[报告]hdu1428 漫步校园

[Source]

http://acm.hdu.edu.cn/showproblem.php?pid=1428

[Description]

LL 最近沉迷于 AC 不能自拔，每天寝室、机房两点一线。由于长时间坐在电脑边，缺乏运动。他决定充分利用每次从寝室到机房的时间，在校园里散散步。整个 HDU 校园呈方形布局，可划分为 n*n 个小方格，代表各个区域。例如 LL 居住的 18 号宿舍位于校园的西北角，即方格(1,1)代表的地方，而机房所在的第三实验楼处于东南端的(n,n)。因有多条路线可以选择，LL 希望每次的散步路线都不一样。另外，他考虑从 A 区域到 B 区域仅当存在一条从 B 到机房的路线比任何一条从 A 到机房的路线更近(否则可能永远都到不了机房了…)。现在他想知道的是，所有满足要求的路线一共有多少条。你能告诉他吗?

最让人郁闷得应该就是“他考虑从 A 区域到 B 区域仅当存在一条从 B 到机房的路线比任何一条从 A 到机房的路线更近(否则可能永远都到不了机房了…)”意思就是，如果用 cost[i][j]表示（i，j）到机房的最短路的话，那么他可以选择四个方向移动，如果移动点的 cost 大于当前位置，那么就是不可以走的，反之就是可以走的。

[Solution]

bfs 求每个点到终点的最短路。

从（1,1）点开始 dfs 记忆化搜索，计算出路径的总数。搜到终点就+1，搜到的地方已经有保存的值就直接返回。

[Code]

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
#include<queue>

using namespace std;

const int INF=0xfffffff;

struct node
{
    int x;
    int y;
};

int map[60][60];//pointtime
bool in[60][60];
int dir[4][2]={{-1,0},{1,0},{0,1},{0,-1}};
__int64 path[60][60];//total method
int cost[60][60];//segtime
int n;

bool inside(int x,int y)
{
    if(x<=n&&x>0&&y<=n&&y>0)
        return 1;
```

```
    return 0;
}

void bfs(void)
{
    memset(in,0,sizeof(in));
    int i,j;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            cost[i][j]=INF;
    cost[n][n]=map[n][n];
    in[n][n]=true;
    queue<node> q;
    node point;
    point.x=n;
    point.y=n;
    q.push(point);
    while(!q.empty())
    {
        node p=q.front();
        in[p.x][p.y]=false;
        q.pop();
        for(i=0;i<=3;i++)
        {
            int x,y;
            x=p.x+dir[i][0];
            y=p.y+dir[i][1];
            if(inside(x,y)&&cost[p.x][p.y]+map[x][y]<cost[x][y])
            {
```

```
                cost[x][y]=cost[p.x][p.y]+map[x][y];
                node a;
                a.x=x;
                a.y=y;
                if(!in[x][y])
                {
                    q.push(a);
                    in[x][y]=true;
                }
            }
        }
    }
}

__int64 dfs(int i,int j)
{
    if(i==n&&j==n)
        return 1LL;
    if(path[i][j]!=-1)
        return path[i][j];
    int a;
    path[i][j]=0LL;
    for(a=0;a<=3;a++)
    {
        int x=i+dir[a][0];
        int y=j+dir[a][1];
        if(inside(x,y)&&cost[x][y]<cost[i][j])
            path[i][j]+=dfs(x,y);
    }
```

```
    return path[i][j];
}

int main(void)
{
    while(scanf("%d",&n)!=EOF)
    {
        int i,j;
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
                scanf("%d",&map[i][j]);
        bfs();
        memset(path,-1,sizeof(path));
        dfs(1,1);//ms
        printf("%I64d\n",path[1][1]);
    }
    return 0;
}
```