

USACO Notebook

Benq

December 23, 2017

Contents

1	(0) Contest	2
1.1	C++ Template	2
1.2	FastScanner	2
1.3	Pragma	3
1.4	Troubleshoot	3
2	(2) Data Structures	4
2.1	(2) Coordinate Compression	4
2.2	(2) Prefix Sums	4
2.3	(2) STL Demo	4
2.4	(3) 1D Range Queries	4
2.4.1	(3) BIT	4
2.4.2	(3) RMQ	5
2.4.3	(3) SegTree	5
2.4.4	(4) BIT with Range Update	5
2.4.5	(4) Lazy SegTree	5
2.4.6	(5) Lazy Persistent SegTree	6
2.5	(4) 2D Range Queries	7
2.5.1	(4) 2D BIT	7
2.5.2	(4) 2D Sparse SegTree	8
2.5.3	(4) Merge-Sort Tree	9
2.6	(4) BBST	9
2.6.1	(4) Treap	9
2.6.2	(5) Link-Cut Tree	10
2.6.3	(5) Splay Tree	11
3	(2) Graphs	12
3.1	(2) Searching Demo	12
3.1.1	(2) BFS on Grid	12
3.1.2	(2) DFS on Graph	12
3.1.3	(2) Tree Diameter	12
3.2	(3) Shortest Path	12
3.2.1	(3) Bellman-Ford	12
3.2.2	(3) Dijkstra	13
3.2.3	(3) Floyd-Warshall	13
3.3	(3) Topological Sort	13
3.4	(4) Biconnected Components	14
3.5	(4) Kosaraju	14
3.6	(6) Euler Tour	15
4	(2) Paradigms	15
4.1	(2) Interval Cover	15
4.2	(4) Discrete Logarithm	16
4.3	(4) Ternary Search	16

5	(3) Dynamic Programming	16
5.1	(3) Distinct Subsequences	16
5.2	(3) LCS	16
5.3	(3) LIS	17
5.4	(4) TSP	17
6	(3) Strings	17
6.1	(3) Hashing	17
6.2	(4) String Searching	18
6.2.1	(4) Aho-Corasick	18
6.2.2	(4) Bitset Trie	18
6.2.3	(4) Z	19
6.3	(4) Suffix Array	19
6.4	(5) Manacher	20
7	(3) Trees	20
7.1	(3) DSU, Kruskal	20
7.2	(4) Tree Queries	20
7.2.1	(4) Centroid Decomposition	20
7.2.2	(4) HLD	21
7.2.3	(4) LCA with Binary Jumps	22
7.2.4	(4) LCA with RMQ	22
8	(4) Flows	23
8.1	(5) Dinic	23
8.2	(5) Push-Relabel	23
8.3	(6) MinCostFlow	24
9	(4) Geometry	25
9.1	(4) Convex Hull	25
9.1.1	(4) Convex Hull	25
9.1.2	(4) LiChao Segment Tree	25
9.1.3	(4) LineContainer	26
9.2	(4) Misc	26
9.2.1	(4) Circles	26
9.2.2	(4) Closest Pair	27
9.2.3	(4) Line Segment Intersection	27
9.2.4	(4) MaxCollinear	28
9.2.5	(4) Pair Operators	28
9.2.6	(4) Point in Polygon	29
9.2.7	(4) Polygon Area	29
9.3	(6) KD Tree	29

10 (4) Math	30
10.1 (4) Matrix	30
10.1.1 (4) Matrix Exponentiation	30
10.1.2 (6) Linear Equation Solver	31
10.2 (4) Number Theory	31
10.2.1 (4) Eratosthenes' Sieve	31
10.2.2 (5) Chinese Remainder Theorem	31
10.2.3 (5) Combinations	32
10.2.4 (5) Phi	33
10.3 (6) Polynomials	33
10.3.1 (6) Base Conversion	33
10.3.2 (6) FFT, NTT	33
11 (6) Sqrt Decomposition	35
11.1 (6) Mo	35

1 (0) Contest

1.1 C++ Template

```

/**
 * Sources: various
 */

#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 vector<int> vi;
typedef pair<int, int> pii;
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 FOR(i, a) for (int i=0; i<(a); i++)
#define FORd(i,a,b) for (int i = (b)-1; i >= a; i--)
#define FORd(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;

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

```

```

// read!read!read!read!read!read!read!
// ll vs. int!

```

1.2 FastScanner

```

/**
 * 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 (!isspace(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 Pragma

```

/**
 * Source: CF Nagini
 */

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

// template

int q, mx[100001], mn[100001];

int main() {
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    cin >> q;
    FOR(i,100001) mx[i] = -MOD, mn[i] = MOD;
    FOR(i,q) {
        int t,l,r,k; cin >> t >> l >> r;
        r -= 1;

        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";
        }
    }
}

```

1.4 Troubleshoot

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 (2) Data Structures

2.1 (2) Coordinate Compression

```
/**
 * Source: own
 */

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

// vi x = {2,4,3,6}; compress(x);
```

2.2 (2) Prefix Sums

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

struct sums {
    ll sum[1001][1001];
    sums () { memset(sum,0,sizeof sum); }
    void init() {
        FOR(i,1,1001) FOR(j,1,1001)
            sum[i][j] +=
                sum[i][j-1]+sum[i-1][j]-sum[i-1][j-1];
    }
    ll 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];
    }
};
```

2.3 (2) STL Demo

```
/**
 * Source: StackOverflow
 */

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

struct hsh {
    size_t operator()(const pii& k) const {
        return
            hash<int>()(k.f)^(hash<int>()(k.s^293849182));
    }
};

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

2.4 (3) 1D Range Queries

2.4.1 (3) BIT

```
/**
 * Source: own
 * Description: 1D point update, range query
 */

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]
};

// BIT<int,1<<17> b;
```

2.4.2 (3) RMQ

```
/**
 * Source: own
 * Description: 1D range query
 */

template<class T, int SZ> struct RMQ {
    T stor[SZ][31-__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,1,31-__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]);
    }
};

// RMQ<int,100000> R;
```

2.4.3 (3) SegTree

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

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],seg[p^1]);
    }

    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 res = MN; r++;
```

```
        for (l += SZ, r += SZ; l < r; l >>= 1, r >>=
            1) {
            if (l&1) res = comb(res,seg[l++]);
            if (r&1) res = comb(res,seg[--r]);
        }
        return res;
    }
};

// Seg<int,1<<17> b;
```

2.4.4 (4) BIT with Range Update

```
/**
 * Source: own
 * Description: 1D range update, range query
 */

template<int SZ> struct BITrange {
    ll bit[2][SZ+1]; // sums piecewise linear functions

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

    void u(int ind, int hi, ll val) {
        for(;hi <= SZ; hi += (hi&-hi)) bit[ind][hi] +=
            val;
    }

    void upd(int hi, int val) {
        u(1,1,val), u(1,hi+1,-val);
        u(0,hi+1,hi*val);
    }

    void upd(int lo, int hi, ll val) { upd(lo-1,-val),
        upd(hi,val); }

    ll qsum(int x) {
        ll c1 = 0, c0 = 0;
        for (int x1 = x; x1 > 0; x1 -= (x1&-x1))
            c1 += bit[1][x1], c0 += bit[0][x1];
        return c1*x+c0;
    }

    ll qsum(int x, int y) { return qsum(y)-qsum(x-1); }
};

// BITrange<1<<17> b;
```

