







#include<iostream>

#include<cstring>

#include<queue>

using namespace std;

const int N = 110,M = (1000+2\*N)\*2,INF = 0x3f3f3f3f;

const double eps = 1e-8;

int n,m,s,t,dis[N],cur[N]; //dinic

int e[M],ne[M],h[N],idx; //链式前向星存图

double w[M]; //容量存成double型

int res,dg[N]; //dg存点的度数

bool vis[N];

pair<int,int> es[M];

void add(int a,int b,double c) //建图模板

{

e[idx] = b;

w[idx] = c;

ne[idx] = h[a];

h[a] = idx++;

e[idx] = a;

w[idx] = c;

ne[idx] = h[b];

h[b] = idx++;

}

void build(double g) //建图函数，每次二分到一个答案时要重新建图

{

memset(h,-1,sizeof h);

idx = 0;

for(int i = 0;i < m;i++)

add(es[i].first,es[i].second,1);

for(int i = 1;i <= n;i++)

{

add(s,i,m);

add(i,t,m+g\*2-dg[i]);

}

}

bool bfs() //dinic模板

{

queue<int> q;

memset(dis,-1,sizeof dis);

q.push(s);

dis[s] = 0;

cur[s] = h[s];

while(q.size())

{

int u = q.front();

q.pop();

for(int i = h[u];~i;i = ne[i])

{

int v = e[i];

if(dis[v] == -1 && w[i])

{

dis[v] = dis[u] + 1;

cur[v] = h[v];

if(v == t)

return 1;

q.push(v);

}

}

}

return 0;

}

double dfs(int u,double limit)

{

if(u == t)

return limit;

double flow = 0;

for(int i = cur[u];~i;i = ne[i])

{

cur[u] = i;

int v = e[i];

if(dis[v] == dis[u]+1 && w[i])

{

double minf = dfs(v,min(w[i],limit-flow));

w[i] -= minf;

w[i^1] += minf;

flow += minf;

if(flow == limit)

return flow;

}

}

return flow;

}

double dinic()

{

double ans = 0;

while(bfs())

ans += dfs(s,INF);

return ans;

}

bool judge(double mid) //二分的判断函数

{

build(mid);

double res = m\*n-dinic();

return res > 0;

}

void find(int u) //一个dfs找方案

{

vis[u] = 1;

if(u != s)

res++;

for(int i = h[u];~i;i = ne[i])

{

int v = e[i];

if(!vis[v] && w[i])

find(v);

}

}

int main()

{

cin >> n >> m;

s = 0,t = n+1; //手动设置s和t

for(int i = 0;i < m;i++)

{

int a,b;

cin >> a >> b;

dg[a]++,dg[b]++; //a和b的度数+1

es[i] = {a,b}; //暂时先将边存下

}

double l = 0,r = m; //定义二分的左右端点

while(r-l > eps) //二分答案g

{

double mid = (r+l)/2;

if(judge(mid))

l = mid;

else

r = mid;

}

build(l); //重新建一个图

dinic(); //跑一边dinic

find(s); //从s出发找能到达的点

if(res)

{

cout << res << endl;

for(int i = 1;i <= n;i++)

if(vis[i])

cout << i << endl;

}

else

cout << 1 << endl << 1 << endl;

return 0;

}