Oct 21

MergeSort(a, n)

If $n = 1$ return $a_1 \leftarrow O(1)$

$\lfloor 0.3 \rfloor = 0$

$\lceil 0.3 \rceil = 1$

$O(n)$ $\begin{cases} a_L = a_1, \ldots, a_{\lfloor \frac{n}{2} \rfloor} \\ a_R = a_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, a_n \end{cases}$ $\leq T(\lfloor \frac{n}{2} \rfloor)$ $\leq T(n - \lfloor \frac{n}{2} \rfloor)$

return MERGE(MergeSort $(a_L, \lfloor \frac{n}{2} \rfloor)$; MergeSort $(a_R, n - \lfloor \frac{n}{2} \rfloor)$)

$O(n)$

$T(n)$ $\stackrel{\text{def}}{=}$ max. runtime of MergeSort over ALL inputs of size n

---

$$T(n) \leq O(1) + O(n) + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor) + O(n)$$

If $n = 1$, $T(1) = O(1)$

$n > 1$, $T(n) = O(n) + T(\lfloor \frac{n}{2} \rfloor) + T(n - \lfloor \frac{n}{2} \rfloor)$

Rewrite $T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + T(\lfloor \frac{n}{2} \rfloor) + T(n - \lfloor \frac{n}{2} \rfloor) & \text{o.w.} \end{cases}$

(By defn of Big O)

$T(n) \leq \begin{cases} c_1 & \text{if } n = 1 \\ c_2 n + T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) \end{cases}$

$[c = \max(c_1, c_2)]$

$T(n) \leq \begin{cases} c & \text{if } n = 1 \\ cn + T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) \end{cases}$

Rule of thumb: for asymptotics of $T(n)$

$T(\lfloor x \rfloor) = T(x)$; $T(\lceil x \rceil) = T(x)$

$T(n) \leq \begin{cases} c & \text{if } n = 1 \\ cn + 2T(\frac{n}{2}) & \text{o.w.} \end{cases}$

Lemma: $T(n) \leq cn\log_2 n + cn$

$\Rightarrow$ MergeSort runs in time $O(n \log n)$

Some remarks:

1. $O(n \log n)$ best known upper bound for general algo.
2. Can do faster if domain of the a_i's is of size $O(n)$ (T/F on piazza $a_i \in \{0,1\}$)
3. Can have faster runtime for "almost" sorted input.
4. Any comparison based ~~algo~~ algo for sorting takes $\Omega(n \log n)$ comparisons.