ASSIGNMENT -4

ARRAYS

QUESTION 1

Given three integer arrays arr1, arr2 and arr3 **sorted** in **strictly increasing** order, return a sorted array of **only** the integers that appeared in **all** three arrays.

Example 1: Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]

Output: [1,5]

Explanation: Only 1 and 5 appeared in the three arrays.

SOLUTION:

Algorithm: - Binary Search (two pointers)

1) Take two pointers left and right
2) Use a loop to iterate and find the mid value.
3) If the mid value is greater than val, assign to right pointer.
4) Else, increment and assign to left pointer and return.
5) Create a list and use a loop to compare between the different arrays.
6) Append the values that are common in all the arrays to a list and return it.
7) TC: **O(logn)**, SC: **O(n)**

Code:

```
class Solution:
    def arraysIntersection(self, arr1: list[int], arr2: list[int], arr3:
list[int]) -> list[int]:
        def find(arr, val):
            left, right = 0, len(arr) - 1
            while left < right:
                mid = (left + right) >> 1
                if arr[mid] >= val:
                    right = mid
                else:
                    left = mid + 1
            return arr[left] == val

        res = []
        for num in arr1:
            if find(arr2, num) and find(arr3, num):
                res.append(num)
        return res
arr1 = [1,2,3,4,5]
arr2 = [1,2,5,7,9]
arr3 = [1,3,4,5,8]
```

```
ans = Solution()
res = ans.arraysIntersection(arr1,arr2,arr3)
print(res)
```

QUESTION 2

Given two **0-indexed** integer arrays nums1 and nums2, return *a list* answer *of size* 2 *where:*

- answer[0] *is a list of all* ***distinct*** *integers in* nums1 *which are* ***not*** *present in* nums2*.*
- answer[1] *is a list of all* ***distinct*** *integers in* nums2 *which are* ***not*** *present in* nums1.

Note that the integers in the lists may be returned in **any** order.

Example 1:

Input: nums1 = [1,2,3], nums2 = [2,4,6]

Output: [[1,3],[4,6]]

Explanation:

For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] = 3 are not present in nums2. Therefore, answer[0] = [1,3].

For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums2. Therefore, answer[1] = [4,6].

SOLUTION:

Algorithm:

1) Create a set to add just the unique elements from both num1 and nums2.
2) Use a list comprehension to find the unique elements from both the arrays and return them.
3) TC: **O(n)**, SC: **O(n)**

Code:

```
class Solution:
    def findDifference(self, nums1: List[int], nums2: List[int]) -> 
List[List[int]]:
        n1=set(nums1)
        n2=set(nums2)
        r1=list(set(x for x in nums1 if x not in n2))
        r2=list(set(x for x in nums2 if x not in n1))
        return [r1,r2]
```

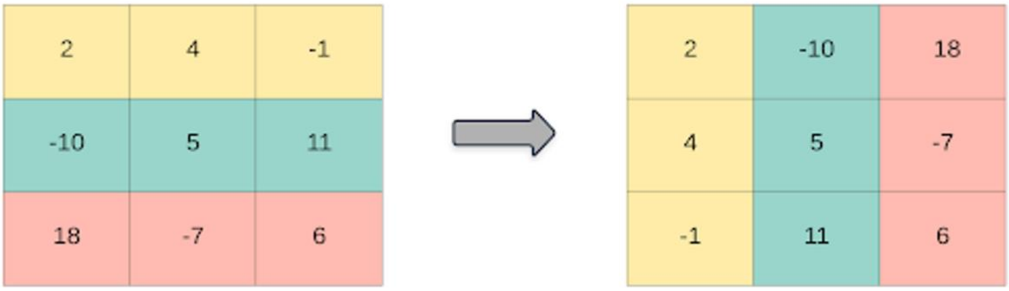
QUESTION 3

Given a 2D integer array matrix, return *the* ***transpose*** *of* matrix. The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

Example 1:

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[1,4,7],[2,5,8],[3,6,9]]



2	4	-1
-10	5	11
18	-7	6

2	-10	18
4	5	-7
-1	11	6

SOLUTION:

Algorithm:

1) Use a list comprehension and exchange the rows and column and return it.
2) TC: **O(n)**, SC: **O(1)**

Code:

```
class Solution:
    def transpose(self, matrix: List[List[int]]) -> List[List[int]]:
        return [[matrix[y][x] for y in range(len(matrix))] for x in
range(len(matrix[0]))]
```

QUESTION 4

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that the sum of min(ai, bi) for all i is **maximized**. Return *the maximized sum*.

Example 1:

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

Output: 4

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4

So the maximum possible sum is 4.

SOLUTION:

Algorithm:

1) Sort the array.
2) Use a loop to iterate over the array with a jump step size of 2.
3) Add the elements to the variable res and return it.
4) TC: O(nlogn), SC: O(1)

Code:

```
class Solution:
    def arrayPairSum(self, nums: List[int]) -> int:
        nums.sort()
        res=0
        for i in range(0,len(nums),2):
            res = res+nums[i]
        return res
```

QUESTION 5

You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase **may be** incomplete.

Given the integer n, return *the number of **complete rows** of the staircase you will build*.

SOLUTION:

Algorithm:

1) Take two variables and assign the values, initialize the value of res to 0.
2) Use a loop to iterate over the loop and find the mid value.
3) Find the value (mid/2)*(mid+1) and assign to count variable.
4) If the value of count is less than the value n then assign value of mid to res.
5) And find the max of both res,mid and keep incrementing the value of low.
6) Else, decrement the value total by 1. Finally return it.
7) TC: **O(logn)**, SC: **O(1)**

Code:

```
class Solution:
            #approach 1 binary solution
    def arrangeCoins(self, n: int) -> int:
        # return int((math.sqrt(8 * n + 1)-1)/2)
        # approach 3
        low,total=1,n
        res=0
        while low<=total:
            mid=(low+total)//2
            count=(mid/2)*(mid+1)
            if count<=n:
                res=mid
                max(res,mid)
                low=mid+1
            else:
                total=mid-1
        return res
```

QUESTION 6

Given an integer array nums sorted in **non-decreasing** order, return *an array of **the squares of each number** sorted in non-decreasing order*.

Example 1:

Input: nums = [-4,-1,0,3,10]

Output: [0,1,9,16,100]

Explanation: After squaring, the array becomes [16,1,0,9,100]. After sorting, it becomes [0,1,9,16,100]

SOLUTION:

Algorithm:

1) Sort the array of numbers.
2) Find the square of the elements in the array and return it.
3) TC: **O(nlogn),** SC: **O(1)**

Code:

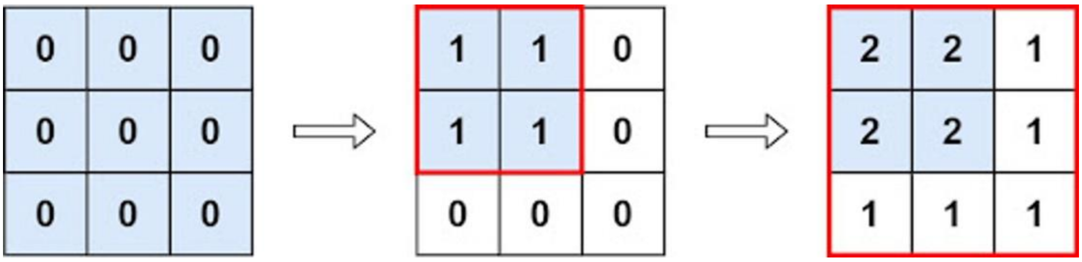
```
class Solution:
      def sortedSquares(self, A: List[int]) -> List[int]:
        return sorted([v**2 for v in A])
```

QUESTION 7

You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return *the number of maximum integers in the matrix after performing all the operations*

Example 1:



Input: m = 3, n = 3, ops = [[2,2],[3,3]]

Output: 4

Explanation: The maximum integer in M is 2, and there are four of it in M. So return 4.

SOLUTION:

Algorithm:

1) Initialize two variables for m and n.
2) Use a loop to iterate over the elements and find the min of min_row and ops.
3) Similarly find the min for the column elements.
4) Return the value of min_row*min_col.
5) TC: **O(n)**, SC: **O(1)**

Code:

```
class Solution:
    def maxCount(self, m: int, n: int, ops: List[List[int]]) -> int:
        min_row = m
        min_col = n
        for i in range(len(ops)):
            min_row=min(min_row, ops[i][0])
            min_col=min(min_col, ops[i][1])
        return min_row*min_col
```

QUESTION 8

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn]. *Return the array in the form* [x1,y1,x2,y2,...,xn,yn].

Example 1:

Input: nums = [2,5,1,3,4,7], n = 3

Output: [2,3,5,4,1,7]

Explanation: Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

SOLUTION:

Algorithm:

1) Create three empty arrays.
2) Use loops to iterate over the elements of the array and append it to the above created empty arrays.
3) Finally, return the arr3 element.
4) TC: **O(n)**, SC: O(1)

Code:

```
class Solution:
    def shuffle(self, nums: List[int], n: int) -> List[int]:
        arr1=[]
        arr2=[]
        arr3=[]
        for i in range(n):
            arr1.append(nums[i])
        for i in range(n,2*n):
            arr2.append(nums[i])
        for i in range(n):
            arr3.append(arr1[i])
            arr3.append(arr2[i])
        return arr3
```