网络流

```cpp
//https://www.luogu.com.cn/problem/P2756
#include <bits/stdc++.h>

using namespace std;
template <int N>
struct Dinic {
    const int INF = INT_MAX;
    struct E {
        int to, v, rev;
    };
    vector<E> G[N];
    int d[N], cur[N];
    void add(int x, int y, int c)
    {
        G[x].push_back({ y, c, (int)G[y].size() });
        G[y].push_back({ x, 0, (int)G[x].size() - 1 });
    }
    bool bfs(int S, int T)
    {
        queue<int> q;
        memset(d, -1, sizeof d);
        d[S] = 0;
        cur[S] = 0;
        q.push(S);
        while (q.size()) {
            int tmp = q.front();
            q.pop();
            for (int k = 0; k < G[tmp].size(); k++) {
                int ver = G[tmp][k].to;
                int dis = G[tmp][k].v;
                if (d[ver] == -1 && dis) {
                    d[ver] = d[tmp] + 1;
                    cur[ver] = 0;
                    if (ver == T)
                        return true;
                    q.push(ver);
                }
            }
        }
        return false;
    }

    int dfs(int u, int T, int limit)
    {
        if (u == T)
            return limit;
        int flow = 0;
        for (int k = cur[u]; k < G[u].size() && flow < limit; k++) {
            cur[u] = k; // 当前弧优化
            E& P = G[u][k];
            int ver = P.to, dis = P.v;
            if (d[ver] == d[u] + 1 && dis) {
```

```
                int flw = dfs(ver, T, min(dis, limit - flow));
                if (!flw)
                    d[ver] = -2; // 炸点优化
                P.v -= flw, G[P.to][P.rev].v += flw, flow += flw;
            }
        }
        return flow;
    }

    long long maxflow(int S, int T)
    {
        long long r = 0, flow;
        while (bfs(S, T)) {
            while (flow = dfs(S, T, INF))
                r += flow;
        }
        return r;
    }
    void path(int S, int T, int m)
    {
        for (int i = 1; i <= m; i++) {
            for (int j = 0; j < G[i].size(); j++) {
                if (G[i][j].v == 0 && G[i][j].to != S && G[i][j].to != T) {
                    cout << i << " " << G[i][j].to << endl;
                }
            }
        }
    }
};
Dinic<1005> dinic;
void solve()
{
    int n, m, a, b;
    cin >> m >> n;
    for (int i = 1; i <= m; i++)
        dinic.add(0, i, 1);
    for (int i = m + 1; i <= n; i++)
        dinic.add(i, 1000, 1);
    while (cin >> a >> b, ~a && ~b) {
        dinic.add(a, b, 1);
    }
    cout << dinic.maxflow(0, 1000) << endl;
    dinic.path(0, 1000, m);
}

signed main()
{
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    solve();
    return 0;
}
```

主席树

```cpp
//https://www.luogu.com.cn/problem/P3834
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=2e5+10;
struct k{
    int l,r,sum;
}node[N*32];
int cnt,root[N];
int a[N];
vector<int>v;
// inline int get(int x){
//     return lower_bound(v.begin(),v.end(),x)-v.begin()+1;
// }
void insert(int l,int r,int pre,int&now,int t){
    node[++cnt]=node[pre];
    now=cnt;
    node[now].sum++;
    if(l>=r)return;
    int mid=(l+r)>>1;
    if(t<=mid)insert(l,mid,node[pre].l,node[now].l,t);
    else insert(mid+1,r,node[pre].r,node[now].r,t);
}
int query(int l,int r,int b,int e,int k){
    if(l==r)return l;
    int mid=(l+r)>>1;
    int temp=node[node[e].l].sum-node[node[b].l].sum;
    if(k<=temp)return query(l,mid,node[b].l,node[e].l,k);
    else return query(mid+1,r,node[b].r,node[e].r,k-temp);
}
signed main(){
    int n,m;cin>>n>>m;
    // for(int i=1;i<=n;i++){
        // cin>>a[i];v.push_back(a[i]);
    // }
    // sort(v.begin(),v.end());//离散化
    // v.erase(unique(v.begin(),v.end()),v.end());
    // for(int i=1;i<=n;i++){
    //     insert(1,n,root[i-1],root[i],get(a[i]));
    // }
    // for(int i=0;i<m;i++){
