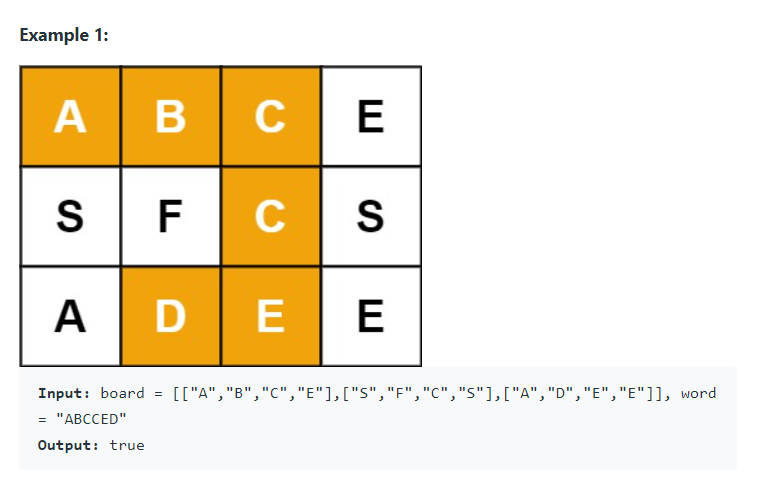
Given an m x n board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where "adjacent" cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.



思路：用DFS

假设我们找到了第一个match的字母，we need to search 4 characters around it., if char doesn't match word[index+1], then return false, if equal, remove current char to avoid repetition temporarily and call recursion dfs, if we reach end, then return true.

class Solution {

public boolean exist(char[][] board, String word) {

for (int i=0;i<board.length;i++){

for(int j=0;j<board[i].length;j++) //search all char, if match first, go dfs

if(board[i][j]==word.charAt(0)&&dfs(board,word,0,i,j))

return true;

}

return false;

}

public boolean dfs(char[][] board, String word,int index,int i,int j){

if(index==word.length()) return true; //reach the end, base case

if(i<0 ||i>=board.length||j<0||j>=board[i].length||board[i][j]!=word.charAt(index)) return false; //if out of bound or not match, return false

char temp=board[i][j]; //avoid repetition ,

board[i][j]=' ';

boolean result = dfs(board,word,index+1,i+1,j)||dfs(board,word,index+1,i-1,j)||dfs(board,word,index+1,i,j+1)||dfs(board,word,index+1,i,j-1) //or的特殊用法，他不一定要在判断局里面，它可以用在赋值里面，实际上四步都代表着最终结果因为他们是recursion最后return回来的;

board[i][j]=temp; //recover board

return result;

}

}