Jan 26, 2016

HW#1 is extended to Tue Feb 2.

Solving Recurrence Relations.

Def A recurrence relation is a sequence $a_1, a_2, \cdots, a_n, \cdots$ that is defined recursively (an element is defined involving its previous element(s)).

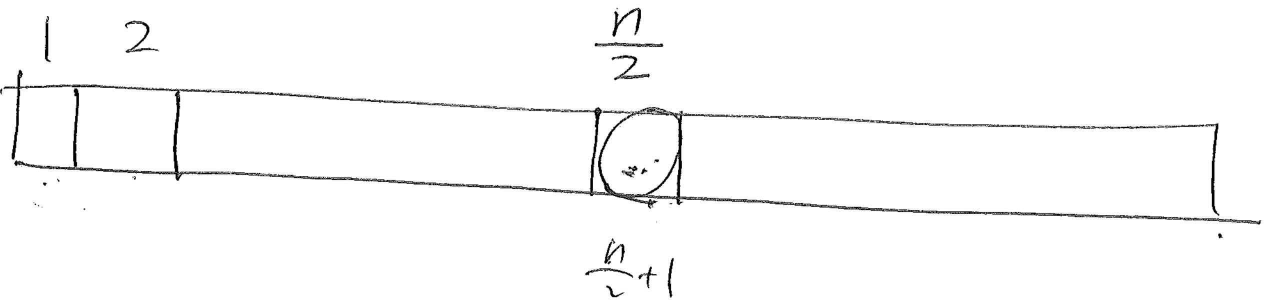
$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$T(1), T(2), T(3), \cdots\ \cdots T(n), \cdots \cdots$$

We are interested in finding an analytical form of the sequence, a_n as a function of n.

$\boxed{\text{SORTED}}$

Given an array $A[1..n]$ of n distinct integers from the set $\{1, 2, \cdots, n+1\}$, find the missing integer



$$\begin{cases} T(n) = 1 + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \\ T(n) = 1 \quad \text{for } n \leq 1 \end{cases}$$

Brute-force

Assume $n = 2^k$ $\qquad k = \lg_2 n = \lg n$

$$\underline{T(2^k)} = T(2^{k-1}) + 1$$

$$= [T(2^{k-2}) + 1 \quad] + 1$$

$$= \underline{T(2^{k-2})} + 2$$

$$= [T(2^{k-3}) + 1 \quad] + 2$$

$$= T(2^{k-3}) + 3$$

$$= \cdots = T(2^{k-k}) + k$$

$$= T(1) + k = 1 + k$$

$$= 1 + \lg n$$

Thus $T(n) = 1 + \lg n = \Theta(\lg n)$

~~insertion sort~~. Bubble Sort.

Given an array $A[1..n]$ of n distinct integers.

~~Paper~~ Bubble sort Strategy.

Find the largest element $A[j]$

Sway $A[j]$ with $A[n]$

Recursively sort $A[1 \ldots n-1]$

$$\begin{cases} T(n) = n + T(n-1) \\ T(1) = 1 \end{cases}$$

④

$T(n) = T(n-1) + n$

$= [T(n-2) + (n-1)] + n$

$= T(n-2) + (n-1) + n$

$= [T(n-3) + (n-2)] + (n-1) + n$

$= T(n-3) + (n-2) + (n-1) + n$

$= \cdots =$

$= T(n-(n-1)) + (n-(n-2)) + \cdots + n$

$= T(1) + 2 + 3 + \cdots + n$

$= 1 + 2 + 3 + \cdots + n$

$= \frac{n \cdot (n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = \Theta(n^2)$

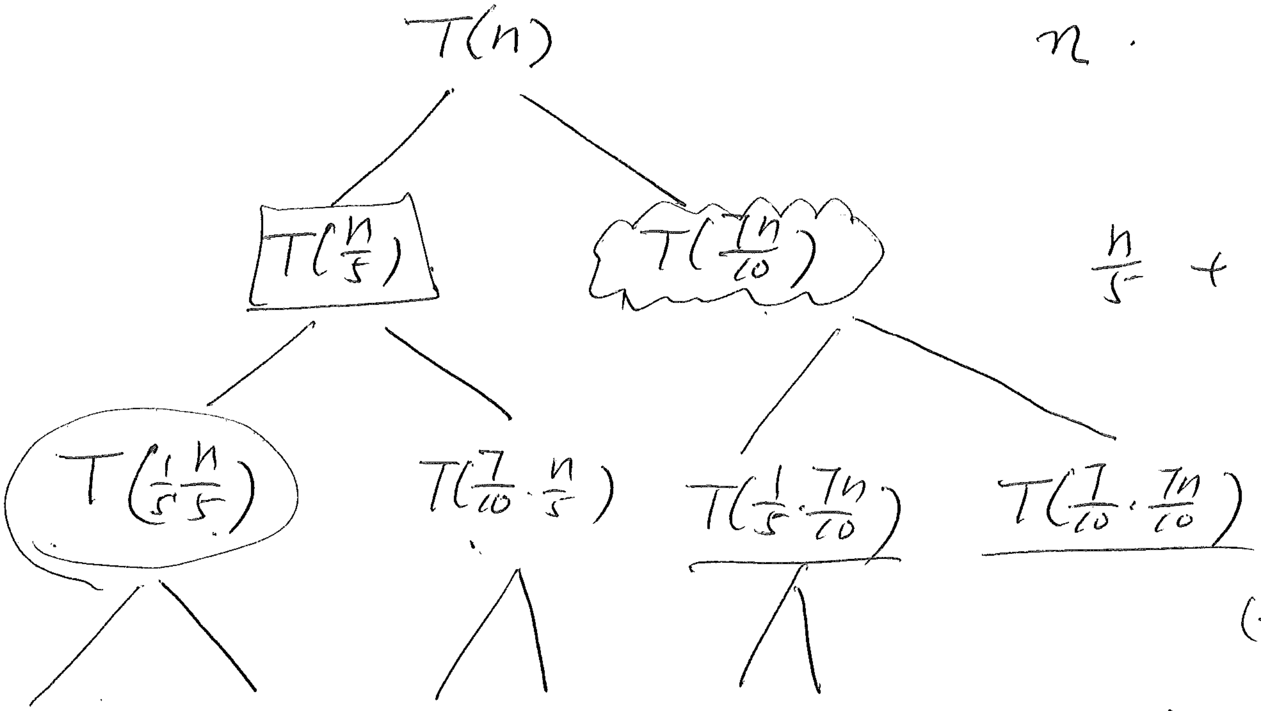
$$\begin{array}{rl} & 1 + 2 + 3 + \cdots + n = S \\ + & n + (n-1) + (n-2) + \cdots + 1 = S \\ \hline & = 2S \end{array}$$

$n(n+1)$

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

2. Recursion Tree Method.

$$\begin{cases} T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + n \\ T(n) = 1 \quad \text{if } n \text{ is constant.} \end{cases}$$



n.

$\frac{n}{5} + \frac{7n}{10} = \frac{9}{10}n$.

$$\begin{aligned} &(\tfrac{1}{5})^2 n + (\tfrac{7}{10} \cdot \tfrac{1}{5})n \\ &+ \tfrac{1}{5} \cdot \tfrac{7}{10} \cdot n \\ &+ (\tfrac{7}{10})^2 n \\ \hline &(\tfrac{9}{10})^2 n \end{aligned}$$

$$T(n) = n + (\tfrac{9}{10})n + (\tfrac{9}{10})^2 n + \cdots$$

$$\leq n \sum_{j=0}^{\infty} (\tfrac{9}{10})^j = n \frac{1 - (\frac{9}{10})^{\infty}}{1 - \frac{9}{10}} = 10n$$

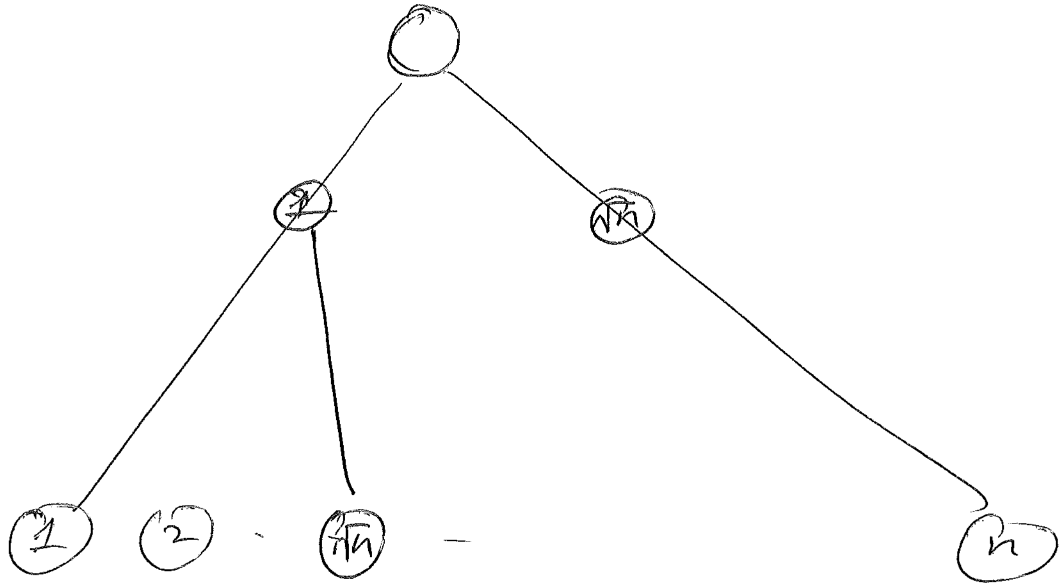
$$= \Theta(n)$$

$$T(n) = T(\tfrac{n}{2}) + 1$$

$$\log n \left\{ \begin{array}{c} T(n) \\ \downarrow \\ T(\frac{n}{2}) \\ \downarrow \\ T(\frac{n}{4}) \\ \downarrow \\ T(\frac{n}{8}) \\ \downarrow \\ \vdots \end{array} \right. \qquad \begin{array}{c} 1 \\ + \\ 1 \\ + \\ 1 \\ + \end{array} = \Theta(\log n)$$

$$\downarrow$$

Very Short Tree on $U = \{1, 2, \cdots n\}$



$$T(n) = T(\sqrt{n}) + 1$$

$$n = 2^{2^k}$$

intelligient.

Guess and ~~Substitution~~ Substitution

Recall the running time of Merge Sort.

$$T(n) = 1 + 2T(\tfrac{n}{2}) + n$$

$$\begin{cases} T(n) = 2T(\frac{n}{2}) + n \\ T(n) = 1 \quad n \text{ is constant} \end{cases}$$

$$\begin{cases} T(n) = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n \\ T(n) = 1 \quad \text{for } n \text{ constant} \end{cases}$$

① Brute force.

