

# DESIGN AND ANALYSIS OF ALGORITHMS

Name: P. Vamshi

Roll No: 2211CS020411

Section: AIML-Epsilon

## 1. Find the Index of the First occurrence in a String

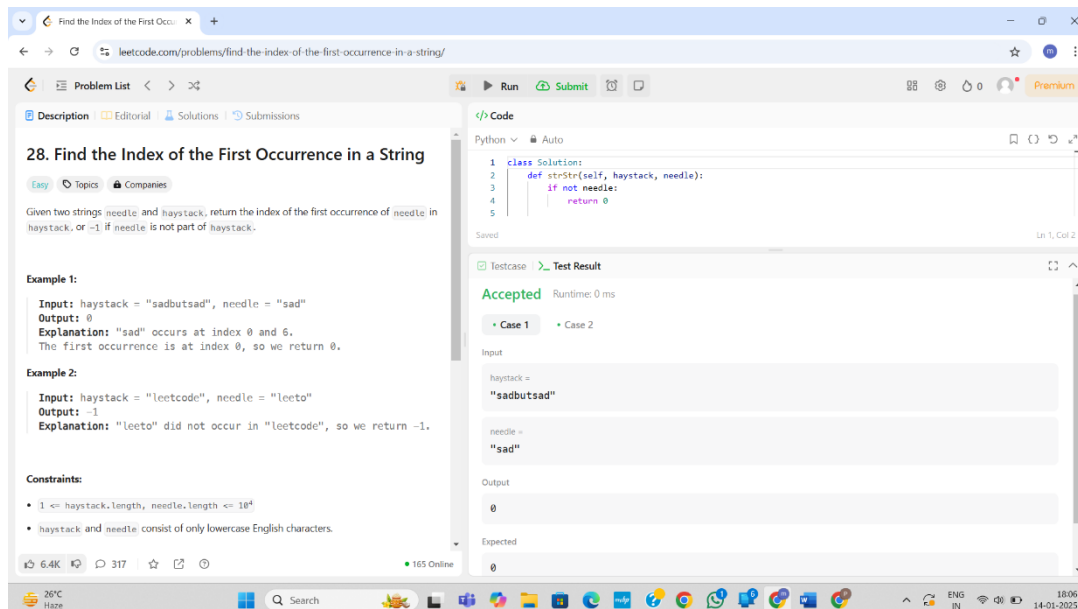
<https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/>

### CODE:

class Solution:

```
def strStr(self, haystack: str, needle: str) -> int:
    if not needle:
        return 0
    haystack_len = len(haystack)
    needle_len = len(needle)
    for i in range(haystack_len - needle_len + 1):
        if haystack[i:i + needle_len] == needle:
            return i
    return -1
```

### Results:



## 2. Bitwise and of Number Range e

<https://leetcode.com/problems/bitwise-and-of-numbers-range/>

### CODE:

class Solution:

```
def rangeBitwiseAnd(self, left, right):
```

```
    shift = 0
```

```
    while left < right:
```

```
        left >>= 1
```

```
        right >>= 1
```

```
        shift += 1
```

```
    return left << shift
```

### Results:

The screenshot displays the LeetCode problem page for '201. Bitwise AND of Numbers Range'. The problem description states: 'Given two integers left and right that represent the range [left, right], return the bitwise AND of all numbers in this range, inclusive.' Three examples are provided: Example 1 (left=5, right=7, output=4), Example 2 (left=0, right=0, output=0), and Example 3 (left=1, right=2147483647, output=0). The constraints are: 0 ≤ left ≤ right ≤ 2<sup>31</sup> - 1. The solution code is shown in Python, implementing a loop to shift bits until left and right are equal. The test results show 'Accepted' with a runtime of 0 ms for Case 1, where left=5 and right=7, resulting in an output of 4.

### 3) Square Root

<https://leetcode.com/problems/sqrtx/>

#### CODE:

class Solution:

```
def mySqrt(self, x):
```

```
    if x < 2:
```

```
        return x # For 0 and 1, return x directly
```

```
    low, high = 0, x // 2 + 1
```

```
    result = 0
```

```
    while low <= high:
```

```
        mid = (low + high) // 2
```

```
        if mid * mid == x:
```

```
            return mid # Perfect square root found
```

```
        elif mid * mid < x:
```

```
            result = mid # Update result and move to the right
```

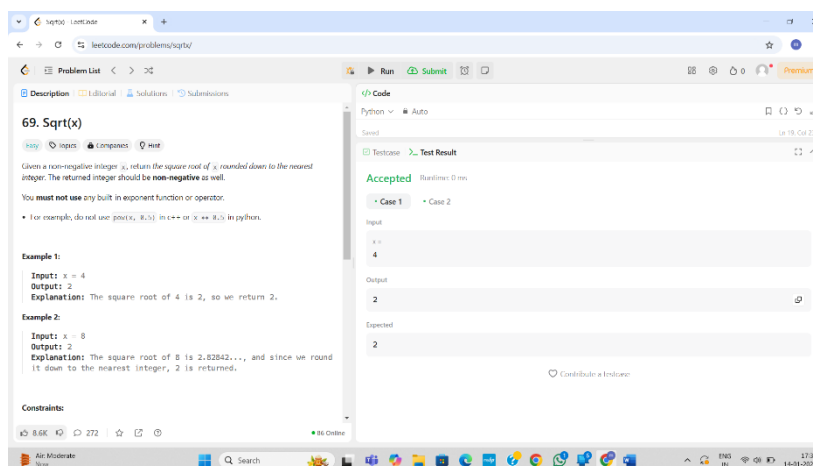
```
            low = mid + 1
```

```
        else:
```

```
            high = mid - 1 # Move to the left
```

```
    return result
```

#### Result:



## 4. largest-number

<https://leetcode.com/problems/largest-number/>

### CODE:

```
from functools import cmp_to_key
class Solution:
    def largestNumber(self, nums):
        nums = list(map(str, nums))
        def compare(x, y):
            if x + y > y + x:
                return -1
            elif x + y < y + x:
                return 1
            else:
                return 0
        nums.sort(key=cmp_to_key(compare))
        result = ".join(nums)
        return '0' if result[0] == '0' else result
```

### Result:

The screenshot displays the LeetCode interface for the 'Largest Number' problem (179). The problem description states: 'Given a list of non-negative integers `nums`, arrange them such that they form the largest number and return it. Since the result may be very large, so you need to return a string instead of an integer.'

**Example 1:**  
Input: `nums = [10,2]`  
Output: `"210"`

**Example 2:**  
Input: `nums = [3,30,34,5,9]`  
Output: `"9534330"`

**Constraints:**

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 109`

The 'Code' tab on the right shows the Python solution using `functools.cmp_to_key` and a custom comparator. The 'Test Result' tab shows the solution is 'Accepted' with a runtime of 0 ms. Two test cases are listed, both passing with the expected output `"210"`.

## 5. Valid Parentheses

<https://leetcode.com/problems/valid-parentheses/description/>

### CODE:

```
class Solution:
    def isValid(self, s):
        stack = []
        bracket_map = {'(': ')', '{': '}', '[': ']'}
        for char in s:
            if char in bracket_map:
                top_element = stack.pop() if stack else '#'
                if bracket_map[char] != top_element:
                    return False
            else:
                stack.append(char)
        return not stack
```

### Result:

The screenshot displays the LeetCode interface for the 'Valid Parentheses' problem (Problem 20). The left sidebar shows the problem description, which states: 'Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if: 1. Open brackets must be closed by the same type of brackets. 2. Open brackets must be closed in the correct order. 3. Every close bracket has a corresponding open bracket of the same type.' Examples provided include 'Input: s = "()", Output: true' and 'Input: s = "()[]{}", Output: true'. The main area shows the Python solution code: 

```
def isValid(self, s):
    stack = []
    bracket_map = {'(': ')', '{': '}', '[': ']'}
    for char in s:
        if char in bracket_map:
            top_element = stack.pop() if stack else '#'
            if bracket_map[char] != top_element:
                return False
        else:
            stack.append(char)
    return not stack
```

 The right sidebar shows the 'Testcase' section with 'Accepted' status and 'Runtime: 0 ms'. Below this, there are four test cases, with 'Case 1' selected, showing input 's = "()", Output: true, and Expected: true. The bottom of the screen shows the Windows taskbar with various application icons and the system clock indicating 18:15 on 14-01-2025.

## 6. merge-two-sorted-lists

<https://leetcode.com/problems/merge-two-sorted-lists/>

### CODE:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def mergeTwoLists(self, list1, list2):
        dummy = ListNode()
        current = dummy
        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next
        if list1:
            current.next = list1
        elif list2:
            current.next = list2
        return dummy.next
```

### Result:

The screenshot displays the LeetCode interface for the 'Merge Two Sorted Lists' problem. The problem description is on the left, and the code editor is on the right. The code is a Python solution that uses a dummy node and a while loop to merge two sorted linked lists. The test result shows 'Accepted' with a runtime of 0 ms.

**Problem Description:**

21. Merge Two Sorted Lists

Easy

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

**Example 1:**

list1 = [1, 2, 4]  
list2 = [1, 3, 4]  
Merged list = [1, 1, 2, 3, 4, 4]

**Code:**

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
class Solution:
    def mergeTwoLists(self, list1, list2):
        dummy = ListNode()
        current = dummy
        while list1 and list2:
            if list1.val < list2.val:
                current.next = list1
                list1 = list1.next
            else:
                current.next = list2
                list2 = list2.next
            current = current.next
        if list1:
            current.next = list1
        elif list2:
            current.next = list2
        return dummy.next
```

**Test Result:**

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input:

list1 = [1, 2, 4]  
list2 = [1, 3, 4]

Output:

[1, 1, 2, 3, 4, 4]

Expected:

[1, 1, 2, 3, 4, 4]

