**Assignment 5 - 2D Arrays**

**Question 1**

**Sol.**

def construct2DArray(original, m, n):

if len(original) != m \* n:

return []

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

for i in range(len(original)):

row = i // n

col = i % n

result[row][col] = original[i]

return result

def construct2DArray(original, m, n):

if len(original) != m \* n:

return []

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

for i in range(len(original)):

row = i // n

col = i % n

result[row][col] = original[i]

return result

[[1, 2], [3, 4]]

**Question 2**

**Sol:**

def arrangeCoins(n):

left = 0

right = n

while left <= right:

mid = (left + right) // 2

coins\_needed = mid \* (mid + 1) // 2

if coins\_needed <= n:

left = mid + 1

else:

right = mid - 1

return right

n = 5

result = arrangeCoins(n)

print(result)

2

**Question 3**

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].

**Sol:**

def sortedSquares(nums):

result = []

for num in nums:

result.append(num \*\* 2)

result.sort()

return result

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

result = sortedSquares(nums)

print(result)

[0, 1, 9, 16, 100]

**Question 4**

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 findDisappearedNumbers(nums1, nums2):

set1 = set(nums1)

set2 = set(nums2)

answer1 = set1 - set2

answer2 = set2 - set1

return [list(answer1), list(answer2)]

nums1 = [1, 2, 3]

nums2 = [2, 4, 6]

result = findDisappearedNumbers(nums1, nums2)

print(result)

[[1, 3], [4, 6]]

**Question 5**

Given two integer arrays arr1 and arr2, and the integer d, *return the distance value between the two arrays*.

The distance value is defined as the number of elements arr1[i] such that there is not any element arr2[j] where |arr1[i]-arr2[j]| <= d.

**Example 1:**

**Input:** arr1 = [4,5,8], arr2 = [10,9,1,8], d = 2

**Output:** 2

**Explanation:**

For arr1[0]=4 we have:

|4-10|=6 > d=2

|4-9|=5 > d=2

|4-1|=3 > d=2

|4-8|=4 > d=2

For arr1[1]=5 we have:

|5-10|=5 > d=2

|5-9|=4 > d=2

|5-1|=4 > d=2

|5-8|=3 > d=2

For arr1[2]=8 we have:

**|8-10|=2 <= d=2**

**|8-9|=1 <= d=2**

|8-1|=7 > d=2

**|8-8|=0 <= d=2**

**Sol:**

def findTheDistanceValue(arr1, arr2, d):

distance = 0

for num1 in arr1:

for num2 in arr2:

if abs(num1 - num2) <= d:

break

else:

distance += 1

return distance

arr1 = [4, 5, 8]

arr2 = [10, 9, 1, 8]

d = 2

result = findTheDistanceValue(arr1, arr2, d)

print(result)

2

**Question 6**

Given an integer array nums of length n where all the integers of nums are in the range [1, n] and each integer appears **once** or **twice**, return *an array of all the integers that appears* ***twice***.

You must write an algorithm that runs in O(n) time and uses only constant extra space.

**Example 1:**

**Input:** nums = [4,3,2,7,8,2,3,1]

**Output:**

[2,3]

**Sol:**

def findDuplicates(nums):

result = []

for num in nums:

idx = abs(num) - 1

if nums[idx] < 0:

result.append(abs(num))

else:

nums[idx] = -nums[idx]

return result

nums = [4, 3, 2, 7, 8, 2, 3, 1]

result = findDuplicates(nums)

print(result)

[2, 3]

**Question 7**

Suppose an array of length n sorted in ascending order is **rotated** between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

* [4,5,6,7,0,1,2] if it was rotated 4 times.
* [0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that **rotating** an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Given the sorted rotated array nums of **unique** elements, return *the minimum element of this array*.

You must write an algorithm that runs in O(log n) time.

**Example 1:**

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

**Output:** 1

**Explanation:**

The original array was [1,2,3,4,5] rotated 3 times.

**Sol:**

def findMin(nums):

left = 0

right = len(nums) - 1

while left < right:

mid = (left + right) // 2

if nums[mid] < nums[mid - 1] and nums[mid] < nums[mid + 1]:

return nums[mid]

elif nums[mid] < nums[right]:

right = mid

else:

left = mid + 1

return nums[left]

nums = [3, 4, 5, 1, 2]

result = findMin(nums)

print(result)

1

**Question 8**

An integer array original is transformed into a doubled array changed by appending twice the value of every element in original, and then randomly shuffling the resulting array.

Given an array changed, return original *if* changed *is a doubled array. If* changed *is not a doubled array, return an empty array. The elements in* original *may be returned in any order*.

**Example 1:**

**Input: changed =** [1,3,4,2,6,8]

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

**Explanation:** One possible original array could be [1,3,4]:

* Twice the value of 1 is 1 \* 2 = 2.
* Twice the value of 3 is 3 \* 2 = 6.
* Twice the value of 4 is 4 \* 2 = 8.

Other original arrays could be [4,3,1] or [3,1,4].

**Sol:**

from collections import defaultdict

def findOriginalArray(changed):

counter = defaultdict(int)

for num in changed:

counter[num] += 1

changed.sort()

original = []

for num in changed:

if counter[num] == 0:

continue

counter[num] -= 1

if counter[num] < 0:

return []

original.append(num)

return original

changed = [1, 3, 4, 2, 6, 8]

result = findOriginalArray(changed)

print(result)

[1, 3, 4]