# P3956 棋盘

## 题目描述

有一个m × m的棋盘，棋盘上每一个格子可能是红色、黄色或没有任何颜色的。你现在要从棋盘的最左上角走到棋盘的最右下角。

任何一个时刻，你所站在的位置必须是有颜色的（不能是无色的）， 你只能向上、 下、左、 右四个方向前进。当你从一个格子走向另一个格子时，如果两个格子的颜色相同，那你不需要花费金币；如果不同，则你需要花费 1 个金币。

另外， 你可以花费 2 个金币施展魔法让下一个无色格子暂时变为你指定的颜色。但这个魔法不能连续使用， 而且这个魔法的持续时间很短，也就是说，如果你使用了这个魔法，走到了这个暂时有颜色的格子上，你就不能继续使用魔法； 只有当你离开这个位置，走到一个本来就有颜色的格子上的时候，你才能继续使用这个魔法，而当你离开了这个位置（施展魔法使得变为有颜色的格子）时，这个格子恢复为无色。

现在你要从棋盘的最左上角，走到棋盘的最右下角，求花费的最少金币是多少？

## 输入输出格式

输入格式：

数据的第一行包含两个正整数 m， n，以一个空格分开，分别代表棋盘的大小，棋盘上有颜色的格子的数量。

接下来的 n 行，每行三个正整数 x， y， c， 分别表示坐标为（ x， y）的格子有颜色 c。

其中 c=1 代表黄色， c=0 代表红色。 相邻两个数之间用一个空格隔开。 棋盘左上角的坐标为（ 1, 1），右下角的坐标为（ m, m）。

棋盘上其余的格子都是无色。保证棋盘的左上角，也就是（ 1， 1） 一定是有颜色的。

输出格式：

输出一行，一个整数，表示花费的金币的最小值，如果无法到达，输出-1。

## 输入输出样例

输入样例#1：

5 7

1 1 0

1 2 0

2 2 1

3 3 1

3 4 0

4 4 1

5 5 0

输出样例#1：

8

输入样例#2：

5 5

1 1 0

1 2 0

2 2 1

3 3 1

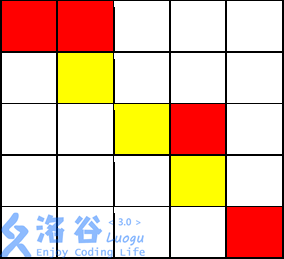
5 5 0

输出样例#2：

-1

## 说明

### 输入输出样例 1 说明



从（ 1， 1）开始，走到（ 1， 2）不花费金币

从（ 1， 2）向下走到（ 2， 2）花费 1 枚金币

从（ 2， 2）施展魔法，将（ 2， 3）变为黄色，花费 2 枚金币

从（ 2， 2）走到（ 2， 3）不花费金币

从（ 2， 3）走到（ 3， 3）不花费金币

从（ 3， 3）走到（ 3， 4）花费 1 枚金币

从（ 3， 4）走到（ 4， 4）花费 1 枚金币

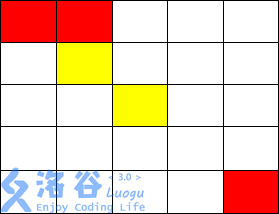
从（ 4， 4）施展魔法，将（ 4， 5）变为黄色，花费 2 枚金币，

从（ 4， 4）走到（ 4， 5）不花费金币

从（ 4， 5）走到（ 5， 5）花费 1 枚金币

共花费 8 枚金币。

### 输入输出样例 2 说明



从（ 1， 1）走到（ 1， 2），不花费金币

从（ 1， 2）走到（ 2， 2），花费 1 金币

施展魔法将（ 2， 3）变为黄色，并从（ 2， 2）走到（ 2， 3）花费 2 金币

从（ 2， 3）走到（ 3， 3）不花费金币

从（ 3， 3）只能施展魔法到达（ 3， 2），（ 2， 3），（ 3， 4），（ 4， 3）

而从以上四点均无法到达（ 5， 5），故无法到达终点，输出－1

### 数据规模与约定

对于 30%的数据， 1 ≤ m ≤ 5， 1 ≤ n ≤ 10。

对于 60%的数据， 1 ≤ m ≤ 20， 1 ≤ n ≤ 200。

对于 100%的数据， 1 ≤ m ≤ 100， 1 ≤ n ≤ 1,000。

代码：

#include<iostream>

#include<cstdio>

#define INF 10000000

using namespace std;

