**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：11624: Fire!**

**解題者：王竣輝**

**解題日期：2019年6月13日**

**題意：**

JOE在一個著火的迷宮內，迷宮內的著火點至少有一個，JOE 跟著火點每分鐘可以以垂直或水平方向移動一格。

在JOE不會碰到火的情況下，求JOE安全走出迷宮的步數為何，如果無法則印出IMPOSSIBLE。

**題意範例：**

(stdin) (stdout)

2 3

4 4 IMPOSSIBLE

####

#JF#

#. .#

#. .#

3 3

###

# J .

# . F

**解法：**

分別對火跟人使用BFS演算法來走迷宮，並利用表格紀錄走到每一格所需的時間，若人走到該格的時間小於火走到該格的時間則表示人可以走到此格。

**解法範例：**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 2 | 1 |  |
|  | 3 | 2 |  |
|  | 4 | 3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 1 |  |  |
|  | 2 |  |  |
|  | 3 |  |  |

**討論：**

(1)迷宮的大小為 m\*n

所以進行一次BFS時間為O(m\*n)

對人與火共做兩次BFS，O(2\*m\*n) = O(m\*n)

**程式：**

#include <cstdio>

#include <queue>

#include <memory.h>

using std::queue;

using std::swap;

typedef struct point{

int x;

int y;

}pt;

int firePoint[1010][1010];

int JoePoint[1010][1010];

char map[1010][1010];

queue<pt> q;

int row,col;

pt fire,joe;

int dir[4][2] ={{1,0},{0,1},{-1,0},{0,-1}};

void clear(){

queue<pt> temp;

swap(q,temp);

}

void ans(){

pt nextFire,nextJoe,tempJoe,tempFire;

clear();

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

for(int j=0;j<col;j++){

if(map[i][j]=='F'){

fire.x = i;

fire.y = j;

q.push(fire);

firePoint[fire.x][fire.y] = 1;

}

}

}

while(!q.empty()){ /\* Fire point \*/

tempFire = q.front();

q.pop();

for(int i=0;i<4;i++){ /\* 4 direction \*/

nextFire.x = tempFire.x + dir[i][0];

nextFire.y = tempFire.y + dir[i][1];

if(nextFire.x<0 || nextFire.x>=row || nextFire.y<0 || nextFire.y >=col || map[nextFire.x][nextFire.y] == '#')

continue;

if(firePoint[nextFire.x][nextFire.y] ==0 ){

firePoint[nextFire.x][nextFire.y] = firePoint[tempFire.x][tempFire.y] +1;

q.push(nextFire);

}

}

}

/\* set joe's position \*/

q.push(joe);

JoePoint[joe.x][joe.y]=1;

while(!q.empty()){

tempJoe = q.front();

q.pop();

if(tempJoe.x ==0 || tempJoe.y ==0 || tempJoe.x == row-1 || tempJoe.y == col-1){

printf("%d\n", JoePoint[tempJoe.x][tempJoe.y]);

return;

}

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

nextJoe.x = tempJoe.x + dir[i][0];

nextJoe.y = tempJoe.y + dir[i][1];

if(nextJoe.x <0 || nextJoe.x >=row || nextJoe.y <0 || nextJoe.y >= col || map[nextJoe.x][nextJoe.y] == '#')

continue;

if(JoePoint[nextJoe.x][nextJoe.y] == 0 &&

(JoePoint[tempJoe.x][tempJoe.y]+1 < firePoint[nextJoe.x][nextJoe.y] || firePoint[nextJoe.x][nextJoe.y] ==0 )){

JoePoint[nextJoe.x][nextJoe.y] = JoePoint[tempJoe.x][tempJoe.y] +1;

q.push(nextJoe);

}

}

}

printf("IMPOSSIBLE\n");

}

int main(){

int tcase;

scanf("%d",&tcase);

while(tcase--){

scanf("%d %d",&row,&col);

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

scanf("%s",map[i]);

for(int j=0;j<col;j++){

if(map[i][j] == 'J'){

joe.x = i;

joe.y = j;

}

firePoint[i][j]=0;

JoePoint[i][j] = 0;

}

}

ans();

}

}