OI 模板

panda_2134

2018 年 4 月 5 日

目录

1 图的 DFS 树

1.1 强连通分量

一有向图上每个点有非负权值，求一条路径，使得路径上点权值和最大。点和边都可以多次经过，但是权值只计入答案一次。

Solution: 缩点后直接在 DAG 上 DP.

GraphTheory/TarjanSCC.cpp

```cpp
#include <bits/stdc++.h>
#define fst first
#define snd second
using namespace std;

typedef pair<int, int> pii;
const int MAXN = 1e5, INF = 0x3f3f3f3f;

struct Graph {
    struct Edge {
        int v, next;
    };

    int n, m, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];

    void add_edge(int u, int v) {
        E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
    }
} G1, G2;

int dfs_clock, scc_cnt, sccno[MAXN+10], dfn[MAXN+10], low[MAXN+10];
int ans, topo_cnt, topo_seq[MAXN+10], w[MAXN+10],
    tot[MAXN+10], vis[MAXN+10], dp[MAXN+10];

stack<int> S;
void dfs(int u) {
    dfn[u] = low[u] = ++dfs_clock;
    S.push(u);
    for(int j=G1.head[u]; j; j=G1.E[j].next) {
        int v = G1.E[j].v;
        if(!dfn[v]) {
            dfs(v);
            low[u] = min(low[u], low[v]);
        } else if(!sccno[v])
            low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u]) {
        int v; ++scc_cnt;
        do {
            v = S.top(); S.pop();
            sccno[v] = scc_cnt;
            tot[scc_cnt] += w[v];
        } while(u != v);
    }
}

void Tarjan() {
    for(int u = 1; u <= G1.n; u++)
        if(!dfn[u]) dfs(u);
}

void scc_graph() {
    set<pii> evis;
    for(int u = 1; u <= G1.n; u++)
        for(int j=G1.head[u]; j; j=G1.E[j].next) {
            int v = G1.E[j].v;
            if(sccno[u] == sccno[v] || evis.count(make_pair(sccno[u], sccno[v])))
                continue;
            else {
                evis.insert(make_pair(sccno[u], sccno[v]));
                G2.add_edge(sccno[u], sccno[v]);
            }
        }
    G2.n = scc_cnt;
}

bool topo_dfs(int u) {
    vis[u] = -1;
    for(int j=G2.head[u]; j; j=G2.E[j].next) {
        int v = G2.E[j].v;
        if(vis[v] == -1 || (vis[v] == 0 && !topo_dfs(v)))
            return false;
    }
    vis[u] = 1;
    topo_seq[topo_cnt--] = u;
    return true;
}

bool toposort() {
    topo_cnt = G2.n;
    for(int u = G2.n; u >= 1; u--)
        if(vis[u] == 0 && !topo_dfs(u)) return false;
    return true;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void init() {
    int u, v;
    G1.n = readint(); G1.m = readint();
    for(int i = 1; i <= G1.n; i++)
        w[i] = readint();
    for(int i = 1; i <= G1.m; i++) {
        u = readint(); v = readint();
        G1.add_edge(u, v);
    }
    Tarjan(); scc_graph();
    assert(toposort());
}
```

```cpp

void work() {
    for(int i = G2.n; i >= 1; i--) {
        int u = topo_seq[i], maxv = 0;
        for(int j=G2.head[u]; j; j=G2.E[j].next) {
            int v = G2.E[j].v;
            if(dp[v] > maxv) maxv = dp[v];
        }
        dp[u] = tot[u] + maxv;
        ans = max(ans, dp[u]);
    }
    printf("%d", ans);
}

int main() {
    init(); work();
    return 0;
}
```

1.2 桥和割点

注意 child 代表 DFS 树中的儿子数目，且只在走完 DFS 树中某个儿子后判断割点条件。桥只需要把 34 行>=改为>即可。

GraphTheory/CutVertex.cpp

```cpp
 1  #include <bits/stdc++.h>
 2  using namespace std;
 3  
 4  struct Edge{ int v, next; };
 5  
 6  const int MAXN = 1e5, MAXM = 1e5;
 7  int n, m, cnt, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
 8  int dfs_clk, iscut[MAXN+10], dfn[MAXN+10], low[MAXN+10];
 9  
10  void add_edge(int u, int v) {
11      E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
12  }
13  
14  void add_pair(int u, int v) {
15      add_edge(u, v); add_edge(v, u);
16  }
17  
18  inline int readint() {
19      int f=1, r=0; char c=getchar();
20      while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
21      while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
22      return f*r;
23  }
24  
25  void dfs(int u, int fa) {
26      int child = 0;
27      dfn[u] = low[u] = ++dfs_clk;
28      for(int j=head[u]; j; j=E[j].next) {
29          int v = E[j].v;
30          if(v == fa) continue;
```

```cpp
        if(!dfn[v]) {
            dfs(v, u); ++child;
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u]) iscut[u] = true;
        } else if(dfn[v] < dfn[u] && v != fa)
            low[u] = min(low[u], dfn[v]);
    }
    if(child == 1 && fa == -1)
        iscut[u] = false;
}

int main() {
    int u, v;
    n = readint(); m = readint();
    for(int i = 1; i <= m; i++) {
        u = readint(); v = readint();
        add_pair(u, v);
    }
    for(int i = 1; i <= n; i++)
        if(!dfn[i]) dfs(i, -1);
    for(int i = 1; i <= n; i++)
        if(iscut[i]) ++cnt;
    printf("%d\n", cnt);
    for(int i = 1; i <= n; i++)
        if(iscut[i]) printf("%d ", i);
    return 0;
}
```

1.3 点双连通分量

GraphTheory/BCCVertex.cpp

```cpp
/*
    [UVaOJ1364] Knights of the Round Table
    好题。
    首先，问题可以转化成求无向图中不属于任何一个奇圈的点的数目。
    补集转换一下，变为求至少属于一个奇圈的点数目。
    和圈相关的问题，可以考虑BCC。和圈和点都有关，考虑点双连通分量。
    如果一个点双里面没有奇圈，那么它里面任何一个点显然都不属于任何一个奇圈。
    只要一个点双里面有一个奇圈，那么点双中任何一个点都至少属于一个奇圈，因为我们
    可以利用已有的奇圈来"包含"这个点双内的某个点。
    奇=奇+偶。如果奇圈上有点v1, v2，这个奇圈外有点u，不妨假设有u->v1, u->v2的路径
    （由双连通性质一定存在这样的v1, v2），则不管v1->u->v2含点数的奇偶性如何，总可以构造
    一个u->v2->现有奇圈一部分->v1->u的新奇圈！
    于是只需要找出所有BCC，然后对每个BCC二分图染色，即可得出答案。
    （注意割点bccno无意义）
*/
#include <bits/stdc++.h>
#define CLEAR(x) memset((x), 0, sizeof(x))
using namespace std;

struct Edge {
    int u, v, next;
};

const int MAXN = 1e3, MAXM = 1e6;
```

```
int n, m, e_ptr = 1, head[MAXN+10], hate[MAXN+10][MAXN+10];
Edge E[(MAXM+10)<<1];
int dfs_clk, bcc_cnt, dfn[MAXN+10], low[MAXN+10], bccno[MAXN+10], iscut[MAXN+10];
vector<int> bcc[MAXN+10]; int clr[MAXN+10];

void add_edge(int u, int v) {
    E[++e_ptr] = (Edge) { u, v, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v) {
    add_edge(u, v); add_edge(v, u);
}

inline int readint(){
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

stack<int> S;
void dfs(int u, int fa) {
    int child = 0;
    dfn[u] = low[u] = ++dfs_clk;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v; if(v == fa) continue;
        if(!dfn[v]) {
            S.push(j);
            dfs(v, u); ++child;
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u]) {
                iscut[u] = true;
                ++bcc_cnt; int cur;
                do {
                    cur = S.top(); S.pop();
                    if(bccno[E[cur].u] != bcc_cnt) {
                        bccno[E[cur].u] = bcc_cnt;
                        bcc[bcc_cnt].push_back(E[cur].u);
                    }
                    if(bccno[E[cur].v] != bcc_cnt) {
                        bccno[E[cur].v] = bcc_cnt;
                        bcc[bcc_cnt].push_back(E[cur].v);
                    }
                } while(E[cur].u != u || E[cur].v != v);
            }
        } else if(dfn[v] < dfn[u] && v != fa) {
            S.push(j);
            low[u] = min(low[u], dfn[v]);
        }
    }
    if(child == 1 && fa == -1)
        iscut[u] = false;
}

void find_bcc() {
    for(int i = 1; i <= n; i++)
        if(!dfn[i]) dfs(i, -1);
}

bool bipartite(int u, int b) {
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v; if(bccno[v] != b) continue;
        if(clr[v] == clr[u]) return false;
        if(!clr[v]) {
            clr[v] = 3 - clr[u];
            if(!bipartite(v, b)) return false;
        }
    }
    return true;
}

bool init() {
    int u, v;
    n = readint(); m = readint();
    if(!n && !m) return false;
    for(int i = 1; i <= m; i++) {
        u = readint(); v = readint();
        hate[u][v] = hate[v][u] = true;
    }
    for(u = 1; u <= n; u++)
        for(v = u + 1; v <= n; v++)
            if(!hate[u][v]) add_pair(u, v);
    return true;
}

void work() {
    int ans = n;
    find_bcc();
    for(int i = 1; i <= bcc_cnt; i++) {
        for(int j = 0; j < (int)bcc[i].size(); j++)
            bccno[bcc[i][j]] = i; // 割点 bccno 无意义
        CLEAR(clr);
        clr[bcc[i][0]] = 1;
        if(!bipartite(bcc[i][0], i))
            ans -= bcc[i].size();
    }
    printf("%d\n", ans);
}

void clear() {
    for(int i = 1; i <= bcc_cnt; i++) bcc[i].clear();
    n = m = 0; e_ptr = 1; CLEAR(head); CLEAR(hate);
    dfs_clk = bcc_cnt = 0;
    CLEAR(dfn); CLEAR(low); CLEAR(bccno); CLEAR(iscut); CLEAR(clr);
}

int main() {
    while(true) {
        if(!init()) break;
        work(); clear();
    }
    return 0;
}
```

1.4 边双连通分量

找出割边后 DFS，同时避免经过割边，即可求出边双连通分量。

2　最短路

2.1　负环

GraphTheory/NegCycle.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Edge {
    int v, len, next;
};

const int MAXN = 2e5, MAXM = 2e5, INF = 0x3f3f3f3f;

int T, cz, e_ptr = 1, n, m, head[MAXN+10], ins[MAXN+10]; Edge E[(MAXM+10)<<1];
int dist[MAXN+10];

void add_edge(int u, int v, int len) {
    E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v, int len) {
    add_edge(u, v, len); add_edge(v, u, len);
}

bool spfa(int u) {
    ins[u] = true;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v, len = E[j].len;
        if(dist[v] > dist[u] + len) {
            dist[v] = dist[u] + len;
            if(ins[v] || (!ins[v] && !spfa(v)))
                return false;
        }
    }
    ins[u] = false; // 回溯
    return true;
}

bool neg_cycle() {
    memset(ins, 0, sizeof(ins));
    fill(dist + 1, dist + n + 1, .0);
    for(int i = 1; i <= n; i++)
        if(!spfa(i)) return true;
    return false;
}

void init() {
    int u, v, w;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= m; i++) {
        scanf("%d%d%d", &u, &v, &w);
        if(w < 0)
            add_edge(u, v, w);
        else
            add_pair(u, v, w);
    }
}

void work() {
    puts(neg_cycle() ? "YES" : "NO");
}

void clear() {
    e_ptr = 2;
    memset(head, 0, sizeof(head));
}

int main() {
    int T;
    scanf("%d", &T);
    while(T--) {
        init(); work(); clear();
    }
    return 0;
}
```

2.2 Dijkstra

2.2.1 Using `std::priority_queue`

GraphTheory/Dijkstra–STL.cpp

```cpp
#include <bits/stdc++.h>
#define fst first
#define snd second
using namespace std;

typedef pair<int, int> HeapNode;

struct Edge {
    int v, len, next;
};

const int MAXN = 1e4, MAXM = 5e5, INF = 0x3f3f3f3f;
int n, m, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
int dist[MAXN+10], done[MAXN+10];

void add_edge(int u, int v, int len) {
    E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v, int len) {
    add_edge(u, v, len); add_edge(v, u, len);
}

void Dijkstra() {
    priority_queue<HeapNode, vector<HeapNode>, greater<HeapNode> > pq;
    memset(done, 0, sizeof(done));
    memset(dist, 0x3f, sizeof(dist));
    dist[s] = 0; pq.push(make_pair(dist[s], s));
    while(!pq.empty()) {
        HeapNode p = pq.top(); pq.pop();
        int u = p.snd;
        if(done[u]) continue;
        done[u] = true;
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v, len = E[j].len;
            if(dist[v] > dist[u] + len) {
                dist[v] = dist[u] + len;
                pq.push(make_pair(dist[v], v));
            }
        }
    }
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int u, v, w;
    n = readint(); m = readint(); s = readint();
    for(int i = 1; i <= m; i++) {
        u = readint(); v = readint(); w = readint();
        add_edge(u, v, w);
    }
    Dijkstra();
    for(int i = 1; i <= n; i++) {
        if(dist[i] < INF)
            printf("%d ", dist[i]);
        else
            printf("%d ", INT_MAX);
    }
    return 0;
}
```

2.2.2 Using `__gnu_pbds::priority_queue`

使用了扩展库 `pb_ds` 中的配对堆，自带修改堆内元素操作，速度更快。仅在允许使用 STL 扩展时才使用。

GraphTheory/Dijkstra–pb_ds.cpp

