

NAME: A. Harsha Mukundha

REG-NO: 192324070

COURSE CODE: CSA0386.

SUBJECT: DESIGN ANALYSIS OF ALGORITHM

1. Given an array of strings words, return the first palindromic string in the array. If there is no such string, return an empty string "". A string is palindromic if it reads the same forward and backward. Example 1: Input: words = ["abc","car","ada","racecar","cool"] Output: "ada"

main.py	Output
<pre>1 words = ["abc", "car", "ada", "racecar", "cool"] 2 palindrome = next((word for word in words if word == word[::-1]), "") 3 print("The first paldromic is:",palindrome) 4</pre>	<pre>The first paldromic is: ada === Code Execution Successful ===</pre>

2. You are given two integer arrays nums1 and nums2 of sizes n and m, respectively. Calculate the following values: answer1 : the number of indices i such that nums1[i] exists in nums2. answer2 : the number of indices i such that nums2[i] exists in nums1 Return [answer1,answer2].

main.py	Output
<pre>1 nums1 = [2, 3, 2] 2 nums2 = [1, 2] 3 4 answer1 = len([i for i in nums1 if i in nums2]) 5 answer2 = len([i for i in nums2 if i in nums1]) 6 7 result = [answer1, answer2] 8 print(result) 9</pre>	<pre>[2, 1] === Code Execution Successful ===</pre>

3. You are given a 0-indexed integer array nums. The distinct count of a subarray of nums is defined as: Let nums[i..j] be a subarray of nums consisting of all the indices from i to j such that $0 \leq i \leq j < \text{nums.length}$. Then the number of distinct values in nums[i..j] is called the distinct count of nums[i..j]. Return the sum of the squares of distinct counts of all subarrays of nums. A subarray is a contiguous non-empty sequence of elements within an array. Example 1: Input: nums = [1,2,1] Output: 15

main.py	Output
<pre>1 nums = [1, 2, 1] 2 n = len(nums) 3 result = 0 4 5 for i in range(n): 6 distinct = set() 7 for j in range(i, n): 8 distinct.add(nums[j]) 9 distinct_count = len(distinct) 10 result += distinct_count ** 2 11 12 print(result) 13</pre>	15 === Code Execution Successful ===

4. Given a 0-indexed integer array `nums` of length `n` and an integer `k`, return the number of pairs (i, j) where $0 \leq i < j < n$, such that `nums[i] == nums[j]` and $(i * j)$ is divisible by `k`. Example 1: Input: `nums = [3,1,2,2,2,1,3]`, `k = 2` Output: 4

main.py	Output
<pre>1 nums = [3, 1, 2, 2, 2, 1, 3] 2 k = 2 3 n = len(nums) 4 count = 0 5 6 for i in range(n): 7 for j in range(i + 1, n): 8 if nums[i] == nums[j] and (i * j) % k == 0: 9 count += 1 10 11 print(count) 12</pre>	4 === Code Execution Successful ===

main.py	Output
<pre>1 input_list = [1, 2, 3, 4, 5] 2 output = max(input_list) 3 print(output) 4</pre>	5 === Code Execution Successful ===

6. You have an algorithm that process a list of numbers. It firsts sorts the list using an efficient sorting algorithm and then finds the maximum element in sorted list. Write the code for the same.

main.py	Run	Output
<pre>1 def sort_and_find_max(input_list): 2 sorted_list = sorted(input_list) 3 max_element = sorted_list[-1] 4 return max_element 5 6 numbers = [5, 2, 8, 1, 9] 7 max_num = sort_and_find_max(numbers) 8 print("Maximum element in the list:", max_num) 9</pre>		<pre>Maximum element in the list: 9 === Code Execution Successful ===</pre>

7. Write a program that takes an input list of n numbers and creates a new list containing only the unique elements from the original list. What is the space complexity of the algorithm?

Some Duplicate Elements • Input: [3, 7, 3, 5, 2, 5, 9, 2]

• Expected Output: [3, 7, 5, 2, 9] (Order may vary based on the algorithm used)

main.py	Run	Output
<pre>1 def get_unique_elements(input_list): 2 return list(set(input_list)) 3 4 input_list = [3, 7, 3, 5, 2, 5, 9, 2] 5 unique_elements = get_unique_elements(input_list) 6 print(unique_elements) 7</pre>		<pre>[2, 3, 5, 7, 9] === Code Execution Successful ===</pre>

8. Sort an array of integers using the bubble sort technique. Analyze its time complexity using Big-O notation. Write the code

main.py	Output
<pre>1 arr = [64, 34, 25, 12, 22, 11, 90] 2 3 n = len(arr) 4 for i in range(n): 5 for j in range(0, n-i-1): 6 if arr[j] > arr[j+1]: 7 arr[j], arr[j+1] = arr[j+1], arr[j] 8 9 print("Sorted array is:", arr) 10</pre>	<pre>Sorted array is: [11, 12, 22, 25, 34, 64, 90] === Code Execution Successful ===</pre>

9. Checks if a given number x exists in a sorted array arr using binary search. Analyze its time complexity using Big-O notation. Test Case: Example X={3,4,6,-9,10,8,9,30} KEY=10 Output: Element 10 is found at position 5

main.py	Output
<pre>1 arr = [-9, 3, 4, 6, 8, 9, 10, 30] 2 key = 10 3 4 left = 0 5 right = len(arr) - 1 6 found = False 7 8 while left <= right: 9 mid = left + (right - left) // 2 10 if arr[mid] == key: 11 print(f"Element {key} is found at position {mid}") 12 found = True 13 break 14 elif arr[mid] < key: 15 left = mid + 1 16 else: 17 right = mid - 1 18 19 if not found: 20 print(f"Element {key} is not found in the array") 21</pre>	<pre>Element 10 is found at position 6 === Code Execution Successful ===</pre>

10. Given an array of integers nums, sort the array in ascending order and return it. You must solve the problem without using any built-in functions in $O(n \log(n))$ time complexity and with the smallest space complexity possible.

```
nums = [5, 2, 9, 1, 5, 6]
n = len(nums)
size = 1
while size < n:
    left = 0
    while left < n - size:
        mid = left + size - 1
        right = min((left + 2 * size - 1), (n - 1))

        left_sub = nums[left:mid + 1]
        right_sub = nums[mid + 1:right + 1]

        i = j = 0
        k = left

        while i < len(left_sub) and j < len(right_sub):
            if left_sub[i] <= right_sub[j]:
                nums[k] = left_sub[i]
                i += 1
            else:
                nums[k] = right_sub[j]
                j += 1
            k += 1

        while i < len(left_sub):
            nums[k] = left_sub[i]
            i += 1
            k += 1

        while j < len(right_sub):
            nums[k] = right_sub[j]
            j += 1
            k += 1

        left += 2 * size

    size *= 2

print(nums)
```

```
IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\T_NIKHIL\Documents\24th.py
[1, 2, 5, 5, 6, 9]
>>>
```