[解题报告] POJ1324

[Source] http://poj.org/problem?id=1324

[Description]

简单的贪吃蛇游戏，n*m(n,m<=20)的迷宫，里面有若干障碍物，蛇的长度为 l(l<=8)，蛇可以向上下左右四个方向走，规则是不能碰到自己的身体和障碍物以及不能走出迷宫。迷宫的出口坐标为（1,1），求蛇走出迷宫的最少步数，若不存在解，输出-1。

[Solution]

宽度优先搜索。迷宫并不大，蛇也不太长，但还是要先计算好空间和时间，毕竟搜索怕的就是超空间和超时。我的想法是把蛇的状态用蛇头的坐标，以及当前蛇节与下一节的相对位置（上下左右四个方向），这样还是比较能省些空间的。空间复杂度应该是 O(n*m*4^l)。然后将蛇的状态哈希来判重（我哈希写的一向很挫），剩下的就是传统的 bfs 过程了。

[Code]

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<algorithm>
#define N 25
using namespace std;

struct Point
{
    char x,y;
};

struct node
{
    Point head;
    char next[7];
    int step;
};

char mx[4]={0,0,1,-1},my[4]={1,-1,0,0};
```

```
char mark[6000000],map[N][N];
int pow[8];
int n,m,l;
queue<node>que;

int hash(Point head,char next[])
{
    int i,res;

    res=head.x*m+head.y;

    for (i=0;i<l-1;i++)
        res+=next[i]*pow[i];

    return res%6000000;
}

int legal(Point head,char next[],int x,int y)
{
    Point pos[8];
    int i;

    pos[0]=head;
    for (i=1;i<l;i++)
    {
        pos[i].x=pos[i-1].x+mx[next[i-1]];
        pos[i].y=pos[i-1].y+my[next[i-1]];

        if (x==pos[i].x&&y==pos[i].y)
            return 0;
    }

    return 1;
}


int main()
{
    int r,i,j,ans,key,flag,ys;
    Point pos[8];
    int x,y;
    node u,v,s;

    ys=0;
```

```
while (scanf("%d%d%d",&n,&m,&l),n||m||l)
{
    pow[0]=n*m;
    for (i=1;i<8;i++)
        pow[i]=pow[i-1]*4;

    memset(map,0,sizeof(map));
    memset(mark,0,sizeof(mark));
    while (!que.empty())
        que.pop();

    for (i=0;i<l;i++)
    {
        scanf("%d%d",&x,&y);
        pos[i].x=x-1;
        pos[i].y=y-1;
    }

    s.step=0;
    s.head=pos[0];
    for (i=0;i<l-1;i++)
    {
        for (j=0;j<4;j++)
        {
            x=pos[i].x+mx[j];
            y=pos[i].y+my[j];

            if (x==pos[i+1].x&&y==pos[i+1].y)
            {
                s.next[i]=j;
                break;
            }
        }
    }

    scanf("%d",&r);
    for (i=0;i<r;i++)
    {
        scanf("%d%d",&x,&y);
        x--;
        y--;
        map[x][y]=1;
    }
```

```
key=hash(s.head,s.next);
mark[key]=1;
ans=-1;
que.push(s);

while (!que.empty())
{
    u=que.front();
    que.pop();

    if (u.head.x==0&&u.head.y==0)
    {
        ans=u.step;
        break;
    }

    for (i=0;i<4;i++)
    {
        x=u.head.x+mx[i];
        y=u.head.y+my[i];

        flag=1;
        if (!(x>=0&&x<n&&y>=0&&y<m&&map[x][y]==0))
            flag=0;

        if (flag)
            flag=legal(u.head,u.next,x,y);

        if (flag)
        {
            v.step=u.step+1;
            v.head.x=x;
            v.head.y=y;
            v.next[0]=i^1;
            for (j=1;j<l-1;j++)
                v.next[j]=u.next[j-1];


            key=hash(v.head,v.next);

            if (mark[key]==0)
            {
                mark[key]=1;
```

```
                    que.push(v);
                }
            }
        }
    }

    printf("Case %d: %d\n",++ys,ans);
}

return 0;
}
```