```
#include <bits/stdc++.h>
#include <bits/extc++.h>
#define fst first
#define snd second
using namespace std;

typedef pair<int, int> HeapNode;
typedef __gnu_pbds::priority_queue<HeapNode, greater<HeapNode>,
        __gnu_pbds::pairing_heap_tag> PairingHeap;

struct Edge {
    int v, len, next;
};

const int MAXN = 1e4, MAXM = 5e5, INF = 0x3f3f3f3f;
int n, m, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
int dist[MAXN+10]; PairingHeap pq; PairingHeap::point_iterator it[MAXN+10];

void add_edge(int u, int v, int len) {
    E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v, int len) {
    add_edge(u, v, len); add_edge(v, u, len);
}

void Dijkstra() {
    memset(it, 0, sizeof(it));
    memset(dist, 0x3f, sizeof(dist));
    dist[s] = 0; it[s] = pq.push(make_pair(dist[s], s));
    while(!pq.empty()) {
        HeapNode p = pq.top(); pq.pop();
        int u = p.snd;
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v, len = E[j].len;
            if(dist[v] > dist[u] + len) {
                dist[v] = dist[u] + len;
                if(it[v] == NULL)
                    it[v] = pq.push(make_pair(dist[v], v));
                else
                    pq.modify(it[v], make_pair(dist[v], v));
            }
        }
    }
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int u, v, w;
    n = readint(); m = readint(); s = readint();
    for(int i = 1; i <= m; i++) {
        u = readint(); v = readint(); w = readint();
        add_edge(u, v, w);
    }
    Dijkstra();
    for(int i = 1; i <= n; i++) {
        if(dist[i] < INF)
            printf("%d ", dist[i]);
        else
            printf("%d ", INT_MAX);
    }
    return 0;
```

```
}
```

3 网络流

3.1 最大流

3.2 Dinic

NetworkFlow/Dinic.cpp

```
#include <bits/stdc++.h>
using namespace std;

struct Edge {
    int v, flow, cap, next;
};

const int MAXN = 1e4, MAXM = 1e5, INF = 0x3f3f3f3f;
int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
int d[MAXN+10], cur[MAXN+10];

void AddEdge(int u, int v, int cap) {
    E[++e_ptr] = (Edge) { v, 0, cap, head[u] }; head[u] = e_ptr;
    E[++e_ptr] = (Edge) { u, 0,   0, head[v] }; head[v] = e_ptr;
}

bool BFS() {
    queue<int> Q;
    memset(d, 0xff, sizeof(d));
    Q.push(s); d[s] = 0;
    while(!Q.empty()) {
        int u = Q.front(); Q.pop();
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v, f = E[j].flow, c = E[j].cap;
            if(f < c && d[v] == -1) {
                d[v] = d[u] + 1;
                if(v == t) return true;
                else Q.push(v);
            }
        }
    }
    return false;
}

int DFS(int u, int flow) {
    if(u == t || flow == 0) return flow; // !!!!!
    int res = flow;
    for(int &j=cur[u]; j; j=E[j].next) { // !!!!!
        int v = E[j].v, f = E[j].flow, c = E[j].cap;
        if(f < c && d[v] == d[u] + 1) {
            int aug = DFS(v, min(res, c-f));
            E[j].flow += aug; E[j^1].flow -= aug;
            res -= aug;
            if(res == 0) break; // !!!!!
        }
    }
    return flow - res;
}

int Dinic() {
    int MaxFlow = 0, CurFlow = 0;
    while(BFS()) {
        memcpy(cur, head, sizeof(head));
        while((CurFlow = DFS(s, INF)))
            MaxFlow += CurFlow;
    }
    return MaxFlow;
}

int main() {
    int u, v, c;
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for(int i = 1; i <= m; i++) {
        scanf("%d%d%d", &u, &v, &c);
        AddEdge(u, v, c);
    }
    printf("%d", Dinic());
    return 0;
}
```

3.3 最小费用最大流

3.3.1 zkw 费用流

NetworkFlow/zkw.cpp

```
#include <bits/stdc++.h>
using namespace std;

typedef long long int64;
struct Edge {
    int u, v;
    int64 flow, cap, cost;
    int next;
};

const int MAXN = 5e3, MAXM = 5e4;
const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;
int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1]; // ** E[(MAXM+10)<<1] **
int64 MaxFlow, MinCost, dist[MAXN+10], inq[MAXN+10], vis[MAXN+10];

void add_edge(int u, int v, int64 cap, int64 cost) {
    E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
    E[++e_ptr] = (Edge) { v, u, 0,  0, -cost, head[v] }; head[v] = e_ptr;
}

bool spfa() {
    queue<int> Q;
    memset(dist, 0x3f, sizeof(dist));
    Q.push(t); dist[t] = 0; inq[t] = true;
    while(!Q.empty()) {
        int u = Q.front(); Q.pop(); inq[u] = false;
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
            if(f < c && dist[v] > dist[u] + len) {
                dist[v] = dist[u] + len;
                if(!inq[v]) {
                    inq[v] = true; Q.push(v);
                }
            }
        }
    }
    return dist[s] != LL_INF;
}

int64 dfs(int u, int64 flow) {
    if(u == t || flow == 0) return flow;
    vis[u] = true;
    int64 res = flow;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
        if(f < c && !vis[v] && dist[v] == dist[u] - len) {
            int64 aug = dfs(v, min(res, c-f));
            E[j].flow += aug; E[j^1].flow -= aug;
            res -= aug;
            if(res == 0LL) break;
        }
    }
    return flow - res;
}

void zkw() {
    int64 CurFlow = 0LL;
    while(spfa()) {
        while(memset(vis, 0, sizeof(vis)),
            CurFlow = dfs(s, LL_INF)) {
            MaxFlow += CurFlow;
            MinCost += dist[s] * CurFlow;
        }
    }
}

template<typename T>
inline void readint(T &x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

int main() {
    int u, v; int64 w, c;
    readint(n); readint(m); readint(s); readint(t);
    for(int i = 1; i <= m; i++) {
        readint(u); readint(v); readint(w); readint(c);
        add_edge(u, v, w, c);
    }
    zkw();
```

```cpp
    printf("%lld %lld", MaxFlow, MinCost);
    return 0;
}
```

3.3.2 Primal Dual

NetworkFlow/PrimalDual.cpp

```cpp
#include <bits/stdc++.h>
#include <bits/extc++.h>
#define fst first
#define snd second

using namespace std;

typedef long long int64;
typedef pair<int64, int> HeapNode;
typedef __gnu_pbds::priority_queue<HeapNode, greater<HeapNode>,
        __gnu_pbds::pairing_heap_tag> PairingHeap;
const int MAXN = 5e3, MAXM = 5e4;
const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;

struct Edge {
    int u, v;
    int64 flow, cap, cost;
    int next;
};

int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
int64 MaxFlow, MinCost, delta, dist[MAXN+10], vis[MAXN+10], inq[MAXN+10];
PairingHeap::point_iterator it[MAXN+10];

void add_edge(int u, int v, int64 cap, int64 cost) {
    E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
    E[++e_ptr] = (Edge) { v, u, 0,  0, -cost, head[v] }; head[v] = e_ptr;
}

void Reduce() {
    for(int i = 2; i <= e_ptr; i++)
        E[i].cost -= (dist[E[i].u] - dist[E[i].v]);
    delta += dist[s];
}

bool BellmanFord() {
    queue<int> Q;
    memset(dist, 0x3f, sizeof(dist));
    Q.push(t); dist[t] = 0; inq[t] = true;
    while(!Q.empty()) {
        int u = Q.front(); Q.pop(); inq[u] = false;
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
            if(f < c && dist[v] > dist[u] + len) {
                dist[v] = dist[u] + len;
                if(!inq[v]) {
                    inq[v] = true; Q.push(v);
                }
            }
        }
    }
    return dist[s] != LL_INF;
}

bool Dijkstra() {
    PairingHeap pq;
    memset(dist, 0x3f, sizeof(dist));
    memset(it, 0, sizeof(it));
    dist[t] = 0; it[t] = pq.push(make_pair(dist[t], t));
    while(!pq.empty()) {
        HeapNode t = pq.top(); pq.pop();
        int u = t.snd;
        for(int j=head[u]; j; j=E[j].next) {
            int v = E[j].v; int64 f = E[j^1].flow, c = E[j^1].cap, len = E[j^1].cost;
            if(f < c && dist[v] > dist[u] + len) {
                dist[v] = dist[u] + len;
                if(it[v] == NULL)
                    it[v] = pq.push(make_pair(dist[v], v));
                else
                    pq.modify(it[v], make_pair(dist[v], v));
            }
        }
    }
    return dist[s] != LL_INF;
}

int64 dfs(int u, int64 flow) {
    if(u == t || flow == 0) return flow;
    vis[u] = true;
    int64 res = flow;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
        if(f < c && !vis[v] && len == 0) {
            int64 aug = dfs(v, min(res, c-f));
            E[j].flow += aug; E[j^1].flow -= aug;
            res -= aug;
            if(res == 0) break;
        }
    }
    return flow - res;
}

void Augment() {
    int64 CurFlow = 0;
    while( memset(vis, 0, sizeof(vis)),
        (CurFlow = dfs(s, LL_INF)) ) {
        MaxFlow += CurFlow;
        MinCost += delta * CurFlow;
    }
}

void PrimalDual() {
    if(!BellmanFord()) return;
    Reduce(); Augment();
    while(Dijkstra()) {
```

```cpp
        Reduce(); Augment();
    }
}

template<typename T>
inline void readint(T &x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

int main() {
    int u, v; int64 w, c;
    readint(n); readint(m); readint(s); readint(t);
    for(int i = 1; i <= m; i++) {
        readint(u); readint(v); readint(w); readint(c);
        add_edge(u, v, w, c);
    }
    PrimalDual();
    printf("%lld %lld", MaxFlow, MinCost);
    return 0;
}
```

4 树

4.1 倍增 LCA

Tree/DoublingLCA.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Edge { int v, next; };

const int MAXN = 1e6, LOG = 20;
int n, q, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
int dep[MAXN+10], anc[MAXN+10][LOG+1];

void add_edge(int u, int v) { E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr; }
void add_pair(int u, int v) { add_edge(u, v); add_edge(v, u); }

void dfs(int u) {
    for(int i = 1; i <= LOG; i++)
        anc[u][i] = anc[anc[u][i-1]][i-1];
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(v == anc[u][0]) continue;
        anc[v][0] = u; dep[v] = dep[u] + 1;
        dfs(v);
    }
}

int lca(int u, int v) {
    if(dep[u] < dep[v]) swap(u, v);
    for(int i = LOG; i >= 0; i--)
        if(dep[anc[u][i]] >= dep[v])
            u = anc[u][i];
    if(u == v) return u;
    for(int i = LOG; i >= 0; i--)
        if(anc[u][i] != anc[v][i])
            u = anc[u][i], v = anc[v][i];
    u = anc[u][0], v = anc[v][0];
    return u;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int u, v;
    n = readint(); q = readint(); s = readint();
    for(int i = 1; i <= n-1; i++) {
        u = readint(); v = readint();
        add_pair(u, v);
    }
    dep[s] = 1; dfs(s);
    while(q--) {
        u = readint(); v = readint();
        printf("%d\n", lca(u, v));
    }
    return 0;
}
```

4.2 欧拉序列求 LCA

Tree/EulerTourLCA.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e6;

struct Edge {
    int v, next;
};

int n, q, s, e_ptr = 1, dfs_clock, head[MAXN+10]; Edge E[(MAXN+10)<<1];
int dfn[MAXN+10], dfs_seq[MAXN+10], idx[MAXN+10], euler_seq[(MAXN+10)<<1], st[(MAXN+10)<<1][22];
/*
    dfn: dfs-clock of vertex u
    idx: the index of vertex u in euler-tour sequence
    dfs_seq: the dfs sequence
*/

void add_edge(int u, int v) {
    E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v) {
    add_edge(u, v); add_edge(v, u);
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void dfs(int u, int fa) {
    euler_seq[++euler_seq[0]] = dfn[u] = ++dfs_clock;
    idx[u] = euler_seq[0]; dfs_seq[dfs_clock] = u;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(v == fa) continue;
        dfs(v, u);
        euler_seq[++euler_seq[0]] = dfn[u];
    }
}

void init_lca() {
    memset(st, 0x3f, sizeof(st));
    for(int i = 1; i <= euler_seq[0]; i++)
        st[i][0] = euler_seq[i];
    for(int j = 1; j <= 21; j++)
        for(int i = 1; i <= euler_seq[0] - (1<<j) + 1; i++) // bounds of sparse-table!
            st[i][j] = min(st[i][j-1], st[i + (1 << (j-1))][j-1]);
}

int query(int l, int r) {
    if(l > r) swap(l, r);
    int j;
    for(j = 0; (1 << (j+1)) <= (r-l+1); j++);
    return min(st[l][j], st[r - (1<<j) + 1][j]);
}

int lca(int u, int v) {
    return dfs_seq[query(idx[u], idx[v])];
}

int main() {
    int u, v;
    n = readint(); q = readint(); s = readint();
    for(int i = 1; i <= n-1; i++) {
        u = readint(); v = readint();
        add_pair(u, v);
    }
    dfs(s, -1); init_lca();
    while(q--) {
        u = readint(); v = readint();
        printf("%d\n", lca(u, v));
    }
    return 0;
}
```

4.3 树链剖分

Tree/HLD.cpp

