Merge Sort

Name: S.Pranav Surya

Reg.No.: 19BCE7598

Code:

```
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[] = {12, 11, 13, 5, 6, 7};

        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);
    }
}
```

Analysis:

Here, to find min key() functions utmost $n(n-1) = n^2 - n$ times. So the graph of the time complexity is $O(n^2)$

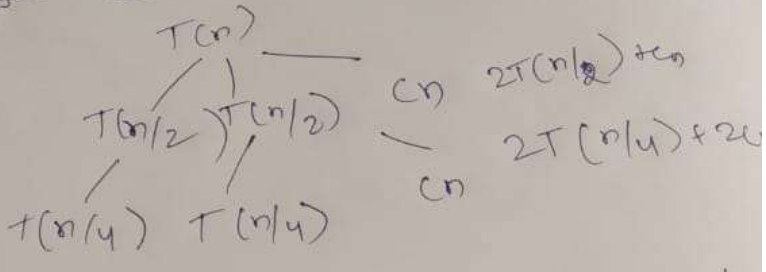
Merge sort:-

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

$$n = 0.$$

$$= 0$$

Using Tree method



$2^k T(n/2^k) + n \cdot cn \Rightarrow 2^k = n \Rightarrow$ k = log n

$\therefore O(n \log n)$