目录

头文件

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define rep(i,a,n) for(int i=a;i<n;i++)
#define per(i,a,n) for(int i=n-1;i>=a;i--)
#define fastio ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define multi int _;cin>>_;while(_--)
#define debug(x) cerr << #x << " = " << (x) << endl;
```

```
#define int long long
ll gcd(ll a,ll b){ return b?gcd(b,a%b):a;}
mt19937 mrand(random_device{}());
int rnd(int x){ return mrand() % x; }
void test() {cerr << "\n";}
template<typename T, typename... Args>
void test(T x, Args... args) {cerr << x << " ";test(args...);}
const ll MOD = 998244353;
//const ll MOD = 1e9+7;
const ll P1 = 9999971, base1 = 101;
const ll P2 = 9999973, base2 = 103;
const ll N = 200005;
//head
```

对拍

```
:loop
 data.exe > 1.in
 my.exe <1.in >my.out
 std.exe <1.in >std.out
 fc my.out std.out
if not errorlevel 1 goto loop
pause
goto loop
```

__int128输入输出

```
__int128 read()
{
    __int128 f=1,w=0;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-')
        f=-1;
        ch=getchar();
    }
    while(ch<='9'&&ch>='0')
    {
        w=w*10+ch-'0';
        ch=getchar();
    }
    return f*w;
}

void print(__int128 x)
{
    if(x<0)
    {
        putchar('-');
        x=-x;
    }
    if(x>9)print(x/10);
```

```
    putchar(x%10+'0');
}
```

快速读入

```
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while (ch<'0'||ch>'9'){if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
    return x*f;
}
```

kmp

```
int nxt[N];
char s[N];
void solve()
{
    cin >> s + 1;
    int n = strlen(s + 1);
    nxt[1] = 0;
    int j = 0;
    for(int i = 2 ; i <= n ; i++ )
    {
        while(j > 0 && s[j + 1] != s[i])
        {
            j = nxt[j];
        }
        if(s[j + 1] == s[i])
        {
            j++;
        }
        nxt[i] = j;
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        cout << nxt[i] << " \n"[i == n];
    }
}
```

manacher

```
int n, p[2 * N + 2];
char s[N + 2], t[2 * N + 3];

void manacher()
{
    n = strlen(s + 1);
    int m = 0;
    t[++m] = '#';
```

```
    for(int i = 1 ; i <= n ; i++ )
    {
        t[++m] = s[i], t[++m] = '#';
    }
    int M = 0, R = 0;
    for(int i = 1 ; i <= m ; i++ )
    {
        if(i > R)
        {
            p[i] = 1;
        }else
        {
            p[i] = min(p[2 * M - i], R - i + 1);
        }
        while(i - p[i] > 0 && i + p[i] <= m && t[i - p[i]] == t[i + p[i]])
        {
            ++p[i];
        }
        if(i + p[i] - 1 > R)
        {
            M = i, R = i + p[i] - 1;
        }
    }
    int ans = 0;
    for(int i = 1 ; i <= m ; i++ )
    {
        ans = max(ans, p[i]);
    }
    printf("%lld\n", ans - 1);
}
```

组合数

```
ll fact[N] = {1}, inv[N] = {1};
ll C(ll x, ll y)
{
    return(((fact[x] * inv[y])% MOD * inv[x-y]) % MOD);
}
ll ksm(ll x, ll y)
{
    ll ans = 1;
    x %= MOD;
    while(y)
    {
        if(y&1)
        {
            ans = ans * x % MOD;
        }
        x = x * x % MOD;
        y /= 2;
    }
    return ans;
}
```

```
void build()
{
    for(int i = 1 ; i < N ; i++ )
    {
        fact[i] = fact[i-1] * i % MOD;
    }
    for(int i = 1 ; i < N ; i++ )
    {
        inv[i] = inv[i-1] * ksm(i, MOD-2) % MOD;
    }
}
```

快速幂

```
ll PowerMod(ll a, ll b, ll c)
{
    ll ans = 1;
    a = a % c;
    while(b>0) {
        if(b % 2 == 1)
        ans = (ans * a) % c;
        b = b/2;
        a = (a * a) % c;
    }
    return ans;
}
```

整数分块