```
// call Dfs1(1) and Dfs2(1, 1)
const int MAXN = 1e5;
int dfs_clock, Fa[MAXN+10], Son[MAXN+10], Sz[MAXN+10],
    Dep[MAXN+10], Top[MAXN+10], Dfn[MAXN+10];

void Dfs1(int u) { // Fa Son Sz Dep
    int maxsz = 0; Sz[u] = 1;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(v == Fa[u]) continue;
        Fa[v] = u; Dep[v] = Dep[u] + 1; // !
        Dfs1(v); Sz[u] += Sz[v];
        if(Sz[v] > maxsz) {
            maxsz = Sz[v];
            Son[u] = v;
        }
    }
}

```

```cpp
void Dfs2(int u, int anc) { // Top Dfn
    Dfn[u] = ++dfs_clock; Top[u] = anc;
    if(Son[u]) Dfs2(Son[u], anc);
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(v == Fa[u] || v == Son[u]) continue;
        Dfs2(v, v);
    }
}

int LCA(int u, int v) {
    while(Top[u] != Top[v]) {
        if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
        u = Fa[Top[u]];
    }
    if(Dep[u] > Dep[v]) swap(u, v);
    return u;
}

int HLDQuery(int u, int v) {
    int ret = -INF;
    while(Top[u] != Top[v]) {
        if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
        ret = max(ret, st_query(1, 1, n, Dfn[Top[u]], Dfn[u]));
        u = Fa[Top[u]];
    }
    if(Dep[u] > Dep[v]) swap(u, v);
    ret = max(ret, st_query(1, 1, n, Dfn[u], Dfn[v]));
    return ret;
}
```

4.4 点分治

Tree/DivConquerOnVertex.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Edge { int v, len, next; };

const int MAXN = 1e4, MAXK = 1e7;

int n, q, k, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
int ans, root, totsz, vis[MAXN+10], f[MAXN+10], sz[MAXN+10],
    dist[MAXN+10], mp[MAXK+10], pths[MAXN+10];

void add_edge(int u, int v, int len) {
    E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
}

void add_pair(int u, int v, int len) {
    add_edge(u, v, len); add_edge(v, u, len);
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void get_centroid(int u, int fa) {
    f[u] = 0, sz[u] = 1;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(vis[v] || v == fa) continue;
        get_centroid(v, u); sz[u] += sz[v];
        if(sz[v] > f[u]) f[u] = sz[v];
    }
    f[u] = max(f[u], totsz - sz[u]);
    if(f[u] < f[root]) root = u;
}

void count_nd(int u, int fa) {
    sz[u] = 1;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v;
        if(vis[v] || v == fa) continue;
        count_nd(v, u); sz[u] += sz[v];
    }
    return;
}

void get_dist(int u, int fa) {
    pths[++pths[0]] = dist[u];
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v, len = E[j].len;
        if(vis[v] || v == fa) continue;
        dist[v] = dist[u] + len;
        get_dist(v, u);
    }
}

int calc(int u, int w) {
    int ret = 0;
    dist[u] = w; pths[0] = 0;
    get_dist(u, -1);
    sort(pths + 1, pths + pths[0] + 1);
    for(int i = 1; i <= pths[0]; i++) {
        if(pths[i] <= k)
            ret += mp[k - pths[i]];
        mp[pths[i]]++;
    }
    for(int i = 1; i <= pths[0]; i++)
        mp[pths[i]]--;
    return ret;
}

void solve(int u) {
    ans += calc(u, 0);
    vis[u] = true;
    for(int j=head[u]; j; j=E[j].next) {
        int v = E[j].v, len = E[j].len;
        if(vis[v]) continue;
        ans -= calc(v, len);
        count_nd(v, -1); totsz = sz[v];
        root = 0;
        get_centroid(v, -1);
        solve(v);
    }
}

int main() {
    int a, b, c;
    n = readint(); q = readint();
    for(int i = 1; i <= n-1; i++) {
        a = readint(); b = readint(); c = readint();
        add_pair(a, b, c);
    }
    while(q--) {
        k = readint();
        f[root=0] = n;
        memset(vis, 0, sizeof(vis));
        get_centroid(1, -1);
        ans = 0;
        solve(root);
        puts(ans ? "AYE" : "NAY");
    }
}
```

5 单调数据结构

5.1 单调队列 (滑动窗口)

Monotonic/SlidingWindow.cpp

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e6;
int n, k, Hd, Tl, A[MAXN+10], Q[MAXN+10];

void SlideMin() {
    Hd = 1, Tl = 0;
    for(int i = 1; i <= k; i++) {
        while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
        Q[++Tl] = i;
    }
    printf("%d ", A[Q[Hd]]);
    for(int i = k+1; i <= n; i++) {
        while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
        while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
        Q[++Tl] = i;
        printf("%d ", A[Q[Hd]]);
    }
}

void SlideMax() {
    Hd = 1, Tl = 0;
    for(int i = 1; i <= k; i++) {
        while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
        Q[++Tl] = i;
    }
    printf("%d ", A[Q[Hd]]);
    for(int i = k+1; i <= n; i++) {
        while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
        while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
        Q[++Tl] = i;
        printf("%d ", A[Q[Hd]]);
    }
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    n = readint(); k = readint();
    for(int i = 1; i <= n; i++) A[i] = readint();
    SlideMin(); putchar(10); SlideMax();
    return 0;
}
```

5.2 单调栈

[JSOI2008] 最大数 注意：下标从栈底到顶递增，而值则递减。（一个数字前面的比它小的数肯定不会成为询问的答案）还有：可能 $L = 0$，此时 `lower_bound` 传入空区间，返回 L ！所以必须特判！

Monotonic/MaxNumber.cpp

```cpp
/*
 * [JSOI2008]最大数
 */

#include <bits/stdc++.h>
using namespace std;

const int MAXN = 2e5;
int q, mod, n, last, a[MAXN+10], s[MAXN+10];

int main() {
    char op; int x;
    cin.sync_with_stdio(false);
    cin.tie(NULL);
    cin >> q >> mod;
    while(q--) {
        cin >> op >> x;
        switch(op) {
            case 'Q':
                if(x == 0)
                    cout << (last = 0) << endl;
                else
                    cout << (last = a[*lower_bound(s + 1, s + s[0] + 1, n-x+1)]) << endl;
                break;
            case 'A':
                x = (x + last) % mod;
                while(s[0] && a[s[s[0]]] < x) --s[0];
                s[++s[0]] = ++n; a[n] = x;
                break;
        }
    }
}
```

6 线段树

6.1 Lazy-Tag

Solution: 暴力拆开式子后（或者根据《重难点手册》的结论），发现要维护区间的 $\sum x_i, \sum y_i, \sum x_i y_i, \sum x_i^2$, 同时要支持区间加和区间设置为 $S+i$ 和T_j. 在线段树上维护 $add_s, add_t, set_s, set_t$，然后推一推式子找出 Lazy-tag 更新主 Tag 的公式即可。几个坑点：

1. add_s, add_t 标记在下推的时候，不能赋值，要累加!!! 累加!!! 累加!!!
2. 只有 set_s, set_t 用 $-\infty$ 来标记不存在，add_s, add_t 必须用 0 标记不存在！不然是给自己找麻烦，多出来各种特判!!!

SegTree/CorrelationAnalyse.cpp

```cpp
/*
    [SDOI2017] 相关分析
    Coded by panda_2134
*/
#include <bits/stdc++.h>
#define LC(o) ((o)*2)
#define RC(o) ((o)*2+1)
#define Mid(x, y) (((x) + (y)) / 2)
using namespace std;

const double eps = 1e-6, NONE = -1e6;
const int MAXN = 1e5;

int dcmp(double x) {
    return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);
}

struct Info {
    double x, y, xy, x2;
    Info() { x = y = xy = x2 = .0; }
    Info(double a, double b, double c, double d):
        x(a), y(b), xy(c), x2(d) {}
    Info operator+(const Info &rhs) const {
        return Info(x + rhs.x, y + rhs.y, xy + rhs.xy, x2 + rhs.x2);
    }
    Info operator+=(const Info &rhs) { return *this = *this + rhs; }
};

struct Node {
    double x, y, xy, x2, add_s, add_t, set_s, set_t;
    Node() {
        x = y = xy = x2 = .0;
        add_s = add_t = .0;
        set_s = set_t = NONE;
    }
    void clear() { x = y = xy = x2 = .0; }
} nd[(MAXN+10)<<2];

int n, q;
double X[MAXN+10], Y[MAXN+10];
```

```

void Maintain(int o, double L, double R) {
    if(dcmp(nd[o].set_s - NONE) == 0) {
        // no set (if set_s exists, then set_t exists, and vice versa)
        assert(dcmp(nd[o].set_t - NONE) == 0);
        nd[o].clear();
        if(L != R) {
            nd[o].x = nd[LC(o)].x + nd[RC(o)].x;
            nd[o].y = nd[LC(o)].y + nd[RC(o)].y;
            nd[o].xy = nd[LC(o)].xy + nd[RC(o)].xy;
            nd[o].x2 = nd[LC(o)].x2 + nd[RC(o)].x2;
        }
    } else {
        nd[o].x2 = (R-L+1) * nd[o].set_s * nd[o].set_s
            + R * (R+1) * (2*R+1) / 6 - L * (L-1) * (2*L-1) / 6
            + nd[o].set_s * (L+R) * (R-L+1);
        nd[o].xy = (R-L+1) * nd[o].set_s * nd[o].set_t
            + (nd[o].set_s + nd[o].set_t) * (L+R) * (R-L+1) / 2
            + R * (R+1) * (2*R+1) / 6 - L * (L-1) * (2*L-1) / 6;
        nd[o].x = (R-L+1) * nd[o].set_s + (L+R) * (R-L+1) / 2;
        nd[o].y = (R-L+1) * nd[o].set_t + (L+R) * (R-L+1) / 2;
    }
    nd[o].x2 += 2 * nd[o].add_s * nd[o].x + (R-L+1) * nd[o].add_s * nd[o].add_s;
    nd[o].xy += nd[o].add_t * nd[o].x
             + nd[o].add_s * nd[o].y + (R-L+1) * nd[o].add_s * nd[o].add_t;
    nd[o].x += (R-L+1) * nd[o].add_s;
    nd[o].y += (R-L+1) * nd[o].add_t; // update last
}

void Pushdown(int o) {
    if(dcmp(nd[o].set_s - NONE) != 0) { // mark exist
        assert(dcmp(nd[o].set_t - NONE) != 0);
        nd[LC(o)].set_s = nd[RC(o)].set_s = nd[o].set_s;
        nd[LC(o)].set_t = nd[RC(o)].set_t = nd[o].set_t;
        nd[LC(o)].add_s = nd[RC(o)].add_s = .0;
        nd[LC(o)].add_t = nd[RC(o)].add_t = .0;
        nd[o].set_s = NONE;
        nd[o].set_t = NONE;
    }
    if(dcmp(nd[o].add_s) != 0) {
        nd[LC(o)].add_s += nd[o].add_s; //add 标记要累加!!!!!!!!!!
        nd[RC(o)].add_s += nd[o].add_s;
        nd[o].add_s = .0;
    }
    if(dcmp(nd[o].add_t) != 0) {
        nd[LC(o)].add_t += nd[o].add_t;
        nd[RC(o)].add_t += nd[o].add_t;
        nd[o].add_t = .0;
    }
}

Info Query(int o, int L, int R, int qL, int qR) {
    Maintain(o, L, R);
    if(qL <= L && R <= qR)
        return Info(nd[o].x, nd[o].y, nd[o].xy, nd[o].x2);
    else {
        Info ret;
        Pushdown(o);
        if(qL <= Mid(L, R)) ret += Query(LC(o), L, Mid(L, R), qL, qR);
        else Maintain(LC(o), L, Mid(L, R));
        if(qR >= Mid(L, R)+1) ret += Query(RC(o), Mid(L, R)+1, R, qL, qR);
        else Maintain(RC(o), Mid(L, R)+1, R);
        return ret;
    }
}

void BuildTree(int o, int L, int R) {
    if(L == R) {
        nd[o].add_s = X[L];
        nd[o].add_t = Y[L];
    } else {
        BuildTree(LC(o), L, Mid(L, R));
        BuildTree(RC(o), Mid(L, R)+1, R);
    }
    Maintain(o, L, R);
}

void Add(int o, int L, int R, int qL, int qR, double S, double T) {
    if(qL <= L && R <= qR) {
        nd[o].add_s += S;
        nd[o].add_t += T;
    } else {
        Pushdown(o);
        if(qL <= Mid(L, R)) Add(LC(o), L, Mid(L, R), qL, qR, S, T);
        else Maintain(LC(o), L, Mid(L, R));
        if(qR >= Mid(L, R)+1) Add(RC(o), Mid(L, R)+1, R, qL, qR, S, T);
        else Maintain(RC(o), Mid(L, R)+1, R);
    }
    Maintain(o, L, R);
}

void Set(int o, int L, int R, int qL, int qR, double S, double T) {
    if(qL <= L && R <= qR) {
        nd[o].add_s = nd[o].add_t = .0; // override 'add' mark
        nd[o].set_s = S;
        nd[o].set_t = T;
    } else {
        Pushdown(o);
        if(qL <= Mid(L, R)) Set(LC(o), L, Mid(L, R), qL, qR, S, T);
        else Maintain(LC(o), L, Mid(L, R));
        if(qR >= Mid(L, R)+1) Set(RC(o), Mid(L, R)+1, R, qL, qR, S, T);
        else Maintain(RC(o), Mid(L, R)+1, R);
    }
    Maintain(o, L, R);
}

void init() {
    scanf("%d%d", &n, &q);
    for(int i = 1; i <= n; i++)
        scanf("%lf", &X[i]);
    for(int i = 1; i <= n; i++)
        scanf("%lf", &Y[i]);
    BuildTree(1, 1, n);
}
```

