Task-02 (i)

Worst case time complexity of merge sort: $O(n \log n)$

Worst case: When the left and right subarray in all merge operation have alternate elements.

Proof by using substitution method:

Recurrence equation

$$T(n) = T(n/2) + T(n/2) + n$$

$$\Rightarrow T(n) = 2T(n/2) + n \quad \text{---(i)}$$

Now,

$$T(n/2) = 2T(n/4) + n/2 \quad \text{---(ii)}$$

[replacing n with $n/2$]

After substituting (ii) in (i),

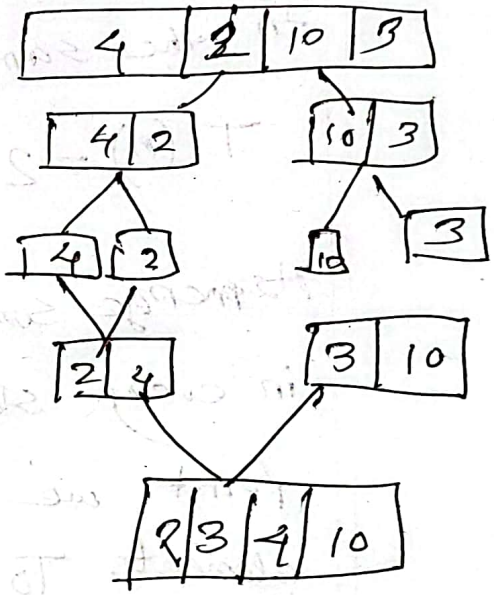
$$T(n) = 2\{2T(n/4) + n/2\} + n$$

$$\boxed{T(n) = 2^2 T\left(\frac{n}{2^2}\right) + 2n} \quad \text{---(iii)}$$

Substituting $n/4$ in (i),

$$T(n/4) = 2T(n/8) + n/4 \quad \text{---(iv)}$$

Simulation



(iv)

Substituting (iii) in (ii)

$$T(n) = 2^2\left\{2T\left(\frac{n}{8}\right) + \frac{n}{4}\right\} + 2n$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

In the same way,

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + in$$

As merge sort divides the array into two parts in every step. If we continue to do so, at one point we ~~reach~~ get an array with only 1 element. To sort an array with only one element, time required is 1. $\therefore T(1) = 1$

let $n/2^i = 1$

$$n = 2^i$$

or $\log_2(n) = i \log_2(2)$

$\therefore \boxed{\log_2(n) = i}$

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + in$$

$$= n\,T(1) + \log_2 n \cdot n$$

$$\therefore T(n) = n + \log_2 n \cdot n$$

$\therefore$ Time complexity $= O(n \log_2 n)$

Proved