Question: https://leetcode.com/problems/word-search/

So we need to find a target string in a mxn matrix, now we first start traversing the matrix and if we find the first char of the target string, then we start checking its 4 adjacent neighbors for the next character, if found then we carry on the whole process until the whole string is found else we return false.

Now to avoid re considering of same elements in a single search, we set the visited elements as ‘0’ and later set them back to their original value once the particular search completes.(Backtracking)

Code:  
class Solution {

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

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

for(int j=0; j<board[0].length; j++){

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

return true;

}

}

}

return false;

}

public boolean helper(char[][] board, String word, int index, int r, int c){

if(index==word.length()){

return true;

}

else if(r<0 || r>=board.length){

return false;

}

else if(c<0 || c>=board[0].length){

return false;

}

else if(word.charAt(index)!=board[r][c]){

return false;

}

else if(board[r][c]=='0'){

return false;

}else{

char temp = board[r][c];

board[r][c] = '0';

if(helper(board, word, index+1, r+1, c)||

helper(board, word, index+1, r-1, c)||

helper(board, word, index+1, r, c+1)||

helper(board, word, index+1, r, c-1)){

board[r][c] = temp;

return true;

}

else{

board[r][c] = temp;

return false;

}

}

}

}

Github Link :<https://lnkd.in/ecwtJeaz>