```cpp

void work() {
    int op, L, R; double S, T;
    Info res;
    while(q--) {
        scanf("%d", &op);
        switch(op) {
            case 1:
                scanf("%d%d", &L, &R);
                res = Query(1, 1, n, L, R);
                printf("%.12lf\n",
                    (res.xy - res.x * res.y / (R-L+1)) / (res.x2 - res.x * res.x / (R-L+1)));
                break;
            case 2:
                scanf("%d%d%lf%lf", &L, &R, &S, &T);
                Add(1, 1, n, L, R, S, T);
                break;
            case 3:
                scanf("%d%d%lf%lf", &L, &R, &S, &T);
                Set(1, 1, n, L, R, S, T);
                break;
        }
    }
}

int main() {
    init(); work();
    return 0;
}
```

6.2 动态开点线段树

[P3380] 二逼平衡树 树状数组套动态开点线段树。

线段树一般都不写指针的，容易错……

SegTree/2BBalancedTree.cpp

```cpp
#include <bits/stdc++.h>
#define Mid(x, y) (((x)+(y)) >> 1)
using namespace std;

const int MAXN = 5e4, NOT_FOUND = 2147483647;

struct Query {
    int type, a, b, c;
} qry[MAXN+10];

int n, q, cnt, lc[MAXN*300], rc[MAXN*300], sumv[MAXN*300];
int rt[MAXN+10], w[MAXN+10], nums[(MAXN+10)<<1];
// nums 要开成所有数字的种类的大小！或者直接开输入的 4 倍！第二次错了！

void maintain(int o, int L, int R) {
    if(L != R)
        sumv[o] = sumv[lc[o]] + sumv[rc[o]];
}

```

```cpp
void st_add(int &o, int L, int R, int p, int val) {
    if(!o) o = ++cnt;
    if(L == R) sumv[o] += val;
    else {
        if(p <= Mid(L, R))
            st_add(lc[o], L, Mid(L, R), p, val);
        else
            st_add(rc[o], Mid(L, R)+1, R, p, val);
        maintain(o, L, R);
    }
}

int st_kth(vector<int> &o, vector<int> &his, int L, int R, int k) {
    if(L == R)
        return L;
    else {
        int lc_sum = 0;
        for(int &x : o)   lc_sum += sumv[lc[x]];
        for(int &x : his) lc_sum -= sumv[lc[x]];
        if(k <= lc_sum) {
            for(int &x : o)   x = lc[x];
            for(int &x : his) x = lc[x];
            return st_kth(o, his, L, Mid(L, R), k);
        } else {
            k -= lc_sum;
            for(int &x : o)   x = rc[x];
            for(int &x : his) x = rc[x];
            return st_kth(o, his, Mid(L, R)+1, R, k);
        }
    }
}

int st_sum(vector<int> &o, vector<int> &his, int L, int R, int qL, int qR) {
    int ret = 0;
    if(qL <= L && R <= qR) {
        for(int &x : o)   ret += sumv[x];
        for(int &x : his) ret -= sumv[x];
    } else {
        vector<int> o2 = o, his2 = his;
        if(qL <= Mid(L, R)) {
            for(int &x : o)   x = lc[x];
            for(int &x : his) x = lc[x];
            ret += st_sum(o, his, L, Mid(L, R), qL, qR);
        }
        o = o2, his = his2;
        if(qR >= Mid(L, R)+1) {
            for(int &x : o)   x = rc[x];
            for(int &x : his) x = rc[x];
            ret += st_sum(o, his, Mid(L, R)+1, R, qL, qR);
        }
    }
    return ret;
}

inline int lowbit(int x) { return x & (-x); }

inline void bit_sum(int p, vector<int> &o) {
    while(p > 0) {
        o.push_back(rt[p]);
        p -= lowbit(p);
    }
}

inline void bit_add(int p, int w, int val) {
    while(p <= n) {
        st_add(rt[p], 1, nums[0], w, val);
        p += lowbit(p);
    }
}

inline int kth(int L, int R, int k) {
    vector<int> o, his;
    bit_sum(R, o); bit_sum(L-1, his);
    return st_kth(o, his, 1, nums[0], k);
}

inline int getrank(int L, int R, int val) {
    vector<int> o, his;
    bit_sum(R, o); bit_sum(L-1, his);
    if(val != 1)
        return st_sum(o, his, 1, nums[0], 1, val - 1) + 1;
    else
        return 1;
}

inline int count(int L, int R, int val) {
    vector<int> o, his;
    bit_sum(R, o); bit_sum(L-1, his);
    return st_sum(o, his, 1, nums[0], val, val);
}

inline void modify(int p, int val) {
    bit_add(p, w[p], -1);
    w[p] = val;
    bit_add(p, w[p], 1);
}

inline int pre(int L, int R, int val) {
    int rk = getrank(L, R, val);
    if(rk == 1) return -NOT_FOUND;
    return kth(L, R, rk-1);
}

inline int suf(int L, int R, int val) {
    int rk = getrank(L, R, val), cnt = count(L, R, val);
    if(rk + cnt - 1 == R - L + 1) return NOT_FOUND;
    return kth(L, R, rk + cnt);
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c))  { r=r*10+c-'0';   c=getchar(); }
    return f*r;
```

```
}

int main() {
    int ans;
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++) {
        w[i] = readint();
        nums[++nums[0]] = w[i];
    }
    for(int i = 1; i <= q; i++) {
        qry[i].type = readint();
        switch(qry[i].type) {
            case 1: case 2: case 4: case 5:
                qry[i].a = readint(); qry[i].b = readint(); qry[i].c = readint();
                if(qry[i].type != 2) nums[++nums[0]] = qry[i].c;
                break;
            case 3:
                qry[i].a = readint(); qry[i].b = readint();
                nums[++nums[0]] = qry[i].b;
                break;
        }
    }

    sort(nums + 1, nums + nums[0] + 1);
    nums[0] = unique(nums + 1, nums + nums[0] + 1) - &nums[1];

    for(int i = 1; i <= n; i++) {
        w[i] = lower_bound(nums + 1, nums + nums[0] + 1, w[i]) - nums;
        bit_add(i, w[i], 1);
    }

    for(int i = 1; i <= q; i++) {
        switch(qry[i].type) {
            case 1: case 4: case 5:
                qry[i].c = lower_bound(nums + 1, nums + nums[0] + 1, qry[i].c) - nums;
                break;
            case 3:
                qry[i].b = lower_bound(nums + 1, nums + nums[0] + 1, qry[i].b) - nums;
                break;
        }
    }
    for(int i = 1; i <= q; i++) {
        switch(qry[i].type) {
            case 1:
                printf("%d\n", getrank(qry[i].a, qry[i].b, qry[i].c));
                break;
            case 2:
                printf("%d\n", nums[kth(qry[i].a, qry[i].b, qry[i].c)]);
                break;
            case 3:
                modify(qry[i].a, qry[i].b);
                break;
            case 4:
                ans = pre(qry[i].a, qry[i].b, qry[i].c);
                if(ans != -NOT_FOUND) ans = nums[ans];
                printf("%d\n", ans);
                break;
            case 5:
                ans = suf(qry[i].a, qry[i].b, qry[i].c);
                if(ans != NOT_FOUND) ans = nums[ans];
                printf("%d\n", ans);
                break;
        }
    }
    return 0;
}
```

6.3 可持久化线段树

SegTree/LongtermSegtree.cpp

```
#include <bits/stdc++.h>
#define Mid(x, y) (((x) + (y)) >> 1)
using namespace std;

const int MAXN = 1e6, BKT = 4e7;

int n, q, cnt, ver, w[MAXN+10], rt[MAXN+10], lc[BKT], rc[BKT], v[BKT];

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void build_tree(int &o, int L, int R) {
    o = ++cnt;
    if(L == R) v[o] = w[L];
    else {
        build_tree(lc[o], L, Mid(L, R));
        build_tree(rc[o], Mid(L, R)+1, R);
    }
}

void modify(int &o, int his, int L, int R, int p, int val) {
    o = ++cnt;
    if(L == R) v[o] = val;
    else {
        if(p <= Mid(L, R)) {
            rc[o] = rc[his];
            modify(lc[o], lc[his], L, Mid(L, R), p, val);
        } else {
            lc[o] = lc[his];
            modify(rc[o], rc[his], Mid(L, R)+1, R, p, val);
        }
    }
}

int query(int o, int L, int R, int p) {
    if(!o) return 0;
    if(L == R) return v[o];
    else {
        if(p <= Mid(L, R))
            return query(lc[o], L, Mid(L, R), p);
        else
            return query(rc[o], Mid(L, R)+1, R, p);
    }
}

void init() {
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++)
        w[i] = readint();
    build_tree(rt[++ver], 1, n);
}

void work() {
    int op, prv, idx, val;
    while(q--) {
        ++ver;
        prv = readint() + 1;
        op = readint();
        switch(op) {
            case 1:
                idx = readint(); val = readint();
                modify(rt[ver], rt[prv], 1, n, idx, val);
                break;
            case 2:
                idx = readint();
                printf("%d\n", query(rt[prv], 1, n, idx));
                rt[ver] = rt[prv];
                break;
        }
    }
}

int main() {
    init(); work();
    return 0;
}
```

7 离线二维数点

7.1 带修改

7.1.1 静态：线段树 + 扫描线

（未实现）

7.1.2 动态：CDQ 分治

陌上花开：三维数点 = 动态二维数点

注意去重处理的坑点：

1. 在分治统计的时候，无论是加点还是查询答案，都一定要考虑到多个重复点的贡献！

2. 注意去重方法：用`map`比较方便。

2D/cdq.cpp

```cpp
#include <bits/stdc++.h>
#define fst first
#define snd second
using namespace std;

struct Point {
    int x, y, z, idx;
    bool operator<(const Point &rhs) const {
        return x == rhs.x ?
        (
            y == rhs.y ?
            z < rhs.z : y < rhs.y
        ) : x < rhs.x;
    }
    bool operator==(const Point &rhs) const {
        return x == rhs.x && y == rhs.y && z == rhs.z;
    }
};

struct Query {
    int x, y, z, idx, type;
    bool operator<(const Query &rhs) const {
        return y == rhs.y ? type < rhs.type : y < rhs.y;
    }
};

const int MAXN = 3e5;
map<Point, int> p_cnt;
int n, k, q_cnt, totv[MAXN+10], ans[MAXN+10], anscnt[MAXN+10], bit[MAXN+10];
Query qry[MAXN+10], T[MAXN+10];

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void init() {
    int x, y, z;
    n = readint(); k = readint();
    for(int i = 1; i <= n; i++) {
        x = readint(); y = readint(); z = readint();
        p_cnt[(Point){ x, y, z, i }]++;
    }
    for(auto p : p_cnt) {
        totv[p.fst.idx] = p.snd; ans[p.fst.idx] = -p.snd;
        qry[++q_cnt] = { p.fst.x, p.fst.y, p.fst.z, p.fst.idx, 1 };
        qry[++q_cnt] = { p.fst.x, p.fst.y, p.fst.z, p.fst.idx, 2 };
    }
}

inline int lowbit(int x) {
    return x & (-x);
```

```
}

void add(int p, int val) {
    while(p <= k) {
        bit[p] += val;
        p += lowbit(p);
    }
}

int sum(int p) {
    int ret = 0;
    while(p > 0) {
        ret += bit[p];
        p -= lowbit(p);
    }
    return ret;
}

void solve(int L, int R) {
    if(L + 1 >= R) return;

    int pl, pr, M, p;
    M = L + (R - L) / 2;
    pl = L, pr = M; p = L;

    solve(L, M); solve(M, R);

    while(pl < M || pr < R) {
        if(pr >= R || (pl < M && qry[pl] < qry[pr])) {
            if(qry[pl].type == 1)
                add(qry[pl].z, totv[qry[pl].idx]);
            T[p++] = qry[pl++];
        } else {
            if(qry[pr].type == 2)
                ans[qry[pr].idx] += totv[qry[pr].idx] * sum(qry[pr].z);
            T[p++] = qry[pr++];
        }
    }

    pl = L, pr = M;
    while(pl < M || pr < R) {
        if(pr >= R || (pl < M && qry[pl] < qry[pr])) {
            if(qry[pl].type == 1)
                add(qry[pl].z, -totv[qry[pl].idx]);
            pl++;
        } else pr++;
    }
    assert(!sum(k));
    for(int i = L; i < R; i++) qry[i] = T[i];
}

void work() {
    sort(qry + 1, qry + q_cnt + 1, [](const Query &lhs, const Query &rhs) {
        return lhs.x == rhs.x ? lhs.type < rhs.type : lhs.x < rhs.x;
    });
    solve(1, q_cnt + 1);
    for(int i = 1; i <= q_cnt; i++) {
        if(!totv[i]) continue;
        anscnt[ans[i] / totv[i]] += totv[i];
    }
    for(int i = 0; i < n; i++)
        printf("%d\n", anscnt[i]);
}

int main() {
    init(); work();
    return 0;
}
```

8 在线二维数点

8.0.1 动态：二维线段树

时间复杂度 插入$O(\lg^2 n)$ – 查询$O(\lg n)$ 空间复杂度 $O(n^2)$

8.0.2 动态：树状数组套动态开点线段树

（见上方二逼平衡树）

8.0.3 动态：树状数组套平衡树

BalancedTree/DynamicInversion.cpp

