Experiment-4

AIM:- Wap to write merge sort and find its complexity

Software used :- Code blocks

Merge Sort algo :-

```
merge_SORT (A, p, r)
    if p < r
        q = Floor[(p+r)/2]
        merge-SORT [A, p, q]
        Merge-sort [A, q+1, r]
        Merge (A, p, q, r)

Merge (A, p, q, r)
n1 = p-q+1
n2 = r-q
Create arrays L[1...n1+1] and R[1...n2+1]
  for i=1 to n1
        L[i] = A[p+i-1]
  for j=1 to n2
        R[j] = A[q+j]
  L1 [n1+1] = ∞
  L2 [n2+1] = ∞
  i=1
  j=1
```

```
for K = p to r
    if L[i] <= R[j]
        A[K] = L[i]
        i = i+1
    else
        A[K] = R[j]
        j = j+1
```

$$T_n = \begin{cases} O(1) & \text{if } n=1 \\ \underbrace{2T(n/2)} + ~~\text{ag}~~ n & \text{if } n>1 \end{cases}$$

merge funn which iterate over the loop.

Since in each call to merge sort two calls are made again to mergesort that divides the array into two parts

Using master's Theorem

$a = 2, \; b = 2$

$T(n) = a\,T(n/b) + f(n)$

$n^{\log_b a} = n^{\log_2 2} = n$;

$f(n) = ~~k~~n$, $n^{\log_b a} = f(n)$

Using 2nd case of master's theorem

$T(n) = \Theta(n \log n)$

1/10/10