a, ll b) { return (a-b+MOD)%MOD; }
ll mul(ll a, ll b) { return a*b%MOD; }
ll divi(ll a, ll b) { return mul(a,inv(b)); }
```

### 10.1.3 NT

```
/**
 * Observation: 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
 * Verification: CF Power Tower
 */

namespace NT {
    vpi fac(int x) {
        vpi pri;

        for (int 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;
    }

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

    ll inv(ll a, ll b) { // 0 < a < b, gcd(a,b) = 1
        a %= b;
        if (a <= 1) return a;
        ll i = inv(b%a,a);
        ll tmp = -((b/a)*i+((b%a)*i)/a) % b;
        if (tmp < 0) tmp += b;
        return tmp;
    }

    pl CRT(pl a, pl b) { // Chinese Remainder Theorem,
         Verified by Kattis generalchineseremainder
        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*inv(A%B,B) % B;
        return {((mul*a.s+a.f)%l+l)%l,l};
    }
};
```

### 10.1.4 Prime Sieve

```
/**
* Source: KACTL
* https://open.kattis.com/problems/primesieve
*/

template<int SZ> struct Sieve {
    bitset<SZ+1> comp;
    Sieve() {
        for (int i = 2; i*i <= SZ; ++i) if (!comp[i])
            for (int j = i*i; j <= SZ; j += i) comp[j]
                 = 1;
    }
    bool isprime(int x) {
        if (x == 1) return 0;
        return !comp[x];
    }
};
```

### 10.1.5 bigint

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

#include <bits/stdc++.h>

using namespace std;

// 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
int main() {
    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);
}
```

## 10.2 Matrix

### 10.2.1 Matrix

```
/**
* Source: KACTL
* Verification: https://dmoj.ca/problem/si17c1p5
*/

struct mat {
    int** d;
    int a, b;

    mat() { a = b = 0; }

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

    mat (vector<vi> v) : mat(sz(v),sz(v[0])) {
        FOR(i,a) FOR(j,b) d[i][j] = v[i][j];
    }

    void print() {
        FOR(i,a) {
            FOR(j,b) cout << d[i][j] << " ";
            cout << "\n";
        }
        cout << "------------\n";
    }

    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]) % MOD;
        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] =
                (r.d[i][k]+(ll)d[i][j]*m.d[j][k]) % MOD;
        return r;
    }

    mat operator^(ll p) {
        mat r(a,a), base(*this);
        FOR(i,a) r.d[i][i] = 1;

        while (p) {
            if (p&1) r = r*base;
            base = base*base;
            p /= 2;
        }

        return r;
    }
};
```

### 10.2.2 Matrix Inverse (6)

```
/*
* Description: Calculates determinant mod a prime via
    gaussian elimination
* Verification: CF Stranger Trees
*/

ll po (ll b, ll p) { return
    !p?1:po(b*b%MOD,p/2)*(p&1?b:1)%MOD; }
ll inv (ll b) { return po(b,MOD-2); }

int ad(int a, int b) { return (a+b)%MOD; }
int sub(int a, int b) { return (a-b+MOD)%MOD; }
int mul(int a, int b) { return (ll)a*b%MOD; }

void elim(mat& m, int a, int c) { // column, todo row
    ll x = m.d[c][a];
    FOR(i,a,m.b) m.d[c][i] =
        sub(m.d[c][i],mul(x,m.d[a][i]));
}

int det(mat& x, bool b = 0) { // determinant of
    1000x1000 matrix in ~1s
    mat m = x;
    ll prod = 1;
    FOR(i,m.a) {
        bool done = 0;
        FOR(j,i,m.a) if (m.d[j][i] != 0) {
            done = 1; swap(m.d[j],m.d[i]);

            if ((j-i)&1) prod = mul(prod,MOD-1);
            prod = mul(prod,m.d[i][i]);

            ll x = inv(m.d[i][i]);
            FOR(k,i,m.b) m.d[i][k] = mul(m.d[i][k],x);

            if (b) {
                FOR(k,m.a) if (k != i) elim(m,i,k);
            } else {
                FOR(k,i+1,m.a) elim(m,i,k);
            }
            break;
        }
        if (!done) return 0;
    }
    if (b) x = m;
    return prod;
}
```

```
mat inv(mat m) {
    mat 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;

    det(x,1);

    mat 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;
}
```

## 10.3 Combinatorics (5)

### 10.3.1 Combo General

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

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

    ll mul(ll a, ll b) { return a*b%MOD; }
    ll po (ll b, ll p) { return
        !p?1:po(b*b%MOD,p/2)*(p&1?b:1)%MOD; }

    void init(ll _MOD) {
        MOD = _MOD;
        factors = NT::fac(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]), ifac[i] =
                NT::inv(fac[i],MOD);
        }
    }

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

### 10.3.2 Combo

```
/**
* Description: calculates combinations mod a large
    prime
* Verification: Combo General
*/

template<int SZ> struct Combo {
    int fac[SZ+1], ifac[SZ+1];

    ll mul(ll a, ll b) { return a*b%MOD; }
    ll po (ll b, ll p) { return
        !p?1:po(b*b%MOD,p/2)*(p&1?b:1)%MOD; }
    ll inv (ll b) { return po(b,MOD-2); }

    Combo() {
        fac[0] = ifac[0] = 1;
        FOR(i,1,SZ+1)
            fac[i] = mul(i,fac[i-1]), ifac[i] =
                inv(fac[i]);
    }

    ll comb(ll a, ll b) {
        if (a < b || b < 0) return 0;
        return mul(mul(fac[a],ifac[b]),ifac[a-b]);
    }
};
```

## 10.4 Polynomials (6)

### 10.4.1 Base Conversion

```
/**
* Description: NTT Application
* Verification: 2017 VT HSPC - Alien Codebreaking
*/

struct Base {
    vl po10[21];
    const int base = 27;

    Base() {
        po10[0] = {10};
        FOR(i,1,21) {
            po10[i] = NTT::conv(po10[i-1],po10[i-1]);
            normalize(po10[i]);
        }
    }

    void normalize(vl& x) {
        FOR(i,sz(x)) if (x[i] >= base) {
            if (i == sz(x)-1) x.pb(0);
            x[i+1] += x[i]/base;
            x[i] %= base;
        }
        while (sz(x) && !x.back()) x.pop_back();
    }
    vl convert(vl in) {
        if (sz(in) == 1) return in;
        vl l =
            convert(vl(in.begin(),in.begin()+sz(in)/2));
        vl r =
            convert(vl(in.begin()+sz(in)/2,in.end()));

        r = NTT::conv(r,po10[get(sz(in))-1]);
        normalize(r);

        int z = max(sz(l),sz(r));
        r.resize(z);
        FOR(i,sz(l)) r[i] += l[i];
        normalize(r);
        return r;
    }
};

Base B;

int main() {
        FOR(i,10) FOR(j,10) FOR(k,10) {
            vl z = {k,j,i};
            vl o = B.transform(z);
            for (ll x: o) cout << x << " ";
            cout << "\n";
        }
}
```

### 10.4.2 FFT Addition

```
/**
* Sources: KACTL, https://pastebin.com/3Tnj5mRu
* Verification: SPOJ polymul, CSA manhattan
*/

typedef vector<cd> vcd;

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

    vcd fft(vcd& a) {
        int n = sz(a), x = get(n);
        vcd res, RES(n), roots(n);
        FOR(i,n) roots[i] =
            cd(cos(2*M_PIl*i/n),sin(2*M_PIl*i/n));

        res = a;
        FOR(i,1,x+1) {
            int inc = n>>i;
            FOR(j,inc) for (int k = 0; k < n; k += inc)
                {
                int t = 2*k%n+j;
                RES[k+j] = res[t]+roots[k]*res[t+inc];
            }
            swap(res,RES);
        }

        return res;
    }

    vcd fft_rev(vcd& a) {
        vcd res = fft(a);
        FOR(i,sz(res)) res[i] /= sz(a);
        reverse(res.begin() + 1, res.end());
        return res;
    }

    vcd brute(vcd& a, vcd& b) {
        vcd 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);

        a.resize(n); a = fft(a);
        b.resize(n); b = fft(b);

        FOR(i,n) a[i] *= b[i];
        a = fft_rev(a);

        a.resize(s);
        return a;
    }

    vl convll(vl a, vl b) {
        vcd A(sz(a)); FOR(i,sz(a)) A[i] = a[i];
        vcd B(sz(b)); FOR(i,sz(b)) B[i] = b[i];
        vcd X = conv(A,B);
        vl x(sz(X)); FOR(i,sz(X)) x[i] =
            round(X[i].real());
        return x;
    }
}
```

