

## COMPETITIVE PROGRAMMING

### ASSIGNMENT-2

Name: V. Vyshnavi

Hallticket No.:2303A51968

Batch-14

#### **Problem Statement :**

A warehouse records package priority scores as integers. To dispatch efficiently, you must sort the scores in non-decreasing order using merge sort (divide and conquer). For each test case, output the sorted list.

#### **Input Format:**

The first line contains integer T.

For each test case:

- First line: N
- Second line: N integers (priority scores)

#### **Output Format:**

For each test case, print the sorted array in one line (space-separated).

#### **Constraints:**

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$  (sum of N over all test cases  $\leq 200000$ )
- $-10^9 \leq A_i \leq 10^9$

#### **Sample Input**

1  
7  
4 1 6 2 5 3 2

#### **Expected Output**

1 2 2 3 4 5 6

#### **Python Code:**

```
def merge_sort(arr):  
    if len(arr) <= 1:
```

```
return arr

mid = len(arr) // 2
left = merge_sort(arr[:mid])
right = merge_sort(arr[mid:])

return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])

    return result

T = int(input())

for _ in range(T):
    N = int(input())
    arr = list(map(int, input().split()))
```

```
sorted_arr = merge_sort(arr)
```

```
print(*sorted_arr)
```

The screenshot shows the OnlineGDB interface with the Python language selected. The code in the editor is:

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result
```

The screenshot shows the OnlineGDB interface with the Python language selected. The code in the editor is:

```
def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result

T = int(input())
for _ in range(T):
    N = int(input())
    arr = list(map(int, input().split()))
    sorted_arr = merge_sort(arr)
    print(sorted_arr)
```

## Output:

The screenshot shows the terminal output of the program. The input was:

```
1
7
4 1 6 2 5 3 2
1 2 2 3 4 5 6
```

The output was:

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

## Java Code:

```
import java.util.*;  
  
public class Main {  
  
    static void mergeSort(long[] a, int l, int r) {  
        if (l >= r) return;  
  
        int m = (l + r) / 2;  
        mergeSort(a, l, m);  
        mergeSort(a, m + 1, r);  
        merge(a, l, m, r);  
    }  
  
    static void merge(long[] a, int l, int m, int r) {  
        long[] t = new long[r - l + 1];  
        int i = l, j = m + 1, k = 0;  
  
        while (i <= m && j <= r)  
            t[k++] = (a[i] <= a[j]) ? a[i++] : a[j++];  
  
        while (i <= m) t[k++] = a[i++];  
        while (j <= r) t[k++] = a[j++];  
  
        for (i = l, k = 0; i <= r; i++, k++)  
            a[i] = t[k];  
    }  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int T = sc.nextInt();  
  
        while (T-- > 0) {
```

```

int N = sc.nextInt();

long[] a = new long[N];

for (int i = 0; i < N; i++)
    a[i] = sc.nextLong();

mergeSort(a, 0, N - 1);

for (int i = 0; i < N; i++)
    System.out.print(a[i] + " ");

