Assignmen 04 Solutions

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.

In [1]:

```
def find_common_elements(arr1, arr2, arr3):
    p1 = p2 = p3 = 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
        elif arr1[p1] <= arr2[p2] and arr1[p1] <= arr3[p3]:
            p1 += 1
        elif arr2[p2] <= arr1[p1] and arr2[p2] <= arr3[p3]:
            p2 += 1
        else:
            p3 += 1

    return result
```

In [3]:

```
arr1 = [1, 2, 3, 4, 5]
arr2 = [1, 2, 5, 7, 9]
arr3 = [1, 3, 4, 5, 8]

result = find_common_elements(arr1, arr2, arr3)
print(result)
```

```
[1, 5]
```

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].

In [4]:

```
def find_missing_elements(nums1, nums2):
    set1 = set(nums1)
    set2 = set(nums2)

    result1 = [num for num in nums1 if num not in set2]
    result2 = [num for num in nums2 if num not in set1]

    return [result1, result2]
```

In [5]:

```
nums1 = [1, 2, 3]
```

```
nums2 = [2, 4, 6]

result = find_missing_elements(nums1, nums2)
print(result)
```

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

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.

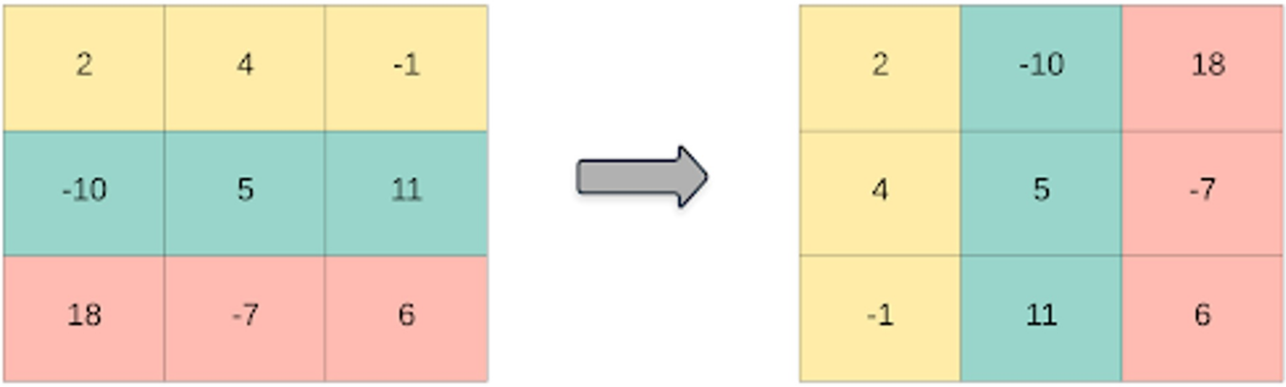
Example 1:

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

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

In [6]:
```
from IPython import display
display.Image(r"C:\Users\hrush\OneDrive\Pictures\Saved Pictures\iamge_v3.png")
```

Out[6]:



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

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

In [7]:
```
def transpose(matrix):
    rows = len(matrix)
    cols = len(matrix[0])
    transposed = [[0] * rows for _ in range(cols)]

    for i in range(rows):
        for j in range(cols):
            transposed[j][i] = matrix[i][j]

    return transposed
```

In [8]:
```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
result = transpose(matrix)
print(result)
```

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

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

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

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

So the maximum possible sum is 4.

In [9]:
```
def array_pair_sum(nums):
    nums.sort()
    result = 0
    for i in range(0, len(nums), 2):
        result += nums[i]
    return result
```

In [10]:

```
nums = [1, 4, 3, 2]
result = array_pair_sum(nums)
print(result)
```

```
4
```