### 10.4.3 FFT And

```
/**
* Description: Similar to FWHT
* Source: CSA - FFT And Variations
*/

typedef vector<double> vd;

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

    vd andConv(vd P, bool inv = 0) {
        for (int len = 1; 2 * len <= sz(P); len <<= 1)
            {
            for (int i = 0; i < sz(P); i += 2 * len) {
                for (int j = 0; j < len; j++) {
```

```
                double u = P[i + j];
                double v = P[i + len + j];

                if (!inv) {
                    P[i + j] = v;
                    P[i + len + j] = u + v;
                } else {
                    P[i + j] = -u + v;
                    P[i + len + j] = u;
                }
            }
        }
    }

    return P;
}

vd conv(vd a, vd b) {
    int s = max(sz(a),sz(b)), L = get(s), n = 1<<L;
    if (s <= 0) return {};

    a.resize(n); a = andConv(a);
    b.resize(n); b = andConv(b);

    FOR(i,n) a[i] = a[i]*b[i];
    a = andConv(a,1);
    return a;
}

vd orConv(vd a, vd b) {
    int s = max(sz(a),sz(b)), L = get(s), n = 1<<L;
    if (s <= 0) return {};

    a.resize(n); reverse(all(a)); a = andConv(a);
    b.resize(n); reverse(all(b)); b = andConv(b);

    FOR(i,n) a[i] = a[i]*b[i];
    a = andConv(a,1);
    reverse(all(a));

    return a;
}

vl orConv(vl a, vl b) {
    vd A; for (ll x: a) A.pb(x);
    vd B; for (ll x: b) B.pb(x);
    vd c = orConv(A,B);
    vl C; for (double x: c) C.pb(round(x));
    return C;
}

vl conv(vl a, vl b) {
    vd A; for (ll x: a) A.pb(x);
    vd B; for (ll x: b) B.pb(x);
    vd c = conv(A,B);
    vl C; for (double x: c) C.pb(round(x));
    return C;
}
}
```

### 10.4.4 FFT Demo

```
int main() {
    int N; cin >> N;
    vl a(N+1), b(N+1);
    FOR(j,N+1) cin >> a[N-j];
    FOR(j,N+1) cin >> b[N-j];
    vl x = FFT::convll(a,b);
    FORd(j,sz(x)) cout << x[j] << " ";
    cout << "\n";
}
```

### 10.4.5 FFT General Mod

```
/*
Description: Allows multiplication of polynomials in
    general moduli.
Verification:
    http://codeforces.com/contest/960/submission/37085144
*/

typedef vector<cd> vcd;

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

    void fft(vcd& a, bool inv){
        int n = sz(a), j = 0;
        vcd roots(n/2);
        FOR(i,1,n) {
                int bit = (n >> 1);
                while (j >= bit){
                        j -= bit;
                        bit >>= 1;
                }
                j += bit;
                if(i < j) swap(a[i], a[j]);
        }

        ld ang = 2 * M_PIl / n * (inv ? -1 : 1);
        FOR(i,n/2) roots[i] = cd(cos(ang * i), sin(ang
            * i));

        for (int i=2; i<=n; i<<=1){
                int step = n / i;
                for(int j=0; j<n; j+=i){
                        for(int k=0; k<i/2; k++){
                                cd u = a[j+k], v =
                                    a[j+k+i/2] *
                                    roots[step * k];
                                a[j+k] = u+v;
                                a[j+k+i/2] = u-v;
                        }
                }
        }

        if (inv) FOR(i,n) a[i] /= n;
    }

    vl conv(vl a, vl b, ll mod){
        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);
        fft(v1, 0); fft(v2, 0);

        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);
        }
        fft(r1, 1); fft(r2, 1);
        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] %= mod; ret[i] += mod; ret[i] %=
                    mod;
        }
        ret.resize(s);
        return ret;
    }
}
```

