Shelby Williams **CSCE 500** **September 22, 2021**

2.3-3

Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2\,, \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

is $T(n) = n \lg n$.

Base Case:

If $n = 2$, then $T(2) = 2$ from the recurrence relation and $T(2) = 2\ lg\ 2 = 2(1) = 2$ from $T(n) = n \lg n$.

Hypothesis Case:

Assume $T(n)$ is true, if $n = 2^k$ for $k > 1$.

Inductive Case:

If $n = 2^k$, then show that $T(2^k) = 2^{k+1} \lg 2^{k+1}$

$$T(2^{k+1}) = 2T(2^{k+1}/2) + 2^{k+1} = 2T(2^k) + 2^{k+1} = 2(2^k \lg 2^k) + 2^{k+1}$$
$$= 2^{k+1} \lg 2^k + 2^{k+1}$$

$$T(2^{k+1}) = 2^{k+1}[\lg 2^k + 1] = 2^{k+1}[\lg 2^k + \lg 2^1] = 2^{k+1}[\lg(2^k \cdot 2^1)\,]$$

Therefore, $T(2^k) = 2^{k+1} \lg 2^{k+1}$.

2.3-6

Observe that the **while** loop of lines 5–7 of the INSERTION-SORT procedure in Section 2.1 uses a linear search to scan (backward) through the sorted subarray $A[1\,..\,j-1]$. Can we use a binary search (see Exercise 2.3-5) instead to improve the overall worst-case running time of insertion sort to $\Theta(n \lg n)$?

No, the overall worst-case running time of insertion sort cannot be $\theta(n \lg n)$ because although the binary search is $\theta(\lg n)$, the worst case for moving the new elements, prior to inserting the new element in the array, is $\theta(n)$. This linear time is not affected by binary search.

4.3-3

We saw that the solution of $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $O(n \lg n)$. Show that the solution of this recurrence is also $\Omega(n \lg n)$. Conclude that the solution is $\Theta(n \lg n)$.

Given: $T(n) = 2T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n$ is $O(n)$.

Show that $T(n) = \Omega(n)$.

Since $\left\lfloor\frac{n}{2}\right\rfloor \leq \frac{n}{2}$, $T(n) \geq 2T\left(\frac{n}{2}\right) + n$.

Solve by substitution. Assume that $T(n) \geq cn\lg n$.

