

RAMAIAH INSTITUTE OF TECHNOLOGY

MSRIT NAGAR, BENGALURU, 560054



A Report on

[Design and Analysis of Algorithms]

*Submitted in partial fulfilment of the OTHER COMPONENT requirements as a part of the DAA
LAB subject with code IS46 for the IV Semester of degree of **Bachelor of Engineering in
Information Science and Engineering***

Submitted by

Candidate Names

PRATHIBHA A

1MS23IS409

Under the Guidance of

Faculty In charge

Shivanandha

Dept. of ISE

Department of Information Science and Engineering

Ramaiah Institute of Technology

2023– 2024

1.Dynamic programming

1.1 longest palindrome

substringCode:

class Solution:

```
def longestPalindrome(self, s) :
```

```
    if len(s) <= 1:
```

```
        return s
```

```
    Max_Len=1
```

```
    Max_Str=s[0]
```

```
    for i in range(len(s)-1):
```

```
        for j in range(i+1,len(s)):
```

```
            if j-i+1 > Max_Len and s[i:j+1] == s[i:j+1][::-1]:
```

```
                Max_Len = j-i+1
```

```
                Max_Str = s[i:j+1]
```

```
    return Max_Str
```

OUTPUT:

The screenshot displays the LeetCode interface for the problem "5. Longest Palindromic Substring". On the left, the problem description states: "Given a string s , return the longest palindromic substring in s ." It includes two examples: Example 1 with input $s = \text{"babad"}$ and output "bab" , and Example 2 with input $s = \text{"cbbd"}$ and output "bb" . Constraints specify that $1 \leq s.length \leq 1000$ and s consists of only digits and English letters. On the right, the code editor shows a Python solution. The test results indicate that the solution is "Accepted" with a runtime of 46 ms. The input field shows $s = \text{"babad"}$, the output field shows "bab" , and the expected output field also shows "bab" .

1.2 ones and zeros

Code:

class Solution:

```
def findMaxForm(self, strs, m, n):
```

```
    xy = [[s.count("0"), s.count("1")] for s in strs]
```

```
    def dp(mm, nn, kk):
```

```
        if mm < 0 or nn < 0: return -float("inf")
```

```
        if kk == len(strs): return 0
```

```
        x, y = xy[kk]
```

```
        return max(1 + dp(mm-x, nn-y, kk + 1), dp(mm, nn, kk + 1))
```

```
    return dp(m, n, 0)
```

OUTPUT:

The screenshot shows the LeetCode interface for problem 474, "Ones and Zeros". The problem description states: "You are given an array of binary strings `strs` and two integers `m` and `n`. Return the size of the largest subset of `strs` such that there are at most `m` 0's and `n` 1's in the subset. A set `x` is a subset of a set `y` if all elements of `x` are also elements of `y`."

Example 1:
Input: `strs = ["10", "0001", "111001", "1", "0"]`, `m = 5`, `n = 3`
Output: 4
Explanation: The largest subset with at most 5 0's and 3 1's is {"10", "0001", "1", "0"}, so the answer is 4. Other valid but smaller subsets include {"0001", "1"} and {"10", "1", "0"}.

Example 2:
Input: `strs = ["10", "0001", "111001", "1", "0"]`, `m = 5`, `n = 3`
Output: 4

The right side of the screenshot shows the test case input and output fields. The input is `strs = ["10", "0001", "111001", "1", "0"]`, `m = 5`, and `n = 3`. The output field is empty.

2. divide and conquer

2.1 median of two sorted

arrayCode:

class Solution:

```
def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:
```

```
    len1 = len(nums1)
```

```
    len2 = len(nums2)
```

```
    total_len = len1 + len2
```

```
    lst = nums1 + nums2
```

```
    lst.sort()
```

```
    if total_len % 2 == 0:
```

```
        median = (lst[total_len // 2 - 1] + lst[total_len // 2]) / 2
```

```
    else:
```

```
        median = lst[total_len // 2]
```

```
    return median
```

OUTPUT:

The screenshot displays the LeetCode interface for the 'Median of Two Sorted Arrays' problem. The problem description states: 'Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).' Two examples are provided: Example 1 with nums1 = [1, 3] and nums2 = [2], resulting in a median of 2.00000; and Example 2 with nums1 = [1, 2] and nums2 = [3, 4], resulting in a median of 2.50000. The 'Testcase' section shows the input arrays and the expected output. The 'Divide and Conquer' solution approach is highlighted.

