Code Template for ACM-ICPC

impulse

November 9, 2018

Contents

1 Graph

1.1 Bellman-Ford

```
const int maxn = 1000 + 11;
int N, M;
struct E {
    int u, v, w;
    E() {}
    E(int u, int v, int w):u(u),v(v),w(w){}
};
vector<E> edge;
vector<int> G[maxn];
int d[maxn];
bool BellmanFord(int s){
    memset(d, 0x3f, sizeof(d));
    d[s] = 0;
    for (int i = 0; i < N; i++){
        bool ok = true;
        for (int u = 0; u < N; u++){
            for (int j = 0; j < G[u].size(); j++){
                E &e = edge[G[u][j]];
                if (d[e.v] > d[u] + e.w) {
                    d[e.v] = d[u] + e.w;
                    ok = false;
                }
            }
        }
        if (ok) return true;
    }
    return false;
}
```

1.2 BiMatch

```
const int maxn = 1000 + 1000;
int N, M, T;
vector<int> G[maxn];
int mx[maxn], my[maxn], vis[maxn];
bool DFS(int ux){
    vis[ux] = 1;
    for (int i = 0; i < G[ux].size(); i++){
        int uy = G[ux][i];
        if (vis[uy]) continue;
        vis[uy] = 1;
        if (my[uy] == -1 || DFS(my[uy])){
            mx[ux] = uy;
            my[uy] = ux;
            return true;
        }
    }
    return false;
}
int BiMatch(){
    int cnt = 0; memset(mx, -1, sizeof(mx));
    memset(my, -1, sizeof(my));
    for (int ux = 0; ux < N; ux++) {
        if (mx[ux] != -1) continue;
        memset(vis, 0, sizeof(vis));
        if (DFS(ux)) cnt++;
    }
    return cnt;
}
```

1.3 Dijkstra

```
const int INF = 0x3f3f3f3f;
const int maxn = 1000 + 11;
int N, M, T;
struct E {
    int u, v, w;
};
vector<E> edge;
vector<int> G[maxn];
struct D{
    int d, x;
    D(){}
    D(int d, int x):d(d),x(x){}
    bool operator < (const D& rhs) const{return d > rhs.d;}
};
int pi[maxn], vis[maxn];
vector<D> v;
void Dijkstra(int s){
    v.clear();
    for (int i = 0; i < N; i++) v.push_back(D(INF, i));
    memset(pi, -1, sizeof(pi)); memset(vis, 0, sizeof(vis));
    priority_queue<D> pq; v[s].d = 0; pq.push(v[s]);
    while (!pq.empty()){
        D uu = pq.top(); pq.pop();
        if (vis[uu.x]) continue;
        vis[uu.x] = true;
        for (int i = 0; i < G[uu.x].size(); i++){
            E &e = edge[G[uu.x][i]];
            D &vv = v[e.v];
            if (vis[vv.x]) continue;
            if (vv.d > uu.d + e.w) {
                vv.d = uu.d + e.w;
                pi[vv.x] = uu.x; pq.push(vv);
            }
        }
    }
}
```

1.4 Floyd-BitSet

```
const int maxn = 2016 + 1;
int N, M, T;
bitset<maxn> bit[maxn];
void floyd(){
    for (int k = 0; k < N; k++){
        for (int i = 0; i < N; i++){
            if (bit[i][k]) bit[i] = bit[i] | bit[k];
        }
    }
}
```

1.5 KM

```
const int maxn = 2e3 + 11;
int N, M, T;

int match[maxn];
int lx[maxn],ly[maxn];
int sx[maxn],sy[maxn];
int weight[maxn][maxn];
int dfs(int x) {
    sx[x]=true;
    for(int i=0; i<M; i++) {
        if(!sy[i]&&lx[x]+ly[i]==weight[x][i]) {
            sy[i]=true;
            if(match[i]==-1||dfs(match[i])) {
                match[i]=x;
                return true;
            }
        }
    }
    return false;
}
int fax(int x) { // x: 0->minimum, 1->maximum
    if(!x) {
        for(int i=0; i<N; i++) {
            for(int j=0; j<M; j++) {
                weight[i][j]=-weight[i][j];
            }
        }
    }
    memset(match,-1,sizeof(match));
    for(int i=0; i<N; i++) {
        ly[i]=0;
        lx[i]=-INF;
        for(int j=0; j<M; j++) {
            if(weight[i][j]>lx[i]) {
                lx[i]=weight[i][j];
            }
        }
    }
    for(int i=0; i<N; i++) {
        while(1) {
            memset(sx,0,sizeof(sx));
            memset(sy,0,sizeof(sy));
            if(dfs(i))
                break;
            int mic=INF;
            for(int j=0; j<N; j++) {
                if(sx[j]) {
                    for(int k=0; k<M; k++) {
                        if(!sy[k]&&lx[j]+ly[k]-weight[j][k]<mic) {
                            mic=lx[j]+ly[k]-weight[j][k];
                        }
                    }
                }
            }
            if(mic==0)
                return -1;
            for(int j=0; j<N; j++)
                if(sx[j]) {
                    lx[j]-=mic;
                }
            for(int j=0; j<M; j++)
                if(sy[j]) {
                    ly[j]+=mic;
                }
        }
    }
    int sum=0;
    for(int i=0; i<M; i++) {
        if(match[i]>=0) {
            sum+=weight[match[i]][i];
        }
    }
    if(!x) {
        sum=-sum;
    }
    return sum;
}
```

1.6 KM-fast

```
int min_assignment(const vector<vector<int>> &c) {
    // c: the weight matrix
    // vector<vector<int> > weight(N, vector<int>(M, 0));
    const int n = c.size(), m = c[0].size(); // assert(n <= m);
    vector<int> v(m), dist(m);     // v: potential
    vector<int> matchL(n,-1), matchR(m,-1); // matching pairs
    vector<int> index(m), prev(m);
    iota(index.begin(), index.end(), 0);

    auto residue = [&](int i, int j) { return c[i][j] - v[j]; };
    for (int f = 0; f < n; ++f) {
        for (int j = 0; j < m; ++j) {
            dist[j] = residue(f, j);
            prev[j] = f;
        }
        int w;
        int j, l;
        bool end = 0;
        for (int s = 0, t = 0;;) {
            if (s == t) {
                l = s; w = dist[index[t++]];
                for (int k = t; k < m; ++k) {
                    j = index[k];
                    int h = dist[j];
                    if (h <= w) {
                        if (h < w) { t = s; w = h; }
                        index[k] = index[t]; index[t++] = j;
                    }
                }
                for (int k = s; k < t; ++k) {
                    j = index[k];
                    if (matchR[j] < 0) {
                        end = 1;
                        break;
                    }
                }
                if(end) break;
            }
            int q = index[s++], i = matchR[q];
            for (int k = t; k < m; ++k) {
                j = index[k];
                int h = residue(i,j) - residue(i,q) + w;
                if (h < dist[j]) {
                    dist[j] = h; prev[j] = i;
                    if (h == w) {
                        if (matchR[j] < 0) {
                            end = 1;
                            break;
                        }
                        index[k] = index[t]; index[t++] = j;
                    }
                }
            }
            if(end) break;
        }
        for(int k = 0; k < l; ++k)
            v[index[k]] += dist[index[k]] - w;
            int i;
            do {
                matchR[j] = i = prev[j];
                swap(j, matchL[i]);
            } while (i != f);
        }
        int opt = 0;
        for (int i = 0; i < n; ++i)
            opt += c[i][matchL[i]]; // (i, matchL[i]) is a solution
        return opt;
    }
```

1.7 Kruskal

```
const int maxn = 100000;
int M, N, T;
struct E {
    int u, v, w;
    E() {}
    E(int u, int v, int w):u(u),v(v),w(w) {}
    bool operator < (const E& rhs) const{
        return w < rhs.w;
    }
};
vector<E> edge;
int p[maxn];
void init() {for (int i = 0; i < N; i++) p[i] = i;}
```

```
int pfind(int x) {
    if (p[x] == x) return x;
    return p[x] = pfind(p[x]);
}
int kruskal() {
    init();
    sort(edge.begin(), edge.end());
    int ans = 0;
    for (int i = 0; i < edge.size(); i++) {
        int u = edge[i].u, v = edge[i].v, w = edge[i].w;
        int pu = pfind(u), pv = pfind(v);
        if (pu != pv) {
            p[pu] = pv; // Very important!!!
            ans += w;
        }
    }
    return ans;
}
```

1.8 K-Shortest-Path

```
const int INF = 0x3f3f3f3f;
const int maxn = 1000 + 11;
int N, M, T, S, E, K;
struct Edge{int v, w;};
vector<Edge> G[maxn], rG[maxn];
int d[maxn]; bitset<maxn> vis;
struct D {
    int g, u;
    bool operator < (const D &rhs) const {
        return !(g + d[u] < rhs.g + d[rhs.u]);
    }
};
void Dijkstra(int s) {
    memset(d, 0x3f, sizeof(d));
    priority_queue<pii> Q; d[s] = 0;
    Q.push({0, s});
    while (!Q.empty()) {
        int u = Q.top().se;
        Q.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (auto e : rG[u]) {
            if (d[e.v] > d[u] + e.w) {
                d[e.v] = d[u] + e.w;
                Q.push({ -d[e.v], e.v});
            }
        }
    }
}
int Astar(int s, int t, int k) {
    if (d[s] == INF) return -1;
    int cnt = 0;
    priority_queue<D> Q;
    Q.push(D{0, s});
    while (!Q.empty()) {
        D cur = Q.top();
        Q.pop();
        if (cur.u == t && ++cnt == k) return cur.g;
        for (auto e : G[cur.u]) {
            Q.push(D{cur.g + e.w, e.v});
        }
    }
    return -1;
}

int main() {
    while (scanf("%d%d",&N,&M)!=EOF){
        scanf("%d%d%d%d",&S,&E,&K,&T);
        for(int i=0;i<=N;i++)G[i].clear(),rG[i].clear();
        vis.reset();
        for(int i=0,u,v,w;i<M;i++){
            scanf("%d%d%d",&u,&v,&w);
            G[u].push_back({v, w});
            rG[v].push_back({u, w});
        }
        Dijkstra(E);
        int ans = Astar(S, E, K);
        if (ans != -1 && ans <= T) puts("yareyaredawa");
        else puts("Whitesnake!");
    }
    return 0;
}
/*
2 2
1 2 2 14
1 2 5
2 1 4

2 1
1 2 2 14
1 2 5
*/
```

1.9 LCA-bsearch

```
const int maxn = 1e5 + 11;
const int maxh = 16;
int N, M, T;
struct E{
    int u, v, w;
};
vector<E> G[maxn];
vector<E> edge;
int p[maxn][maxh], dep[maxn], Dis[maxn];
void dfs(int u, int pu = -1, int d = 0){
    p[u][0] = pu; dep[u] = d;
    for (E e : G[u]){
        if (e.v != pu) {
            Dis[e.v] = Dis[u] + e.w;
            dfs(e.v, u, d + 1);
        }
    }
}
```

```
void init(){
    dfs(0);
    for (int i = 0; i + 1 < maxh; i++){
        for (int u = 1; u <= N; u++){
            if (p[u][i] == -1) p[u][i + 1] = -1;
            else p[u][i + 1] = p[p[u][i]][i];
        }
    }
}
int lca(int u, int v){
    if (dep[u] > dep[v]) swap(u, v); // Here u is higher
    int len = abs(dep[u] - dep[v]);
    for (int i = 0; i < maxh; i++) {
        if ((len >> i) & 1) v = p[v][i];
    }
    if (u == v) return u;
    else{
        for (int i = maxh - 1; i >= 0; i--){
            if (p[u][i] != p[v][i]) u = p[u][i], v = p[v][i];
        }
        return p[u][0];
    }
}
int dist(int u, int v){
    return Dis[u] + Dis[v] - 2 * Dis[lca(u, v)];
}
```

1.10 LCA-rmq

```
const int maxn = 1e5 + 11;
int N, M, T;
struct E{
    int u, v, w;
    E(){}
    E(int u, int v, int w):u(u),v(v),w(w){}
};
vector<int> G[maxn];
vector<E> edge;
void addEdge(int u, int v, int w){
    edge.push_back(E(u, v, w));
    edge.push_back(E(v, u, w));
    G[u].push_back(edge.size() - 2);
    G[v].push_back(edge.size() - 1);
}
int ver[maxn], dep[maxn], First[maxn], Dis[maxn]; int cur = -1;
void dfs_rmq(int u = 0, int deep = 0, int pu = -1){
    ver[++cur] = u; First[u] = cur; dep[cur] = deep;
    for (int i = 0; i < G[u].size(); i++){
        E &e = edge[G[u][i]];
        if (e.v != pu){
            Dis[e.v] = Dis[u] + e.w;
            dfs_rmq(e.v, deep + 1, u);
            ver[++cur] = u; dep[cur] = deep;
        }
    }
}
/************Segment Tree*************/
```

```
struct Node{
    int L, R;
    int mmin, arg;
}node[maxn<<2];
int inline LC(int i) {return i*2;}
int inline RC(int i) {return i*2+1;}
void build(int L, int R, int i){
    node[i].L = L, node[i].R = R;
    if (L == R){node[i].mmin = dep[L]; node[i].arg = L; return;}
    int M = (L + R) / 2;
    build(L, M, LC(i));
    build(M + 1, R, RC(i));
    if (node[LC(i)].mmin > node[RC(i)].mmin){
        node[i].mmin = node[RC(i)].mmin;
        node[i].arg = node[RC(i)].arg;
    }else{
        node[i].mmin = node[LC(i)].mmin;
        node[i].arg = node[LC(i)].arg;
    }
}
int pos, val; //
void update(int L, int R, int i){
    if (L == R){node[i].mmin = val; node[i].arg = L; return;}
    int M = (L + R) / 2;
    if (pos <= M) update(L, M, LC(i));
    if (pos > M) update(M + 1, R, RC(i));
    if (node[LC(i)].mmin > node[RC(i)].mmin){
        node[i].mmin = node[RC(i)].mmin;
        node[i].arg = node[RC(i)].arg;
    }else{
        node[i].mmin = node[LC(i)].mmin;
        node[i].arg = node[LC(i)].arg;
    }
}
int x1, x2, res, arg; //          [x1,x2]
void query(int L, int R, int i){
    if (x1 <= L && R <= x2){
        if (res > node[i].mmin) res = node[i].mmin, arg = node[i].arg;
        return;
    }
    int M = (L + R) / 2;
    if (x1 <= M) query(L, M, LC(i));
    if (x2 > M) query(M + 1, R, RC(i));
}
/****************************************/
// Get index of the minimum depth, instead of the minimum height.
int lca_rmq(int u, int v){
    u = First[u], v = First[v];
    if (u > v) swap(u, v);
    x1 = u, x2 = v, res = INF, arg = u;
    query(0, cur, 1);
    return ver[arg];
}
int dis_rmq(int u, int v){
    return Dis[u] + Dis[v] - 2 * Dis[lca_rmq(u, v)];
}
void init_rmq(){
    dfs_rmq(); build(0, cur, 1);
}
```

1.11 Maxflow-Dinic

```
const int INF = 0x3f3f3f3f;

struct Edge {
	int from, to, cap, flow, index;
	Edge(int from, int to, int cap, int flow, int index):
		from(from), to(to), cap(cap), flow(flow), index(index) {}
};

struct Dinic {
	int N;
	vector<vector<Edge> > G;
	vector<Edge *> dad;
	vector<int> Q;

	Dinic(int N): N(N), G(N), dad(N), Q(N) {}

	void AddEdge(int from, int to, int cap) {
		G[from].push_back(Edge(from, to, cap, 0, G[to].size()));
		G[to].push_back(Edge(to, from, 0, 0, G[from].size() - 1));
	}
	int BlockingFlow(int s, int t) {
		fill(dad.begin(), dad.end(), (Edge*)NULL);
		dad[s] = &G[0][0] - 1;

		int head = 0, tail = 0;
		Q[tail++] = s;
		while(head < tail) {
			int x = Q[head++];
			for(int i = 0; i < G[x].size(); i++) {
				Edge &e = G[x][i];
				if(!dad[e.to] && e.cap - e.flow > 0) {
					dad[e.to] = &G[x][i];
					Q[tail++] = e.to;
				}
			}
		}
		if(!dad[t]) return 0;

		int totflow = 0;
		for(int i = 0; i < G[t].size(); i++) {
			Edge *start = &G[G[t][i].to][G[t][i].index];
			int amt = INF;
			for(Edge *e = start; amt && e != dad[s]; e = dad[e->from]) {
				if(!e) {amt = 0; break;}
				amt = min(amt, e->cap - e->flow);
			}
			if(amt == 0) continue;
			for(Edge *e = start; amt && e != dad[s]; e = dad[e->from]) {
				e->flow += amt;
				G[e->to][e->index].flow -= amt;
			}
			totflow += amt;
		}
```

```
            return totflow;
        }

        int GetFlow(int s, int t) {
                int totflow = 0;
                while(int flow = BlockingFlow(s, t))
                        totflow += flow;
                return totflow;
        }
};
```