```
#include <bits/stdc++.h>
using namespace std;

typedef long long int64;

const int MAXN = 1e5;

//---------------Treap--------------------
struct Node *null;
struct Node {
    Node *ch[2];
    int v, r, sz;
    void init(int v_) {
        v = v_; r = rand(); sz = 1;
        ch[0] = ch[1] = null;
    }
    Node() {
        init(0);
    }
    int cmp(int x) {
        return (x == v ? -1 : (x > v ? 1 : 0));
    }
    void maintain() {
        if(this != null)
            sz = ch[0]->sz + ch[1]->sz + 1;
    }
};

```

```
int n, m, w[MAXN+10], mp[MAXN+10], bitval[MAXN+10];
Node* bit[MAXN+10]; int64 ans;

const int alloc_size = 65536;
queue<Node*> pool;
void renew() {
    Node* pit = new Node[alloc_size];
    for(int i = 0; i < alloc_size; i++)
        pool.push(pit++);
}

Node* newnode(int v) {
    if(pool.empty()) renew();
    Node* ret = pool.front(); pool.pop();
    ret->init(v);
    return ret;
}

void delnode(Node* &o) {
    pool.push(o); o = null;
}

void rotate(Node* &o, int d) {
    Node* k = o->ch[d^1];
    o->ch[d^1] = k->ch[d];
    k->ch[d] = o;
    o->maintain(); k->maintain();
    o = k;
}

void insert(Node* &o, int val) {
    if(o == null)
        o = newnode(val);
    else {
        int d = o->cmp(val);
        if(d == -1) return;
        insert(o->ch[d], val);
        o->maintain();
        if((o->r) > (o->ch[d]->r))
            rotate(o, d^1);
    }
}

void erase(Node* &o, int val) {
    if(o == null) return;
    int d = o->cmp(val);
    if(d == -1) {
        if(o->ch[1] == null) {
            Node* lhs = o->ch[0];
            delnode(o); o = lhs;
        } else if(o->ch[0] == null) {
            Node* rhs = o->ch[1];
            delnode(o); o = rhs;
        } else {
            int d = (o->ch[0]->r) < (o->ch[1]->r) ? 1 : 0;
            rotate(o, d);
            erase(o->ch[d], val);
        }
    } else
        erase(o->ch[d], val);
    o->maintain();
}

int getrank(Node* o, int val) {
    if(o == null) return 0;
    int d = o->cmp(val);
    if(d == -1) return o->ch[0]->sz;// ´ËÊ±rank = <valµÄÔªËØ¸öÊý
    return getrank(o->ch[d], val) + d * (o->ch[0]->sz + 1);
}

//----------------------------------------

void init_null() {
    null = new Node(); null->sz = 0;
    for(int i = 0; i <= MAXN; i++)
        bit[i] = null;
}

inline int lowbit(int x) { return x & (-x); }

int bit_sum(int p) {
    int ret = 0;
    while(p > 0) {
        ret += bitval[p];
        p -= lowbit(p);
    }
    return ret;
}

void bit_add(int p, int val) {
    while(p <= n) {
        bitval[p] += val;
        p += lowbit(p);
    }
}

void nd_bit_sum(int p, int &sz, Node* o[]) {
    while(p > 0) {
        o[sz++] = bit[p];
        p -= lowbit(p);
    }
}

void nd_bit_add(int p, int val) {
    while(p <= n) {
        insert(bit[p], val);
        p += lowbit(p);
    }
}

void nd_bit_del(int p, int val){
    while(p <= n) {
        erase(bit[p], val);
        p += lowbit(p);
    }
}

int query(int x, int y) {
    int ret = 0, sz = 0; Node* vec[50];
    nd_bit_sum(x, sz, vec);
    for(int i = 0; i < sz; i++) {
        Node* ptr = vec[i];
        ret += getrank(ptr, y);
    }
    return ret;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void init() {
    n = readint(); m = readint();
    for(int i = 1; i <= n; i++) {
        w[i] = readint(); mp[w[i]] = i;
        bit_add(w[i], 1); ans += bit_sum(n) - bit_sum(w[i]);
        nd_bit_add(i, w[i]);
    }
}

void work() {
    int i;
    while(m--) {
        i = mp[readint()];
        printf("%lld\n", ans);
        ans -= query(i-1, n+1) - query(i-1, w[i]+1);
        ans -= query(n, w[i]) - query(i, w[i]);
        nd_bit_del(i, w[i]);
    }
}

int main() {
    srand(66623333);
    init_null();
    init(); work();
}
```

9 平衡树

9.1 Treap

BalancedTree/Treap.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Node *null, *rt;
struct Node {
    int v, r, sz, cnt;
    Node *ch[2];
    Node(int v_) {
        v = v_; r = rand(); sz = cnt = 1;
        ch[0] = ch[1] = null;
    }
    int cmp(int val) {
        return val == v ? -1 : (val > v ? 1 : 0);
    }
    void maintain() {
        if(this == null) return;
        sz = ch[0]->sz + ch[1]->sz + cnt;
    }
};

void init_null() {
    null = new Node(0); null->sz = null->cnt = 0;
    rt = null;
}

void rotate(Node* &o, int d) {
    Node* k = o->ch[d^1];
    o->ch[d^1] = k->ch[d];
    k->ch[d] = o;
    o->maintain(); k->maintain();
    o = k;
}

void insert(Node* &o, int val) {
    if(o == null) {
        o = new Node(val);
        return;
    } else {
        int d = o->cmp(val);
        if(d == -1) {
            ++o->cnt; o->maintain();
        } else {
            insert(o->ch[d], val);
            o->maintain();
            if(o->ch[d]->r < o->r) rotate(o, d^1);
        }
    }
}

void erase(Node* &o, int val) {
    int d = o->cmp(val);
    if(d == -1) {
        if(o->cnt == 1) {
            if(o->ch[1] == null) {
                Node* lhs = o->ch[0];
                delete o;
                o = lhs;
            } else if(o->ch[0] == null) {
                Node* rhs = o->ch[1];
                delete o;
                o = rhs;
            } else {
                int d2 = (o->ch[0]->r) > (o->ch[1]->r);
                rotate(o, d2^1);
                erase(o->ch[d2^1], val);
            }
        } else
            --o->cnt;
    } else
        erase(o->ch[d], val);
    o->maintain();
}

Node* kth(Node* o, int k) {
    int d = (k >= o->ch[0]->sz + 1 && k <= o->ch[0]->sz + o->cnt) ? -1 :
            (k <= o->ch[0]->sz ? 0 : 1);
    if(d == -1) return o;
    if(d == 1) k -= (o->sz - o->ch[1]->sz);
    return kth(o->ch[d], k);
}

int get_rank(Node* o, int val) {
    if(o == null) return 1;
    int d = o->cmp(val);
    if(d == -1) return o->ch[0]->sz + 1;
    return get_rank(o->ch[d], val) + d * (o->sz - o->ch[1]->sz);
}

Node* find(Node* o, int val) {
    if(o == null) return o;
    int d = o->cmp(val);
    if(d == -1) return o;
    else return find(o->ch[d], val);
}

Node* pre(int val) {
    int rk = get_rank(rt, val);
    return rk != 1 ? kth(rt, rk-1) : null;
}

Node* succ(int val) {
    int rk = get_rank(rt, val);  // !!!!!!!!!
    return rk != (rt->sz) ? kth(rt, rk+find(rt, val)->cnt) : null;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    srand(66623333);
    int q, op, x;
    init_null();
    q = readint();
    while(q--) {
        op = readint();
        switch(op) {
            case 1:
                x = readint(); insert(rt, x);
                break;
            case 2:
                x = readint(); erase(rt, x);
                break;
            case 3:
                x = readint(); insert(rt, x);
                printf("%d\n", get_rank(rt, x));
                erase(rt, x);
                break;
            case 4:
                x = readint();
                printf("%d\n", kth(rt, x)->v);
                break;
            case 5:
                x = readint(); insert(rt, x);
                assert(pre(x) != null);
                printf("%d\n", pre(x)->v);
                erase(rt, x);
                break;
            case 6:
                x = readint(); insert(rt, x);
                assert(succ(x) != null);
                printf("%d\n", succ(x)->v);
                erase(rt, x);
                break;
        }
    }
    return 0;
}
```

9.2 Splay

BalancedTree/Splay.cpp

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e5;

struct Node *null, *rt;
struct Node {
    int v, sz; bool flip;
    Node* ch[2];
    Node(int v_) { v = v_, sz = 1; flip = false; ch[0] = ch[1] = null; }
    int cmp(int k) {
        return k == ch[0]->sz + 1 ? -1 : (k > ch[0]->sz + 1 ? 1 : 0);
    }
    void rev() {
        if(this == null) return;
        flip ^= 1;
    }
    void maintain() {
        if(this == null) return;
        sz = ch[0]->sz + ch[1]->sz + 1;
    }
    void pushdown() {
        if(flip) {
            flip = false;
            ch[0]->rev(); ch[1]->rev();
            swap(ch[0], ch[1]);
        }
    }
};
int n, m;

void init_null() {
    null = new Node(0); null->sz = 0;
    rt = null;
}

void rotate(Node* &o, int d) {
    Node* k = o->ch[d^1];
    o->pushdown(); k->pushdown();
    o->ch[d^1] = k->ch[d];
    k->ch[d] = o;
    o->maintain(); k->maintain();
    o = k;
}

void splay(Node* &o, int k) {
    o->pushdown();
    int d = o->cmp(k);
    if(d == 1) k -= (o->ch[0]->sz + 1);
    if(d != -1) {
        Node* p = o->ch[d];
        p->pushdown();
        int d2 = p->cmp(k);
        if(d2 == 1) k -= (p->ch[0]->sz + 1);
        if(d2 != -1) {
            splay(p->ch[d2], k);
            if(d == d2) rotate(o, d^1);
            else rotate(o->ch[d], d);
        }
        rotate(o, d^1);
    }
}

Node* merge(Node* lhs, Node* rhs) {
    splay(lhs, lhs->sz);
    lhs->pushdown();
    lhs->ch[1] = rhs;
    lhs->maintain();
    return lhs;
}

void split(Node* o, int k, Node* &lhs, Node* &rhs) {
    splay(o, k);
    o->pushdown();
    lhs = o, rhs = o->ch[1];
    o->ch[1] = null; o->maintain(); // 赋值后再断开和右儿子的连接，并维护 sz!
}

void traverse(Node* o) {
    if(o == null) return;
    o->pushdown();
    traverse(o->ch[0]);
    if(o->v > 0) printf("%d ", o->v);
    traverse(o->ch[1]);
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int l, r; Node *a, *b, *c;
    init_null();

    n = readint(); m = readint();

    rt = new Node(0); // dummy
    for(int i = 1; i <= n; i++) rt = merge(rt, new Node(i));
    rt = merge(rt, new Node(0)); // dummy

    while(m--) {
        l = readint() + 1, r = readint() + 1;
        split(rt, l-1, a, b); split(b, r-l+1, b, c);
        b->rev();
        rt = merge(a, merge(b, c));
    }

    traverse(rt);
    return 0;
}
```

10 动态树

10.1 Link-cut Tree

(似乎发现了以前模板里面判断边是否存在的一个错误……)

LCT/LCT.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 3e5;

struct Node *null;
struct Node {
    int v, sumv; bool rev;
    Node *fa, *ch[2];
    Node(int v_) {
        v = sumv = v_; rev = false;
        fa = ch[0] = ch[1] = null;
    }
    bool splayrt() { return fa->ch[0] != this && fa->ch[1] != this; }
    int rel() { return splayrt() ? -1 : (fa->ch[0] == this ? 0 : 1); }
    void mark_rev() { rev ^= 1; }
    void maintain() {
        if(this == null) return;
        sumv = ch[0]->sumv ^ v ^ ch[1]->sumv;
    }
    void pushdown() {
        if(rev) {
            rev = false;
            ch[0]->mark_rev(); ch[1]->mark_rev();
            swap(ch[0], ch[1]);
        }
    }
} *nd[MAXN+10];

set<pair<Node*, Node*> > edges;

void init_null() {
    null = new Node(0);
    for(int i = 0; i <= MAXN; i++)
        nd[i] = null;
}

void rotate(Node* o) {
    Node *x, *y, *k; int d, d2;
    x = o->fa; y = x->fa;
    d = o->rel(); d2 = x->rel();
    k = o->ch[d^1];
    if(!x->splayrt()) y->ch[d2] = o;
    o->fa = y;
    o->ch[d^1] = x; x->fa = o;
    x->ch[d] = k; k->fa = x;
    x->maintain(); o->maintain();
}

