22 Feb 2019

Announcements.

1. No lunch-with-the-instructors today. Let's reschedule.
2. Prelim 1 is graded, make-up prelim not yet. Will release P1 grades on Gradescope within 24 hrs. Make-up prelim grades: goal is by end of Feb. break.
3. Homework 4 (single question) is on CMS, due Friday night 11:59 pm.

Clicker question #1: I think my prelim grade was

(A) < 25 (B) 25–30 (C) 30–35 (D) 35–40 (E) 40–45

DIVIDE & CONQUER (Chapter 5)

E.g. mergesort, binary search, quicksort.

Recap of running time analysis for mergesort.

Mergesort sorts a list of n objects by:

- partitioning into lists of size $n/2$ ($\lfloor \frac{n}{2} \rfloor$ & $\lceil \frac{n}{2} \rceil$) $O(n)$
- recursively sorting each of them ?? $2 \cdot T(\frac{n}{2})$
- combining the results into a sorted list in $O(n)$ time.

Let $T(n)$ be the running time of Mergesort on size-n input.

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n) \quad \leftarrow \text{This is fine.}$$

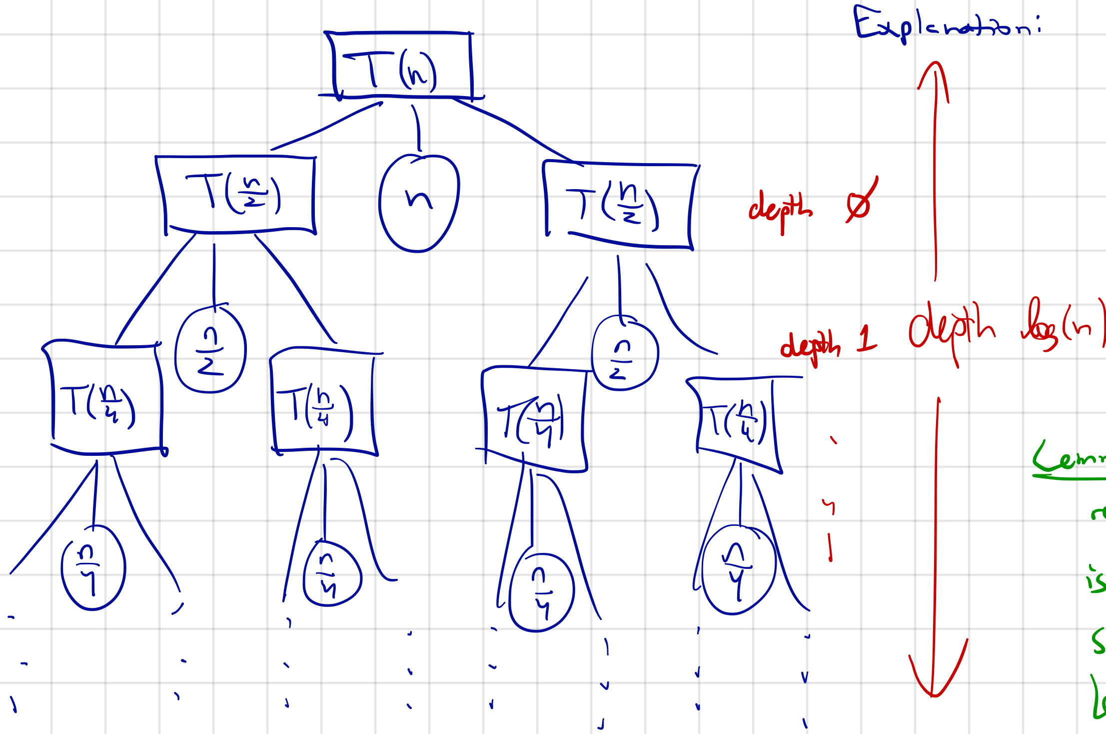
$$\left(T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n).\right) \leftarrow \text{Never mind}$$

$\forall n \geq 1 \qquad T(n) = 2 \cdot T(\frac{n}{2}) + O(n) \qquad T(1) = O(1).$

$$= 2 \cdot \left[2 \cdot T(\tfrac{n}{4}) + O(\tfrac{n}{2}) \right] + O(n)$$

$$= 2 \cdot \left[2 \cdot \left\{ 2 \cdot T(\tfrac{n}{8}) + O(\tfrac{n}{4}) \right\} + O(\tfrac{n}{2}) \right] + O(n)$$

$= \ \dots$ (iterate this substitution process)



Explanation: $\square$ = sum of children

$\bigcirc$ = leaf representing a number. (ignoring const factors.)