1.12 Maxflow-EK

```
const int INF = 0x3f3f3f3f;
const int maxn = 1000 + 11;
int N, M, T, P, Q;
int kase = 0;
struct E {
    int u, v, cap, flow;
    E(){}
    E(int u, int v, int cap, int flow):u(u),v(v),cap(cap),flow(flow){}
};
vector<E> edge;
vector<int> G[maxn];
int Index[maxn], pi[maxn], btn[maxn]; bool vis[maxn];
int BFS(int s, int t){
    memset(vis, 0, sizeof(vis));
    memset(pi, -1, sizeof(pi));
    memset(btn, 0x3f, sizeof(btn));
    queue<int> Q; Q.push(s); pi[s] = s;
    while (!Q.empty()){
        int u = Q.front(); Q.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (int i = 0; i < G[u].size(); i++){
            E &e = edge[G[u][i]];
            if (!vis[e.v] && e.cap > 0){
                btn[e.v] = min(btn[u], e.cap);
                Index[e.v] = G[u][i];
                pi[e.v] = u; Q.push(e.v);
                if (e.v == t) return btn[t];
            }
        }
    }
    return 0;
}
int MF(int s, int t) {
    int F, dF;
    for (F = 0; (dF = BFS(s, t)) > 0; F += dF) {
        for (int i = t; i != s; i = pi[i]){
            int ind = Index[i];
            edge[ind].cap -= dF; edge[ind].flow += dF;
            edge[ind^1].cap += dF;
        }
    }
    return F;
}
```

```
void addEdge(int u, int v, int cap) {
    edge.push_back(E(u, v, cap, 0));
    edge.push_back(E(v, u, 0, 0));
    G[u].push_back(edge.size() - 2);
    G[v].push_back(edge.size() - 1);
}
```

1.13 MCMaxFlow

```
const int maxn = 1000 + 11;
int N, M, T;
int pi[maxn], Index[maxn], d[maxn]; bool vis[maxn];
struct E{
    int u, v, cap, flow, cost;
    E(){}
    E(int u, int v, int cap, int flow, int cost):u(u),v(v),cap(cap),flow(flow),cost(cost){}
};
vector<E> edge;
vector<int> G[maxn];
int Spfa(int s, int t){
    memset(vis, 0, sizeof(vis)); memset(d, 0x3f, sizeof(d));
    queue<int> Q; pi[s] = s; vis[s] = 1; d[s] = 0; Q.push(s);
    while(!Q.empty()){
        int u = Q.front(); Q.pop(); vis[u] = 0;
        for (int i = 0; i < G[u].size(); i++){
            E &e = edge[G[u][i]];
            if (e.cap > 0 && d[e.v] > d[u] + e.cost){
                pi[e.v] = u; Index[e.v] = G[u][i];
                d[e.v] = d[u] + e.cost;
                if (!vis[e.v]){
                    Q.push(e.v); vis[e.v] = 1;
                }
            }
        }
    }
    if (d[t] == INF) return 0;
    int dF = INF;
    for (int i = t; i != s; i = pi[i]) dF = min(dF, edge[Index[i]].cap);
    return dF;
}
int MCMF(int s, int t){
    int F, dF, Cost;
    for (F = 0, Cost = 0; (dF = Spfa(s, t)) > 0; F += dF, Cost += dF * d[t]){
        for (int j = t; j != s; j = pi[j]){
            int ind = Index[j];
            edge[ind].flow += dF; edge[ind].cap -= dF;
            edge[ind^1].cap += dF;
        }
    }
    return Cost;
}
void addEdge(int u, int v, int cap, int cost){
    edge.push_back(E(u, v, cap, 0, cost));
    edge.push_back(E(v, u, 0, 0, -cost));
    G[u].push_back(edge.size() - 2);
    G[v].push_back(edge.size() - 1);
}
```

1.14 Prim

```
const int INF = 0x3f3f3f3f;
const int maxn = 100000;
int M, N, T;
struct E {
    int u, v, w;
    E() {}
    E(int u, int v, int w):u(u),v(v),w(w){}
};
vector<E> edge;
vector<int> G[maxn];
struct D{
    int d, x;
    D(){}
    D(int d, int x):d(d), x(x){};
    bool operator < (const D& rhs) const{return d > rhs.d;}
};
vector<D> v;
bool vis[maxn];
int Prim(){
    v.clear();
    for (int i = 0; i < N; i++) v.push_back(D(INF, i));
    v[0].d = 0; int res = 0;
    memset(vis, 0, sizeof(vis));
    priority_queue<D> pq; pq.push(v[0]);
    while (!pq.empty()){
        D uu = pq.top(); pq.pop();
        vis[uu.x] = true, res += uu.d;
        for (int i = 0; i < G[uu.x].size(); i++) {
            E &e = edge[G[uu.x][i]];
            D &vv = v[e.v];
            if (vis[vv.x]) continue;
            if (vv.d > e.w) {
                vv.d = e.w; pq.push(vv);
            }
        }
    }
    return res;
}
```

1.15 SCC-Tarjan

```
const int maxn = 100000 + 11;
int N, M, T;
vector<int> G[maxn];
stack<int> stk;
bool in[maxn]; int low[maxn], ID[maxn], idx[maxn];
int cnt, id;
void Tarjan(int u){
    idx[u] = ++cnt; low[u] = cnt; in[u] = 1;
    stk.push(u);
    for (int v : G[u]) {
        if (idx[v] == -1){
            Tarjan(v);
            low[u] = min(low[u], low[v]);
        } else if (in[v]) low[u] = min(low[u], idx[v]);
    }
    if (idx[u] == low[u]) {
        in[u] = 0; ID[u] = id;
        while (stk.top() != u){
            in[stk.top()] = 0;
            ID[stk.top()] = id;
            stk.pop();
        }
        stk.pop(); id++;
    }
}
void solve(){
    memset(in, 0, sizeof(in)); memset(idx, -1, sizeof(idx)); memset(low, -1, sizeof(low));
    cnt = id = 0;
    for (int i = 0; i < N; i++){
        if (idx[i] == -1) Tarjan(i);
    }
}
```

1.16 Spfa

```
//Spfa is BellmanFord with queue acceleration.(Like BFS)
const int maxn = 1000 + 11;
int N, M, T;
struct E {
    int u, v, w;
    E() {}
    E(int u, int v, int w):u(u),v(v),w(w){}
};
vector<E> edge;
vector<int> G[maxn];
int d[maxn]; bool vis[maxn];
void Spfa(int s){
    memset(vis, 0, sizeof(vis));
    memset(d, 0x3f, sizeof(d));
    queue<int> Q; Q.push(s);
    d[s] = 0; vis[s] = 1;
    while (!Q.empty()){
        int u = Q.front(); Q.pop(); vis[u] = 0;
        for (int i = 0; i < G[u].size(); i++){
            E &e = edge[G[u][i]];
            if (d[e.v] > d[u] + e.w) {
                d[e.v] = d[u] + e.w;
                if (!vis[e.v]) {
                    vis[e.v] = 1;
                    Q.push(e.v);
                }
            }
        }
    }
}
```

1.17 TopoSort

```
const int maxn = 1e5 + 11;
int idg[maxn], q[maxn], qn; //idg is in-degree
//Return 0 if there is a cycle
bool topoSort(){
    qn = 0;
    for (int u = 0; u < N; u++) if (!idg[u]) q[qn++] = u;
    for (int i = 0; i < qn; i++){
        int u = q[i];
        for (int v : G[u]){
            idg[v]--;
            if (idg[v] == 0) q[qn++] = v;
        }
    }
    return qn == N;
}
```

1.18 VBCC-Tarjan

```
const int maxn = 1e5 + 11;
int N, M, T;
vector<pair<int,int> > edge;
vector<int> G[maxn];
vector<int> ebcc[maxn], vbcc[maxn];
stack<pair<int,int> > stk;
map<pair<int,int>, int> eid;
int low[maxn], idx[maxn], ID[maxn]; bool iscut[maxn];
int cnt, id;
void Tarjan(int u, int pu) {
    idx[u] = ++cnt; low[u] = cnt;
    int child = 0;
    for (int i = 0; i < G[u].size(); i++){
        int v = G[u][i];
        if (idx[v] == -1){
            stk.push({u, v}); child++;
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] >= idx[u]){
                iscut[u] = 1;
                pair<int,int> t;
                do{
                    t = stk.top(); stk.pop();
                    ebcc[id].push_back(eid[t]);
                    if (ID[t.first] != id) vbcc[id].push_back(t.first), ID[t.first] = id;
                    if (ID[t.second] != id) vbcc[id].push_back(t.second), ID[t.second] = id;
                }while (t.first != u || t.second != v);
                id++;
            }
        }else if (idx[v] < idx[u] && v != pu){
            stk.push({u, v});
            low[u] = min(low[u], idx[v]);
        }
    }
    if(pu < 0 && child == 1) iscut[u] = 0;
}
```

```
void solve() {
    memset(idx, -1, sizeof(idx)); memset(low, -1, sizeof(low)); memset(iscut, 0, sizeof(iscut));
        memset(ID, -1, sizeof(ID));
    cnt = id = 0;
    for (int i = 1; i <= N; i++) {
        if (idx[i] == -1) Tarjan(i, -1);
    }
}
```

2 DataStructure

2.1 AddMul-SegmentTree

```
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
using namespace std;
#define lowbit(x) ((x)&(-(x)))
#define MP make_pair
#define fi first
#define se second
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count()); //mt19937_64 for 64-bits
bool Finish_read;
template<class T>inline void read(T &x) {Finish_read = 0; x = 0; int f = 1; char ch = getchar();
    while(!isdigit(ch)) {if(ch == '-')f = -1; if(ch == EOF)return; ch = getchar();}
    while(isdigit(ch))x = x * 10 + ch - '0', ch = getchar(); x *= f; Finish_read = 1;}
typedef unsigned long long LL;
typedef pair<int,int> pii;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 2e5 + 11;

#define lson o<<1,l,mid
#define rson o<<1|1,mid+1,r
LL sum[maxn<<2],add[maxn<<2],mul[maxn<<2];
void build(int o,int l,int r) {
        add[o]=0,mul[o]=1;
        if (l==r) {sum[o]=0;return;}
        int mid=l+r>>1;
        build(lson);
        build(rson);
        sum[o]=sum[o<<1]+sum[o<<1|1];
}
inline void pushdown(int o,int len) {
        int lf=len-(len>>1),rg=len>>1;
        if (mul[o]^1) {
                add[o<<1]*=mul[o],mul[o<<1]*=mul[o],sum[o<<1]*=mul[o];
                add[o<<1|1]*=mul[o],mul[o<<1|1]*=mul[o],sum[o<<1|1]*=mul[o];
                mul[o]=1;
        }
        if (add[o]) {
                add[o<<1]+=add[o],sum[o<<1]+=lf*add[o];
                add[o<<1|1]+=add[o],sum[o<<1|1]+=rg*add[o];
                add[o]=0;
        }
}
```

```
void update(int o,int l,int r,int L,int R,LL v,int opt) {
        if (L<=l&&r<=R) {
                if (opt&1) add[o]*=v,mul[o]*=v,sum[o]*=v;
                else add[o]+=v,sum[o]+=(r-l+1)*v;
                return;
        }
        pushdown(o,r-l+1);
        int mid=l+r>>1;
        if (L<=mid) update(lson,L,R,v,opt);
        if (mid<R) update(rson,L,R,v,opt);
        sum[o]=sum[o<<1]+sum[o<<1|1];
}
LL query(int o,int l,int r,int L,int R) {
        if (L<=l&&r<=R) return sum[o];
        pushdown(o,r-l+1);
        int mid=l+r>>1;
        LL ret=0;
        if (L<=mid) ret+=query(lson,L,R);
        if (mid<R) ret+=query(rson,L,R);
        return ret;
}

int N, M, Q, T, tot;
int p[maxn], dep[maxn], ver[maxn<<1], id[maxn], top[maxn], sz[maxn], son[maxn];
vector<int> G[maxn];
void dfs1(int u = 1, int pu = 1, int d = 0){
    p[u] = pu; dep[u] = d; sz[u] = 1;
    int mx = 0;
    for (int v : G[u]){
        if (v != pu){
            dfs1(v, u, d + 1);
            sz[u] += sz[v];
            if (sz[v] > mx) mx = sz[v], son[u] = v;
        }
    }
}
void dfs2(int u = 1, int s = 1){
    id[u] = ++tot; ver[tot] = u; top[u] = s;
    if (son[u]) dfs2(son[u], s);
    for (int v : G[u]){
        if (v != p[u] && v != son[u]){
            dfs2(v, v);
        }
    }
}
LL Query_Sum(int u, int v){
    int fu = top[u], fv = top[v]; LL ret = 0;
    while (fu != fv){
        if (dep[fu] < dep[fv]) swap(fu, fv), swap(u, v);
        ret += query(1, 1, tot, id[fu], id[u]);
        u = p[fu], fu = top[u];
    }
    if (dep[u] < dep[v]) swap(u, v);
    ret += query(1, 1, tot, id[v], id[u]);
    return ret;
}
void Update(int u, int v, LL val, int opt){
    int fu = top[u], fv = top[v];
    while (fu != fv){
        if (dep[fu] < dep[fv]) swap(fu, fv), swap(u, v);
        update(1, 1, tot, id[fu], id[u], val, opt);
        u = p[fu], fu = top[u];
    }
    if (dep[u] < dep[v]) swap(u, v);
    update(1, 1, tot, id[v], id[u], val, opt);
}

int main(){
    while (scanf("%d", &N) != EOF) {
        tot = 0;
        memset(son, 0, sizeof(int)*(N+11));
        for (int i = 1; i <= N; i++) G[i].clear();
        for (int i = 2, fa; i <= N; i++){
            scanf("%d", &fa);
            G[fa].push_back(i);
        }
        dfs1(); dfs2();
        build(1, 1, tot);
        cin >> Q;
        int op, u, v; LL x;
        while (Q--){
            scanf("%d%d%d", &op, &u, &v);
            if (op == 1){
                scanf("%llu", &x);
                Update(u, v, x, 1);
            }else if (op == 2){
                scanf("%llu", &x);
                Update(u, v, x, 2);
            }else if (op == 3){
                Update(u, v, 1LL * -1, 1);
                Update(u, v, 1LL * -1, 2);
            }else{
                LL res = Query_Sum(u, v);
                printf("%llu\n", res);
            }
        }
    }
    return 0;
}

/*
7
1 1 1 2 2 4
5
2 5 6 1
1 1 6 2
4 5 6
3 5 2
4 2 2
2
1
4
3 1 2
4 1 2
3 1 1
4 1 1
7
1 1 1 2 2 4
```

```
5
2 5 6 1
1 1 6 2
4 5 6
3 5 2
4 2 2
2
1
4
3 1 2
4 1 2
3 1 1
4 1 1

*/
```

2.2 DynamicSegmentTree

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int maxn = 1e7 + 5e6;
int N, Q, M, tot, rt;
int lson[maxn], rson[maxn], sum[maxn], lazy[maxn];
int x1, x2, val;
void update(int L, int R, int& i) {
    if (!i) { i = ++tot; }
    if (x1 <= L && R <= x2) {sum[i] = val * (R - L + 1); lazy[i] = val; return;}
    int M = (L + R) / 2;
    if (lazy[i] != -1) {
        if (L != R){
            if (!lson[i]) { lson[i] = ++tot; }
            if (!rson[i]) { rson[i] = ++tot; }
            lazy[lson[i]] = lazy[rson[i]] = lazy[i];
            sum[lson[i]] = (M - L + 1) * lazy[i];
            sum[rson[i]] = (R - M) * lazy[i];
        }
        lazy[i] = -1;
    }
    if (x1 <= M) { update(L, M, lson[i]); }
    if (x2 > M) { update(M + 1, R, rson[i]); }
    sum[i] = sum[lson[i]] + sum[rson[i]];
}

int res;
void query(int L, int R, int& i) {
    if (x1 <= L && R <= x2) {res += sum[i]; return;}
    int M = (L + R) / 2;
    if (lazy[i] != -1) {
        if (!lson[i]) { lson[i] = ++tot; }
        if (!rson[i]) { rson[i] = ++tot; }
        lazy[lson[i]] = lazy[rson[i]] = lazy[i];
        sum[lson[i]] = (M - L + 1) * lazy[i];
        sum[rson[i]] = (R - M) * lazy[i];

        lazy[i] = -1;
    }
    if (x1 <= M) { query(L, M, lson[i]); }
    if (x2 > M) { query(M + 1, R, rson[i]); }
}

int main() {
    cin >> N >> Q;
    int l, r, k;
    memset(lazy, -1, sizeof(lazy));
    while (Q--) {
        scanf("%d%d%d", &l, &r, &k);
        x1 = l, x2 = r, val = 2 - k;
        update(1, N, rt);
        printf("%d\n", N - sum[1]);
    }
    return 0;
}
```

2.3 BIT-2D-range

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
using namespace std;
#define lowbit(x) ((x)&(-(x)))
#define MP make_pair
#define fi first
#define se second
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> pii;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 2048 + 11;
int read() {
    int num = 0; char c; bool flag = false;
    while ((c = getchar()) == ' ' || c == '\n' || c == '\r');
    if (c == '-') flag = true;
    else num = c - '0';
    while (isdigit(c = getchar())) num = num * 10 + c - '0';
    return (flag ? -1 : 1) * num;
}
int N, M, T; char op[5];
int bit1[maxn][maxn], bit2[maxn][maxn], bit3[maxn][maxn], bit4[maxn][maxn];
void add(int n, int m, int x){
    for (int i = n; i <= N; i += lowbit(i)) {
        for (int j = m; j <= M; j += lowbit(j)) {
            bit1[i][j] += x;
            bit2[i][j] += n*x;
            bit3[i][j] += m*x;
            bit4[i][j] += n*m*x;
        }
    }
}
int sum(int n, int m){
    int res = 0;
    for (int i = n; i > 0; i -= lowbit(i)) {
        for (int j = m; j > 0; j -= lowbit(j)) {
            res += (n+1)*(m+1)*bit1[i][j];
            res -= (m+1)*bit2[i][j];
            res -= (n+1)*bit3[i][j];
            res += bit4[i][j];
        }
    }
    return res;
}

int main(){
    scanf("%s%d%d", op, &N, &M);
    while (scanf("%s", op) != EOF){
        int a = read(), b = read(), c = read(), d = read();
        if (op[0] == 'L') {
            int x = read();
            add(a, b, x);
            add(a, d + 1, -x);
            add(c + 1, b, -x);
            add(c + 1, d + 1, x);
        } else {
            int ans = sum(c, d) - sum(a - 1, d) - sum(c, b - 1) + sum(a - 1, b - 1);
            printf("%d\n", ans);
        }
    }
    return 0;
}
/*
  \sum_{i=1}^x\sum_{j=1}^y (d[i][j]*(x-i+1)*(y-j+1))
= \sum_{i=1}^x\sum_{j=1}^y (d[i][j]*(x+1)*(y+1)d[i][j]*i*(y+1)d[i][j]*j*(x+1)+d[i][j]*i*j)
      :
d[i][j],d[i][j]*i,d[i][j]*j,d[i][j]*i*j
*/
```