### 10.4.6 FFT XOR

```
/**
* Description: FWHT, similar to FFT
* Source: CSA - FFT And Variations
* Verification: HackerRank XOR Subsequence
*/

typedef vector<double> vd;

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

    vd fwht(vd P) {
        for (int len = 1; 2 * len <= sz(P); len <<= 1)
            {
            for (int i = 0; i < sz(P); i += 2 * len) {
                for (int j = 0; j < len; j++) {
                    double u = P[i + j];
                    double v = P[i + len + j];
                    P[i + j] = u+v;
                    P[i + len + j] = u-v;
                }
            }
        }

        return P;
    }

    vd fwht_rev(vd& a) {
        vd res = fwht(a);
        FOR(i,sz(res)) res[i] /= sz(a);
        return res;
    }

    vd conv(vd a, vd b) {
        int s = max(sz(a),sz(b)), L = get(s), n = 1<<L;
        if (s <= 0) return {};

        a.resize(n); a = fwht(a);
        b.resize(n); b = fwht(b);

        FOR(i,n) a[i] = a[i]*b[i];
        a = fwht_rev(a);
        return a;
    }

    vl conv(vl a, vl b) {
        vd A; for (ll x: a) A.pb(x);
        vd B; for (ll x: b) B.pb(x);
        vd c = conv(A,B);
        vl C; for (double x: c) C.pb(round(x));
        return C;
    }
}
```

### 10.4.7 Lagrange Interpolation

```
namespace Poly {
    ll norm(ll x) { return (x%MOD+MOD)%MOD; }

    vl operator+(vl a, vl b) {
        a.resize(max(sz(a),sz(b)));
        FOR(i,sz(a)) a[i] = norm(a[i]+b[i]);
        return a;
    }

    vl operator*(vl a, vl b) {
        vl x(sz(a)+sz(b)-1);
        FOR(i,sz(a)) FOR(j,sz(b)) x[i+j] =
            norm(x[i+j]+a[i]*b[j]);
        return x;
    }
    vl operator*(vl a, ll b) {
        for (ll& i: a) i = norm(i*b);
        return a;
    }

    vl interpolate(vector<pl> v) {
        vl ret;
        FOR(i,sz(v)) {
            vl prod = {1};
            ll todiv = 1;
            FOR(j,sz(v)) if (i != j) {
                todiv = norm(todiv*(v[i].f-v[j].f));
                vl tmp = {norm(-v[j].f),1};
                prod = prod*tmp;
            }
            prod = prod*inv(todiv);
            prod = prod*v[i].s;
            ret = ret+prod;
        }
        return ret;
    }

    void prin(vl x) {
        for (ll i: x) cout << i << " ";
        cout << "\n";
    }
}
```

#### 10.4.8 NTT

```
/**
* Description: Use if you are working with
    non-negative integers
* Verification:
    http://codeforces.com/contest/632/submission/33953285
*/


namespace NTT {
    const ll 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.

    ll modpow(ll b, ll p) { return
        !p?1:modpow(b*b%mod,p/2)*(p&1?b:1)%mod; }

    ll inv (ll b) { return modpow(b,mod-2); }

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

    vl ntt(vl& a) {
        int n = a.size(), x = get(n);
        vl res, RES(n), roots(n);
        roots[0] = 1, roots[1] =
            modpow(root,(mod-1)/n);
        FOR(i,2,n) roots[i] = roots[i-1]*roots[1] %
            mod;

        res = a;
        FOR(i,1,x+1) {
            int inc = n>>i;
            FOR(j,inc) for (int k = 0; k < n; k += inc)
                {
                int t = 2*k%n+j;
                RES[k+j] = (res[t]+roots[k]*res[t+inc])
                    % mod;
            }
            swap(res,RES);
        }

        return res;
    }

    vl ntt_rev(vl& a) {
        vl res = ntt(a);
        ll in = inv(sz(a));
        FOR(i,sz(res)) res[i] = res[i]*in % mod;
        reverse(res.begin() + 1, res.end());
        return res;
    }

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

    vl conv(vl a, vl 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);

        a.resize(n); a = ntt(a);
        b.resize(n); b = ntt(b);

        FOR(i,n) a[i] = a[i]*b[i] % mod;
        a = ntt_rev(a);

        a.resize(s);
        return a;
    }
}
```

