地震伤害

一. 考察内容：

网络流 最小割 拆点

二. 题目分析：

[题目大意]

给出一张无向图，和一些不连通点x，表示x与1的所有路径上都有节点是损坏的，且x不是损坏的。求出满足这些不连通信息，最少需要损坏多少节点。

注：道路不会损坏

[写题思路]

首先考虑建出一个流网络，求出使得流网络无法从1点流到不连通点，最少需要损坏多少点。

建图：

1. 把每个点拆为两个节点，不妨设为a和a'，a表示a点的入点，a'表示a点的出点，a和a'之间连接一条容量为1的边，a—>a' flow=1，表示切断该点需要1的花费。
2. 对于原图两个节点a、b之间的一条边在网络上连接一条a'—>b flow=INF，表示道路是不可以切断的。
3. 对于每个不连通点和1节点，需要补上一条附加边a—>a' flow=INF，表示该点不可切断。
4. 最后，为1节点与s连接一条边s—>1 flow=INF，不连通点与t连接一条边a'—>t flow=INF。

建图完成！

最后用dinic算法跑出最小割即可。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 09 08
*文件类型：源代码文件
*题目来源：BZOJ
*当前状态：已通过
*备忘录：图论 网络流 最小割 Dinic
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
#include <bitset>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 3010
#define MAXM 40010
int n,m,k,s,t,head[2*MAXN],_edge,d[2*MAXN],hu[2*MAXN];
struct EDGE
```

```
{
    int a,b,flow,next;
    EDGE(int a=0,int b=0,int flow=0,int next=0):a(a),b(b),flow(flow),next(next){}
}edge[2*MAXM];
/* Variable explain:

*/
void adde(int a,int b,int flow,int double_line)
{
    edge[++_edge]=EDGE(a,b,flow,head[a]);
    head[a]=_edge;
    if(!double_line)return;
    edge[++_edge]=EDGE(b,a,0,head[b]);
    head[b]=_edge;
}
void read()
{
    int ls1,ls2;
    scanf("%d%d%d",&n,&m,&k);
    _edge=1;
    s=2*n+1,t=s+1;
    for(int i=1;i<=m;+
+i)scanf("%d%d",&ls1,&ls2),adde(ls1+n,ls2,1e9,0),adde(ls2+n,ls1,1e9,0);
    adde(s,1,1e9,1);adde(1,n+1,1e9,1);
    for(int i=1;i<=n;++i)adde(i,i+n,1,1);
    for(int i=1;i<=k;++i)scanf("%d",&ls1),adde(ls1+n,t,1e9,1),adde(ls1,ls1+n,1e9,1);
    return;
}
bool bfs()
{
    deque <int> q;
    for(int i=1;i<=t;++i)d[i]=0;
    q.push_back(s);
    d[s]=1;
    while(!q.empty())
    {
        int a=q.front();
        q.pop_front();
        for(int i=head[a];i;i=edge[i].next)
        {
            int b=edge[i].b,flow=edge[i].flow;
            if(!d[b]&&flow)d[b]=d[a]+1,q.push_back(b);
        }
    }
    return d[t];
}
int dfs(int a,int nowflow)
{
    if(a==t||!nowflow)return nowflow;
    int flow,ans=0;
    for(int &i=hu[a];i;i=edge[i].next)
    {
        int b=edge[i].b;
        if(d[b]==d[a]+1&&(flow=dfs(b,min(edge[i].flow,nowflow))))
        {
            edge[i].flow-=flow;
            edge[i^1].flow+=flow;
            nowflow-=flow;
            ans+=flow;
            if(!nowflow)break;
        }
    }
    return ans;
}
int dinic()
{
    int ans=0;
    while(bfs())
    {
        for(int i=1;i<=t;++i)hu[i]=head[i];
        ans+=dfs(s,1e9);
    }
```

```
    return ans;
}
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    read();
    printf("%d\n",dinic());
    return 0;
}
```

<题目跳转> <查看代码>