2.2 convert sorted list to binary search

Code:

```
class Solution:
```

```
    def sortedListToBST(self, head: Optional[ListNode]) -> Optional[TreeNode]:
```

```
        def middle(curr = None):
```

```
            if curr:
```

```
                prev = slow = fast =
```

```
                curr
```

```
                while fast and
```

```
                    fast.next:
```

```
                        prev = slow
```

```
                        slow =
```

```
                        slow.next
```

```
                        fast = fast.next.next
```

```
            return prev
```

```
    def helper(curr=None):
```

```
        if curr:
```

```
            pre_mid = middle(curr)
```

```
            if not pre_mid.next:
```

```
                return TreeNode(pre_mid.val)
```

```
            parent = pre_mid.next
```

```
            pre_mid.next = None
```

```
            root = TreeNode(parent.val)
```

```
            root.right = helper(parent.next)
```

```
            root.left = helper(curr)
```

```
            return root
```

```
    return helper(head)
```

OUTPUT:

The screenshot shows the LeetCode interface for the problem "109. Convert Sorted List to Binary Search Tree". The problem description states: "Given the head of a singly linked list where elements are sorted in ascending order, convert it to a height-balanced binary search tree." Example 1 shows a linked list with nodes -10, -3, 0, 5, 9 being converted into a binary search tree with root 0. The right panel shows the test result for Case 1, which is "Accepted" with a runtime of 51 ms. The input is head = [-10, -3, 0, 5, 9] and the output is [0, -3, 9, -10, null, 5].

3. Greedy Technique

3.1 Best time to buy and sell

stockCode:

class Solution(object):

def maxProfit(self, prices):

max = 0

start = prices[0]

len1 = len(prices)

for i in range(0 ,

len1):if start <

prices[i]:

max += prices[i] - start

```

        start = prices[i]

    return max

```

OUTPUT:

The screenshot shows the LeetCode interface for problem 122. On the left, the problem description states: "You are given an integer array `prices` where `prices[i]` is the price of a given stock on the i^{th} day. On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**. Find and return the **maximum** profit you can achieve."

Example 1:
Input: `prices = [7,1,5,3,6,4]`
Output: `7`
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4. Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3. Total profit is 4 + 3 = 7.

Example 2:
Input: `prices = [1,2,3,4,5]`
Output: `4`
Explanation: Buy on day 1 (price = 1) and sell on day 2 (price = 2), profit = 2-1 = 1. Then buy on day 3 (price = 3) and sell on day 4 (price = 4), profit = 4-3 = 1. Total profit is 1 + 1 = 2.

On the right, the submission interface shows the code editor with the following Python code:

```

class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        if not prices:
            return 0
        start = prices[0]
        profit = 0
        for i in range(1, len(prices)):
            if prices[i] > start:
                profit += prices[i] - start
                start = prices[i]
        return profit

```

The test results show "Accepted" with a runtime of 20 ms. The input is `prices = [1,2,3,4,5]` and the output is `4`, which matches the expected result.

3.2 remove duplicate letters

Code:

class Solution:

```
def removeDuplicateLetters(self, s):
```

```
    last_index = { }
```

```
    for k, v in enumerate(s):
```

```
        last_index[v] = k
```

```
    stack = [ ]
```

```
    visited = set()
```

```
    for i in
```

```
        range(len(s)):if
```

```
            s[i] in visited:
```

```
continue
```

```
while stack and stack[-1] > s[i] and last_index[stack[-1]] >
```

```
i:visited.remove(stack[-1])
```

```
stack.pop()
```

```
stack.append(s[i])
```

```
visited.add(s[i])
```

```
return "".join(stack)
```

OUTPUT:

The screenshot shows the LeetCode interface for problem 316, "Remove Duplicate Letters". The problem is categorized as "Medium". The description states: "Given a string *s*, remove duplicate letters so that every letter appears once and only once. You must make sure your result is the smallest in lexicographical order among all possible results."

Example 1:
Input: *s* = "bcabc"
Output: "abc"

Example 2:
Input: *s* = "cbacdcbc"
Output: "acdb"

Constraints:

- $1 \leq s.length \leq 10^4$
- s* consists of lowercase English letters.

The right side of the image shows a code editor with Python code and a "Testcase" tab. The test result is "Accepted" with a runtime of 11 ms. The input is *s* = "cbacdcbc", the output is "acdb", and the expected result is "acdb".

