79. Word Search

题目描述：https://leetcode.com/problems/word-search/

> 给定一个二维数组代表一个字母棋盘，判断给定的字符串可不可在棋盘中找到，要求相邻字母在棋盘中必须也相邻（上下左右）棋盘中每个字母最多用一次
> 例如：
>
> 棋盘是：
>
> ```
> [['A','B','C','E'],
>  ['S','F','C','S'],
>  ['A','D','E','E']]
> ```
>
> 字符串：
>
> ```
> word = "ABCCED", -> returns true,
> word = "SEE", -> returns true,
> word = "ABCB", -> returns false.
> word = "BDF", -> return false.
> ```

解题思路：

利用回溯的方法判断每种可能
注意：为了排除那些重复使用某字母的情况，判断这个字母的上下左右时，将本字母设置为'\0'
注意：为了节省时间，判断某方向分支为true之后不再判断其他方向分支直接返回true

代码：

```
class Solution {
public:
    bool find(vector<vector<char>> &board, string word, int wordloc, int i, int j)
{
        if(wordloc == word.size() - 1)
            return board[i][j] == word.at(wordloc);
        if(board[i][j] == word.at(wordloc)){
            board[i][j] = 0;
            bool a = false, b = false, c = false, d = false;
            if(j < board[i].size() - 1)
                a = find(board, word, wordloc+1,i,j+1);
            if(!a && j > 0)
                b = find(board, word, wordloc+1,i,j-1);
            if(!a && !b && i < board.size() - 1)
                c = find(board, word, wordloc+1,i+1,j);
            if(!a && !b && !c && i > 0)
                d = find(board, word, wordloc+1,i-1,j);
            board[i][j] = word.at(wordloc);
            return a|b|c|d;
        }
        return false;
    }
    bool exist(vector<vector<char>>& board, string word) {
        for(int i = 0; i < board.size(); i++){
            for(int j = 0; j < board[i].size(); j++){
                if(find(board, word, 0, i, j))
                    return true;
            }
        }
        return false;
    }
};
```