75 Curated Leetcode Problems & Solutions Bank

The problems on this page are widely accepted to be sufficient to pass any junior-level coding interview:

You can find all solutions here:

GitHub - tuomaskivioja/Leetcode75

Contribute to tuomaskivioja/Leetcode75 development by creating an account on GitHub.

https://github.com/tuomaskivioja/Leetcode75

tuomaskivioja/
Leetcode75

1 Contributor · 0 Issues · 0 Stars · 1 Fork

▼ Array

- [ ] Two Sum
- [ ] Best Time to Buy and Sell Stock
- [ ] Contains Duplicate
- [ ] Product of Array Except Self

- [ ] Maximum Subarray
- [ ] Maximum Product Subarray
- [ ] Find Minimum in Rotated Sorted Array
- [ ] Search in Rotated Sorted Array
- [ ] 3 Sum
- [ ] Container With Most Water

▼ Binary

- [ ] Sum of Two Integers
- [ ] Number of 1 Bits
- [ ] Counting Bits
- [ ] Missing Number
- [ ] Reverse Bits

▼ Dynamic Programming

- [ ] Climbing Stairs
- [ ] Coin Change
- [ ] Longest Increasing Subsequence
- [ ] Word Break Problem
- [ ] Combination Sum
- [ ] House Robber

▼ Graphs

- [ ] Clone Graph
- [ ] Course Schedule
- [ ] Pacific Atlantic Water Flow
- [ ] Number of Islands
- [ ] Longest Consecutive Sequence
- [ ] Alien Dictionary (Leetcode Premium)

- [ ] Graph Valid Tree (Leetcode Premium)
- [ ] Number of Connected Components in an Undirected Graph (Leetcode Premium)

▼ Interval

- [ ] Insert Interval
- [ ] Merge Intervals
- [ ] Non-overlapping Intervals
- [ ] Merge Intervals
- [ ] Non-overlapping Intervals
- [ ] Meeting Rooms (Leetcode Premium)
- [ ] Meeting Rooms II (Leetcode Premium)

▼ Linked List

- [ ] Reverse a Linked List (Leetcode)
- [ ] Detect Cycle in a Linked List (Leetcode)
- [ ] Merge Two Sorted Lists (Leetcode)
- [ ] Merge K Sorted Lists (Leetcode)
- [ ] Detect Cycle in a Linked List (Leetcode)
- [ ] Remove Nth Node From End of List (Leetcode)
- [ ] Reorder List (Leetcode)

▼ Matrix

- [ ] Set Matrix Zeroes

▼ Solution

Set Matrix Zeroes

Given an `m x n` matrix. If an element is `0`, set its entire row and column to `0`. Do it in-place.

Example 1:

```
Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]
```

Example 2:

```
Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]
```

Solution

```
class Solution:
    def setZeroes(self, matrix: List[List[int]]) -> Non
e:
        """
        Do not return anything, modify matrix in-place i
nstead.
        """
        m, n = len(matrix), len(matrix[0])
        rows, cols = set(), set()
        for i in range(m):
            for j in range(n):
                if matrix[i][j] == 0:
                    rows.add(i)
                    cols.add(j)
        for i in range(m):
            for j in range(n):
                if i in rows or j in cols:
                    matrix[i][j] = 0
```

Time Complexity: `O(m*n)`, where `m` is the number of rows and `n` is the number of columns.

Space Complexity: `O(m + n)`.

- [ ] Spiral Matrix

▼ Solution

Spiral Matrix

Given an `m x n` matrix, return all elements of the matrix in spiral order.

Example 1:

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [1,2,3,6,9,8,7,4,5]
```

Example 2:

```
Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```

Solution

```
class Solution:
    def spiralOrder(self, matrix: List[List[int]]) -> Li
st[int]:
        if not matrix:
            return []
        m, n = len(matrix), len(matrix[0])
        top, bottom, left, right = 0, m-1, 0, n-1
        res = []
        while top <= bottom and left <= right:
            for j in range(left, right+1):
                res.append(matrix[top][j])
            for i in range(top+1, bottom+1):
                res.append(matrix[i][right])
            if top < bottom and left < right:
                for j in range(right-1, left, -1):
                    res.append(matrix[bottom][j])
                for i in range(bottom, top, -1):
                    res.append(matrix[i][left])
            top += 1
```

```
            bottom -= 1
            left += 1
            right -= 1
        return res
```

Time Complexity: `O(m*n)`, where `m` is the number of rows and `n` is the number of columns.