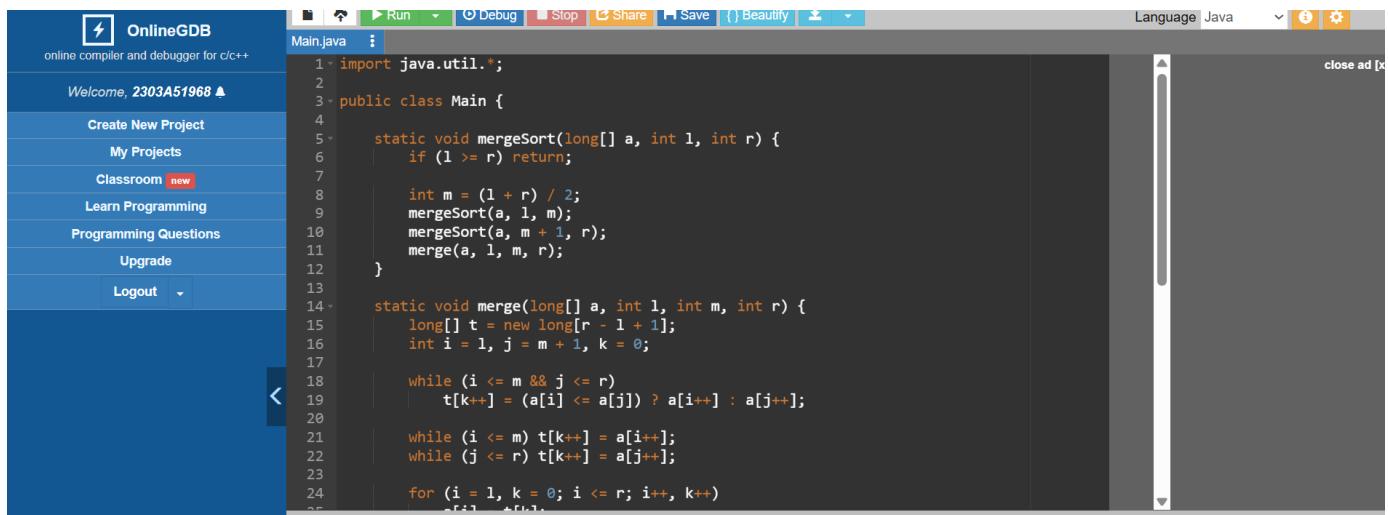
System.out.println();

}

sc.close();
}

}

```



The screenshot shows the OnlineGDB Java compiler interface. The left sidebar has a dark blue header with the OnlineGDB logo and a light blue footer with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main workspace has a light gray header with tabs for 'Main.java' (selected), 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and language dropdowns for 'Java'. The code editor area contains the following Java code:

```

import java.util.*;
public class Main {
    static void mergeSort(long[] a, int l, int r) {
        if (l >= r) return;
        int m = (l + r) / 2;
        mergeSort(a, l, m);
        mergeSort(a, m + 1, r);
        merge(a, l, m, r);
    }
    static void merge(long[] a, int l, int m, int r) {
        long[] t = new long[r - l + 1];
        int i = l, j = m + 1, k = 0;
        while (i <= m && j <= r)
            t[k++] = (a[i] <= a[j]) ? a[i++] : a[j++];
        while (i <= m) t[k++] = a[i++];
        while (j <= r) t[k++] = a[j++];
        for (i = l, k = 0; i <= r; i++, k++)
            a[i] = t[k];
    }
}

```

The screenshot shows the OnlineGDB interface. The left sidebar has options like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Upgrade', and 'Logout'. The main area shows a Java code editor with the file 'Main.java' containing the following code:

```
1  public class Main {
2      public static void main(String[] args) {
3          Scanner sc = new Scanner(System.in);
4          int T = sc.nextInt();
5
6          while (T-- > 0) {
7              int N = sc.nextInt();
8              long[] a = new long[N];
9
10             for (int i = 0; i < N; i++)
11                 a[i] = sc.nextLong();
12
13             mergeSort(a, 0, N - 1);
14
15             for (int i = 0; i < N; i++)
16                 System.out.print(a[i] + " ");
17             System.out.println();
18         }
19     }
20 }
```

## Output:

The terminal window shows the execution of the Java program. The input is provided on the first line, followed by two lines of test cases. The output shows the sorted arrays for each test case.

```
input
1
7
4 1 6 2 5 3 2
1 2 2 3 4 5 6

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

## Maximum Profit Streak (Divide and Conquer)

### Problem Statement:

A company tracks daily profit changes as an array A (values can be negative). A “profit streak” is any non-empty contiguous subarray. Find the maximum possible sum of a profit streak using a divide-and-conquer approach (split into left, right, and crossing subproblems).

### Input Format:

The first line contains integer T. For each test case:

- First line: N
- Second line: N integers A1.

