Mar 29

# Merge Sort (a, n)

$\lfloor 0.3 \rfloor = 0 \quad \lceil 0.3 \rceil = 1$

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

$a_L = a_1, \ldots, a_{\lfloor \frac{n}{2} \rfloor}$

$a_R = a_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, a_n$ } $O(n)$

return MERGE( Merge Sort $(a_L, \lfloor \frac{n}{2} \rfloor)$, $\leq T(\lfloor \frac{n}{2} \rfloor)$

Merge Sort $(a_R, n - \lfloor \frac{n}{2} \rfloor)$ ) $\leq T(n - \lfloor \frac{n}{2} \rfloor)$

$O(n)$

---

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

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

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

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

$$T(n) = \begin{cases} O(1) & \text{if } n=1 \\ O(n) + T\left(\lfloor \tfrac{n}{2} \rfloor\right) + T\left(n - \lfloor \tfrac{n}{2} \rfloor\right) & \text{if } n>1 \end{cases}$$

By defn of Big-Oh

$$T(n) \leq \begin{cases} c_1 & \text{if } n=1 \\ c_2 \cdot n + T\left(\lfloor \tfrac{n}{2} \rfloor\right) + T\left(n - \lfloor \tfrac{n}{2} \rfloor\right) & \text{o.w.} \end{cases}$$

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

$$T(n) \leq \begin{cases} c & \text{if } n=1 \\ c.n + T\left(\lfloor \frac{n}{2} \rfloor\right) + T\left(n - \lfloor \frac{n}{2} \rfloor\right) \end{cases}$$

$$\Downarrow$$

$$T\left(\lceil \frac{n}{2} \rceil\right)$$

Rule of thumb

$$T(\lfloor x \rfloor) = T(x)$$

$$T(\lceil x \rceil) = T(x)$$

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

**Lemma:** $T(n) \leq c.n.\log_2 n + c.n$

$\Rightarrow$ Merge sort runs in $O(n.\log n)$ time

## 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 $O(1)$
(T/F or pizza $a_i \in \{0,1\}$)

(3) Can have faster runtime for "almost" sorted input

(4) Any comparison based alg. for sorting takes $\Omega(n \cdot \log n)$ comparisons.

# Strategies for solving recurrences

1. "Unroll" the recurrence and use the pattern
2. Guess the answer and verify using induction on $n$