2.4.5 (4) Lazy SegTree

```
/**
 * Source: USACO Counting Haybales
 * 1D range update, range query
 */

const ll INF = 1e18;

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);
}
};

```

```
// LazySegTree<ll,1<<17> b;
```

2.4.6 (5) Lazy Persistent SegTree

```

/**
 * Source:
 * http://codeforces.com/blog/entry/47108?#comment-315047
 */

struct Node { // without lazy updates
    int val = 0;
    Node* c[2];

    Node* copy() {
        Node* x = new Node(); *x = *this;
        return x;
    }

    int query(int low, int high, int L, int R) {
        if (low <= L && R <= high) return val;
        if (R < low || high < L) return MOD;
        int M = (L+R)/2;
        return min(c[0]->query(low,high,L,M),
            c[1]->query(low,high,M+1,R));
    }

    Node* upd(int ind, int v, int L, int R) {
        if (R < ind || ind < L) return this;
        Node* x = copy();

        if (ind <= L && R <= ind) {
            x->val += v;
            return x;
        }

        int M = (L+R)/2;
        x->c[0] = x->c[0]->upd(ind,v,L,M);
        x->c[1] = x->c[1]->upd(ind,v,M+1,R);
        x->val = min(x->c[0]->val,x->c[1]->val);

        return x;
    }

    void build(vi& arr, int L, int R) {
        if (L == R) {
            if (L < (int)arr.size()) val = arr[L];
            else val = 0;
            return;
        }
        int M = (L+R)/2;
        c[0] = new Node();
        c[0]->build(arr,L,M);
        c[1] = new Node();
        c[1]->build(arr,M+1,R);
        val = min(c[0]->val,c[1]->val);
    }
};

struct node { // with lazy updates

```

```

int val = 0, lazy = 0;
node* c[2];

node* copy() {
    node* x = new node(); *x = *this;
    return x;
}

void push() {
    if (!lazy) return;
    FOR(i,2) if (c[i]) {
        c[i] = new node(*c[i]);
        c[i]->lazy += lazy;
    }
    lazy = 0;
}

int query(int low, int high, int L, int R) {
    if (low <= L && R <= high) return val;
    if (R < low || high < L) return MOD;
    int M = (L+R)/2;
    return lazy+min(c[0]->query(low,high,L,M),
        c[1]->query(low,high,M+1,R));
}

node* upd(int low, int high, int v, int L, int R) {
    if (R < low || high < L) return this;
    node* x = copy();
    if (low <= L && R <= high) {
        x->lazy += v, x->val += v;
        return x;
    }
    push();

    int M = (L+R)/2;
    x->c[0] = x->c[0]->upd(low,high,v,L,M);
    x->c[1] = x->c[1]->upd(low,high,v,M+1,R);
    x->val = min(x->c[0]->val,x->c[1]->val);

    return x;
}

void build(vi& arr, int L, int R) {
    if (L == R) {
        if (L < sz(arr)) val = arr[L];
        else val = 0;
        return;
    }
    int M = (L+R)/2;
    c[0] = new node();
    c[0]->build(arr,L,M);
    c[1] = new node();
    c[1]->build(arr,M+1,R);
    val = min(c[0]->val,c[1]->val);
}

};

template<int SZ> struct pers {
    node* loc[SZ+1]; // stores location of root after
        ith update
    int nex = 1;

```

```

pers() { loc[0] = new node(); }

void upd(int low, int high, int val) {
    loc[nex] =
        loc[nex-1]->upd(low,high,val,0,SZ-1);
    nex++;
}

void build(vi& arr) {
    loc[0]->build(arr,0,SZ-1);
}

int query(int ti, int low, int high) {
    return loc[ti]->query(low,high,0,SZ-1);
}

};

pers<8> p;

int main() {
    vi arr = {1,7,2,3,5,9,4,6};
    p.build(arr);

    p.upd(1,2,2); // 1 9 4 3 5 9 4 6

    FOR(i,8) {
        FOR(j,i,8) cout << p.query(1,i,j) << " ";
        cout << "\n";
    }
    cout << "\n";

    p.upd(4,7,5); // 1 9 4 3 10 14 9 11
    FOR(i,8) {
        FOR(j,i,8) cout << p.query(2,i,j) << " ";
        cout << "\n";
    }
    cout << "\n";

    FOR(i,8) {
        FOR(j,i,8) cout << p.query(1,i,j) << " ";
        cout << "\n";
    }
    cout << "\n";
}

```

2.5 (4) 2D Range Queries

2.5.1 (4) 2D BIT

```

/**
 * Source: own
 */

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

```

```

        for (; X > 0; X -= (X&-X))
            for (int Y1 = Y; Y1 > 0; Y1 -= (Y1&-Y1))
                ans += bit[X][Y1];
        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);
    }
};

// BIT2D<int,1000> b;

```

2.5.2 (4) 2D Sparse SegTree

```

/**
 * Source: USACO Mowing the Field
 * Description: 2D Point Update, Range Query
 */

const int SZ = 1<<17;

// Sparse 1D SegTree
struct node {
    int val = 0;
    node* c[2];

    void upd(int ind, int v, int L = 0, int R = SZ-1)
    { // set an element equal to 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;
    }

    int 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 = 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;
    }
};

// 2D SegTree, sparse segtree of sparse 1D segtrees
struct Node {
    node seg;

```

```

    Node* c[2];

    void upd(int x, int y, int v, int L = 0, int R =
        SZ-1) { // set an element equal to v
        seg.upd(y,v);
        if (L == x && R == x) 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);
        }
    }

    int 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 = 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;
    }
};

// SegTree + BIT
// Array of Sparse Segtrees
struct SegBit {
    node seg[SZ+1];

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

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

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

Node n;
SegBit s;

int main() {
    n.upd(5,7,2);
    n.upd(3,2,20);
    n.upd(5,8,200);
    cout << n.query(3,5,2,7) << "\n"; // 22

```



```

s.upd(5,7,2);
s.upd(3,2,20);
s.upd(5,8,200);
cout << s.query(3,5,2,7) << "\n"; // 22
}

```

2.5.3 (4) Merge-Sort Tree

```

/**
 * Source: own
 */

template<int SZ> struct mstree {
    Tree<pii> 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) 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);
    }
};

int main() {
    mstree<100000> m;
    m.upd(3,6); m.upd(4,5);
    cout << m.query(3,5,4,6) << " " <<
        m.query(3,5,4,5); // 2, 1
}

```

2.6 (4) BBST

2.6.1 (4) Treap

```

/**
 * Source: own?
 * Description: Easiest BBST
 * Note: Also see lazy persistent treap.
 */

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

