Task 2 (i)

```
Merge Sort (A, l, h) — T(n)
    if l<h
        mid =(l+h)//2 —— 1
        Merge Sort (~~A~~l, mid) — T(n/2)
        Merge Sort (~~A~~mid +1, h) — T(n/2)
        Merge (A, l, mid, h) —— n
```

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

$$T(n) = \begin{cases} 1 & ; n=1 \\ 2T(n/2)+n & ; n>1 \end{cases}$$

$$T(n) = \left[2T(n/2)+n \right.$$

Let, $n = 2^m$

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

$$= 2T(2^{m-1}) + 2^m$$

$$= 2^2 T(2^{m-2}) + 2^{1+m-1} + 2^m$$

$$= 2^2\left[2T(2^{m-3}) + 2^{m-2}\right] + 2^m + 2^m$$

$$= 2^3 T(2^{m-3}) + 2^{2+m-2} + 2^m + 2^m$$

$$= 2^3 T(2^{m-3}) + 2^m + 2^m + 2^m$$

$$T(2^m) = 2^3 T(2^{m-3}) + 3\,2^m$$

$$\vdots$$

$$= 2^k T(2^{m-k}) + k\,2^m$$

For $T(2^{m-k}) = T(1)$

$m - k = 0$

$\therefore m = k$

$$= 2^m T(2^{m-m}) + m\,2^m$$

$$= 2^m + m\,2^m \quad [T(1) = 1]$$

$$= n + mn \quad [\because 2^m = n]$$

$$= n + n\log n$$

Here, $n = 2^m$

$\log_2 n = \log_2 2^m$

$\therefore m = \log n$

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

$= O(n\log n)$ is the worst case time complexity for merge sort.