<u>Lemma</u>: The number represented by any $\square$ is equal to the sum of the round-node leaves in its subtree.

<u>Proof</u>: Induction on depth of subtree.

At depth d we have 2^d round nodes labeled with $\frac{n}{2^d}$.

Sum of nodes in level d is n.

Sum of all round nodes in tree is $n \cdot (\#\text{ levels}) = n \cdot \log(n)$.

<u>Multiplying numbers (and polynomials)</u>

Current fastest algorithm: $O(n \cdot \log n \cdot 2^{\log_*(n)})$ (Fürer, 2007)
Plausibly fastest possible: $O(n \log n)$
Today's algorithm (Karatsuba): $O(n^{1.58\ldots})$

If $P_0(x) = a_0 x + b_0$

$P_1(x) = a_1 x + b_1$

then $P_0(x) \cdot P_1(x) = a_0 a_1 x^2 + (a_0 b_1 + a_1 b_0) x + b_0 b_1$

$= a_0 a_1 x^2 + \left[(a_0 + b_0)(a_1 + b_1) - a_0 a_1 - b_0 b_1 \right] x + b_0 b_1$

To compute coefficients on 2^{nd} line:

$c_0 = a_0 a_1$

$c_1 = (a_0 + b_0)(a_1 + b_1)$

$c_2 = b_0 b_1$

$$P_0(x)\, P_1(x) = c_0 x^2 + (c_1 - c_0 - c_2) x + c_2$$

(More subtraction, less multiplication)

Now say $P_0(x)$, $P_1(x)$ are polynomials of degree $\leq n-1$ where n is a power of 2.

$P_0(x) = A_0(x) \cdot x^{n/2} + B_0(x)$

$P_1(x) = A_1(x) \cdot x^{n/2} + B_1(x)$

A_0, A_1, B_0, B_1 are polyn's of degree $\leq \frac{n}{2} - 1$.

$$P_0(x) \cdot P_1(x) = A_0(x) A_1(x)\, x^n + \left[(A_0(x) + B_0(x)) \cdot (A_1(x) + B_1(x)) - A_0(x) A_1(x) - B_0(x) B_1(x) \right] x^{n/2} + B_0(x) B_1(x).$$

