Oct27

MergeSort(a, n) $\leq T(n)$

floor $\lfloor 0.3 \rfloor = 0$

Ceil $\lceil 0.3 \rceil = 1$

$O(1)$ → If $n = 1$ return a_1

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

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

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

$O(n)$

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

$T(n) \stackrel{\text{def}}{=}$ max # steps MergeSort takes on any input of size n.

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

$\lceil \frac{n}{2} \rceil$

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

o/w $T(n) = O(n) + T(\lfloor \frac{n}{2} \rfloor) + T(n - \lfloor \frac{n}{2} \rfloor)$

$\leq O(n) + T(\frac{n}{2}) + T(n - \frac{n}{2})$

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

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

Oct30

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

By definition of Big-Oh, $\exists$ ~~constants~~ constants c_1, c_2 s.t

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

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 $T(n)$ replace $T(\lfloor x \rfloor) \leftarrow T(x)$, $T(\lceil x \rceil) \leftarrow T(x)$

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

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

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

Some remarks:

1. $O(n \log n)$ time is the ~~best~~ known upper bound for general sorting algo.
2. We can do sorting in $O(n)$ time if e.g. the domain of a_i is is of size $O(n)$ {e.g. T/F #1, $a_i \in \{0,1\}$}
3. Can faster runtime if the input is "mostly" sorted
4. Any comparison based algo needs to make $\Omega(n \log n)$ comparisons.

Strategies for solving recurrences

(1) "Unroll" the recurrence for few steps & identify a "pattern" → use this pattern to prove an upper bound.

(2) Guess the answer & verify by induction

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

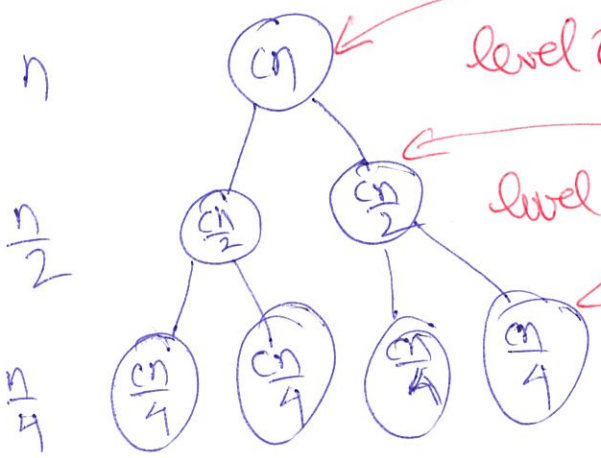
(n → $n/2$, $n/2$ → $n/4$)

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

Assume: n is a power of 2

Strategy 1: "Unroll" the recurrence.

$n > 1$



level 0 $T(n) \le cn + 2\left[T\left(\frac{n}{2}\right)\right]$ $n>2$

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

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

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

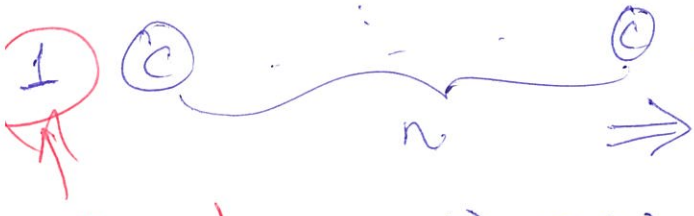
$= cn + cn + cn + 8T\left(\frac{n}{8}\right)$

level i

At level i you will have 2^i terms of exactly $\frac{cn}{2^i}$

total at level i $= \frac{cn}{2^i}\cdot 2^i$

$= cn$



level l

$\Rightarrow$ for all $0 \le i \le l$, we get cn

$\Rightarrow T(n) \le cn(l+1)$

$= cn(\log_2 n + 1)$

$= cn\log_2 n + cn$ ∎

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

$\Leftrightarrow 2^n = 2^l$

$\Leftrightarrow l = \log_2 n$

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

Verify this by induction on n

Base case: $n=1$ By def $T(1) \leq c$ ← (1)

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

$= 0 + c$

$= 0$

$\Rightarrow T(1) \leq c$ ⇒ for $n=1$ ✓

↑ (1)

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

$\Rightarrow T(\frac{n}{2}) \leq c\cdot\frac{n}{2}\cdot\log_2\frac{n}{2} + \frac{cn}{2}$

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

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

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

I.S. By dfn of T at $n \geq 2$

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

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

$= cn + cn\log_2 n$

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

Collaborative filtering

Each user: ranking of n shows $1, \ldots, n$

∈ ranking $a_1, \ldots, a_n$ $a_i \in [n]$

a_1 → most preferred; a_n ← least prefer

Netflix: given a user / ranking, a close enough user / ranking.

Simpler problem: Given 2 rankings $a_1, \ldots, a_n$
$b_1, \ldots, b_n$

Compute how "close" these 2 rankings are

One notion: $\sum_{i=1}^{n} |a_i - b_i|$ ← named distance

Other notion: count # pairs of shows ranked differently.