user: sakura

第22题——矩阵中的路径

给定一个 m x n 二维字符网格 board 和一个字符串单词 word 。如果 word 存在于网格中，返回 true ； 否则，返回 false 。

单词必须按照字母顺序，通过相邻的单元格内的字母构成，其中“相邻”单元格是那些水平相邻或垂直相邻的单元格。同一个单元格内的字母不允许被重复使用。

例如，在下面的 3×4 的矩阵中包含单词 "ABCCED"（单词中的字母已标出）。

示例 1：

```
输入：board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"
```

```
输出：true
```

示例 2：

```
输入：board = [["a","b"],["c","d"]], word = "abcd"
```

```
输出：false
```

提示：

1 <= board.length <= 200 1 <= board[i].length <= 200 board 和 word 仅由大小写英文字母组成

DFS（1）

核心思想：

```
先暴力遍历矩阵，寻找与给定字符串相等的字符进行递归：

        递归内容：

                if 字符匹配，word的下标+1

                        1.保存节点值

                        2.节点值改为  '/'  表示已经遍历过

                        3.接着递归上下左右四个方向，有一个为true就行，使用have接收该值
```

4.递归结束后 将结点的值复原同时返回have

else if word[len]=='\0' 表示递归完成，返回true

else 不匹配 返回true

递归结果为真，返回true

函数调用完成。

C语言

```
/*  参数列表：

board:矩阵
boardSize:横坐标
boardColSize:纵坐标
word: 带查找的字符串指针
i:横坐标
j:纵坐标
have:匹配，返回1
len:word的下标

*/
bool search(char **board,int boardSize,int boardColSize,char *word,int i,int j,int
have,int len)
{

    if(word[len]=='\0')
    {
        return true;//如果  word[len]==0 则表示word的内容已经全部找到，返回true即可
    }
    if(i>=0 && i<boardSize && j>=0 && j<boardColSize && board[i][j]==word[len])
    {
        len++;//如果字符匹配，则word的指针后移一位
    }
    else//不匹配则返回true
    {
        return false;
    }

    /*下面的代码只有字符匹配才会执行*/

    char tmp = board[i][j];//保存当前结点的值
    board[i][j] = '/';//将当前满足条件的字母值替换成一个非字母的符号，用来代表已经遍历过了

    /*  向上下左右四个方向递归，寻找word中相邻的字母  */
    have=search(board, boardSize, boardColSize, word, i-1, j,have,len)||
    search(board,boardSize,boardColSize,word,i+1,j,have,len)||
    search(board,boardSize,boardColSize,word,i, j-1,have,len)||
    search(board,boardSize,boardColSize,word, i, j+1,have,len);
```

```
    /* 四个方向有一个匹配即可 */

    board[i][j]=tmp;//回溯时将值复原
    return have;
}


bool exist(char** board, int boardSize, int* boardColSize, char* word)
{
    int i,j;
    for(i=0;i<boardSize;i++)
    {
        for(j=0;j<*boardColSize;j++)
        {
            /*寻找与word首个字符串匹配的结点     找到则传入参数开始递归
              初始参数：
               坐标：i,j、
               是否匹配的标志have:初值为0（匹配则返回true）
               word字符串的下标：初值为0
               */
            if(search(board,boardSize,*boardColSize,word, i,j,0,0))
            {
                return true;
            }
        }
    }
    return false;//矩阵中没有入口
}
```

```
class Solution {
public:
    bool exist(vector<vector<char>>& board, string word) {
        for(int i=0;i<board.size();i++){
            for(int j=0;j<board[0].size();j++){
                if(dfs(board,i,j,word,0))//寻找入口并调用递归
                    return true;
            }
        }
        return false;
    }
private:
    bool dfs(vector<vector<char>>& board,int i,int j,string word,int count){
        if(word[count]=='\0')
            return true; //遍历完了,返回true即可
        if(i>=0 && i<board.size() && j>=0 && j<board[0].size() && board[i]
[j]==word[count])
            count++;//字符匹配则返回查看下一个字符
        else
            return false;
        char temp=board[i][j];//保存该值
        board[i][j]='#';//标记已经访问过的字符
        bool ex=dfs(board,i-1,j,word,count)||
                dfs(board,i+1,j,word,count)||
```

```
                dfs(board,i,j+1,word,count)||
                dfs(board,i,j-1,word,count);
        board[i][j]=temp;//递归返回时将值还原
        return ex;
    }
};
```