

Day – 2 : Arrays- II


Problem 1: Given a matrix, your task is to rotate the matrix 90 degrees clockwise.

```
def rotate_matrix(matrix):  
    n = len(matrix)  
    for i in range(n):  
        for j in range(i, n):  
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]  
  
    for i in range(n):  
        matrix[i] = matrix[i][::-1]  
  
    return matrix
```

Example 1

```
arr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print(rotate_matrix(arr))  
print("Rotated Image")  
for i in range(len(arr)):  
    for j in range(len(arr[0])):  
        print(arr[i][j], end=" ")  
    print()
```

```
12 # Example 1
13 arr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
14 print(rotate_matrix(arr))
15 print("Rotated Image")
16 for i in range(len(arr)):
17     for j in range(len(arr[0])):
18         print(arr[i][j], end=" ")
19     print()
20
21
22
23
```



```
Rotated Image
7 4 1
8 5 2
9 6 3
```

Problem -2: Given an array of intervals, merge all the overlapping intervals and return an array of non-overlapping intervals.

```
def merge_intervals(intervals):
    if not intervals:
        return []

    intervals.sort(key=lambda x: x[0])
    merged = [intervals[0]]

    for interval in intervals[1:]:
        if interval[0] <= merged[-1][1]:
            merged[-1][1] = max(merged[-1][1], interval[1])
        else:
            merged.append(interval)

    return merged

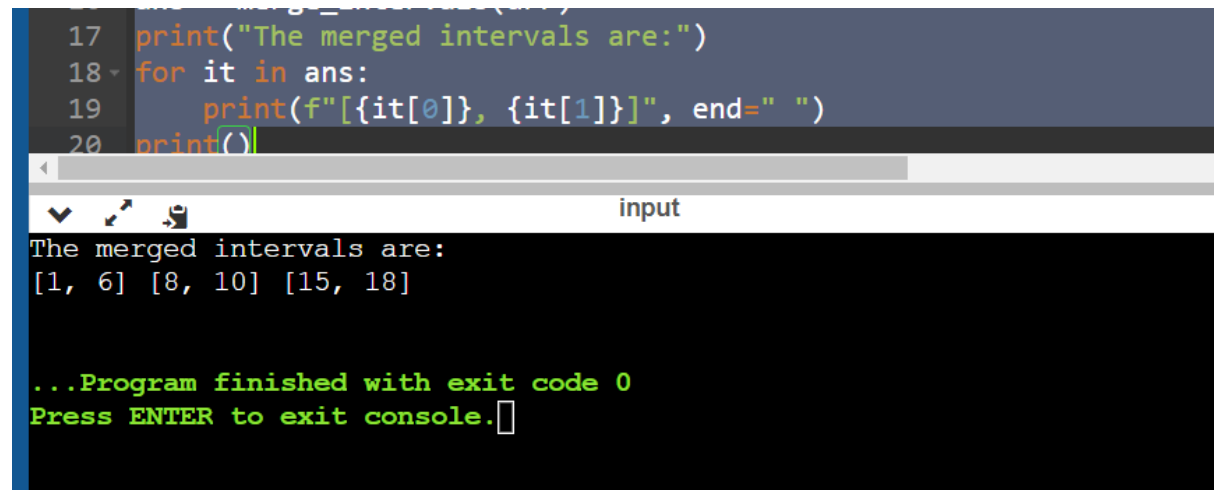
arr = [[1, 3], [8, 10], [2, 6], [15, 18]]
ans = merge_intervals(arr)
```

```
print("The merged intervals are:")
```

```
for it in ans:
```

```
    print(f"[{it[0]}, {it[1]]", end=" ")
```

```
print()
```



```
17 print("The merged intervals are:")
18 for it in ans:
19     print(f"[{it[0]}, {it[1]]", end=" ")
20 print()

input

The merged intervals are:
[1, 6] [8, 10] [15, 18]

...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 3: Given two sorted arrays **arr1[]** and **arr2[]** of sizes **n** and **m** in non-decreasing order. Merge them in sorted order. Modify arr1 so that it contains the first N elements and modify arr2 so that it contains the last M elements.

```
def merge_sorted_arrays(arr1, arr2):
```

```
    n=len(arr1)
```

```
    m=len(arr2)
```

```
    left = n - 1
```

```
    right = 0
```

```
    while left >= 0 and right < m:
```

```
        if arr1[left] > arr2[right]:
```

```
            arr1[left], arr2[right] = arr2[right], arr1[left]
```

```
            left -= 1
```

```
            right += 1
```

```
        else:
```

```

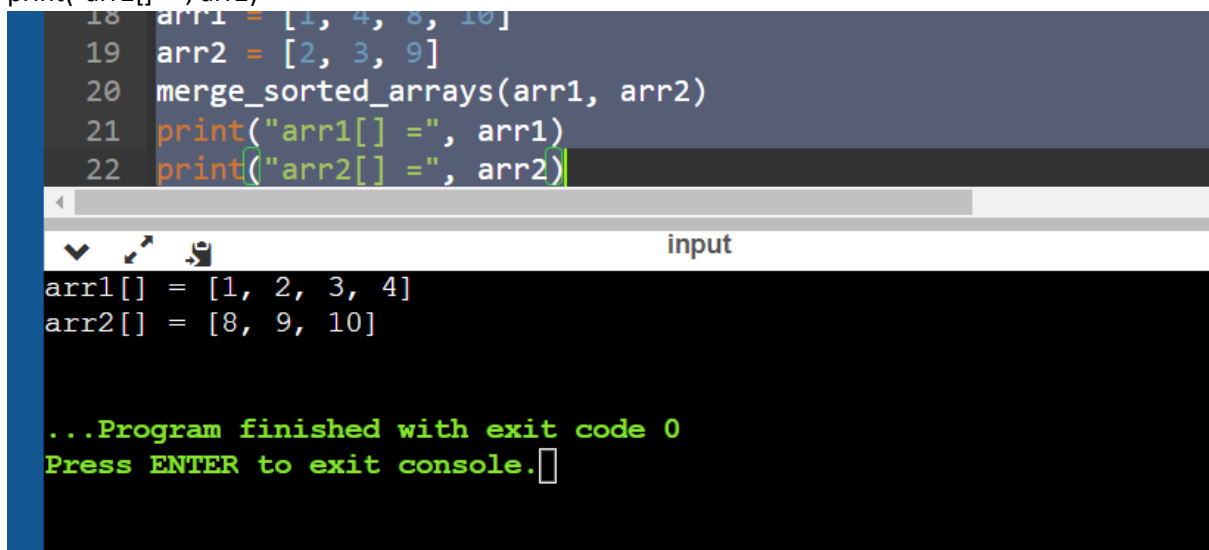
        break

arr1.sort()

arr2.sort()


arr1 = [1, 4, 8, 10]
arr2 = [2, 3, 9]
merge_sorted_arrays(arr1, arr2)
print("arr1[] =", arr1)
print("arr2[] =", arr2)

```



The screenshot shows a code editor with the following Python code:

```

18 arr1 = [1, 4, 8, 10]
19 arr2 = [2, 3, 9]
20 merge_sorted_arrays(arr1, arr2)
21 print("arr1[] =", arr1)
22 print("arr2[] =", arr2)

```

Below the code editor is a terminal window titled "input". It displays the output of the program:

```

arr1[] = [1, 2, 3, 4]
arr2[] = [8, 9, 10]

...Program finished with exit code 0
Press ENTER to exit console.

```

Problem 4: Given an array of $N + 1$ size, where each element is between 1 and N . Assuming there is only one duplicate number, your task is to find the duplicate number.

```

def find_duplicate(arr):

    slow = arr[0]

    fast = arr[0]

    while True:

        slow = arr[slow]

        fast = arr[arr[fast]]

        if slow == fast:

```

```
break
```

```
slow = arr[0]
```

```
while slow != fast:
```

```
    slow = arr[slow]
```

```
    fast = arr[fast]
```

```
return slow
```

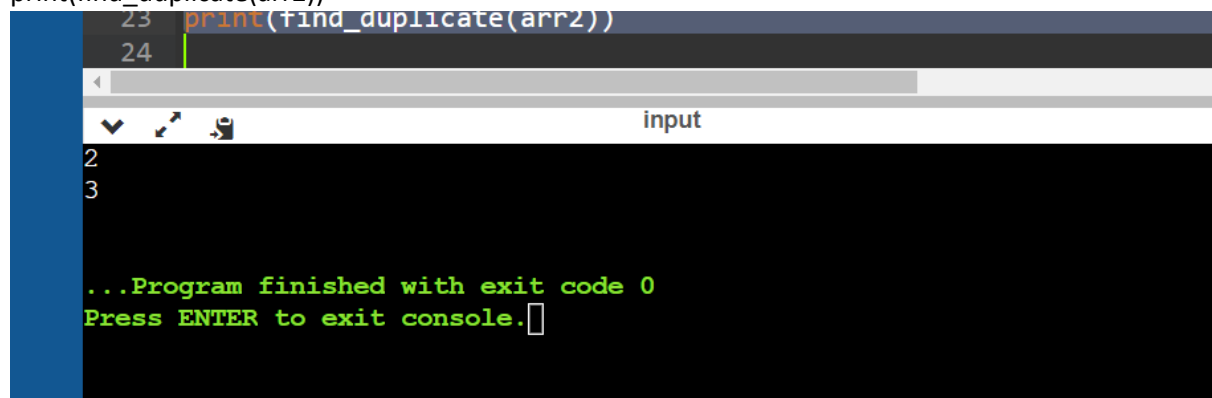
```
# Test cases
```

