一、常用

头文件

```cpp
/*

_/      _/    _/      _/    _/      _/   _/_/_/_/_/     _/_/       _/      _/
 _/    _/     _/      _/     _/    _/        _/       _/    _/     _/      _/
  _/  _/      _/      _/      _/  _/         _/      _/      _/    _/_/    _/_/
   _/_/       _/_/_/_/_/        _/           _/      _/      _/    _/  _/  _/
  _/  _/      _/      _/        _/           _/      _/      _/    _/      _/
 _/    _/     _/      _/        _/           _/       _/    _/     _/      _/
_/      _/    _/      _/        _/           _/         _/_/       _/      _/

*/
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("Ofast")
#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
#define pb push_back
#define eb emplace_back
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;
int ksm(int x,int y){int ans=1;x%=MOD;while(y){if(y&1)ans=ans*x%MOD;x=x*x%MOD,y/=2;}return ans;}

const ll P1 = 999971, base1 = 101;
const ll P2 = 999973, base2 = 103;
const ll N = 200005;
//head


signed main()
```

```
{
#ifdef localfreopen
    // freopen("1.in","r",stdin);
#endif
    fastio

    return 0;
}
```

快读

```
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;
}
```

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

```
__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');
}
```

二、字符串

kmp

```
vector<int> kmp(string s)
{//string的形式为'#' + t1 + '#' + s
    int n = s.size() - 1;
    vector<int> nxt(s.size());
    int j = 0;
    for(int i = 2 ; i <= n ; i++ ){
        while(j && s[j + 1] != s[i]) j = nxt[j];
        if(s[j + 1] == s[i]) j++;
        nxt[i] = j;
    }
    return nxt;
}//从第lent + 2 位 到 lent + lens + 1位为 s
```

manacher

```
vector<int> manacher(string s)
{//string为#A#B#C#...#Z#
    int n = s.size();
    vector<int> d1(n);
    for (int i = 0, l = 0, r = -1; i < n; i++)
    {
        int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
        while (0 <= i - k && i + k < n && s[i - k] == s[i + k])  k++;
        d1[i] = k--;
        if (i + k > r)
        {
            l = i - k;
            r = i + k;
        }
    }
    return d1;
}
```

最小表示法

```
string minrep(string s)
```

```
{//s从s[0]开始存
    int k = 0, i = 0, j = 1, n = s.size();
    while (k < n && i < n && j < n) {
        if (s[(i + k) % n] == s[(j + k) % n]) {
            k++;
        } else {
            s[(i + k) % n] > s[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;
            if (i == j) i++;
            k = 0;
        }
    }
    i = min(i, j);
    return s.substr(i, N) + s.substr(0, i);
}
```

Z函数

```
vector<int> exkmp(string s)
{
    vector<int> p(s.size());
    int n = s.size() - 1;
    int L = 1, R = 0;
    p[1] = 0;
    for(int i = 2 ; i <= n ; i++ )
    {
        if(i > R)
        {
            p[i] = 0;
        }else{
            int k = i - L + 1;
            p[i] = min(p[k], R - i + 1);
        }
        while(i + p[i] <= n && s[p[i] + 1] == s[i + p[i]])
        {
            ++p[i];
        }
        if(i + p[i] - 1 > R)
        {
            L = i;
            R = i + p[i] - 1;
        }
    }
    return p;
}//从lent + 2位到lent + lens + 1位为 s
//******p[1] = 0, 但实际从第一位往后能匹配lent的总长
```

AC自动机

```
int n, idx;
struct Node{
    int fail, nxt[26], end;
}trie[150000];