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

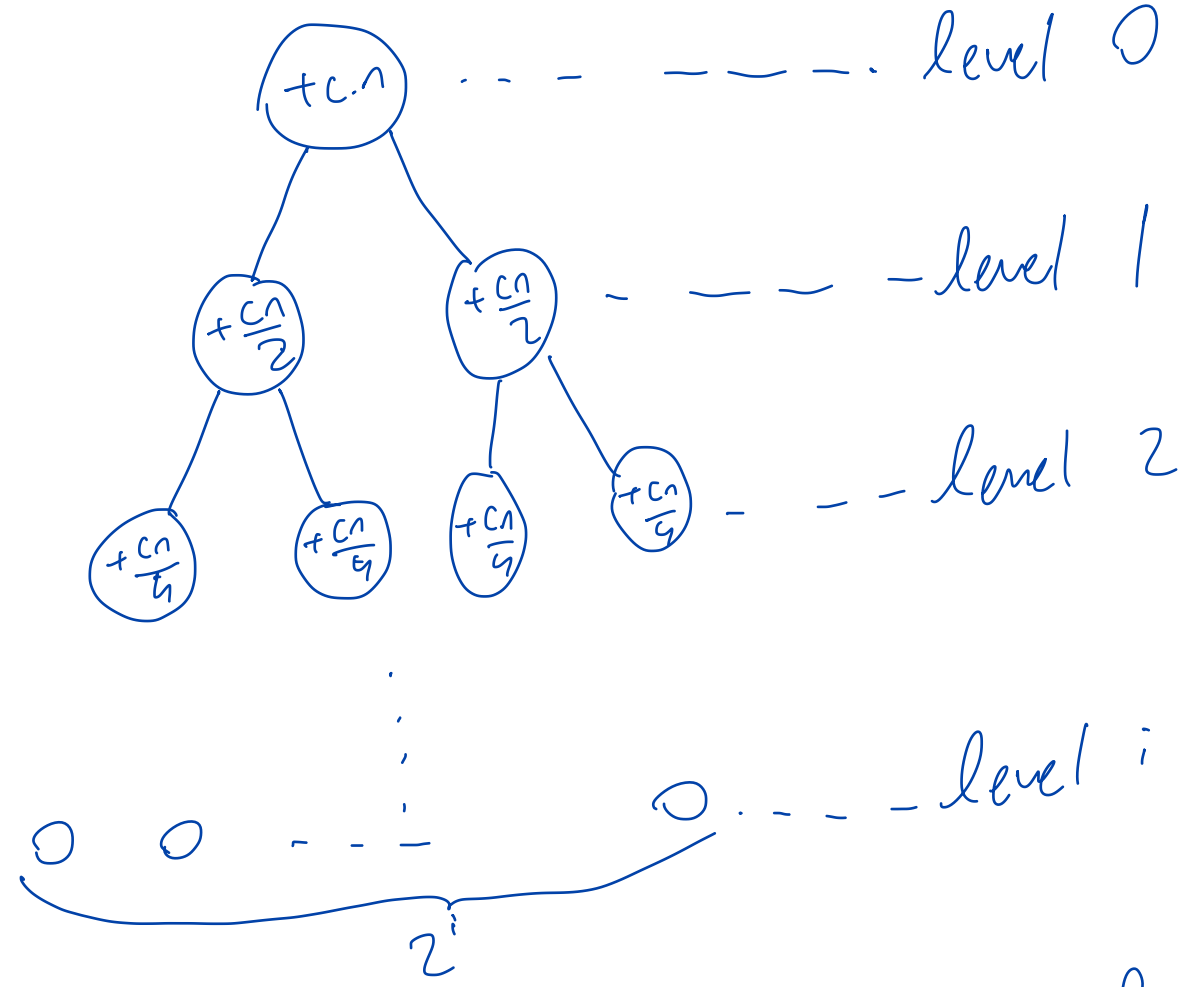
Pf. of Lemma: $T(n) \leq c \cdot n + 2 \cdot T\left(\frac{n}{2}\right)$

$$\leq cn + 2\left(\frac{cn}{2} + 2 \cdot T\left(\frac{n}{4}\right)\right)$$

$$= cn + cn + 4 \cdot T\left(\frac{n}{4}\right)$$

$$\leq cn + cn + 4\left(\frac{cn}{4} + 2 \cdot T\left(\frac{n}{8}\right)\right)$$

$$= cn + cn + cn + 8 \cdot T\left(\frac{n}{8}\right)$$




→ Contribution from level $i$ : $= 2^i \cdot \frac{cn}{2^i} = c.n$

$\Rightarrow T(n) \leq c.n.(\#\text{ levels})$

Note: $n = 2^l \Rightarrow l = \log_2 n$

$\Rightarrow T(n) \leq c.n.(\log_2 n + 1)$

$= c.n.\log_2 n + c.n$

Strategy 2: Guess; $T(n) \leq c \cdot n \cdot \log_2 n + cn$

Base case: $\Rightarrow T(1) = c \cdot 1 \cdot \log_2 1 + c$

$= c$ ✓

Ind. hypothesis: $T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2} \cdot \log_2\left(\frac{n}{2}\right) + c \cdot \frac{n}{2}$

$= \frac{cn}{2}\left(\log_2\left(\frac{n}{2}\right) + 1\right)$

$= \frac{cn}{2}\left(\log_2 n - \log_2 2 + 1\right)$

$= \frac{cn}{2} \cdot \log_2 n \quad (\star)$

Ind. step: By recursion

$T(n) \leq c \cdot n + 2 \cdot T\left(\frac{n}{2}\right)$

by $(\star)$ $\leq c \cdot n + 2\left(\frac{cn}{2} \cdot \log_2 n\right)$

$= cn + cn \cdot \log_2 n$ ∎