

PPT Assignment 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. $(-1 + 2 + 1 = 2)$.

```
3
4  def func1(nums, target):
5      inf_sum=float('inf')
6      if(len(nums)>2):
7          for i in range(0, len(nums)):
8              for j in range(i+1, len(nums)):
9                  for k in range(j+1, len(nums)):
10                     sum = nums[i]+nums[j]+nums[k]
11                     if abs(target - sum) < abs(target - inf_sum):
12                         inf_sum = abs(target - sum)
13
14     print(inf_sum)
15
16     nums = [-1,2,1,-4]
17     target =3
18     func1(nums, target)
19
20
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
>>> target =3
>>> func1(nums, target)
1
```

PPT Assignment 3

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]]`

```
1  # Input: nums = [1,0,-1,0,-2,2], target = 0
2  # Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
3
4  from itertools import combinations
5
6  def func1(nums, target):
7      a=[]
8      all_possible_combinations = list(combinations(nums,4))
9
10     for lists in all_possible_combinations:
11         if sum(lists) == target:
12             a.append(lists)
13
14     print(a)
15
16     nums = [1,0,-1,0,-2,2]
17     target =0
18     func1(nums, target)
19
20
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
>>> target =0
>>> func1(nums, target)
[(1, 0, -1, 0), (1, -1, -2, 2), (0, 0, -2, 2)]
>>>
```

PPT Assignment 3

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

For example, for $\text{arr} = [1,2,3]$, the following are all the permutations of arr : $[1,2,3]$, $[1,3,2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3,1,2]$, $[3,2,1]$.

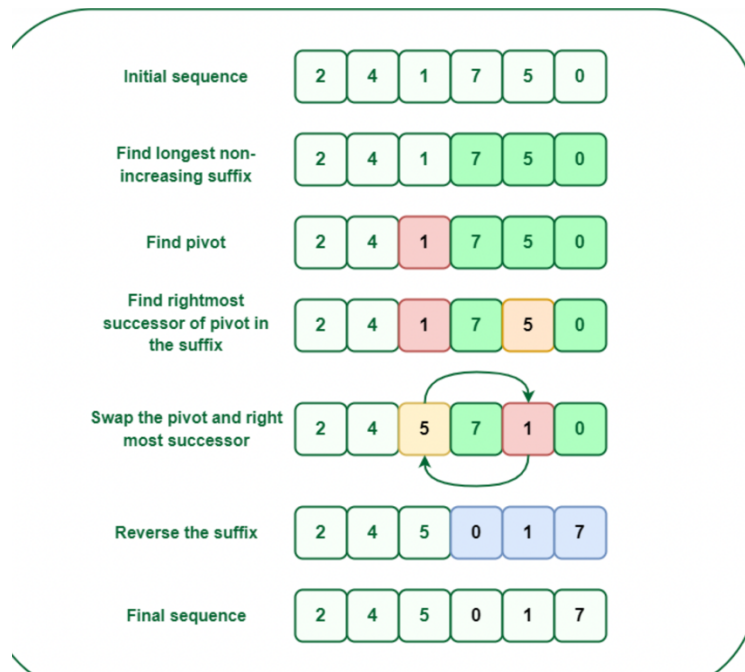
The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows 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 $\text{arr} = [1,2,3]$ is $[1,3,2]$. • Similarly, the next permutation of $\text{arr} = [2,3,1]$ is $[3,1,2]$. • While the next permutation of $\text{arr} = [3,2,1]$ 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: $\text{nums} = [1,2,3]$ Output: $[1,3,2]$



PPT Assignment 3

```
1  from typing import List
2  def func1(nums):
3      for i in range(len(nums)-1,-1,-1):
4          if(nums[i]>nums[i-1]):
5              nums = func2(nums, i)
6              print(nums)
7              break
8
9
10 def func2(nums,i) -> List[int]:
11     nums[i:] = sorted(nums[i:])
12     for j in range(i, len(nums)-1):
13         if (nums[j]>nums[i-1]):
14             nums[i-1], nums[j] = nums[j], nums[i-1]
15             break
16
17     return(nums)
18
19 nums = [1,7,8,6,4,7,5,3]
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
...         break
...     return(nums)
...
>>> nums = [1,7,8,6,4,7,5,3]
>>> func1(nums)
[1, 7, 8, 6, 5, 3, 4, 7]
```

PPT Assignment 3

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

```
def func1(nums, target):
    left = 0
    right = len(nums) - 1

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

        if nums[mid] == target:
            return mid

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

    return left

nums = [1, 3, 5, 6]
target = 5
result = func1(nums, target)
print(result)
```

PPT Assignment 3

Question 5 You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the *i*th 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]`.

```
1  def func1(digits):
2      length = len(digits) - 1
3      while digits[length] == 9:
4          digits[length] == 0
5          length -= 1
6
7      if digits[length] > 0:
8          digits[length] += 1
9
10     return digits
11
12     digits = [1,2,3]
13     func1(digits)
14
```

PROBLEMS

8

OUTPUT

DEBUG CONSOLE

TERMINAL

[1, 2, 4]

>>>

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

```
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        for i in nums:
            if(nums.count(i) == 1):
                return(i)
```



PPT Assignment 3

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]

PPT Assignment 3

```
def find_missing_ranges(nums, lower, upper):
    missing_ranges = []

    if nums[0] > lower:
        missing_ranges.append([lower, nums[0] - 1])

    for i in range(len(nums) - 1):
        if nums[i + 1] != nums[i] + 1:
            if nums[i + 1] > nums[i] + 1:
                missing_ranges.append([nums[i] + 1, nums[i + 1] - 1])

    if nums[-1] < upper:
        missing_ranges.append([nums[-1] + 1, upper])

    return missing_ranges

nums = [0, 1, 3, 50, 75]
lower = 0
upper = 99
output = find_missing_ranges(nums, lower, upper)
print(output)
```

Question 8

Given an array of meeting time intervals where $\text{intervals}[i] = [\text{start}_i, \text{end}_i]$, determine if a person could attend all meetings.

Example 1:

Input: $\text{intervals} = [[0,30],[5,10],[15,20]]$

Output: false

```
1
2  def canAttendMeetings(intervals):
3      intervals.sort(key = lambda x: x[0])
4      for i in range(0, len(intervals)-1):
5          if(intervals[i][1] < intervals[i+1][0]):
6              return False
7      return True
8
9  intervals = [[0, 30], [5, 10], [15, 20]]
10 canAttendMeetings(intervals) |
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL

```
...     intervals.sort(key = lambda x: x[0])
...     for i in range(0, len(intervals)-1):
...         if(intervals[i][1] < intervals[i+1][0]):
...             return False
...     return True
...
>>> intervals = [[0, 30], [5, 10], [15, 20]]
>>> canAttendMeetings(intervals)
False
```