C：

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*pNext;

};

struct Node tab[1001];

int visit[1001]={0};

int way[1001]={0};

int count[1001]={0};

int cnt=0;

void Insert(int n,int x);

void Init(int n);

void dfs(int x,int y,int n);

int fun(int n);

int main()

{

int x,y,n,m,u,v;

scanf("%d%d",&n,&m);

Init(n);

while(m--)

{

scanf("%d%d",&u,&v);

Insert(u,v);

Insert(v,u);

}

scanf("%d%d",&x,&y);

dfs(x,y,0);

int ret=fun(n);

printf("%d\n",ret);

return 0;

}

int fun(int n)

{

int i;

int ret=0;

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

{

if(count[i]==cnt)

{

ret++;

}

}

return (ret-2);

}

void dfs(int x,int y,int n)

{

visit[x]=1;

way[n]=x;

struct Node \*p=&tab[x];

if(x==y)

{

int i;

cnt++;

for(i=0;i<=n;i++)

{

count[way[i]]++;

}

return ;

}

while((p=p->pNext)!=NULL)

{

if(visit[p->data]!=1)

{

dfs(p->data,y,n+1);

visit[p->data]=0;

}

}

}

void Init(int n)

{

int i;

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

{

tab[i].data=i;

tab[i].pNext=NULL;

}

}

void Insert(int n,int x)

{

struct Node \*p=&tab[n];

while(p->pNext!=NULL)

{

p=p->pNext;

}

struct Node \*new=(struct Node \*)malloc(sizeof(struct Node));

p->pNext=new;

new->data=x;

new->pNext=NULL;

}

C++：

#include <iostream>

#include <cmath>

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <map>

#include <set>

#include <list>

#include <queue>

#include <stack>

#include <string>

#include <vector>

#include <iomanip>

#include <algorithm>

using namespace std;

typedef long long LL;

#define clr(x, y) memset(x, y, sizeof(x))

#define sf scanf

#define pf printf

const double pi = 4.0 \* atan(1.0);

const double eps = 1e-8;

const int INF = 1 << 29;

const int maxn = 1005;

int N, M, roadNum, vis[maxn], mark[maxn];

vector<int> edge[maxn];

void init() {

for (int i = 1; i <= N; i++) {

edge[i].clear();

}

roadNum = 0;

clr(vis, 0), clr(mark, 0);

}

void dfs(int u, int tar) {

vis[u] = 1;

if (u == tar) {

roadNum++;

for (int i = 1; i <= N; i++) {

if (vis[i]) mark[i]++;

}

return ;

}

for (int i = 0; i < edge[u].size(); i++) {

int v = edge[u][i];

if (vis[v]) continue;

dfs(v, tar);

vis[v] = false;

}

}

int solve() {

init();

for (int i = 1; i <= M; i++) {

int u, v; sf("%d%d",&u,&v);

edge[u].push\_back(v);

edge[v].push\_back(u);

}

int start, end;

sf("%d%d",&start,&end);

dfs(start, end);

int ans = 0;

for (int i = 1; i <= N; i++) {

if (mark[i] == roadNum && i != start && i != end) {

ans++;

}

}

return roadNum == 0 ? -1 : ans;

}

int main() {

while (~sf("%d%d",&N,&M)) {

pf("%d\n", solve());

}

return 0;

}

Java：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.StringTokenizer;

public class Main {

static int n,m;

public static void main(String[] args) throws IOException {

BufferedReader bfr=new BufferedReader(new InputStreamReader(System.in));

StringTokenizer tok=new StringTokenizer(bfr.readLine());

// long begin=System.currentTimeMillis();

n=Integer.parseInt(tok.nextToken());

m=Integer.parseInt(tok.nextToken());

ArrayList<Integer>[] way=new ArrayList[n];

int count=0;

int i;

for(i=0;i<n;i++) way[i]=new ArrayList<Integer>();

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

{

tok=new StringTokenizer(bfr.readLine());

int x=Integer.parseInt(tok.nextToken())-1, y=Integer.parseInt(tok.nextToken())-1;

way[x].add(y); way[y].add(x);

}

tok=new StringTokenizer(bfr.readLine());

int g1=Integer.parseInt(tok.nextToken())-1, g2=Integer.parseInt(tok.nextToken())-1;

for(i=0;i<n;i++)

{

if(i==g1 || i==g2) continue;

boolean[] flag=new boolean[n];

int[] que=new int [n];

int front=0,rear=1;

que[0]=g1;

while(front<rear)

{

for(int p=0;p<way[que[front]].size();p++)

{

if(way[que[front]].get(p)==i) continue;

if (!flag[way[que[front]].get(p)])

{

que[rear] = way[que[front]].get(p);

flag[que[rear]]=true;

if(que[rear]==g2) break;

rear++;

}

}

if(flag[g2]) { count++; break; }

front++;

}

}

System.out.println(n-2-count);

// System.out.println(System.currentTimeMillis()-begin);

}

}