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

$$T(n) \geq 2\left(d\frac{n}{2}\lg\frac{n}{2}\right) + n.$$

$$T(n) \geq dn\lg\frac{n}{2} + n.$$

$$T(n) \geq dn\lg n - dn + n.$$

$T(n) \geq dn\lg n - dn + n$, if $-dn\ +\ n \geq 0;\ (d \leq 1)$

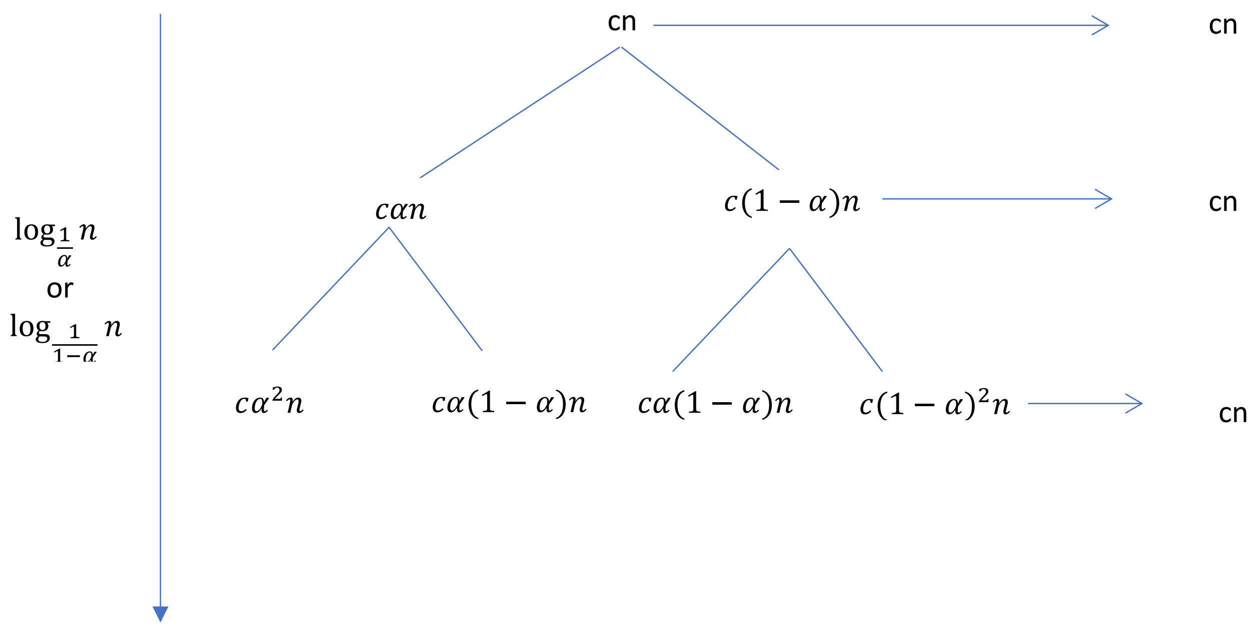
$$T(n) \geq dn\lg n$$

Therefore, $T(n) = \Omega(n)$.

By Theorem 3.1, since $T(n) = O(n)\ and\ T(n) = \Omega(n)$, then $\boldsymbol{T(n) = \theta(n)}$.

4.4-9

Use a recursion tree to give an asymptotically tight solution to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ and $c > 0$ is also a constant.



Total: $T(n) = O(n \lg n)$

If $\alpha > \frac{1}{2}$, then the longest path is: $cn \rightarrow \ c\alpha n \rightarrow c\alpha^2 n \ldots \rightarrow 1, k = \log_{\frac{1}{\alpha}} n$

If $\alpha > \frac{1}{2}$, then the shortest path is: $cn \rightarrow \ c(1-\alpha)n \rightarrow c(1-\alpha)^2 n \ldots \rightarrow 1, k = \log_{\frac{1}{1-\alpha}} n$

$T(n) = O(n \lg n)$ from longest path.
$T(n) = \Omega(n \lg n)$ from shortest path.

From Theorem 3.1, if $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, then $\boldsymbol{T(n) = \Theta(n \lg n)}$.

4-1 Recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \le 2$. Make your bounds as tight as possible, and justify your answers.

f. $T(n) = 2T(n/4) + \sqrt{n}$.

Using the master method: $a = 2\ and\ b = 4, f(n) = \sqrt{n}$

Case 2: $\boldsymbol{T(n) = \theta\left(\sqrt{n}\lg n\right)}$

$$0 \le c_1\sqrt{n}\lg n \le \sqrt{n}\lg n \le c_2\sqrt{n}\lg n$$

$$\text{Let } c_1 = \frac{1}{10}\ and\ c_2 = 10.$$

4-3 More recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

f. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

At the root node, the cost = n.
One level below the root, T(n) = (n/2) + (n/4) +(n/8), the cost = (7/8)n
Two levels below the root, T(n) = (n/4) + (n/8) +(n/16)+ (n/8) + (n/16) +(n/32)+ (n/16) + (n/32) +(n/64), the cost = $(7/8)^2$n.