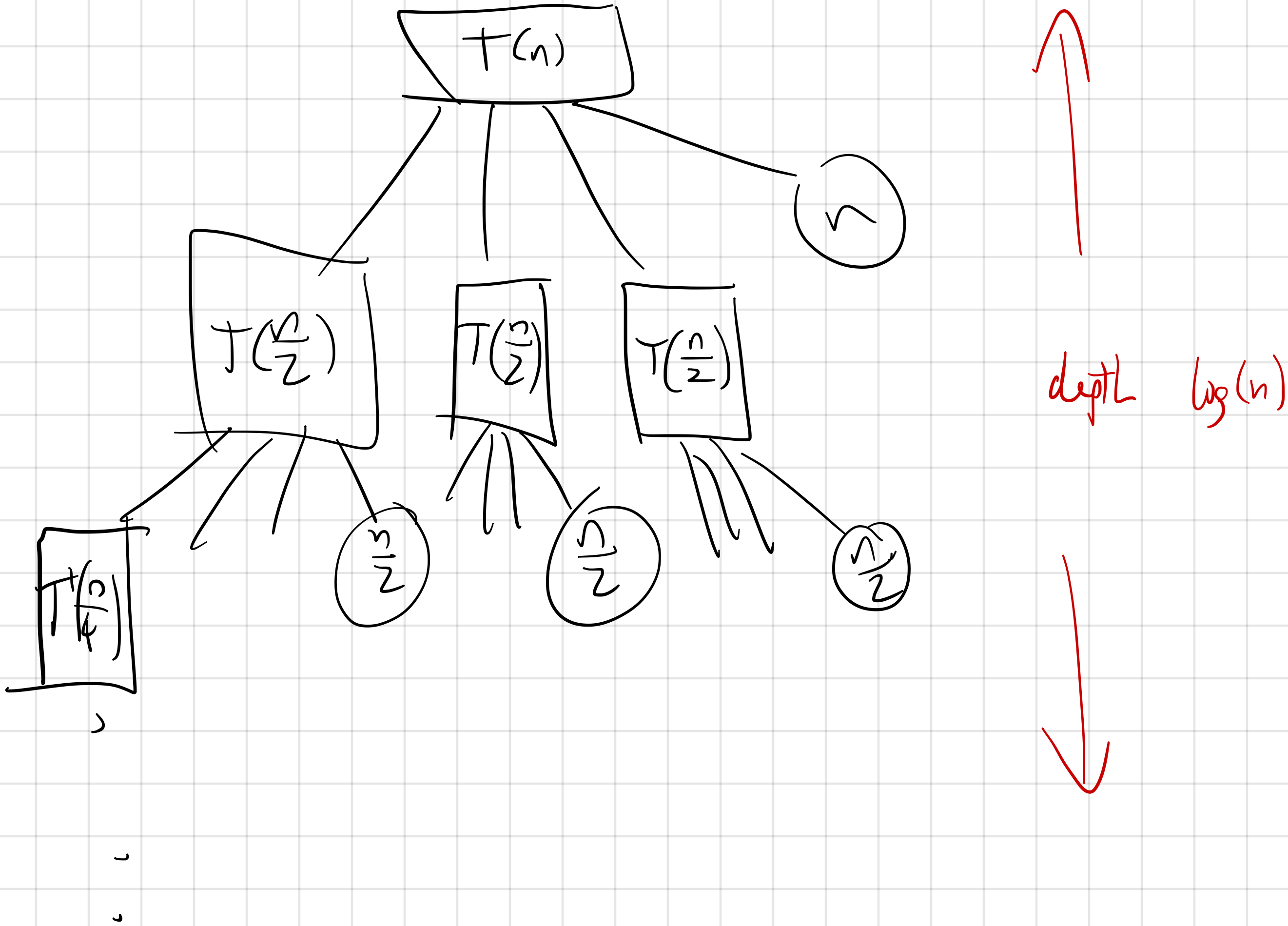
<u>KARATSUBA(P_0, P_1)</u>: If degree(P_0), degree$(P_1) < n$ and $n = 1$: just multiply them.

Else

$$\left.\begin{array}{l} C_0(x) = A_0(x) A_1(x) \\ C_1(x) = (A_0 + B_0)(A_1 + B_1) \\ C_2(x) = B_0 \cdot B_1 \end{array}\right\} \text{3 lines called Recursively}$$

Output $C_0(x) \cdot x^n + \left[C_1(x) - C_0(x) - C_2(x) \right] \cdot x^{n/2} + C_2(x)$.

$O(n)$



Level d has 3^d nodes labeled with $\frac{n}{2^d}$.

Running time: $$\sum_{d=0}^{\log(n)} 3^d \cdot \frac{n}{2^d}$$

$$= n \cdot \sum_{d=0}^{\log(n)} \left(\frac{3}{2}\right)^d$$

$$= O\left(n \cdot \left(\frac{3}{2}\right)^{\log_2(n)}\right)$$

$$\boxed{\text{Sum(geom series)} = O(\text{largest term})}$$

$$= O\left(n \cdot 2^{\log_2(3/2) \cdot \log_2(n)}\right)$$

$\log_2(3/2) \approx 0.58\ldots$

$$= O\left(n \cdot \underbrace{2^{\log_2(n)}}_{=n}{}^{\cdot \log_2(3/2)}\right) = O\left(n \cdot n^{\log_2(3/2)}\right) = O\left(n^{1.58\ldots}\right)$$