Nov 3

MergeSort (a, n)

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

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

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

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

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

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

MERGE: $O(n)$

$T(n)$ = max runtime of MergeSort on any input of size n.

Overall: $\rightarrow T(1) \leq O(1)$

$\rightarrow$ If $n > 1$

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

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

Rewrite this:

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

(Use definition of $O(\cdot)$)

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

$c = \max(c_1, c_2)$

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

Rule of thumb: For asymptotics can replace $T(\lfloor x \rfloor)$ by $T(x)$; $T(\lceil x \rceil)$ by $T(x)$

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

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

$\Rightarrow$ MergeSort is $O(n \log n)$

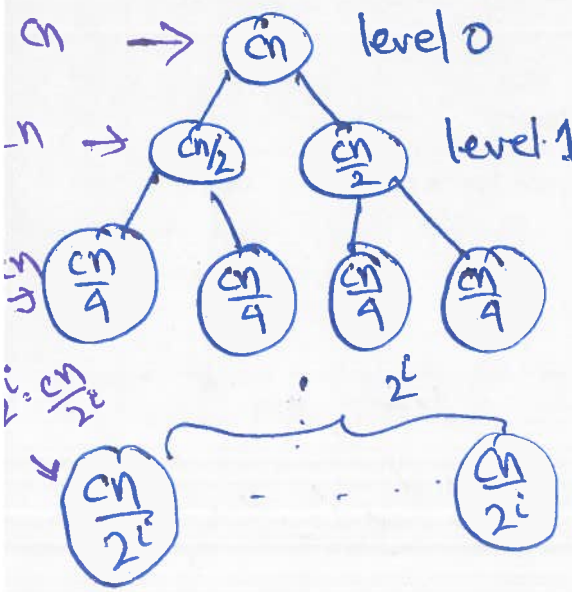
Some Remarks:

1. $O(n \log n)$ is the best known upper bound for the general sorting problem.
2. Can do faster e.g. if a_i values are bounded.
   ↳ $O(n)$ time (Q1 on HW0 showed this)
3. Can have faster runtimes for "almost" sorted input
4. Any comparison based algo has to make $\Omega(n \log n)$ comparisons.

Strategies to solve recurrences

(i) "Unroll" the recurrence and use the pattern
(ii) Guess the final answer and verify by using induction on n
   ↖ (for LEMMA: $cn \log_2 n + cn$)

Strategy 1:



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

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

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

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

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

Contribution of level i
$= 2^i \cdot \frac{cn}{2^i} = cn$

$\Rightarrow T(n) \leq cn$ (# levels)
$= cn(\ell + 1)$

where $\frac{n}{2^\ell} = 1 \Rightarrow 2^\ell = n$

$\Rightarrow \ell = \log_2 n$

$\Rightarrow T(n) \leq cn(\log_2 n + 1)$

$= cn\log_2 n + cn$ ∎

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

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

Base case: $n = 1$ we know $T(1) \leq c$

$c \cdot 1 \cdot \log_2 1 + c \cdot 1 = c$ ✓

Inductive Hypothesis: Assume $T(\frac{n}{2}) \leq \frac{cn}{2}\log_2\frac{n}{2} + \frac{cn}{2}$

Inductive step:

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

by (*) $\rightarrow \leq cn + 2(\frac{cn}{2}\log_2 n)$

$= cn + cn\log_2 n$ ∎

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

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

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