```

```
string ss[155];
int cnt[155];

void add_string(string s, int num)
{
    int p = 0;
    for(int i = 0 ; i < s.size() ; i++ )
    {
        int x = s[i] - 'a';
        if(!trie[p].nxt[x])
        {
            trie[p].nxt[x] = ++idx;
        }
        p = trie[p].nxt[x];
    }
    trie[p].end = num;
}

void get_fail()
{
    queue<int> q;
    rep(i, 0, 26)
    {
        if(trie[0].nxt[i])
        {
            trie[trie[0].nxt[i]].fail = 0;
            q.push(trie[0].nxt[i]);
        }
    }
    while(!q.empty())
    {
        int x = q.front();
        q.pop();
        rep(i, 0, 26)
        {
            if(trie[x].nxt[i])
            {
                trie[trie[x].nxt[i]].fail = trie[trie[x].fail].nxt[i];
                q.push(trie[x].nxt[i]);
            }else{
                trie[x].nxt[i] = trie[trie[x].fail].nxt[i];
            }
        }
    }
}
void query_string(string s)
{
    int p = 0;
    for(int i = 0 ; i < s.size() ; i++ )
    {
        int x = s[i] - 'a';
        if(trie[p].nxt[x])
        {
            p = trie[p].nxt[x];
        }else{
            p = trie[trie[p].fail].nxt[x];
        }
        for(int i = p ; i ; i = trie[i].fail)
        {
            cnt[trie[i].end]++;
        }
        //cout << p << " \n"[i == s.size() - 1];
    }
}

signed main()
{
    fastio
    //freopen("1.in","r",stdin);
    string s;
    while(cin >> n)
    {
        if(n == 0) break;
        idx = 0;
        memset(trie, 0, sizeof(trie));
        memset(cnt, 0, sizeof(cnt));
        rep(i, 1, n + 1)
        {
            cin >> ss[i];
            add_string(ss[i], i);
        }
        get_fail();
        cin >> s;
        query_string(s);
        ll ans = *max_element(cnt + 1, cnt + n + 1);
        cout << ans << endl;
        rep(i, 1, n + 1)
        {
            if(cnt[i] == ans) cout << ss[i] << endl;
        }
    }
```

SA(nlogn)

```
struct SA{
    vector<int> sa, rk, oldrk, id, key1, cnt;
    int i, m = 127, p, w;
    bool cmp(int x, int y, int w) {
        return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
    }// key1[i] = rk[id[i]]（作为基数排序的第一关键字数组）
    int n;
    SA(string s)
    {
        n = s.size() - 1;
        oldrk.resize(2 * n + 5);
        sa.resize(n + 2);
        rk.resize(n + 2);
        id.resize(n + 2);
        key1.resize(n + 2);
        cnt.resize(max(n, 130));
        for (i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
        for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
        for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;
        for (w = 1;; w <<= 1, m = p) {  // m=p 就是优化计数排序值域
            for (p = 0, i = n; i > n - w; --i) id[++p] = i;
            for (i = 1; i <= n; ++i)
                if (sa[i] > w) id[++p] = sa[i] - w;
            fill(cnt.begin(), cnt.end(), 0);
            for (i = 1; i <= n; ++i) ++cnt[key1[i] = rk[id[i]]];
            // 注意这里px[i] != i，因为rk没有更新，是上一轮的排名数组

            for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
            for (i = n; i >= 1; --i) sa[cnt[key1[i]]--] = id[i];
            for(int i = 1 ; i <= n ; i++)
            {
                oldrk[i] = rk[i];
            }
            for (p = 0, i = 1; i <= n; ++i)
                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
            if (p == n) {
                break;
            }
        }
    }
};//传***入的string为1~n
```

三、图论

dinic

```
const int V = 1010;
const int E = 101000;
using ll = long long;

template<typename T>
struct MaxFlow
{
    int s, t, vtot;
    int head[V], etot;
    int dis[V], cur[V];
    struct edge
    {
        int v, nxt;
        T f;
    }e[E * 2];
    void addedge(int u, int v, T f)
    {
        e[etot] = {v, head[u], f}; head[u] = etot++;
        e[etot] = {u, head[v], 0}; head[v] = etot++;
    }
    bool bfs()
    {
        for(int i = 1 ; i <= vtot ; i++ )
        {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push(s); dis[s] = 1;
        while(!q.empty())
        {
            int u = q.front(); q.pop();
            for(int i = head[u] ; ~i ; i = e[i].nxt)
            {
                if(e[i].f && !dis[e[i].v])
                {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if(v == t) return true;
                    q.push(v);
                }
            }
        }
        return false;
    }
    T dfs(int u, T m)
    {
        if(u == t) return m;
        T flow = 0;
        for(int i = cur[u]; ~i ; cur[u] = i = e[i].nxt)
        {
            if(e[i].f && dis[e[i].v] == dis[u] + 1)
            {
                T f = dfs(e[i].v, min(m, e[i].f));
                e[i].f -= f;
                e[i ^ 1].f += f;
                m -= f;
                flow += f;
                if(!m) break;
            }
        }
        if(!flow) dis[u] = -1;
        return flow;
    }
    T dinic()
    {
        T flow = 0;
        while(bfs()) flow += dfs(s, numeric_limits<T>::max());
        return flow;
    }
    void init(int s_, int t_, int vtot_ )
    {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for(int i = 1 ; i <= vtot ; i++ )
        {
            head[i] = -1;
        }
    }
};

MaxFlow<ll> g;
//***记得每次init,