2.4 BIT

```
//
//        :
//        :add(i, a[i]-a[i-1]) for all i
//     : sum(i)
//     : add(i, k), add(j + 1, -k)

int bit[maxn];
void add(int i, int x){
    while (i <= N) bit[i] += x, i += lowbit(i);
}
int sum(int i){
    int res = 0;
    while (i > 0) res += bit[i], i -= lowbit(i);
    return res;
}
```

2.5 BIT-Range

```
//d[i]: a[i]-a[i-1]
```

```
//bit1: d[i]
//bit2: i * d[i]
//a[n]: (n+1)*sum(d[i])-sum(i*d[i])
LL bit1[maxn], bit2[maxn]; int a[maxn], d[maxn];
void add(int i, int x, LL *bit){
    while (i <= N) bit[i] += 1LL * x, i += lowbit(i);
}
LL sum(int i, LL *bit){
    LL res = 0;
    while (i) res += 1LL * bit[i], i -= lowbit(i);
    return res;
}
int main(){
    cin >> N >> M;
    for (int i = 1; i <= N; i++) scanf("%d", &a[i]);
    adjacent_difference(a + 1, a + N + 1, d + 1);
    for (int i = 1; i <= N; i++) {
        add(i, d[i], bit1);
        add(i, 1LL*i*d[i], bit2);
    }
    int op, x, y, k;
    while (M--){
        scanf("%d", &op);
        if (op == 1){
            scanf("%d%d%d", &x, &y, &k);
            add(x, k, bit1); add(x, 1LL*x*k, bit2); //  x
            add(y + 1, -k, bit1); add(y + 1, -1LL*(y+1)*k, bit2); //  y + 1
        }else{
            scanf("%d%d", &x, &y);
            LL sy = 1LL * (y + 1) * sum(y, bit1) - sum(y, bit2);
            LL sx = 1LL * x * sum(x - 1, bit1) - sum(x - 1, bit2);
            printf("%lld\n", sy - sx);
        }
    }
    return 0;
}
```

2.6 DSU

```
map<int, int> *cnt[maxn];
void dfs(int v, int p){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
       if(u != p){
           dfs(u, v);
           if(sz[u] > mx)
               mx = sz[u], bigChild = u;
       }
    if(bigChild != -1)
        cnt[v] = cnt[bigChild];
    else
        cnt[v] = new map<int, int> ();
    (*cnt[v])[ col[v] ] ++;
    for(auto u : g[v])
       if(u != p && u != bigChild){
           for(auto x : *cnt[u])
               (*cnt[v])[x.first] += x.second;
```

```
      }
    //now (*cnt[v])[c] is the number of vertices in subtree of vertex v that has color c. You can
        answer the queries easily.
}
```

2.7 Hash-matching

```
#include<bits/stdc++.h>
#define MAXN 100005
using namespace std;
typedef unsigned long long ull;
const ull B=1000000007;
bool contain(string a,string b){
        int al=a.length(),bl=b.length();
        if(al>bl) return false;
        ull t=1;
        for(int i=0;i<al;i++)
                t*=B;
        ull ah=0,bh=0;
        for(int i=0;i<al;i++) ah=ah*B+a[i];
        for(int i=0;i<al;i++) bh=bh*B+b[i];
        for(int i=0;i+al<=bl;i++)
        {
                if(ah==bh) return true;
                if(i+al<bl) bh=bh*B+b[i+al]-b[i]*t;
        }
        return false;
}
```

2.8 HLD

```
const int maxn = 1e5 + 11;
int N, Q, T;
int p[maxn], dep[maxn], ver[maxn<<1], id[maxn], top[maxn], sz[maxn], son[maxn], a[maxn];
vector<int> G[maxn];
void dfs1(int u = 1, int pu = 1, int d = 0){
    p[u] = pu; dep[u] = d; sz[u] = 1;
    int mx = 0;
    for (int v : G[u]){
        if (v != pu){
            dfs1(v, u, d + 1);
            sz[u] += sz[v];
            if (sz[v] > mx) mx = sz[v], son[u] = v;
        }
    }
}
int tot;
void dfs2(int u = 1, int s = 1){
    id[u] = ++tot; ver[tot] = u; top[u] = s;
    if (son[u]) dfs2(son[u], s);
    for (int v : G[u]){
        if (v != p[u] && v != son[u]){
            dfs2(v, v);
        }
    }
```

```
}
int mx[maxn<<2], sum[maxn<<2];
void build(int L, int R, int o = 1) {
    if (L == R) { sum[o] = mx[o] = a[ver[L]]; return; }
    int M = (L + R) >> 1;
    build(L, M, LC(o));
    build(M + 1, R, RC(o));
    sum[o] = sum[LC(o)] + sum[RC(o)];
    mx[o] = max(mx[LC(o)], mx[RC(o)]);
}
int query_sum(int x1, int x2, int L = 1, int R = tot, int o = 1) {
    int res = 0;
    if (x1 <= L && R <= x2) return sum[o];
    int M = (L + R) >> 1;
    if (x1 <= M) res += query_sum(x1, x2, L, M, LC(o));
    if (x2 > M) res += query_sum(x1, x2, M + 1, R, RC(o));
    return res;
}
int query_max(int x1, int x2, int L = 1, int R = tot, int o = 1) {
    int res = -INF;
    if (x1 <= L && R <= x2) return mx[o];
    int M = (L + R) >> 1;
    if (x1 <= M) res = max(res, query_max(x1, x2, L, M, LC(o)));
    if (x2 > M) res = max(res, query_max(x1, x2, M + 1, R, RC(o)));
    return res;
}
int Query_Max(int u, int v){
    int fu = top[u], fv = top[v], ret = -INF;
    while (fu != fv){
        if (dep[fu] < dep[fv]) swap(fu, fv), swap(u, v);
        ret = max(ret, query_max(id[fu], id[u]));
        u = p[fu], fu = top[u];
    }
    if (dep[u] < dep[v]) swap(u, v);
    ret = max(ret, query_max(id[v], id[u]));
    return ret;
}
int Query_Sum(int u, int v){
    int fu = top[u], fv = top[v], ret = 0;
    while (fu != fv){
        if (dep[fu] < dep[fv]) swap(fu, fv), swap(u, v);
        ret += query_sum(id[fu], id[u]);
        u = p[fu], fu = top[u];
    }
    if (dep[u] < dep[v]) swap(u, v);
    ret += query_sum(id[v], id[u]);
    return ret;
}
void update(int pos, int val, int L = 1, int R = tot, int o = 1) {
    if (L == R) {sum[o] = mx[o] = val; return;}
    int M = (L + R) >> 1;
    if (pos <= M) update(pos, val, L, M, LC(o));
    if (pos > M) update(pos, val, M + 1, R, RC(o));
    mx[o] = max(mx[LC(o)], mx[RC(o)]);
    sum[o] = sum[LC(o)] + sum[RC(o)];
}
int main(){
    cin >> N;
    for (int i = 0, u, v; i < N - 1; i++){
        scanf("%d%d", &u, &v);
        G[u].push_back(v); G[v].push_back(u);
    }
    for (int i = 1; i <= N; i++) scanf("%d", &a[i]);
    dfs1(); dfs2();
    build(1, tot, 1);
    cin >> Q;
    char op[10]; int u, v;
    while (Q--){
        scanf("%s%d%d", op, &u, &v);
        if (op[0] == 'C'){
            update(id[u], v);
        }else if (op[1] == 'M'){
            printf("%d\n", Query_Max(u, v));
        }else{
            printf("%d\n", Query_Sum(u, v));
        }
    }
    return 0;
}
/*
4
1 2
2 3
4 1
4 2 1 3
12
QMAX 3 4
QMAX 3 3
QMAX 3 2
QMAX 2 3
QSUM 3 4
QSUM 2 1
CHANGE 1 5
QMAX 3 4
CHANGE 3 6
QMAX 3 4
QMAX 2 4
QSUM 3 4

*/
```

2.9 Kmp

```
#include <iostream>
#include <cstring>
using namespace std;
const int maxn = 1e5 + 11;
int Next[maxn];
int slen, tlen;
string S, T;
void getNext() {
    int j, k;
    j = 0; k = -1; Next[0] = -1;
    while (j < tlen){
        if (k == -1 || T[j] == T[k]) Next[++j] = ++k;
        else k = Next[k];
    }
}
int KMP_Index() {
    int i = 0, j = 0;
    getNext();
    while (i < slen && j < tlen) {
        if (j == -1 || S[i] == T[j]) i++, j++;
        else j = Next[j];
    }
    if (j == tlen) return i - tlen;
    else return -1;
}
int KMP_Count() {
    int ans = 0;
    int i, j = 0;
    if (slen == 1 && tlen == 1) {
        if (S[0] == T[0]) return 1;
        else return 0;
    }
    getNext();
    for (i = 0; i < slen; i++) {
        while (j > 0 && S[i] != T[j]) j = Next[j];
        if (S[i] == T[j]) j++;
        if (j == tlen) ans++, j = Next[j];
    }
    return ans;
}
int main() {

    int TT;
    int i, cc;
    cin >> TT;
    while (TT--) {
        cin >> S >> T;
        slen = S.size();
        tlen = T.size();
        cout << KMP_Index() << endl;
        cout << KMP_Count() << endl;
    }
    return 0;
}
/*
4
aaaaaa a
abcd d
aabaa b
*/
```

2.10 Mo

```
const int maxn = 5e4 + 11;
int N, M, S, n, col[maxn];
int blk[maxn]; LL cnt[maxn], ans1[maxn], ans2[maxn];
struct D{
    int l, r, id;
    bool operator < (const D& rhs) const{
        return blk[l] < blk[rhs.l] || (blk[l] == blk[rhs.l] && r < rhs.r);
    }
} q[maxn];
inline void upd(LL &sum, int c, int x){
    sum -= cnt[c] * cnt[c];
    cnt[c] += x;
    sum += cnt[c] * cnt[c];
}
int main(){
    cin >> N >> M; S = int(sqrt(N)) + 1;
    for (int i = 1; i <= N; i++) scanf("%d", &col[i]);
    for (int i = 1; i <= N; i++) blk[i] = (i - 1) / S;
    for (int i = 0, l, r; i < M; i++){
        scanf("%d%d", &l, &r);
        q[i] = {l, r, i};
    }
    sort(q, q + M);
    int L = 1, R = 0; LL sum = 0;
    for (int i = 0; i < M; i++){
        while (L < q[i].l) upd(sum, col[L++], -1);
        while (R > q[i].r) upd(sum, col[R--], -1);
        while (L > q[i].l) upd(sum, col[--L], 1);
        while (R < q[i].r) upd(sum, col[++R], 1);
        if (q[i].l == q[i].r) {
            ans1[q[i].id] = 0, ans2[q[i].id] = 1;
            continue;
        }
        ans1[q[i].id] = sum - (R - L + 1);
        ans2[q[i].id] = 1LL * (R - L + 1) * (R - L);
    }
    for (int i = 0; i < M; i++) {
        LL j = __gcd(ans1[i], ans2[i]);
        printf("%lld/%lld\n", ans1[i] / j, ans2[i] / j);
    }
    return 0;
}
```

2.11 PresidengTree

```
const int maxn = 1e5 + 11;
int Rank[maxn], root[maxn], cnt;
int N, Q;
struct Node{
    int sum, lson, rson;
    Node(){sum = lson = rson = 0;}
}T[maxn<<5];
int CreateNode(int sum, int lson, int rson){
    int idx = ++cnt;
    T[idx].sum = sum;
    T[idx].lson = lson;
    T[idx].rson = rson;
    return idx;
}
void Insert(int& root, int prt, int pos, int L, int R){
    root = CreateNode(T[prt].sum + 1, T[prt].lson, T[prt].rson);
    if (L == R) return;
    int M = (L + R) / 2;
    if (pos <= M) Insert(T[root].lson, T[prt].lson, pos, L, M);
    if (pos > M) Insert(T[root].rson, T[prt].rson, pos, M + 1, R);
}
int Query(int i, int j, int L, int R, int k){
    if (L == R) return L;
    int M = (L + R) / 2;
    int sum = T[T[j].lson].sum - T[T[i].lson].sum;
    if (k <= sum) return Query(T[i].lson, T[j].lson, L, M, k);
    else return Query(T[i].rson, T[j].rson, M + 1, R, k - sum);
}
int main() {
    cin >> N >> Q;
    for (int i = 1; i <= N; i++) scanf("%d", &a[i].first), a[i].second = i;
    sort(a + 1, a + N + 1);
    for (int i = 1; i <= N; i++) Rank[a[i].second] = i;
    cnt = root[0] = 0;
    for (int i = 1; i <= N; i++){
        Insert(root[i], root[i - 1], Rank[i], 1, N);
    }
    while (Q--){
        int l, r, k;
        scanf("%d%d%d", &l, &r, &k);
        int pos = Query(root[l - 1], root[r], 1, N, k);
        printf("%d\n", a[pos].first);
    }
    return 0;
}
```

2.12 SegmentTree

```
// to replace or to modify!
#define LC(i) ((i)*2)
#define RC(i) ((i)*2+1)
const int maxn = 1e5 + 11;
int sum[maxn<<2], a[maxn];
int N, M, T;
void build(int L = 1, int R = N, int o = 1) {
    if (L == R) {sum[o] = a[L]; return;}
    int M = (L + R) >> 1;
    build(L, M, LC(o));
    build(M + 1, R, RC(o));
    sum[o] = sum[LC(o)] + sum[RC(o)];
}
void update(int pos, int val, int L = 1, int R = N, int o = 1) {
    if (L == R) { sum[o] += val; return; }
    int M = (L + R) >> 1;
    if (pos <= M) update(pos, val, L, M, LC(o));
    if (pos > M) update(pos, val, M + 1, R, RC(o));
    sum[o] = sum[LC(o)] + sum[RC(o)];
}
int query(int x1, int x2, int L = 1, int R = N, int o = 1) {
    if (x1 <= L && R <= x2) return sum[o];
    int res = 0, M = (L + R) >> 1;
    if (x1 <= M) res += query(x1, x2, L, M, LC(o));
    if (x2 > M) res += query(x1, x2, M + 1, R, RC(o));
    return res;
}
```

2.13 SegmentTree-Range

```
// to replace or to modify!
#define LC(i) ((i)*2)
#define RC(i) ((i)*2+1)
const int maxn = 1e5 + 11;
int N, M, T;
int a[maxn];
int sum[maxn << 2], lazy[maxn << 2];
void build(int L, int R, int o = 1) {
    lazy[o] = 0;
    if (L == R) { sum[o] = a[L]; return; }
    int M = (L + R) >> 1;
    build(L, M, LC(o));
    build(M + 1, R, RC(o));
    sum[o] = sum[LC(o)] + sum[RC(o)];
}
inline void pushdown(int L, int M, int R, int o) {
    lazy[LC(o)] += lazy[o]; lazy[RC(o)] += lazy[o];
    sum[LC(o)] += (M - L + 1) * lazy[o]; sum[RC(o)] += (R - M) * lazy[o];
    lazy[o] = 0;
}
void update(int x1, int x2, int val, int L = 1, int R = N, int o = 1) {
    if (x1 <= L && R <= x2) {sum[o] += val * (R - L + 1); lazy[o] += val; return;}
    int M = (L + R) >> 1;
    if (lazy[o]) pushdown(L, M, R, o);
    if (x1 <= M) update(x1, x2, val, L, M, LC(o));
    if (x2 > M) update(x1, x2, val, M + 1, R, RC(o));
    sum[o] = sum[LC(o)] + sum[RC(o)];
}
int query(int x1, int x2, int L = 1, int R = N, int o = 1) {
    if (x1 <= L && R <= x2) return sum[o];
    int res = 0, M = (L + R) >> 1;
    if (lazy[o]) pushdown(L, M, R, o);
    if (x1 <= M) res += query(x1, x2, L, M, LC(o));
    if (x2 > M) res += query(x1, x2, M + 1, R, RC(o));
    return res;
}
```

2.14 SlidingWindow

```
int N, M, K;
int a[maxn], X[maxn], x[maxn];
int qX[maxn], qx[maxn], s1, s2, t1, t2;
int main() {
    cin >> N >> K;
    for (int i = 0; i < N; i++) scanf("%d", &a[i]);
    for (int i = 0; i < N; i++) {
        while (s1 < t1 && a[qX[t1 - 1]] < a[i]) t1--;
        qX[t1++] = i;
        if (i - K + 1 >= 0) X[i - K + 1] = a[qX[s1]];
        if (i - K + 1 == qX[s1]) s1++;

        while (s2 < t2 && a[qx[t2 - 1]] > a[i]) t2--;
        qx[t2++] = i;
        if (i - K + 1 >= 0) x[i - K + 1] = a[qx[s2]];
        if (i - K + 1 == qx[s2]) s2++;
    }
    for (int i = 0; i + K - 1 < N; i++) printf("%d%c", x[i], " \n"[i + K - 1 == N - 1]);
    for (int i = 0; i + K - 1 < N; i++) printf("%d%c", X[i], " \n"[i + K - 1 == N - 1]);
    return 0;
}

/*
1
10 6 10 5 5 5 5
3 2 2 1 5 7 6 8 2 9

*/
```

