矩阵中的路径\_剑指Offer\_12\_回溯法

# 剑指Offer12：矩阵中的路径

## 题目描述

\* 请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。

\* 路径可以从矩阵中的任意一个格子开始，每一步可以在矩阵中向左，向右，向上，向下移动一个格子。

\* 如果一条路径经过了矩阵中的某一个格子，则之后不能再次进入这个格子。

\* 例如 a b c e s f c s a d e e 这样的3 X 4 矩阵中包含一条字符串"bcced"的路径，

\* 但是矩阵中不包含"abcb"路径，因为字符串的第一个字符b占据了矩阵中的第一行第二个格子之后，

\* 路径不能再次进入该格子。

## 思路分析

\* 思路分析：利用回溯法。蛮力法的升级版。

\* 试着所有的路径走一遍，若不行，则退回上一步。

\* 具体思路：递归方法。核心方法：hasPathHelper；利用pathIndex记录袋查询路径的索引，一步一步地往前走，若某一步走不通，直接返回false；否则，继续走下一步。

\* 路径不能再次进入该格子，因此需要使用一个与matrix一一对应的布尔矩阵。

## Java代码

public boolean hasPath(char[] matrix, int rows, int cols, char[] str){

if(matrix == null||rows < 1||cols<1||str==null) return false;

if(str.length == 0) return true;

boolean[] hasVisisted = new boolean[rows\*cols];//标记当前格子是否走过

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

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

**if(hasPathHelper(matrix,rows,cols,str,new int[]{0},row,col,hasVisisted))**

**return true;**

}

}

return false;

}

/\*\*

\* 递归方法

\*/

public boolean hasPathHelper(char[] matrix, int rows, int cols, char[] path,int[] pathIndex,int row,int col,boolean[] hasVisisted ){

if(pathIndex[0] > path.length-1) return true;

if(row>=0&&row<rows&&col>=0&&col<cols&&matrix[row\*cols+col]==path[pathIndex[0]]&&!hasVisisted[row\*cols+col]) {//当前步正确

hasVisisted[row \* cols + col] = true;//标记当前方块走过了

//需要走下一步

pathIndex[0]++;

boolean result = hasPathHelper(matrix, rows, cols, path, pathIndex, row + 1, col, hasVisisted)//上

|| hasPathHelper(matrix, rows, cols, path, pathIndex, row, col + 1, hasVisisted)//左

|| hasPathHelper(matrix, rows, cols, path, pathIndex, row - 1, col, hasVisisted)//下

|| hasPathHelper(matrix, rows, cols, path, pathIndex, row, col - 1, hasVisisted);//右

if(result){

return true;//如果以后步骤走通了,走通了不需要退步，即pathIndex[0]没必要减1

}else{

**pathIndex[0]--;**

**hasVisisted[row\*cols+col] = false;**

return false;

}

}else {//如果当前步不对，则直接返回false即可

return false;

}

}