# 11 Graphs Hard (4)

## 11.1 SCC

### 11.1.1 2SAT

```
/*
* Verification: https://www.spoj.com/problems/BUGLIFE/
* Also useful: at most one
```

```
*
    (http://codeforces.com/contest/1007/submission/40284510)
*/

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 Edmonds-Karp

```
/**
* Source: GeeksForGeeks
*/

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

template<int SZ> struct EdmondsKarp {
    pi pre[SZ];
    int SC, SNC;
    ll flow[SZ];
    vector<Edge> adj[SZ];

    void addEdge(int u, int v, int 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);
    }

    bool bfs() {
        memset(flow,0,sizeof flow);
        flow[SC] = INF;

        queue<int> todo; todo.push(SC);
        while (todo.size()) {
            if (flow[SNC]) break;
            int x = todo.front(); todo.pop();
            for (auto a: adj[x]) if (!flow[a.v] &&
                a.flow < a.C) {
                pre[a.v] = {x,a.rev};
                flow[a.v] = min(flow[x],a.C-a.flow);
                todo.push(a.v);
            }
        }

        return flow[SNC];
    }

    ll maxFlow(int sc, int snc) {
        SC = sc, SNC = snc;

        ll ans = 0;
        while (bfs()) {
            ans += flow[SNC];
            for (int x = SNC; x != SC; x = pre[x].f) {
                adj[x][pre[x].s].flow -= flow[SNC];
                int t = adj[x][pre[x].s].rev;
                adj[pre[x].f][t].flow += flow[SNC];
            }
        }

        return ans;
    }
};
```

### 11.2.2 Flows Demo

```
/**
* Link: http://www.spoj.com/problems/FASTFLOW/
* Use with Dinic, Push-Relabel, Edmonds-Karp
*/

int N,M;
PushRelabel<5001> D;

int main() {
    cin >> N >> M;
    FOR(i,M) {
        int a,b,c; cin >> a >> b >> c;
        D.addEdge(a,b,c);
        D.addEdge(b,a,c);
    }
    cout << D.maxFlow(1,N);
}
```

### 11.2.3 Dinic (5)

```
/**
* Source: GeeksForGeeks
* Verification: Problem Fashion (RMI 2017 Day 1)
* Code: https://pastebin.com/VJxTvEg1
*/

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

template<int SZ> struct Dinic {
    int level[SZ], start[SZ];
    vector<Edge> adj[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);
    }

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

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

        return level[t] >= 0;
    }

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

        for ( ; start[u] < sz(adj[u]); start[u] ++) {
            Edge &e = adj[u][start[u]];

            if (level[e.v] == level[u]+1 && e.flow <
                 e.C) {
                ll curr_flow = min(flow, e.C - e.flow);
                ll temp_flow = sendFlow(e.v, curr_flow,
                     t);

                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)) {
            FOR(i,SZ) start[i] = 0;
            while (ll flow = sendFlow(s, INT_MAX, t))
                 total += flow;
        }

        return total;
    }
};
```

### 11.2.4 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];
    }
};
```

### 11.2.5 MinCostFlow (6)

```
/**
* Source: GeeksForGeeks
*/

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();
        }
    }
};

mcf<100> m;

int main() {
    m.addEdge(0, 1, 16, 5);
    m.addEdge(1, 2, 13, 7);
    m.addEdge(1, 2, 13, 8);

    pi x = m.mincostflow(0,2);
    cout << x.f << " " << x.s;
}
```

## 11.3 Tarjan BCC

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

template<int SZ> struct BCC {
    int N, ti = 0;
    vi adj[SZ];
    int disc[SZ], low[SZ], comp[SZ], par[SZ];
    vector<vpi> fin;
    vpi st;

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

    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() {
        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: extra log factor
* Verification:
    https://open.kattis.com/problems/eulerianpath
*/

struct Euler {
    vi circuit;
    multiset<int> adj[MX], ADJ[MX];
    int N,M, out[MX], in[MX];

    void find_circuit(int x) { // directed graph,
        possible that resulting circuit is not valid
        while (sz(adj[x])) {
            int j = *adj[x].begin();
                adj[x].erase(adj[x].begin());
            find_circuit(j);
        }
        circuit.pb(x);
    }

    int a,b,start;

    vi solve() {
        FOR(i,N) {
            adj[i].clear(), ADJ[i].clear();
            out[i] = in[i] = 0;
        }
        circuit.clear();
        FOR(i,M) {
            cin >> a >> b; // add edges
            adj[a].insert(b), ADJ[a].insert(b);
            out[a] ++, in[b] ++;
        }
        start = a;
        FOR(i,N) if (out[i]-in[i] == 1) start = i;

        find_circuit(start);
        reverse(all(circuit));

        if (sz(circuit) != M+1) return {};

        FOR(i,M) { // verify that circuit is valid
            if (ADJ[circuit[i]].find(circuit[i+1]) ==
                ADJ[circuit[i]].end()) return {};
            int t = circuit[i];
            ADJ[t].erase(ADJ[t].find(circuit[i+1]));
        }

        return circuit;
    }
};
```

# 12 Geometry (4)

## 12.1 Techniques

### 12.1.1 Complex Operators

```
/**
* Description: Easy Geo
* Source: http://codeforces.com/blog/entry/22175
*/

template<class T> istream& operator>> (istream& is,
    complex<T>& p) {
    T value;
    is >> value; p.real(value);
    is >> value; p.imag(value);
    return is;
}

bool operator<(const cd& a, const cd& b) {
    if (a.real() != b.real()) return a.real() <
        b.real();
    return a.imag() < b.imag();
}
bool operator>(const cd& a, const cd& b) {
    if (a.real() != b.real()) return a.real() >
        b.real();
    return a.imag() > b.imag();
}
bool operator<=(const cd& a, const cd& b) { return a <
    b || a == b; }
bool operator>=(const cd& a, const cd& b) { return a >
    b || a == b; }
cd max(const cd& a, const cd& b) { return a>b?a:b; }
cd min(const cd& a, const cd& b) { return a<b?a:b; }

ld cross(cd a, cd b) { return (conj(a)*b).imag(); }
ld area(cd a, cd b, cd c) { return cross(b-a,c-a); }
ld dot(cd a, cd b) { return (conj(a)*b).real(); }

cd reflect(cd p, cd a, cd b) { return
    a+conj((p-a)/(b-a))*(b-a); }
cd proj(cd p, cd a, cd b) { return
    (p+reflect(p,a,b))/(ld)2; }

cd line(cd a, cd b, cd c, cd d) {
    ld x = area(a,b,c), y = area(a,b,d);
    return (x*d-y*c)/(x-y);
}

vcd segment(cd A, cd B, cd C, cd D) { // kattis
    segmentintersection
    if (A > B) swap(A,B);
    if (C > D) swap(C,D);

    ld a1 = area(A,B,C), a2 = area(A,B,D);
    if (a1 > a2) swap(a1,a2);
    if (!(a1 <= 0 && a2 >= 0)) return {};

    if (a1 == 0 && a2 == 0) {
        if (area(A,C,D) != 0) return {};
        cd x1 = max(A,C), x2 = min(B,D);
        if (x1 > x2) return {};
        if (x1 == x2) return {x1};
        return {x1,x2};
    }

    cd z = line(A,B,C,D);
    if (A <= z && z <= B) return {z};
    return {};
}
```

### 12.1.2 Polygon Area

```
/**
* Description: Shoelace Formula
* Verification:
    https://open.kattis.com/problems/polygonarea
*/

ld area(vector<cd> v) {
    ld x = 0;
    FOR(i,sz(v)) {
        int j = (i+1)%sz(v);
        x += (ld)v[i].real()*v[j].imag();
        x -= (ld)v[j].real()*v[i].imag();
    }
    return abs(x)/2;
}
```

### 12.1.3 Point in Polygon (5)