2.15 SparseTable

```
int mx[maxn][maxb], mi[maxn][maxb], a[maxn];
//Here is [1,N], [0,N) is also ok.
//We can also make 2 dimentions by mx[maxn][maxn][maxb] (square) or mx[maxn][maxb][maxn][maxb]
     (rectangle)
void pre() {
    for (int j = 0; j < maxb; j++) {
        for (int i = 1; i <= N; i++) {
            if (i + (1 << j) - 1 <= N) {
                mx[i][j] = (j ? max(mx[i][j - 1], mx[i + (1 << (j - 1))][j - 1]) : a[i]);
                mi[i][j] = (j ? min(mi[i][j - 1], mi[i + (1 << (j - 1))][j - 1]) : a[i]);
            }
        }
    }
}
int query(int l, int r) {
    int j = (int)(log2(r - l + 1));
    return max(mx[l][j], mx[r - (1 << j) + 1][j]) - min(mi[l][j], mi[r - (1 << j) + 1][j]);
}
```

2.16 Splay

```
#include<bits/stdc++.h>
using namespace std;
#define MAXN 1000000
int ch[MAXN][2], f[MAXN], sz[MAXN], cnt[MAXN], key[MAXN];
int n, root;
inline void _clear(int x) {
    ch[x][0] = ch[x][1] = f[x] = sz[x] = cnt[x] = key[x] = 0;
}
inline bool _get(int x) {
    return ch[f[x]][1] == x;
}
inline void _update(int x) {
    if (x) {
        sz[x] = cnt[x];
        if (ch[x][0]) sz[x] += sz[ch[x][0]];
        if (ch[x][1]) sz[x] += sz[ch[x][1]];
    }
}
```

```
inline void _rotate(int x) {
    int old = f[x], oldf = f[old], whichx = _get(x);
    ch[old][whichx] = ch[x][whichx ^ 1]; f[ch[old][whichx]] = old;
    ch[x][whichx ^ 1] = old; f[old] = x;
    f[x] = oldf;
    if (oldf)
        ch[oldf][ch[oldf][1] == old] = x;
    _update(old); _update(x);
}
inline void splay(int x) {
    for (int fa; fa = f[x]; _rotate(x))
        if (f[fa])
            _rotate((_get(x) == _get(fa)) ? fa : x);
    root = x;
}
inline void _insert(int x) {
    if (root == 0) {
        n++;
        ch[n][0] = ch[n][1] = f[n] = 0;
        root = n;
        sz[n] = cnt[n] = 1;
        key[n] = x;
        return;
    }
    int now = root, fa = 0;
    while(1) {
        if (x == key[now]) {
            cnt[now]++; _update(now); _update(fa); splay(now); break;
        }
        fa = now;
        now = ch[now][key[now] < x];
        if (now == 0) {
            n++;
            ch[n][0] = ch[n][1] = 0;
            f[n] = fa;
            sz[n] = cnt[n] = 1;
            ch[fa][key[fa] < x] = n;
            key[n] = x;
            _update(fa);
            splay(n);
            break;
        }
    }
}
inline int _find(int x) {
    int now = root, ans = 0;
    while(1) {
        if (x < key[now])
            now = ch[now][0];
        else {
            ans += (ch[now][0] ? sz[ch[now][0]] : 0);
            if (x == key[now]) {
                splay(now); return ans + 1;
            }
            ans += cnt[now];
            now = ch[now][1];
        }
    }
}
```

```
inline int _findx(int x) {
    int now = root;
    while(1) {
        if (ch[now][0] && x <= sz[ch[now][0]])
            now = ch[now][0];
        else {
            int temp = (ch[now][0] ? sz[ch[now][0]] : 0) + cnt[now];
            if (x <= temp) return key[now];
            x -= temp; now = ch[now][1];
        }
    }
}
inline int _pre() {
    int now = ch[root][0];
    while (ch[now][1]) now = ch[now][1];
    return now;
}
inline int _next() {
    int now = ch[root][1];
    while (ch[now][0]) now = ch[now][0];
    return now;
}
inline void _del(int x) {
    int whatever = _find(x);
    if (cnt[root] > 1) {
        cnt[root]--;
        _update(root);
        return;
    }
    if (!ch[root][0] && !ch[root][1]) {
        _clear(root);
        root = 0;
        return;
    }
    if (!ch[root][0]) {
        int oldroot = root; root = ch[root][1]; f[root] = 0; _clear(oldroot); return;
    }
    else if (!ch[root][1]) {
        int oldroot = root; root = ch[root][0]; f[root] = 0; _clear(oldroot); return;
    }
    int leftbig = _pre(), oldroot = root;
    splay(leftbig);
    ch[root][1] = ch[oldroot][1];
    f[ch[oldroot][1]] = root;
    _clear(oldroot);
    _update(root);
}
int main() {
    int n, opt, x;
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d%d", &opt, &x);
        switch(opt) {
        case 1: _insert(x); break;
        case 2: _del(x); break;
        case 3: printf("%d\n", _find(x)); break;
        case 4: printf("%d\n", _findx(x)); break;
        case 5: _insert(x); printf("%d\n", key[_pre()]); _del(x); break;
        case 6: _insert(x); printf("%d\n", key[_next()]); _del(x); break;
        }
    }
}
```

2.17 SuffixArray

```
const int maxn = 1e5 + 11;
int N, M, T, P, Q;
int x[maxn], y[maxn], buc[maxn], Rank[maxn], Height[maxn];
int SA[maxn], s[maxn];
int cmp(int *s, int a, int b, int l) {
    return s[a] == s[b] && s[a + l] == s[b + l];
}
// m: the number of the biggest char + 1!!!
// n: length of the string to int
// Rank,SA:0-N-1, Height:1-N-1
void da(int *s, int n, int m) {
    for (int i = 0; i < m; i++) buc[i] = 0;
    for (int i = 0; i < n; i++) buc[x[i] = s[i]]++;
    for (int i = 1; i < m; i++) buc[i] += buc[i - 1];
    for (int i = n - 1; i >= 0; i--) SA[--buc[x[i]]] = i;
    for (int k = 1; k <= n; k <<= 1){
        int p = 0;
        for (int i = n - 1; i >= n - k; i--) y[p++] = i;
        for (int i = 0; i < n; i++) if (SA[i] >= k) y[p++] = SA[i] - k;
        for (int i = 0; i < m; i++) buc[i] = 0;
        for (int i = 0; i < n; i++) buc[x[y[i]]]++;
        for (int i = 1; i < m; i++) buc[i] += buc[i - 1];
        for (int i = n - 1; i >= 0; i--) SA[--buc[x[y[i]]]] = y[i];
        swap(x, y);
        p = 1; x[SA[0]] = 0;
        for (int i = 1; i < n; i++) {
            x[SA[i]] = cmp(y, SA[i - 1], SA[i], k) ? p - 1 : p++;
        }
        if (p >= n) break;
        m = p;
    }
}
void build_height(int *s, int n) {
    int k = 0;
    for (int i = 0; i < n; i++) Rank[SA[i]] = i;
    for (int i = 0; i < n; i++) {
        if (Rank[i] == 0) {Height[0] = 0; continue;}
        if (k) k--;
        int j = SA[Rank[i] - 1];
        while (s[i + k] == s[j + k] && i + k < n && j + k < n) k++;
        Height[Rank[i]] = k;
    }
}
bool contain(string S, string T){
    int l = 0, r = S.length();
    while (l < r){
        int m = (l + r) / 2;
        if (S.compare(SA[m], T.length(), T) < 0) l = m + 1;
        else r = m;
    }
    return S.compare(SA[r], T.length(), T) == 0;
}
int main() {
    string A; cin >> A; N = A.length();
    for (int i = 0; i < N; i++) s[i] = A[i] - 'a' + 1;
    da(s, N, 26 + 1);
    build_height(s, N);
    return 0;
}
```

2.18 SuffixArrayDC3

```
#define F(o) ((o)/3+((o)%3==1?0:tb))
#define G(o) ((o)<tb?(o)*3+1:((o)-tb)*3+2)
int N, M, T, P, Q;
const int maxn = 1e5 + 11;
char S[maxn];
int buc[maxn], x[maxn], y[maxn], z[maxn], Rank[maxn], Height[maxn];
int s[3*maxn], SA[3*maxn];
int c0(int *s, int a, int b) {
    return s[a] == s[b] && s[a + 1] == s[b + 1] && s[a + 2] == s[b + 2];
}
int c12(int k, int *s, int a, int b) {
    if (k == 2) return s[a] < s[b] || s[a] == s[b] && c12(1, s, a + 1, b + 1);
    else return s[a] < s[b] || s[a] == s[b] && z[a + 1] < z[b + 1];
}
void Sort(int *s, int *a, int *b, int n, int m) {
    int i;
    for (i = 0; i < n; i++) z[i] = s[a[i]];
    for (i = 0; i < m; i++) buc[i] = 0;
    for (i = 0; i < n; i++) buc[z[i]]++;
    for (i = 1; i < m; i++) buc[i] += buc[i - 1];
    for (i = n - 1; i >= 0; i--) b[--buc[z[i]]] = a[i];
    return;
}
void dc3(int *s, int *SA, int n, int m) {
    int i, j, *rn = s + n, *san = SA + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
    s[n] = s[n + 1] = 0;
    for (i = 0; i < n; i++) if (i % 3 != 0) x[tbc++] = i;
    Sort(s + 2, x, y, tbc, m);
    Sort(s + 1, y, x, tbc, m);
    Sort(s, x, y, tbc, m);
    for (p = 1, rn[F(y[0])] = 0, i = 1; i < tbc; i++)
        rn[F(y[i])] = c0(s, y[i - 1], y[i]) ? p - 1 : p++;
    if (p < tbc) dc3(rn, san, tbc, p);
    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
    for (i = 0; i < tbc; i++) if (san[i] < tb) y[ta++] = san[i] * 3;
    if (n % 3 == 1) y[ta++] = n - 1;
    Sort(s, y, x, ta, m);
    for (i = 0; i < tbc; i++) z[y[i] = G(san[i])] = i;
    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
        SA[p] = c12(y[j] % 3, s, x[i], y[j]) ? x[i++] : y[j++];
    for (; i < ta; p++) SA[p] = x[i++];
    for (; j < tbc; p++) SA[p] = y[j++];
    return;
}
void build_height(int *s, int n) {
    int k = 0;
    for (int i = 0; i < n; i++) Rank[SA[i]] = i;
    for (int i = 0; i < n; i++) {
        if (Rank[i] == 0) {Height[0] = 0; continue;}
        if (k) k--;
        int j = SA[Rank[i] - 1];
        while (s[i + k] == s[j + k] && i + k < n && j + k < n) k++;
        Height[Rank[i]] = k;
    }
}
void init(int *s, int n){
    s[n] = 0;
    dc3(s, SA, n + 1, *max_element(s, s + n) + 1);
    for (int i = 0; i < n; i++) swap(SA[i], SA[i + 1]);
    build_height(s, n);
}
```

2.19 Treap-all

```
// luogu-judger-enable-o2
#pragma GCC optimize(3)
#include<bits/stdc++.h>
inline unsigned RAND() {
    static unsigned x = 123456789;
    static unsigned y = 362436069;
    static unsigned z = 521288629;
    static unsigned w = 88675123;
    unsigned t;
    t = x ^ (x << 11);
    x = y; y = z; z = w;
    return w = w ^ (w >> 19) ^ (t ^ (t >> 8));
}

inline int max(int x,int y){if (x>y) return x;return y;}
inline int min(int x,int y){if (x<y) return x;return y;}
inline void swap(int &a,int &b){a^=b,b^=a,a^=b;}

typedef long long LL;
typedef unsigned long long ULL;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 5e5 + 11;
int N, M, T, a[maxn], root, tot;
int sz[maxn], ch[maxn][2], rnd[maxn], val[maxn], sum[maxn];
int lmx[maxn], rmx[maxn], mx[maxn];
bool lazy_rev[maxn], lazy_mod[maxn];
void readin() {
    srand(19990130);
    std::cin >> N >> M;
    tot = root = 0;
    for (int i = 1; i <= N; i++) scanf("%d", &a[i]);
}
//-fsanitize=undefined

std::queue<int> trashcan;
void trash(int x){
    if (!x) return;
    trashcan.push(x);
    trash(ch[x][0]);
    trash(ch[x][1]);
    sz[x]=ch[x][0]=ch[x][1]=val[x]=sum[x]=lmx[x]=rmx[x]=mx[x]=lazy_mod[x]=lazy_rev[x]=0;
}
inline int newNode(int v) {
    int x = 0;
    if (!trashcan.empty()) x = trashcan.front(), trashcan.pop();
    else x = ++tot;
    sz[x] = 1; rnd[x] = rand();
    sum[x] = val[x] = v;
    lmx[x] = rmx[x] = max(v, 0);
    mx[x] = v;
    return x;
}
inline void Update(int x) {
    if (ch[x][0] && ch[x][1]) {
        sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + 1;
        sum[x] = sum[ch[x][0]] + sum[ch[x][1]] + val[x];
        mx[x] = max(mx[ch[x][0]], mx[ch[x][1]]);
        mx[x] = max(mx[x], rmx[ch[x][0]] + val[x] + lmx[ch[x][1]]);
        lmx[x] = max(lmx[ch[x][0]], sum[ch[x][0]] + val[x] + lmx[ch[x][1]]);
        rmx[x] = max(rmx[ch[x][1]], sum[ch[x][1]] + val[x] + rmx[ch[x][0]]);
    } else if (ch[x][0]) {
        sz[x] = sz[ch[x][0]] + 1;
        sum[x] = sum[ch[x][0]] + val[x];
        mx[x] = max(mx[ch[x][0]], rmx[ch[x][0]] + val[x]);
        lmx[x] = max(lmx[ch[x][0]], sum[ch[x][0]] + val[x]);
        lmx[x] = max(0, lmx[x]);
        rmx[x] = max(0, val[x] + rmx[ch[x][0]]);
    } else if (ch[x][1]) {
        sz[x] = sz[ch[x][1]] + 1;
        sum[x] = sum[ch[x][1]] + val[x];
        mx[x] = max(mx[ch[x][1]], lmx[ch[x][1]] + val[x]);
        rmx[x] = max(rmx[ch[x][1]], sum[ch[x][1]] + val[x]);
        rmx[x] = max(0, rmx[x]);
        lmx[x] = max(0, lmx[ch[x][1]] + val[x]);
    } else {
        sz[x] = 1, sum[x] = mx[x] = val[x];
        lmx[x] = rmx[x] = max(val[x], 0);
    }
}
inline void pushDown(int x) {
    int& ls = ch[x][0];
    int& rs = ch[x][1];
    if (lazy_rev[x]) {
        swap(ls, rs);
        if (ls) lazy_rev[ls] ^= 1, swap(lmx[ls], rmx[ls]);
        if (rs) lazy_rev[rs] ^= 1, swap(lmx[rs], rmx[rs]);
        lazy_rev[x] = 0;
    }
    if (lazy_mod[x]) {
        if (ls)
            lazy_mod[ls]=1,val[ls]=val[x],sum[ls]=val[x]*sz[ls],mx[ls]=max(sum[ls],val[ls]),lmx[ls]=rmx[ls]=max(sum
        if (rs)
            lazy_mod[rs]=1,val[rs]=val[x],sum[rs]=val[x]*sz[rs],mx[rs]=max(sum[rs],val[rs]),lmx[rs]=rmx[rs]=max(sum
        lazy_mod[x] = 0;
    }
}
```

```
void Split(int now, int k, int &x, int &y) {
    if (!now) {
        x = y = 0;
        return;
    } else { // <= to left, > to right
        pushDown(now);
        if (sz[ch[now][0]] + 1 <= k) x = now, Split(ch[now][1], k - sz[ch[now][0]] - 1, ch[now][1],
             y);
        else y = now, Split(ch[now][0], k, x, ch[now][0]);
        Update(now);
        // Split "by position"
    }
}
int Merge(int x, int y) {
    if (!x || !y) return x + y;
    if (rnd[x] < rnd[y]) {
        pushDown(x);
        ch[x][1] = Merge(ch[x][1], y);
        Update(x);
        return x;
    } else {
        pushDown(y);
        ch[y][0] = Merge(x, ch[y][0]);
        Update(y);
        return y;
    }
}
int build(int L = 1, int R = N) {
    if (L > R) return 0;
    int M = (L + R) / 2;
    int now = newNode(a[M]);
    ch[now][0] = build(L, M - 1);
    ch[now][1] = build(M + 1, R);
    Update(now);
    return now;
}
inline void Insert(int pos, int n) {
    int x, y;
    Split(root, pos, x, y);
//   printf("%d %d\n", x, y);
    int new_root = build(1, n);
    root = Merge(Merge(x, new_root), y);
}
inline void Del(int pos, int n) {
    int x, y, z;
    Split(root, pos + n - 1, x, z);
    Split(x, pos - 1, x, y);
    root = Merge(x, z);
    trash(y);
}
inline void Rev(int pos, int n) {
    int x, y, z;
    Split(root, pos + n - 1, x, z);
    Split(x, pos - 1, x, y);
    lazy_rev[y] ^= 1, swap(lmx[y], rmx[y]);
    root = Merge(Merge(x, y), z);
}
inline void Modify(int pos, int n, int c) {
    int x, y, z;
    Split(root, pos + n - 1, x, z);
    Split(x, pos - 1, x, y);
    lazy_mod[y]=1,val[y]=c,sum[y]=sz[y]*c,mx[y]=max(sum[y],val[y]),lmx[y]=rmx[y]=max(sum[y],0);
    root = Merge(Merge(x, y), z);
}
inline int getSum(int pos, int n){
    int x, y, z;
    Split(root, pos + n - 1, x, z);
    Split(x, pos - 1, x, y);
    int res = sum[y];
    root = Merge(Merge(x, y), z);
    return res;
}
int main() {
//    freopen("in.txt", "r", stdin);
    readin();
    root = build(1, N);
    char op[20]; int pos, n, c;
    while (M--) {
        scanf("%s", op);
        switch (op[0]) {
            case 'I':
                scanf("%d%d", &pos, &n);
                for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
                Insert(pos, n);
                break;
            case 'D':
                scanf("%d%d", &pos, &n);
                for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
                Del(pos, n);
                break;
            case 'R':
                scanf("%d%d", &pos, &n);
                Rev(pos, n);
                break;
            case 'G':
                scanf("%d%d", &pos, &n);
                printf("%d\n", getSum(pos, n));
                break;
            case 'M':
                if (op[2] == 'X'){
                    printf("%d\n", mx[root]);
                } else if (op[2] == 'K'){
                    scanf("%d%d%d", &pos, &n, &c);
                    Modify(pos, n, c);
                }
                break;
        }
    }
    return 0;
}
/*
15 36
-6 -5 -6 -5 -3 -3 -20 -100 -1 -50 -2 -30 -3 -20 -100 -1
GET-SUM 5 4
MAX-SUM
INSERT 8 3 -5 7 2
MAKE-SAME 5 3 -10
DELETE 12 1
DELETE 2 5
DELETE 10 8
MAX-SUM
MAKE-SAME 3 3 2
REVERSE 3 6
GET-SUM 5 4
INSERT 5 4 -10 -20 -30 -40
MAX-SUM
DELETE 10 1
MAX-SUM
DELETE 12 1
DELETE 2 5
MAKE-SAME 3 3 2
REVERSE 3 6
GET-SUM 5 4
INSERT 2 4 -4 -3 -2 -5
DELETE 10 1
MAX-SUM
REVERSE 2 10
GET-SUM 7 10
DELETE 2 5
MAKE-SAME 3 3 2
REVERSE 3 6
GET-SUM 5 4
INSERT 10 4 -107 -207 -370 -470
MAX-SUM
DELETE 10 1
MAX-SUM
DELETE 12 1
DELETE 2 5
MAX-SUM

*/
```