    //     int l,r,k;cin>>l>>r>>k;
    //     cout<<v[query(1,n,root[l-1],root[r],k)-1]<<endl;
    // }
    for(int i=0;i<n;i++)cin>>a[i];
    for(int i=0;i<n;i++){
        insert(0,1e9+10,root[i],root[i+1],a[i]);
    }
    for(int i=0;i<m;i++){
        int l,r,k;cin>>l>>r>>k;
        cout<<query(0,1e9+10,root[l-1],root[r],k)<<endl;
    }

}
```

珂朵莉树

```cpp
//https://www.luogu.com.cn/problem/CF896C
#include<bits/stdc++.h>
/*
随机生成的一组数据，进行以下操作

1.区间l到r加上x

2.区间l到r赋值成x

3.输出区间l到r中第x小的数（这能过估计是数据随机的原因）

4.输出l到r中每个数字的x次方和模y的值
*/
using namespace std;

typedef long long ll;
const ll MOD = 1000000007;
const ll MAXN = 100005;

struct Node {
    ll l, r;//l和r表示这一段的起点和终点
    mutable ll v;//v表示这一段上所有元素相同的值是多少

    Node(ll l, ll r = 0, ll v = 0) : l(l), r(r), v(v) {}

    bool operator<(const Node &a) const {
        return l < a.l;//规定按照每段的左端点排序
    }
};

ll n, m, seed, vmax, a[MAXN];
set<Node> s;

//以pos去做切割，找到一个包含pos的区间，把它分成[l,pos-1],[pos,r]两半
set<Node>::iterator split(int pos) {
    set<Node>::iterator it = s.lower_bound(Node(pos));
    if (it != s.end() && it->l == pos) {
        return it;
    }
    it--;
    if (it->r < pos) return s.end();
    ll l = it->l;
    ll r = it->r;
    ll v = it->v;
    s.erase(it);
    s.insert(Node(l, pos - 1, v));
    //insert函数返回pair，其中的first是新插入结点的迭代器
    return s.insert(Node(pos, r, v)).first;
}

/*
 * 这里注意必须先计算itr。
 * 比如现在区间是[1,4]，如果要add的是[1,2],如果先split(1)
 * 那么返回的itl是[1,4]，但是下一步计算itr的时候会把这个区间删掉拆成[1,2]和[3,4]
 * 那么itl这个指针就被释放了
 * */
void add(ll l, ll r, ll x) {
```

```
    set<Node>::iterator itr = split(r + 1), itl = split(l);
    for (set<Node>::iterator it = itl; it != itr; ++it) {
        it->v += x;
    }
}

void assign(ll l, ll r, ll x) {
    set<Node>::iterator itr = split(r + 1), itl = split(l);
    s.erase(itl, itr);
    s.insert(Node(l, r, x));
}

struct Rank {
    ll num, cnt;

    bool operator<(const Rank &a) const {
        return num < a.num;
    }

    Rank(ll num, ll cnt) : num(num), cnt(cnt) {}
};

ll rnk(ll l, ll r, ll x) {
    set<Node>::iterator itr = split(r + 1), itl = split(l);
    vector<Rank> v;
    for (set<Node>::iterator i = itl; i != itr; ++i) {
        v.push_back(Rank(i->v, i->r - i->l + 1));
    }
    sort(v.begin(), v.end());
    int i;
    for (i = 0; i < v.size(); ++i) {
        if (v[i].cnt < x) {
            x -= v[i].cnt;
        } else {
            break;
        }
    }
    return v[i].num;
}

ll ksm(ll x, ll y, ll p) {
    ll r = 1;
    ll base = x % p;
    while (y) {
        if (y & 1) {
            r = r * base % p;
        }
        base = base * base % p;
        y >>= 1;
    }
    return r;
}

ll calP(ll l, ll r, ll x, ll y) {
    set<Node>::iterator itr = split(r + 1), itl = split(l);
    ll ans = 0;
    for (set<Node>::iterator i = itl; i != itr; ++i) {
        ans = (ans + ksm(i->v, x, y) * (i->r - i->l + 1) % y) % y;
```

```
    }
    return ans;
}

ll rnd() {
    ll ret = seed;
    seed = (seed * 7 + 13) % MOD;
    return ret;
}

int main() {
    cin >> n >> m >> seed >> vmax;
    for (int i = 1; i <= n; ++i) {
        a[i] = (rnd() % vmax) + 1;
        s.insert(Node(i, i, a[i]));
    }
    for (int i = 1; i <= m; ++i) {
        ll op, l, r, x, y;
        op = (rnd() % 4) + 1;
        l = (rnd() % n) + 1;
        r = (rnd() % n) + 1;
        if (l > r) swap(l, r);
        if (op == 3)
            x = (rnd() % (r - l + 1)) + 1;
        else
            x = (rnd() % vmax) + 1;
        if (op == 4)
            y = (rnd() % vmax) + 1;
        if (op == 1)
            add(l, r, x);
        else if (op == 2)
            assign(l, r, x);
        else if (op == 3)
            cout << rnk(l, r, x) << endl;
        else
            cout << calP(l, r, x, y) << endl;
    }
    return 0;
}
```