```
for(ll l = 1 ; l <= n ; l++ )
    {
        ll d = n / l, r = n / d;
        cout << l << " : " << r << " = " << d << endl;
        l = r;
    }
```

最大流（Dinic)

```
struct MF {
  struct edge {
    int v, nxt, cap, flow;
  } e[N];

  int fir[N], cnt = 0;

  int n, S, T;
  ll maxflow = 0;
  int dep[N], cur[N];

  void init() {
    memset(fir, -1, sizeof fir);
```

```
    cnt = 0;
  }

  void addedge(int u, int v, int w) {
    e[cnt] = {v, fir[u], w, 0};
    fir[u] = cnt++;
    e[cnt] = {u, fir[v], 0, 0};
    fir[v] = cnt++;
  }

  bool bfs() {
    queue<int> q;
    memset(dep, 0, sizeof(int) * (n + 1));

    dep[S] = 1;
    q.push(S);
    while (q.size()) {
      int u = q.front();
      q.pop();
      for (int i = fir[u]; ~i; i = e[i].nxt) {
        int v = e[i].v;
        if ((!dep[v]) && (e[i].cap > e[i].flow)) {
          dep[v] = dep[u] + 1;
          q.push(v);
        }
      }
    }
    return dep[T];
  }

  int dfs(int u, int flow) {
    if ((u == T) || (!flow)) return flow;

    int ret = 0;
    for (int& i = cur[u]; ~i; i = e[i].nxt) {
      int v = e[i].v, d;
      if ((dep[v] == dep[u] + 1) &&
          (d = dfs(v, min(flow - ret, e[i].cap - e[i].flow)))) {
        ret += d;
        e[i].flow += d;
        e[i ^ 1].flow -= d;
        if (ret == flow) return ret;
      }
    }
    return ret;
  }

  void dinic() {
    while (bfs()) {
      memcpy(cur, fir, sizeof(int) * (n + 1));
      maxflow += dfs(S, INF);
    }
  }
} mf;
```

欧拉筛（质数）

```
const ll MAXN = 1e6 + 5;
ll prime[MAXN], idxprime = 0;
bool isprime[MAXN];

void prime_build()
{
    for(int i = 2 ; i < MAXN ; i++ )
    {
        if(isprime[i] == 0)
        {
            prime[++idxprime] = i;
        }
        for(int j = 1 ; j <= idxprime && i * prime[j] < MAXN ; j++ )
        {
            isprime[i * prime[j]] = 1;
            if(i % prime[j] == 0) break;
        }
    }
}
```

欧拉筛(约数个数)

```
ll prim[50000005], sum[50000005], d[50000005], len;
bool vis[50000005];

inline void sieve(int x) {
    for(int i = 2;i <= x;i ++) {
        if(! vis[i]) {
            prim[++ len] = i;
            d[i] = 2;
            sum[i] = 1;
        }
        for(int j = 1;j <= len && i * prim[j] <= x;j ++) {
            vis[i * prim[j]] = 1;
            if(i % prim[j] == 0) {
                sum[i * prim[j]] = sum[i] + 1;
                d[i * prim[j]] = d[i] / (sum[i] + 1) * (sum[i] + 2);
                break;
            }
            sum[i * prim[j]] = 1;
            d[i * prim[j]] = d[i] * 2;
        }
    }
}
```

欧拉筛（最小素因子）

```
int MAXN = 50;
```

```
int p[N], pr[N], idx;

void build()
{
    for(int i = 2 ; i < MAXN ; i++ )
    {
        if(!p[i])
        {
            p[i] = i;
            pr[++idx] = i;
        }
        for(int j = 1 ; j <= idx && pr[j] * i < MAXN ; j++ )
        {
            p[i * pr[j]] = pr[j];
            if(p[i] == pr[j]) break;
        }
    }
}
```

ST表

```
for(int i = 1 ; i <= n ; i++ )
{
    a[i] = read();
    f[0][i] = a[i];
}
for(int i = 1 ; i <= 22 ; i++ )
{
    for(int j = 1 ; j + (1 << i) - 1 <= n ; j++ )
    {
        f[i][j] = max(f[i-1][j], f[i-1][j + (1 << i - 1)]);
    }
}
for(int i = 1 ; i <= m ; i++ )
{
    int l = read(), r = read();
    int len = __lg(r - l + 1);
    printf("%d\n", max(f[len][l], f[len][r - (1 << len) + 1]));
}
```

二分图最大匹配