4.Back Tracking:

4.1 Find unique binary string

Code:

```
class Solution(object):  
    def findDifferentBinaryString(self, nums):  
        result = ""  
  
        for i in range(len(nums)):  
            result += '1' if nums[i][i] == '0' else '0'  
  
        return result
```

OUTPUT:

The screenshot shows the LeetCode interface for the problem "1980. Find Unique Binary String". The left sidebar contains the problem description, which states: "Given an array of strings `nums` containing `n` **unique** binary strings each of length `n`, return a *binary string of length `n` that **does not appear** in `nums`*. If there are multiple answers, you may return **any** of them." It includes three examples: Example 1 with input ["01", "10"] and output "11"; Example 2 with input ["00", "01"] and output "11"; and Example 3 with input ["000", "011", "001"] and output "000". The right sidebar shows the "Code" editor with a Python solution, the "Testcase" tab, and the "Test Result" showing "Accepted" with a runtime of 14 ms. The input field contains the array ["111", "011", "001"] and the output field shows "000".

4.2 Closest desert

costCode:

```
class Solution:  
    def closestCost(self, baseCosts: List[int], toppingCosts: List[int], target: int) -> int:  
        ans = float('inf')
```

```

# memo to accelerate
@lru_cache(None)
def helper(idx, comb):
    nonlocal ans

    if abs(comb - target) == abs(ans - target):
        ans = min(ans, comb)

    if abs(comb - target) < abs(ans - target):
        ans = comb

    for i in range(idx,
                    len(toppingCosts)):
        helper(i + 1, comb)
        helper(i + 1, comb + toppingCosts[i])
        helper(i + 1, comb + 2 * toppingCosts[i])

    for bc in baseCosts:
        helper(0, bc)

    return ans

```

OUTPUT:

The screenshot shows the LeetCode problem page for "1774. Closest Dessert Cost". The problem is categorized as "Medium". The description states: "You would like to make dessert and are preparing to buy the ingredients. You have n ice cream base flavors and m types of toppings to choose from. You must follow these rules when making your dessert: There must be **exactly one** ice cream base. You can add **one or more** types of topping or have no toppings at all. There are **at most two of each type** of topping. You are given three inputs: `baseCosts`, an integer array of length n , where each `baseCosts[i]` represents the price of the i^{th} ice cream base flavor. `toppingCosts`, an integer array of length m , where each `toppingCosts[i]` is the price of **one** of the i^{th} topping. `target`, an integer representing your target price for dessert. You want to make a dessert with a total cost as close to `target` as possible. Return the closest possible cost of the dessert to `target`. If there are multiple, return the minimum cost." The input values shown are `baseCosts = [1, 7]`, `toppingCosts = [3, 4]`, and `target = 10`. The output is `10`, which matches the expected result.

5.Brute Force:

5.1 Reverse

integerCode:

class Solution:

```
def reverse(self, x: int) ->
```

```
    int:res = 0
```

```
    if x < 0:
```

```
        res = int(str(x)[1:][::-1]) * -
```

```
    else:
```

```
        res = int(str(x)[::-1])
```

```
    if res > 2 ** 31 - 1 or res < -2 ** 31:
```

```
        return 0
```

```
    return res
```

OUTPUT:

The screenshot displays the LeetCode interface for the 'Reverse Integer' problem. On the left, the problem description states: 'Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-2³¹, 2³¹ - 1], then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).' It includes three examples: Example 1 (Input: 123, Output: 321), Example 2 (Input: -123, Output: -321), and Example 3 (Input: 120, Output: 21). The right panel shows the 'Code' editor with a Python3 solution and a 'Test Result' section indicating 'Accepted' with a runtime of 43 ms. The test case details show an input of 123, an output of 321, and an expected result of 321. The browser tabs at the top include 'leetcode login - Yahoo India S...', 'Reverse Integer - LeetCode', and 'ChatGPT'. The Windows taskbar at the bottom shows the time as 12:47 on 16-07-2024.

5.2 Factorial trailing

zerosCode:

class Solution:

```
def trailingZeroes(self, n: int) ->
    int: fact = 1

    for i in range(1, n+1):
        fact = fact * i

    sys.set_int_max_str_digits(100000)

    l =
    list(str(fact))

    ans = 0

    for i in range(len(l)-1, -1, -
        1):if l[i] == '0':
            ans += 1

            if l[i] !=
                '0':break

    return ans
```

OUTPUT:

The screenshot displays a web browser window with the LeetCode website. The main content area shows the problem '172. Factorial Trailing Zeroes' with a 'Medium' difficulty rating. The problem description states: 'Given an integer n, return the number of trailing zeroes in n!.' It also provides a formula for n! and three examples: Example 1 (n=3, output=0), Example 2 (n=5, output=1), and Example 3 (n=0, output=0). On the right side, the 'Code' tab is active, showing a Python solution. The solution defines a function 'trailingZeroes' that calculates the factorial of n and then counts the number of trailing zeros by iterating through the digits of the factorial string from right to left. The output of the code is shown as 'Accepted' with a runtime of 35 ms. The browser's address bar shows the URL 'leetcode.com/problems/factorial-trailing-zeroes/'.

