Contents

1 Boruvka

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

const int MaxN = 5000 + 5, MaxM = 200000 + 5;

int N, M;
int U[MaxM], V[MaxM], W[MaxM];
bool used[MaxM];
int par[MaxN], Best[MaxN];

void init() {
    scanf("%d␣%d", &N, &M);
    for (int i = 1; i <= M; ++i)
        scanf("%d␣%d␣%d", &U[i], &V[i], &W[i]);
}

void init_dsu() {
    for (int i = 1; i <= N; ++i)
        par[i] = i;
}

int get_par(int x) {
    if (x == par[x]) return x;
    else return par[x] = get_par(par[x]);
}

inline bool Better(int x, int y) {
    if (y == 0) return true;
    if (W[x] != W[y]) return W[x] < W[y];
    return x < y;
}

void Boruvka() {
    init_dsu();

    int merged = 0, sum = 0;

    bool update = true;
    while (update) {
        update = false;
        memset(Best, 0, sizeof Best);
```

```
        for (int i = 1; i <= M; ++i) {
            if (used[i] == true) continue;
            int p = get_par(U[i]), q = get_par(V[i]);
            if (p == q) continue;
            if (Better(i, Best[p]) == true) Best[p] = i;
            if (Better(i, Best[q]) == true) Best[q] = i;
        }

        for (int i = 1; i <= N; ++i)
            if (Best[i] != 0 && used[Best[i]] == false) {
                update = true;
                merged++; sum += W[Best[i]];
                used[Best[i]] = true;
                par[get_par(U[Best[i]])] = get_par(V[Best
                    [i]]);
            }
    }

    if (merged == N - 1) printf("%d\n", sum);
    else puts("orz");
}

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

2 djk

```
vector<ll> dis(n + 1, 1E18);
auto djikstra = [&](int s = 1) -> void {
    using PII = pair<ll, ll>;
    std::priority_queue<PII, vector<PII>, greater<PII>> q
        ;
    q.emplace(0, s);
    dis[s] = 0;
    vector<int> vis(n + 1);
    while (!q.empty()) {
        int x = q.top().second;
        q.pop();
        if (vis[x]) continue;
        vis[x] = 1;
        for (auto [y, w] : E[x]) {
            if (dis[y] > dis[x] + w) {
                dis[y] = dis[x] + w;
                q.emplace(dis[y], y);
            }
        }
    }
};
```

3 floyd

```
int v[maxn][maxn];
int a[maxn][maxn];
void floyd{
memset(v,INF,sizeof v)
for(int k=1;k<=n;k++){
    for(int i=1;i<=n;++i){
        for(int j=1;j<=n;j++){
            if(v[i][k]<1<<30&&v[k][j]<1<<30){
                v[i][j]=min(v[i][j],v[i][k]+v[k][j]);
            }
        }
    }
}
}
```