## 7. remove-duplicates-from-sorted-list

<https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/>

### CODE:

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def deleteDuplicates(self, head):
        current = head

        while current and current.next:
            if current.val == current.next.val:
                # Skip the next node if it's a duplicate
                current.next = current.next.next
            else:
                current = current.next

        return head
```

### Result:

The screenshot displays the LeetCode problem page for "83. Remove Duplicates from Sorted List". The problem description is: "Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well." Example 1 shows a linked list with nodes 1, 1, and 2, which is transformed into a linked list with nodes 1 and 2. The input is head = [1, 1, 2] and the output is [1, 2]. The code editor on the right shows the Python solution. The test result shows "Accepted" with a runtime of 0 ms for Case 1.

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def deleteDuplicates(self, head):
        current = head

        while current and current.next:
            if current.val == current.next.val:
                # Skip the next node if it's a duplicate
                current.next = current.next.next
            else:
                current = current.next

        return head
```

Testcase: Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head = [1, 1, 2]

Output

[1, 2]

Expected

[1, 2]

## 8. find-peak-element

<https://leetcode.com/problems/find-peak-element/description/>

### CODE:

class Solution:

```
def findPeakElement(self, nums):
    left, right = 0, len(nums) - 1
    while left < right:
        mid = (left + right) // 2
        if nums[mid] < nums[mid + 1]:
            left = mid + 1
        else:
            right = mid
    return left
```

### Result:

The screenshot displays the LeetCode interface for the 'Find Peak Element' problem (162). The problem description states: 'A peak element is an element that is strictly greater than its neighbors. Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks. You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array. You must write an algorithm that runs in O(log n) time.'

**Example 1:**  
Input: nums = [1,2,3,1]  
Output: 2  
Explanation: 3 is a peak element and your function should return the index number 2.

**Example 2:**  
Input: nums = [1,2,1,3,5,6,4]  
Output: 5  
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

The solution code is written in Python and uses a binary search approach:

```
def findPeakElement(self, nums):
    left, right = 0, len(nums) - 1
    while left < right:
        mid = (left + right) // 2
        if nums[mid] < nums[mid + 1]:
            left = mid + 1
        else:
            right = mid
    return left
```

The test results show 'Accepted' with a runtime of 0 ms. The input is [1,2,3,1] and the output is 2, which matches the expected result.



## 9. Binary-tree-inorder-traversal

<https://leetcode.com/problems/binary-tree-inorder-traversal/description/>

### CODE:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

```
class Solution:
    def inorderTraversal(self, root):
        result = []
        def inorder(node):
            if node:
                inorder(node.left)
                result.append(node.val)
                inorder(node.right)

        inorder(root)
        return result
```

### Result:

The screenshot shows the LeetCode web interface for the problem '94. Binary Tree Inorder Traversal'. The left sidebar contains the problem description, which states: 'Given the root of a binary tree, return the inorder traversal of its nodes' values.' Example 1 is provided with input root = [1, null, 2, 3] and output [1, 3, 2]. A diagram of the binary tree is shown: root node 1 has a left child 3 and a right child 2. The right sidebar shows the code editor with the Python solution, the test case input [1, null, 2, 3], and the test result 'Accepted' with runtime 0 ms.

## 10. N-queens

<https://leetcode.com/problems/n-queens/description/>

### CODE:

class Solution:

```
def solveNQueens(self, n):
```

```
    result = []
```

```
    def backtrack(row, cols, diag1, diag2, current_board):
```

```
        if row == n:
```

```
            result.append(["".join(row) for row in current_board])
```

```
            return
```

```
        for col in range(n):
```

```
            if col in cols or (row - col) in diag1 or (row + col) in diag2:
```

```
                continue
```

```
            cols.add(col)
```

```
            diag1.add(row - col)
```

```
            diag2.add(row + col)
```

```
            current_board[row][col] = 'Q'
```

```
            backtrack(row + 1, cols, diag1, diag2, current_board)
```

```
            cols.remove(col)
```

```
            diag1.remove(row - col)
```

```
            diag2.remove(row + col)
```

```
            current_board[row][col] = '.'
```

```
    current_board = [['.' for _ in range(n)] for _ in range(n)]
```

```
    cols = set()
```

```
    diag1 = set()
```

```
    diag2 = set()
```

```
    backtrack(0, cols, diag1, diag2, current_board)
```

```
    return result
```

### Result:

The screenshot displays the LeetCode '51. N-Queens' problem page. On the left, the problem description is visible, including the goal of placing  $n$  queens on an  $n \times n$  chessboard such that no two queens share the same row, column, or diagonal. Two example chessboard configurations are shown for  $n=4$ . The main area on the right shows the 'Code' editor with a Python solution using backtracking. The solution defines a `backtrack` function that explores possible queen placements row by row, using sets to track occupied columns and diagonals. The 'Test Result' panel at the bottom right shows the solution passed all test cases, with a runtime of 0 ms for the provided input `n = 4`. The expected output is a list of two distinct board configurations for  $n=4$ .