```

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;

```

```cpp

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;
}
```

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

```cpp
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;
}
```

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;
}
```

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)
{
```

```cpp
    //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;
```

```cpp
    }
    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] << " ";
    }
}
```

笛卡尔树

```cpp
//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);
    }
}
```

dfs序求lca

```
int main()
{
    int idx = 0;
    vector<int> dfn(n + 5);
    vector st(__lg(n) + 2, vector<int> (n + 5));//****不能改成23****
    function<int(int,int)> get = [&](int x, int y)
    {
        return dfn[x] < dfn[y] ? x : y;
    };
    function<void(int,int)> dfs = [&](int x, int fa)
    {
        st[0][dfn[x] = ++idx] = fa;
        for(int y : adj[x]) if(y != fa) dfs(y, x);
    };
    function<int(int,int)> lca = [&](int u, int v)
    {
        if(u == v) return u;
        if((u = dfn[u]) > (v = dfn[v])) swap(u, v);
        int d = __lg(v - u++);
        return get(st[d][u], st[d][v - (1 << d) + 1]);
    };
    dfs(s, 0);
    for(int i = 1 ; i <= __lg(n) ; i++ )//****不能改成23****
    {
        for(int j = 1 ; j + (1 << i - 1) <= n  ; j++ ) // ****注意边界****
        {
            st[i][j] = get(st[i - 1][j], st[i - 1][j + (1 << i - 1)]);
        }
    }
    /// lca(u, v);
}
```

点分治

```
signed main()
{
    fastio
    int n, k, ans = 0;
    cin >> n >> k;
    ans = n + 1;
    vector<vector<pair<int,int>>> adj(n + 1);
    vector<int> sz(n + 1, 0), maxsz(n + 1, 0), del(n + 1, 0);
    vector<int> mark(k + 1, 0), c(k + 1, 0);
    int T = 1;
    int u, v, w;
    for(int i = 1 ; i < n ; i++ )
    {
        cin >> u >> v >> w;
        u++;
```

```cpp
        v++;
        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }
    function<void(int, int)> solve = [&](int x, int s)
    {
        T++;
        int mxs = s + 1, root = -1;
        function<void(int, int)> dfs1 = [&](int x, int fx)
        {
            sz[x] = 1;
            maxsz[x] = 0;
            for(auto [y, w] : adj[x])
            {
                if(del[y] || y == fx) continue;
                dfs1(y, x);
                sz[x] += sz[y];
                maxsz[x] = max(maxsz[x], sz[y]);
            }
            maxsz[x] = max(maxsz[x], s - sz[x]);
            if(maxsz[x] < mxs)
            {
                mxs = maxsz[x], root = x;
            }
        };
        dfs1(x, -1);
        /////////////////////////////////
        mark[0] = T;
        c[0] = 0;
        for(auto [y, w] : adj[root])
        {
            if(del[y]) continue;
            vector<pair<int, int>> self;
            function<void(int, int, int, int)> dfs2 = [&](int x, int fx, int dis, int dep)
            {
                self.emplace_back(dis, dep);
                for(auto [y, w] : adj[x])
                {
                    if(del[y] || y == fx) continue;
                    dfs2(y, x, dis + w, dep + 1);
                }
            };
            dfs2(y, root, w, 1);
            for(auto [dis, dep] : self)
            {
                if(k - dis >= 0 && mark[k - dis] == T)
                {
                    ans = min(ans, c[k - dis] + dep);
                }
            }
            for(auto [dis, dep] : self)
            {
                if(dis > k) continue;
                if(mark[dis] == T)
                {
                    c[dis] = min(c[dis], dep);
                }else{
                    c[dis] = dep;
                    mark[dis] = T;
                }
            }
        }
        ////////////////////////////////
        del[root] = 1;
        for(auto [y, w] : adj[root])
        {
            if(del[y]) continue;
            solve(y, sz[y]);
        }
    };
    solve(1, n);
    cout << (ans > n ? -1 : ans) << "\n";
    return 0;
}
```

四、数论

exgcd