4 jiangly 费用流

```
template<class T>
struct MinCostFlow {
    struct _Edge {
        int to;
        T cap;
        T cost;
        _Edge(int to_, T cap_, T cost_) : to(to_), cap(
            cap_), cost(cost_) {}
    };
    int n;
    std::vector<_Edge> e;
    std::vector<std::vector<int>> g;
    std::vector<T> h, dis;
    std::vector<int> pre;
    bool dijkstra(int s, int t) {
        dis.assign(n, std::numeric_limits<T>::max());
        pre.assign(n, -1);
        std::priority_queue<std::pair<T, int>, std::
            vector<std::pair<T, int>>, std::greater<std::
            pair<T, int>>> que;
        dis[s] = 0;
        que.emplace(0, s);
        while (!que.empty()) {
            T d = que.top().first;
            int u = que.top().second;
            que.pop();
            if (dis[u] != d) {
                continue;
            }
            for (int i : g[u]) {
                int v = e[i].to;
                T cap = e[i].cap;
                T cost = e[i].cost;
                if (cap > 0 && dis[v] > d + h[u] - h[v] +
                     cost) {
                    dis[v] = d + h[u] - h[v] + cost;
                    pre[v] = i;
                    que.emplace(dis[v], v);
                }
            }
        }
        return dis[t] != std::numeric_limits<T>::max();
    }
    MinCostFlow() {}
```

```
MinCostFlow(int n_) {
    init(n_);
}
void init(int n_) {
    n = n_;
    e.clear();
    g.assign(n, {});
}
void addEdge(int u, int v, T cap, T cost) {
    g[u].push_back(e.size());
    e.emplace_back(v, cap, cost);
    g[v].push_back(e.size());
    e.emplace_back(u, 0, -cost);
}
std::pair<T, T> flow(int s, int t) {
    T flow = 0;
    T cost = 0;
    h.assign(n, 0);
    while (dijkstra(s, t)) {
        for (int i = 0; i < n; ++i) {
            h[i] += dis[i];
        }
        T aug = std::numeric_limits<int>::max();
        for (int i = t; i != s; i = e[pre[i] ^ 1].to)
            {
            aug = std::min(aug, e[pre[i]].cap);
        }
        for (int i = t; i != s; i = e[pre[i] ^ 1].to)
            {
            e[pre[i]].cap -= aug;
            e[pre[i] ^ 1].cap += aug;
        }
        flow += aug;
        cost += aug * h[t];
    }
    return std::make_pair(flow, cost);
}
struct Edge {
    int from;
    int to;
    T cap;
    T cost;
    T flow;
};
std::vector<Edge> edges() {
    std::vector<Edge> a;
```

```
        for (int i = 0; i < e.size(); i += 2) {
            Edge x;
            x.from = e[i + 1].to;
            x.to = e[i].to;
            x.cap = e[i].cap + e[i + 1].cap;
            x.cost = e[i].cost;
            x.flow = e[i + 1].cap;
            a.push_back(x);
        }
        return a;
    }
};
```

5 johnson 全源最短路

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int inf=0x3f3f3f3f;
const ll Inf=0x3f3f3f3f3f3f3f3f;
ll n,m;
vector<pair<ll,ll>>v[3009];
vector<pair<ll,ll>>v2[3009];
ll dis0[3009];
ll vis0[3009];
//ll vis2[3009][3009];
ll vis2[3009];
ll cnt[3009];
ll dis[3009][3009];
struct node{
    ll order;
    ll d;
    bool operator <(const node& b)const{
        return d>b.d;
    }
};
ll spfa(){
    int i,j;
    for(i=1;i<=n;i++){
        v[0].emplace_back(i,0);
    }
    dis0[0]=0;
    queue<ll>q;
    vis0[0]=1;
    q.push(0);
    while(!q.empty()){
        ll cur=q.front();
        q.pop();
        vis0[cur]=0;
        cnt[cur]++;
        if(cnt[cur]>n){  //n+1个点，这里是n
            return -1;
        }
        for(i=0;i<(int)v[cur].size();i++){
            if(dis0[v[cur][i].first]>dis0[cur]+v[cur][i].
               second){
                dis0[v[cur][i].first]=dis0[cur]+v[cur][i
                    ].second;
                if(vis0[v[cur][i].first]==0){
```

```
                    q.push(v[cur][i].first);
                    vis0[v[cur][i].first]=1;
                }
            }
        }
    }
    return 1;
}
void dijkstra(ll s){  //这里不再处理新增的那个点
    memset(vis2,0,sizeof(vis2));
    priority_queue<node>q2;
    q2.push(node{s,0});
    dis[s][s]=0;
    ll i;
    while(!q2.empty()){
        node cur=q2.top();
        q2.pop();
        //if(vis2[s][cur.order]==1)continue;
        //vis2[s][cur.order]=1;
        if(vis2[cur.order]==1)continue;
        vis2[cur.order]=1;
        for(i=0;i<(ll)v2[cur.order].size();i++){
            if(dis[s][v2[cur.order][i].first]>dis[s][cur.
                order]+v2[cur.order][i].second){
                dis[s][v2[cur.order][i].first]=dis[s][cur
                    .order]+v2[cur.order][i].second;
                q2.push(node{v2[cur.order][i].first,dis[s
                    ][cur.order]+v2[cur.order][i].second})
                    ;
            }
        }
    }
    return ;
}
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin>>n>>m;
    ll i,j;
    for(i=1;i<=m;i++){
        ll x,y,w;
        cin>>x>>y>>w;
        if(x==y)continue;
        v[x].emplace_back(y,w);
    }
    memset(dis0,0x3f,sizeof(dis0));
```