```

```

tnode (int v) {
    val = v, 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";
}

};

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;
        return {p.f, t};
    } else {
        auto p = split(t->c[1], v); t->c[1] = p.f;
        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);
        return l;
    } else {
        r->c[0] = merge(l,r->c[0]);
        return r;
    }
}

tnode* ins(tnode* x, int v) { // insert value v
    auto a = split(x,v);
    return merge(merge(a.f, new tnode(v)),a.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 main() {
    root = ins(root,1);
    root = ins(root,9);
    root = ins(root,3);

    root->inOrder(1);

    root = ins(root,7);
    root = ins(root,4);
}

```

```

    root = del(root,9);

    root->inOrder(1);
}

```

2.6.2 (5) Link-Cut Tree

```

/**
 * Source: Dhruv Rohatgi
 */

int p[100001], pp[100001], c[100001][2], sum[100001];

int getDir(int x, int y) {
    return c[x][0] == y ? 0 : 1;
}

void setLink(int x, int y, int d) {
    c[x][d] = y, p[y] = x;
}

void rotate(int y, int d) {
    int x = c[y][d], z = p[y];
    setLink(y, c[x][d^1], d);
    setLink(x, y, d^1);
    setLink(z, x, getDir(z, y));

    sum[x] = sum[y];
    sum[y] = sum[c[y][0]] + sum[c[y][1]] + 1;
    pp[x] = pp[y]; pp[y] = 0;
}

void splay(int x) {
    while (p[x]) {
        int y = p[x], z = p[y];
        int dy = getDir(y, x), dz = getDir(z, y);
        if (!z) rotate(y, dy);
        else if (dy == dz) rotate(z, dz), rotate(y, dy);
        else rotate(y, dy), rotate(z, dz);
    }
}

void dis(int v, int d) {
    p[c[v][d]] = 0, pp[c[v][d]] = v;
    sum[v] -= sum[c[v][d]];
    c[v][d] = 0;
}

void con(int v, int d) {
    c[pp[v]][d] = v;
    sum[pp[v]] += sum[v];
    p[v] = pp[v], pp[v] = 0;
}

void access(int v) {
    // v is brought to the root of auxiliary tree
    // modify preferred paths

    splay(v);
    dis(v, 1);
}

```

```

while (pp[v]) {
    int w = pp[v]; splay(w);
    dis(w, 1), con(v, 1);
    splay(v);
}

int find_root(int v) {
    access(v);
    while (c[v][0]) v = c[v][0];
    access(v);
    return v;
}

int find_depth(int v) {
    access(v);
    return sum[c[v][0]];
}

void cut(int v) {
    // cut link between v and par[v]
    access(v);
    pp[c[v][0]] = p[c[v][0]] = 0; // fix
    sum[v] -= sum[c[v][0]];
    c[v][0] = 0;
}

void link(int v, int w) {
    // v, which is root of another tree, is now child
    // of w
    access(v), access(w);
    pp[w] = v; con(w, 0);
}

int anc(int v, int num) {
    if (find_depth(v) < num) return 0;
    access(v);
    v = c[v][0];

    while (1) {
        if (sum[c[v][1]] >= num) v = c[v][1];
        else if (sum[c[v][1]] + 1 == num) return v;
        else num -= (sum[c[v][1]] + 1), v = c[v][0];
    }
}

int main() {
    FOR(i, 1, 100001) sum[i] = 1;

    link(2, 1);
    link(3, 1);
    link(4, 1);
    link(5, 4);
    link(10, 4);
    link(7, 6);
    link(8, 7);
    link(9, 8);

    FOR(i, 1, 11) cout << i << " " << find_root(i) << " "
        << find_depth(i) << " " << anc(i, 2) << "\n";
    cout << "\n";
}

```

```

cut(4);
link(4,8);

FOR(i,1,11) cout << i << " " << find_root(i) << "
" << find_depth(i) << " " << anc(i,2) << "\n";
}

```

2.6.3 (5) Splay Tree

```

/*
* Description: based off treap code
*/

struct snode {
    int val;
    snode *p, *c[2];
    snode (int v) {
        val = v;
        c[0] = c[1] = p = 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 setLink(snode* x, snode* y, int d) {
    if (x) x->c[d] = y;
    if (y) y->p = x;
}

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

void rot(snode* x, int d) {
    snode *y = x->c[d], *z = x->p;
    setLink(x, y->c[d^1], d);
    setLink(y, x, d^1);
    setLink(z, y, getDir(z, x));
}

snode* splay(snode* 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;
}

pair<snode*,snode*> find(snode *cur, int v) { // x.f
    is result, x.s is lowest
    if (!cur) return {cur,cur};
}

```

```

pair<snode*,snode*> x;
if (cur->val >= v) {
    x = find(cur->c[0],v);
    if (!x.f) x.f = cur;
} else x = find(cur->c[1],v);
if (!x.s) x.s = cur;
return x;
}

snode* getmx(snode* x) {
    return x->c[1]?getmx(x->c[1]):x;
}

pair<snode*,snode*> split(snode* x, int v) {
    if (!x) return {x,x};
    auto y = find(x,v); y.s = splay(y.s);
    if (!y.f) return {y.s,NULL};

    y.f = splay(y.f);
    auto z = y.f->c[0]; setLink(y.f,NULL,0),
        setLink(NULL,z,0);
    return {z,y.f};
}

snode* merge(snode* x, snode* y) {
    if (!x) return y;
    x = splay(getmx(x));
    setLink(x,y,1);
    return x;
}

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

snode* del(snode* 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);
}

snode* root;

int main() {
    root = ins(root,1);
    root = ins(root,9);
    root = ins(root,3);

    root->inOrder(1);

    root = ins(root,7);
    root = ins(root,4);
    root = del(root,9);

    root->inOrder(1);
}

```

3 (2) Graphs

3.1 (2) Searching Demo

3.1.1 (2) BFS on Grid

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

void process(pii x) {
    FOR(i,4) { // easily iterate through adjacent points
        pii 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
}
```

3.1.2 (2) DFS on Graph

```
int n, visit[100001];
vi adj[100001];

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);
}
```

```
}
```

3.1.3 (2) Tree Diameter

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

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

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

void dfs(int cur) {
    memset(dist,0,sizeof dist);
    dfs(cur,-1);
}

int treeDiameter() {
    dfs(0);
    int bes = 0; FOR(i,n) if (dist[i] > dist[bes]) bes = i;
    dfs(bes); FOR(i,n) if (dist[i] > dist[bes]) bes = i;
    return dist[bes];
}
```

3.2 (3) Shortest Path

3.2.1 (3) Bellman-Ford

```
/**
 * Source: own
 * Description: can be useful with linear programming
 * Usage: https://open.kattis.com/problems/shortestpath3
 */

const ll INF = 1e18;

int n,m,q,s,bad[1000];
vector<pair<pii,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";
}

```

3.2.2 (3) Dijkstra

```

/**
 * Source: own
 */

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

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

        q.push({0,0});
        while (q.size()) {
            pii x = q.top(); q.pop();
            if (dist[x.s] < x.f) continue;
            for (pii 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});
            }
        }
    }
};

Dijkstra<100> D;

int main() {
    FOR(i,100) FOR(j,100) if (rand() % 10 == 0)
        D.adj[i].pb({j, rand() % 10+1});
    D.gen();
    FOR(i,100) cout << D.dist[i] << "\n";
}