2.20 Treap

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
using namespace std;
#define lowbit(x) ((x)&(-(x)))
#define MP make_pair
#define fi first
#define se second
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int, int> pii;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 1e5 + 11;
inline unsigned RAND() {
    static unsigned x = 123456789;
    static unsigned y = 362436069;
    static unsigned z = 521288629;
    static unsigned w = 88675123;
    unsigned t;
    t = x ^ (x << 11);
    x = y; y = z; z = w;
    return w = w ^ (w >> 19) ^ (t ^ (t >> 8));
}
inline int read() {
    int num = 0; char c; bool flag = false;
    while ((c = getchar()) == ' ' || c == '\n' || c == '\r');
    if (c == '-') flag = true;
    else num = c - '0';
    while (isdigit(c = getchar())) num = num * 10 + c - '0';
    return (flag ? -1 : 1) * num;
}
int N, M, T, root, tot;
int sz[maxn], ch[maxn][2], rnd[maxn], val[maxn];
inline int newNode(int x) {
    sz[++tot] = 1; val[tot] = x;
    rnd[tot] = rand();
    return tot;
}
inline void Update(int x) {
    sz[x] = 1 + sz[ch[x][0]] + sz[ch[x][1]];
}
void Split(int now, int k, int &x, int &y) {
    if (!now) {
        x = y = 0;
        return;
    } else { // <= to left, > to right
        if (val[now] <= k) x = now, Split(ch[now][1], k, ch[now][1], y);
        else y = now, Split(ch[now][0], k, x, ch[now][0]);
        Update(now);
        // The split is "by value", we can also change it to "by size".
    }
}
/*
void Split(int now, int k, int &x, int &y) {
    if (!now) {
        x = y = 0;
        return;
    } else { // <= to left, > to right
        pushDown(now);
        if (sz[ch[now][0]] + 1 <= k) x = now, Split(ch[now][1], k - sz[ch[now][0]] - 1, ch[now][1],
            y);
        else y = now, Split(ch[now][0], k, x, ch[now][0]);
        Update(now);
        // Split "by position"
    }
}
*/
int Merge(int x, int y) {
    if (!x || !y) return x + y;
    if (rnd[x] < rnd[y]) {
        ch[x][1] = Merge(ch[x][1], y);
        Update(x);
        return x;
    } else {
        ch[y][0] = Merge(x, ch[y][0]);
        Update(y);
        return y;
    }
}
```

```
inline int Kth(int now, int k) {
    while (1) {
        if (k <= sz[ch[now][0]]) now = ch[now][0];
        else if (k == sz[ch[now][0]] + 1) return now;
        else k -= sz[ch[now][0]] + 1, now = ch[now][1];
    }
}
inline int Rank(int v) {
    int x, y;
    Split(root, v - 1, x, y);
    int res = sz[x] + 1;
    root = Merge(x, y);
    return res;
}
inline void Insert(int v) {
    int x, y;
    Split(root, v, x, y);
    root = Merge(Merge(x, newNode(v)), y);
}
inline void Del(int v) {
    int x, y, z;
    Split(root, v, x, z);
    Split(x, v - 1, x, y);
    y = Merge(ch[y][0], ch[y][1]);
    root = Merge(Merge(x, y), z);
}
inline int Pre(int v) {
    int x, y;
    Split(root, v - 1, x, y);
    int res = val[Kth(x, sz[x])];
    root = Merge(x, y);
    return res;
}
inline int Next(int v) {
    int x, y;
    Split(root, v, x, y);
    int res = val[Kth(y, 1)];
    root = Merge(x, y);
    return res;
}
int main() {
    srand(time(NULL));
    N = read();
    for (int i = 1, opt, x; i <= N; i++) {
        opt = read(); x = read();
        switch(opt) {
            case 1:
                Insert(x);
                break;
            case 2:
                Del(x);
                break;
            case 3:
                printf("%d\n", Rank(x));
                break;
            case 4:
                printf("%d\n", val[Kth(root, x)]);
                break;
            case 5:
                printf("%d\n", Pre(x));
                break;
            case 6:
                printf("%d\n", Next(x));
                break;
        }
    }
}
/*
10
1 106465
4 1
1 317721
1 460929
1 644985
1 84185
1 89851
6 81968
1 492737
5 493598

*/
```

2.21 Treap-Range

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
using namespace std;
#define lowbit(x) ((x)&(-(x)))
#define MP make_pair
#define fi first
#define se second
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int, int> pii;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 1e5 + 11;
int N, M, T, root, tot;
int sz[maxn], ch[maxn][2], rnd[maxn], val[maxn];
bool lazy[maxn];
inline int read() {
    int num = 0; char c; bool flag = false;
    while ((c = getchar()) == ' ' || c == '\n' || c == '\r');
    if (c == '-') flag = true;
    else num = c - '0';
    while (isdigit(c = getchar())) num = num * 10 + c - '0';
    return (flag ? -1 : 1) * num;
}
inline int newNode(int x) {
    sz[++tot] = 1; val[tot] = x;
    rnd[tot] = rand();
    return tot;
}
inline void Update(int x) {
    sz[x] = 1 + sz[ch[x][0]] + sz[ch[x][1]];
```

```
}
inline void pushDown(int x) {
    if (x && lazy[x]) {
        swap(ch[x][0], ch[x][1]);
        if (ch[x][0]) lazy[ch[x][0]] ^= 1;
        if (ch[x][1]) lazy[ch[x][1]] ^= 1;
        lazy[x] = 0;
    }
}
void Split(int now, int k, int &x, int &y) {
    if (!now) { x = y = 0; return; }
    pushDown(now); // <= to left, > to right
    if (sz[ch[now][0]] + 1 <= k) x = now, Split(ch[now][1], k - sz[ch[now][0]] - 1, ch[now][1], y);
    else y = now, Split(ch[now][0], k, x, ch[now][0]);
    Update(now);
}
int Merge(int x, int y) {
    if (!x || !y) return x + y;
    if (rnd[x] < rnd[y]) {
        pushDown(x);
        ch[x][1] = Merge(ch[x][1], y);
        Update(x);
        return x;
    } else {
        pushDown(y);
        ch[y][0] = Merge(x, ch[y][0]);
        Update(y);
        return y;
    }
}
void Reverse(int l, int r) {
    int x, y, z;
    Split(root, r, x, z);
    Split(x, l - 1, x, y);
    lazy[y] ^= 1;
    root = Merge(Merge(x, y), z);
}
vector<int> ans;
void dfs(int now) {
    pushDown(now);
    if (ch[now][0]) dfs(ch[now][0]);
    if (val[now] >= 1 && val[now] <= N) ans.push_back(val[now]);
    if (ch[now][1]) dfs(ch[now][1]);
}
int build(int L = 1, int R = N) {
    if (L > R) return 0;
    int M = (L + R) / 2;
    int now = newNode(M);
    ch[now][0] = build(L, M - 1);
    ch[now][1] = build(M + 1, R);
    Update(now);
    return now;
}
int main() {
    srand(time(NULL));
    N = read(); M = read();
    root = build();
    for (int i = 1, l, r; i <= M; i++) {
        l = read(); r = read();
        Reverse(l, r);
    }
    dfs(root);
    for (int i = 0; i < ans.size(); i++) printf("%d%c", ans[i], " \n"[i==ans.size()-1]);
}
/*
5 3
1 3
1 3
1 4

*/
```

2.22 Trie

```
struct Trie{
    int ch[maxn][maxc];
    int val[maxn];
    int sz;
    Trie(){sz = 1; clr(ch[0]); val[0] = 1;}
    int idx(char c) {return c - 'a';}
    void add(char* s, int v){
        int u = 0, n = strlen(s);
        for (int i = 0; i < n; i++){
            int c = idx(s[i]);
            if (!ch[u][c]){
                clr(ch[sz]);
                val[sz] = 0;
                ch[u][c] = sz++;
            }
            u = ch[u][c];
        }
        val[u] = v;
    }
    void dfs(int u = 0, int dep = 0){
        for (int i = 0; i < maxc; i++){
            int v = ch[u][i];
            if (v) dfs(v, dep + 1);
        }
    }
}trie;
```

3 Math

3.1 BernolliNumber

```
LL qpow(LL a, int n) {
    LL res = 1;
    while (n) {
        if (n & 1) res = (res * a) % MOD;
        a = (a * a) % MOD;
        n >>= 1;
    }
    return res;
}
LL inv[maxn], fac[maxn], invf[maxn], B[maxn];
inline LL C(LL x, LL y) {
    return ((fac[x] * invf[y]) % MOD * invf[x - y]) % MOD;
}
inline LL pow_sum(int m, int n){
    LL res = 0;
    for (int i = 0; i <= n; i++) res = (res + C(n + 1, i) * B[i] % MOD * qpow(m, n + 1 - i)) % MOD;
    res = (res * inv[n + 1]) % MOD;
    return res;
}
void init(){
    invf[0] = inv[0] = inv[1] = fac[0] = fac[1] = 1;
    for (LL i = 1; i < maxn; i++) fac[i] = fac[i - 1] * i % MOD;
    for (int i = 2; i < maxn; i++) inv[i] = 1LL * (MOD - MOD / i) * inv[MOD % i] % MOD;
    for (int i = 1; i < maxn; i++) invf[i] = invf[i - 1] * inv[i] % MOD;
    B[0] = 1;
    for (int i = 1; i < maxn; i++) {
        B[i] = 0;
        for(int j = 0; j < i; j++) B[i] = (B[i] + C(i + 1, j) * B[j]) % MOD;
        B[i] = ((B[i] * -inv[i + 1]) % MOD + MOD) % MOD;
    }
    B[1] = (MOD - B[1]) % MOD;
}
```

3.2 BM

```
const int MOD = 1000000007;
int inverse(int a) {
        return a == 1 ? 1 : (long long)(MOD - MOD / a) * inverse(MOD % a) % MOD;
}
// Berlekamp-Massey Algorithm
// Requirement: const MOD, inverse(int)
// Input: vector<int> the first elements of the sequence
// Output: vector<int> the recursive equation of the given sequence
// Example: In: {1, 1, 2, 3} Out: {1, 1000000006, 1000000006} (MOD = 1e9+7)
struct Poly {
        vector<int> a;
        Poly() { a.clear(); }
        Poly(vector<int> &a): a(a) {}
        int length() const { return a.size(); }
        Poly move(int d) {
                vector<int> na(d, 0);
                na.insert(na.end(), a.begin(), a.end());
                return Poly(na);
        }
        int calc(vector<int> &d, int pos) {
                int ret = 0;
                for (int i = 0; i < (int)a.size(); ++i) {
                        if ((ret += (long long)d[pos - i] * a[i] % MOD) >= MOD) {
                                ret -= MOD;
                        }
                }
                return ret;
        }
        Poly operator - (const Poly &b) {
                vector<int> na(max(this->length(), b.length()));
```

```
            for (int i = 0; i < (int)na.size(); ++i) {
                int aa = i < this->length() ? this->a[i] : 0,
                    bb = i < b.length() ? b.a[i] : 0;
                na[i] = (aa + MOD - bb) % MOD;
            }
            return Poly(na);
        }
};
Poly operator * (const int &c, const Poly &p) {
        vector<int> na(p.length());
        for (int i = 0; i < (int)na.size(); ++i) {
            na[i] = (long long)c * p.a[i] % MOD;
        }
        return na;
}
vector<int> solve(vector<int> a) {
        int n = a.size();
        Poly s, b;
        s.a.push_back(1), b.a.push_back(1);
        for (int i = 1, j = 0, ld = a[0]; i < n; ++i) {
            int d = s.calc(a, i);
            if (d) {
                if ((s.length() - 1) * 2 <= i) {
                    Poly ob = b;
                    b = s;
                    s = s - (long long)d * inverse(ld) % MOD * ob.move(i - j);
                    j = i;
                    ld = d;
                } else {
                    s = s - (long long)d * inverse(ld) % MOD * b.move(i - j);
                }
            }
        }
        return s.a;
}
```

3.3 Det

```
const int MOD = 1e9 + 7;
int N, M, T, K;
LL mat[maxn][maxn];
//By gauss elimination and the idea of gcd
LL det(int n) {
    LL ans = 1, f = 1;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int x = i, y = j;
            while (mat[y][i]) {
                LL t = mat[x][i] / mat[y][i];
                for (int k = i; k < n; k++)
                    mat[x][k] = (mat[x][k] - mat[y][k] * t % MOD) % MOD;
                swap(x, y);
                print();
            }
            if (x != i) {
                for (int k = 0; k < n; k++)
                    swap(mat[i][k], mat[j][k]);
                f = -f;
            }
        }
        ans = ans * mat[i][i] % MOD;
    }
    return (ans * f + MOD) % MOD;
}
```

3.4 Exgcd

```
//exgcd
void ex_gcd(int &d, int a, int b, int &x, int &y){
    if (b ==0) {d = a; x = 1; y = 0; return;}
    else {ex_gcd(d, b, a % b, y, x); y -= x*(a/b);}
}
//get inv
const int MOD = 1e9 + 7;
int inverse(int a) {
        return a == 1 ? 1 : (long long)(MOD - MOD / a) * inverse(MOD % a) % MOD;
}
```

3.5 FWT

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5e5 + 10;
const int N = 19;
const int MOD = 998244353;

inline int add(int x) {
    return x >= MOD ? x - MOD : x;
}
inline int sub(int x) {
    return x < 0 ? x + MOD : x;
}
inline int mul(int x, int y) {
    return (long long) x * y % MOD;
}
inline int pwr(int x, int y) {
    int ans = 1;
    for (; y; y >>= 1, x = mul(x, x)) {
        if (y & 1) ans = mul(ans, x);
    }
    return ans;
}

int n;
int a[maxn];
int cnt[1 << N | 10], sum;
void fwt(int a[], int b = 0) {
    int inv = (MOD + 1) / 2;
    for (int d = 1; d < (1 << N); d <<= 1)
        for (int m = d << 1, i = 0; i < (1 << N); i += m)
            for (int j = 0; j < d; j++) {
                int x = a[i + j], y = a[i + j + d];
                a[i + j] = x + y;
                if (a[i + j] >= MOD) a[i + j] -= MOD;
                a[i + j + d] = x - y + MOD;
                if (a[i + j + d] >= MOD) a[i + j + d] -= MOD;
                if (b) {
                    // Do many FWT once
                    a[i + j] = mul(a[i + j], inv);
                    a[i + j + d] = mul(a[i + j + d], inv);
                }
            }
}

bool check(int x) {
    static int f[1 << N | 10];
    for (int i = 0; i < (1 << N); i++) {
        // Do many FWT once
        f[i] = pwr(cnt[i], x);
    }
    fwt(f, 1);
    return f[sum];
}
int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; i ++) {
        scanf("%d", a + i);
        sum ^= a[i];
        cnt[a[i]] = 1;
    }
    cnt[0] = 1;
    fwt(cnt);
    if (sum == 0) {
        printf("%d\n", n);
    }
    else {
        int l = 1, r = min(20, n), ans = r;
        while(l <= r) {
            int mid = (l + r) >> 1;
            if (check(mid)) {
                ans = mid;
                r = mid - 1;
            }
            else {
                l = mid + 1;
            }
        }
        printf("%d\n", n - ans);
    }
}
```

3.6 GauseElimination

```
const double eps = 1e-6;
const int maxn = 50 + 11;
int N, M, T, K;
double matrix[maxn][maxn + 1];
double ans[maxn];
void exchange(int p1,int p2,int n){
   double t; int i;
   for(i = 0; i <= n; i++) {t = matrix[p1][i], matrix[p1][i] = matrix[p2][i], matrix[p2][i] = t;}
}
bool gauss(int n){
   for (int i = 0, p; i < n - 1; i++){
       p = i;
       for (int j = i + 1; j < n; j++) if (fabs(matrix[j][i]) > fabs(matrix[p][i])) p = j;
       if (p != i) exchange(p, i, n);
       if (fabs(matrix[i][i]) < eps) return false;
       for (int j = n; j >= i; j--) matrix[i][j] /= matrix[i][i];
       for (int j = i + 1; j < n; j++){
           for (int k = n; k >= i; k--){
               matrix[j][k] -= matrix[i][k] * matrix[j][i];
           }
       }
   }
   for (int i = n - 1; i >= 0; i--){
       ans[i] = matrix[i][n];
       for (int j = i - 1; j >= 0; j--){
           matrix[j][n] -= matrix[i][n] * matrix[j][i];
       }
   }
   return true;
}
```

3.7 LinearBase

