题意

很久以前，在一个遥远的星系，一个黑暗的帝国靠着它的超级武器统治者整个星系。某一天，凭着一个偶然的机遇，一支反抗军摧毁了帝国的超级武器，并攻下了星系中几乎所有的星球。这些星球通过特殊的以太隧道互相直接或间接地连接。 但好景不长，很快帝国又重新造出了他的超级武器。凭借这超级武器的力量，帝国开始有计划地摧毁反抗军占领的星球。由于星球的不断被摧毁，两个星球之间的通讯通道也开始不可靠起来。现在，反抗军首领交给你一个任务：给出原来两个星球之间的以太隧道连通情况以及帝国打击的星球顺序，以尽量快的速度求出每一次打击之后反抗军占据的星球的连通快的个数。（如果两个星球可以通过现存的以太通道直接或间接地连通，则这两个星球在同一个连通块中）。

思路

删除一个点，就要删除该点以及该点的所有边。逆过程就是，增加一个点，然后将这个点和某些点连边

按操作顺序处理的话要在删除点的同时维护图的形态（即图具体的连边情况），这是几乎不可做的

按照删除点的倒序，依次增加点，那么我们就可以用并查集维护他们的联通关系

具体看代码

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
vector<int>edge[400005];
int que[400005];
int ans[400005];
int fa[400005];
vector<int>v;
bool exist[400005];
int getf(int x)
{
    if(x == fa[x]) return x;
    return fa[x] = getf(fa[x]);
}
int main()
```

```
{
    int n,m;
    scanf("%d%d",&n,&m);
    int x,y;
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d",&x,&y);
        edge[x].push_back(y);
        edge[y].push_back(x);
    }
    for(int i=0;i<n;i++)
    {
        fa[i] = i;
    }
    int k;
    scanf("%d",&k);
    memset(exist,true,sizeof(exist));
    for(int i=1;i<=k;i++)
    {
        scanf("%d",&que[i]);
        exist[que[i]] = false;
    }
    for(int i=0;i<n;i++)
    {
        if(exist[i] == true)
        {
            v.push_back(i);
        }
    }
    ans[k+1] = v.size();
    for(int i=0;i<v.size();i++)
    {
        int x = v[i];
        for(int j = 0;j<edge[x].size();j++)
        {
            int y = edge[x][j];
            if(exist[y] == false) continue;
            int fx = getf(x);
            int fy = getf(y);
            if(fx!=fy)
            {
                ans[k+1] --;
                fa[fx] = fy;
            }
        }
    }
```

```
62      for(int num = k;num>=1;num--)
63      {
64
65          ans[num] = ans[num+1]+1;
66          v.push_back(que[num]);
67          int x = que[num];
68          exist[x] = true;
69          for(int j = 0;j<edge[x].size();j++)
70          {
71              int y = edge[x][j];
72              if(exist[y] == false) continue;
73              int fx = getf(x);
74              int fy = getf(y);
75              if(fx!=fy)
76              {
77                  ans[num] --;
78                  fa[fx] = fy;
79              }
80          }
81      }
82      for(int i=1;i<=k+1;i++)
83      {
84          printf("%d\n",ans[i]);
85      }
86  }
```