```

```
void splay(Node* o) {
    static Node *x, *stk[MAXN+10]; int d, d2, p = 0;
    for(stk[p=1] = o; !stk[p]->splayrt(); p++)
        stk[p+1] = stk[p]->fa;
    for(; p; p--) stk[p]->pushdown();
    while(!o->splayrt()) {
        x = o->fa;
        d = o->rel(); d2 = x->rel();
        if(d2 != -1) {
            if(d == d2) rotate(x);
            else rotate(o);
        }
        rotate(o);
    }
}

void access(Node* o) {
    for(Node* t = null; o != null; t = o, o = o->fa) {
        splay(o); o->ch[1] = t; o->maintain();
    }
}

Node* get_root(Node* o) {
    access(o); splay(o);
    while(o->ch[0] != null) o = o->ch[0];
    splay(o); return o;
}

void make_root(Node* o) {
    access(o); splay(o); o->mark_rev();
}

void add_edge(Node* u, Node* v) {
    if(u > v) swap(u, v);
    edges.insert(make_pair(u, v));
}

bool has_edge(Node* u, Node* v) {
    if(u > v) swap(u, v); // 统一存储
    return edges.count(make_pair(u, v)) > 0;
}

void link(Node *u, Node *v) {
    if(get_root(u) == get_root(v)) return;
    make_root(u); splay(u); u->fa = v;
    add_edge(u, v);
}

void cut(Node *u, Node *v) {
    if(get_root(u) != get_root(v)) return;
    make_root(u); access(v); splay(u);
    u->pushdown();
    if(has_edge(u, v)) { // 不是 u->ch[1] == v!!!
        u->ch[1] = null; v->fa = null; // v->fa !
    }
    u->maintain(); v->maintain();
}
```

```cpp

int n, q;

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int op, x, y;
    init_null();
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++)
        nd[i] = new Node(readint());
    while(q--) {
        op = readint(); x = readint(); y = readint();
        switch(op) {
            case 0:
                assert(get_root(nd[x]) == get_root(nd[y]));
                make_root(nd[x]); access(nd[y]); splay(nd[x]);
                printf("%d\n", nd[x]->sumv);
                break;
            case 1:
                link(nd[x], nd[y]);
                break;
            case 2:
                cut(nd[x], nd[y]);
                break;
            case 3:
                splay(nd[x]); nd[x]->v = y; nd[x]->maintain();
                break;
        }
    }
    return 0;
}
```

11 字符串

11.1 KMP 字符串匹配

1-indexed

String/KMP.cpp

```cpp
#include <bits/stdc++.h>
const int MAXN = 1000000;

int fail[MAXN+5];

void KMP(char* a,char* b){
    int na = strlen(a+1),nb = strlen(b+1); //注意 +1
    //Init
    fail[1] = 0;//! static 在每次进入函数时保留上次修改的值
    for(int i = 2; i <= nb; i++){
        int j = fail[i-1];//尝试扩展已经匹配部分
        //无法匹配则缩短前后缀
        while(j != 0 && b[j+1] != b[i]) j = fail[j];
        //用已经匹配部分更新 fail 数组
        if(b[j+1] == b[i]) fail[i] = j+1;// !!
        //无法匹配前后缀
        else fail[i] = 0;
    }
    //Match
    for(int i = 1,j = 0; i <= na; i++){
        //缩短前后缀
        while(j != 0 && b[j+1] != a[i]) j = fail[j];
        if(b[j+1] == a[i]) j++; //成功匹配
        if(j == nb){
            printf("%d\n", i - nb + 1);
            j = fail[j];
            //j = 0; //如果两个匹配部分不可重叠
        }
    }
}

int main() {
    char a[MAXN+5], b[MAXN+5];
    scanf("%s", a+1);
    scanf("%s", b+1);
    KMP(a, b); int t = strlen(b+1);
    for(int i = 1; i <= t; i++)
        printf("%d ", fail[i]);
    return 0;
}
```

11.2 AC 自动机

0-indexed

String/ACAutomaton.cpp

```cpp
#include <bits/stdc++.h>
```

```cpp
#define CLEAR(x) memset((x), 0, sizeof(x))
using namespace std;

const int SIGMA = 26, MAX_TEMP_LEN = 70, MAXN = 150,
MAX_LEN = 1e6, MAX_NODE = MAXN * MAX_TEMP_LEN;

int N, sz, ch[MAX_NODE + 10][SIGMA + 2], f[MAX_NODE + 10], last[MAX_NODE+10],
    val[MAX_NODE + 10], found_cnt[MAX_NODE+10];
char str[MAX_LEN+10], tpl[MAXN+10][MAX_TEMP_LEN+10];
unordered_map<string, int> ms;

inline int idx(char c) { return c - 'a' + 1; }

void insert(char *str) {
    int u = 0, len = strlen(str);
    for(int i = 0; i < len; i++) {
        int c = idx(str[i]);
        if(!ch[u][c]) ch[u][c] = ++sz;
        u = ch[u][c];
    }
    ms[string(str)] = u;
    ++val[u];
}

void get_fail() {
    queue<int> Q;
    f[0] = 0;
    for(int c = 1; c <= SIGMA; c++) if(ch[0][c]) {
        int v = ch[0][c];
        f[v] = last[v] = 0;
        Q.push(v);
    }
    while(!Q.empty()) {
        int u = Q.front(); Q.pop();
        for(int c = 1; c <= SIGMA; c++) {
            int v = ch[u][c];
            if(!v) {
                ch[u][c] = ch[f[u]][c];
                continue;
            }

            Q.push(v);

            int u2 = f[u];
            while(u2 && !ch[u2][c]) u2 = f[u2];
            f[v] = ch[u2][c];
            last[v] = val[f[v]] ? f[v] : last[f[v]];
        }
    }
}

void found(int u) {
    for(; u; u = last[u])
        found_cnt[u] += val[u];
}

void search(char *str) {
    int u = 0, len = strlen(str);
    for(int i = 0; i < len; i++) {
        int c = idx(str[i]);
        u = ch[u][c];
        if(val[u]) found(u);
        else if(last[u]) found(last[u]);
    }
}

inline void readstr(char *str) {
    char c=getchar(); int p=0;
    while(!isalnum(c) && !ispunct(c)) c = getchar();
    while(isalnum(c) || ispunct(c)) {
        str[p++] = c;
        c = getchar();
    }
    str[p++] = '\0';
}

int main() {
    while(true) {
        int ans = 0;
        sz = 0; CLEAR(ch); CLEAR(f); CLEAR(found_cnt);
        CLEAR(last); CLEAR(tpl); CLEAR(val); CLEAR(str);

        scanf("%d", &N); if(N == 0) break;
        for(int i = 1; i <= N; i++) {
            readstr(tpl[i]); insert(tpl[i]);
        }
        get_fail();

        readstr(str); search(str);

        for(int i = 0; i <= sz; i++)
            ans = max(ans, found_cnt[i]);
        printf("%d\n", ans);
        for(int i = 1; i <= N; i++)
            if(found_cnt[ms[string(tpl[i])]] == ans)
                printf("%s\n", tpl[i]);
    }
    return 0;
}
```

11.3 后缀数组

0-indexed

String/SuffixArray.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXLEN = 1e6, SIGMA = 100;

inline int idx(char c) {
    if(!c) return 0;
    else if(isdigit(c)) return c - '0' + 1;
    else if(isupper(c)) return c - 'A' + 1 + 10;
    else if(islower(c)) return c - 'a' + 1 + 10 + 26;
    else throw "Invalid Character";
}

struct SuffixArray {
    int sa[MAXLEN+10], rk[MAXLEN+10], buf[3][MAXLEN+10], height[MAXLEN+10], c[MAXLEN+10];
    void build_sa(char *s, int len) {
        int m = SIGMA + 10, n = len + 1, *x = buf[0], *y = buf[1];
        for(int i = 0; i < m; i++) c[i] = 0;
        for(int i = 0; i < n; i++) ++c[x[i] = idx(s[i])];
        for(int i = 1; i < m; i++) c[i] += c[i-1];
        for(int i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
        for(int k = 1; k <= n; k <<= 1) {
            int p = 0;
            for(int i = n-k; i < n; 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++) c[i] = 0;
            for(int i = 0; i < n; i++) ++c[x[y[i]]];
            for(int i = 1; i < m; i++) c[i] += c[i-1];
            for(int i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
            swap(x, y);
            p = 1, x[sa[0]] = 0;
            for(int i = 1; i < n; i++)
                x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i] + k] == y[sa[i-1] + k] ? p-1 : p++);
            if(p >= n) break;
            m = p;
        }
        memcpy(rk, x, sizeof(rk));
        int k = 0;
        for(int i = 0; i < n; i++) {
            if(!rk[i]) continue;
            if(k) k--;
            int j = sa[rk[i]-1];
            while(s[i+k] == s[j+k]) k++;
            height[rk[i]] = k;
        }
    }
} SA;
inline void readstr(char* str) {
    char c=getchar(); int p=0;
    while(!isalnum(c) && !ispunct(c)) c=getchar();
    while(isalnum(c) || ispunct(c)) {
        str[p++] = c;
        c = getchar();
    }
    str[p++] = '\0';
}

int len;
char str[MAXLEN+10];

int main() {
    readstr(str); len = strlen(str);
    SA.build_sa(str, len);
    for(int i = 1; i <= len; i++)
```

```
        printf("%d ", SA.sa[i]+1);
    return 0;
}
```

12 Miscellaneous

12.1 ST 表

Misc/ST.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e5;

int n, q, a[MAXN+10], st[MAXN+10][22];

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

void init_st() {
    for(int i = 1; i <= n; i++) st[i][0] = a[i];
    for(int j = 1; j <= 20; j++)
        for(int i = 1; i <= n - (1<<j) + 1; i++)
            st[i][j] = max(st[i][j-1], st[i+(1<<(j-1))][j-1]);
}

int query(int L, int R) {
    if(L > R) return 0;
    int j;
    for(j = 0; (1<<(j+1)) <= (R-L+1); j++);
    return max(st[L][j], st[R-(1<<j)+1][j]);
}

int main() {
    int l, r;
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++) a[i] = readint();
    init_st();
    while(q--) {
        l = readint(); r = readint();
        printf("%d\n", query(l, r));
    }
    return 0;
}
```

12.2 Fenwick Tree

Misc/BIT.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 5e5;

```

```cpp
int n, q, a[MAXN+10];

inline int lowbit(int x) { return x & (-x); }

void add(int p, int val) {
    while(p <= n) {
        a[p] += val;
        p += lowbit(p);
    }
}

int query(int p) {
    int ret = 0;
    while(p > 0) {
        ret += a[p];
        p -= lowbit(p);
    }
    return ret;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++)
        add(i, readint());
    while(q--) {
        int op, x, y;
        op = readint(); x = readint(); y = readint();
        switch(op) {
            case 1:
                add(x, y);
                break;
            case 2:
                printf("%d\n", query(y) - query(x-1));
                break;
        }
    }
    return 0;
}
```

12.3 左偏树

Misc/LefiestTree.cpp

```cpp
#include <bits/stdc++.h>
#define fst first
#define snd second
using namespace std;

typedef pair<int, int> pii;
```

```
const int MAXN = 1e5;

extern struct Node *null;
struct Node {
    pii val; int dist;
    Node *ch[2];
    Node() {
        ch[0] = ch[1] = null;
        dist = -1; //!!!
    };
    Node(pii v_) {
        ch[0] = ch[1] = null;
        dist = -1; val = v_;
    }
} Tnull, *null=&Tnull, *rt[MAXN+10];
int n, q, fa[MAXN+10], del[MAXN+10];

int get_fa(int x) { return x == fa[x] ? x : fa[x] = get_fa(fa[x]); }
void union_set(int x, int y) { fa[get_fa(y)] = get_fa(x); } // 顺序