```
const int maxn = 1e4 + 11;
const int maxb = 63;
int N, T, Q;
LL a[maxn], b[100];
vector<LL> base;
void pre(){
    base.clear(); memset(b, 0, sizeof(b));
    for (int i = 0; i < N; i++){
        for (int j = maxb; j >= 0; j--){
            if ((a[i] >> j) & 1LL){
                if (b[j]) a[i] ^= b[j];
                else{
                    b[j] = a[i];
                    for (int k = j - 1; k >= 0; k--) if (b[k] && ((b[j] >> k) & 1LL)) b[j] ^= b[k];
                    for (int k = j + 1; k <= maxb; k++) if ((b[k] >> j) & 1LL) b[k] ^= b[j];
                    break;
                }
            }
        }
    }
    for (int i = 0; i <= maxb; i++) if (b[i]) base.push_back(b[i]);
}
```

3.8 LinearRecursion

```
const int maxn = 1000 + 11;
const int MOD = 1e9 + 7;
int N, M, K;
```

```
//a_k = \sum_{i=0}^{k-1}(c_i a_i)
//Given a_0,...,a_{k-1}; c_0, ..., c_{k-1}
//O(n^2logk)
LL c[maxn<<1], a[maxn<<1], tmp[maxn<<1];
inline void mul(LL *A, LL *B){
    fill(tmp, tmp + 2*K-1, 0);
    for (int i = 0; i < K; i++){
        for (int j = 0; j < K; j++){
            tmp[i+j] = (tmp[i+j] + A[i]*B[j]) % MOD;
        }
    }
    for (int i = 2*(K-1); i >= K; i--){
        for (int j = 0; j < K; j++){
            tmp[i-K+j] = (tmp[i-K+j] + tmp[i]*c[j]) % MOD;
        }
        tmp[i] = 0;
    }
    copy(tmp, tmp + 2*K-1, A);
}
void qpow(LL *A, int n){
    LL ret[maxn<<1]; ret[0] = 1;
    while (n){
        if (n&1) mul(ret, A);
        mul(A, A);
        n >>= 1;
    }
    copy(ret, ret + K, A);
}
int main(){
    cin >> K >> N;
    N--;
    for (int i = 0; i < K; i++) cin >> a[i];
    for (int i = 0; i < K; i++) cin >> c[i];
    LL b[maxn<<1] = {0,1,};
    qpow(b, N);

    LL ans = 0;
    for (int i = 0; i < K; i++) ans = (ans + b[i] * a[i]) % MOD;
    cout << ans << endl;
    return 0;
}
```

3.9 LIS

```
list<int> lis(){
    vector<int> M(a.size() + 1, INF);
    int *p = new int[a.size() + 1];
    memset(p, -1, sizeof(p));
    int len = 0;
    for (int i = 0; i < a.size(); i++){
        int pos = lower_bound(M.begin(), M.end(), a[i]) - M.begin();
        p[a[i]] = pos ? M[pos - 1] : -1;
        M[pos] = a[i];
        if (pos + 1 > len) len = pos + 1;
    }
    list<int> res;
    for (int x = M[len - 1]; x >= 0; x = p[x]) res.push_front(x);
    return res;
}
```

3.10 MatrixInv

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn = 1000 + 11;
const int MOD = 1e9 + 7;
LL Mat[maxn][maxn], Inv[maxn][maxn];
LL qpow(LL x, LL n) {
    LL res = 1;
    x = (x % MOD + MOD) % MOD;
    while (n > 0LL) {
        if (n & 1LL) res = (res * x) % MOD;
        x = (x * x) % MOD;
        n >>= 1;
    }
    return res;
}
void getInv(int n) {
    int p; LL t;
    for (int i = 0; i < n; i++) Inv[i][i] = 1;
    for (int i = 0; i < n; i++) {
        p = i;
        for (int j = i + 1; j < n; j++)
            if (abs(Mat[j][i]) > abs(Mat[p][i])) p = j;
        for (int j = 0; j < n; j++) {
            swap(Mat[i][j], Mat[p][j]);
            swap(Inv[i][j], Inv[p][j]);
        }
        t = qpow(Mat[i][i], MOD - 2);
        for (int j = 0; j < n; j++) {
            Mat[i][j] = (Mat[i][j] * t) % MOD;
            Inv[i][j] = (Inv[i][j] * t) % MOD;
        }
        for (int j = 0; j < n; j++){
            if (i != j && Mat[j][i] != 0) {
                t = Mat[j][i];
                for (int k = 0; k < n; k++) {
                    Mat[j][k] = (Mat[j][k] - (Mat[i][k] * t) % MOD) % MOD;
                    Inv[j][k] = (Inv[j][k] - (Inv[i][k] * t) % MOD) % MOD;
                }
            }
        }
    }
}
int main(){
    Mat[0][0]=93;Mat[0][1]=2;Mat[0][2]=3;
    Mat[1][0]=5;Mat[1][1]=7;Mat[1][2]=10;
    Mat[2][0]=2;Mat[2][1]=99;Mat[2][2]=5;
    getInv(3);
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            LL t=0;
            for(int k=0;k<3;k++) t=(((t+Mat[i][k]*Inv[k][j])%MOD)+MOD)%MOD;
            cout<<t<<" \n"[j==2];
        }
    }
}
```

3.11 Matrix-qpow

```
const int MAXN = 105;
const LL modular = 1000000007;
int n; // order of matrices

struct matrix{
    LL m[MAXN][MAXN];

    void operator *=(matrix& a){
        static LL t[MAXN][MAXN];
        Rep (i, n){
            Rep (j, n){
                t[i][j] = 0;
                Rep (k, n){
                    t[i][j] += (m[i][k] * a.m[k][j]) % modular;
                    t[i][j] %= modular;
                }
            }
        }
        memcpy(m, t, sizeof(t));
    }
};

matrix r;
void m_powmod(matrix& b, LL e){
    memset(r.m, 0, sizeof(r.m));
    Rep(i, n)
        r.m[i][i] = 1;
    while (e){
        if (e & 1) r *= b;
        b *= b;
        e >>= 1;
    }
}
```

3.12 MatrixQPow

```
const LL INF = 0x3f3f3f3f3f3f3f3f;
const int maxt = 100 + 3;
const int MOD = 1e9 + 7;
int N, M, T;
struct Mat{
    LL a[maxt][maxt]; int n;
    Mat(int n):n(n){}
    void clr(){memset(a, 0, sizeof(a));}
    void eye(){memset(a, 0, sizeof(a)); for (int i = 0; i < n; i++) a[i][i] = 1LL;}
    void maxi(){memset(a, 0x3f, sizeof(a));}
};
Mat cal_mul(Mat A, Mat B){
    int n = A.n;
    Mat C = Mat(n); C.clr();
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            for (int k = 0; k < n; k++){
                C.a[i][j] = (C.a[i][j] + A.a[i][k] * B.a[k][j]) % MOD;
            }
        }
    }
    return C;
}
Mat qpow(Mat A, int n){
    Mat res = Mat(A.n); res.eye();
    while (n > 0){
        if (n & 1) res = cal_mul(res, A);
        A = cal_mul(A, A);
        n >>= 1;
    }
    return res;
}
```

3.13 MillerRabin

```
const int S = 20;
LL N, M, T;
LL qpow(__int128 a, LL n, __int128 wm) {
    __int128 r = 1;
    while (n) {
        if (n & 1) r = r * a % m;
        a = a * a % m;
        n >>= 1;
    }
    return LL(r);
}
//composite : true
bool check(LL a, LL n, LL x, LL t) {
    LL ret = qpow(a, x, n);
    LL last = ret;
    for (int i = 1; i <= t; i++) {
        ret = qpow(ret, 2, n);
        if (ret == 1 && last != 1 && last != n - 1) return true; //
        last = ret;
    }
    if (ret != 1) return true;
    return false;
}
// Miller_Rabin()
bool Miller_Rabin(LL n) {
    if (n < 2) return false;
    if (n == 2) return true;
    if ((n & 1) == 0) return false; //
    LL x = n - 1;
    LL t = 0;
    while ((x & 1) == 0) {x >>= 1; t++;}
    for (int i = 0; i < S; i++) {
        LL a = rand() % (n - 1) + 1;
        if (check(a, n, x, t)) return false;//
    }
    return true;
}

//pollard_rho
LL factor[100];
int tol;
LL Pollard_rho(LL x, LL c) {
    LL i = 1, k = 2;
    LL x0 = rand() % x;
    LL y = x0;
    while (1) {
        i++;
        x0 = (qpow(x0, 2, x) + c) % x;
        LL d = abs(__gcd(y - x0, x));
        if (d != 1 && d != x) return d;
        if (y == x0) return x;
        if (i == k) {y = x0; k += k;}
    }
}
void fac(LL n) {
    if (Miller_Rabin(n)) {
        factor[tol++] = n;
        return;
    }
    LL p = n;
    while (p >= n) p = Pollard_rho(p, rand() % (n - 1) + 1);
    fac(p);
    fac(n / p);
}
int main() {
    srand(time(NULL));
    cin >> T;
    while (T--) {
        cin >> N;
        tol = 0;
        fac(N);
        sort(factor, factor + tol);
        if (Miller_Rabin(N)) printf("Prime\n");
        else printf("%lld\n", factor[0]);
    }
    return 0;
}
```

3.14 MOD

```
inline int add(int x) {return x >= MOD ? x - MOD : x;}
inline int sub(int x) {return x < 0 ? x + MOD : x;}
inline int mul(int x, int y) {return (long long) x * y % MOD;}
inline int pwr(int x, int y) {
    int ans = 1;
    for (; y; y >>= 1, x = mul(x, x)) {
        if (y & 1) ans = mul(ans, x);
    }
    return ans;
}
```

```
LL inv[maxn], fac[maxn], invf[maxn];
inline LL C(LL x, LL y) {
    return ((fac[x] * invf[y]) % MOD * invf[x - y]) % MOD;
}
void init(){
    invf[0] = inv[0] = inv[1] = fac[0] = fac[1] = 1;
    for (LL i = 1; i < maxn; i++) fac[i] = fac[i - 1] * i % MOD;
    for (int i = 2; i < maxn; i++) inv[i] = 1LL * (MOD - MOD / i) * inv[MOD % i] % MOD;
    for (int i = 1; i < maxn; i++) invf[i] = invf[i - 1] * inv[i] % MOD;
}
```

3.15 FFT

```
#define rep(i,n) for(int i=0;i<(n);i++)
typedef complex<double> cplx;
typedef vector<cplx> vc;
const int NMAX = 1 << 21;
int get(int s) { return s > 1 ? 32 - __builtin_clz(s - 1) : 0;}
struct FFT{
    int rev[NMAX];
    cplx omega[NMAX], oinv[NMAX];
    void dft(vc &a, cplx* w){
        int n = a.size();
        rep (i, n) if (i < rev[i]) swap(a[i], a[rev[i]]);
        for (int l = 2; l <= n; l *= 2){
            int m = l/2;
            for (int p = 0; p < n; p += l)
                rep (k, m){
                    cplx t = w[n/l*k] * a[p+k+m];
                    a[p+k+m] = a[p+k] - t; a[p+k] += t;
                }
        }
    }
    void fft(vc &a){ dft(a, omega); }
    void ifft(vc &a){
        int n = a.size();
        dft(a, oinv);
        rep (i, n) a[i] /= n;
    }
    vc brute(vc& a, vc& b) {
        vc c(a.size()+b.size()-1);
        rep(i,a.size()) rep(j,b.size()) c[i+j] += a[i]*b[j];
        return c;
    }
    vc conv(vc a, vc b){
        int s = a.size()+b.size()-1, k = get(s), n = 1<<k;
        if (s <= 0) return {};
        if (s <= 200) return brute(a,b);
        rep (i, n){
            rev[i] = (rev[i>>1]>>1) | ((i&1)<<(k-1));
            omega[i] = polar(1.0, 2.0 * PI / n * i);
            oinv[i] = conj(omega[i]);
        }
        a.resize(n); fft(a);
        b.resize(n); fft(b);
        rep(i,n) a[i] *= b[i];
        ifft(a);
        a.resize(s);
        return a;
    }
} fft;
```

3.16 NTT

```
const int P = 998244353;
const LL G = 3;
#define rep(i,n) for(int i=0;i<(n);i++)
#define Rep(i,a,b) for(int i=(a);i<(b);i++)
typedef vector<LL> vl;
int get(int s) { return s > 1 ? 32 - __builtin_clz(s - 1) : 0;}
namespace NTT {
    LL qpow(LL b, LL p) { return !p?1:qpow(b*b%P,p/2)*(p&1?b:1)%P; }
    LL inv(LL b) { return qpow(b,P-2); }
    void ntt(vl& a) {
        int n = a.size(), x = get(n);
        vl b(n), omega(n);
        omega[0] = 1, omega[1] = qpow(G,(P-1)/n);
        Rep(i,2,n) omega[i] = omega[i-1]*omega[1] % P;
        Rep(i,1,x+1){
            int inc=n>>i;
            rep(j,inc) for(int k=0;k<n;k+=inc){
                int t = 2*k%n+j;
                b[k+j] = (a[t]+omega[k]*a[t+inc]) % P;
            }
            swap(a,b);
        }
    }
    void ntt_rev(vl& a) {
        ntt(a);
        LL r = inv(a.size());
        rep(i,a.size()) a[i] = a[i] * r % P;
        reverse(a.begin() + 1, a.end());
    }
    vl brute(vl& a, vl& b) {
        vl c(a.size()+b.size()-1);
        rep(i,a.size()) rep(j,b.size()) c[i+j] = (c[i+j]+a[i]*b[j])%P;
        return c;
    }
    vl conv(vl a, vl b) {
        int s = a.size()+b.size()-1, L = get(s), n = 1<<L;
        if (s <= 0) return {};
        if (s <= 200) return brute(a,b);

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

        rep(i,n) a[i] = a[i] * b[i] % P;
        ntt_rev(a);

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

3.17 NTT-3MOD

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef unsigned long long ULL;
const int INF = 0x3f3f3f3f;
const int maxn = 1e5 + 11;
int N, M, T;

int P = 998244353;
const LL G = 3;
#define rep(i,n) for(int i=0;i<(n);i++)
#define Rep(i,a,b) for(int i=(a);i<(b);i++)
typedef vector<LL> vl;
int get(int s) { return s > 1 ? 32 - __builtin_clz(s - 1) : 0;}
namespace NTT {
    LL qpow(LL b, LL p) { return !p?1:qpow(b*b%P,p/2)*(p&1?b:1)%P; }
    LL inv(LL b) { return qpow(b,P-2); }

    void ntt(vl& a) {
        int n = a.size(), x = get(n);
        vl b(n), omega(n);
        omega[0] = 1, omega[1] = qpow(G,(P-1)/n);
        Rep(i,2,n) omega[i] = omega[i-1]*omega[1] % P;
        Rep(i,1,x+1){
            int inc=n>>i;
            rep(j,inc) for(int k=0;k<n;k+=inc){
                int t = 2*k%n+j;
                b[k+j] = (a[t]+omega[k]*a[t+inc]) % P;
            }
            swap(a,b);
        }
    }

    void ntt_rev(vl& a) {
        ntt(a);
        LL r = inv(a.size());
        rep(i,a.size()) a[i] = a[i] * r % P;
        reverse(a.begin() + 1, a.end());
    }

    vl brute(vl& a, vl& b) {
        vl c(a.size()+b.size()-1);
        rep(i,a.size()) rep(j,b.size()) c[i+j] = (c[i+j]+a[i]*b[j])%P;
        return c;
    }

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

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

        rep(i,n) a[i] = a[i] * b[i] % P;
        ntt_rev(a);
        a.resize(s);
        return a;
    }
}

const int MOD = 1e9 + 7;
const int MO[] = {0, 998244353,1004535809,469762049};
const ULL MO1_MO2 = 1002772198720536577ULL;
const int MO1_INV_MO2 = 669690699; /*modPow<MO2>(MO1, MO2 - 2)*/
const int MO2_INV_MO1 = 332747959; /*modPow<MO1>(MO2, MO1 - 2)*/
const int MO1_MO2_INV_MO3 = 354521948;
/*modPow<MO3>(MO1_MO2 % MO3, MO3 - 2)*/
const int MO1_MO2_MOD = 701131240;/*MO1_MO2 % MOD*/
inline int crt(int a1, int a2, int a3) {
    ULL a = ((ULL)MO[2] * ((ULL)a1 * MO2_INV_MO1 % MO[1]) +
               (ULL)MO[1] * ((ULL)a2 * MO1_INV_MO2 % MO[2])) % MO1_MO2;
        return (a + (MO[3] + a3 - a % MO[3]) % MO[3] *
            MO1_MO2_INV_MO3 % MO[3] * MO1_MO2_MOD % MOD) % MOD;
}

vl C[4];
int main(){
    scanf("%d%d", &N, &M);
    vl A(N + 1, 0), B(M + 1, 0);
    for (int i = 0; i <= N; i++) scanf("%d", &A[i]);
    for (int i = 0; i <= M; i++) scanf("%d", &B[i]);
    for (int i = 1; i <= 3; i++) {
        P = MO[i];
        C[i] = NTT::conv(A, B);
    }
    for (int i = 0; i <= N + M; i++) printf("%d%c", crt(C[1][i],C[2][i],C[3][i]), " \n"[i==N+M]);
    return 0;
}
/*
1 2
1 2
1 2 1

*/
```

