# 问题描述

1、 从键盘输入数据文件名；
2、 打开文件读入迷宫的行列值 m 和 n，以及迷宫的值 maze[m][n]；
3、 从文件中输入迷宫的入口和出口的坐（is，js）和（ie，je）；
4、 利用回溯法搜索迷宫各条通路；
5、 若迷宫有解，则输出迷宫及最短通路，最短通路用‘*’表示。

# 算法思路

采用回溯法。回溯法是一种不断试探且及时纠正错误的搜索方法。从迷宫入口点出发，向四周搜索，记下所有能到达的坐标点；若所有的方向均没有通路，则沿原路返回前一点；然后依次再从这些点出发，再记下所有一步能到达的坐标点；以此类推，直到到达迷宫的出口点为止，然后从出口点沿搜索路径回溯直至入口。这样，就找到了一条迷宫的最短路径，否则迷宫无路径。

# 算法描述

1. 表示迷宫的数据结构

设迷宫为 m 行 n 列，利用 maze[m][n]来表示一个迷宫。

maze[i][j]=0 或 1，其中：0 表示通路，1 表示不通。

当从某点向下试探时，中间点有 8 个方向可以试探，而 4 个角有 3 个方向，其它边缘点有 5 个方向。为使问题简单化，用 maze[m+2][n+2]来表示迷宫，而迷宫的四周的值全部为 1.这样做使问题简单化了，每个点的试探方向全部为 8，不用再判断当前点的试探方向有几个，同时与迷宫周围是墙壁这一实际问题相一致。

2. 试探方向

在上述表示迷宫的情况下，每个点有 8 个方向去试探。为了简化问题，方便地求出新点的坐标，将从正东开始沿顺时针进行的这 8 个方向的坐标增量放在一个结构数组 move[8]中。

```
typedef struct
{
int x,y;
}item;
item move[8]={{0,1},{1,1},{1,0},{1,-1},{0,-1},{-1,-1},{-1,0},{-1,1}};
/*坐标增量数组  move  的初始化*/
```

3. 栈的设计

栈中元素的设计如下：

```
typedef struct
{
int x , y , d ;      //  行列坐标及方向
}sq ;
```

压入栈的是一个由行、列、方向组成的三元组，方向是指从该点沿那个方向到达了下一点。同时，设定 front 和 rear 作为标志标明是否通过该点。

4. 如何防止重复到达某点，以避免发生死循环

方法是当到达某点(i,j)后使 maze[i][j]置-1，以便区别未到达过的点，起到防止走重复点的目的，算法结束前可恢复原迷宫。

## 源程序及驱动程序

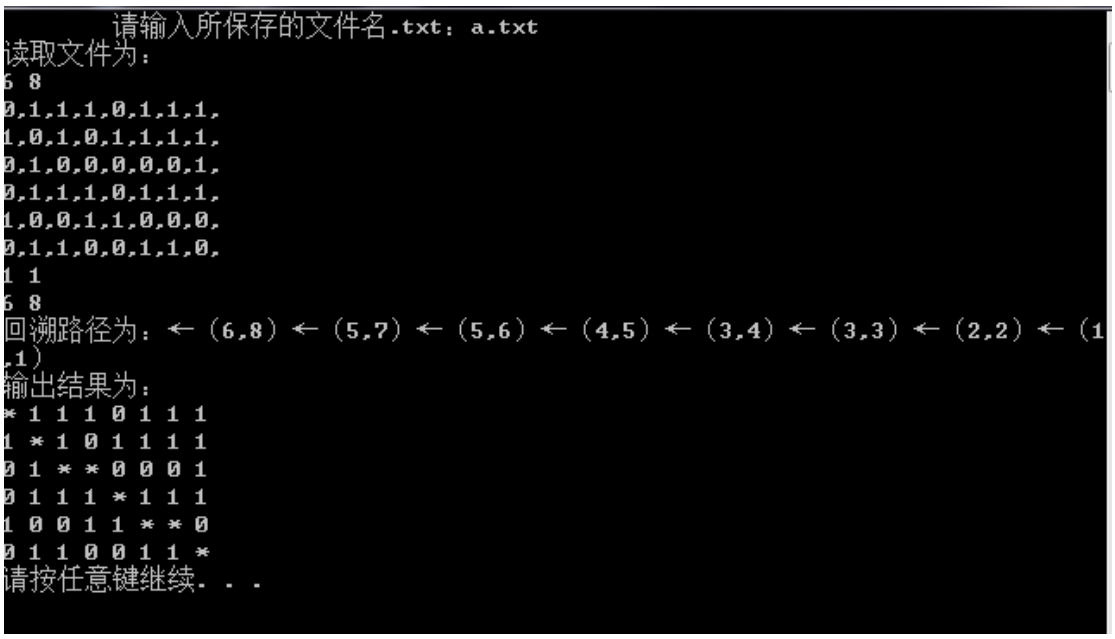
见附件

## 测试数据

```
6   8
0   1   1   1   0   1   1   1
1   0   1   0   1   1   1   1
0   1   0   0   0   0   0   1
0   1   1   1   0   1   1   1
1   0   0   1   1   0   0   0
0   1   1   0   0   1   1   0
1   1
6   8
```

上述数值存为 a.txt

运行结果正常




## 结果分析和结论

通过回溯法可以很好解决其它算法无法解决的重复试探工作。
为迷宫构造虚拟的墙可以避免边角值判断带来的麻烦。

## 心得体会

这次作业调试花费了很多很多时间以至于现在才提交作业。

C 与 C++对参数指针与引用的不同导致有些很简单的语句无法实现。