```

3.2.3 (3) Floyd-Warshall

```

/**
 * Source: own
 * Usage: https://open.kattis.com/problems/allpairspath
 */

```

```

const ll INF = 1e18;

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";
}

```

3.3 (3) Topological Sort

```

int N,M, in[100001];
vi res, adj[100001];

void topo() {
    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);
        }
    }
}

int main() {
    cin >> N >> M;
}

```

```

FOR(i,M) {
    int x,y; cin >> x >> y;
    adj[x].pb(y), in[y] ++;
}
topo();
for (int i: res) cout << i << " ";
}

```

3.4 (4) Biconnected Components

```

/**
 * Source:
 * http://www.geeksforgeeks.org/biconnected-components/
 * Some corrections!
 */

struct BCC {
    int V, ti = 0;
    vector<vi> adj;
    vi par, disc, low;
    vector<vector<pii>> fin;
    vector<pii> st;

    void init(int _V) {
        V = _V;
        par.resize(V), disc.resize(V), low.resize(V),
        adj.resize(V);
        FOR(i,V) par[i] = disc[i] = low[i] = -1;
    }
    void addEdge(int u, int v) {
        adj[u].pb(v), adj[v].pb(u);
    }

    void BCCutil(int u) {
        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 ((disc[u] == 0 && child > 1) ||
                    (disc[u] != 0 && disc[u] <=
                     low[i])) { // articulation point!
                    vector<pii> 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,V) if (disc[i] == -1) {
        BCCutil(i);
        if (st.size()) fin.pb(st);
        st.clear();
    }
}

};

int main() {
    BCC g; g.init(12);
    g.addEdge(0,1);
    g.addEdge(1,2);
    g.addEdge(1,3);
    g.addEdge(2,3);
    g.addEdge(2,4);
    g.addEdge(3,4);
    g.addEdge(1,5);
    g.addEdge(0,6);
    g.addEdge(5,6);
    g.addEdge(5,7);
    g.addEdge(5,8);
    g.addEdge(7,8);
    g.addEdge(8,9);
    g.addEdge(10,11);
    g.bcc();

    for (auto a: g.fin) {
        for (pii b: a) cout << b.f << " " << b.s << "
            | ";
        cout << "\n";
    }
}

```

3.5 (4) Kosaraju

```

/**
 * Source: Wikipedia
 * Description: generates SCC in topological order
 */

const int MX = 100001;

struct scc {
    vi adj[MX], radj[MX], todo;
    int comp[MX], N, M;
    bool visit[MX];

    scc() {
        memset(comp,0,sizeof comp);
        memset(visit,0,sizeof 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]) dfs2(w,val);
}

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

void genSCC() {
    FOR(i,1,N+1) if (!visit[i]) dfs(i);
    reverse(all(todo)); // toposort
    for (int i: todo) if (!comp[i]) dfs2(i,i);
}
};

scc S;

int main() {
    cin >> S.N >> S.M;
    FOR(i,S.M) {
        int a,b; cin >> a >> b;
        S.addEdge(a,b);
    }
    S.genSCC();
}

```

3.6 (6) Euler Tour

```

/**
 * Description: extra log factor
 * Usage: https://open.kattis.com/problems/eulerianpath
 */

vi circuit;
multiset<int> adj[10000], adj1[10000];
int N,M, out[10000], in[10000];

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

int a,b,start;

void solve() {
    FOR(i,N) {
        adj[i].clear(), adj1[i].clear();
        out[i] = in[i] = 0;
    }
    circuit.clear();
    FOR(i,M) {
        cin >> a >> b;
        adj[a].insert(b), adj1[a].insert(b);
        out[a] ++, in[b] ++;
    }
}

```

```

}
start = a;
FOR(i,N) if (out[i]-in[i] == 1) start = i;

find_circuit(start);
reverse(circuit.begin(),circuit.end());

if (circuit.size() != M+1) {
    cout << "Impossible\n";
    return;
}

FOR(i,M) {
    if (adj1[circuit[i]].find(circuit[i+1]) ==
        adj1[circuit[i]].end()) {
        cout << "Impossible\n";
        return;
    }
    int t = circuit[i];
    adj1[t].erase(adj1[t].find(circuit[i+1]));
}
FOR(i,M+1) cout << circuit[i] << " ";
cout << "\n";
}

```

4 (2) Paradigms

4.1 (2) Interval Cover

```

/**
 * Source: own
 * Usage: https://open.kattis.com/problems/intervalcover
 */

double A,B,cur;
vector<pair<pdd,int>> in;
int N,nex;
vi ans;

void solve() {
    nex = 0; ans.clear();
    cin >> N; in.resize(N);
    FOR(i,N) {
        cin >> in[i].f.f >> in[i].f.s;
        in[i].s = i;
    }

    sort(all(in));
    pair<double,int> mx = {-DBL_MAX,-1};

    while (nex < in.size() && in[nex].f.f <= A) {
        mx = max(mx,{in[nex].f.s,in[nex].s});
        nex++;
    }
    if (nex == 0) {
        cout << "impossible\n";
        return;
    }
    ans.pb(mx.s);
}

```

```

while (mx.f < B) {
    cur = mx.f;
    while (nex < in.size() && in[nex].f.f <= cur) {
        mx = max(mx, {in[nex].f.s, in[nex].s});
        nex++;
    }
    if (mx.f == cur) {
        cout << "impossible\n";
        return;
    }
    ans.pb(mx.s);
}
cout << ans.size() << "\n";
for (int i: ans) cout << i << " ";
cout << "\n";
}

```

4.2 (4) Discrete Logarithm

```

/**
 * Description: find k such that primitive^k=x, meet
 *             in the middle!
 * Source: Own
 */

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";
}

```

4.3 (4) Ternary Search

```

/**
 * Source: Own
 */

double eval(double x) { // strictly decreasing then
    strictly increasing
    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

```

5 (3) Dynamic Programming

5.1 (3) Distinct Subsequences

```

/**
 * Source: own
 */

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;
}

```

5.2 (3) LCS

```

/**
 * Source: own
 */

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)];
}

```



```
}
```

5.3 (3) LIS

```
/**
 * Source: own
 */

vi bes = {0};
int n;

void ad(int x) {
    int lo = 0, hi = sz(bes)-1;
    while (lo < hi) {
        int mid = (lo+hi+1)/2;
        if (bes[mid] < x) lo = mid;
        else hi = mid-1;
    }
    if (lo == sz(bes)-1) bes.pb(0);
    bes[lo+1] = x;
}

int main() {
    cin >> n;
    FOR(i,n) {
        int x; cin >> x;
        ad(x);
    }
    cout << sz(bes)-1;
}
```

5.4 (4) TSP

```
/**
 * Source: own
 * Description: Example of bitset DP
 */

const int MX = 18;
const double INF = 1e18;

double dp[MX][1<<MX], dist[MX][MX];

double solve() {
    FOR(i,MX) FOR(j,1<<MX) dp[i][j] = INF;

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

    double ans = INF;
    FOR(j,1,MX) ans =
        min(ans, dp[j][(1<<MX)-1]+dist[j][0]);
    return ans;
}
```