```
    if(spfa()==-1){
        cout<<-1<<endl;
        return 0;
    }
    for(i=1;i<=n;i++){
        for(j=0;j<(int)v[i].size();j++){
            v2[i].emplace_back(v[i][j].first,v[i][j].
                second+dis0[i]-dis0[v[i][j].first]);
        }
    }
    memset(dis,0x3f,sizeof(dis));
    for(i=1;i<=n;i++){
        dijkstra(i);
    }
    for(i=1;i<=n;i++){
        ll res=0;
        for(j=1;j<=n;j++){
            if(i==j)continue;
            if(dis[i][j]==Inf)res+=j*1e9;
            else res+=j*(dis[i][j]-dis0[i]+dis0[j]);   //
                记得再处理回来
        }
        cout<<res<<"\n";
    }
    return 0;
}
```

6 lca

```
template<typename T> struct Tre
{
    int n, m = 0;
    vector<vector<int>> e;
    vector<int> to;
    vector<T> wt;
    int maxk;
    vector<vector<int>> fa;
    vector<int> dep;
    void add_arc(const int u, const int v, const T w = 0)
        {e[u].push_back(m++); to.push_back(v); wt.
        push_back(w);}
    void add_edge(const int u, const int v, const T w =
        0) {add_arc(u, v, w); add_arc(v, u, w);}
    Tre(const int n, const int maxk = 25): n(n), e(n),
        maxk(maxk), fa(n, vector<int>(maxk + 1, -1)), dep(
        n) {}
    void dfs(const int u, const int f)
    {
        fa[u][0] = f;
        dep[u] = f == -1 ? 0 : dep[f] + 1;
        for (int i = 1; i <= maxk; i++) fa[u][i] = fa[u][
            i - 1] == -1 ? -1 : fa[fa[u][i - 1]][i - 1];
        for (const int i : e[u])
        {
            const int v = to[i];
            if (v == f) continue;
            dfs(v, u);
        }
    }

    int lca(int u, int v) const
    {
        if (dep[u] < dep[v]) swap(u, v);
        for (int i = maxk; i >= 0; i--)
        {
            if (fa[u][i] != -1 && dep[fa[u][i]] >= dep[v
                ]) u = fa[u][i];
        }
        if (u == v) return u;
        for (int i = maxk; i >= 0; i--)
        {
            if (fa[u][i] != fa[v][i]) u = fa[u][i], v =
                fa[v][i];
```

```
            }
            return fa[u][0];
        }
    };
```

7 O1 LCA

```
struct Trh {
    std::vector<int> dep, parent, in;
    int cur, n;
    int logn;
    std::vector<std::vector<int>> e;
    vector<vector<int>>a;
    Trh(int _n) : n(_n),dep(_n), parent(_n, -1), e(_n),
        in(_n), cur(1) {
        logn = std::__lg(n);
        a.assign(logn + 1, std::vector<int>(n + 1));
    }
    void addEdge(int u, int v) {
        e[u].push_back(v);
        e[v].push_back(u);
    }
    void dfs (int x) {
        in[x] =cur++;
        if (cur > 1) {
            a[0][cur - 2] = parent[x];
        }
        for (auto y : e[x]) {
            if (y == parent[x]) {
                continue;
            }
            parent[y] = x;
            dep[y] = dep[x] + 1;
            dfs(y);
        }
    }
    void init(int s) {
        dfs(s);
        for (int j = 0; j < logn; j++) {
            for (int i = 1; i + (2 << j) <= n; i++) {
                a[j + 1][i] = dep[a[j][i]] < dep[a[j][i +
                    (1 << j)]] ? a[j][i] : a[j][i + (1 <<
                    j)];
            }
        }
    }
    int  lca(int x, int y) {
        if (x == y) {
            return x;
        }
        if (in[x] > in[y]) {
```

```
            std::swap(x, y);
        }
        int k = std::__lg(in[y] - in[x]);
        int u = a[k][in[x]];
        int v = a[k][in[y] - (1 << k)];
        return dep[u] < dep[v] ? u : v;
    }
};
```

8 On-O1 lca

