

Assignment Questions 3

Question 1 Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to the target. Return the sum of the three integers.

You may assume that each input would have exactly one solution.

Example 1: Input: `nums = [-1,2,1,-4]`, `target = 1` Output: 2

Explanation: The sum that is closest to the target is 2. $(-1 + 2 + 1 = 2)$.

```
In [2]: def output_a(nums, target):
        nums.sort()
        n = len(nums)
        closest_sum = float('inf')
        for i in range(n - 2):
            left = i + 1
            right = n - 1
            while left < right:
                current_sum = nums[i] + nums[left] + nums[right]
                if current_sum == target:
                    return target
                if abs(current_sum - target) < abs(closest_sum - target):
                    closest_sum = current_sum
                if current_sum < target:
                    left += 1
                else:
                    right -= 1

        return closest_sum

result = output_a([-1, 2, 1, -4], 1)
print(result)
```

2

In []:

Question 2 Given an array `nums` of `n` integers, return an array of all the unique quadruplets `[nums[a], nums[b], nums[c], nums[d]]` such that: • $0 \leq a, b, c, d < n$ • `a, b, c, and d` are distinct. • `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in any order.

Example 1: Input: `nums = [1,0,-1,0,-2,2]`, `target = 0` Output: `[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]`

```
In [3]: def output_b(nums, target):
    nums.sort()
    n = len(nums)
    result = []

    for a in range(n - 3):
        if a > 0 and nums[a] == nums[a - 1]:
            continue

        for b in range(a + 1, n - 2):
            if b > a + 1 and nums[b] == nums[b - 1]:
                continue
            left = b + 1
            right = n - 1

            while left < right:
                current_sum = nums[a] + nums[b] + nums[left] + nums[right]
                if current_sum == target:
                    result.append([nums[a], nums[b], nums[left], nums[right]])


                    while left < right and nums[left] == nums[left + 1]:
                        left += 1
                    while left < right and nums[right] == nums[right - 1]:
                        right -= 1

                    left += 1
                    right -= 1
                elif current_sum < target:
                    left += 1
                else:
                    right -= 1

            return result
result = output_b([1,0,-1,0,-2,2], 0)
print(result)
```

[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]

In []:

 **Question 3** A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container.

If such an arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

• For example, the next permutation of arr = [1,2,3] is [1,3,2]. • Similarly, the next permutation of arr = [2,3,1] is [3,1,2]. • While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, find the next permutation of nums. The replacement must be in place and use only constant extra memory.

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

```
In [6]: def output_c(nums):
        n = len(nums)
        i = n - 2

        while i >= 0 and nums[i] >= nums[i + 1]:
            i -= 1

        if i >= 0:
            j = n - 1

            while j > i and nums[j] <= nums[i]:
                j -= 1

            nums[i], nums[j] = nums[j], nums[i]

        left = i + 1
        right = n - 1
        while left < right:
            nums[left], nums[right] = nums[right], nums[left]
            left += 1
            right -= 1
        nums = [1, 2, 3]
        output_c(nums)
        print(nums)
```

[1, 3, 2]

In []:

Question 4 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 [8]: def output_d(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
nums = [1, 3, 5, 6]
target = 5

result = output_d(nums, target)
print(result)
```

2

In []:

 **Question 5** 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 [9]: def output_e(digits):
        n = len(digits)
        carry = 1

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

        if carry == 1:
            digits.insert(0, 1)

        return digits
digits = [1, 2, 3]
result = output_e(digits)
print(result)
```

[1, 2, 4]

In []:

Question 6 Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

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

```
In [10]: def output_f(nums):
        result = 0

        for num in nums:
            result ^= num

        return result
nums = [2, 2, 1]
result = output_f(nums)
print(result)
```

1

In []:

Question 7

You are given an inclusive range [\[lower, upper\]](#) and a sorted unique integer array nums, where all elements are within the inclusive range.

A number x is considered missing if x is in the range `[lower, upper]` and x is not in `nums`.

Return the shortest sorted list of ranges that exactly covers all the missing numbers. That is, no element of `nums` is included in any of the ranges, and each missing number is covered by one of the ranges.

Example 1:

Input: `nums = [0,1,3,50,75]`, `lower = 0`, `upper = 99`

Output: `[[2,2],[4,49],[51,74],[76,99]]`.

Explanation: The ranges are:

`[2,2]`.

`[4,49]`.

`[51,74]`.

`[76,99]`.

```
In [15]: def output_g(nums, lower, upper):
    result = []
    start = lower

    for num in nums:
        if num > start:
            result.append(k(start, num - 1))
            start = num + 1

    if start <= upper:
        result.append(k(start, upper))

    return result

def k(start, end):
    if start == end:
        return str(start)
    else:
        return [str(start) + "," + str(end)]

nums = [0, 1, 3, 50, 75]
lower = 0
upper = 99

result = output_g(nums, lower, upper)
print(result)
```

```
['2', ['4,49'], ['51,74'], ['76,99']]
```

In []:

Question 8 Given an array of meeting time intervals where intervals[i] = [starti, endi], determine if a person could attend all meetings.

Example 1: Input: intervals = [[0,30],[5,10],[15,20]] Output: false

```
In [16]: def output_h(intervals):
          intervals.sort(key=lambda x: x[0])
          for i in range(1, len(intervals)):
              if intervals[i][0] < intervals[i - 1][1]:
                  return False
          return True
          intervals = [[0, 30], [5, 10], [15, 20]]
          result = output_h(intervals)
          print(result)
```

False

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []: