Assignment

June 24, 2023

[3]:

```
'''
Q.1 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]
'''



# Program :-
nums = [2, 7, 11, 15]
target = 9

num_map = {}
for i, num in enumerate(nums):
    complement = target - num
    if complement in num_map:
        # Return the indices of the two numbers
        indices = [num_map[complement], i]
        print(indices)  # Output: [0, 1]
        break
    else:
        # Add the current element and its index
        num_map[num] = i
```

```
[0, 1]
```

[4]:

```
'''
```

```
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)

'''

# Program :-
nums = [3, 2, 2, 3]
val = 3

i = 0
j = 0

while i < len(nums):
    if nums[i] != val:
        nums[j] = nums[i]
        j += 1
    i += 1

count = j  # Count of elements not equal to val
result = nums[:count]  # Updated array with non-val elements

print(count)  # Output: 2
print(result)  # Output: [2, 2]
```

```
2
[2, 2]
```

[5]:

```
'''
```

```
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
'''

# Program :-
nums = [1, 3, 5, 6]
target = 5

left = 0
right = len(nums) - 1

while left <= right:
    mid = (left + right) // 2
    if nums[mid] == target:
        print(mid)  # Output: 2
        break
    elif nums[mid] < target:
        left = mid + 1
    else:
        right = mid - 1
else:
    print(left)  # Output: 2
```

```
2
```

[6]:

```
'''
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].
'''

# Program :-
digits = [1, 2, 3]

n = len(digits)

for i in range(n - 1, -1, -1):
    digits[i] += 1
    if digits[i] < 10:
        break
    digits[i] = 0

if digits[0] == 0:
    digits.insert(0, 1)

result = digits
print(result)  # Output: [1, 2, 4]
```

```
[1, 2, 4]
```

[7]:

```
'''
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.
'''

# Program :-
nums1 = [1, 2, 3, 0, 0, 0]
```

```
m = 3
nums2 = [2, 5, 6]
n = 3

# Start merging
i = m - 1
j = n - 1
k = m + n - 1

while i >= 0 and j >= 0:
    if nums1[i] > nums2[j]:
        nums1[k] = nums1[i]
        i -= 1
    else:
        nums1[k] = nums2[j]
        j -= 1
    k -= 1

# Copy any remaining elements from nums2
while j >= 0:
    nums1[k] = nums2[j]
    j -= 1
    k -= 1

# Output the merged array
print(nums1)  # Output: [1, 2, 2, 3, 5, 6]
```

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

[8]:

```
'''
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

'''


# Program :-
nums = [1, 2, 3, 1]

# Create an empty set to store unique values
unique_set = set()

for num in nums:
    if num in unique_set:
        # Found a duplicate value
        print(True)  # Output: True
        break
    unique_set.add(num)
else:
    # No duplicates found
    print(False)  # Output: False
```

```
True
```

[9]:

```
'''
 **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]
'''

# Program :-
nums = [0, 1, 0, 3, 12]

i = 0

for j in range(len(nums)):
    if nums[j] != 0:
        # Swap non-zero element
        nums[i], nums[j] = nums[j], nums[i]
        i += 1

# Fill the remaining positions with zeros
while i < len(nums):
    nums[i] = 0
    i += 1

# Output the modified array
print(nums)  # Output: [1, 3, 12, 0, 0]
```

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

[10]:

```
'''
```

```
 **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]

'''

# Program :-
nums = [1, 2, 2, 4]

n = len(nums)

unique_set = set()

# Variables to store the number that occurs twice and the missing number
duplicate_num = -1
missing_num = -1

# Iterate through nums
for num in nums:
    if num in unique_set:
        duplicate_num = num
    unique_set.add(num)

# Calculate the sum of all numbers
total_sum = sum(range(1, n + 1))

# Calculate the sum of nums
nums_sum = sum(nums)

# Calculate the missing number
missing_num = total_sum - nums_sum + duplicate_num

# Output the result as an array
result = [duplicate_num, missing_num]
print(result)  # Output: [2, 3]
```

```
[2, 3]
```