Advanced Algorithms

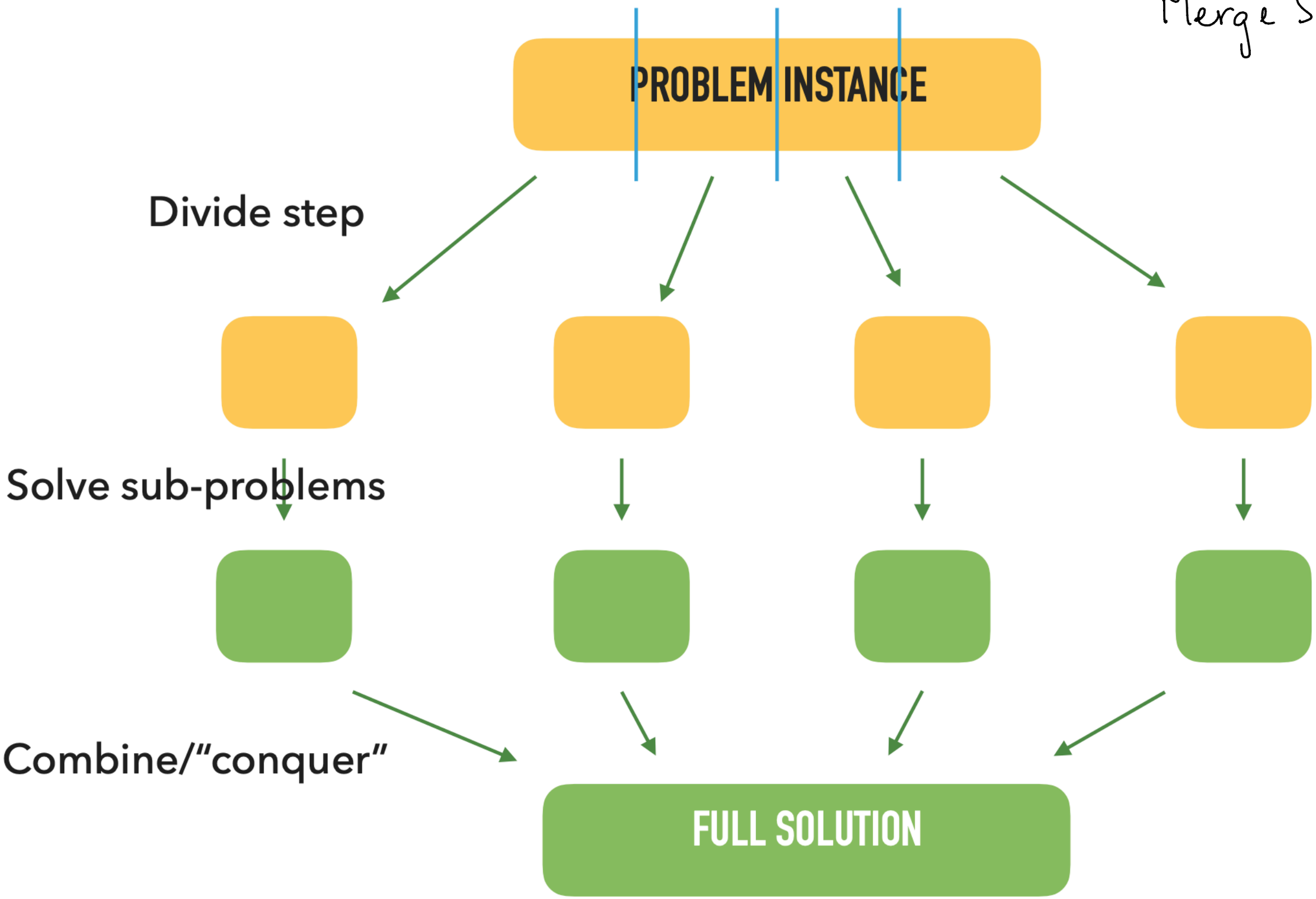
Lecture 4: Divide and Conquer, Recurrences

Announcements

- HW 1 is out — start early!
- HW 0 feedback → 2pm - 3pm
- *Video for last lecture* (hopefully by tomorrow).

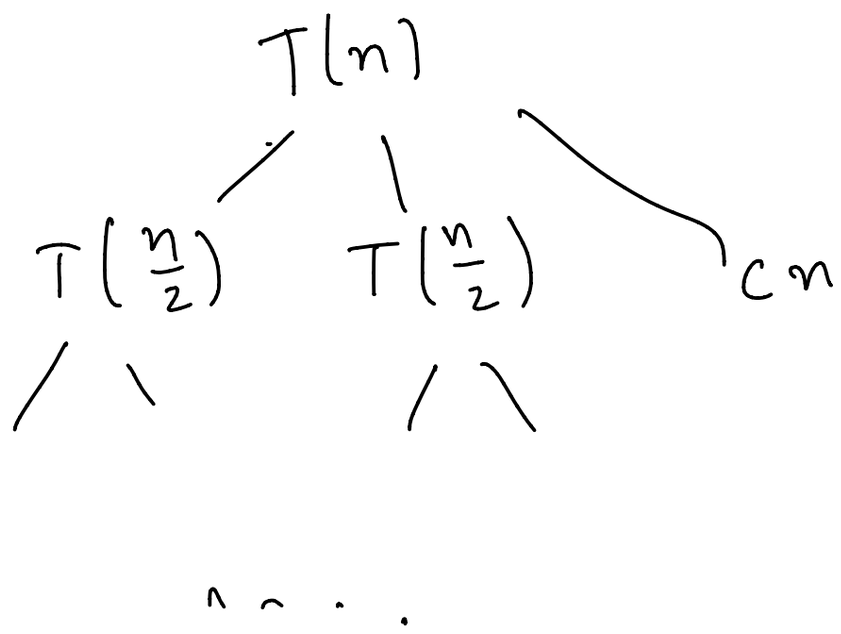
Example last class:

Merge Sort.



Last class

- Merge sort — detailed proof of correctness via induction
- Running time via "recurrence relation" $T(n) = 2T(n/2) + cn$



Recurrences

- General form: f(n) = function (n, f(1), …, f(n-1))
- Finding a "closed form" can be challenging
  - no silver bullet
- many "techniques" — recursion tree, plug-and-chug, guess-and-prove, *master theorem*, Akra-Bazzi theorem, …
- personal favorites…

See notes.

Plug-n-chug: $T(n) = 2T(n/2) + cn$

Base case: $T(1) = 1$

$$T(n) = cn + 2T(n/2)$$

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

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

$$= \ldots = r \cdot cn + 2^r T\left(\frac{n}{2^r}\right)$$

To reach the base case, set $r = \log_2 n$; $cn \log_2 n + n \cdot T(1)$

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

Generative Functionology ← "generative functions" - Wilf

$c \geq 1$

Guess-n-prove: $T(n) = 2T(n/2) + cn$

Start with the guess that $T(n) \leq 2c \cdot n \log(2n)$ → ⊛

$\geq \frac{c}{2} n \log n$

Proof by induction: for $n = 1$; $T(1) = 1$ → true in ⊛.

Inductive step: assume that ⊛ holds for all $n < N$;
& then prove for $n = N$.

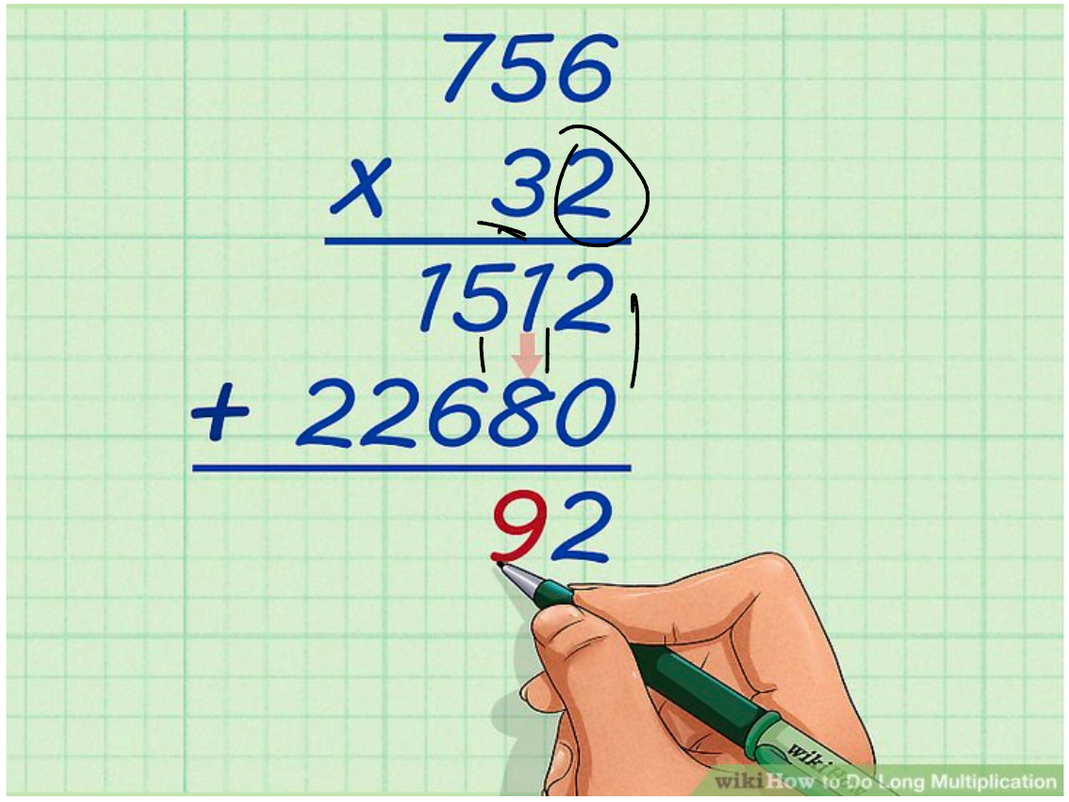
$$T(N) = 2\left(T\left(\frac{N}{2}\right) + cN\right) \leq 2\left[2c \cdot \frac{N}{2} \log N\right] + cN$$

$$= 2cN \log N + cN \overset{\log 2}{=} 2cN(\log N + 1) \leq 2cN \log(2N)$$

Example: integer multiplication

Problem: given two n-digit numbers $a = a_1a_2\ldots a_n$, and $b=b_1b_2\ldots b_n$, find the product a * b.

Elementary school algorithm



$$752 \times (30 + 2)$$

$$= \begin{cases} 752 \times 2 \\ 752 \times 3 \times 10 \\ \cdots \end{cases}$$

$$\begin{array}{r} a_1\ a_2\ \cdots\ a_n \\ \times\ b_1\ b_2\ \cdots\ b_n \\ \hline \left\{ \begin{array}{cccc} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & \end{array} \right. \quad n \end{array}$$

$O(n^2)$ time.

"Idea": Can we use mult. of $\frac{n}{2}$-digit #'s to get prod. of n-digit #s.

Divide and conquer?

$a = A_1 \; n \; A_2$ (each $n/2$) $= A_1 \cdot 10^{n/2} + A_2$

$b = B_1 \; B_2$ (each $n/2$) $= B_1 \cdot 10^{n/2} + B_2$

n

$$a \cdot b = A_1 B_1 \cdot 10^{n} + (A_1 B_2 + A_2 B_1) 10^{n/2} + A_2 B_2$$

To compute this "combine" step, we have 4 add ops each involving $\approx 2n$ digits. $\rightarrow O(n)$ time.

Know that base case is $T(1)=1$

Recurrence: $T(n) = 4T\left(\frac{n}{2}\right) + c \cdot n$

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

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

$$= cn + 2cn + 4^2\left[c \cdot \frac{n}{2^2} + 4T\left(\frac{n}{2^3}\right)\right]$$

$$= cn + 2cn + 2^2 \cdot cn + 4^3 T\left(\frac{n}{2^3}\right) = \cdots$$

$$= cn + 2cn + \ldots + 2^{r-1} \cdot cn + 4^r T\left(\frac{n}{2^r}\right)$$

Again, set $r = \log_2 n$.

$\frac{n}{2^r} = 1 \Leftrightarrow r = \log_2 n$

$$T(n) = \underbrace{cn + 2cn + \dots + 2^{r-1} \cdot cn}_{\text{III}} + 4^r T(1)$$

III

2^{2r}

$$= cn\left(2^r - 1\right) + \left(2^r\right)^2$$

$$= cn \cdot (n-1) + n^2 .$$

$$= \Theta(n^2)$$

$T(n) = 2T(\frac{n}{2}) + cn$ ⟹ $n \log n$

$T(n) = 4T(\frac{n}{2}) + cn$ ⟹ n^2

$\log_2 3$ ⟹ $n^{1.58\ldots}$

Better algorithm?

Karatsuba's idea: $(A_1 \, 10^{n/2} + A_2)(B_1 \cdot 10^{n/2} + B_2)$

need: $\underline{A_1 B_1}$; $\underline{A_2 B_2}$ and $\underline{(A_1 B_2 + A_2 B_1)}$

✓ ✓ ✓

$(A_1 + A_2)(B_1 + B_2)$

$(A_1 - A_2)(B_1 - B_2)$

$\rightarrow$ $A_1 B_1 + A_2 B_2$ & $A_1 B_2 + A_2 B_1$

$A_1 B_1$

$T(n) = 3T(\frac{n}{2}) + c \cdot n$

↑ 4

Median/order finding

Problem: given *n* (distinct, unsorted) integers A[0], A[1], …, A[n-1] and parameter *k*, find the *k*'th smallest integer

- An easy bound?
- Divide and conquer?

Almost-median

What if… for any array of size *n,* we have a procedure that:

(a) runs in O(n) time, and

(b) returns an element that is an "almost median", i.e., it is the *k*'th largest in the array, for $n/4 < k < 3n/4$

Recurrence fun

Ackermann functions

$T(m,n) := fn\{ T(m-1, n-1), \ldots$

$T(1) = 1.$

$\rightsquigarrow \quad T(n) = \underline{T(n-1) + T\left(\frac{n}{2}\right)} + 1 \quad -$ (I)

Fibonacci: $T(n) = \underline{T(n-1)} + \underline{T(n-2)}$ $\quad T(0) = T(1) = 1$

(II)

$(1.61\ldots)^n$ $\quad \nearrow \quad > 2\,T(n-2)$

$\sim \phi^n$ $\quad > 2^2\,T(n-4) \ldots \ldots$

$\downarrow$

.. golden ratio .. $\quad > 2^{n/2}$

$$T(n) = T(n-1) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1 \qquad T(1) = 1$$

$$= T(n-2) + T\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 2$$

$$= T(n-3) + T\left(\left\lfloor \frac{n-2}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 3$$

$$= \vdots$$

Can T be cn?

$$\text{RHS: } c(n-1) + c \cdot \frac{n}{2} + 1 > cn$$

$\{c \cdot n^{10}\}$? ; RHS: $c(n-1)^{10} + \left(\frac{n}{2}\right)^{10} + 1$

$n^{10} - 10n^{9} + \cdots$