The total cost excluding the leaves is: $\sum_{i=0}^{k}\left(\frac{7}{8}\right)^i n \le \sum_{i=0}^{\infty}\left(\frac{7}{8}\right)^i n = \frac{1}{1-\frac{7}{8}}n = 8n$

The longest path is: $n \to \frac{n}{2} \to \frac{n}{4} \to \cdots \frac{n}{2^k} = 1, k = \lg n.$

The shortest path is: $n \to \frac{n}{8} \to \frac{n}{64} \to \cdots \frac{n}{8^k} = 1, k = \log_8 n = \frac{lg\, n}{\lg 8} = \frac{1}{3}\lg n$

The total number of leaves at the longest path if the tree was completely filled is $lg\ lg\ n$.

$$T(n) = O(n\lg n) + \theta(\lg\lg n) = O(n\lg n)$$

$$T(n) = \Omega(n\lg n) + \theta(\lg\lg n) = \Omega(n\lg n)$$

Therefore, $\boldsymbol{T(n) = \theta(n\lg n)}$.

Use the substitution method to solve $T(n) \ = T(n/3) + T(2n/3) + cn$.

Assume that $T(n) = O(n \lg n)$.

Therefore, $T(n) \leq kn \lg n$,for some positive constant k.

$T(n) \leq T(n/3) + T(2n/3) + cn$

$T(n) \leq k(n/3)\lg(n/3) + k(2n/3)\lg(2n/3) + cn$

$T(n) = k3(n/3)\lg n - k((n/3)\lg 3 + (2n/3)\lg(3/2) + cn$

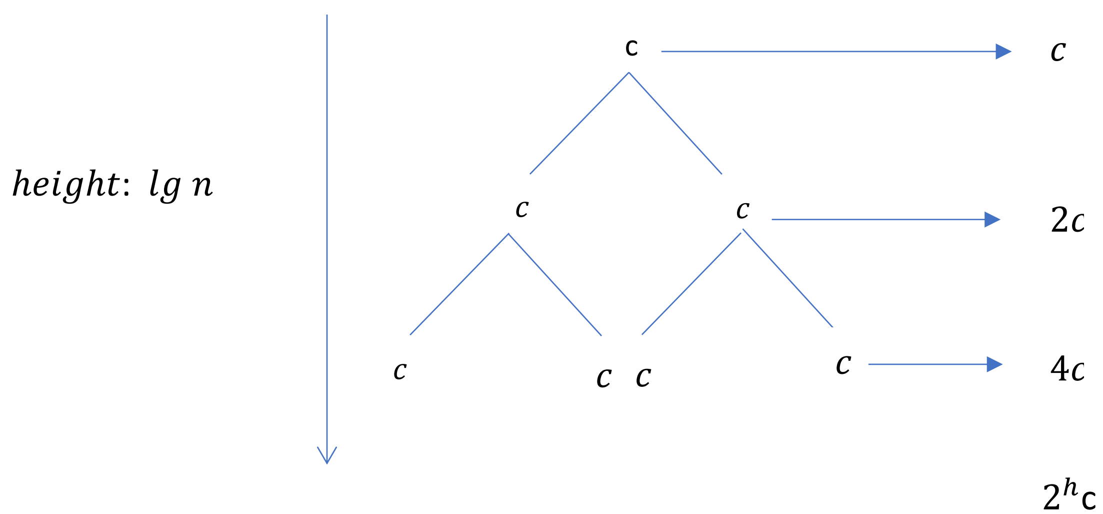
$T(n) = kn \lg n - kn(\lg 3 - (2/3)\lg 2 + cn$

$T(n) \leq kn \lg n$. for $b \geq c/(\lg 3 - 2/3 \lg 2)$

Therefore, $\boldsymbol{T(n) = O(n \lg n)}$**.**

Use the recursion-tree method to solve

1. $T(n) \ = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + c$.



The number of leaves is $2^h = 2^{\lg n} = n$.

There are n-1 internal nodes, so $T(n) = 2n - 1$

$T(n) = c + 2c + 4c + \cdots + nc = \boldsymbol{O(n)}$.

2. $T(n) = 2T(n/2) + cn(\lg n)$.

3. $T(n) = T(n/3) + T(2n/3) + n(\lg n)$.