Oct25

floor $\lfloor 0.3 \rfloor = 0$

ceil $\lceil 0.3 \rceil = 1$

MergeSort (a, n)

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

O(n) { $a_L \leftarrow a_1, \ldots, a_{\lfloor \frac{n}{2} \rfloor}$

$a_R \leftarrow a_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, a_n$

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

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

O(n)

$T(n)$ def max runtime of MS over all i/ps of len n

Thm: $T(n)$ is $O(n \log n)$

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

o/

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

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

$$= O(n) + T(\lfloor \tfrac{n}{2} \rfloor) + T(\lceil \tfrac{n}{2} \rceil)$$

$$\approx O(n) + T(\tfrac{n}{2}) + T(\tfrac{n}{2})$$

$$= O(n) + 2T(\tfrac{n}{2})$$

$$T(n) \leq \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T(\frac{n}{2}) & \text{if } n > 1 \end{cases}$$

Oct28

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

$O(n) \leq c_2 n$

By defn of $O(1)$ $\exists$ constants c_1 & c_2

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

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

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

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

$T(\frac{n}{2})$ $T(\frac{n}{2})$

recurrence

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

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

$(\leq 2cn \log_2 n \leq O(n \log n))$

COR: MergeSort runs in $O(n \log n)$.

Some remarks:

1. $O(n \log n)$ is the best known upper bound for general sorting algo.
2. Can do sorting in $O(n)$ time if each a_i is $O(n)$ {eg. HW4 Q1}
3. Algo that runs faster on "almost sorted" input.
4. Any comparison based sorting algo needs to use $\Omega(n \log n)$ comparisons.

Strategies to solve recurrences

Strategy 1: "Unroll" the recurrence for few steps.
& hopefully notice a "pattern" → use this pattern to finish this proof.

Strategy 2: Guess the ~~on~~ the answer & use induction on n.

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

$n \geq 2$ ↖ $n/2$

Goal: $T(n) \leq cn \log_2 n + cn$ → $n \geq 2$

$T(n) \leq$ $cn + 2T(\frac{n}{2})$ ↖ $n/2$ ↖ $n/4$

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

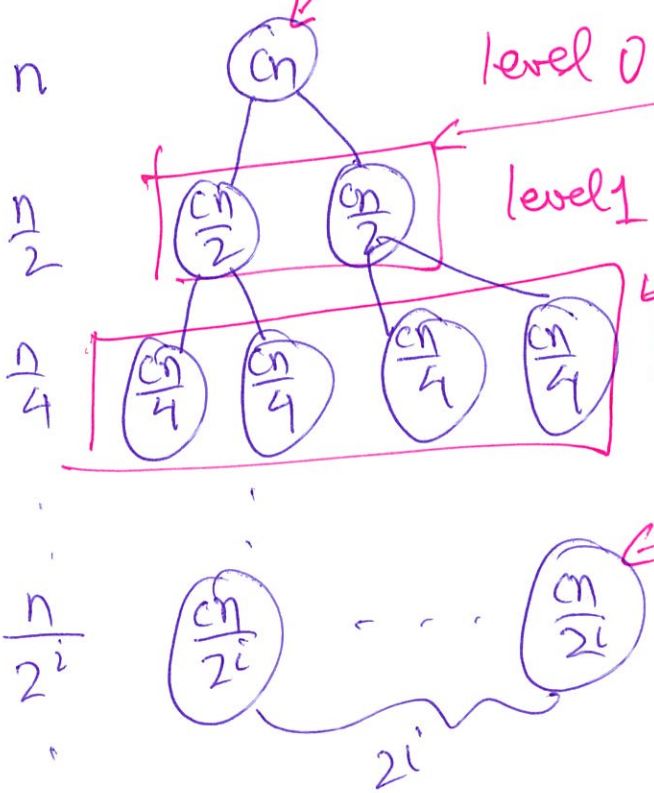
$= cn + cn + 4T(\frac{n}{4})$

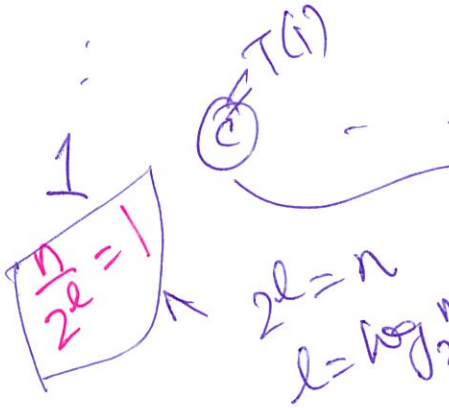
$\leq cn + cn + 4\left(\frac{cn}{4} + 2T(\frac{n}{8})\right)$

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

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

$\leq cn + \dots + cn + 2^i T(\frac{n}{2^i})$





At level i, we'll get 2^i copies $\frac{cn}{2^i}$ ⇒ total from level i $2^i \cdot \frac{cn}{2^i} = cn$

⇒ for all $0 \leq i \leq \ell$

Total overall $= cn(\ell+1)$

$= cn(\log_2 n + 1)$

$= cn \log n + cn$

Strategy 2: Prove by induction on n — (*)

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

Base case: $T(1) \le c = cn\log_2 n$ then for $n=1$

$cn\log_2 n + cn \xrightarrow{n=1} c\cdot 1\log_2 1 + c\cdot 1$ ($\log_2 1 = 0$)

I-H. Assume (*) is true for $\frac{n}{2}$

$$T\left(\frac{n}{2}\right) \le \frac{cn}{2}\log_2\frac{n}{2} + \frac{cn}{2}$$

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

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

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

I-S. By def $n \ge 2$

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

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

$$= cn + cn\log_2 n$$ ∎

Collaborative Filtering

Netflix each user: ranking of shows $a_1, \dots, a_n$

$b_1, \dots, b_n$

How far are the two rankings

(i) Spearman's footrule: $\sum_{i=1}^{n} |a_i - b_i|$

(ii) Kendal Tau dist: how many pairs i & j are ranked diff.