Assume $n = 2^k \Rightarrow k = \lg n$

$$\frac{2^k}{2^1} = 2^{k-1}$$

$$T(n) = T(2^k) = 2T(2^{k-1}) + 2^k$$

$$= 2[2T(2^{k-2}) + 2^{k-1}] + 2^k$$

$$= 2^2 T(2^{k-2}) + 2^k + 2^k$$

$$= 2^2 [2T(2^{k-3}) + 2^{k-2}] + 2^k + 2^k$$

$$= 2^3 T(2^{k-3}) + \underbrace{2^k + 2^k + 2^k}_{3}$$

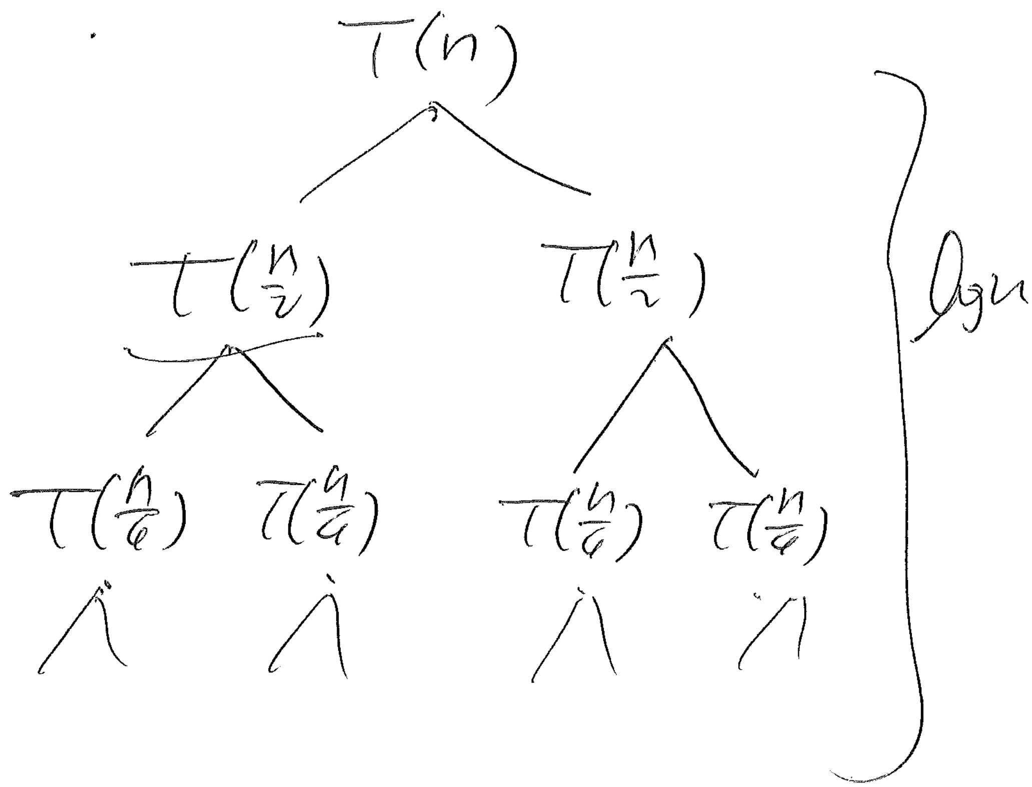
$$\cdots$$

$$= 2^k T(2^{k-k}) + \underbrace{2^k + \cdots + 2^k}_{k}$$

$$= 2^k + k 2^k$$

$$= (k+1) 2^k = (\lg n + 1) n$$

$$= n \lg n + n = \Theta(n \lg n)$$



n

$\frac{n}{2} + \frac{n}{2} = n$

$4 \cdot \frac{n}{4} = n$

$= n \lg n$

Guess and Substitution

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

$\Theta(n)$

Guess $T(n) = \boxed{\Theta(n\log n)}$

$\Theta(n^2)$

This means $T(n) = O(n\log n)$ and $\Omega(n\log n)$

We will prove $T(n) = O(n\log n)$ for illustrations.

Note that $T(n) = O(n\log n)$ implies that there exists c and n_0, $c, n_0 > 0$ such that for all $n \geq n_0$, $T(n) \leq c \cdot n\log n$

Claim. For all $n \geq \underset{\uparrow \atop n_0}{2}$, $T(n) \leq \underset{\uparrow \atop c}{2} \cdot n\log n$

Basis $n=2$ $\quad T(2) = 2T(1)+2$

$$= 4 \overset{?}{\leq} 2 \cdot n \log n$$

$$2 \cdot 2 \log 2 = 4$$

I.S.

Assume for all $2 \leq n < k$ $\quad T(n) \leq 2n \log n$

Need to show $T(k) \leq 2k \log k$

$$T(k) = 2T(\tfrac{k}{2}) + k$$

$$\leq 2 \cdot \tfrac{k}{2} \log \tfrac{k}{2} + k$$

$$= k(\log k - \log 2) + k$$

$$= k \log k - k + k = k \log k \leq 2k \log k$$

Hence the induction goes through and the claim is correct.

$$\log(xy) = \log x + \log y$$

$$\log\left(\frac{k}{2}\right) = \log\left(k \cdot \frac{1}{2}\right) = \log k + \log\frac{1}{2}$$

$$\log(x^y) = y\log x$$

$$\log\frac{1}{2} = \log(2^{-1}) = -1\log 2$$

$$\log\frac{k}{2} = \log k - \log 2$$

The short cut

$f \in O(g)$, $\exists\, c, n_0 > 0$ and for all $n \geq n_0$, $f \leq c \cdot g$

---

if c works, than $2c$ also works.
large c's always work

if n_0 works does $2n_0$ work?

Thus for c, n_0 notation, larger c and n_0 always work. When we only care about the asymptotic behavior, we don't need to have the tight c and n_0

Substitution.

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

$$\le 2\cdot\left(c\,\tfrac{n}{2}\log\tfrac{n}{2}\right) + n$$

$$= cn(\log n - \log 2) + n$$

$$= cn\log n - cn + n$$

$$\nleq cn\log n$$

Observe that as long as $c \ge 2$

$-cn+n < 0$, and

$$T(n) \le cn\log n - cn + n \le cn\log n$$