

CSE 3004 DAA LAB TASK - 2

NAME : V.SAIKRISHNA

REG. NO :19BCE7638

MERGE SORT

Merge Sort is a divide and conquer algorithm. It works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

CODE:

```
public class Main
{
    void merge(int arr[], int l, int m, int r)
    {
        int n1 = m - l + 1;
        int n2 = r - m;

        int L[] = new int [n1];
        int R[] = new int [n2];
```

```
for (int i=0; i<n1; ++i)
    L[i] = arr[l + i];
for (int j=0; j<n2; ++j)
    R[j] = arr[m + 1+ j];
```

```
int i = 0, j = 0;
int k = l;
while (i < n1 && j < n2)
{
    if (L[i] <= R[j])
    {
        arr[k] = L[i];
        i++;
    }
    else
    {
        arr[k] = R[j];
        j++;
    }
    k++;
}
while (i < n1)
{
    arr[k] = L[i];
    i++;
    k++;
}
while (j < n2)
{
    arr[k] = R[j];
```

```

        j++;
        k++;
    }
}

void sort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = (l+r)/2;
        sort(arr, l, m);
        sort(arr, m+1, r);
        merge(arr, l, m, r);
    }
}

static void printArray(int arr[])
{
    int n = arr.length;
    for (int i=0; i<n; ++i)
        System.out.print(arr[i] + " ");
    System.out.println();
}

public static void main(String args[])
{
    int arr[] = {32, 5, 55, 8, 62, 14};
    System.out.println("Given Array");
    printArray(arr);

    Main ob = new Main();
    ob.sort(arr, 0, arr.length-1);
    System.out.println("\nSorted array");
    printArray(arr);
}

```

```
}  
}
```

SCREENSHOTS:

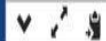
```
8 public class Main  
9 {  
10     void merge(int arr[], int l, int m, int r)  
11     {  
12         int n1 = m - l + 1;  
13         int n2 = r - m;  
14  
15         int L[] = new int [n1];  
16         int R[] = new int [n2];  
17  
18         for (int i=0; i<n1; ++i)  
19             L[i] = arr[l + i];  
20         for (int j=0; j<n2; ++j)  
21             R[j] = arr[m + 1+ j];  
22  
23         int i = 0, j = 0;  
24         int k = l;  
25         while (i < n1 && j < n2)  
26         {  
27             if (L[i] <= R[j])  
28             {  
29                 arr[k] = L[i];  
30                 i++;  
31             }  
32             else  
33             {  
34                 arr[k] = R[j];  
35                 j++;
```

```
36             }  
37             k++;  
38         }  
39         while (i < n1)  
40         {  
41             arr[k] = L[i];  
42             i++;  
43             k++;  
44         }  
45         while (j < n2)  
46         {  
47             arr[k] = R[j];  
48             j++;  
49             k++;  
50         }  
51     }  
52     void sort(int arr[], int l, int r)  
53     {  
54         if (l < r)  
55         {  
56             int m = (l+r)/2;  
57             sort(arr, l, m);  
58             sort(arr, m+1, r);  
59             merge(arr, l, m, r);  
60         }  
61     }  
62     static void printArray(int arr[])
```

```

54     if (l < r)
55     {
56         int m = (l+r)/2;
57         sort(arr, l, m);
58         sort(arr, m+1, r);
59         merge(arr, l, m, r);
60     }
61 }
62 static void printArray(int arr[])
63 {
64     int n = arr.length;
65     for (int i=0; i<n; ++i)
66         System.out.print(arr[i] + " ");
67     System.out.println();
68 }
69 public static void main(String args[])
70 {
71     int arr[] = {32, 5, 55, 8, 62, 14};
72     System.out.println("Given Array");
73     printArray(arr);
74
75     Main ob = new Main();
76     ob.sort(arr, 0, arr.length-1);
77     System.out.println("\nSorted array");
78     printArray(arr);
79 }
80 }
81

```



input

Given Array
32 5 55 8 62 14

Sorted array
5 8 14 32 55 62

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

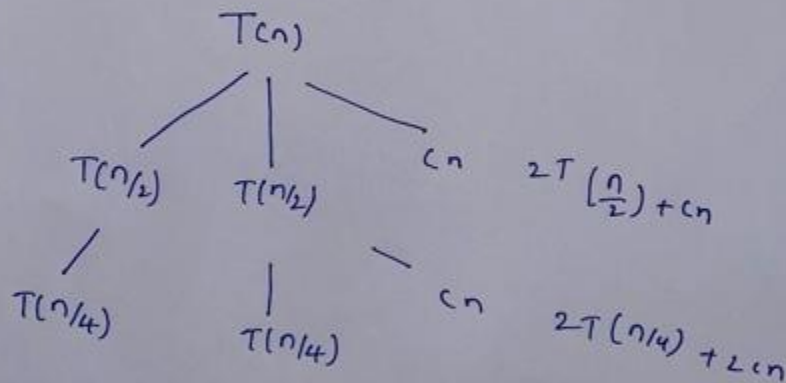
ANALYSIS:

Merge sort:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn \quad n > 0$$

$$= 0 \quad n = 0$$

Using Tree method



$$2 + \left(\frac{n}{2}\right) + n cn \Rightarrow 2^k = n \Rightarrow k = \log n$$

$$\therefore O(n \log n)$$

