**Class Notes 1**

* Python

<aside> 💡 Question 1

Q1. Given an array of size N. The task is to find the maximum and the minimum element of the array using the minimum number of comparisons. Examples: Input: arr[] = {3, 5, 4, 1, 9} Output: Minimum element is: 1 Maximum element is: 9

TC : O(n) SC: O(n)

class pair:

def \_\_init\_\_(self):

self.min = 0

self.max = 0

def getMinMax(arr: list, n: int) -> pair:

minmax = pair()

# If there is only one element then return it as min and max both

if n == 1:

minmax.max = arr[0]

minmax.min = arr[0]

return minmax

# If there are more than one elements, then initialize min

# and max

if arr[0] > arr[1]:

minmax.max = arr[0]

minmax.min = arr[1]

else:

minmax.max = arr[1]

minmax.min = arr[0]

for i in range(2, n):

if arr[i] > minmax.max:

minmax.max = arr[i]

elif arr[i] < minmax.min:

minmax.min = arr[i]

return minmax

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_":

arr = [1000, 11, 445, 1, 330, 3000]

arr\_size = 6

minmax = getMinMax(arr, arr\_size)

print("Minimum element is", minmax.min)

print("Maximum element is", minmax.max)

</aside>

<aside> 💡 Question 2

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock. Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0. Example : Input: prices = [7,1,5,3,6,4] Output: 5 Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5. Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell. Solution:

class Solution:

def maxProfit(self, prices: List[int]) -> int:

min\_price = float('inf')

max\_profit = 0

for i in range(len(prices)):

if prices[i] < min\_price:

min\_price = prices[i]

elif prices[i] - min\_price > max\_profit:

max\_profit = prices[i] - min\_price

return max\_profit

</aside>

<aside> 💡 Question 3

Given an integer array nums, find a subarray that has the largest product, and return the product. The test cases are generated so that the answer will fit in a 32-bit integer. Example: Input: nums = [2,3,-2,4] Output: 6 Explanation: [2,3] has the largest product 6.

class Solution:

def maxProduct(self, nums: List[int]) -> int:

if len(nums) == 0:

return 0

max\_so\_far = nums[0]

min\_so\_far = nums[0]

result = max\_so\_far

for i in range(1, len(nums)):

curr = nums[i]

temp\_max = max(curr, max\_so\_far \* curr, min\_so\_far \* curr)

min\_so\_far = min(curr, max\_so\_far \* curr, min\_so\_far \* curr)

max\_so\_far = temp\_max

result = max(max\_so\_far, result)

return result

</aside>

<aside> 💡 Question 4

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0. Notice that the solution set must not contain duplicate triplets. Example: Input: nums = [-1,0,1,2,-1,-4] Output: [[-1,-1,2],[-1,0,1]] Explanation: nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0. nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0. nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0. The distinct triplets are [-1,0,1] and [-1,-1,2]. Notice that the order of the output and the order of the triplets does not matter.

class Solution:

def threeSum(self, nums: List[int]) -> List[List[int]]:

res = []

nums.sort()

for i in range(len(nums)):

if nums[i] > 0:

break

if i == 0 or nums[i - 1] != nums[i]:

self.twoSumII(nums, i, res)

return res

def twoSumII(self, nums: List[int], i: int, res: List[List[int]]):

lo, hi = i + 1, len(nums) - 1

while (lo < hi):

sum = nums[i] + nums[lo] + nums[hi]

if sum < 0:

lo += 1

elif sum > 0:

hi -= 1

else:

res.append([nums[i], nums[lo], nums[hi]])

lo += 1

hi -= 1

while lo < hi and nums[lo] == nums[lo - 1]:

lo += 1

</aside>

<aside> 💡 Question 5

Given an integer array nums and an integer k, return the kth largest element in the array. Note that it is the kth largest element in the sorted order, not the kth distinct element. Example 1: Input: nums = [3,2,1,5,6,4], k = 2 Output: 5

class Solution:

def findKthLargest(self, nums, k):

"""

:type nums: List[int]

:type k: int

:rtype: int

"""

return heapq.nlargest(k, nums)[-1]

</aside>

<aside> 💡 Question 6

Given an integer array nums and an integer k, return the kth smallest element in the array. Note that it is the kth smallest element in the sorted order, not the kth distinct element. Example 1: Input: nums = [3,2,1,5,6,4], k = 2 Output: 2

</aside>

import heapq

def find\_kth\_smallest(nums, k):

return heapq.nsmallest(k, nums)[-1]