```
int a[N];
int v[N], n1, n2;
int to[N], b[N];
int n;
vector<int> e[N];
//n1为左边点数量，n2为右边点数量，v为右边的点连向左边哪条边
bool find(int x)
{
    b[x] = true;
    for(auto y : e[x])
    {
```

```
        if(!v[y] || (!b[v[y]] && find(v[y])))
        {
            v[y] = x;
            return true;
        }
    }
    return false;
}

int match()
{
    int ans = 0;
    memset(v, 0 ,sizeof(v));
    for(int i = 1 ; i <= n1 ; i++ )
    {
        memset(b, 0, sizeof(b));
        if(find(i))
        {
            ++ans;
        }
    }
    return ans;
}
```

2—SAT—Tarjan

```
vector<int> e[N];
int dfn[N], ins[N], low[N], bel[N], idx, cnt;
stack<int> st;
vector<vector<int> > scc;


void dfs(int u)
{
    dfn[u] = low[u] = ++idx;
    ins[u] = true;
    st.push(u);
    for(auto v : e[u])
    {
        if(!dfn[v])
        {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }else{
            if(ins[v]) low[u] = min(low[u], dfn[v]);
        }
    }
    if(dfn[u] == low[u])
    {
        vector<int> c;
        ++cnt;
        while(true)
        {
            int v = st.top();
```

```
            c.push_back(v);
            ins[v] = false;
            bel[v] = cnt;
            st.pop();
            if(v == u) break;
        }
        sort(c.begin(), c.end());
        scc.push_back(c);
    }

}
int main()
{
    fastio
    //freopen("1.in","r",stdin);
    int n, m;
    cin >> n >> m;
    for(int i = 1 ; i <= m ; i++ )
    {
        int u, ch1, v, ch2;
        cin >> u >> ch1 >> v >> ch2;
        u = 2 * u + (ch1 == 0);
        v = 2 * v + (ch2 == 0);
        e[u ^ 1].push_back(v);
        e[v ^ 1].push_back(u);
    }
    for(int i = 1 ; i <= 2 * n ; i++ )
    {
        if(!dfn[i]) dfs(i);
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        if(bel[2 * i] == bel[2 * i + 1])
        {
            cout << "IMPOSSIBLE\n";
            return 0;
        }
    }
    cout << "POSSIBLE\n";
    for(int i = 1 ; i <= n ; i++ )
    {
        cout << (bel[2 * i] < bel[2 * i + 1]) << " ";
    }
    cout << endl;
    return 0;
}
```

ax-by=1的解

```
ll exgcd(ll a, ll b, ll &x, ll &y)
{
    if(b == 0)
    {
        x = 1;
```

```
        y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}

void solve()
{
    ll a, b;
    cin >> a >> b;
    ll x, y;
    ll d = exgcd(a, b, x, y);
    y = -y;
    while(x < 0 || y < 0)
    {
        x += b/d;
        y += a/d;
    }
    while(x >= b/d && y >= a/d)
    {
        x -= b/d;
        y -= a/d;
    }
    cout << x << " " << y << "\n";
}
```

树状数组

```
template<class T>
struct BIT{
    T c[N];
    void change(int x, T y)
    {
        for(; x < N ; x += x & (-x))
        {
            c[x] += y;
        }
    }
    T query(int x)
    {
        T s = 0;
        for(; x ; x -= x & (-x))
        {
            s += c[x];
        }
        return s;
    }
};
```

并查集

```cpp
struct DSU {
    std::vector<int> f, siz;
    DSU(int n) : f(n), siz(n, 1) { std::iota(f.begin(), f.end(), 0); }
    int leader(int x) {
        while (x != f[x]) x = f[x] = f[f[x]];
        return x;
    }
    bool same(int x, int y) { return leader(x) == leader(y); }
    bool merge(int x, int y) {
        x = leader(x);
        y = leader(y);
        if (x == y) return false;
        siz[x] += siz[y];
        f[y] = x;
        return true;
    }
    int size(int x) { return siz[leader(x)]; }
};
```

exgcd

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

exkmp（Z函数）

```cpp
int z[N];
int n;
string s;
void exkmp()
{
    int L = 1, R = 0;
    z[1] = 0;
    for(int i = 2 ; i <= n ; i++ )
    {
        if(i > R)
        {
            z[i] = 0;
        }else{
            int k = i - L + 1;
            z[i] = min(z[k], R - i + 1);
        }
        while(i + z[i] <= n && s[z[i] + 1] == s[i + z[i]])
```

```
        {
            ++z[i];
        }
        if(i + z[i] - 1 > R)
        {
            L = i;
            R = i + z[i] - 1;
        }
    }
}

void solve()
{
    string s1, s2;
    cin >> s1 >> s2;
    s = '#' + s2 + '#' + s1;
    int len1 = s1.size(), len2 = s2.size();
    n = len1 + 1 + len2;
    exkmp();
    ll ans = 0;
    vector<int> v;
    for(int i = len2 + 2 ; i <= n ; i++ )
    {
        if(z[i] == len2) ans++, v.push_back(i - len2 - 1);
    }
    if(ans)
    {
        cout << ans << "\n";
        for(auto x : v)
        {
            cout << x << " ";
        }
        cout << "\n";
    }else{
        cout << "-1\n-1\n";
    }
}
```

SCC hosoraju

```
int vis[N], n, m;
vector<int> out, c, e[N], erev[N];
int sz[N];
int bel[N], cnt;
vector<vector<int> >scc;

void dfs1(int u)
{
    vis[u] = 1;
    for(auto v : e[u])
    {
        if(!vis[v]) dfs1(v);
    }
    out.push_back(u);
```

```
}

void dfs2(int u, int cnt)
{

    vis[u] = 1;
    for(auto v : erev[u])
    {
        if(!vis[v]) dfs2(v, cnt);
    }
    bel[u] = cnt;
    sz[cnt]++;
    c.push_back(u);
}

int main()
{
    fastio
    //freopen("1.in","r",stdin);
    int n, m, x, y;
    cin >> n >> m;
    for(int i = 1 ; i <= m ; i++ )
    {
        cin >> x >> y;
        e[x].push_back(y);
        erev[y].push_back(x);
    }
    memset(vis, 0, sizeof(vis));
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!vis[i])
        {
            dfs1(i);
        }
    }
    reverse(out.begin(), out.end());
    memset(vis, 0, sizeof(vis));
    for(auto u : out)
    {
        if(!vis[u])
        {
            c.clear();
            dfs2(u, ++cnt);
            sort(c.begin(), c.end());
            scc.push_back(c);
        }

    }
    sort(scc.begin(), scc.end());
    for(auto c : scc)
    {
        for(auto x : c)
        {
            cout << x << " ";
        }
```

```
        cout << "\n";
    }
    return 0;
}
```

SCC Tarjan

```
vector<int> e[N];
int dfn[N], ins[N], low[N], bel[N], idx, cnt;
stack<int> st;
vector<vector<int> > scc;


void dfs(int u)
{
    dfn[u] = low[u] = ++idx;
    ins[u] = true;
    st.push(u);
    for(auto v : e[u])
    {
        if(!dfn[v])
        {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }else{
            if(ins[v]) low[u] = min(low[u], dfn[v]);
        }
    }
    if(dfn[u] == low[u])
    {
        vector<int> c;
        ++cnt;
        while(true)
        {
            int v = st.top();
            c.push_back(v);
            ins[v] = false;
            bel[v] = cnt;
            st.pop();
            //cout << v << " ";
            if(v == u) break;
        }
        //cout << endl;
        sort(c.begin(), c.end());
        scc.push_back(c);
    }

}

int main()
{
    fastio
    //freopen("1.in","r",stdin);
    int n, m;
```

```
    cin >> n >> m;
    for(int i = 1 ; i <= m ; i++ )
    {
        int x, y;
        cin >> x >> y;
        e[x].push_back(y);
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!dfn[i])
        {
            dfs(i);
        }
    }
    sort(scc.begin(), scc.end());
    for(auto c : scc)
    {
        for(auto x : c)
        {
            cout << x << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