6 (3) Strings

6.1 (3) Hashing

```
/**
 * Source: own
 */

typedef pair<ll, ll> pll;

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<pll> po, ipo, cum;
    pll base = mp(948392576, 573928192);

    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] = {inv(po[i].f), inv(po[i].s)};
        }
        FOR(i,sz(S)) cum[i+1] =
            cum[i]+po[i]*(ll)(S[i]-'a'+1);
    }

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

```

int lcp(hsh& a, hsh& b) {
    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);
}

```

6.2 (4) String Searching

6.2.1 (4) Aho-Corasick

```

/**
 * Source: https://ideone.com/0cMjZJ
 * Usage:
 * https://open.kattis.com/problems/stringmultimatching
 */

template<int SZ> struct Aho {
    int link[SZ], dict[SZ], sz = 1, num = 0;
    vector<pii> 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) {
        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) {
        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<100001> A;

int n;

void solve() {
    A = Aho<100001>();
    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";
    }
}

```

6.2.2 (4) Bitset Trie

```

/**
 * Source: own?
 */

```

```

template<int MX> struct tri {
    int nex = 0, ans = 0;
    int trie[MX][2]; // easily changed to character

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

    void ins(int x) {
        int cur = 0;
        FORd(i,30) {
            int t = (x&(1<<i))>>i;
            if (!trie[cur][t]) trie[cur][t] = ++nex;
            cur = trie[cur][t];
        }
    }

    void test(int x) {
        int cur = 0;
        FORd(i,30) {
            int t = ((x&(1<<i))>>i) ^ 1;
            if (!trie[cur][t]) t ^= 1;
            cur = trie[cur][t];
            if (t) x ^= (1<<i);
        }
        ans = max(ans,x);
    }
};

```

6.2.3 (4) Z

```

/**
 * Source: http://codeforces.com/blog/entry/3107
 * similar to KMP
 */

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 << " ";
}

```

6.3 (4) Suffix Array

```

/**
 * Source: SuprDewd CP Course
 * Task: https://open.kattis.com/problems/suffixsorting
 * KACTL version is slightly faster
 */

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;
}
};

```

6.4 (5) Manacher

```

/**
 * Source: http://codeforces.com/blog/entry/12143
 * Description: Calculates length of largest palindrome
               centered at each character of string
 */

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) {
        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,ans.size()) if ((i&1) == (ans[i]&1)) ans[i]
        ++; // adjust lengths
    return ans;
}

int main() {
    vi a1 = manacher("abacaba");
    for (int i: a1) cout << i << " ";
    cout << "\n";

    vi a2 = manacher("aabbaaccaabbbaa");
    for (int i: a2) cout << i << " ";
}

```

7 (3) Trees

7.1 (3) DSU, Kruskal

```

/**
 * Source: own
 * Description: computes the minimum spanning tree in
               O(ElogE) time
 */

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;
    }
};

int ans = 0;
vector<pair<int,pii>> edge;

DSU<100> D;

void kruskal() {
    sort(edge.begin(),edge.end());
    for (auto a: edge) if (D.unite(a.s.f,a.s.s))
        ans += a.f;
}

int main() {
    FOR(i,100) FOR(j,i+1,100) if (rand() % 5 == 0)
        edge.pb({rand() % 100+1,{i,j}});
    kruskal();
    cout << D.sz[D.get(5)] << " " << ans;
}

```

7.2 (4) Tree Queries

7.2.1 (4) Centroid Decomposition

```

/**
 * Source: own
 */

const int MX = 100001;

int N, visit[MX], sub[MX], par[MX];

```

```

vi adj[MX];

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

int get_centroid(int x) {
    par[x] = 0;
    dfs(x);
    int sz = sub[x];
    while (1) {
        pii mx = {0,0};
        for (int i: adj[x]) if (!visit[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 = get_centroid(x); visit[x] = 1;
    // do stuff
    cout << x << "\n";
    for (int i: adj[x]) if (!visit[i]) solve(i);
}

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

```

7.2.2 (4) HLD

```

/**
 * Source: http://codeforces.com/blog/entry/22072
 * Task: USACO Grass Planting
 */

// insert LazySegTree Template

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 = graph.size();

```

```

        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";
    }
}

```

7.2.3 (4) LCA with Binary Jumps

```

/**
 * Source: USACO Camp
 */

const int MAXN = 100001, MAXK = 17;

int Q;

struct LCA {
    int V;
    vi edges[MAXN];
    int parK[MAXK][MAXN];
    int depth[MAXN];

    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(1, 0);
        FOR(k,1,MAXK) FOR(i,1,V+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)];
    }
};

```

```

LCA L;

int main(){
    cin >> L.V >> Q;
    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.dist(u,v) << "\n";
    }
}

```

7.2.4 (4) LCA with RMQ

```

/**
 * Description: Euler Tour LCA w/ O(1) query
 * Source: own
 */

const int MAXN = 100001, MAXK = 17;

int Q;

struct RMQ2 {
    vi edges[MAXN];
    pii rmq[MAXK][2*MAXN];
    int depth[MAXN], pos[MAXN];

    int N, nex=0;

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

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

    void construct() {
        dfs(1, 0);
        FOR(k,1,MAXK) FOR(i,nex) if (i+(1<<(k-1)) <
            nex) rmq[k][i] =
                min(rmq[k-1][i],rmq[k-1][i+(1<<(k-1))]);
    }

    int lca(int u, int v){
        u = pos[u], v = pos[v];
        if (u > v) swap(u,v);
        int x = 31-__builtin_clz(v-u+1);
        return min(rmq[x][u],rmq[x][v-(1<<x)+1]).s;
    }
}

```

```

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

RMQ2 R;

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

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

```

8 (4) Flows

8.1 (5) Dinic

```

/**
 * Source: GeeksForGeeks
 */

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

template<int SZ> struct Dinic {
    int level[SZ], start[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(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;
    }
};

```

```

    }

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

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

                if (level[e.v] == level[u]+1 && e.flow < e.C) {
                    int curr_flow = min(flow, e.C - e.flow);
                    int 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;
    }

    int maxFlow(int s, int t) {
        if (s == t) return -1;
        int total = 0;

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

        return total;
    }
};

Dinic<1000> D;

int main() {
    D.addEdge(0, 1, 16 );
    D.addEdge(0, 2, 13 );
    D.addEdge(1, 2, 10 );
    D.addEdge(1, 3, 12 );
    D.addEdge(2, 1, 4 );
    D.addEdge(2, 4, 14);
    D.addEdge(3, 2, 9 );
    D.addEdge(3, 5, 20 );
    D.addEdge(4, 3, 7 );
    D.addEdge(4, 5, 4);

    cout << "Maximum flow " << D.maxFlow(0, 5);
}

```

8.2 (5) Push-Relabel

```

/**

```

```

* Source: http://codeforces.com/blog/entry/14378
* Unused
*/

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

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

    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) {
        int 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);
    }
}

int 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];
}

};

PushRelabel<1000> D;

int main() {
    D.addEdge(0, 1, 16 );
    D.addEdge(0, 2, 13 );
    D.addEdge(1, 2, 10 );
    D.addEdge(1, 3, 12 );
    D.addEdge(2, 1, 4 );
    D.addEdge(2, 4, 14);
    D.addEdge(3, 2, 9 );
    D.addEdge(3, 5, 20 );
    D.addEdge(4, 3, 7 );
    D.addEdge(4, 5, 4);

    cout << "Maximum flow " << D.maxFlow(0, 5);
}

```

8.3 (6) MinCostFlow

```

/**
* Source: GeeksForGeeks
*/

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

template<int SZ> struct mcf {
    pii 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<pii,vector<pii>,greater<pii>>
        todo; todo.push({0,SC});

    while (todo.size()) {
        pii 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];
    }
}

pii 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);

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

```

9 (4) Geometry

9.1 (4) Convex Hull

9.1.1 (4) Convex Hull

```

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

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

vector<pii> convex_hull(vector<pii> P) {
    sort(P.begin(),P.end());
    P.erase(unique(P.begin(),P.end()),P.end());
    if (P.size() == 1) return P;

    int n = P.size();

    vector<pii> bot = {P[0]};
    FOR(i,1,n) {
        while (bot.size() > 1 &&
            cross(bot[bot.size()-2], bot.back(), P[i])
                <= 0) bot.pop_back();
        bot.pb(P[i]);
    }
    bot.pop_back();

    vector<pii> up = {P[n-1]};
    FORd(i,n-1) {
        while (up.size() > 1 && cross(up[up.size()-2],
            up.back(), P[i]) <= 0) up.pop_back();
        up.pb(P[i]);
    }
    up.pop_back();

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

int main() {
    int n;
    while (cin >> n) {
        if (n == 0) break;
        vector<pii> P(n); FOR(i,n) cin >> P[i].f >>
            P[i].s;
        vector<pii> hull = convex_hull(P);

        cout << hull.size() << "\n";
        for (auto a: hull) cout << a.f << " " << a.s
            << "\n";
    }
}

```

9.1.2 (4) LiChao Segment Tree

```

/**
 * Source:
 *   http://codeforces.com/blog/entry/51275?#comment-351413
 * Unused
 */

const int N = 100000 + 5;

int n, m;
int vis[N << 1];
char op[100];

struct line {
    double k, b;
    line(double _k = 0, double _b = 0) { k = _k; b =
        _b; }
    double get(double x) { return k * x + b; }
} c[N << 1];

void modify(int x, int l, int r, line v) {
    if (!vis[x]) { vis[x] = 1, c[x] = v; return; }
    if (c[x].get(l) > v.get(l) && c[x].get(r) >
        v.get(r)) return;
    if (c[x].get(l) < v.get(l) && c[x].get(r) <
        v.get(r)) { c[x] = v; return; }
    int m = (l + r) >> 1;
    if (c[x].get(l) < v.get(l)) swap(c[x], v);
    if (c[x].get(m) > v.get(m)) modify(x << 1 | 1, m + 1,
        r, v);
    else {swap(c[x], v); modify(x << 1, l, m, v);}
}

double get(int x, int l, int r, int pos) {
    if (l == r) return c[x].get(l);
    int m = (l + r) >> 1; double ans = c[x].get(pos);
    if (pos <= m) ans = max(ans, get(x << 1, l, m, pos));
    else ans = max(ans, get(x << 1 | 1, m + 1, r, pos));
    return ans;
}

int main() {
    cin >> n;
    FOR(i, n) {
        cin >> op;
        if (op[0] == 'Q') {
            int x; cin >> x;
            cout << get(1, 1, n, x) << "\n";
        } else {
            double k, b; cin >> b >> k;
            line l = line(k, b);
            modify(1, 1, n, l);
        }
    }
}

```

9.1.3 (4) LineContainer

```

/**
 * Source: KACTL

```

```

 * Unused
 */

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

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

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

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

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

int main() {
}

```

9.2 (4) Misc

9.2.1 (4) Circles

```

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

```

```

*/

typedef complex<double> cd;
typedef pair<cd,double> circle;

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

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

double 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;
}

int main() {
    cout << intersect({0,2},{1,1}) << "\n";
    cout << arc({0,1},cd(1,0),cd(0,1)) << "\n";
    cout << on({0,1},1) << "\n";
}

```

9.2.2 (4) Closest Pair

```

/**
 * Source: GeeksForGeeks
 * Description:  $N \log^2 N$ , can be improved
 * Use: https://open.kattis.com/problems/closestpair2
 */

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

int n;

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

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

pair<double,pair<pdd,pdd>> strip(vector<pdd> v, double
    di) {
    pair<double,pair<pdd,pdd>> ans = MN;
    FOR(i,v.size()) FOR(j,i+1,v.size()) {
        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<pdd,pdd>> bes (vector<pdd> v) {
    if (v.size() == 1) return MN;
    int M = v.size()/2;
    vector<pdd> 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<pdd> 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<pdd> v(n);
        FOR(i,n) cin >> v[i].f >> v[i].s;
        sort(v.begin(),v.end());
        auto a = bes(v);
        cout << a.s.f.f << " " << a.s.f.s << " " <<
            a.s.s.f << " " << a.s.s.s << "\n";
    }
}

```

9.2.3 (4) Line Segment Intersection

```

/**
 * Source:
 *   https://open.kattis.com/problems/segmentintersection
 * If numbers are small enough, fractions are
 *   recommended.
 */

typedef pair<double,double> pdd;

pii A,B,C,D;

pdd operator*(int x, pdd y) {
    return {x*y.f,x*y.s};
}

pdd operator/(pdd y, int x) {
    return {y.f/x,y.s/x};
}

pdd operator+(pdd l, pdd r) {
    return {l.f+r.f,l.s+r.s};
}

int sgn(pii a, pii b, pii c) {
    return (b.s-a.s)*(c.f-a.f)-(b.f-a.f)*(c.s-a.s);
}

```

```

pdd get(pii a, pii b, pii c, pii d) {
    return (abs(sgn(a,b,c))*d+abs(sgn(a,b,d))*c)
           /(abs(sgn(a,b,c))+abs(sgn(a,b,d)));
}

void solve() {
    cin >> A.f >> A.s >> B.f >> B.s >> C.f >> C.s >>
        D.f >> D.s;
    if (A > B) swap(A,B);
    if (C > D) swap(C,D);
    int a1 = sgn(A,B,C), a2 = sgn(A,B,D);
    if (a1 > a2) swap(a1,a2);
    if (!(a1 <= 0 && a2 >= 0)) {
        cout << "none\n";
        return;
    }
    if (a1 == 0 && a2 == 0) {
        if (sgn(A,C,D) != 0) {
            cout << "none\n";
            return;
        }
        pii x1 = max(A,C), x2 = min(B,D);
        if (x1 > x2) cout << "none\n";
        else if (x1 == x2) cout << (double)x1.f << " " <<
            << (double)x1.s << "\n";
        else cout << (double)x1.f << " " <<
            << (double)x1.s << " " << (double)x2.f << " " <<
            << (double)x2.s << "\n";
        return;
    }
    pdd z = get(A,B,C,D);
    if (mp((double)A.f,(double)A.s) <= z && z <=
        mp((double)B.f,(double)B.s)) cout << z.f << "
        " << z.s << "\n";
    else cout << "none\n";
}

int main() {
    int n; cin >> n;
    cout << fixed << setprecision(2);
    FOR(i,n) solve();
}

```

9.2.4 (4) MaxCollinear

