Lecture 22 Exercise 3

Consider the modification of MergeSort, that instead dividing the input array into, halves, dividing into thirds, recursively sort each third, and finally combine the results, with a Merge function. What is the running time of this algorithm ignoring constant factors and lower-order terms?

Claim : This version of Mergesort has the same complexity as the classic Mergesort. $O(n \log n)$.

Proof :

① for classic Mergesort we have

$$T_c(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + O(n) \quad *1$$
$$= T(n/2) + T(n/2) + O(n)$$
$$= 2T(n/2) + O(n)$$
$$= \cdots \text{ Master Theorem or recurrence tree Method}$$
$$= O(n \log_2 n)$$

② for the modified Mergesort we have

$$T_m(n) = T(n/3) + T(n/3) + T(n/3) + O(n) \quad *2$$
$$= 3T(n/3) + O(n)$$
$$= \cdots \text{ Master Theorem or recurrence tree Method}$$
$$= O(n \log_3 n)$$

③ from ① and ② we deduce

$$T_c(n) = O(n \log_2 n)$$

$$T_s(n) = O(n \log_3 n)$$

④ $\log_2 n = c \cdot \log_3 n$ where c constant.

Proof : $\log_2 x = \dfrac{\log_3 x}{\log_3 2} \Rightarrow$

$\log_2 x = \dfrac{1}{\log_3 2} \log_3 x \Rightarrow$

$\log_2 x = c \log_3 x$ for $c = \dfrac{1}{\log_3 2}$

⑤ from ③ and ④ we can see that

$$T_c(n) = T_m(n) = O(n \log n) \quad \blacksquare$$

$*1$: assuming $n = 2^k$ for some $k \in \mathbb{N}$

$*2$: assumming $n = 3^k$ for some $k \in \mathbb{N}$