边双连通分量

```
int head[N], e[N], nxt[N], idx = 1, n, m;
int dfn[M], low[M], cnt, b[N], bel[N], anscnt[M];
vector<vector<int> > dcc;
void add(int x, int y)
{
    nxt[++idx] = head[x];
    head[x] = idx;
    e[idx] = y;
}
void tarjan(int x, int e_in)
{
    dfn[x] = low[x] = ++cnt;
    for(int i = head[x] ; i ; i = nxt[i])
    {
        int y = e[i];
        if(!dfn[y])
        {
            tarjan(y, i);
            if(dfn[x] < low[y])
            {
                b[i] = b[i ^ 1] = 1;
            }
            low[x] = min(low[x], low[y]);
        }else if (i != (e_in ^ 1))
        {
            low[x] = min(low[x], dfn[y]);
        }
```

```
    }
}

vector<int> v;

void dfs(int x, int cnt)
{
    bel[x] = cnt;
    v.push_back(x);
    anscnt[cnt]++;
    for(int i = head[x] ; i ; i = nxt[i])
    {
        int y = e[i];
        if(bel[y] || b[i]) continue;
        dfs(y, cnt);
    }

}
signed main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m;
    int x, y;
    for(int i = 1 ; i <= m ; i++ )
    {
        cin >> x >> y;
        if(x == y) continue;
        add(x, y);
        add(y, x);
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!dfn[i]) tarjan(i, 0);
    }
    int ans = 0;
    for(int i = 1 ; i <= n ; i++ )
    {
        if(!bel[i])
        {
            v.clear();
            dfs(i, ++ans);
            dcc.push_back(v);
        }

    }
    int sz = dcc.size();
    cout << dcc.size() << "\n";
    for(int i = 0 ; i < sz ; i++ )
    {
        auto v = dcc[i];
        cout << anscnt[i + 1] << " ";
        for(auto x : v)
        {
            cout << x << " ";
```

```
        }
        cout << "\n";
    }
    return 0;
}
```

笛卡尔树

```
//每个父节点都小于其所有子节点

int a[N], n, l[N], r[N];
int root = 0;

void build()
{
    stack<int> st;
    for(int i = 1 ; i <= n ; i++ )
    {
        int last = 0;
        while(!st.empty() && a[st.top()] > a[i])
        {
            last = st.top();
            st.pop();
        }
        if(!st.empty())
        {
            r[st.top()] = i;
        }else{
            root = i;
        }
        l[i] = last;
        st.push(i);
    }
}
```

割点

```
int n, m;
int dfn[N], idx, low[N];
bool vis[N], cut[N];
vector<int> e[N];
int cnt;

void dfs(int u, int root)
{
    vis[u] = 1;
    dfn[u] = ++idx;
    low[u] = idx;
    int child = 0;
    for(auto v : e[u])
    {
        if(!vis[v])
        {
```

```
            dfs(v, root);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u] && u != root)
            {
                cut[u] = 1;
            }
            if(u == root)
            {
                child++;
            }
        }
        low[u] = min(low[u], dfn[v]);
    }
    if(child >= 2 && u == root)
    {
        cut[u] = 1;
    }
}

int main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m;
    rep(i, 1, m + 1)
    {
        int x, y;
        cin >> x >> y;
        e[x].push_back(y);
        e[y].push_back(x);
    }
    rep(i, 1, n + 1)
    {
        if(!vis[i])
        {
            dfs(i, i);
        }
    }
    cout << accumulate(cut + 1, cut + n + 1, 0ll) << "\n";
    rep(i, 1, n + 1)
    {
        if(cut[i])
        {
            cout << i << " ";
        }
    }
    return 0;
}
```

简单计算几何

```
using i64 = long long;

using T = double;
```

```
struct Point {
    T x;
    T y;
    Point(T x = 0, T y = 0) : x(x), y(y) {}

    Point &operator+=(const Point &p) {
        x += p.x, y += p.y;
        return *this;
    }
    Point &operator-=(const Point &p) {
        x -= p.x, y -= p.y;
        return *this;
    }
    Point &operator*=(const T &v) {
        x *= v, y *= v;
        return *this;
    }
    friend Point operator-(const Point &p) {
        return Point(-p.x, -p.y);
    }
    friend Point operator+(Point lhs, const Point &rhs) {
        return lhs += rhs;
    }
    friend Point operator-(Point lhs, const Point &rhs) {
        return lhs -= rhs;
    }
    friend Point operator*(Point lhs, const T &rhs) {
        return lhs *= rhs;
    }
};

T dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y;
}

T cross(const Point &a, const Point &b) {
    return a.x * b.y - a.y * b.x;
}
```