```
constexpr int maxn=1e5+7;
int dval[maxn], dfn[maxn], tot,Dfn[maxn];
namespace Rmq {
    int st[20][maxn / 32];
    int pre[maxn], p[maxn], w[maxn];
    inline int getmin(int x, int y) { return dfn[x] < dfn
        [y] ? x : y; }
    inline void down(int & x, int y) { if (dfn[x] > dfn[y
        ]) x = y; }
    inline int qry(int l, int r) {
        const int lg = std::__lg(r - l);
        return l >= r ? 0 : getmin(st[lg][l], st[lg][r -
            (1 << lg)]);
    }
    inline int rmq(int l, int r) {
        if (l >> 5 == r >> 5) return p[l + __builtin_ctz(
            w[r] >> l)];
        else return getmin(qry((l >> 5) + 1, r >> 5),
            getmin(dval[l], pre[r]));
    }
    inline void build(int n) {
        ++ (n |= 31);
        memcpy(p, dval, n << 2);
        for (int i = 0; i < n; i += 32) {
            static int st[33];
            pre[i] = dval[i];
            int * top = st + 1, s = 1; w[*top = i] = s;
            for (int j = i + 1; j < i + 32; ++j) {
                for (; top != st && dfn[dval[j]] < dfn[
                    dval[*top]]; --top) s ^= 1 << *top;
                w[j] = s |= 1 << j; *++top = j; pre[j] =
                    dval[st[1]];
            }
            for (int j = i + 30; j >= i; --j) down(dval[j
                ], dval[j + 1]);
            Rmq::st[0][i >> 5] = dval[i];
        }
        for (int i = 1; i < 15; ++i)
            for (int j = 0; j + (1 << i) - 1 <= n / 32;
                ++j)
                st[i][j] = getmin(st[i - 1][j], st[i -
                    1][j + (1 << i - 1)]);
    }
}
```

```
struct T { int to, nxt; } way[maxn << 1];
int h[maxn], num;
inline void addEdge(int x, int y) {
    way[++num] = {y, h[x]}, h[x] = num;
    way[++num] = {x, h[y]}, h[y] = num;
}
inline void dfs(int x, int f) {
    dval[tot] = f; dfn[x] = ++tot;
    for (int i = h[x]; i; i = way[i].nxt) if (way[i].to
       != f)
            dfs(way[i].to, x);
    Dfn[x]=tot;
}
inline int lca(int x, int y) {
    if (dfn[x] > dfn[y]) std::swap(x, y);
    return x == y ? x : Rmq::rmq(dfn[x], dfn[y] - 1);
}
void init() {
    for (int i = 0; i <= num; i++) {
        way[i].to = 0;
        way[i].nxt = 0;
    }
    for (int i = 0; i <= tot; i++) {
        dval[i] = 0;
        h[i] = 0;
    }
    tot = 0;
    num = 0;
}
void Getfa(int s,int n) {
    dfs(s, 0); *dfn = 1e9; Rmq::build(n - 1);
}
```

9 spfa 判负环

```
vector<pair<int, double> > e[N];
int f[N], cnt[N];
double dist[N];

bool spfa()
{
    queue<int> que;
    for(int i=1;i<=n;i++) que.push(i), f[i] = 1, dist[i]
       = 1, cnt[i] = 0;

    while(que.size())
    {
        int x = que.front(); que.pop();
        f[x] = 0;

        for(auto it : e[x])
        {
            int tx = it.fi; double bi = it.se;
            if(dist[x] + + bi > dist[tx])
            {
                dist[tx] = dist[x] + bi;
                cnt[tx] = cnt[x] + 1;
                if(cnt[tx] >= n) return 0;
                if(!f[tx]) f[tx] = 1, que.push(tx);
            }
        }
    }
    return 1;
}
```

10 tarjan 求 scc

```
struct SCC {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> stk;
    std::vector<int> dfn, low, bel;
    int cur, cnt;

    SCC() {}
    SCC(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        adj.assign(n, {});
        dfn.assign(n, -1);
        low.resize(n);
        bel.assign(n, -1);
        stk.clear();
        cur = cnt = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
    }

    void dfs(int x) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = std::min(low[x], low[y]);
            } else if (bel[y] == -1) {
                low[x] = std::min(low[x], dfn[y]);
            }
        }

        if (dfn[x] == low[x]) {
            int y;
            do {
                y = stk.back();
                bel[y] = cnt;
```

```
                stk.pop_back();
            } while (y != x);
            cnt++;
        }
    }

    std::vector<int> work() {
        for (int i = 0; i < n; i++) {
            if (dfn[i] == -1) {
                dfs(i);
            }
        }
        return bel;
    }
};
```

11 tarjan 求点双

```
#include<bits/extc++.h>
using namespace __gnu_cxx;
using namespace __gnu_pbds;
using namespace std;
using ll = long long;
#define LNF 0x3f3f3f3f3f3f3f3f
#define INF 0x3f3f3f3f
#define IOS ios::sync_with_stdio(false);cin.tie(0);cout.
    tie(0);
#define pll pair<int,int>
#define fi first
#define se second
constexpr int N = 1e5 + 6;
std::vector<int> e[N];
int dfn[N], low[N], idx,cut[N],sz;
void dfs(int u, int f) {
    dfn[u] = low[u] = ++idx;
    int ch = 0;
    for (auto v : e[u]) {
        if (!dfn[v]){
            dfs(v,u);
            ch++;
            low[u]=min(low[u],low[v]);
            if(low[v]>=dfn[u])cut[u]=1;
        }else if(v!=f){
            low[u]=min(low[u],dfn[v]);
        }
    }
    if(u==1 and ch<=1)cut[u]=0;
    sz+=cut[u];
}
void solve(void) {
    ll n, m;
    cin>>n>>m;
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    for(int i=1;i<=n;i++)if(!dfn[i])
    dfs(i, -1);
    cout<<sz<<"\n";
    for(int i=1;i<=n;i++)if(cut[i])cout<<i<<"\n";
```

```
}
int main() {

    IOS;
    int t = 1;
    // cin>>t;
    while (t--)
        solve();
    return 0;
}
```

12 tarjan 求边双

```
struct EBCC {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> stk;
    std::vector<int> dfn, low, bel;
    int cur, cnt;

    EBCC() {}
    EBCC(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        adj.assign(n, {});
        dfn.assign(n, -1);
        low.resize(n);
        bel.assign(n, -1);
        stk.clear();
        cur = cnt = 0;
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x, int p) {
        dfn[x] = low[x] = cur++;
        stk.push_back(x);
        for (auto y : adj[x]) {
            if (y == p) {
                continue;
            }
            if (dfn[y] == -1) {
                dfs(y, x);
                low[x] = std::min(low[x], low[y]);
            } else if (bel[y] == -1 && dfn[y] < dfn[x]) {
                low[x] = std::min(low[x], dfn[y]);
            }
        }

        if (dfn[x] == low[x]) {
            int y;
```

```
            do {
                y = stk.back();
                bel[y] = cnt;
                stk.pop_back();
            } while (y != x);
            cnt++;
        }
    }

    std::vector<int> work() {
        for (int i = 1; i < n; i++) {
            if (dfn[i] == -1) {
                dfs(i, 0);
            }
        }
        return bel;
    }

    struct Graph {
        int n;
        std::vector<std::pair<int, int>> edges;
        std::vector<int> siz;
        std::vector<int> cnte;
    };
    Graph compress() {
        Graph g;
        g.n = cnt;
        g.siz.resize(cnt);
        g.cnte.resize(cnt);
        for (int i = 1; i < n; i++) {
            g.siz[bel[i]]++;
            for (auto j : adj[i]) {
                if (bel[i] < bel[j]) {
                    g.edges.emplace_back(bel[i], bel[j]);
                } else if (i < j) {
                    g.cnte[bel[i]]++;
                }
            }
        }
        return g;
    }
};
```

13 twosat