```
/**
* Source: own
* Verification:
    https://open.kattis.com/problems/pointinpolygon
*/

int n,m;
pi p[1000];

int area(pi x, pi y, pi z) {
    return (y.f-x.f)*(z.s-x.s)-(y.s-x.s)*(z.f-x.f);
}

bool on(pi x, pi y, pi z) {
    if (area(x,y,z) != 0) return 0;
    return min(x,y) <= z && z <= max(x,y);
}

double get(pi x, pi y, int z) {
    return double((z-x.s)*y.f+(y.s-z)*x.f)/(y.s-x.s);
}

string test(pi z) {
    int ans = 0;
    FOR(i,n) {
        pi x = p[i], y = p[(i+1)%n];
        if (on(x,y,z)) return "on";
        if (x.s > y.s) swap(x,y);
        if (x.s <= z.s && y.s > z.s) {
            double t = get(x,y,z.s);
            if (t > z.f) ans++;
        }
    }
    if (ans % 2 == 1) return "in";
    return "out";
}
```

### 12.1.4 3D Geometry (6)

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

int n;
vector<vl> cur;

vl operator-(vl a, vl b) {
    vl c(sz(a)); FOR(i,sz(a)) c[i] = a[i]-b[i];
    return c;
}

bool ismult(vl b, vl c) {
    if ((ld)b[0]*c[1] != (ld)b[1]*c[0]) return 0;
    if ((ld)b[0]*c[2] != (ld)b[2]*c[0]) return 0;
    if ((ld)b[2]*c[1] != (ld)b[1]*c[2]) return 0;
    return 1;
}

bool collinear(vl a, vl b, vl c) {
    b = b-a, c = c-a;
    return ismult(b,c);
}

vl cross(vl a, vl 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 coplanar(vl a, vl b, vl c, vl d) {
    b = b-a, c = c-a, d = d-a;
    return ismult(cross(b,c),cross(b,d));
}
```

### 12.1.5 Circles (6)

```
/**
* Source: Own
* Verification:
    https://codefights.com/tournaments/s8thqrnQL2YPK7XQt/L
*/

typedef pair<cd,ld> circle;

cd intersect(circle a, circle b, int x = 0) {
    ld d = sqrt(norm(a.f-b.f));
    ld co = (a.s*a.s+d*d-b.s*b.s)/(2*a.s*d);
    ld theta = acos(co);

    cd tmp = (b.f-a.f)/d;
    if (x == 0) return
        a.f+tmp*a.s*polar((ld)1.0,theta);
    return a.f+tmp*a.s*polar((ld)1.0,-theta);
}

ld arc(circle x, cd a, cd b) {
    cd d = (a-x.f)/(b-x.f);
    return x.s*acos(d.real());
}

bool on (circle x, cd y) {
    return norm(y-x.f) == x.s*x.s;
}
```

## 12.2 Sweep Line

### 12.2.1 Convex Hull

```
/**
* Source: Wikibooks
* Verification:
    https://open.kattis.com/problems/convexhull
*/

ll cross(pi O, pi A, pi B) {
    return (ll)(A.f-O.f)*(B.s-O.s)
           -(ll)(A.s-O.s)*(B.f-O.f);
}

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

    vpi 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();

    vpi 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 Max Rectangle

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

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

void solve(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
        solve(j);
    }
    return ans;
}
```

