Oct 25

MergeSort (a, n)

$\lfloor 3.3 \rfloor = 3$, $\lceil 3.3 \rceil = 4$; $\lfloor 0.3 \rfloor = 0$, $\lceil 0.3 \rceil = 1$

$O(1)$ { If $n = 1$ then return a_1

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

If x is integer: $\lfloor x \rfloor = x$, $\lceil x \rceil = x$

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

$\leq T(\lfloor \frac{n}{2} \rfloor)$

$O(n)$

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

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

$T(n - \lfloor \frac{n}{2} \rfloor)$

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

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

Oct 26

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

$n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$

By def of Big-Oh, $\exists$ constants c_1, c_2

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

Pick $c = \max(c_1, c_2)$

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

Rule of thumb: for asymptotics of $T(n)$ enough: $T(\lfloor x \rfloor) \to T(x)$, $T(\lceil x \rceil) \to T(x)$

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

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

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

Some remarks:

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

Strategies for solving recurrences

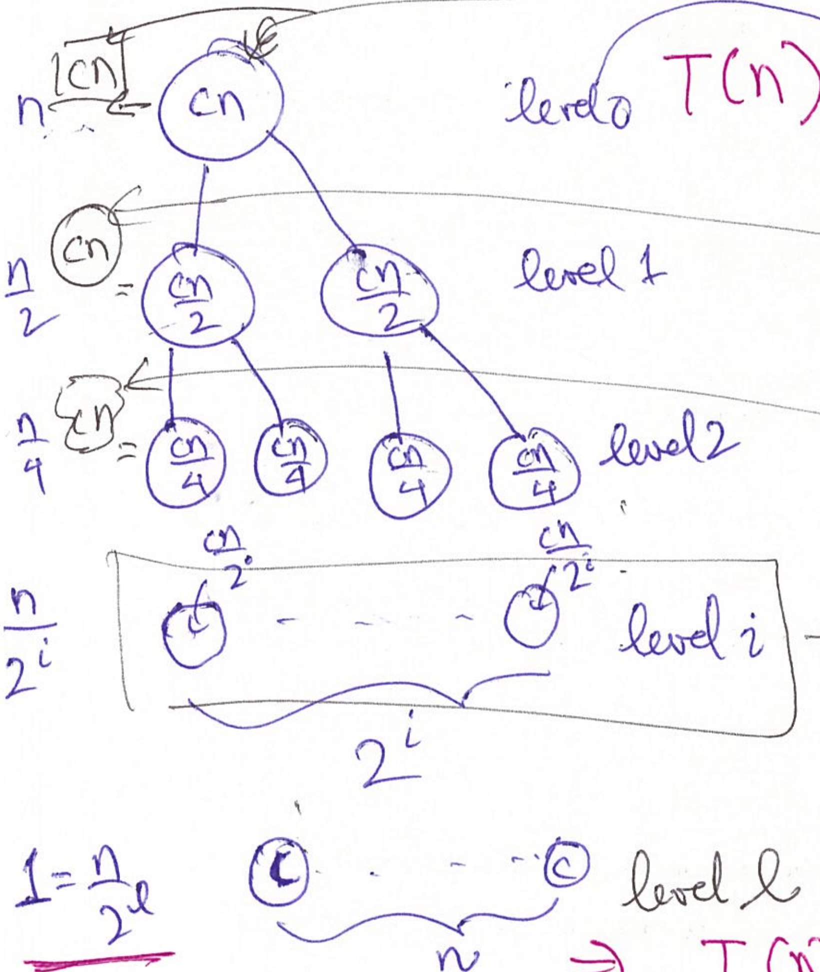
1. "Unroll" the recurrence & identify a pattern; then use the pattern
2. Guess the answer & verify by induction on n.

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

Goal: $T(n) \le cn \log_2 n + cn$

Strategy 1: "unroll" + use "pattern"

Assume: n is a power of 2



$$T(n) \le cn + 2T\left(\frac{n}{2}\right)$$

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

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

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

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

$\vdots$

Contribution from level i $\left(\frac{n}{2^i} > 1\right)$

$$2^i \cdot \frac{cn}{2^i} = cn$$

$$\frac{n}{2^l} = 1 \iff n = 2^l \iff l = \log_2 n$$

$$\Rightarrow T(n) \le cn\,(\#\text{levels})$$

$$= cn\,(l+1)$$

$$= cn\,(\log_2 n + 1)$$

$$= cn \log_2 n + cn \quad \blacksquare$$

Strategy 2 Guess $T(n) \le cn \log_2 n + cn$ —— (*)

Verify by induction on n.

Base case: $n = 1$ need to show $c \le c \cdot 1 \cdot \log_2 1 + c \cdot 1 = c$

(*) hold for $n < \frac{n}{2}$ $c \le c$ ✓

I.H. $T(\frac{n}{2}) \le c\frac{n}{2} \log_2 \frac{n}{2} + \frac{cn}{2}$

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

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

$= \frac{cn}{2} \log_2 n$

I.S: By recurrence:

$T(n) \le cn + 2T(\frac{n}{2})$

I.H $\rightarrow \le cn + 2\left(\frac{cn}{2} \log_2 n\right)$

$= cn + cn \log_2 n$

$= cn(\log_2 n + 1)$ ∎

Collaborative filtering