```

/**
 * Source: own
 * Usage: https://open.kattis.com/problems/maxcollinear
 */

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

pair<pii,int> getline(pii a, pii b) {
    pii 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";
}

```

9.2.5 (4) Pair Operators

```

/**
 * Source: own
 */

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

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

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

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

template<class T> double mag(pair<T,T> p) {
    return sqrt(p.f*p.f+p.s*p.s);
}

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

template<class T> pair<T,T> operator/(const pair<T,T>&
    l, const pair<T,T>& r) {
    // l.f+l.s*i, r.f+r.s*i
    pair<T,T> z =
        {r.f/(r.f*r.f+r.s*r.s),-r.s/(r.f*r.f+r.s*r.s)};
    return l*z;
}

template<class T> double area(pair<T,T> a, pair<T,T>
    b, pair<T,T> c) {
    b = b-a, c = c-a;
}

```

```

    return (b.f*c.s-b.s*c.f)/2;
}

template<class T> double dist(pair<T,T> l, pair<T,T>
r) {
    return mag(r-l);
}

template<class T> double dist(pair<T,T> o, pair<T,T>
x, pair<T,T> d) { // signed distance
    return 2*area(o,x,x+d)/mag(d);
}

```

9.2.6 (4) Point in Polygon

```

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

int n,m;
pii p[1000];

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

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

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

void test(pii z) {
    int ans = 0;
    FOR(i,n) {
        pii x = p[i], y = p[(i+1)%n];
        if (on(x,y,z)) {
            cout << "on\n";
            return;
        }
        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) cout << "in\n";
    else cout << "out\n";
}

void solve() {
    FOR(i,n) cin >> p[i].f >> p[i].s;
    cin >> m;
    FOR(i,m) {
        pii z; cin >> z.f >> z.s;

```

```

        test(z);
    }
}

```

9.2.7 (4) Polygon Area

```

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

double area(vector<pii> v) {
    double x = 0;
    FOR(i,sz(v)) {
        int j = (i+1)%sz(v);
        x += (ll)v[i].f*v[j].s;
        x -= (ll)v[j].f*v[i].s;
    }
    return x/2;
}

```

9.3 (6) KD Tree

```

/**
 * Source: KACTL
 * Description: Supports nearest neighbor query in
 *   O(log n) assuming random distribution
 * unused
 */

int t = 0, cur = 0;

struct point {
    ll d[2];
    point(ll x, ll y) {
        d[0] = x, d[1] = y;
    }
    point() {
        d[0] = 0, d[1] = 0;
    }
};

ll distance(point a, point b) {
    ll d = 0;
    FOR(i,2) d += (a.d[i]-b.d[i])*(a.d[i]-b.d[i]);
    return d;
}

bool comp(point a, point b) {
    return a.d[cur] < b.d[cur];
}

struct node {
    point* pt = NULL;
    point lo, hi;
    node* c[2];
    int ax = 0;

```

```

11 dist(point p) {
    ll d = 0;
    FOR(i,2) {
        if (p.d[i] < lo.d[i]) d +=
            (p.d[i]-lo.d[i])*(p.d[i]-lo.d[i]);
        else if (p.d[i] > hi.d[i]) d +=
            (p.d[i]-hi.d[i])*(p.d[i]-hi.d[i]);
    }
    return d;
}

node(int axis, point low, point high,
    vector<point> p) {
    lo = low, hi = high, ax = axis;
    if (p.size() > 1) {
        cur = ax;
        sort(p.begin(),p.end(),comp);

        int M = p.size()/2;
        while(M > 0 && p[M].d[ax] == p[M-1].d[ax])
            M--;

        point lo1 = lo; lo1.d[ax] = p[M].d[ax];
        point hi1 = hi; hi1.d[ax] = p[M].d[ax]-1;

        if (M) c[0] = new node((ax+1)%2,lo,hi1,
            {p.begin(),p.begin()+M});
        c[1] = new node((ax+1)%2,lo1,hi,
            {p.begin()+M,p.end()});
    } else if (p.size() == 1) {
        pt = new point(p[0]);
    }
}

point get(point p) {
    if (pt) return *pt;
    if (!c[0]) return c[1]->get(p);

    int t = c[0]->dist(p) < c[1]->dist(p) ? 0 : 1;
    point z = c[t]->get(p);
    if (distance(p,z) <= c[t^1]->dist(p)) return z;

    point z1 = c[t^1]->get(p);
    if (distance(p,z) < distance(p,z1)) return z;
    return z1;
}

};

node* root;

int main() {
    vector<point> x;
    FOR(i,100000) x.pb(point(rand() % 1000000000,
        rand() % 1000000000));

    root = new
        node(0,point(-MOD,-MOD),point(MOD,MOD),x);
    FOR(i,100000) {
        point y(rand() % 1000000000, rand() %
            1000000000);
        cout << y.d[0] << " " << y.d[1] << " " <<
            root->get(y).d[0] << " " <<

```

```

        root->get(y).d[1] << "\n";
    }
}

```

10 (4) Math

10.1 (4) Matrix

10.1.1 (4) Matrix Exponentiation

```

/**
 * Source: KACTL
 */

template<int SZ> struct mat {
    array<array<ll,SZ>,SZ> d;

    mat() {
        FOR(i,SZ) FOR(j,SZ) d[i][j] = 0;
    }

    mat operator+(const mat& m) {
        mat<SZ> a;
        FOR(i,SZ) FOR(j,SZ) a.d[i][j] =
            (d[i][j]+m.d[i][j]) % MOD;
        return a;
    }

    mat operator*(const mat& m) {
        mat<SZ> a;
        FOR(i,SZ) FOR(j,SZ) FOR(k,SZ)
            a.d[i][j] = (a.d[i][j]+d[i][k]*m.d[k][j]) %
                MOD;
        return a;
    }

    mat operator^(ll p) {
        mat<SZ> a, b(*this);
        FOR(i,SZ) a.d[i][i] = 1;

        while (p) {
            if (p&1) a = a*b;
            b = b*b;
            p /= 2;
        }

        return a;
    }

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

/*

```

```

mat<2> x; x.d[0][0] = 1, x.d[1][0] = 2, x.d[1][1] = 1,
      x.d[0][1] = 3;
mat<2> y = x*x;
mat<2> z = x^5;
x.print(), y.print(), z.print();
*/

```

10.1.2 (6) Linear Equation Solver

```

/**
 * Source: Own
 * Usage:
 *   https://open.kattis.com/problems/equationsolverplus
 */

typedef long double ld;
typedef vector<vector<ld>> mat;

ld EPS = 1e-10;
int n;

void elim(mat& a, int i, int j, int k) {
    ld t = a[k][i];
    FOR(ind,n+1) a[k][ind] -= t*a[j][ind];
}

void prin(mat& a) {
    FOR(i,n) {
        FOR(j,n+1) cout << a[i][j] << " ";
        cout << "\n";
    }
    cout << "----\n";
}

void solve() {
    mat a(n); FOR(i,n) a[i].resize(n+1);
    FOR(i,n) FOR(j,n) cin >> a[i][j];
    FOR(i,n) cin >> a[i][n];
    int done[n]; FOR(i,n) done[i] = -1;

