Assignment 01 Solutions

Q1. Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

Example: Input: nums = [2,7,11,15], target = 9 Output0 [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1][

In [9]:
```
def twoSum(nums, target):
    # Create a dictionary to store the complement of each number
    complement_dict = {}

    # Iterate through the array
    for i, num in enumerate(nums):
        complement = target - num
        # If the complement exists in the dictionary, return its index and the current index
        if complement in complement_dict:
            return [complement_dict[complement], i]
        # Otherwise, add the current number and its index to the dictionary
        complement_dict[num] = i

    # If no solution is found, return an empty list
    return []

# Example usage
nums = [2, 7, 11, 15]
target = 9
result = twoSum(nums, target)
print(result)
```
```
[0, 1]
```

Q2. Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val. Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- Return k.

Example :

Input: nums = [3,2,2,3], val = 3 Output: 2, nums = [2,2,,]

Explanation: Your function should return k = 2, with the first two elements of nums being 2. It does not matter what you leave beyond the returned k (hence they are underscores)[

In [10]:
```
def removeElement(nums, val):
    k = 0
    for i in range(len(nums)):
        if nums[i] != val:
            nums[k] = nums[i]
            k += 1
    return k

nums = [3, 2, 2, 3]
val = 3
result = removeElement(nums, val)
print(result)  # Output: 2
print(nums)  # Output: [2, 2, _, _]
```
```
2
[2, 2, 2, 3]
```

Q3. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted

in order. You must write an algorithm with O(log n) runtime complexity.

Example 1: Input: nums = [1,3,5,6], target = 5

Output: 2

In [11]:
```python
def searchInsert(nums, target):
    left = 0
    right = len(nums) - 1

    while left <= right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return left

# Example usage
nums = [1, 3, 5, 6]
target = 5
index = searchInsert(nums, target)
print("Output:", index)
```

```
Output: 2
```

Q4. You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's. Increment the large integer by one and return the resulting array of digits.

Example 1: Input: digits = [1,2,3] Output: [1,2,4]

Explanation: The array represents the integer 123.

Incrementing by one gives 123 + 1 = 124. Thus, the result should be [1,2,4]

In [12]:
```python
def plusOne(digits):
    n = len(digits)

    # Start from the least significant digit
    for i in range(n - 1, -1, -1):
        # If the current digit is less than 9, we can simply increment it and return the array
        if digits[i] < 9:
            digits[i] += 1
            return digits
        # Otherwise, set the current digit to 0 and continue to the next digit
        digits[i] = 0

    # If all digits are 9, we need to add an additional digit at the front
    digits.insert(0, 1)
    return digits

# Example usage
digits = [1, 2, 3]
result = plusOne(digits)
print("Output:", result)
```

```
Output: [1, 2, 4]
```

Q5. You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.Merge nums1 and nums2 into a single array sorted in non-decreasing order.The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

Example 1: Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3 Output: [1,2,2,3,5,6]

Explanation: The arrays we are merging are [1,2,3] and [2,5,6]. The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

In [13]:
```
def merge(nums1, m, nums2, n):
    # Initialize pointers for nums1, nums2, and the merged array
    p1 = m - 1
    p2 = n - 1
    p = m + n - 1

    # Compare and merge elements starting from the end of both arrays
    while p1 >= 0 and p2 >= 0:
        if nums1[p1] >= nums2[p2]:
            nums1[p] = nums1[p1]
            p1 -= 1
        else:
            nums1[p] = nums2[p2]
            p2 -= 1
        p -= 1

    # Copy any remaining elements from nums2 to nums1
    while p2 >= 0:
        nums1[p] = nums2[p2]
        p2 -= 1
        p -= 1

# Example usage
nums1 = [1, 2, 3, 0, 0, 0]
m = 3
nums2 = [2, 5, 6]
n = 3
merge(nums1, m, nums2, n)
print("Output:", nums1)
```

```
Output: [1, 2, 2, 3, 5, 6]
```

Q6. 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

In [14]:
```
def containsDuplicate(nums):
    num_set = set()

    for num in nums:
        # If the current element is already in the set, it's a duplicate
        if num in num_set:
            return True
        # Otherwise, add the current element to the set
        num_set.add(num)

    # No duplicates found
    return False

# Example usage
nums = [1, 2, 3, 1]
result = containsDuplicate(nums)
print("Output:", result)
```

```
Output: True
```

Q7. 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]

In [15]:
```
def moveZeroes(nums):
    # Initialize two pointers: one to track the position of the last non-zero element,
    # and another to iterate through the array
    last_non_zero = 0

    # Iterate through the array
    for i in range(len(nums)):
        # If the current element is non-zero, swap it with the element at the last non-zero position
        if nums[i] != 0:
            nums[i], nums[last_non_zero] = nums[last_non_zero], nums[i]
            last_non_zero += 1

# Example usage
nums = [0, 1, 0, 3, 12]
moveZeroes(nums)
print("Output:", nums)
```

```
Output: [1, 3, 12, 0, 0]
```

Q8. You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number. You are given an integer array nums representing the data status of this set after the error. Find the number that occurs twice and the number that is missing and return them in the form of an array.

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

In [17]:

```
def findErrorNums(nums):
    num_set = set()
    duplicate = -1
    missing = -1

    # Find the duplicate and missing numbers
    for num in nums:
        if num in num_set:
            duplicate = num
        else:
            num_set.add(num)

    # Find the missing number by checking the range from 1 to n
    for i in range(1, len(nums) + 1):
        if i not in num_set:
            missing = i
            break

    return [duplicate, missing]

# Example usage
nums = [1, 2, 2, 4]
result = findErrorNums(nums)
print("Output:", result)
```

```
Output: [2, 3]
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js