Node* merge(Node* lhs, Node* rhs) {
    if(lhs == null) return rhs;
    else if(rhs == null) return lhs;
    else {
        if(lhs->val > rhs->val) swap(lhs, rhs);
        lhs->ch[1] = merge(lhs->ch[1], rhs);
        if(lhs->ch[0]->dist < lhs->ch[1]->dist)
            swap(lhs->ch[0], lhs->ch[1]);
        lhs->dist = lhs->ch[1]->dist + 1; // 距离应该是左右儿子的最小 dist + 1 （定义）
        return lhs;
    }
}

void pop(Node* &o) {
    Node *lhs = o->ch[0], *rhs = o->ch[1];
    delete o;
    o = merge(lhs, rhs);
}

void push(Node* &o, pii val) {
    o = merge(o, new Node(val));
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    int op, x, y;
    n = readint(); q = readint();
    for(int i = 1; i <= n; i++) {
        fa[i] = i;
        rt[i] = new Node(make_pair(readint(), i));
    }
    while(q--) {
        op = readint();
        switch(op) {
            case 1:
                x = readint(); y = readint();
                if(del[x] || del[y] || get_fa(x) == get_fa(y))
                    continue;
                rt[get_fa(x)] = merge(rt[get_fa(x)], rt[get_fa(y)]);
                union_set(x, y);
                break;
            case 2:
                x = readint();
                if(del[x]) puts("-1");
                else {
                    pii u = rt[get_fa(x)]->val;
                    printf("%d\n", u.fst);
                    del[u.snd] = true;
                    pop(rt[get_fa(x)]);
                }
                break;
        }
    }
    return 0;
}
```

13 莫队

13.1 普通莫队

MoQueue/HH.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 5e4, MAXQ = 2e5, MAXC = 1e6;
int N, Q, BlkSize, L, R, NowAns, A[MAXN+10], M[MAXC+10], Ans[MAXQ+10];

struct Query {
    int L, R, id;
    Query() {}
    Query(int l, int r, int i): L(l),R(r),id(i) {}
    inline bool operator<(const Query& rhs) const {
        return L/BlkSize == rhs.L/BlkSize ?
            R < rhs.R : L/BlkSize < rhs.L/BlkSize;
    }
} q[MAXQ+10];

template<typename T>
inline void readint(T& x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

inline char readc() {
    char c=getchar();
    while(!isalnum(c) && !ispunct(c))
        c=getchar();
    return c;
}

inline void readstr(char *str) {
    char c=getchar(); int p=0;
    while(!isalnum(c) && !ispunct(c)) c=getchar();
    while(isalnum(c) || ispunct(c)) {
        str[p++]=c;
        c=getchar();
    }
    str[p]='\0';
}

void Init() {
    int u, v;
    readint(N); BlkSize = ceil(sqrt(N));
    for(int i=1; i<=N; i++)
        readint(A[i]);
    readint(Q);
    for(int i=1; i<=Q; i++) {
        readint(u); readint(v);
        q[i] = Query(u, v, i);
    }
    sort(q+1, q+Q+1);
}

inline void Add(int Clr) {
    if(M[Clr]++ == 0) NowAns++;
}

inline void Sub(int Clr) {
    if(--M[Clr] == 0) NowAns--;
}

void Work() {
    L=1, R=0; NowAns=0;
    for(int i=1; i<=Q; i++) {
        while(R < q[i].R) Add(A[++R]);
        while(L > q[i].L) Add(A[--L]);
        while(R > q[i].R) Sub(A[R--]);
        while(L < q[i].L) Sub(A[L++]);
        Ans[q[i].id] = NowAns;
    }
    for(int i=1; i<=Q; i++)
        printf("%d\n", Ans[i]);
}

int main() {
    Init(); Work();
    return 0;
}
```

13.2 带修改莫队

MoQueue/Color.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e4, MAXC = 1e6;
int N, Q, Q1, Q2, L, R, T, BlkSize, NowAns, M[MAXC+10], A[MAXN+10], B[MAXN+10], Ans[MAXN+10];

struct Query {
    int L, R, T, id;
    Query() {}
    Query(int l, int r, int t, int id_): L(l), R(r), T(t), id(id_) {}
    bool operator<(const Query& rhs) const {
        if(L/BlkSize == rhs.L/BlkSize) {
            if(R/BlkSize == rhs.R/BlkSize)
                return T < rhs.T;
            else return R/BlkSize < rhs.R/BlkSize;
        } else return L/BlkSize < rhs.L/BlkSize;
    }
} q[MAXN+10];

struct Modify {
    int p, val, orig, id;
    Modify() {}
    Modify(int p_, int val_, int orig_, int id_): p(p_), val(val_), orig(orig_), id(id_) {}
```

```
} mod[MAXN+10];

template<typename T>
inline void readint(T& x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

inline char readc() {
    char c=getchar();
    while(!isalnum(c) && !ispunct(c))
        c=getchar();
    return c;
}

void Init() {
    static int u, v; char op;
    readint(N); readint(Q);
    BlkSize = ceil(pow(N, 0.67));
    for(int i=1; i<=N; i++) {
        readint(A[i]); B[i] = A[i];
    }
    for(int i=1; i<=Q; i++) {
        op = readc(); readint(u); readint(v);
        switch(op) {
            case 'Q':
                q[++Q1] = Query(u, v, Q2, i);
                break;
            case 'R':
                mod[++Q2] = Modify(u, v, B[u], i);
                B[u] = v;
                break;
        }
    }
    sort(q+1, q+Q1+1);
}

inline void add(int Clr) {
    if(M[Clr]++ == 0) NowAns++;
}

inline void sub(int Clr) {
    if(--M[Clr] == 0) NowAns--;
}

inline void goforth(int t) {
    //先把修改点纳入当前区间!
    while(L > mod[t].p) add(A[--L]);
    while(R < mod[t].p) add(A[++R]);
    A[mod[t].p] = mod[t].val;
    sub(mod[t].orig); add(mod[t].val);
}

inline void goback(int t) {
    while(L > mod[t].p) add(A[--L]);
    while(R < mod[t].p) add(A[++R]);
    A[mod[t].p] = mod[t].orig; //改回去!
    sub(mod[t].val); add(mod[t].orig);
}

void Work() {
    L=1, R=0, T=0;
    for(int i=1; i<=Q1; i++) {
        while(T < q[i].T) goforth(++T);
        while(T > q[i].T) goback(T--); //先调整时间后调整区间
        while(R < q[i].R) add(A[++R]);
        while(L > q[i].L) add(A[--L]);
        while(R > q[i].R) sub(A[R--]);
        while(L < q[i].L) sub(A[L++]);
        Ans[q[i].id] = NowAns;
    }
    for(int i=1; i<=Q; i++)
        if(Ans[i]) {
            printf("%d\n", Ans[i]);
        }
}

int main() {
    Init(); Work();
    return 0;
}
```

14 分块相关

14.1 分块

例题：教主的魔法

Block/Magic.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1e6, INF = 0x3f3f3f3f;
int N, Q, BlkSize, A[MAXN+10], B[MAXN+10], Blk[MAXN+10],
    L[MAXN+10], R[MAXN+10], Addv[MAXN+10];

template<typename T>
inline void readint(T& x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

inline char readc() {
    char c=getchar();
    while(!isalnum(c) && !ispunct(c))
        c=getchar();
    return c;
}

inline void InitBlk() {
    BlkSize = ceil(sqrt(N));
    for(int i=0; i*BlkSize + 1 <= N; i++) { //注意在分块时考虑末尾块的情况!
        L[i] = i*BlkSize + 1, R[i] = min((i+1)*BlkSize, N);
        for(int j=L[i]; j<=R[i]; j++) {
            Blk[j] = i; B[j] = A[j];
        }
        sort(B+L[i], B+R[i]+1);
    }
}

inline void Maintain(int o) {
    for(int i=L[o]; i<=R[o]; i++)
        B[i] = A[i];
    sort(B+L[o], B+R[o]+1);
}

inline void Add(int qL, int qR, int v) {
    if(Blk[qL] == Blk[qR]) {
        for(int i=qL; i<=qR; i++)
            A[i] += v;
        Maintain(Blk[qL]);
    } else {
        for(int i=qL; Blk[i] == Blk[qL]; i++)
            A[i] += v;
        for(int i=qR; Blk[i] == Blk[qR]; i--)
            A[i] += v;
        Maintain(Blk[qL]); Maintain(Blk[qR]);
        for(int i=Blk[qL]+1; i<=Blk[qR]-1; i++)
            Addv[i] += v;
    }
}

inline int Query(int qL, int qR, int v) { //>=v
    int ret = 0, p;
    if(Blk[qL] == Blk[qR]) {
        for(int i=qL; i<=qR; i++)
            if(A[i] + Addv[Blk[i]] >= v) ret++;
        return ret;
    } else {
        for(int i=qL; Blk[i] == Blk[qL]; i++)
            if(A[i] + Addv[Blk[i]] >= v) ret++;
        for(int i=qR; Blk[i] == Blk[qR]; i--)
            if(A[i] + Addv[Blk[i]] >= v) ret++;
        for(int i=Blk[qL]+1; i<=Blk[qR]-1; i++) {
            p = lower_bound(B+L[i], B+R[i]+1, v-Addv[i]) - (B+L[i]);
            ret += R[i] - L[i] + 1 - p;
        }
        return ret;
    }
}

void Init() {
    readint(N); readint(Q);
    for(int i=1; i<=N; i++) readint(A[i]);
    InitBlk();
}

void Work() {
    int l, r, v; char op;
    while(Q--) {
        op = readc(); readint(l); readint(r); readint(v);
        switch(op) {
        case 'M':
            Add(l, r, v);
            break;
        case 'A':
            printf("%d\n", Query(l, r, v));
            break;
        }
    }
}

int main() {
    Init(); Work();
    return 0;
}
```

14.2 区间众数

Block/Mode.cpp

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

const int MAXN = 1e5, SQN = 316;
int N, Q, BlkSize, MxBlk, A[MAXN+10], Blk[MAXN+10], L[MAXN+10], R[MAXN+10];
int Num[MAXN+10], Mode[SQN+10][SQN+10], PreCnt[SQN+10][MAXN+10];

inline void InitBlk() {
    static int Md, Bkt[MAXN+10];
    BlkSize = ceil(sqrt(N)) + 0.5;
    for(int i=0; i * BlkSize + 1 <= N; i++) {
        MxBlk = i;
        L[i] = i * BlkSize + 1, R[i] = min(N, (i+1) * BlkSize);
        for(int j = L[i]; j <= R[i]; j++) Blk[j] = i;
    }
    for(int i = 1; i <= N; i++)
        PreCnt[Blk[i]][A[i]]++;
    for(int i = 1; i <= MxBlk; i++)
        for(int j = 1; j <= N; j++)
            PreCnt[i][j] += PreCnt[i-1][j];
    for(int i = 0; i <= MxBlk; i++) {
        for(int j = i; j <= MxBlk; j++) {
            if(i < j) Md = Mode[i][j-1];
            else Md = 0;
            for(int k = L[j]; k <= R[j]; k++) {
                Bkt[A[k]]++;                                    // !!!
                int lhs = Bkt[A[k]] + (i < j ? PreCnt[j-1][A[k]] - (i>=1 ? PreCnt[i-1][A[k]] : 0) : 0);
                int rhs = Bkt[Md] + (i < j ? PreCnt[j-1][Md] - (i>=1 ? PreCnt[i-1][Md] : 0) : 0);
                if(lhs > rhs || (lhs == rhs && A[k] < Md))
                    Md = A[k];
            }
            Mode[i][j] = Md;
            for(int k = L[j]; k <= R[j]; k++) Bkt[A[k]]--;
        }
    }
}

int Query(int qL, int qR) {
    static int Md, Bkt[MAXN+10];
    if(Blk[qL] == Blk[qR]) {
        Md = 0;
        for(int i = qL; i <= qR; i++) {
            Bkt[A[i]]++;
            if(Bkt[A[i]] > Bkt[Md] || (Bkt[A[i]] == Bkt[Md] && A[i] < Md))
                Md = A[i];
        }
        for(int i = qL; i <= qR; i++) Bkt[A[i]]--;
        return Md;
    } else {
        if(Blk[qL] + 1 <= Blk[qR] - 1)
            Md = Mode[Blk[qL]+1][Blk[qR]-1];
        else Md = 0;
        for(int i = qL; Blk[i] == Blk[qL]; i++) {
            ++Bkt[A[i]];
            int lhs = Bkt[A[i]] + (Blk[qL]+1 <= Blk[qR]-1 ?
                PreCnt[Blk[qR]-1][A[i]] - PreCnt[Blk[qL]][A[i]] : 0);
            int rhs = Bkt[Md] + (Blk[qL]+1 <= Blk[qR]-1 ?
                PreCnt[Blk[qR]-1][Md] - PreCnt[Blk[qL]][Md] : 0);
            if(lhs > rhs || (lhs == rhs && A[i] < Md))
                Md = A[i];
        }
        for(int i = qR; Blk[i] == Blk[qR]; i--) {
            ++Bkt[A[i]];
            int lhs = Bkt[A[i]] + (Blk[qL]+1 <= Blk[qR]-1 ?
                PreCnt[Blk[qR]-1][A[i]] - PreCnt[Blk[qL]][A[i]] : 0);
            int rhs = Bkt[Md] + (Blk[qL]+1 <= Blk[qR]-1 ?
                PreCnt[Blk[qR]-1][Md] - PreCnt[Blk[qL]][Md] : 0);
            if(lhs > rhs || (lhs == rhs && A[i] < Md))
                Md = A[i];
        }
        for(int i = qL; Blk[i] == Blk[qL]; i++) --Bkt[A[i]];
        for(int i = qR; Blk[i] == Blk[qR]; i--) --Bkt[A[i]];
        return Md;
    }
}

template<typename T>
inline void readint(T& x) {
    T f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

void Init() {
    readint(N); readint(Q);
    for(int i=1; i<=N; i++) {
        readint(A[i]);
        Num[++Num[0]] = A[i];
    }
    sort(Num+1, Num+Num[0]+1);
    Num[0] = unique(Num+1, Num+Num[0]+1) - Num - 1;
    for(int i=1; i<=N; i++)
        A[i] = lower_bound(Num+1, Num+Num[0]+1, A[i]) - Num;
    InitBlk();
}

void Work() {
    int l, r, LastAns = 0;
    for(int i=1; i<=Q; i++) {
        readint(l); readint(r);
        l = (l + LastAns - 1) % N + 1;
        r = (r + LastAns - 1) % N + 1;
        if(l > r) swap(l, r);
        printf("%d\n", LastAns = Num[Query(l, r)]);
    }
}

int main() {
    Init(); Work();
    return 0;
}
```

15 多项式

（为何比别人多了 4 倍常数……）

15.1 快速傅里叶变换

Poly/FFT.cpp