Space Complexity: `O(1)`.

- [ ] Rotate Image

▼ Solution

Rotate Image

You are given an `n x n` 2D `matrix` representing an image, rotate the image by 90 degrees (clockwise).

You have to rotate the image in-place, which means you have to modify the input `2D matrix` directly. DO NOT allocate another 2D matrix and do the rotation.

Example 1:

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6

Continuing from the previous message...

Example 1:

```python
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
```
```

Example 2:

```
Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,1
4,12,16]]
Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]
```

Solution

```
class Solution:
    def rotate(self, matrix: List[List[int]]) -> None:
        """
        Do not return anything, modify matrix in-place i
nstead.
        """
        n = len(matrix)
        # Transpose the matrix
        for i in range(n):
            for j in range(i, n):
                matrix[i][j], matrix[j][i] = matrix[j]
[i], matrix[i][j]
        # Reverse each row
        for i in range(n):
            matrix[i] = matrix[i][::-1]
```

Time Complexity: `O(n^2)`, where `n` is the length of the matrix.

Space Complexity: `O(1)`.

- [ ] Word Search

▼ Solution

Word Search

Given an `m x n` grid of characters `board` and a string `word`, return `true` if `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.

Example 1:

```
Input: board = [["A","B","C","E"],["S","F","C","S"],
["A","D","E","E"]], word = "ABCCED"
Output: true
```

Example 2:

```
Input: board = [["A","B","C","E"],["S","F","C","S"],
["A","D","E","E"]], word = "SEE"
Output: true
```

Solution

```
class Solution:
    def exist(self, board: List[List[str]], word: str) -
> bool:
        def dfs(i, j, k):
            if not 0 <= i < len(board) or not 0 <= j < l
en(board[0]) or board[i][j] != word[k]:
                return False
            if k == len(word) - 1:
                return True
            temp, board[i][j] = board[i][j], '/'
            res = dfs(i+1, j, k+1) or dfs(i-1, j, k+1) o
r dfs(i, j+1, k+1) or dfs(i, j-1, k+1)
            board[i][j] = temp
            return res

        for i in range(len(board)):
            for j in range(len(board[0])):
                if dfs(i, j, 0):
                    return True
        return False
```

Time Complexity: `O(m*n*4^k)`, where `m` is the number of rows, `n` is the number of columns, and `k` is the length of the word.

Space Complexity: `O(k)`, where `k` is the length of the word.

▼ String

- [ ] Longest Substring Without Repeating Characters
- [ ] Longest Repeating Character Replacement
- [ ] Minimum Window Substring
- [ ] Longest Repeating Character Replacement
- [ ] Minimum Window Substring
- [ ] Valid Anagram
- [ ] Group Anagrams
- [ ] Valid Parentheses
- [ ] Valid Palindrome
- [ ] Longest Palindromic Substring
- [ ] Palindromic Substrings
- [ ] Encode and Decode Strings (Leetcode Premium)

▼ Tree

- [ ] Maximum Depth of Binary Tree
- [ ] Same Tree
- [ ] Invert/Flip Binary Tree
- [ ] Same Tree
- [ ] Invert/Flip Binary Tree
- [ ] Binary Tree Maximum Path Sum
- [ ] Binary Tree Level Order Traversal
- [ ] Serialize and Deserialize Binary Tree
- [ ] Subtree of Another Tree

- [ ] Construct Binary Tree from Preorder and Inorder Traversal
- [ ] Validate Binary Search Tree
- [ ] Kth Smallest Element in a BST
- [ ] Lowest Common Ancestor of a Binary Search Tree
- [ ] Add and Search Word - Data structure design
- [ ] Word Search II

▼ Heap

- [ ] Merge K-Sorted Lists
- [ ] Top K Frequent Elements
- [ ] Find Median from Data Stream