```
struct TwoSat {
    int n;
    std::vector<std::vector<int>> e;
    std::vector<bool> ans;
    TwoSat(int n) : n(n), e(2 * n), ans(n) {}
    void addClause(int u, bool f, int v, bool g) {
        e[2 * u + !f].push_back(2 * v + g);
        e[2 * v + !g].push_back(2 * u + f);
    }
    bool satisfiable() {
        std::vector<int> id(2 * n, -1), dfn(2 * n, -1),
            low(2 * n, -1);
        std::vector<int> stk;
        int now = 0, cnt = 0;
        std::function<void(int)> tarjan = [&](int u) {
            stk.push_back(u);
            dfn[u] = low[u] = now++;
            for (auto v : e[u]) {
                if (dfn[v] == -1) {
                    tarjan(v);
                    low[u] = std::min(low[u], low[v]);
                } else if (id[v] == -1) {
                    low[u] = std::min(low[u], dfn[v]);
                }
            }
            if (dfn[u] == low[u]) {
                int v;
                do {
                    v = stk.back();
                    stk.pop_back();
                    id[v] = cnt;
                } while (v != u);
                ++cnt;
            }
        };
        for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1)
            tarjan(i);
        for (int i = 0; i < n; ++i) {
            if (id[2 * i] == id[2 * i + 1]) return false;
            ans[i] = id[2 * i] > id[2 * i + 1];
        }
        return true;
    }
    std::vector<bool> answer() { return ans; }
```

```
};
```

14 二分图匹配

```
vector<int>edge[maxn];
int n,m,n1,n2,v[maxn];
bool b[maxn];
bool find(int x){
    b[x]=true;
    for(auto y:edge[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;
}
```

15 圆方树

```
#include<bits/extc++.h>
using namespace std;
const int MN = 100005;
int N, M, cnt;
std::vector<int> G[MN], T[MN * 2];

int dfn[MN], low[MN], dfc;
int stk[MN], tp;

void Tarjan(int u) {
    printf("␣␣Enter␣:␣#%d\n", u);
    low[u] = dfn[u] = ++dfc; // low 初始化为当前节点 dfn
    stk[++tp] = u; // 加入栈中
    for (int v : G[u]) { // 遍历 u 的相邻节点
        if (!dfn[v]) { // 如果未访问过
            Tarjan(v); // 递归
            low[u] = std::min(low[u], low[v]); // 未访问
                的和 low 取 min
            if (low[v] == dfn[u]) { // 标志着找到一个以 u
                为根的点双连通分量
                ++cnt; // 增加方点个数
                printf("␣␣Found␣a␣New␣BCC␣#%d.\n", cnt -
                    N);
                // 将点双中除了 u 的点退栈，并在圆方树中
                    连边
                for (int x = 0; x != v; --tp) {
                    x = stk[tp];
                    T[cnt].push_back(x);
                    T[x].push_back(cnt);
                    printf("␣␣␣␣BCC␣#%d␣has␣vertex␣#%d\n"
                        , cnt - N, x);
                }
                // 注意 u 自身也要连边（但不退栈）
                T[cnt].push_back(u);
                T[u].push_back(cnt);
                printf("␣␣␣␣BCC␣#%d␣has␣vertex␣#%d\n",
                    cnt - N, u);
            }
        } else low[u] = std::min(low[u], dfn[v]); // 已访
            问的和 dfn 取 min
    }
    printf("␣␣Exit␣:␣#%d␣:␣low␣=␣%d\n", u, low[u]);
    printf("␣␣Stack:\n␣␣␣␣");
    for (int i = 1; i <= tp; ++i) printf("%d,␣", stk[i]);
```

```
    puts("");
}

int main() {
    scanf("%d%d", &N, &M);
    cnt = N; // 点双 / 方点标号从 N 开始
    for (int i = 1; i <= M; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        G[u].push_back(v); // 加双向边
        G[v].push_back(u);
    }
    // 处理非连通图
    for (int u = 1; u <= N; ++u)
        if (!dfn[u]) Tarjan(u), --tp;
        // 注意到退出 Tarjan 时栈中还有一个元素即根，将其
           退栈
    return 0;
}
```

16 最大流