```cpp
#include <bits/stdc++.h>

const int MAXN = 4e6;
const double PI = 3.141592653589793238462643383;

struct cpx {
    double real, imag;
    cpx() { real = imag = .0; }
    cpx(double x) { real = x, imag = .0; }
    cpx(double x, double y) { real = x, imag = y; }
    friend cpx operator+(const cpx &lhs, const cpx &rhs) {
        return cpx(lhs.real + rhs.real, lhs.imag + rhs.imag);
    }
    friend cpx operator-(const cpx &lhs, const cpx &rhs) {
        return cpx(lhs.real - rhs.real, lhs.imag - rhs.imag);
    }
    friend cpx operator*(const cpx &lhs, const cpx &rhs) {
        return cpx(lhs.real * rhs.real - lhs.imag * rhs.imag,
            lhs.imag * rhs.real + lhs.real * rhs.imag);
    }
    cpx operator*=(const cpx &rhs) { return (*this) = (*this) * rhs; }
    cpx conj() const { return cpx(real, -imag); }
    friend cpx operator/(const cpx &lhs, double rhs) {
        return cpx(lhs.real / rhs, lhs.imag / rhs);
    }
    friend cpx operator/(const cpx &lhs, const cpx &rhs) {
        cpx ret = lhs * rhs.conj();
        ret.real /= (rhs.real * rhs.real + rhs.imag * rhs.imag);
        ret.imag /= (rhs.real * rhs.real + rhs.imag * rhs.imag);
        return ret;
    }
    cpx operator/=(const cpx &rhs) { return (*this) = (*this) / rhs; }
};

int n, m, R[MAXN+10]; double A[MAXN+10], B[MAXN+10], C[MAXN+10];

inline cpx get_rt(int step, bool inv) { // rotation factor
    return inv ? cpx(std::cos(2*PI / step), -std::sin(2*PI / step)) :
                 cpx(std::cos(2*PI / step), std::sin(2*PI / step));
}

void fft(cpx A[], int len, bool inv) {
    for(int i = 0; i < len; i++)
        if(R[i] > i) std::swap(A[i], A[R[i]]);
    for(int step = 1; step < len; step <<= 1) {
        for(int i = 0; i < len; i += (step<<1)) {
            cpx omega = 1, rt = get_rt(step<<1, inv);
            for(int j = 0; j < step; j++, omega *= rt) {
                cpx t = omega * A[i+j+step];
                A[i+j+step] = A[i+j] - t;
                A[i+j] = A[i+j] + t;
            }
        }
    }
    if(inv)
        for(int i = 0; i < len; i++) A[i] /= len;
}

void conv(double dA[], double dB[], int deg1, int deg2, double dC[]) { // deg: 输入多项式的度数
    int len;
    static cpx A[MAXN+10], B[MAXN+10], C[MAXN+10];
    for(len = 1; len < deg1+deg2+1; len <<= 1); // 考虑乘完后的长度
    for(int i = 0; i < len; i++) {
        A[i] = dA[i], B[i] = dB[i];
    }

    R[0] = 0;
    for(int i = 1; i < len; i++)
        R[i] = ((R[i>>1]>>1) | (len >> (i&1))) & (len-1);

    fft(A, len, false); fft(B, len, false);
    for(int i = 0; i < len; i++) C[i] = A[i] * B[i];
    fft(C, len, true);
    for(int i = 0; i < len; i++) dC[i] = C[i].real;
}

inline int readint() {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    return f*r;
}

int main() {
    n = readint(); m = readint();
    for(int i = 0; i <= n; i++) A[i] = readint();
    for(int i = 0; i <= m; i++) B[i] = readint();
    conv(A, B, n, m, C);
    for(int i = 0; i <= n+m; i++) std::printf("%d ", int(round(C[i])));
    return 0;
}
```

15.2 快速数论变换

998244353 的原根是 3

Poly/NTT.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long int64;
const int MAXN = 4e6, MOD = 998244353, G = 3;

```

```cpp
int n, m, A[MAXN+10], B[MAXN+10], C[MAXN+10], R[MAXN+10];

int64 fastpow(int64 a, int64 x) {
    int64 ret = 1; a %= MOD;
    while(x) {
        if(x & 1) ret = ret * a % MOD;
        x >>= 1; a = a * a % MOD;
    }
    return ret;
}

int get_rt(int step, bool inv) {
    return !inv ? fastpow(G, (MOD-1) / step) : fastpow(G, (MOD-1) / step * (step-1));
}

void ntt(int A[], int len, bool inv) {
    for(int i = 0; i < len; i++)
        if(R[i] > i) swap(A[R[i]], A[i]);
    for(int step = 1; step < len; step <<= 1) {
        for(int i = 0; i < len; i += (step << 1)) {
            int64 omega = 1, rt = get_rt(step << 1, inv);
            for(int j = 0; j < step; j++, omega = (omega * rt) % MOD) {
                int t = omega * A[i+j+step] % MOD;
                A[i+j+step] = ((A[i+j] - t) % MOD + MOD) % MOD;
                A[i+j] = (A[i+j] + t) % MOD;
            }
        }
    }
    if(inv) {
        int64 inv_ele = fastpow(len, MOD-2);
        for(int i = 0; i < len; i++) A[i] = (A[i] * inv_ele) % MOD;
    }
}

void conv(int A[], int B[], int deg1, int deg2, int C[]) {
    int len; for(len = 1; len < deg1+deg2+1; len <<= 1);
    R[0] = 0;
    for(int i = 1; i < len; i++)
        R[i] = ((R[i>>1]>>1) | (len >> (i & 1))) & (len-1);
    ntt(A, len, false); ntt(B, len, false);
    for(int i = 0; i < len; i++) C[i] = int64(A[i]) * B[i] % MOD;
    ntt(A, len, true); ntt(B, len, true); ntt(C, len, true);
}

template<typename T>
inline void readint(T &x) {
    int f=1, r=0; char c=getchar();
    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
    x = f*r;
}

template<typename T>
inline void writeint(T &x) {
    static char buf[32];
    char *ptr = buf;
    if(x < 0) putchar('-'), x = -x;
    do {
        *ptr++ = (x % 10) + '0';
        x /= 10;
    } while(x);
    do
        putchar(*--ptr);
    while(ptr != buf);
}

int main() {
    readint(n); readint(m);
    for(int i = 0; i <= n; i++) readint(A[i]);
    for(int i = 0; i <= m; i++) readint(B[i]);
    conv(A, B, n, m, C);
    for(int i = 0; i <= n+m; i++) writeint(C[i]), putchar(' ');
    return 0;
}
```

16 数论

16.1 线性求逆元

推导 令 $p = ki + r (0 \le r < i)$

则 $0 \equiv ki + r \pmod p$

$\Rightarrow ki \cdot i^{-1} r^{-1} + r \cdot i^{-1} r^{-1} \equiv 0$

$\Rightarrow i^{-1} \equiv -k \cdot r^{-1} \equiv p - \left\lfloor \frac{p}{i} \right\rfloor + p \bmod i \pmod p$

Math/inv.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 3e6;
int n, p, inv[MAXN+10];
int main() {
    cin >> n >> p;
    inv[1] = 1; printf("%d\n", inv[1]);
    for(int i = 2; i <= n; i++)
        printf("%d\n", inv[i] = (p - (long long)(p / i) * inv[p % i] % p) % p);
    return 0;
}
```

16.2 线性筛

16.2.1 求 $\varphi(n)$

Math/phi.cpp

```cpp
int64 GetPhi(int64 x) { // 单个数的 Phi
    int64 ret = x;
    for(int i = 2; i <= x; i++) if(x % i == 0) {
        ret = ret / i * (i-1);
        while(x % i == 0) x /= i;
    }
    if(x > 1)
        ret = ret / x * (x-1);
    return ret;
}

int PrimeCnt, Phi[MAXNUM+10], PrimeLst[MAXNUM+10], NotPrime[MAXNUM+10];
void PhiSieve() { // 线性筛 Phi
    NotPrime[1] = true;
    Phi[1] = 1;
    for(int i = 2; i <= MAXNUM; i++) {
        if(!NotPrime[i]) {
            PrimeLst[++PrimeCnt] = i;
            Phi[i] = i-1;
        }
        for(int j = 1; j <= PrimeCnt; j++) {
            if(i * PrimeLst[j] > MAXNUM) break;
            NotPrime[i * PrimeLst[j]] = true;
            if(i % PrimeLst[j] == 0) {
                Phi[i * PrimeLst[j]] = Phi[i]  * PrimeLst[j];
                break;
            } else
                Phi[i * PrimeLst[j]] = Phi[i] * Phi[PrimeLst[j]];
        }
    }
}
```

16.2.2 求 $\mu(n)$

Math/mu.cpp

```cpp
int PrimeCnt, PrimeLst[MAXNUM+10], NotPrime[MAXNUM+10], Mu[MAXNUM+10];

void MuSieve() {
    NotPrime[1] = true;
    Mu[1] = 1;
    for(int i = 2; i <= MAXNUM; i++) {
        if(!NotPrime[i]) {
            PrimeLst[++PrimeCnt] = i;
            Mu[i] = -1;
        }
        for(int j = 1; j <= PrimeCnt; j++) {
            if(i * PrimeLst[j] > MAXNUM) break;
            NotPrime[i * PrimeLst[j]] = true;
            if(i % PrimeLst[j] == 0) {
                Mu[i * PrimeLst[j]] = 0;
                break;
            } else
                Mu[i * PrimeLst[j]] = -Mu[i];
        }
    }
}
```

16.2.3 求 $d(n)$

16.3 扩展欧几里得定理

Math/exgcd.cpp

```cpp
void exgcd(int a, int b, int &d, int &x, int &y) {
    if(b == 0) { d = a, x = 1, y = 0; }
    else {
        exgcd(b, a % b, d, y, x); y -= x * (a / b);
    }
}
```

16.4 中国剩余定理

https://blog.csdn.net/ruoruo_cheng/article/details/52075213

Math/crt.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long int64;
const int MAXN = 1e5;
int n; int64 a[MAXN+10], m[MAXN+10];

void exgcd(int64 a, int64 b, int64 &d, int64 &x, int64 &y) {
    if(b == 0) { d = a, x = 1, y = 0; }
    else {
        exgcd(b, a % b, d, y, x); y -= x * (a / b);
    }
}

int64 china(int n, int64 a[], int64 m[]) {
    int64 M = 1, ret = 0, Mi, Minv, d, y;
    for(int i = 1; i <= n; i++)
        M *= m[i];
    for(int i = 1; i <= n; i++) {
        Mi = M / m[i];
        exgcd(Mi, m[i], d, Minv, y);
        assert(d == 1);
        ret = (ret + Mi * Minv % M * a[i] % M) % M;
    }
    ret = (ret + M) % M;
    return ret;
}

int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
        scanf("%lld%lld", &a[i], &m[i]);
    printf("%lld\n", china(n, a, m));
    return 0;
}
```

16.5 扩展欧拉定理

$$a^b \equiv a^{b \bmod \varphi(n)+[b\geq\varphi(n)]\cdot\varphi(n)} \pmod n$$

Math/exteuler.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXP = 1e7;

int T, p, phi[MAXP+10];

void InitPhi() {
    phi[1] = 1; //do not set phi[2], for it is a prime
    for(int i=2; i<=MAXP; i++)
        if(!phi[i]) {
            for(int j=i; j<=MAXP; j+=i) {
                if(!phi[j]) phi[j] = j;
                phi[j] = phi[j] / i * (i-1);
            }
        }
}

inline int fastmul(int a, int x, int mod) {
    int ret = 0;
    while(x) {
        if(x&1) ret = ((ret%mod) + (a%mod))%mod;
        x>>=1; a = ((a%mod) + (a%mod)) %mod;
    }
    return ret;
}

inline int fastpow(int a, int x, int mod) {
    int ret = 1;
    while(x) {
        if(x&1) ret = fastmul(ret, a, mod) % mod;
        x>>=1; a = fastmul(a, a, mod) % mod;
    }
    return ret;
}

int solve(int mod) {
    if(mod == 1) return 0;
    return fastpow(2, solve(phi[mod])+phi[mod], mod);
}

int main() {
    InitPhi();
    scanf("%d", &T);
    while(T--) {
        scanf("%d", &p);
        printf("%d\n", solve(p));
    }
}
```

16.6 Lucas 定理

Math/lucas.cpp

```cpp
#include <bits/stdc++.h>
using std::cin;
using std::cout;

typedef long long int64;
const int MAXN = 1e5;
int T, n, m, p, fact[MAXN+10], factinv[MAXN+10];

int C(int n, int m) {
    if(m < 0 || m > n) return 0;
    return ((int64)fact[n] * factinv[m]) % p * factinv[n-m] % p;
}

int lucas(int n, int m) {
    if(n < p && m < p) return C(n, m);
    return C(n % p, m % p) * (int64)lucas(n / p, m / p) % p;
}

inline int64 fastpow(int64 a, int64 x) {
    int64 ret = 1;
    while(x) {
        if(x & 1) ret = ret * a % p;
        x >>= 1; a = a * a % p;
    }
    return ret;
}

void init_inv() {
    fact[0] = factinv[0] = 1;
    for(int i = 1; i <= p-1; i++)
        fact[i] = (int64)fact[i-1] * i % p;
    factinv[p-1] = fastpow(fact[p-1], p-2);
    for(int i = p-2; i >= 1; i--)
        factinv[i] = factinv[i+1] * (int64)(i+1) % p;
}

int main() {
    cin >> T;
    while(T--) {
        cin >> n >> m >> p;
        init_inv();
        cout << lucas(n+m, m) << '\n';
    }
    return 0;
}
```