### 12.2.3 Closest Pair (6)

```
/**
* Source: GeeksForGeeks
* Description: Nlog^2N, can be improved
* Usage: https://open.kattis.com/problems/closestpair2
*/

pair<double,pair<pd,pd>> MN = {INF,{{0,0},{0,0}}};

int n;

bool cmp(pd a, pd b) {
    return a.s < b.s;
}

double dist(pd a, pd b) {
    b.f -= a.f, b.s -= a.s;
    return sqrt(b.f*b.f+b.s*b.s);
}

pair<double,pair<pd,pd>> strip(vector<pd> v, double
     di) {
    pair<double,pair<pd,pd>> ans = MN;
    FOR(i,sz(v)) FOR(j,i+1,sz(v)) {
        if (v[i].s+di <= v[j].s) break;
        ans = min(ans,{dist(v[i],v[j]),{v[i],v[j]}});
    }
    return ans;
}

pair<double,pair<pd,pd>> bes (vector<pd> v) {
    if (v.size() == 1) return MN;
    int M = v.size()/2;
    vector<pd> v1(v.begin(),v.begin()+M),
        v2(v.begin()+M,v.end());
    auto a = bes(v1), b = bes(v2);
    double di = min(a.f,b.f);

    vector<pd> V;
    FOR(i,v.size()) if (v[i].f > v[M].f-di && v[i].f <
        v[M].f+di) V.pb(v[i]);
    sort(V.begin(),V.end(),cmp);

    auto z = strip(V,di);
    return min(min(a,b),z);
}

int main() {
        cout << fixed << setprecision(2);
        while (cin >> n) {
            if (n == 0) break;
            vector<pd> v(n);
            FOR(i,n) cin >> v[i].f >> v[i].s;
            sort(all(v));
            auto a = bes(v);
            cout << a.s.f.f << " " << a.s.f.s << " " <<
                 a.s.s.f << " " << a.s.s.s << "\n";
        }
}
```

### 12.2.4 LineContainer (6)

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

```
        }
};
```

## 12.3 Delaunay

```
/*
* Bowyer-Watson O(n^2logn)
* Verification: 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";
    }
};
```

## 12.4 Max Collinear

```
/**
* 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";
}
```

# 13 Additional (4)

## 13.1 Mo

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

int N, Q, A[MX], ans[MX], oc[MX];
vector<array<int,3>> todo;

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

## 13.2 Misc

### 13.2.1 Connectivity

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

int n,m,q; // vertices, edges, # queries
pi p[MX]; // connectivity queries
int l[MX],r[MX];
vi tri[MX];
vpi ed; // edges

bool left() {
    FOR(i,sz(ed)) 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<MX> D = DSU<MX>();
    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();
}
```

### 13.2.2 Discrete Logarithm

```
/**
 * Description: find k such that primitive^k=x
 * meet in the middle, O(sqrt(MOD))
 * Source: Own
 * Verification: PA 2006 - Professor Laugh's Numbers
 */

const int BLOCK = 32000;

int primitive = 5, invy[BLOCK];
unordered_map<int,int> u;

ll po (ll b, ll p) {
    return !p?1:po(b*b%MOD,p/2)*(p&1?b:1)%MOD;
}

ll inv (ll b) { return po(b,MOD-2); }

ll query(int x) {
        FOR(i,BLOCK) if (u.count(x*invy[i]%MOD))
            return i*BLOCK+u[x*invy[i]%MOD];
        return -1;
}

int main() {
    ll cur = 1;
        FOR(i,BLOCK) {
            u[cur] = i;
            cur = primitive*cur%MOD;
        }
        ll t = 1;
        FOR(i,BLOCK) {
            invy[i] = inv(t);
            t = t*cur%MOD;
        }
        ll x; cin >> x;
        cout << query(x) << "\n";
}
```

## 13.3 Pragma Optimization (6)

```
/**
 * Source: Misc solutions to CF Nagini
 * Description: 10^{10} operations are ok!
 * Passes the occasional disgusting CF task
 * Also see "Welcome home, Chtholly"
 */

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

int q, mx[MX], mn[MX];

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin >> q;
    FOR(i,MX) mx[i] = -MOD, mn[i] = MOD;
    FOR(i,q) {
        int t,l,r,k; cin >> t >> l >> r;
        r -= l;

        auto a = mx+l, b = mn+l;
        if (t == 1) {
            cin >> k;
            if (k > 0) FOR(j,r) b[j] = min(b[j],k);
            else FOR(j,r) a[j] = max(a[j],k);
        } else {
            ll ans = 0;
            FOR(j,r) if (a[j] != -MOD && b[j] != MOD)
                ans += b[j]-a[j];
            cout << ans << "\n";
        }
    }
}
```