Merge Sort

Merging two sorted arrays which have a total of *n* elements requires at most *n-1* comparisons.

e.g. arr1= [1], arr2=[2, 3, 4, 5, 6, 7, 8, 9, 10]

Final array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] => number of comparisons = 1

e.g. arr1=[10], arr2=[1, 2, 3, 4, 5, 6, 7, 8, 9]

Final array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] => number of comparisons = 9

e.g. arr1=[1, 2, 3, 4, 5], arr2=[6, 7, 8, 9, 10]

Final array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] => number of comparisons = 5

Since after every comparison one element gets positioned at its correct location, we will never require more than n-1 comparisons to merge two sorted arrays having a total of n elements.

In merge sort we split the array in half, and then call mergesort on each half, and finally merge the two halves. Thus the total amount of comparisons needed, are the number of comparisons to mergesort each half, plus the number of comparisons necessary to merge the two halves.

Let *T(n)* be the function denoting the number of comparisons necessary to mergesort an array with *n* elements.

$T(n) <= T(n/2) + T(n/2) + n-1$

$T(n) <= 2 \cdot T(n/2) + n-1$

$T(n) < 2 \cdot T(n/2) + n$

Question : How many comparisons will be performed if we merge sort an array with only 1 element?

Answer : 0

$T(1) = 0$

Question : How many comparisons will be performed if we merge sort an array with 2 elements?

*T(2) < 2*T(2/2) + 2*

*T(2) < 2*T(1) + 2*

T(2) < 2

Question : How many comparisons will be performed if we merge sort an array with 4 elements?

*T(4) < 2*T(4/2) + 4*

*T(4) < 2*T(2) + 4*

*T(4) < 2*2 + 4*

T(4) < 8

Question : How many comparisons will be performed if we merge sort an array with 8 elements?

*T(8) < 2*T(8/2) + 8*

*T(8) < 2*T(4) + 8*

*T(8) < 2*8 + 8*

T(8) < 24

Question : How many comparisons will be performed if we merge sort an array with 16 elements?

*T(16) < 2*T(16/2) + 16*

*T(16) < 2*T(8) + 16*

*T(16) < 2*24 + 16*

T(16) < 64

n	T(n)
$1 = 2^0$	0
$2 = 2^1$	2*T(1)+2 = 2 = 2*1
$4 = 2^2$	2*T(2)+4 = 8 = 4*2
$8 = 2^3$	2*T(4)+8 = 24 = 8*3
$16 = 2^4$	2*T(8)+16 = 64 = 16*4
$32 = 2^5$	2*T(16)+32 = 160 = 32*5
...	...
$n = 2^k$	2*T(n/2)+n = n*k = $n \cdot log_2 n$

T(n) < n log₂ n

Note : (*n log₂ n) is upper bound, the exact number of comparisons performed to merge sort an array of n element, will be less than* (*n log₂ n) but it can never be greater than* (*n log₂ n)*

Time Complexity = O(nlogn)

Space Complexity

We divide the array in two halves and store it in leftSubArray and rightSubArray. So in merge sort we will be creating lot of temporary arrays, **but they won't coexist at the same time.**

So the maximum extra space required to execute mergesort for an array of n elements will be equal to space required to store n elements (temporarily in leftSubArray and rightSubArray)

Space Complexity = O(n)