3.18 PolynomialInverse

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[31];
int dbit(int x)
{
    while(x!=(x&-x)) x+=(x&-x);
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
```

```
int n,A[MAXN],B[MAXN],C[MAXN];
void find_inverse(int A[],int n)
{
        if(n==1) {B[0]=pow_mod(A[0],MOD-2); return;}
        find_inverse(A,(n+1)/2);
        int len=dbit(n)*2;
        for(int i=0;i<n;i++)
                C[i]=A[i];
        for(int i=n;i<len;i++) C[i]=0;
        ntt(len,C,1);ntt(len,B,1);
        for(int i=0;i<len;i++)
                C[i]=1LL*B[i]*B[i]%MOD*C[i]%MOD;
        ntt(len,C,-1);ntt(len,B,-1);
        for(int i=0;i<n;i++)
                B[i]=((2*B[i]-C[i])%MOD+MOD)%MOD;
}
int main()
{
        for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    for(int i=0;i<2;i++)
        A[i]=1;
    find_inverse(A,3);
    for(int i=0;i<4;i++) printf("%d ",B[i]);
    return 0;
}
```

3.19 PolynomialSquareRoot

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[31];
int dbit(int x)
{
    while(x!=(x&-x)) x+=(x&-x);
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
```

```
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
int n,A[MAXN],B[MAXN],C[MAXN],D[MAXN];
void find_inverse(int A[],int n)
{
        if(n==1) {B[0]=pow_mod(A[0],MOD-2); return;}
        find_inverse(A,(n+1)/2);
        int len=dbit(n);
        for(int i=0;i<n;i++)
                C[i]=A[i];
        for(int i=n;i<len;i++) C[i]=0;
        ntt(len,C,1);ntt(len,B,1);
        for(int i=0;i<len;i++)
                C[i]=1LL*B[i]*B[i]%MOD*C[i]%MOD;
        ntt(len,C,-1);ntt(len,B,-1);
        for(int i=0;i<n;i++)
```

```
                B[i]=((2*B[i]-C[i])%MOD+MOD)%MOD;
}
void find_sqr(int A[],int n)
{
    if(n==1)
    {
        D[0]=A[0];
        return;
    }
    find_sqr(A,(n+1)/2);
    memset(B,0,sizeof(B));
    find_inverse(D,(n+1)/2);
    for(int i=0;i<(n+1)/2;i++)
        B[i]=1LL*B[i]*((MOD+1)/2)%MOD;
    int len=dbit(n)*2;
    ntt(len,D,1);
    for(int i=0;i<len;i++)
        D[i]=1LL*D[i]*D[i]%MOD;
    ntt(len,D,-1);
    for(int i=0;i<n;i++)
        D[i]=(D[i]+A[i])%MOD;
    ntt(len,D,1);ntt(len,B,1);
    for(int i=0;i<len;i++)
        D[i]=1LL*D[i]*B[i]%MOD;
    ntt(len,D,-1);
    for(int i=n;i<2*n;i++) D[i]=0;
}
int main()
{
        for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    A[0]=1;
    A[1]=MOD-2;
    A[2]=1;
    find_sqr(A,4);
    for(int i=0;i<4;i++) printf("%d ",D[i]);
    return 0;
}
```

3.20 PrimeCount

```
#include<bits/stdc++.h>
#define MAXN 1000005// MAXN=sqrt(upper_bound)
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long LL;
typedef pair<int, int> P;
LL f[MAXN], g[MAXN], n, k; //f[i]:pi(n/i),g[i]:pi(i)
// PrimeCount: g(n,j) = g(n,j1)    (1) * [g([n/Pj],j1) g ( Pj1 , j1 )] iff n is prime
// PrimeSum: g(n,j) = g(n,j1)    (n) * [g([n/Pj],j1) g ( Pj1 , j1 )] iff n is prime
// the i-th iteration (2<=i<=m) is according to g(:,i-1) -> g(:,i)
// f is used to save space

LL PrimeCount(LL n) {
   LL i, j, m = 0;
   for (m = 1; m * m <= n; m++) f[m] = n / m - 1;
   for (i = 2; i <= m; i++) g[i] = i - 1;
   for (i = 2; i <= m; i++) {
       if (g[i] == g[i - 1]) continue;
       for (j = 1; j <= min(m - 1, n / i / i); ++j) {
           if (i * j < m) f[j] -= f[i * j] - g[i - 1];
           else f[j] -= g[n / i / j] - g[i - 1];
       }
       for (j = m; j >= i * i; j--) g[j] -= g[j / i] - g[i - 1];
   }
   return f[1];
}

// Be caution that it may cause long long overflow.
LL f[maxn], g[maxn];
inline LL C2(LL x) {
   if (x >= MOD) x %= MOD;
   return x * (x - 1) / 2;
}
LL PrimeSum(LL n) {
   LL i, j, m = 0;
   for (m = 1; m * m <= n; m++) f[m] = (C2(n / m + 1) - 1 + MOD) % MOD;
   for (i = 2; i <= m; i++) g[i] = (C2(i + 1) - 1 + MOD) % MOD;
   for (i = 2; i <= m; i++) {
       if (g[i] == g[i - 1]) continue;
       for (j = 1; j <= min(m - 1, n / i / i); ++j) {
           if (i * j < m) f[j] -= (__int128)i * (f[i * j] - g[i - 1] + MOD) % MOD;
           else f[j] -= (__int128)i * (g[n / i / j] - g[i - 1] + MOD) % MOD;
           f[j] = (f[j] % MOD + MOD) % MOD;
       }
       for (j = m; j >= i * i; j--) {
           g[j] -= (__int128)i * (g[j / i] - g[i - 1] + MOD) % MOD;
           g[j] = (g[j] % MOD + MOD) % MOD;
       }
   }
   return f[1];
}


int main() {
   while(scanf("%lld", &n) == 1) {
       printf("%lld\n", PrimeSum(n));
       printf("%lld\n", PrimeCount(n));
   }
   return 0;
}
```

3.21 PrimitiveRoot

```
#include<bits/stdc++.h>
const int maxn = 1e6 + 11;
using namespace std;
typedef long long LL;
vector<LL> a;
LL pow_mod(LL a, LL i, LL mod) {
    if (i == 0) return 1;
    LL s = 1;
    while(i > 0) {
        if (i & 1) s = (s * a) % mod;
        a = (a * a) % mod;
        i >>= 1;
    }
    return s;
}
bool g_test(LL g, LL p) {
    for (LL i = 0; i < a.size(); i++) if (pow_mod(g, (p - 1) / a[i], p) == 1) return 0;
    return 1;
}
LL primitive_root(LL p) {
    LL tmp = p - 1;
    for (LL i = 2; i <= tmp / i; i++)
        if (tmp % i == 0) {
            a.push_back(i);
            while(tmp % i == 0) tmp /= i;
        }
    if (tmp != 1) a.push_back(tmp);
    LL g = 1;
    while(true) {
        if (g_test(g, p)) return g;
        ++g;
    }
}
int main() {
    LL n;
    while(scanf("%lld", &n) == 1) printf("%lld\n", primitive_root(n));
    return 0;
}
```

3.22 Sieves

```
#include<bits/stdc++.h>
using namespace std; typedef long long LL;
const int maxn = 1e6 + 11;
const int MOD = 1e9 + 7;
int N, M, T;
vector<int> prime;
bool checked[maxn];
int phi[maxn], mu[maxn];
int smu[maxn], sphi[maxn];
int d[maxn], e[maxn], md[maxn];
LL inv[maxn];
//Linear Seives
void get_prime(int n){
    memset(checked, 0, sizeof(checked)); prime.clear();
    for (int i = 2; i <= n; i++){
        if (!checked[i]) prime.push_back(i);
        for (int j = 0; j < prime.size() && i * prime[j] <= n; j++){
            checked[i * prime[j]] = true;
            if (i % prime[j] == 0) break;
        }
    }
}
void get_phi(int n){
    memset(checked, 0, sizeof(checked)); prime.clear();
    phi[1] = 1;
    for (int i = 2; i <= n; i++){
        if (!checked[i]) {prime.push_back(i); phi[i] = i - 1;}
        for (int j = 0; j < prime.size() && i * prime[j] <= n; j++){
            checked[i * prime[j]] = true;
            if (i % prime[j] == 0) {phi[i * prime[j]] = phi[i] * prime[j]; break;}
            else {phi[i * prime[j]] = phi[i] * (prime[j] - 1);}
        }
    }
}
void get_mu(int n){
    memset(checked, 0, sizeof(checked)); prime.clear();
    mu[1] = 1;
    for (int i = 2; i <= n; i++){
        if (!checked[i]) {prime.push_back(i); mu[i] = -1;}
        for (int j = 0; j < prime.size() && i * prime[j] <= n; j++){
            checked[i * prime[j]] = true;
            if (i % prime[j] == 0) {mu[i * prime[j]] = 0; break;}
            else {mu[i * prime[j]] = -mu[i];}
        }
    }
}
//Linear getting inverse mod MOD
void get_inv(LL n){
    inv[1] = 1;
    for (LL i = 2; i <= n; i++){
        inv[i] = 1LL * (MOD - (MOD / i)) * inv[MOD % i] % MOD;
    }
}
void get_facnum(int n){
    //md is the minimum divisor
    md[1] = d[1] = 1;
    for(int i = 2; i <= n; i++){
        if(!checked[i]){
            prime.push_back(i);
            d[i] = 2; e[i] = 1;
            md[i] = i;
        }
        for (int j = 0; j < prime.size() && i * prime[j] <= n; j++){
            int &curp = prime[j];
            checked[i * curp] = true;
            md[i * curp] = curp;
            if(i % curp == 0){
                d[i * curp] = d[i] / (e[i] + 1) * (e[i] + 2);
                e[i * curp] = e[i] + 1;
                break;
            }
            d[i * curp] = d[i] * 2;
            e[i * curp] = 1;
        }
    }
}

int main(){
    get_phi(1e4); get_mu(1e4);
    int UB = 100; int sum = 0; smu[0] = 0;
    for (int i = 1; i <= UB; i++) smu[i] = smu[i - 1] + mu[i];
    // Accelerating summation skill
    for (int i = 1, last; i <= UB; i = last + 1){
        last = UB / (UB / i);
        sum += (UB / i) * (UB / i + 1) / 2 * (smu[last] - smu[i - 1]);
    }
    cout << sum << endl;
    return 0;
}
```

3.23 Simplex

```
/*


    nn               x1,x2,  ,xnx1,x2,  ,xn    mm                  ii
     nj =1  aijxjbij  =1  naijxjbi
         nn                                         xj0xj0
                                                             xjxj
    F= nj =1cjxjF= j =1ncjxj

                       n,m,t n,m,   t      t {0,1} t {0,1}
            nn          c1,c2,  ,cnc1,c2,  ,                 cn
         mm                                    ii       n+1n+1
    ai1,ai2,  ,ain,biai1,ai2,  ,ain,                 bi

                                                     "Infeasible"
                                           MM                                  MM        "Unbounded"

                                   FF
    10                  6106
      t=1t=1                                                         nn
    x1,x2,  ,xnx1,x2,  ,xn
                                                  Fnj   =1   cjxjFj   =1ncjxj
    00                     ii     min{0,  binj  =1aijxj}min{0,  bij  =1naijxj}
    00                                     00                  00
    S+S+           SS        S+S+       SS
    10            6106
      t=0t=0              Infeasible    Unbounded
*/
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int N = 200;
const double eps = 1e-6;
const int maxn = 1000;
double a[N][N], ans[N];
int n, m, t, id[N << 1];
int MM, NN, TT;
void pivot(int l, int e) {
    swap(id[e], id[n + l]);
    double r = a[l][e]; a[l][e] = 1;
    for (int j = 0; j <= n; ++j)
        a[l][j] /= r;
    for (int i = 0; i <= m; ++i)
        if (i != l) {
            r = a[i][e]; a[i][e] = 0;
            for (int j = 0; j <= n; ++j)
                a[i][j] -= r * a[l][j];
        }
}
void read(){
    scanf("%d%d%d", &n, &m, &t);
    for (int j = 1; j <= n; ++j) scanf("%lf", &a[0][j]), id[j] = j;
    for (int i = 1; i <= m; ++i) {
        for (int j = 1; j <= n; ++j)
            scanf("%lf", &a[i][j]);
        scanf("%lf", &a[i][0]);
    }
}
bool solve(){
    int i, j, l, e; double k, kk;
    while (true) {
        l = e = 0; k = -eps;
        for (i = 1; i <= m; ++i)
            if (a[i][0] < k) {
                k = a[i][0];
                l = i;
            }
        if (!l) break;
        k = -eps;
        for (j = 1; j <= n; ++j)
            if (a[l][j] < k && (!e || (rand() & 1))) {
                k = a[l][j];
                e = j;
            }
        if (!e) {puts("Infeasible"); return false;}
        pivot(l, e);
    }
    while (true) {
        for (j = 1; j <= n; ++j)
            if (a[0][j] > eps)
                break;
        if ((e = j) > n) break;
        k = 1e18; l = 0;
        for (i = 1; i <= m; ++i)
            if (a[i][e] > eps && (kk = (a[i][0] / a[i][e])) < k) {
                k = kk;
                l = i;
            }
        if (!l) {puts("Unbounded"); return false;}
        pivot(l, e);
    }
    return true;
}
int main() {
    scanf("%d", &TT);
    while (TT--){
        //
        memset(a, 0, sizeof(a)); memset(ans, 0, sizeof(ans)); memset(id, 0, sizeof(id));
        read();
        bool ok = solve(); if (!ok) continue;
        printf("%.10lf\n", -a[0][0]);
        if (!t) return 0;
        for (int i = 1; i <= m; ++i) ans[id[n + i]] = a[i][0];
        for (int i = 1; i <= n; ++i) printf("%.10lf ", ans[i]);
        eend: ;
    }
    return 0;
}
```

3.24 SumMiu

```
#include<bits/stdc++.h>
#define MAXN 5000005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
bool is_prime[MAXN];
int cnt,mu[MAXN],prime[MAXN];
ll n,m,f[MAXN];
map<ll,ll> mp;
void genmiu(int n)
{
    int p=0;
    for(int i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    memset(mu,0,sizeof(mu));
    mu[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i]) {prime[p++]=i; mu[i]=-1;}
        for(int j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            is_prime[prime[j]*i]=false;
            mu[i*prime[j]]=i%prime[j]?-mu[i]:0;
            if(i%prime[j]==0) break;
        }
    }
    for(int i=1;i<=n;i++) f[i]=f[i-1]+mu[i];
}
ll calc(ll x)
{
        if(x<=5000000) return f[x];
        if(mp.find(x)!=mp.end()) return mp[x];
        ll ans=1;
        for(ll i=2,r;i<=x;i=r+1)
        {
                r=x/(x/i);
                ans-=calc(x/i)*(r-i+1);
        }
        return mp[x]=ans;
}
int main()
{
        genmiu(5000000);
        scanf("%lld%lld",&n,&m);
        printf("%lld\n",calc(m)-calc(n-1));
        return 0;
}
```

3.25 SumPhi

```
#include<bits/stdc++.h>
#define MAXN 5000005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
bool is_prime[MAXN];
ll cnt,phi[MAXN],prime[MAXN];
ll n,f[MAXN];
map<ll,ll> mp;
ll mul_mod(ll a,ll i)
{
        ll s=0;a%=MOD;
        while(i)
        {
                if(i&1) s=(s+a)%MOD;
                a=(a+a)%MOD;
                i>>=1;
        }
        return s;
}

ll pow_mod(ll a,ll i)
{
        ll s=1;
        while(i)
        {
                if(i&1) s=mul_mod(s,a);
                a=mul_mod(a,a);
                i>>=1;
        }
        return s;
}
void genphi(ll n)
{
    ll p=0;
    memset(phi,0,sizeof(phi));
    phi[1]=1;
     for(ll i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    for(ll i=2;i<=n;i++)
    {
        if(is_prime[i]) {prime[p++]=i; phi[i]=i-1;}
        for(ll j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            is_prime[prime[j]*i]=false;
            phi[i*prime[j]]=phi[i]*(i%prime[j]?prime[j]-1:prime[j]);
            if(i%prime[j]==0) break;
```

```
       }
    }
    for(ll i=1;i<=n;i++) f[i]=(f[i-1]+phi[i])%MOD;
}
ll calc(ll x)
{
       if(x<=5000000) return f[x];
       if(mp.find(x)!=mp.end()) return mp[x];
       ll ans=mul_mod(mul_mod(x,x+1),pow_mod(2,MOD-2));
       for(ll i=2,r;i<=x;i=r+1)
       {
              r=x/(x/i);
              ans=(ans-calc(x/i)*((r-i+1)%MOD)%MOD+MOD)%MOD;
       }
       return mp[x]=ans;
}
int main()
{
       genphi(5000000);
       scanf("%lld",&n);
       printf("%lld\n",calc(n));
       return 0;
}
```

4 Java

4.1 BigInteger

```
BigInteger abs()
BigInteger add(BigInteger val)
BigInteger and(BigInteger val)
BigInteger andNot(BigInteger val)
BigInteger divide(BigInteger val)
double doubleValue()              double
float floatValue()              float
BigInteger gcd(BigInteger val)
int intValue()
long longValue()           long
BigInteger max(BigInteger val)
BigInteger min(BigInteger val)
BigInteger mod(BigInteger val)           val
BigInteger multiply(BigInteger val)
BigInteger negate()
BigInteger not()
BigInteger or(BigInteger val)
BigInteger pow(int exponent)             exponent
BigInteger remainder(BigInteger val)               val
BigInteger leftShift(int n)                n
BigInteger rightShift(int n)                n
BigInteger subtract(BigInteger val)
byte[] toByteArray(BigInteger val)                      byte
String toString(int k)                       k
BigInteger xor(BigInteger val)
```

