

## 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:**

```
Programiz Python Online Compiler
```

main.py	Run	Output
1 arr = list(map(int, input("Enter array: ").split())) 2 print("Min =", min(arr), "Max =", max(arr))		Enter array: 3 4 5 2 Min = 2 Max = 5 == Code Execution Successful ==

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

### 2. Min and Max in Sorted Array

#### Aim:

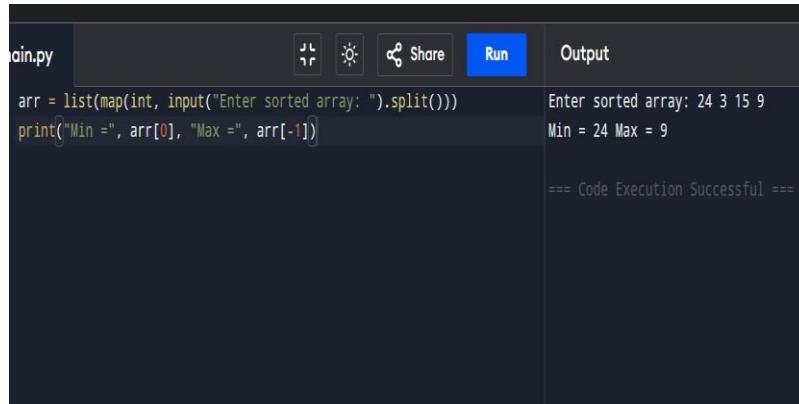
To find min and max in a sorted array.

#### 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])
```



```
main.py
arr = list(map(int, input("Enter sorted array: ").split()))
print("Min =", arr[0], "Max =", arr[-1])

Enter sorted array: 24 3 15 9
Min = 24 Max = 9
== Code Execution Successful ==
```

**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    while j < len(R):        arr[k] = R[j];        j+=1
```

```

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)

```

```

Build with AI. Win prizes. Get featured on the Wall. Join Challenge X
Programiz Python Online Compiler
main.py | Run Share code
1 - def merge_sort(arr):
2 -     if len(arr)>1:
3 -         mid=len(arr)//2; L,R=arr[:mid],arr[mid:]
4 -         merge_sort(L); merge_sort(R)
5 -         i=j=k=0
6 -         while i<len(L) and j<len(R):
7 -             if L[i]<R[j]: arr[k]=L[i]; i+=1
8 -             else: arr[k]=R[j]; j+=1
9 -             k+=1
10 -            while i<len(L): arr[k]=L[i]; i+=1; k+=1
11 -            while j<len(R): arr[k]=R[j]; j+=1; k+=1
12 -
13 arr = list(map(int, input("Enter array: ").split()))
14 merge_sort(arr)
15 print("Sorted:", arr)
16

```

Output

```

Enter array: 5 9 12 43 7
Sorted: [5, 7, 9, 12, 43]
== Code Execution Successful ==

```

**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:**

```

Programiz Python Online Compiler
main.py Run Output
1 count=0
2 def merge_sort(arr):
3     global count
4     if len(arr)>1:
5         mid=len(arr)//2; L,R=arr[:mid],arr[mid:]
6         merge_sort_count(L); merge_sort_count(R)
7         i=j=k=0
8         while i<len(L) and j<len(R):
9             count+=1
10            if L[i]<R[j]: arr[k]=L[i]; i+=1
11            else: arr[k]=R[j]; j+=1
12            k+=1
13         while i<len(L): arr[k]=L[i]; i+=1; k+=1
14         while j<len(R): arr[k]=R[j]; j+=1; k+=1
15
16 arr = list(map(int, input("Enter array: ").split())))
17 count=0; merge_sort_count(arr)
18 print("Sorted:", arr, "Comparisons:", count)
19

```

Enter array: 34 56 2 11 7 9  
Sorted: [2, 7, 9, 11, 34, 56] Comparisons: 10  
== Code Execution Successful ==

**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:**

The screenshot shows a Python code editor interface from Programiz. The code in the editor is:

```
main.py
1 def quick_sort(arr, left=0, right=None):
2     if right is None: right=len(arr)-1
3     if left<right:
4         pivot=arr[left]
5         i,j=left,right
6         while i<=j:
7             while arr[i]<pivot: i+=1
8             while arr[j]>pivot: j-=1
9             if i<=j: arr[i],arr[j]=arr[j],arr[i]; i+=1; j-=1
10        quick_sort(arr,left,j)
11        quick_sort(arr,i,right)
12
13 arr = list(map(int, input("Enter array: ").split()))
14 quick_sort(arr)
15 print("Sorted:", arr)
16
```

The output window shows the input "Enter array: 32 5 12 87" and the resulting sorted array "Sorted: [5, 12, 32, 87]". Below the output, a message indicates "Code Execution Successful".

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