6. Time and Trade Off

6.1 Contains

Duplicate Code:

class Solution:

```
def containsDuplicate(self, nums: List[int]) ->
```

```
    bool: n = len(nums)
```

```
    for i in range(n - 1):
```

```
        for j in range(i + 1, n):
```

```
            if nums[i] ==
```

```
                nums[j]: return
```

```
                True
```

```
    return False
```

OUTPUT:

The screenshot displays the LeetCode interface for the '217. Contains Duplicate' problem. The left sidebar shows the problem description: 'Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.' It includes three examples with their respective inputs and outputs. The right sidebar shows the 'Code' editor with a Python3 solution and a 'Test Result' section indicating 'Accepted' with a runtime of 38 ms. The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 12:55 on 16-07-2024.

217. Contains Duplicate

Given an integer array `nums`, return `true` if any value appears **at least twice** in the array, and return `false` if every element is distinct.

Example 1:

```
Input: nums = [1,2,3,1]
Output: true
```

Example 2:

```
Input: nums = [1,2,3,4]
Output: false
```

Example 3:

```
Input: nums = [1,1,1,3,3,4,3,2,4,2]
Output: true
```

Constraints:

- 1 ≤ `nums.length` ≤ 10⁵
- 10⁹ ≤ `nums[i]` ≤ 10⁹

Test Result: Accepted Runtime: 38 ms

Input: `nums = [1,2,3,1]`

Output: `true`

Expected: `true`

6.2 merge

intervalsCode:

```
class Solution:
def merge(self, intervals: List[List[int]]) -> List[List[int]]:
    merged = []
    intervals.sort(key=lambda x: x[0])

    prev = intervals[0]

    for interval in intervals[1:]:
        if interval[0] <= prev[1]:
            prev[1] = max(prev[1], interval[1])
        else:
            merged.append(prev)
            prev = interval

    merged.append(prev)

    return merged
```

OUTPUT:

The screenshot displays the LeetCode interface for the 'Merge Intervals' problem (Problem 56). The left pane shows the problem description, which asks to merge overlapping intervals. Example 1 shows intervals [[1,3],[2,6],[8,10],[15,18]] being merged into [[1,6],[8,10],[15,18]]. Example 2 shows intervals [[1,4],[4,5]] being merged into [[1,5]]. The right pane shows the 'Code' editor with a Python solution and the 'Test Result' section, which indicates that the solution is 'Accepted' with a runtime of 52 ms. The test case input is [[1,3],[2,6],[8,10],[15,18]] and the output is [[1,6],[8,10],[15,18]].

7.decrease and conquer

7.1 Validate binary search

treeCode:

class Solution(object):

```
def helper(self, root, prev, result):
```

```
    if root is None: return
```

```
    self.helper(root.left, prev, result)
```

```
    if prev[0] and root.val <=
```

```
        prev[0].val: result[0] = False
```

```
    return
```

```
    prev[0] = root
```

```
    self.helper(root.right, prev, result)
```

```
def isValidBST(self,
```

```
    root): prev = [None]
```

```
    result = [True]
```

```
    self.helper(root, prev, result)
```

```
    return result[0]
```

OUTPUT:

The screenshot displays the LeetCode interface for the problem '98. Validate Binary Search Tree'. The problem is categorized as 'Medium'. The description states: 'Given the root of a binary tree, determine if it is a valid binary search tree (BST). A valid BST is defined as follows: The left subtree of a node contains only nodes with keys less than the node's key. The right subtree of a node contains only nodes with keys greater than the node's key. Both the left and right subtrees must also be binary search trees.' An example tree is shown with root 2, left child 1, and right child 3. The test result interface on the right shows 'Accepted' for Case 1 with input 'root = [2, 1, 3]' and output 'true'. The runtime is 11 ms.

7.2 pow(x,n)

Code:

class Solution:

```
def myPow(self, x: float, n: int) -> float:
```

```
def calc_power(x, n):
```

```
    if x == 0:
```

```
        return 0
```

```
    if n == 0:
```

```
        return 1
```

```
    res = calc_power(x, n // 2)
```

```
    res = res * res
```

```
    if n % 2 == 1:
```

```
        return res * x
```

```
    return res
```

```
ans = calc_power(x, abs(n))
```

```
if n >= 0:
```

```
    return ans
```

```
return 1 / ans
```

OUTPUT:

The screenshot shows the LeetCode interface for the problem "50. Pow(x, n)". The problem is categorized as "Medium" and involves implementing a function to calculate x raised to the power n . The interface includes a description, examples, and a test result section. The test result shows that the solution was "Accepted" with a runtime of 31 ms. The input values are $x = 2.00000$ and $n = 10$, resulting in an output of 1024.00000 , which matches the expected output.

50. Pow(x, n)

Implement `pow(x, n)`, which calculates x raised to the power n (i.e., x^n).

Example 1:

Input: $x = 2.00000$, $n = 10$
Output: 1024.00000

Example 2:

Input: $x = 2.10000$, $n = 3$
Output: 9.26100

Example 3:

Input: $x = 2.00000$, $n = -2$
Output: 0.25000
Explanation: $2^{-2} = 1/2^2 = 1/4 = 0.25$

Constraints:

- $-100.0 < x < 100.0$
- $-2^{31} \leq n \leq 2^{31} - 1$
- The answer is accepted to 10^{-5} precision.

Testcase **Test Result**

Accepted Runtime: 31 ms

Case 1 **Case 2** **Case 3**

Input

$x =$
 2.00000

$n =$
 10

Output

1024.00000

Expected

1024.00000

Contribute a testcase

8.Transform Conquare

8.1 group

anagramsCode:

class Solution:

```
def groupAnagrams(self, strs):  
    anagram_map = defaultdict(list)  
  
    for word in strs:  
        sorted_word = ".join(sorted(word))  
        anagram_map[sorted_word].append(word)  
  
    return list(anagram_map.values())
```

OUTPUT:

The screenshot displays the LeetCode interface for the '49. Group Anagrams' problem. The left sidebar contains the problem description, which states: 'Given an array of strings `strs`, group the **anagrams** together. You can return the answer in **any order**. An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.' It includes three examples: Example 1 with input `strs = ["eat","tea","tan","ate","nat","bat"]` and output `[["bat"],["nat","tan"],["ate","eat","tea"]]`; Example 2 with input `strs = [""]` and output `[[""]]`; and Example 3 with input `strs = ["a"]` and output `[["a"]]`. The main area shows the Python solution code, which is the same as provided in the text. Below the code, the 'Testcase' tab is active, showing 'Accepted' status with a runtime of 36 ms. The input is `strs = ["eat","tea","tan","ate","nat","bat"]`, the output is `[["eat","tea","ate"],["tan","nat"],["bat"]]`, and the expected output is `[["bat"],["nat","tan"],["ate","eat","tea"]]`. The bottom of the image shows a Windows taskbar with various application icons and the system clock indicating 13:10 on 16-07-2024.

8.2 Integer to

RomanCode:

class Solution:

```
def intToRoman(self, num: int) -> str:
```

```
    Roman = ""
```

```
    storeIntRoman = [[1000, "M"], [900, "CM"], [500, "D"], [400, "CD"], [100, "C"], [90, "XC"], [50, "L"], [40, "XL"], [10, "X"], [9, "IX"], [5, "V"], [4, "IV"], [1, "I"]]
```

```
    for i in range(len(storeIntRoman)):
```

```
        while num >=
```

```
            storeIntRoman[i][0]: Roman +=
```

```
            storeIntRoman[i][1] num -=
```

```
            storeIntRoman[i][0]
```

```
    return Roman
```

OUTPU

The screenshot shows a web browser with multiple tabs. The active tab is 'Integer to Roman - LeetCode'. The page displays the problem description for '12. Integer to Roman', which is a Medium-level problem. It includes a table of Roman numeral symbols and their values, and a description of the conversion rules. The solution code is shown in a code editor, and the test results are displayed on the right. The test case 'Case 1' is selected, showing an input of 'num = 3749' and an output of 'MMMDCCLIX', which matches the expected result.

12. Integer to Roman

Medium

Seven different symbols represent Roman numerals with the following values:

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Roman numerals are formed by appending the conversions of decimal place values from highest to lowest. Converting a decimal place value into a Roman numeral has the following rules:

Testcase: **Accepted** Runtime: 29 ms

Case 1 Case 2 Case 3

Input

num = 3749

Output

"MMMDCCLIX"

Expected

"MMMDCCLIX"

Contribute a testcase