    FOR(i,n) {
        FOR(j,n) if (done[j] == -1 && abs(a[j][i]) >
            EPS) {
            ld t = a[j][i];
            FOR(k,n+1) a[j][k] /= t;

            FOR(k,n) if (j != k) elim(a,i,j,k);
            done[j] = i; break;
        }
    }

    int num = 0;
    FOR(i,n) if (done[i] == -1) {
        num++;
        if (abs(a[i][n]) > EPS) {
            cout << "inconsistent\n";
            return;
        }
    }
}

```

```

ld ans[n]; FOR(i,n) ans[i] =
    numeric_limits<double>::max();
FOR(i,n) if (done[i] != -1) {
    bool bad = 0;
    FOR(j,n) if (j != done[i] && abs(a[i][j]) >
        EPS) {
        bad = 1;
        break;
    }
    if (!bad) ans[done[i]] = a[i][n];
}
FOR(i,n) {
    if (ans[i] != numeric_limits<double>::max())
        cout << ans[i];
    else cout << "?";
    cout << " ";
}
cout << "\n";
}

```

10.2 (4) Number Theory

10.2.1 (4) Eratosthenes' 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];
    }
};

```

10.2.2 (5) Chinese Remainder Theorem

```

/**
 * Source: Own
 * Usage: Kattis generalchineseremainder
 */

ll n,m,a,b;
map<ll,pii> M;
bool bad;

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;
    while (tmp < 0) tmp += b;
    return tmp;
}

ll naive(ll n, ll m, ll a, ll b) {
    ll x = (a-b)*inv(m,n) % n;
    ll ans = (m*x+b) % (m*n);
    while (ans < 0) ans += (m*n);
    return ans;
}

void process(ll a, ll n) {
    vector<pii> z;
    for (int i = 2; i*i <= n; ++i) if (n % i == 0) {
        int co = 0;
        while (n % i == 0) n /= i, co++;
        z.pb({i,co});
    }
    if (n != 1) z.pb({n,1});
    for (auto A: z) {
        if (M.count(A.f)) {
            pii p1 = M[A.f];
            pii p2 = {A.s,a%(ll)pow(A.f,A.s)};
            if (p1 > p2) swap(p1,p2);
            if (p2.s%(ll)pow(A.f,p1.f) != p1.s) bad = 1;
            M[A.f] = p2;
        } else M[A.f] = {A.s,a%(ll)pow(A.f,A.s)};
    }
}

ll po(ll b, ll p) {
    ll z = 1;
    FOR(i,p) z *= b;
    return z;
}

void solve() {
    bad = 0, M.clear();
    long long aa,nn,bb,mm; cin >> aa >> nn >> bb >> mm;
    a = aa, n = nn, b = bb, m = mm;
    process(a,n), process(b,m);
    if (bad) {
        cout << "no solution\n";
        return;
    }
    ll a1 = 0, a2 = 1;
    for (auto& x: M) {
        a1 = naive(a2,po(x.f,x.s.f),a1,x.s.s);
        a2 *= po(x.f,x.s.f);
    }
    cout << (ll)a1 << " " << (ll)a2 << "\n";
}

int main() {
    int T; cin >> T;
    FOR(i,T) solve();
}

```

10.2.3 (5) Combinations

```

/**
 * Source: Own
 * MOD is a large prime
 * see ComboExtended for mods in the form p^k
 */

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

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

    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 comb(ll a, ll b) {
        if (a < b) return 0;
        ll tmp = fac[a]*ifac[b] % MOD;
        tmp = tmp*ifac[a-b] % MOD;
        return tmp;
    }
};

typedef pair<ll,ll> pll;

template<int SZ> struct ComboExtended {
    pll fac[SZ+1], ifac[SZ+1], mod;
    ll MOD = 1;

    void init(pll _mod) { // prime, power
        mod = _mod; FOR(i,mod.s) MOD *= mod.f;

        fac[0] = ifac[0] = {1,0};
        FOR(i,1,SZ+1) {
            fac[i] = fac[i-1];
            int I = i, z = 0;
            while (I % mod.f == 0) I /= mod.f, z++;
            fac[i].f = fac[i].f*I%MOD; fac[i].s += z;
            ifac[i] = {inv(fac[i].f,MOD),fac[i].s};
        }
    }

    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;
        while (tmp < 0) tmp += b;
        return tmp;
    }
}

```



```

11 comb(11 a, 11 b) {
    if (a < b) return 0;
    11 tmp = (fac[a].f*ifac[b].f%MOD)*ifac[a-b].f
        % MOD;
    11 z = fac[a].s-fac[b].s-fac[a-b].s;
    if (z >= mod.s) return 0;
    FOR(i,z) tmp = tmp*mod.f % MOD;
    return tmp;
}
};

```

10.2.4 (5) Phi

```

/**
 * Source: Own
 * the number of operations needed s.t.
 *   phi(phi(...phi(n)...))=1 is O(log n)
 */

int phi(int x) {
    if (x == 1) return 1;
    int X = x;

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

    if (x > 1) pri.pb(x);
    for (int i: pri) { X /= i; X *= i-1; }
    return X;
}

```

10.3 (6) Polynomials

10.3.1 (6) Base Conversion

```

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

// insert NTT template

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

    Base() {
        po10[0] = {10};
        FOR(i,1,21) {
            po10[i] =
                Poly::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 = Poly::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 (11 x: o) cout << x << " ";
        cout << "\n";
    }
}

```

10.3.2 (6) FFT, NTT

```

/**
 * Sources: KACTL, https://pastebin.com/3Tnj5mRu
 * Usage: https://open.kattis.com/problems/polymul2/
 */

typedef complex<double> cd;
typedef vector<cd> vcd;
typedef vector<ll> vl;

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

namespace FFT {
    vcd fft(vcd& a) {
        int n = a.size(), x = get(n);
        vcd res, RES(n), roots(n);
        FOR(i,n) roots[i] =
            cd(cos(2*M_PI*i/n),sin(2*M_PI*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] /= a.size();
    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;
}
}

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); }

    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(a.size());
        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;
    }
}

```

```
int main() {
    vl x = Poly::FFT::conv1l({1,2,3,4,5,6,7,8},
        {1,2,3,4,5,6,7,8});
    for (auto a: x) cout << a << "\n";
    cout << "\n";

    vl X = Poly::NTT::conv({1,2,3,4,5,6,7,8},
        {1,2,3,4,5,6,7,8});
    for (auto a: X) cout << a << "\n";
}
```

11 (6) Sqrt Decomposition

11.1 (6) Mo

```
int block = 300; // sqrt(N)

bool cmp(vi a, vi b) {
    if (a[0]/block != b[0]/block) return a[0] < b[0];
    return a[1] < b[1];
}
```
