Problema

79 leetcode.com

Word Search

Dado uma matriz 2D e uma palavra, encontre se ela pode ser formada.

A palavra é formada por uma sequência de letras de células adjacentes, onde as células “adjacentes” são as vizinhas horizontais e verticais. A mesmas células não podem ser utilizadas mais de uma vez.

Exemplo

board = [

['A','B','C','E'],

['S','F','C','S'],

['A','D','E','E']

]

word = "ABCCED" ->True.

word = "SEE" -> True.

word = "ABCB" -> False.

Dicas

Tente achar onde a palavra pode começar a ser formada

Use algum algoritmo de busca para vasculhar ao redor dessa primeira letra

Marque as células que você já passou temporariamente para evitar cair em loops

Solução

Itere sobre o board e veja se a célula atual existe e é a primeira letra da palavra desejada, se existir. Caso positivo, substitua a letra por ‘#’ para sabermos que essa célula já foi computada. Faça um Depht First Search para as células adjacentes que, de maneira recursiva, procuram pela palavra que retorna True se acharem em alguma das direções (esquerda, direita, cima, baixo) e False se caso contrário.

A complexidade será O(m∗n∗4\*\*s) uma vez que ele itera sobre a matriz n,m e, para cada letra de uma palavra de tamanho s existem 4 possibilidades para a DFS.

class Solution:

def DFS(self,row, col, curr, board, word):

if 0 <= row < len(board) and 0 <= col < len(board[0]) and board[row][col] == word[curr]:

temp = board[row][col]

board[row][col] = '#'

if curr == len(word)-1:

return True

result = self.DFS(row-1, col, curr+1, board, word) or self.DFS(row+1, col, curr+1, board, word) or self.DFS(row, col-1, curr+1, board, word) or self.DFS(row, col+1, curr+1, board, word)

board[row][col] = temp

return result

else:

return False

def exist(self, board: List[List[str]], word: str) -> bool:

for row in range(len(board)):

for col in range(len(board[0])):

check = self.DFS(row, col, 0, board, word)

if check:

return True

return False

Habilidades

Matizes, Recursão, Algoritmos de busca, Arrays, Backtracking

Erros Comuns

A maioria dos entrevistados não pensou, inicialmente, sobre alguma forma de impedir que o algoritmo não passasse mais de uma vez pela mesma letra na matriz. Outro erro comum foi não pensar em casos que a primeira letra da palavra aparece mais de uma vez, assim se durante a varredura da matriz fosse achada a letra inicial, porém não fosse possível formar a palavra a partir dela mas sim de uma outra em outro lugar na matriz, o programa não retornaria o resultado correto.

Aplicar o mesmo problema com pessoas diferentes ajudou a perceber a linha de raciocínio das pessoas, algumas começaram já traçando o plano que fariam, outras pensaram por blocos e iam implementando outras iam pensando enquanto programavam. Outra experiência interessante foi fazer a entrevista com o Rafael, que não faz parte da disciplina, pois ele não teve as mesmas experiencias de resolver problemas que os outros, isso ficou claro pois a primeira solução que ele criou foi iterativa e não recursiva como as dos alunos da disciplina. Mas em geral, todos os entrevistados pensaram em algoritmos que se assemelham ao Depth-first search, começando pela primeira letra da palavra e entrando em uma recursão para cada letra em volta no tabuleiro.