DEPARTMENTOF
COMPUTERSCIENCE&ENGINEERING
Discover. Learn. Empower.

Experiment-1(A)

StudentName: Pranav Saini
UID: 22BCS15994
Branch: CSE
Section/Group:602-A
Semester:6
DateofPerformance:17-02-25
SubjectName: AdvancedProgrammingLab-2
SubjectCode:22CSH-359

1. **Title:**Arrays and logic building (Remove duplicates from a sorted array)
   https://leetcode.com/problems/remove-duplicates-from-sorted-array/

2. **Objective:**The goal is to modify the input sorted array nums by removing the duplicates in-place, such that each unique element appears only once, and return the number of unique elements. The relative order of the elements should be kept the same.

3. **Algorithm:**

   - **InitialCheck:**

   - Ifthearrayisempty,return0,astherearenoelementstoprocess.

   - **PointerInitialization:**

   - Use two pointers: i for the index where the next unique element should be placed, and j for iterating through the array.

   - **TraversetheArray:**

   - Startwithj=1,andcompareeachelementnums[j]withtheelementatindexi.

   - If nums[i] is not equal to nums[j], it means a newunique element is found. Increment i and assign nums[i] = nums[j].

   - **ReturntheCountofUnique Elements:**

   - Aftertheloopfinishes,returni+ 1,asiwillbetheindexofthelastuniqueelement.

4. **Implementation/Code:**

```
classSolution:
    def removeDuplicates(self, nums):
        # If the array is empty
           ifnotnums:
              return0
```

```
#Pointerfortheplacetoinsertthenextuniqueelement i = 0

# Traverse the array starting from index 1
for j in range(1, len(nums)):
    if nums[i] != nums[j]:
        i += 1
        nums[i]=nums[j]

# The number of unique elements is i + 1
return i + 1
```

5. **Output:**

6. **TimeComplexity:**O(n)

7. **SpaceComplexity:**O(1)

8. **LearningOutcomes:**

- **In-Place Modification:** Understanding how to modify an array in-place while maintaining its integrity, which is a key concept in optimizing space complexity.

- **Two Pointer Technique:** Learn howto use the two-pointer technique to solve problems where elements need to be moved or compared based on certain conditions.

Experiment1(B)

1. **Title:**Containsduplicate(https://leetcode.com/problems/contains-duplicate/description/)

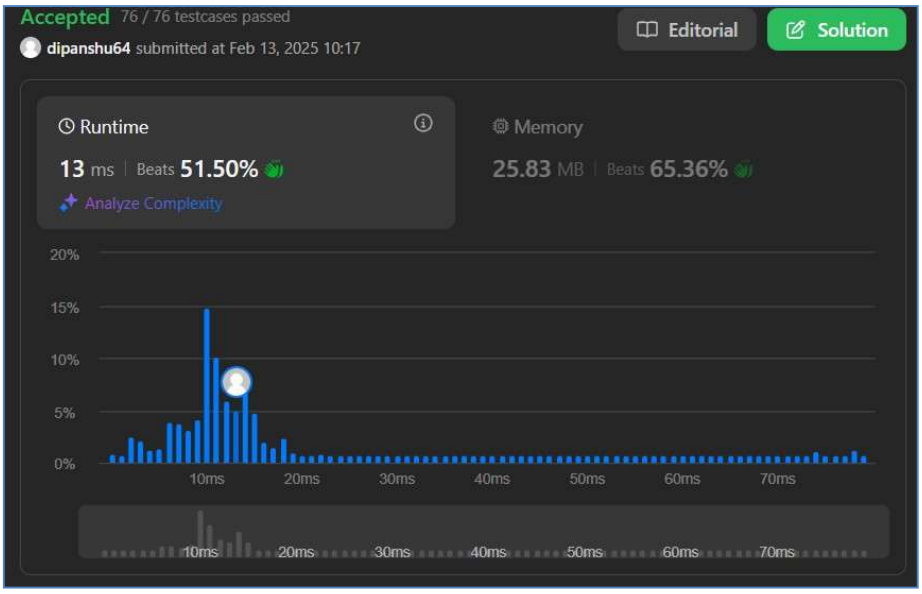
2. **Objective:**The goal is to determine if any value appears at least twice in the given integer array nums, and return true if any value is duplicated, orfalse if allelements are distinct.

3. **Algorithm:**
   - **Convertthe ArraytoaSet:**
     - A set datastructureautomaticallyremovesduplicateelements.By converting the input list `nums`to a set, we can easily determine if any duplicates existed.
   - **CompareLengths:**
     - If the length of the set is smaller than the length of the array `nums`, it means some elements were duplicates (since the set removed them). In this case, return`true`.
     - Ifthelengthsarethesame,thentherewerenoduplicates,andwereturn`false`.

4. **Implementation/Code:**

```
classSolution:
    defcontainsDuplicate(self,nums):
        #Comparethelengthofthearrayandthesetofthearray return len(nums)
        != len(set(nums))
```

6. **Output:**



Experiment1(C)

1. **Title:**TwoSum(https://leetcode.com/problems/two-sum/)

2. **Objective:**Given an array of integers nums and a target integer target, the task is to return the indices of the two numbers such that they add up to the target. The solution should be efficient and must not use the same element twice.

3. **Algorithm:**

- **UseaHashMap(Dictionary):**
  - Createa dictionaryseen tostoreeach numberand itscorrespondingindex as we iterate through the array.
  - Foreachelementinthearray:
    - Calculatethe**complement**astarget-currentnumber.
    - Checkifthecomplementisalreadyinthedictionary:
      - Ifitis,returntheindicesofthecomplementandthecurrentnumber.
      - Ifnot,storethecurrentnumberalongwithitsindexinthedictionary.
  - Thisallowsustofindthesolutioninasinglepassthroughthearray.

4. **Implementation/Code:**

```
classSolution:
    def twoSum(self, nums, target):
        seen = {}
        for i, num in enumerate(nums):
            complement = target - num
            if complement in seen:
                return [seen[complement], i]
            seen[num] = i
```

5. **Output:**