**Assignment – 4**

Q1. 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.

Sol.

def arraysIntersection(arr1, arr2, arr3):

p1, p2, p3 = 0, 0, 0

result = []

while p1 < len(arr1) and p2 < len(arr2) and p3 < len(arr3):

if arr1[p1] == arr2[p2] == arr3[p3]:

result.append(arr1[p1])

p1 += 1

p2 += 1

p3 += 1

else:

smallest = min(arr1[p1], arr2[p2], arr3[p3])

if arr1[p1] == smallest:

p1 += 1

if arr2[p2] == smallest:

p2 += 1

if arr3[p3] == smallest:

p3 += 1

return result

Q2. 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].

Sol.

def findDisjoint(nums1, nums2):

set1 = set(nums1)

set2 = set(nums2)

diff1 = set1 - set2

diff2 = set2 - set1

return [list(diff1), list(diff2)]

Q3. 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]]

Sol.

def transpose(matrix):

n = len(matrix[0])

m = len(matrix)

transpose = [[0] \* m for \_ in range(n)]

for row in range(m):

for col in range(n):

transpose[col][row] = matrix[row][col]

return transpose

Q4. 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.

Sol.

def arrayPairSum(nums):

nums.sort()

sum\_min = 0

for i in range(0, len(nums), 2):

sum\_min += nums[i]

return sum\_min

Q5. 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*.

**Example 1:**

**Input:** n = 5

**Output:** 2

**Explanation:** Because the 3rd row is incomplete, we return 2.

Sol.

def arrangeCoins(n):

row = 1

coins = n

while coins >= row:

coins -= row

row += 1

return row - 1