4.2 BinarySearch

```
int lower_bound(BigInteger x, int low, int high, BigInteger [] a) {
int left = low, right = high;
while (left < right) {
int mid = left + (right - left) / 2;
if (a[mid].compareTo(x) == -1) {
left = mid + 1;
}else { right = mid; }
}
return left;
}
int upper_bound(BigInteger x, int low, int high, BigInteger [] a) {
int left = low, right = high;
while (left < right) {
int mid = left + (right - left) / 2;
if (a[mid].compareTo(x) != 1) {
left = mid + 1;
}else { right = mid; }
}
return left;
}
```

4.3 FFT

```
import java.util.*;
import java.math.*;
import java.io.*;
public class Main {
Scanner cin = new Scanner(new BufferedInputStream(System.in));
public class Complex{
double real, imag;
Complex(double a, double b){
this.real = a; this.imag = b;
}
void set(double a, double b) {
this.real = a; this.imag = b;
}
Complex add(Complex rhs) {
double a = this.real + rhs.real;
double b = this.imag + rhs.imag;
return new Complex(a, b);
}
Complex subtract(Complex rhs) {
double a = this.real - rhs.real;
double b = this.imag - rhs.imag;
return new Complex(a, b);
}
Complex multiply(Complex rhs) {
double a = this.real * rhs.real - this.imag * rhs.imag;
double b = this.real * rhs.imag + this.imag * rhs.real;
return new Complex(a, b);
}
Complex divide(Complex rhs) {
Complex c = new Complex(this.real, this.imag).multiply(new Complex(rhs.real, -rhs.imag));
double div = rhs.real * rhs.real + rhs.imag * rhs.imag;
c.set(c.real / div, c.imag / div);
return c;
}
}
int N, M, logM;
double PI = Math.acos(-1.0);
Complex [] A, B, C;
int [] BitReverse(int [] X) {
for (int i = 0; i < M; i++)
{
int tmp = X[i], cur = 0;
for (int j = 0; j < logM; j++, tmp >>= 1) cur = (cur << 1) | (tmp & 1);
X[i] = cur;
}
return X;
}

Complex [] FFT(Complex [] X, int [] P, int dir) {
Complex [] XX = X.clone();
if (dir == 1) {
for (int i = 0; i < M; i++) X[i] = XX[P[i]];
}else {
for (int i = 0; i < M; i++) X[P[i]] = XX[i];
}
for (int s = 1, m = 1; s <= logM; s++) {
m <<= 1;
Complex wm = new Complex(Math.cos(2*PI/m), dir * Math.sin(2*PI/m));
for (int k = 0; k < M; k += m) {
Complex w = new Complex(1.0, 0);
for (int j = 0; j < (m >> 1); j++) {
Complex u = X[k + j], t = X[k + j + m / 2].multiply(w);
X[k + j] = u.add(t);
X[k + j + m / 2] = u.subtract(t);
w = w.multiply(wm);
}
}
}
return X;
}

void Iter_FFT() {
int [] P = new int[M]; for (int i = 0; i < M; i++) P[i] = i;
P = BitReverse(P);
A = FFT(A, P, 1); B = FFT(B, P, 1);
for (int i = 0; i < M; i++) C[i] = A[i].multiply(B[i]);
C = FFT(C, P, -1);
for (int i = 0; i < M; i++) C[i].set(C[i].real / M, C[i].imag / M);
}

void run() {
N = cin.nextInt();
for (M = 1, logM = 0; M < 2 * N; M <<= 1, logM++);
A = new Complex[M];
B = new Complex[M];
C = new Complex[M];
for (int i = 0; i < M; i++) {A[i] = new Complex(0.0, 0.0); B[i] = new Complex(0.0, 0.0);}
for (int i = 0; i < N; i++) {A[i].set(1.0, 0.0); B[i].set(1.0, 0.0);}
Iter_FFT();
```

```java
for (int i = 0; i < M; i++) System.out.print((int)(C[i].real + 0.5) + " ");
System.out.println();

}
public static void main(String[] args) {
new Main().run();
}
}
```

4.4 IO

```java
import java.util.*;
import java.io.*;
public class Main
{
public static void main(String[] args)
{
//stdin
Scanner cin1 = new Scanner(System.in);
Scanner cin2 = new Scanner(new BufferedInputStream(System.in));
//                        cin2cin1

//fileout
File file = new File("output.txt");
if(!file.exists()) file.createNewFile();
FileWriter fw = new FileWriter("output.txt", true);
PrintWriter pw = new PrintWriter(fw);
pw.print("China");
pw.flush();
pw.println("Foreign");
pw.flush();
}
}
```

4.5 MatrixPow-BigInteger

```java
import java.util.*;
import java.math.*;
import java.io.*;
public class Main {
Scanner cin = new Scanner(new BufferedInputStream(System.in));
BigInteger [][] A = new BigInteger[2][2];

BigInteger [][] mul(BigInteger [][] a, BigInteger [][] b){
int size = a.length;
BigInteger [][] c = new BigInteger[size][size];
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
c[i][j] = BigInteger.ZERO;
}
}
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
for (int k = 0; k < size; k++) {
c[i][j] = c[i][j].add(a[i][k].multiply(b[k][j]));
}
}
}
return c;
}

BigInteger [][] qpow(BigInteger [][] a, int n){
int size = a.length;
BigInteger [][] res = new BigInteger [size][size];
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
res[i][j] = (i == j) ? BigInteger.ONE : BigInteger.ZERO;
}
}
while (n > 0) {
if ((n & 1) == 1) {
res = mul(res, a);
}
a = mul(a, a);
n >>= 1;
}
return res;
}

void run() {
A[0][0] = A[0][1] = A[1][0] = BigInteger.ONE;
A[1][1] = BigInteger.ZERO;
System.out.println(qpow(A, 10)[0][0]);
}

public static void main(String[] args) {
       new Main().run();
}
}
```

4.6 Pell

```
import java.util.*;
import java.io.*;
import java.math.*;
public class G {
       Scanner cin = new Scanner(new BufferedInputStream(System.in));
       Set<BigInteger> ans = new TreeSet<>();
       BigInteger [] res = new BigInteger[10000];
       int n = 0;

       BigInteger [][] A = new BigInteger[2][2];
       BigInteger [][] B = new BigInteger[2][2];
       BigInteger lim = new
           BigInteger("1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

       int lower_bound(BigInteger x, int low, int high, BigInteger [] a) {
              int left = low, right = high;
              while (left < right) {
                     int mid = left + (right - left) / 2;
                     if (a[mid].compareTo(x) == -1) {
                            left = mid + 1;
```

```
                }else { right = mid; }
            }
            return left;
        }
        void run() {
            A[0][0] = BigInteger.valueOf(0); A[0][1] = BigInteger.valueOf(1); A[1][0] =
                BigInteger.valueOf(-1); A[1][1] = BigInteger.valueOf(6);
            B[0][0] = BigInteger.valueOf(0); B[0][1] = BigInteger.valueOf(1); B[1][0] =
                BigInteger.valueOf(-1); B[1][1] = BigInteger.valueOf(14);
            BigInteger n0 = BigInteger.valueOf(0), n1 = BigInteger.valueOf(2);
            while (true) {
                ans.add(n0);
                BigInteger _n0 = A[0][0].multiply(n0).add(A[0][1].multiply(n1));
                BigInteger _n1 = A[1][0].multiply(n0).add(A[1][1].multiply(n1));
                n0 = _n0; n1 = _n1;
                if (n0.compareTo(lim) > 0) break;
            }
            n0 = BigInteger.valueOf(0); n1 = BigInteger.valueOf(6);
            while (true) {
                ans.add(n0);
                BigInteger _n0 = B[0][0].multiply(n0).add(B[0][1].multiply(n1));
                BigInteger _n1 = B[1][0].multiply(n0).add(B[1][1].multiply(n1));
                n0 = _n0; n1 = _n1;
                if (n0.compareTo(lim) > 0) break;
            }
            for (BigInteger v : ans) {
                res[n++] = v;
            }
            while (cin.hasNext()) {
                BigInteger m = cin.nextBigInteger();
                int l = lower_bound(m, 0, n - 1, res);
                System.out.println(res[l]);
            }
        }

        public static void main(String[] args) {
            new G().run();
        }

}
```

4.7 STL

```
import java.util.*;
public class Main {
public static void main(String[] args) {
List<String> mylist1 = new ArrayList<>();
List<String> mylist2 = new LinkedList<>();
List<String> mylist3 = new Vector<>();

Vector<String> vec = new Vector<>();

Queue<String> que = new LinkedList<>();

Stack<String> sta = new Stack<>();

Set<String> myset = new HashSet<>();
Set<String> myset2 = new TreeSet<>(); // Good

Map<String, Integer> mymap = new HashMap<>(); // Good
Map<String, Integer> mymap2 = new TreeMap<>();
}
}
```

5 Others

5.1 2SAT

```
const int MAXN = 100005;
struct twoSAT{
    int n;
    vector<int> G[MAXN*2];
    bool mark[MAXN*2];
    int S[MAXN*2], c;

    void init(int n){
        this->n = n;
        for (int i=0; i<n*2; i++) G[i].clear();
        memset(mark, 0, sizeof(mark));
    }

    bool dfs(int x){
        if (mark[x^1]) return false;
        if (mark[x]) return true;
        mark[x] = true;
        S[c++] = x;
        for (int i=0; i<G[x].size(); i++)
            if (!dfs(G[x][i])) return false;
        return true;
    }

    void add_clause(int x, bool xval, int y, bool yval){
        x = x * 2 + xval;
        y = y * 2 + yval;
        G[x^1].push_back(y);
        G[y^1].push_back(x);
    }

    bool solve() {
        for (int i=0; i<n*2; i+=2){
            if (!mark[i] && !mark[i+1]){
                c = 0;
                if (!dfs(i)){
                    while (c > 0) mark[S[--c]] = false;
                    if (!dfs(i+1)) return false;
                }
            }
        }
        return true;
    }

    inline bool value(unsigned i){return mark[2*i+1];}
};
```

5.2 Euclid

```
LL similar_euclid(LL a, LL c, LL b, LL n) {
    if(!a) return (b / c) * (n + 1) % MOD;
    if(a >= c) return ((__int128)(a / c) % MOD * (__int128)n * (n + 1) / 2 % MOD +
        similar_euclid(a % c, c, b, n)) % MOD;
    if(b >= c) return ((__int128)(b / c) * (__int128)(n + 1) % MOD + similar_euclid(a, c, b % c,
        n)) % MOD;
    LL m = ((__int128)a * n + b) / c;
    return ((__int128)n * m % MOD - similar_euclid(c, a, c - b - 1, m - 1) + MOD) % MOD;
}
```

5.3 int128

```
inline void input(__int128 &s) {
    s = 0;
    char c = ' ';
    while (c > '9' || c < '0') c = getchar();
    while (c >= '0' && c <= '9') {
        s = s * 10 + c - '0';
        c = getchar();
    }
}
inline void output(__int128 x) {
    if (x > 9) output(x / 10);
    putchar(x % 10 + '0');
}
```

5.4 IO

```
int read() {
    int num = 0; char c; bool flag = false;
    while ((c = getchar()) == ' ' || c == '\n' || c == '\r');
    if (c == '-') flag = true;
    else num = c - '0';
    while (isdigit(c = getchar())) num = num * 10 + c - '0';
    return (flag ? -1 : 1) * num;
}
```

5.5 Largest-Rectangle

```
#include<bits/stdc++.h>
#define MAXN 100000
using namespace std;
int n;
int h[MAXN];
int L[MAXN],R[MAXN];
int st[MAXN];
void solve()
{
        int t=0;
        for(int i=0;i<n;i++)
       {
               while(t>0&&h[st[t-1]]>=h[i]) t--;
               L[i]=t==0?0:(st[t-1]+1);
               st[t++]=i;
       }
       t=0;
       for(int i=n-1;i>=0;i--)
       {
               while(t>0&&h[st[t-1]]>=h[i]) t--;
               R[i]=t==0?n:st[t-1];
               st[t++]=i;
       }
       long long res=0;
       for(int i=0;i<n;i++)
       {
               res=max(res,(long long)h[i]*(R[i]-L[i]));
       }
       printf("%lld\n",res);
}
```

5.6 Multiple-Backpack

```
#include<bits/stdc++.h>
#define MAXN 100005
int w[MAXN],v[MAXN],m[MAXN];
int dp[MAXW+1];
int deq[MAXW+1];
int deqv[MAXW+1];
void solve()
{
    for(int i=0;i<n;i++)
    {
        for(int a=0;a<w[i];a++)
        {
            int s=0,t=0;
            for(int j=0;j*w[i]+a<=W;j++)
            {
                int val=dp[j*w[i]+a]-j*v[i];
                while(s<t&&deqv[t-1]<=val) t--;
                deq[t]=j;
                deqv[t++]=val;
                dp[j*w[i]+a]=deqv[s]+j*v[i];
                if(deq[s]==j-m[i]) s++;
            }
        }
    }
    printf("%d\n",dp[W]);
}
```

5.7 Sum-over-subsets

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
```

```
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int n,a[MAXN],f[MAXN];
int main()
{
        scanf("%d",&n);
        for(int i=0;i<(1<<n);i++)
                scanf("%d",&a[i]);
        for(int i=0;i<(1<<n);i++)
                f[i]=a[i];
        for(int i=0;i<n;i++)
        {
                for(int mask=0;mask<(1<<n);mask++)
                        if(mask&(1<<i))
                                f[mask]+=f[mask^(1<<i)];
        }
        for(int i=0;i<(1<<n);i++)
                printf("%d ",f[i]);
        puts("");
        return 0;
}
```

5.8 SweepLine

```
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
using namespace std;
#define lowbit(x) ((x)&(-(x)))
#define MP make_pair
#define fi first
#define se second
// to replace or to modify!
#define LC(o) ((o)*2)
#define RC(o) ((o)*2+1)
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> pii;
const double PI = acos(-1.0);
const double eps = 1e-6;
const int INF = 0x3f3f3f3f;
const int maxn = 1e5 + 11;
int N, M, T;
struct E{
    double h; int l, r, f;
    bool operator < (const E& rhs) const {
        return h < rhs.h || (h == rhs.h && f > rhs.f);
    }
};
vector<E> lines;
double xl[maxn], yl[maxn], xr[maxn], yr[maxn];
vector<double> sorted;
double rev[maxn];

int mx[maxn<<3], mi[maxn<<3]; int cnt[maxn<<3];
```

```
void build(int L = 1, int R = M, int o = 1) {
    cnt[o] = mi[o] = mx[o] = 0;
    if (L == R) return;
    int M = (L + R) >> 1;
    build(L, M, LC(o));
    build(M + 1, R, RC(o));
}
void pushDown(int o) {
    if (cnt[o]) {
        cnt[LC(o)] += cnt[o]; cnt[RC(o)] += cnt[o];
        mi[LC(o)] += cnt[o]; mi[RC(o)] += cnt[o];
        mx[LC(o)] += cnt[o]; mx[RC(o)] += cnt[o];
        cnt[o] = 0;
    }
}
void update(int x1, int x2, int val, int L = 1, int R = M, int o = 1) {
    if (x1 <= L && R <= x2) {
        cnt[o] += val;
        mi[o] += val;
        mx[o] += val;
        return;
    }
    pushDown(o);
    int M = (L + R) >> 1;
    if (x1 <= M) update(x1, x2, val, L, M, LC(o));
    if (x2 > M) update(x1, x2, val, M + 1, R, RC(o));
    mi[o] = min(mi[LC(o)], mi[RC(o)]);
    mx[o] = max(mx[LC(o)], mx[RC(o)]);
}
//
double query(int L = 1, int R = M, int o = 1) {
    if (mi[o] >= 2) return rev[R+1] - rev[L];
    pushDown(o);
    double ans = 0.0;
    int M = (L + R) / 2;
    if (mx[LC(o)] >= 2) ans += query(L, M, LC(o));
    if (mx[RC(o)] >= 2) ans += query(M + 1, R, RC(o));
    return ans;
}

int main(){
    cin >> T;
    while (T--){
        cin >> N;
        double ans = 0;
        lines.clear(); sorted.clear();

        for (int i = 0; i < N; i++) {
            scanf("%lf%lf%lf%lf", &xl[i], &yl[i], &xr[i], &yr[i]);
            sorted.push_back(xl[i]); sorted.push_back(xr[i]);
        }
        sort(sorted.begin(), sorted.end());
        sorted.erase(unique(sorted.begin(), sorted.end()), sorted.end());
        for (int i = 0; i < N; i++) {
            int pos = lower_bound(sorted.begin(), sorted.end(), xl[i]) - sorted.begin() + 1;
            rev[pos] = xl[i];
            xl[i] = pos;
            pos = lower_bound(sorted.begin(), sorted.end(), xr[i]) - sorted.begin() + 1;
            rev[pos] = xr[i];
            xr[i] = pos;
        }

        M = sorted.size();
        for (int i = 0; i < N; i++){
            lines.push_back({yl[i],int(xl[i]),int(xr[i]),1});
            lines.push_back({yr[i],int(xl[i]),int(xr[i]),-1});
        }
        sort(lines.begin(), lines.end());

        build(1, M);
        update(lines[0].l, lines[0].r-1, lines[0].f, 1, M);
        for (int i = 1; i < 2*N; i++){
            ans += (lines[i].h-lines[i-1].h) * query();
            update(lines[i].l, lines[i].r-1, lines[i].f, 1, M);
        }

        printf("%.2f\n", ans);
    }
    return 0;
}
```

5.9 UsefulThings

```
int: 1e9+7,1e9+9,233,19260817,19660813,19990129
long long: 951970612352230049,963284339889659609,1048364250160580293,1045571042176595707
compiler settings: -fsanitize=undefined
linker settings/other linker options: -lubsan
```