int n,m;

int map[110][110];

int fee[110][110];

int minm=INF;

int dx[]={1,0,-1,0};

int dy[]={0,-1,0,1};

void dfs(int x,int y,int color,bool used,int cost){

int nx,ny;

if (cost > minm) return;

if (x == m && y == m){

if (minm > cost) minm=cost;

return;

}

for (int i = 0;i < 4;++i){

nx=x+dx[i];

ny=y+dy[i];

if (nx > 0 && ny > 0 && nx <= m && ny <= m){

if (map[nx][ny] == color && fee[nx][ny] > fee[x][y]){

fee[nx][ny]=fee[x][y];

dfs(nx,ny,color,0,cost);

}

if (map[nx][ny] == -1){

if (used == 0){

if (color == 0) {

if (fee[nx][ny] > fee[x][y]+2){

fee[nx][ny]=fee[x][y]+2;

dfs(nx,ny,0,!used,cost+2);

}

if (fee[nx][ny] > fee[x][y]+3){

fee[nx][ny]=fee[x][y]+3;

dfs(nx,ny,1,!used,cost+2);

}

}

else {

if (fee[nx][ny] > fee[x][y]+3){

fee[nx][ny]=fee[x][y]+3;

dfs(nx,ny,0,!used,cost+3);

}

if (fee[nx][ny] > fee[x][y]+2){

fee[nx][ny]=fee[x][y]+2;

dfs(nx,ny,1,!used,cost+2);

}

}

}

}

if(map[nx][ny] != -1 && map[nx][ny] != color && fee[nx][ny] > fee[x][y]+1) {

fee[nx][ny]=fee[x][y]+1;

dfs(nx,ny,map[nx][ny],0,cost+1);

}

}

}

}

int main(){

scanf("%d%d",&m,&n);

int a,b,c;

for (int i = 1;i <= m;++i){

for (int j = 1;j <= m;++j){

map[i][j]=-1;

fee[i][j]=INF;

}

}

for (int i = 0;i < n;++i){

scanf("%d%d%d",&a,&b,&c);

map[a][b]=c;

}

fee[1][1]=0;

dfs(1,1,map[1][1],0,0);

if (minm == INF) printf("-1\n");

else printf("%d\n",minm);

return 0;

}

逃到海狸的宫殿（存盘文件名为slikar.pas/c/cpp）

【问题描述】

邪恶的国王凯克特丝有几个魔法水桶，魔法水桶可以让水泛滥淹掉一切。善良的皮特与三个小刺猬一定要在洪水来到前逃到海狸的宫殿中。

R行C列的字符构成了魔幻地图：其中“.”表示空地；“\*”表示魔法水桶所在位置；大写字母“X”表示岩石；大写字母“D”表示海狸的宫殿，而皮特与三个刺猬从发地用大写字母“S”表示。

每个时刻皮特与三个刺猬都有可以移动到四个方向（上、下、左、右）。每个时刻魔法水桶都让水淹没四周的空地。皮特和三个刺猬即不能走过洪水区，也不能走过岩石。当然洪水是不能淹掉海狸的宫殿的。

写一个程序，在所给的地图中找出皮特与三个小刺猬逃到海狸的宫殿的最短时间。注意的是在同一个时刻他们不能移到洪水也漫到的地方。

【输入格式】（输入文件名为slikar.in）

输入文件slikar.in中第一行：输入地图的R，C它们均小等于50；接下来的R行C列由字符“.” “\*” “X” “D” “S”组成。

【输出格式】（输出文件名为slikar.out）

输出文件slikar.in中仅一行：如果皮特与三个小刺猬能安全到达海狸的宫殿，则输出最短时间；否则输出大写单词“KAKTUS”。

【样例】

|  |  |
| --- | --- |
| slikar.in  3 6  D...\*.  .X.X..  ....S. | slikar.out  6 |

#include<iostream>

#include<fstream>

#include<algorithm>

#include<queue>

#define MAXN 1000

using namespace std;

char maze[MAXN][MAXN];

int t[MAXN][MAXN];

int c\_w[MAXN],c\_s[MAXN];

int sx,sy; //起始位置

int gx,gy; //结束位置

const int INF = 10000000;

queue<int> qx;

queue<int> qy;

queue<int> kx;

queue<int> ky;

int x[]={0,1,0,-1};

int y[]={1,0,-1,0};

int r,c;

//ifstream fin("slikar.in");

//ofstream fout("slikar.out");

int bfs(){

int dx,dy,nx,ny,k,count1=0,count2=0,i,j;

for (i = 0;i < r;i++)

for (j = 0;j < c;j++)

t[i][j]=INF;

t[sx][sy]=0;

while (t[gx][gy] == INF){

for (i = 0;i < c\_w[count1];i++){

nx=kx.front();

ny=ky.front();

kx.pop();

ky.pop();

for (k = 0;k < 4;k++){

dx=nx+x[k];

dy=ny+y[k];

if (maze[dx][dy] == '.' && dx >= 0

&& dx < r && dy >= 0 && dy < c){

c\_w[count1+1]++;

maze[dx][dy]='\*';

kx.push(dx);

ky.push(dy);

}

}

}

count1++;

for (k = 0;k < c\_s[count2];k++){

nx=qx.front();

ny=qy.front();

qx.pop();

qy.pop();

for (i = 0;i < 4;i++){

dx=nx+x[i];

dy=ny+y[i];

if (maze[dx][dy] == '.' && dx >= 0 && dx < r

&& dy >= 0 && dy < c && t[dx][dy] == INF){

qx.push(dx);

qy.push(dy);

t[dx][dy]=t[nx][ny]+1;

c\_s[count2+1]++;

}

if (maze[dx][dy] == 'D' && dx >= 0 && dx < r

&& dy >= 0 && dy < c && t[dx][dy] == INF){

qx.push(dx);

qy.push(dy);

t[dx][dy]=t[nx][ny]+1;

c\_s[count2+1]++;

}

}

}

if (c\_s[count2+1] == 0) break;

count2++;

}

return t[gx][gy];

}

int main(){

int i,j;

cin>>r>>c;

for (i = 0;i < r;i++)

for (j = 0;j < c;j++){

cin>>maze[i][j];

if (maze[i][j] == 'D'){

gx=i;

gy=j;

}

if (maze[i][j] == 'S'){

sx=i;

sy=j;

}

}

for (i = 0;i < r;i++)

for (j = 0;j < c;j++){

if (maze[i][j] == '\*') {

kx.push(i);

ky.push(j);

c\_w[0]++;

}

}

c\_s[0]=1;

qx.push(sx);

qy.push(sy);

int res=bfs();

if (res == INF) cout<<"KAKTUS"<<endl;

else cout<<res<<endl;

return 0;

}

# P1468 派对灯 Party Lamps

## 题目描述

在IOI98的节日宴会上，我们有N(10<=N<=100)盏彩色灯，他们分别从1到N被标上号码。 这些灯都连接到四个按钮：

按钮1：当按下此按钮，将改变所有的灯：本来亮着的灯就熄灭，本来是关着的灯被点亮。

按钮2：当按下此按钮，将改变所有奇数号的灯。

按钮3：当按下此按钮，将改变所有偶数号的灯。

按钮4：当按下此按钮，将改变所有序号是3\*K+1(K>=0)的灯。例如：1,4,7...

一个计数器C记录按钮被按下的次数。当宴会开始，所有的灯都亮着，此时计数器C为0。

你将得到计数器C(0<=C<=10000)上的数值和经过若干操作后某些灯的状态。写一个程序去找出所有灯最后可能的与所给出信息相符的状态，并且没有重复。

## 输入输出格式

输入格式：

不会有灯会在输入中出现两次。

第一行: N。

第二行: C最后显示的数值。

第三行: 最后亮着的灯,用一个空格分开，以-1为结束。

第四行: 最后关着的灯,用一个空格分开，以-1为结束。

输出格式：

每一行是所有灯可能的最后状态(没有重复)。每一行有N个字符，第1个字符表示1号灯，最后一个字符表示N号灯。0表示关闭，1表示亮着。这些行必须从小到大排列（看作是二进制数）。

如果没有可能的状态，则输出一行'IMPOSSIBLE'。

## 输入输出样例

输入样例#1：

10

1

-1

7 -1

输出样例#1：

0000000000

0101010101

0110110110

## 说明

在这个样例中，有三种可能的状态：

所有灯都关着

1,4,7,10号灯关着，2,3,5,6,8,9亮着。

1,3,5,7,9号灯关着，2, 4, 6, 8, 10亮着。

#include<iostream>

#define MAXN 100+20

using namespace std;

int n,c;

bool jud,ok;

int on[MAXN],off[MAXN];

int biao1[8][6]={0,0,0,0,0,0,

0,0,1,1,1,0,

0,1,0,1,0,1,

0,1,1,0,1,1,

1,0,0,1,0,0,

1,0,1,0,1,0,

1,1,0,0,0,1,

1,1,1,1,1,1};

int biao2[4][6]={0,0,0,0,0,0,

0,1,0,1,0,1,

0,1,1,0,1,1,

1,0,1,0,1,0};

int biao3[7][6]={0,0,0,0,0,0,

0,0,1,1,1,0,

0,1,0,1,0,1,

1,0,0,1,0,0,

1,0,1,0,1,0,

1,1,0,0,0,1,

1,1,1,1,1,1};

int main(){

int a;

int now;

int i,j;

cin>>n>>c;

while (cin>>a){

if (a == -1) break;

on[a]=1;

}

while (cin>>a){

if (a == -1) break;

off[a]=1;

}

if (c == 0) {

jud=1;

for (i = 1;i <= n;++i){

if (off[i] == 1) {

jud=0;

break;

}

}

if (jud) for (i = 1;i <= n;++i) cout<<"1";

else cout<<"IMPOSSIBLE";

cout<<"\n";

}

int p;

ok=0;

if (c == 1){

for (i = 0;i < 4;++i){

jud=1;

p=0;

for (j = 0;j < 6 && jud;++j){

now=j;

while (now <= n){

if (on[now+1] == 1 && !biao2[i][now%6]

|| off[now+1] == 1 && biao2[i][now%6]){

jud=0;

break;

}

now+=6;

}

}

if (jud) {

ok=1;

//cout<<i<<endl;

for (j = 0;j < n;++j){

cout<<biao2[i][p];

//cout<<p;

p++;

if (p == 6) p=0;

}

cout<<"\n";

}

}

if (!ok) cout<<"IMPOSSIBLE\n";

}

if (c == 2){

for (i = 0;i < 7;++i){

jud=1;

p=0;

for (j = 0;j < 6 && jud;++j){

now=j;

while (now <= n){

if (on[now+1] == 1 && !biao3[i][now%6]

|| off[now+1] == 1 && biao3[i][now%6]){

jud=0;

break;

}

now+=6;

}

}

if (jud) {

ok=1;

for (j = 0;j < n;++j){

cout<<biao3[i][p];

p++;

if (p == 6) p=0;

}

cout<<"\n";

}

}

if (!ok) cout<<"IMPOSSIBLE\n";

}

if (c > 2) {

for (i = 0;i < 8;++i){

jud=1;

p=0;

for (j = 0;j < 6 && jud;++j){

now=j;

while (now <= n){

if (on[now+1] == 1 && !biao1[i][now%6]

|| off[now+1] == 1 && biao1[i][now%6]){

jud=0;

break;

}

now+=6;

}

}

if (jud) {

ok=1;

for (j = 0;j < n;++j){

cout<<biao1[i][p];

p++;

if (p == 6) p=0;

}

cout<<"\n";

}

}

if (!ok) cout<<"IMPOSSIBLE\n";

}

return 0;

}