```
arr1 = [1, 3, 4, 2, 2]
```

```
print(find_duplicate(arr1))
```

```
arr2 = [3, 1, 3, 4, 2]
```

```
print(find_duplicate(arr2))
```

A screenshot of a code editor and terminal. The code editor shows two lines of Python code: line 23: `print(find_duplicate(arr2))` and line 24: `print(find_duplicate(arr2))`. The terminal window below shows the output of the program. It displays the number '2' on the first line and '3' on the second line. Below the output, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor.

Problem 5: You are given a read-only array of N integers with values also in the range [1, N] both inclusive. Each integer appears exactly once except A which appears twice and B which is missing. The task is to find the repeating and missing numbers A and B where A repeats twice and B is missing.

```
def findRepeatingAndMissing(array):
```

```
    N = len(array)
```

```

sumOfArray = 0
sumOfSquares = 0

for i in range(N):
    sumOfArray += array[i]
    sumOfSquares += array[i] * array[i]

sumOfIntegers = N * (N + 1) // 2
sumOfSquaresIntegers = N * (N + 1) * (2 * N + 1) // 6

diff = sumOfArray - sumOfIntegers
diffSquares = sumOfSquares - sumOfSquaresIntegers

A = (diffSquares // diff + diff) // 2
B = A - diff

return (A, B)

array1 = [3, 1, 2, 5, 3]
result1 = findRepeatingAndMissing(array1)
print(result1)

```

```

22 print(result1)
(3, 4)
...Program finished with exit code 0
Press ENTER to exit console.

```

Problem Statement: Given an array of N integers, count the inversion of the array (using [merge-sort](#)).

What is an inversion of an array? Definition: for all i & $j < \text{size of array}$, if $i < j$ then you have to find pair $(A[i], A[j])$ such that $A[j] < A[i]$.

```
def mergeSortAndCountInversions(array):
```

```
    inversions = 0
```

```
    if len(array) <= 1:
```

```
        return inversions
```

```
    mid = len(array) // 2
```

```
    left_half = array[:mid]
```

```
    right_half = array[mid:]
```

```
    inversions += mergeSortAndCountInversions(left_half)
```

```
    inversions += mergeSortAndCountInversions(right_half)
```

```
    i = j = 0
```

```
    while i < len(left_half) and j < len(right_half):
```

```
        if left_half[i] > right_half[j]:
```

```
            inversions += mid - i
```

```
            array[i + j] = right_half[j]
```

```
            j += 1
```

```
        else:
```

```
            array[i + j] = left_half[i]
```

```
            i += 1
```

```
    while i < len(left_half):
```

```
        array[i + j] = left_half[i]
```

```
        i += 1
```

```
    while j < len(right_half):
```

```
        array[i + j] = right_half[j]
```

```
j += 1
```

```
return inversions
```

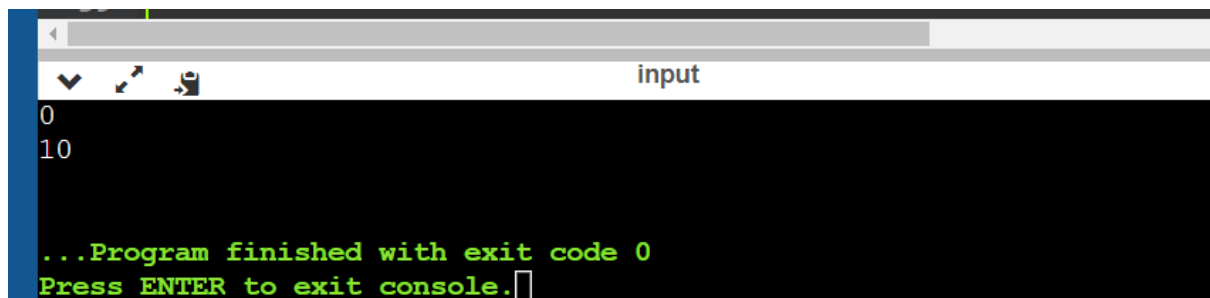
```
# Test cases
```

```
array1 = [1, 2, 3, 4, 5]
```

```
print(mergeSortAndCountInversions(array1))
```

```
array2 = [5, 4, 3, 2, 1]
```

```
print(mergeSortAndCountInversions(array2))
```

A screenshot of a terminal window with a dark background. The title bar at the top is light gray and contains the word "input". Below the title bar, there are three small icons: a checkmark, a magnifying glass, and a document. The terminal output shows the number "0" on the first line and "10" on the second line. At the bottom, there is a green text message that reads "...Program finished with exit code 0" followed by "Press ENTER to exit console." and a cursor icon.

```
0
10

...Program finished with exit code 0
Press ENTER to exit console.
```