题意解释

有一个 n 个点 m 条边的无向图。给定一个起点为 1，终点为 n 的路径 $x_1, x_2, \cdots, x_k$。如果存在 (x_i, x_j) 边且 $i < j$，则可以从 x_i 直接到x_j。 求 1 到 n 的最短路。

知识点提炼

建图、最短路

核心解题思路

思路1：二进制枚举(30pts)

$n \leq 15$时可以枚举剩下了哪些点，判断剩下的点之间有没有连边。如果全部有边则更新答案。

```
#include<bits/stdc++.h>
using namespace std;
int ans,n,m,k;
int e[20][20],a[20];
void dfs(vector<int>now,int d)//now中维护了当前选中了哪些点
{
    if(d==k+1)
    {
        now.push_back(a[k+1]);//a[k+1]=n，必选
        for(int i=0;i+1<now.size();i++)
            if(e[now[i]][now[i+1]]==0)//如果相邻两个点之间没有边则return
                return ;
        ans=min(ans,int(now.size()-1));//更新答案
        return ;
    }
    dfs(now,d+1);//不选第d个点
    now.push_back(a[d]);//选第d个点
    dfs(now,d+1);
}
void work()
{
    scanf("%d%d%d",&n,&m,&k);
    ans=k;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            e[i][j]=0;//清空边
    for(int i=1;i<=k+1;i++)
        scanf("%d",&a[i]);
    for(int i=1;i<=k;i++)
        e[a[i]][a[i+1]]=e[a[i+1]][a[i]]=1;//读入边
    for(int i=1;i<=m-k;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        e[x][y]=e[y][x]=1;//读入边
    }
    dfs({1},2);//开始dfs
```

```
    cout<<ans<<'\n';
}
int main()
{
    int t;
    for(scanf("%d",&t);t;t--)
        work();
}
```

时间复杂度$O(2^n * n)$，期望得分30分。

思路2：建图后最短路(100pts)

由于数据保证了路径是一条合法的1到n的路径，路径上的点两两不同，故我们给每个原始路上的点标记一个时间戳，则只允许走从时间戳小的连时间戳大的边，不允许反过来或者走到无时间戳的点。于是将所有无向边定向，跑最短路即可

```
#include<bits/stdc++.h>
using namespace std;
int n,m,k,t[50010],d[50010],a[50010];
queue<int>q;
vector<int>e[50010];
void work()
{
    scanf("%d%d%d",&n,&m,&k);
    for(int i=1;i<=n;i++){
        d[i]=0x3f3f3f3f;
        t[i]=0;
        e[i].clear();//初始化最短路数组、时间戳数组、边集
    }
    for(int i=1;i<=k+1;i++)
    {
        scanf("%d",&a[i]);//读入ai，设置时间戳
        t[a[i]]=i;
    }
    for(int i=1;i<=k;i++)//将原旅游计划的边放入边集中
        e[a[i]].push_back(a[i+1]);
    for(int i=1;i<=m-k;i++)//对于剩下的边
    {
        int x,y;
        scanf("%d%d",&x,&y);
        if(t[x]==t[y])//如果是自环或者x、y的时间戳都为0则不加边
            continue;
        if(t[x]<t[y])//否则从时间戳小的点向时间戳大的点连边
            e[x].push_back(y);
        else
            e[y].push_back(x);
    }
    d[1]=0;//准备开始bfs
    q.push(1);
    while(q.size())
    {
        int x=q.front();
        q.pop();
        for(auto y:e[x])
```

```
            if(d[y]>d[x]+1)//更新其他点的最短距离
            {
                d[y]=d[x]+1;
                q.push(y);
            }

    }
    cout<<d[n]<<'\n';
}
int main()
{
    int t;
    for(scanf("%d",&t);t;t--)
        work();
}
```

时间复杂度$O(n+m)$，期望得分100分。

本题易错点

容易理解错误题意。

多组数据，容易不清空数组、赋初值。

标准代码

C++

```
#include<bits/stdc++.h>
using namespace std;
int n,m,k,t[50010],d[50010],a[50010];
queue<int>q;
vector<int>e[50010];
void work()
{
    scanf("%d%d%d",&n,&m,&k);
    for(int i=1;i<=n;i++){
        d[i]=0x3f3f3f3f;
        t[i]=0;
        e[i].clear();
    }
    for(int i=1;i<=k+1;i++)
    {
        scanf("%d",&a[i]);
        t[a[i]]=i;
    }
    for(int i=1;i<=k;i++)
        e[a[i]].push_back(a[i+1]);
    for(int i=1;i<=m-k;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        if(t[x]==t[y])
            continue;
        if(t[x]<t[y])
```

```
            e[x].push_back(y);
        else
            e[y].push_back(x);
    }
    d[1]=0;
    q.push(1);
    while(q.size())
    {
        int x=q.front();
        q.pop();
        for(auto y:e[x])
            if(d[y]>d[x]+1)
            {
                d[y]=d[x]+1;
                q.push(y);
            }
    }
    cout<<d[n]<<'\n';
}
int main()
{
    int t;
    for(scanf("%d",&t);t;t--)
        work();
}
```