无向图欧拉图

```
vector<pair<int ,int > > e[N];
int d[N], n, m;
int f[N], b[N], sz[N], ans[N], idxans;

void dfs(int x)
{
    //cout << "dfs = " << x << endl;
    for(; f[x] < sz[x] ; )
    {
        int y = e[x][f[x]].first, id = e[x][f[x]].second;
        if(!b[id])
        {
            b[id] = 1;
```

```
            f[x]++;
            dfs(y);
            ans[++idxans] = y;
        }else{
            f[x]++;
        }
    }
}

void Euler()
{
    memset(f, 0, sizeof(f));
    memset(b, 0 ,sizeof(b));
    int cnt = 0, x = 0;
    for(int i = 1 ; i <= n ; i++ )
    {
        if(d[i] & 1)
        {
            cnt++;
            x = i;
        }
    }
    if(!(cnt == 0 || cnt == 2))
    {
        cout << "No\n";
        return;
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        sz[i] = e[i].size();
        if(!x)
            if(d[i])
            {
                x = i;
            }
    }
    dfs(x);
    ans[++idxans] = x;
    if(idxans == m + 1)
    {
        cout << "Yes\n";
    }else{
        cout << "No\n";
    }
}
int main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m;
    int idx = 0;
    for(int i = 1 ; i <= m ; i++ )
    {
        int x, y;
        cin >> x >> y;
```

```
        ++idx;
        ++d[x];
        ++d[y];
        e[x].push_back({y, idx});
        e[y].push_back({x, idx});

    }
    Euler();
    return 0;
}
```

有向图欧拉图

```
int n;
vector<int> e[N];
int ind[N], outd[N], f[N], sz[N], ans[N], idx = 0;

void dfs(int x)
{
    for(; f[x] < sz[x] ;)
    {
        int y = e[x][f[x]];
        f[x]++;
        dfs(y);
        ans[++idx] = y;
    }
}
void Euler()
{
    memset(f, 0, sizeof(f));
    int cntdiff = 0;
    int cntin = 0;
    int x = 0;
    for(int i = 1 ; i <= n ; i++ )
    {
        if(ind[i] != outd[i])
        {
            cntdiff++;
        }
        if(ind[i] + 1 == outd[i])
        {
            cntin++;
            x = i;
        }
    }
    if(!(cntdiff == 2 && cntin == 1 || cntdiff == 0))
    {
        cout << "No\n";
        return;
    }
    for(int i = 1 ; i <= n ; i++ )
    {
        sz[i] = e[i].size();
        //cout << e[i].size();
```

```
        if(!x)
        {
            if(ind[i])
            {
                x = i;
            }
        }
    }
    dfs(x);
    ans[++idx]= x;
    if(idx == n + 1)
    {
        cout << "Yes\n";
    }else{
        cout << "No\n";
    }
    for(int i = idx ; i > 0 ; i--)
    {
        cout << ans[i] << " ";
    }
}
```

二维树状数组维护区间查询，修改

```
ll c1[N][N], c2[N][N], c3[N][N], c4[N][N];

int n, m, k, q;

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

void add(ll x, ll y, ll d)
{
    for(int i = x ; i <= n ; i += lowbit(i))
    {
        for(int j = y ; j <= m ; j += lowbit(j))
        {
            //cout << "test" << endl;
            c1[i][j] += d;
            c2[i][j] += d * x;
            c3[i][j] += d * y;
            c4[i][j] += d * x * y;
        }
    }
}

void modify(int x1, int y1, int x2, int y2, int d)
{
    add(x1, y1, d);
    add(x1, y2 + 1, -d);
    add(x2 + 1, y1, -d);
    add(x2 + 1, y2 + 1, d);
```

```
}

ll sum(ll x, ll y)
{
    ll ans = 0;
    for(int i = x ; i ; i -= lowbit(i))
    {
        for(int j = y ; j ; j -= lowbit(j))
        {
            ans += (x + 1) * (y + 1) * c1[i][j];
            ans -= (y + 1) * c2[i][j];
            ans -= (x + 1) * c3[i][j];
            ans += c4[i][j];
        }
    }
    return ans;
}
ll query(int x1, int y1, int x2, int y2)
{
    return (sum(x2, y2) - sum(x1 - 1, y2) - sum(x2, y1 - 1) + sum(x1 - 1, y1 -
1));
}
int h[100005];
int main()
{
    fastio
    //freopen("1.in","r",stdin);
    cin >> n >> m >> k >> q;
    for(int i = 1 ; i <= k ; i++ )
    {
        cin >> h[i];
    }
    for(int i = 1 ; i <= q ; i++ )
    {
        int op;
        cin >> op;
        if(op == 1)
        {
            int a, b, c, d, id;
            cin >> a >> b >> c >> d >> id;
            modify(a, b, c, d, h[id]);
        }else{
            int a, b, c, d;
            cin >> a >> b >> c >> d;
            cout << query(a, b, c, d) << "\n";
        }
    }
    return 0;
}
```

线段树（区间查询最小值，最小值个数）

```
struct Node{
```

```
    int minx, cntminx;
};

ll a[N];

Node tr[4 * N];

void pushup(int u, int L, int R)
{
    if(tr[u << 1].minx < tr[u << 1 | 1].minx)
    {
        tr[u].minx = tr[u << 1].minx;
        tr[u].cntminx = tr[u << 1].cntminx;
    }
    if(tr[u << 1].minx > tr[u << 1 | 1].minx)
    {
        tr[u].minx = tr[u << 1 | 1].minx;
        tr[u].cntminx = tr[u << 1 | 1].cntminx;
    }
    if(tr[u << 1].minx == tr[u << 1 | 1].minx)
    {
        tr[u].minx = tr[u << 1 | 1].minx;
        tr[u].cntminx = tr[u << 1].cntminx + tr[u << 1 | 1].cntminx;
    }
}


void build(int u, int L, int R)
{
    int mid = L + R >> 1;
    if(L == R)
    {
        tr[u].minx = a[L];
        tr[u].cntminx = 1;
        return;
    }
    build(u << 1, L, mid);
    build(u << 1 | 1, mid + 1, R);
    pushup(u, L, R);

}

void change(int u, int L, int R, int x, int y)
{
    int mid = L + R >> 1;
    if(L == R)
    {
        tr[u].minx = y;
        return;
    }
    if(x <= mid)
    {
        change(u << 1, L, mid, x, y);
    }
    if(x > mid)
```

```
    {
        change(u << 1 | 1, mid + 1, R, x, y);
    }
    pushup(u, L, R);
}

pair<int, int> query(int u, int L, int R, int l, int r)
{
    int mid = L + R >> 1;
    if(l <= L && R <= r)
    {
        return {tr[u].minx, tr[u].cntminx};
    }
    if(r <= mid)
    {
        return query(u << 1, L, mid, l, r);
    }
    if(l >= mid + 1)
    {
        return query(u << 1 | 1, mid + 1, R, l, r);
    }
    auto s1 = query(u << 1, L, mid, l, r);
    auto s2 = query(u << 1 | 1, mid + 1, R, l, r);
    if(s1.first < s2.first)
    {
        return s1;
    }
    if(s1.first > s2.first)
    {
        return s2;
    }
    return {s1.first, s1.second + s2.second};
}

int main()
{
    fastio
    //freopen("1.in","r",stdin);
    int n, m;
    cin >> n >> m;
    for(int i = 1 ; i <= n ; i++ )
    {
        cin >> a[i];
    }
    build(1, 1, n);
    for(int i = 1 ; i <= m ; i++ )
    {
        int op, x, y;
        cin >> op >> x >> y;
        if(op == 1)
        {
            change(1, 1, n, x, y);
        }else{
            auto [_,__] =  query(1, 1, n, x, y);
            cout << _ << " " << __ << "\n";
```

```
        }
    }
    return 0;
}
```