**AN Output Format:** For each test case, print one integer: maximum subarray sum.

### **Constraints**

- $1 \leq T \leq 20$
- $1 \leq N \leq 200000$  (sum of N over all test cases  $\leq 200000$ )
- $-10^9 \leq A_i \leq 10^9$

### **Sample Input**

1  
9

-2 1 -3 4 -1 2 1 -5 4

**Expected Output : 6**

**Python code:**

```
def max_sum(arr, l, r):  
    if l == r: return arr[l]  
    m = (l + r) // 2  
    left = max_sum(arr, l, m)  
    right = max_sum(arr, m+1, r)  
    cross = sum_ = 0  
    c = -10**18  
    for i in range(m, l-1, -1):  
        sum_ += arr[i]  
        c = max(c, sum_)  
    sum_ = 0  
    d = -10**18  
    for i in range(m+1, r+1):  
        sum_ += arr[i]  
        d = max(d, sum_)  
    return max(left, right, c+d)  
  
T = int(input())  
for _ in range(T):  
    N = int(input())  
    arr = list(map(int, input().split()))  
    print(max_sum(arr, 0, N-1))
```

The screenshot shows the OnlineGDB interface. The top bar includes buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The main area displays the following Python code:

```
1 def max_sum(arr, l, r):
2     if l == r: return arr[l]
3     m = (l + r) // 2
4     left = max_sum(arr, l, m)
5     right = max_sum(arr, m+1, r)
6     cross = sum_ = 0
7     c = -10**18
8     for i in range(m, l-1, -1):
9         sum_ += arr[i]
10        c = max(c, sum_)
11    sum_ = 0
12    d = -10**18
13    for i in range(m+1, r+1):
14        sum_ += arr[i]
15        d = max(d, sum_)
16    return max(left, right, c+d)
17
18 T = int(input())
19 for _ in range(T):
20     N = int(input())
21     arr = list(map(int, input().split()))
22     print(max_sum(arr, 0, N-1))
```

## Output:

The terminal window shows the input data and the resulting output. The input data is:

```
1
9
-2 1 -3 4 -1 2 1 -5 4
6
```

The output shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

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

## Java Code:

```
import java.util.*;
public class Main {
    static long maxSum(long[] a, int l, int r) {
        if(l==r) return a[l];
        int m=(l+r)/2;
        long left=maxSum(a,l,m), right=maxSum(a,m+1,r);
        long sum=0,c=-Long.MAX_VALUE;
        for(int i=m;i>=l;i--) {sum+=a[i]; c=Math.max(c,sum);}
        sum=0; long d=-Long.MAX_VALUE;
        for(int i=m+1;i<=r;i++) {sum+=a[i]; d=Math.max(d,sum);}
        return Math.max(Math.max(left,right),c+d);
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int T=sc.nextInt();
```

```

while(T-->0){

    int N=sc.nextInt();

    long[] a=new long[N];

    for(int i=0;i<N;i++) a[i]=sc.nextLong();

    System.out.println(maxSum(a,0,N-1));

}

sc.close();

}

}

```

The screenshot shows the OnlineGDB interface. The left sidebar displays user information and navigation links. The main area shows the Java code for calculating the maximum sum of a subarray. The code uses a divide-and-conquer approach with a helper method `maxSum` and a `main` method that reads input from `System.in`.

```

import java.util.*;
public class Main {
    static long maxSum(long[] a, int l, int r) {
        if(l==r) return a[l];
        int m=(l+r)/2;
        long left=maxSum(a,l,m), right=maxSum(a,m+1,r);
        long sum=0, c=-Long.MAX_VALUE;
        for(int i=m;i>l;i--) {sum+=a[i]; c=Math.max(c,sum);}
        sum=0; long d=-Long.MAX_VALUE;
        for(int i=m+1;i<r;i++) {sum+=a[i]; d=Math.max(d,sum);}
        return Math.max(Math.max(left,right),c+d);
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int T=sc.nextInt();
        while(T-->0){
            int N=sc.nextInt();
            long[] a=new long[N];
            for(int i=0;i<N;i++) a[i]=sc.nextLong();
            System.out.println(maxSum(a,0,N-1));
        }
        sc.close();
    }
}

```

### Output:

The terminal window shows the input data and the resulting output. The input consists of two lines: the first line contains the numbers 1, 9, -2, 1, -3, 4, -1, 2, 1, -5, 4, and the second line contains the number 6. The output shows the program's response to this input.

```

1
9
-2 1 -3 4 -1 2 1 -5 4
6
...Program finished with exit code 0
Press ENTER to exit console.

```