

Rosen, Discrete Mathematics and Its Applications, 7th edition
Extra Examples
Section 8.3—Divide-and-Conquer Algorithms and Recurrence Relations



— Page references correspond to locations of Extra Examples icons in the textbook.

p.528, icon at Example 1

#1. Suppose $f(n) = 3f(n/2) + 4$ and $f(1) = 5$. Find $f(8)$.

Solution:

$$\begin{aligned}f(2) &= 3f(2/2) + 4 = 3 \cdot 5 + 4 = 19, \\f(4) &= 3f(4/2) + 4 = 3 \cdot 19 + 4 = 57 + 4 = 61, \\f(8) &= 3f(8/2) + 4 = 3 \cdot 61 + 4 = 183 + 4 = 187.\end{aligned}$$

p.528, icon at Example 1

#2. Suppose $f(n) = 2f(n/3) - 1$ and $f(1) = 2$. Find $f(9)$.

Solution:

$$\begin{aligned}f(3) &= 2f(3/3) - 1 = 2 \cdot 2 - 1 = 3, \\f(9) &= 2f(9/3) - 1 = 2 \cdot 3 - 1 = 5.\end{aligned}$$

p.528, icon at Example 1

#3. Suppose $f(n) = 5f(n/2) + 2n - 1$ and $f(4) = 40$. Find $f(1)$.

Solution:

First use $f(4)$ to find $f(2)$: $f(4) = 5f(4/2) + 2 \cdot 4 - 1$. Therefore $40 = 5f(2) + 7$, or $f(2) = 33/5$.

Then use $f(2)$ to find $f(1)$: $f(2) = 5f(2/2) + 2 \cdot 2 - 1$.

Therefore $33/5 = 5f(1) + 3$, or $f(1) = 18/25$.

p.531, icon at Example 6

#1. Suppose $f(n) = 2f(n/3) + 3$. Find a big-oh function for f .

Solution:

Using Theorem 1 of Section 7.3, $f(n)$ is $O(n^{\log_3 2})$.

p.531, icon at Example 6

#2. A recursive algorithm for finding the maximum of a list of numbers divides the list into three equal (or nearly equal) parts, recursively finds the maximum of each sublist, and then finds the largest of these three maxima. Let $f(n)$ be the total number of comparisons needed to find the maximum of a list of n numbers (n a power of 3). Set up a recurrence relation for $f(n)$ and give a big-oh estimate for f .

Solution:

A recurrence relation for the number of steps in this algorithm with an input of size $n > 1$ (n a power of 3) is

$$f(n) = 3f(n/3) + 2$$

(assuming that two operations are required to compare the three maxima). Using Theorem 1 of Section 7.3, $f(n)$ is $O(n^{\log_3 3})$. But $n^{\log_3 3} = n^1 = n$. Therefore $f(n)$ is $O(n)$.