线段树（区间修改加法，区间查询）

```
struct Node{
    ll sum, lazy, size;
};
Node tr[N * 4];
ll a[N];

void pushup(int u, int L, int R)
{
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}


void build(int u, int L, int R)
{
    int mid = L + R >> 1;
    tr[u].size = R - L + 1;
    tr[u].sum = tr[u].lazy = 0;
    if(L == R)
    {
        tr[u].sum = a[L];
        return;
    }
    build(u << 1, L, mid);
    build(u << 1 | 1, mid + 1, R);
    pushup(u, L, R);

}

void pushdown(int u)
{
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if(root.lazy)
    {
        left.sum += root.lazy * left.size;
        left.lazy += root.lazy;
        right.sum += root.lazy * right.size;
        right.lazy += root.lazy;
        root.lazy = 0;
    }
}

void pushup(int u)
{
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}

ll query(int u, int L, int R, int l, int r)
```

```
{
    int mid = L + R >> 1;
    if(l <= L && R <= r)
    {
        return tr[u].sum;
    }
    ll ans = 0;
    pushdown(u);
    if(l <= mid)
    {
        ans += query(u << 1, L, mid, l, r);
    }
    if(r > mid)
    {
        ans += query(u << 1 | 1, mid + 1, R, l, r);
    }
    return ans;
}

void modify(int u, int L, int R, int l, int r, int x)
{
    int mid = L + R >> 1;
    if(l <= L && R <= r)
    {
        tr[u].lazy += x;
        tr[u].sum += x * tr[u].size;
        return;
    }
    pushdown(u);
    if(l <= mid)
    {
        modify(u << 1, L, mid, l , r, x);
    }
    if(r > mid)
    {
        modify(u << 1 | 1, mid + 1, R, l, r, x);
    }
    pushup(u);
}
```

线段树（区间修改加与乘，区间查询）

```
struct Node{
    ll sum, mul, add, size;
} tr[4 * N];
ll a[N];

void pushup(int u)
{
    tr[u].sum = (tr[u << 1].sum % P + tr[u << 1 | 1].sum % P) % P;
}

void pushdown(int u)
{
```

```
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    root.mul  %= P, root.add %= P;
    left.sum  *= root.mul;                left.sum  %= P;
    left.sum  += root.add * left.size;    left.sum  %= P;
    right.sum *= root.mul;                right.sum %= P;
    right.sum += root.add * right.size;   right.sum %= P;
    left.add  *= root.mul;                left.add  %= P;
    left.mul  *= root.mul;                left.mul  %= P;
    right.add *= root.mul;                right.add %= P;
    right.mul *= root.mul;                right.mul %= P;
    left.add  += root.add;                left.add  %= P;
    right.add += root.add;                right.add %= P;
    root.mul  = 1;
    root.add  = 0;
}

void build(int u, int L, int R)
{
    int mid = L + R >> 1;
    tr[u].size = R - L + 1;
    tr[u].mul = 1;
    tr[u].add = 0;
    if(L == R)
    {
        tr[u].sum = a[L] % P;
        return;
    }
    build(u << 1, L, mid);
    build(u << 1 | 1, mid + 1, R);
    pushup(u);
}

void modify_add(int u, int L, int R, int l, int r, int x)
{
    int mid = L + R >> 1;
    if(l <= L && R <= r)
    {
        tr[u].sum += tr[u].size * x;    tr[u].sum %= P;

        tr[u].add += x;                 tr[u].add %= P;
        return;
    }
    pushdown(u);
    if(l <= mid)
    {
        modify_add(u << 1, L, mid, l, r, x);
    }
    if(r >= mid + 1)
    {
        modify_add(u << 1 | 1, mid + 1, R, l, r, x);
    }
    pushup(u);
}

void modify_mul(int u, int L, int R, int l, int r, int x)
```

```
{
    int mid = L + R >> 1;
    if(l <= L && R <= r)
    {
        tr[u].sum *= x; tr[u].sum %= P;
        tr[u].add *= x; tr[u].add %= P;
        tr[u].mul *= x; tr[u].mul %= P;
        return;
    }
    pushdown(u);
    if(l <= mid)
    {
        modify_mul(u << 1, L, mid, l, r, x);
    }
    if(r >= mid + 1)
    {
        modify_mul(u << 1 | 1, mid + 1, R, l, r, x);
    }
    pushup(u);
}

ll query(int u, int L, int R, int l, int r)
{
    if(l <= L && R <= r)
    {
        return tr[u].sum % P;
    }
    pushdown(u);
    ll ans = 0;
    int mid = L + R >> 1;
    if(l <= mid)
    {
        ans += query(u << 1, L, mid, l, r);
        ans %= P;
    }
    if(r >= mid + 1)
    {
        ans += query(u << 1 | 1, mid + 1, R, l, r);
        ans %= P;
    }
    pushup(u);
    return ans % P;
}
```