**DSA – Test -1**

1. Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the nonzero elements.

Note that you must do this in-place without making a copy of the array.

**Example 1:** Input: nums = [0,1,0,3,12] Output: [1,3,12,0,0]

**Solution: Quick Sort**

1. Use two pointers, left and right.
2. If the element is non-zero add it to the left and if its zero add it to the right.
3. Keep incrementing the left pointers, and traverse through array of elements.
4. **TC: O(n)** – traverse only once.
5. **SC: O(1)** – as no extra space is used.

**Code:**

class Solution:

    def moveZeroes(self, nums: List[int]) -> None:

        """

        Do not return anything, modify nums in-place instead.

        """

        l = 0

        for r in range(len(nums)):

            if nums[r] !=0: # if nums[r]:

                nums[l],nums[r]=nums[r],nums[l]

                l += 1

        return nums

1. Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1.  
     
   **Example 1:**  
   Input: s = "leetcode"  
   Output: 0  
     
   **Example 2:**  
   Input: s = "loveleetcode"  
   Output: 2  
     
   **Example 3:**  
   Input: s = "aabb"  
   Output: -1

**Solution:**

**Algorithm:**

1. Import collections counter package.
2. Use an object count to traverse through the string to count the occurrences of the char.
3. Use a loop to iterate and return the position of the char, if the count [c] == 1.
4. If not found return -1.
5. TC: O(n) – as it traverses through the given input string.
6. SC: O(1) – as its done at constant space.

**Code:**

class Solution:

  def firstUniqChar(self, s: str) -> int:

    count = collections.Counter(s)

    for i, c in enumerate(s):

      if count[c] == 1:

        return i

    return -1