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

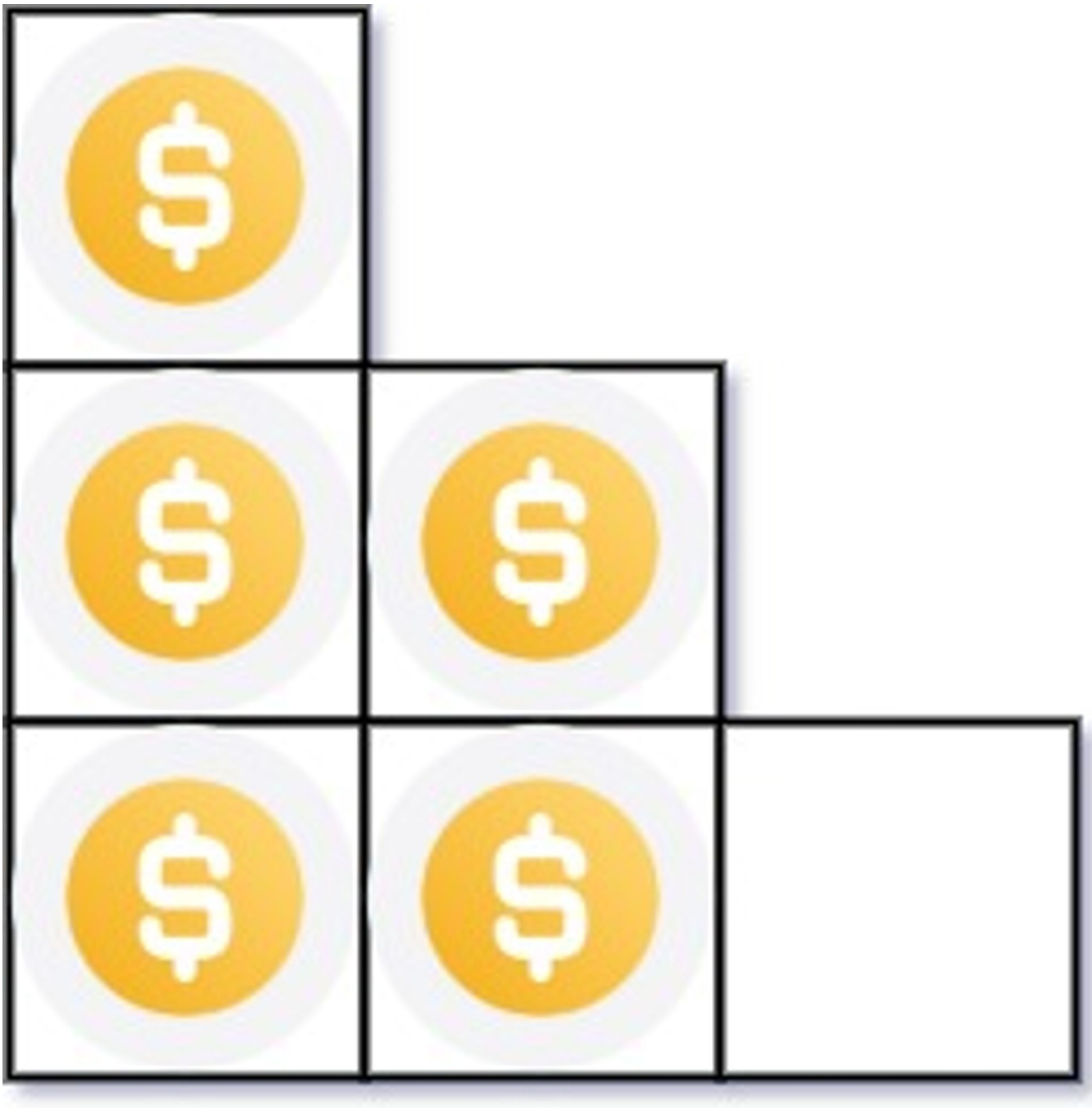
Output: 2

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

In [16]:

```
from IPython import display
display.Image(r"C:\Users\hrush\OneDrive\Pictures\Saved Pictures\v2.jpg")
```

Out[16]:



In [11]:

```
def arrange_coins(n):
    complete_rows = 0
    remaining_coins = n
    i = 1

    while remaining_coins >= i:
        complete_rows += 1
        remaining_coins -= i
        i += 1

    return complete_rows
```

In [12]:

```
n = 5
result = arrange_coins(n)
print(result)
```

```
2
```

Q6. 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]

In [13]:

```
def sorted_squares(nums):
    squared_nums = [num * num for num in nums]
    squared_nums.sort()
    return squared_nums
```

In [14]:

```
nums = [-4, -1, 0, 3, 10]
result = sorted_squares(nums)
print(result)
```

```
[0, 1, 9, 16, 100]
```

Q7. 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]]

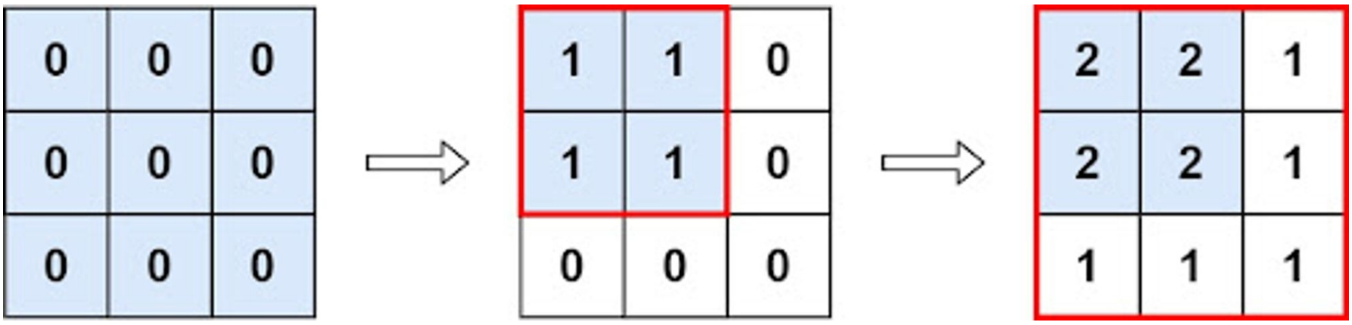
Output: 4

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

In [17]:

```
from IPython import display
display.Image(r"C:\Users\hrush\OneDrive\Pictures\Saved Pictures\q4.jpg")
```

Out[17]:



0	0	0
0	0	0
0	0	0

1	1	0
1	1	0
0	0	0

2	2	1
2	2	1
1	1	1

In [18]:

```
from typing import List
def maxCount(m: int, n: int, ops: List[List[int]]) -> int:
    min_ai = m
    min_bi = n

    for op in ops:
        min_ai = min(min_ai, op[0])
        min_bi = min(min_bi, op[1])

    return min_ai * min_bi
```

In [19]:

```
m = 3
```

```
n = 3
ops = [[2,2],[3,3]]

result = maxCount(m, n, ops)
print(result)
```

```
4
```

Q8. 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].

In [20]:

```
def rearrange_array(nums, n):
    result = []
    for i in range(n):
        result.append(nums[i])
        result.append(nums[i+n])
    return result
```

In [21]:

```
nums = [2, 5, 1, 3, 4, 7]
n = 3
print(rearrange_array(nums, n))
```

```
[2, 3, 5, 4, 1, 7]
```

In [ ]: