D —— 解题报告

题意：Mr.Black 新买了一幢别墅，可是里面的电灯线路是混乱的，即每盏灯和其对应的开关不一定在同一个房间里。有一天 Mr.Black 回家发现屋子里的灯都是关的（除了他所在的这一间，即起点），现在他想回到卧室去，可是他又很怕黑，所以他不会进入没有开灯的房间。要求找出一条最短的，使得他既能回到卧室，又能关闭除了卧室以外的其他灯的路线。

思路：由于房间数 r 不超过 10,而对于任意的一组 case，只有 r*(2^r)种可能的状态，显然可以 bfs。注意本题没有 spj，那么行动顺序要按字典序最小的来，即 move>off>on，而且开关灯时，灯也要按字典序来操作，所以要对输入数据进行 sort，并且 bfs 时按照 move>off>on 的顺序搜索。利用 stl 的话很好写的。

```
#include <iostream>
#include <sstream>
#include <vector>
#include <queue>
#include <algorithm>

using namespace std;

int main(void)
{
    int R,D,S;
    for(int cas=1;cin>>R>>D>>S && R;cas++)
    {
        vector<vector<int> >g(R);
        for(int i=0;i<D;i++)
        {
            int u,v;
            cin>>u>>v;
            u--,v--;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        vector<vector<int> >switches(R);
        for(int i=0;i<S;i++)
        {
            int u,v;
            cin>>u>>v;
            if(u!=v)
            {
                u--,v--;
                switches[u].push_back(v);
            }
        }
```

```
for(int i=0;i<R;i++)
    sort(switches[i].begin(),switches[i].end());

queue<pair<int,int> >q;
q.push(make_pair(0,1));
vector<vector<int> >dist(R,vector<int>(1<<R,1000000000));
vector<vector<pair<bool,int> > >pre(R,vector<pair<bool,int> >(1<<R));
dist[0][1]=0;
cout<<" Villa #"<<cas<<endl;
bool flag=0;
while(!q.empty())
{
    int n=q.front().first;
    int s=q.front().second;
    q.pop();
    const int d=dist[n][s];
    if(n==R-1 && s==(1<<(R-1)))
    {
        cout<<"The problem can be solved in "<<d<<" steps:"<<endl;
        vector<string> msg;
        for(int i=0;i<d;i++)
        {
            ostringstream oss;
            const int j=pre[n][s].second;
            if(pre[n][s].first)
            {
                oss<<"- Switch "<<(s & (1<<j)?"on":"off")<<" light in room "<<j+1<<".";
                s ^= (1<<j);
            }
            else
            {
                oss<<"- Move to room "<<n+1<<".";
                n=j;
            }
            msg.push_back(oss.str());
        }
        for(vector<string>::reverse_iterator it=msg.rbegin();it!=msg.rend();it++)
            cout<<*it<<endl;
        flag=1;
        break;
    }
    for(vector<int>::const_iterator it(g[n].begin());it!=g[n].end();it++)
        if(s & (1<<*it) && d+1<dist[*it][s])
```

```
                {
                    dist[*it][s]=d+1;
                    pre[*it][s]=make_pair(0,n);
                    q.push(make_pair(*it,s));
                }
            for(vector<int>::const_iterator it(switches[n].begin());it!=switches[n].end();it++)
            {
                const int t=s^(1<<*it);
                if(d+1<dist[n][t])
                {
                    dist[n][t]=d+1;
                    pre[n][t]=make_pair(1,*it);
                    q.push(make_pair(n,t));
                }
            }
        }
        if(!flag)       cout<<"The problem cannot be solved."<<endl;
        cout<<endl;
    }
    return 0;
}
```