A题题解

题目

众所周知，蜘蛛是一种擅长织网的生物。

然而，其编织的网常常因为莫名原因遭到破坏，因此他不得不重新修补他的网。

蜘蛛的网可以看作一个有 n 个点 m条边的无向联通图，对于每次修补，蜘蛛实际上可以只把被破坏的部分拆开再重新织被破坏的部分就可以完成修补。

因此对于蜘蛛来说，他想知道他至少需要拆开多少条边，使得蛛网不再联通，这样他才可以完成修补。

Standard Input

第一行两个正整数 n,m,m 含义如题所示。

接下来 mm 行，每行两个正整数 x,y，表示 x 和 y 之间有一条无向边。

输入数据保证合法且无自环。

Standard Output

输出共一行，即为最少需要断掉的边

Samples

Input	Output
`10 20 1 2 2 3 1 4 3 5 2 6 5 7 5 8 1 9 5 10 7 8 2 6 5 4 4 9 1 4 10 6 8 10 3 5 5 10 7 3 8 6`	`2`

Constraints

n \leq 300, m \leq 1000$n \leq 300, m \leq 1000$

Problem ID	2709
Problem Title	蜘蛛的网
Time Limit	1000 ms
Memory Limit	256 MiB
Output Limit	64 MiB
Source	2022 UESTC ICPC Training for Graph

题解

题目分析

根据题意我们可以知道，本题我们主要求解无向连通图的最小割的求解，根据最大流最小割定理，我们可以根据StoerWagner算法来求解

相比于普通的网络流算法，StoerWangner算法取消了源汇点的定义。无源汇点的最小割问题，包含的弧的权和最小的割。也称为全局最小割。

根据时间复杂度分析显然行不通。

定义介绍

S-T割：

使得顶点S与顶点T不再连通的割，称为S-T割

S-T最小割：

包含的弧的权和最小的S-T割，称为S-T最小割。

全局最小割：

包含的弧的权和最小的割，称为全局最小割。

算法流程

每次在当前的无向图 $G(V,E)$ 中不断维护一个点集 A ,开始点集 A包含一个任选的点 a .

当 A ≠ V 时选取属于V但不属于A且 $\sum_{uinAuvinE} W(u,v)$最大的点 v 加入点集 A ， 直到 A=V 时结束。

此时令倒数第二次加入A的点为 s ， 最后一次加入的点为 t ， 则s-t最小割为割 $(A-t,t)$ ， 此时的全局最小割即为操作中的 s-t 最小割的最小值。

我们重复进行这些操作$n-1$次，总复杂度最优可以做到$\mathrm{O}((|\mathrm{E}|+|\mathrm{V}|)\log|\mathrm{V}|)$

代码

```
//A题
//网络流最小割最大流问题
//StoerWagner算法
#include <bits/stdc++.h>
using namespace std;
const int MAXN=505;
const int INF=1e9;
int mp[MAXN][MAXN];
int vis[MAXN];
int dis[MAXN],part[MAXN];
int N;
int run(int x)
{
    int ret=INF;
    for(int i=0;i<x;i++) part[i]=i;
    while(x>1)
    {
        int pre=0;
        for(int i=0;i<=N;i++) vis[i]=dis[i]=0;
        for(int i=1;i<x;i++)
        {
```

```
            int k=-1;
            for(int j=1;j<x;j++)
            {
                if(vis[part[j]])continue;
                dis[part[j]]+=mp[part[pre]][part[j]];
                if(k==-1||dis[part[j]]>dis[part[k]])
                    k=j;
            }
            vis[part[k]]=1;
            if(i==x-1){
                int s=part[pre],t=part[k];
                if(ret>dis[t]&&dis[t])
                    ret=dis[t];
                for(int j=0;j<x;j++)
                {
                    mp[s][part[j]]+=mp[t][part[j]];
                    mp[part[j]][s]+=mp[t][part[j]];
                }
                part[k]=part[--x];
            }
            pre=k;
        }
    }
    return ret;
}
int main()
{
    int m=0,n=0;
    scanf("%d %d",&n,&m);N=n;
    for(int i=1;i<=m;i++)
    {
        int u,v,w;
        scanf("%d %d",&u,&v);
        mp[u][v]++;
        mp[v][u]++;
    }
    int ans=run(n);
    cout<<ans<<endl;
    return 0;
}
```