```
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;
}
```

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;
    }
```

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;
        }
    }
}
```

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";
}
```

pollard_rho

```
using i64 = long long;
using i128 = __int128;
i64 power(i64 a, i64 b, i64 m) {
```

```cpp
    i64 res = 1;
    for (; b; b >>= 1, a = i128(a) * a % m) {
        if (b & 1) {
            res = i128(res) * a % m;
        }
    }
    return res;
}

bool isprime(i64 p) {
    if (p < 2) {
        return 0;
    }
    i64 d = p - 1, r = 0;
    while (!(d & 1)) {
        r++;
        d >>= 1;
    }
    int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    for (auto a : prime) {
        if (p == a) {
            return true;
        }
        i64 x = power(a, d, p);
        if (x == 1 || x == p - 1) {
            continue;
        }
        for (int i = 0; i < r - 1; i++) {
            x = i128(x) * x % p;
            if (x == p - 1) {
                break;
            }
        }
        if (x != p - 1) {
            return false;
        }
    }
    return true;
}

mt19937 rng((unsigned int)
chrono::steady_clock::now().time_since_epoch().count());

i64 pollard_rho(i64 x) {
    i64 s = 0, t = 0;
    i64 c = i64(rng()) % (x - 1) + 1;
    i64 val = 1;
    for (int goal = 1; ; goal <<= 1, s = t, val = 1) {
        for (int step = 1; step <= goal; step++) {
            t = (i128(t) * t + c) % x;
            val = i128(val) * abs(t - s) % x;
            if (step % 127 == 0) {
                i64 g = gcd(val, x);
                if (g > 1) {
                    return g;
                }
                }
            }
            i64 g = gcd(val, x);
            if (g > 1) {
                return g;
            }
        }
}

unordered_map<i64, int> getprimes(i64 x) {
    unordered_map<i64, int> p;
    function<void(i64)> get = [&](i64 x) {
        if (x < 2) {
            return;
        }
        if (isprime(x)) {
            p[x]++;
            return;
        }
        i64 mx = pollard_rho(x);
        get(x / mx);
        get(mx);
    };
    get(x);
    return p;
}

```

五、数据结构

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]));
}
```

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

```
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)]; }
};
```

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);
```

```cpp
}

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);

}

```

```cpp
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;
```

```cpp

        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;
}
```

pdbs

```
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;
__gnu_pbds::tree<ll, null_type, less<ll>, rb_tree_tag,
tree_order_statistics_node_update> T;

if(op == 1)
{
    T.insert({x, i});
}else if (op == 2)
{
    T.erase(T.lower_bound({x, 0}));
}else if (op == 3)
{
    cout << T.order_of_key({x, 0}) + 1 << "\n";
}else if (op == 4)
{
    cout << T.find_by_order(x - 1)->first << "\n";
}else if (op == 5)
{
    cout << prev(T.lower_bound({x, 0}))->first << "\n";
}else if (op == 6)
{
    cout << T.lower_bound({x + 1, 0})->first << "\n";
}
```

六、简单计算几何

点

```
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) {
```

```cpp
        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;
}
```

七、杂项

矩阵快速幂

```cpp
struct Matrix{
    int n , m ;
    vector<vector<ll>> s;

    Matrix(int n , int m):n(n) ,m(m) , s(n , vector<ll>(m ,0)){}

    friend Matrix operator * (Matrix a , Matrix b){
        assert(a.m == b.n);
        Matrix res(a.n , b.m);
        for(int k = 0 ; k < a.m ; k ++ )
            for(int i = 0 ; i < a.n ; i ++ )
                for(int j = 0 ; j < b.m ; j ++ )
                    res.s[i][j] = (res.s[i][j] + a.s[i][k] * b.s[k][j] %
mod) % mod;
        return res;
    }

    Matrix qmi(ll b){
        assert(n == m);
        Matrix res(n , n);
        for(int i = 0 ; i < n ; i ++ )
```

```
            res.s[i][i] = 1;
        while(b){
            if(b & 1)res = ((*this) * res );
            b >>= 1;
            *this = (*this) * (*this);
        }
        return (*this) = res;
    };

};
```

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 P(ll x, ll y)
{
    return fact[x] * 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;
    }
}
```