

牛客网-华为机试练习题 43

题目描述

定义一个二维数组N*M (其中 $2 \leq N \leq 10$; $2 \leq M \leq 10$) , 如 5×5 数组下所示:

```
int maze[5][5] = {  
  
    0, 1, 0, 0, 0,  
  
    0, 1, 0, 1, 0,  
  
    0, 0, 0, 0, 0,  
  
    0, 1, 1, 1, 0,  
  
    0, 0, 0, 1, 0,  
  
};
```

它表示一个迷宫, 其中的1表示墙壁, 0表示可以走的路, 只能横着走或竖着走, 不能斜着走, 要求编程找出从左上角到右下角的最短路线。入口点为[0,0], 既第一空格是可以走的路。

Input

一个 $N \times M$ 的二维数组, 表示一个迷宫。数据保证有唯一解, 不考虑有多解的情况, 即迷宫只有一条通道。

Output

左上角到右下角的最短路径, 格式如样例所示。

Sample Input

```
0 1 0 0 0  
0 1 0 1 0  
0 0 0 0 0  
0 1 1 1 0  
0 0 0 1 0
```

Sample Output

```
(0, 0)  
(1, 0)  
(2, 0)  
(2, 1)
```

(2, 2)

(2, 3)

(2, 4)

(3, 4)

(4, 4)

输入描述:

输入两个整数，分别表示二位数组的行数，列数。再输入相应的数组，其中的1表示墙壁，0表示可以走的路。数据保证有唯一解,不考虑有多解的情况，即迷宫只有一条通道。

输出描述:

左上角到右下角的最短路径，格式如样例所示。

示例1

输入

复制

```
5 5
0 1 0 0 0
0 1 0 1 0
0 0 0 0 0
0 1 1 1 0
0 0 0 1 0
```

输出

复制

```
(0,0)
(1,0)
(2,0)
(2,1)
(2,2)
(2,3)
(2,4)
(3,4)
(4,4)
```

解决代码:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str;
```

```

while ((str = br.readLine()) != null) {
    String[] rowAndColumn = str.split(" ");

    int N = Integer.valueOf(rowAndColumn[0]); //行
    int M = Integer.valueOf(rowAndColumn[1]); //列
    if (N >= 2 && N <= 10 && M >= 2 && M <= 10) {
        int[][] maza = new int[N][M];
        int row = 0;
        while (row < N) {
            str = br.readLine();
            String[] inputs = str.split(" ");
            if (inputs.length == M) {
                for (int i = 0; i < M; i++) {
                    maza[row][i] = Integer.valueOf(inputs[i]);
                }
            }
            row++;
        }

        findShortestPath(maza);
    }
}

public static void findShortestPath(int[][] maza) {
    //不考虑多解情况，迷宫只有一条通道
    //可以横着走或者竖着走
    int i = 0;
    int j = 0;
    while (i < maza.length) {
        while(j < maza[0].length) {
            if (maza[i][j] == 0) {
                printPath(i, j);
                j++; //右
            } else { //下
                j--;
                i++;
            }
        }
        i++;
        if(j == maza[0].length) j--; //下
    }
}

public static void printPath(int i, int j) {
    System.out.println("(" + i + "," + j + ")");
}
}

```

```

import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int row = sc.nextInt();
        int col = sc.nextInt();
        int[][] edges = new int[row][col];
        for(int i=0;i<row;i++){
            for(int j=0;j<col;j++){

```

```

        edges[i][j]=sc.nextInt();
    }
}
findShortestPath(edges);
}
public static void findShortestPath(int[][] edges){
    int i=0;
    int j=0;
    while(i<edges.length){
        while(j<edges[0].length){
            if(edges[i][j]==0){
                printPath(i,j);
                j++;
            }else{
                j--;
                i++;
            }
        }
        i++;
        if(j==edges[1].length) j--;
    }
}

public static void printPath(int i,int j){
    System.out.println("(" + i + ", " + j + ")");
}
}

```