1. Word Search

Medium

Given a 2D board and a word, find if the word exists in the grid.

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

**Example:**

board =  
[  
 ['A','B','C','E'],  
 ['S','F','C','S'],  
 ['A','D','E','E']  
]  
  
Given word = "ABCCED", return true.  
Given word = "SEE", return true.  
Given word = "ABCB", return false.

**Solution**

dfs search

int dy[4] = {1, -1, 0, 0};  
int dx[4] = {0, 0, 1, -1};  
class Solution {  
public:  
 bool exist(vector<vector<char>>& board, string word){  
 //int flag[256] = {0};  
 //bool visited[board.size()][board[0].size()] = {false};  
 vector<vector<bool>> visited(board.size(), vector<bool>(board[0].size(), false));  
 //for(char ch : word)flag[ch]++;  
 bool ans = false;  
 for(int i = 0; i < board.size(); ++i){  
 for(int j = 0; j < board[0].size(); ++j){  
 if(word[0] == board[i][j]){  
 int cnt = 0;  
 dfs(board, word, i, j, visited, int(word.size()), cnt, ans);  
 //printf("searching finish\n");  
 if(ans)return true;  
 }  
 }  
 }  
 return ans;  
 }  
 void dfs(vector<vector<char>>&board, string &word, int x, int y, vector<vector<bool>>&visited, int len, int &cnt, bool &ans){  
 visited[x][y] = true;  
 cnt++;  
 if(cnt == len){  
 ans = true;  
 return;  
 }  
 if(ans)return;  
 //printf("cur pos : %d, %d, char : %c cnt : %d len: %d\n", x, y, board[x][y], cnt, len);  
 for(int i = 0; i < 4; ++i){  
 int tmpX = x + dx[i], tmpY = y + dy[i];  
 if(tmpX < 0 || tmpX >= board.size() || tmpY < 0 || tmpY >= board[0].size())continue;  
 //if(flag[board[tmpX][tmpY]] == 0)continue;  
 if(visited[tmpX][tmpY] == true)continue;  
 if(board[tmpX][tmpY] != word[cnt])continue;  
 //printf("tmpX, tmpY : %d, %d\n", tmpX, tmpY);  
 dfs(board, word, tmpX, tmpY, visited, len, cnt, ans);  
 if(ans)return;  
 }  
 if(ans)return;  
 //flag[board[x][y]]++;  
 visited[x][y] = false;  
 cnt--;  
 }  
};