```
template<class T>
struct Flow {
    const int n;
    struct Edge {
        int to;
        T cap;
        Edge(int to, T cap) : to(to), cap(cap) {}
    };
    std::vector<Edge> e;
    std::vector<std::vector<int>> g;
    std::vector<int> cur, h;
    Flow(int n) : n(n), g(n) {}
    bool bfs(int s, int t) {
        h.assign(n, -1);
        std::queue<int> que;
        h[s] = 0;
        que.push(s);
        while (!que.empty()) {
            const int u = que.front();
            que.pop();
            for (int i : g[u]) {
                auto [v, c] = e[i];
                if (c > 0 && h[v] == -1) {
                    h[v] = h[u] + 1;
                    if (v == t) {
                        return true;
                    }
                    que.push(v);
                }
            }
        }
        return false;
    }

    T dfs(int u, int t, T f) {
        if (u == t) {
            return f;
        }
        auto r = f;
        for (int &i = cur[u]; i < int(g[u].size()); ++i)
            {
            const int j = g[u][i];
            auto [v, c] = e[j];
            if (c > 0 && h[v] == h[u] + 1) {
```

```
                auto a = dfs(v, t, std::min(r, c));
                e[j].cap -= a;
                e[j ^ 1].cap += a;
                r -= a;
                if (r == 0) {
                    return f;
                }
            }
        }
        return f - r;
    }
    void addEdge(int u, int v, T c) {
        g[u].push_back(e.size());
        e.emplace_back(v, c);
        g[v].push_back(e.size());
        e.emplace_back(u, 0);
    }
    T maxFlow(int s, int t) {
        T ans = 0;
        while (bfs(s, t)) {
            cur.assign(n, 0);
            ans += dfs(s, t, std::numeric_limits<T>::max
                ());
        }
        return ans;
    }
};
```

17 费用流

```
#include<bits/stdc++.h>
using namespace std;
using ll=long long;
#define INF 0x3f3f3f3f
#define IOS ios::sync_with_stdio(false);cin.tie(0);cout.
    tie(0);
#define pll pair<int,int>
#define fi first
#define se second
const int V=20100;
const int E=201000;
template<typename T>
struct MinCostGraph
{
    int s,t,vtot;
    int head[V],cur[V],etot;
    T dis[V],flow,cost;
    int pre[V];
    bool vis[V];
    struct edge{
        int v,nxt;
        T f,c;
    }e[E*2];
    void addedge(int u,int v,T f,T c,T f2=0){
        e[etot]={v,head[u],f,c};head[u]=etot++;
        e[etot]={u,head[v],f2,-c};head[v]=etot++;
    }
    bool spfa(){
        T inf=numeric_limits<T>::max()/2;
        for(int i=1;i<=vtot;i++){
            dis[i]=inf;
            vis[i]=false;
            pre[i]=-1;
        }
        dis[s]=0;
        vis[s]=true;
        queue<int>q;
        q.push(s);
        while(!q.empty()){
            int u=q.front();
            for(int i=head[u];~i;i=e[i].nxt){
                int v=e[i].v;
                if(e[i].f&&dis[v]>dis[u]+e[i].c){
                    dis[v]=dis[u]+e[i].c;
```

```
                    pre[v]=i;
                    if(!vis[v]){
                        vis[v]=1;
                        q.push(v);
                    }
                }
            }
            q.pop();
            vis[u]=false;
        }
        return dis[t]!=inf;
    }
    void augment(){
        int u=t;
        T f=numeric_limits<T>::max();
        while(~pre[u]){
            f=min(f,e[pre[u]].f);
            u=e[pre[u]^1].v;
        }
        flow+=f;
        cost+=f*dis[t];
        u=t;
        while(~pre[u]){
            e[pre[u]].f-=f;
            e[pre[u]^1].f+=f;
            u=e[pre[u]^1].v;
        }
    }
    pair<T,T>solve(){
        flow=0;
        cost =0;
        while(spfa())augment();
        return {flow,cost};
    }
    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;
    }

};
MinCostGraph<int>g;
int n,m,s,t;
void solve(void){
```

```
    cin>>n>>m>>s>>t;
    g.init(s,t,n);
    for(int i=1;i<=m;i++){
        int u,v,f,c;
        cin>>u>>v>>f>>c;
        g.addedge(u,v,f,c);

    }
    auto [flow,cost]=g.solve();
    cout<<flow<<" "<<cost;

}
int main() {

    IOS;
    int t=1;
    // cin>>t;
    while(t--)
    solve();
    return 0;
}
```