Topic-3 Divide and Conquer

1: Find Maximum and Minimum in Array

Aim:

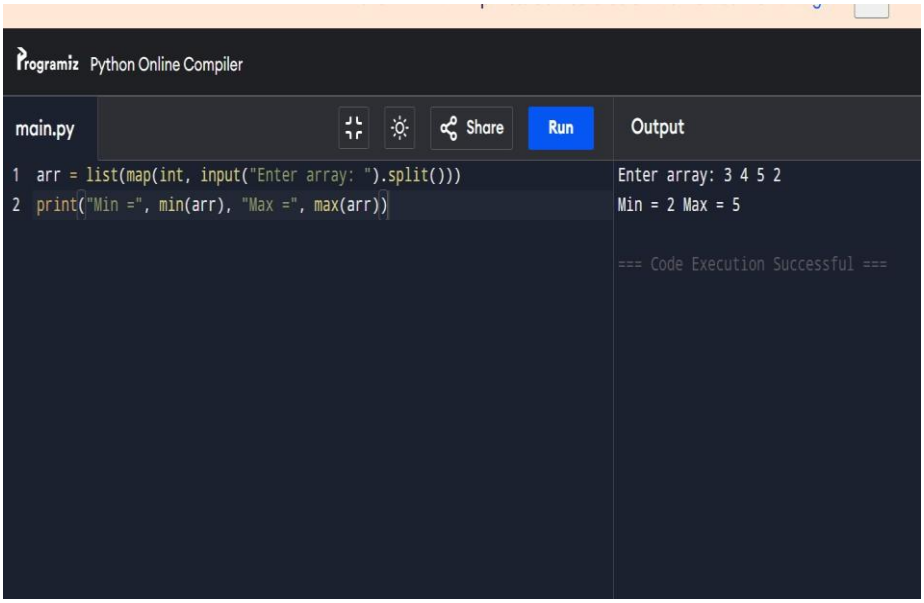
To find both the maximum and minimum elements in an array.

Algorithm:

1. Read size N and array elements.
2. Initialize min and max with first element.
3. Traverse array, update min if smaller and max if larger.
4. Print final values. **Code**

arr = list(map(int, input("Enter array: ").split()))

print("Min =", min(arr)) print("Max =", max(arr)) **Output:**



Result: The program has been successfully executed.

2.Min and Max in Sorted Array

Aim:

To find min and max in a sorted array.

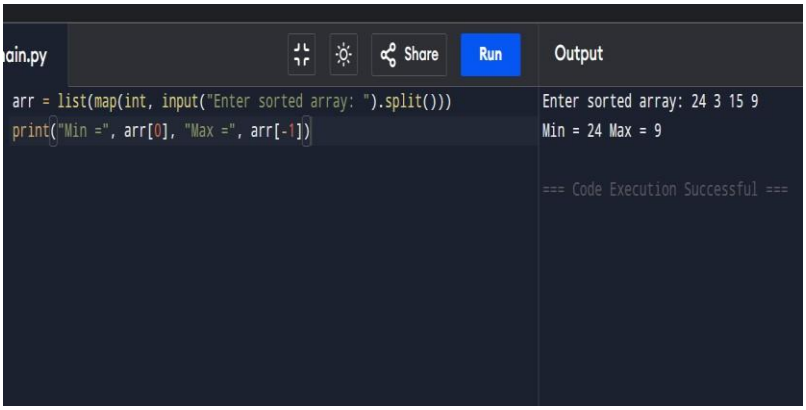
Algorithm:

1. In a sorted array, min = first element, max = last element.
2. Print both values.

Code:

arr = list(map(int, input("Enter sorted array: ").split()))

print("Min =", arr[0]) print("Max =", arr[-1]) **Output:**



Result: The program has been successfully executed.

3: Merge Sort

Aim:

To sort an unsorted array using Merge Sort.

Algorithm:

1. Divide array into two halves.
2. Recursively sort both halves.
3. Merge sorted halves.

Code:

def merge_sort(arr): if len(arr)

> 1: mid = len(arr)//2 L =

arr[:mid] R = arr[mid:]

merge_sort(L) merge_sort(R)

i=j=k=0 while i < len(L) and j

< len(R): if L[i] < R[j]:

arr[k] = L[i];

i+=1 else:

arr[k] = R[j]; j+=1

k+=1 while i < len(L):

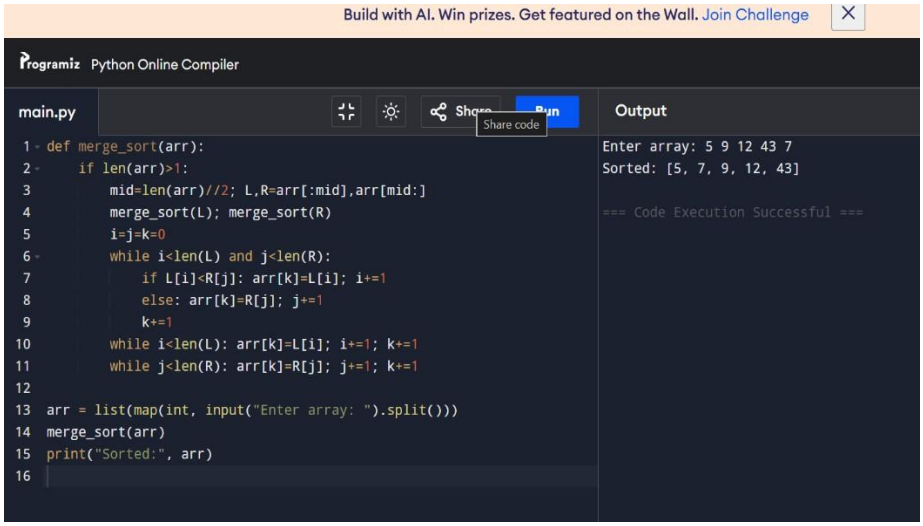
arr[k] = L[i]; i+=1; k+=1

while j < len(R):

arr[k] = R[j]; j+=1; k+=1 arr = list(map(int,

input("Enter array: ").split())) merge_sort(arr)

print("Sorted:", arr) Output:



Result: The program has been successfully executed**.**

4: Merge Sort with Comparisons

Aim:

To sort array using Merge Sort and count comparisons**.**

Algorithm:

1. In a sorted array, min = first element, max = last element.
2. Print both values.

Code:

comparisons = 0 def

merge_sort(arr): global

comparisons if len(arr) > 1:

mid = len(arr)//2 L, R =

arr[:mid], arr[mid:]

merge_sort(L) merge_sort(R)

i=j=k=0 while i < len(L) and j

< len(R):

comparisons += 1

if L[i] < R[j]:

arr[k] = L[i]; i+=1 else:

arr[k] = R[j]; j+=1 k+=1 while i <

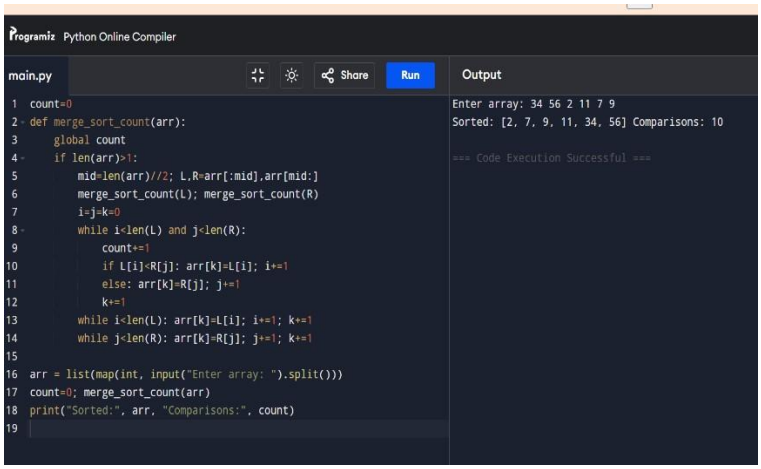
len(L): arr[k] = L[i]; i+=1; k+=1 while j <

len(R): arr[k] = R[j]; j+=1; k+=1 arr = list(map(int,

input("Enter array: ").split())) merge_sort(arr)

print("Sorted:", arr) print("Comparisons:",

comparisons) **Output:**



Result: The program has been successfully executed.

5: Quick Sort (First Element Pivot)

Aim:

To sort array using Quick Sort with first element as pivot.

Algorithm:

1. Choose first element as pivot.
2. Partition array into < pivot and > pivot.
3. Recursively quicksort subarrays.

Code: def quick_sort(arr): if len(arr) <= 1:

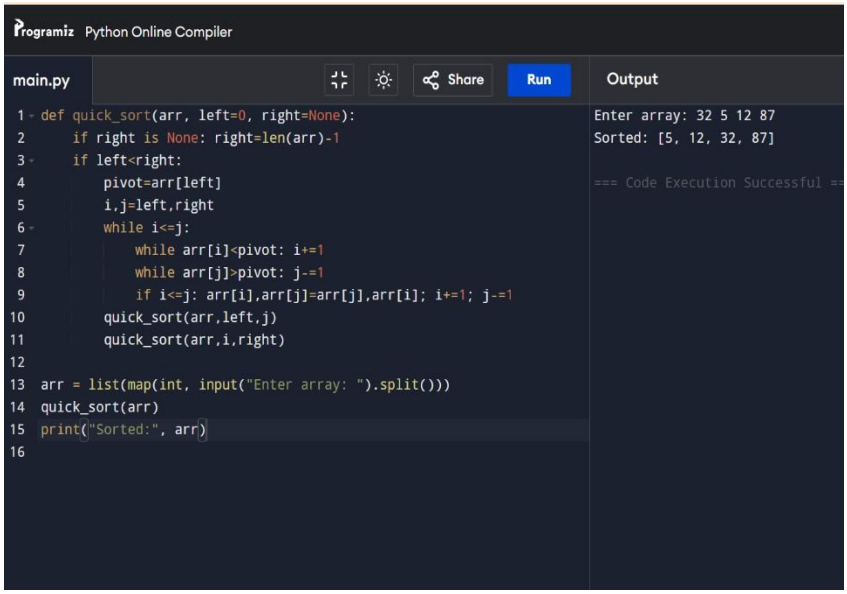
return arr pivot = arr[0] left = [x for x in arr[1:] if

x <= pivot] right = [x for x in arr[1:] if x > pivot]

return quick_sort(left) + [pivot] + quick_sort(right) arr

= list(map(int, input("Enter array: ").split()))

print("Sorted:", quick_sort(arr)) **Output:**



Result: The program has been successfully executed**.**