$$\begin{aligned}
T(b^k) &= aT(b^{k-1}) + f(b^k)\\
&= a[aT(b^{k-2}) + f(b^{k-1})] + f(b^k) = a^2T(b^{k-2}) + af(b^{k-1}) + f(b^k)\\
&= a^2[aT(b^{k-3}) + f(b^{k-2})] + af(b^{k-1}) + f(b^k)\\
&= a^3T(b^{k-3}) + a^2f(b^{k-2}) + af(b^{k-1}) + f(b^k)\\
&= \cdots\\
&= a^kT(1) + a^{k-1}f(b^1) + a^{k-2}f(b^2) + \cdots + a^0f(b^k)\\
&= a^k[T(1) + \sum_{j=1}^{k} f(b^j)/a^j].
\end{aligned}$$

Since $a^k = a^{\log_b n} = n^{\log_b a}$, we get the following formula for the solution to recurrence (B.14) for $n = b^k$:

$$T(n) = n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} f(b^j)/a^j]. \qquad \textbf{(B.15)}$$

Obviously, the order of growth of solution $T(n)$ depends on the values of the constants a and b and the order of growth of the function $f(n)$. Under certain assumptions about $f(n)$ discussed in the next section, we can simplify formula (B.15) and get explicit results about the order of growth of $T(n)$.

Smoothness Rule and the Master Theorem We mentioned earlier that the time efficiency of decrease-by-a-constant-factor and divide-and-conquer algorithms is usually investigated first for n's that are powers of b. (Most often $b = 2$, as it is in binary search and mergesort; sometimes $b = 3$, as it is in the better algorithm for the fake-coin problem of Section 4.4, but it can be any integer greater than or equal to 2.) The question we are going to address now is when the order of growth observed for n's that are powers of b can be extended to all its values.

DEFINITION Let $f(n)$ be a nonnegative function defined on the set of natural numbers. $f(n)$ is called ***eventually nondecreasing*** if there exists some nonnegative integer n_0 so that $f(n)$ is nondecreasing on the interval $[n_0, \infty)$, i.e.,

$$f(n_1) \le f(n_2) \quad \text{for all } n_2 > n_1 \ge n_0.$$

For example, the function $(n - 100)^2$ is eventually nondecreasing, although it is decreasing on the interval [0, 100], and the function $\sin^2 \frac{\pi n}{2}$ is a function that is not eventually nondecreasing. The vast majority of functions we encounter in the analysis of algorithms *are* eventually nondecreasing. Most of them are, in fact, nondecreasing on their entire domains.

DEFINITION Let $f(n)$ be a nonnegative function defined on the set of natural numbers. $f(n)$ is called ***smooth*** if it is eventually nondecreasing and

$$f(2n) \in \Theta(f(n)).$$

It is easy to check that functions which do not grow too fast, including $\log n$, n, $n \log n$, and n^α where $\alpha \geq 0$, are smooth. For example, $f(n) = n \log n$ is smooth because

$$f(2n) = 2n \log 2n = 2n(\log 2 + \log n) = (2 \log 2)n + 2n \log n \in \Theta(n \log n).$$

Fast-growing functions, such as a^n where $a > 1$ and $n!$, are not smooth. For example, $f(n) = 2^n$ is not smooth because

$$f(2n) = 2^{2n} = 4^n \notin \Theta(2^n).$$

THEOREM 3 Let $f(n)$ be a smooth function as just defined. Then, for any fixed integer $b \geq 2$,

$$f(bn) \in \Theta(f(n)),$$

i.e., there exist positive constants c_b and d_b and a nonnegative integer n_0 such that

$$d_b f(n) \leq f(bn) \leq c_b f(n) \quad \text{for } n \geq n_0.$$

(The same assertion, with obvious changes, holds for the O and Ω notations.)

PROOF We will prove the theorem for the O notation only; the proof of the Ω part is the same. First, it is easy to check by induction that if $f(2n) \leq c_2 f(n)$ for $n \geq n_0$, then

$$f(2^k n) \leq c_2^k f(n) \quad \text{for } k = 1, 2, \ldots \text{ and } n \geq n_0.$$

The induction basis for $k = 1$ checks out trivially. For the general case, assuming that $f(2^{k-1}n) \leq c_2^{k-1} f(n)$ for $n \geq n_0$, we obtain

$$f(2^k n) = f(2 \cdot 2^{k-1}n) \leq c_2 f(2^{k-1}n) \leq c_2 c_2^{k-1} f(n) = c_2^k f(n).$$

This proves the theorem for $b = 2^k$.

Consider now an arbitrary integer $b \geq 2$. Let k be a positive integer such that $2^{k-1} \leq b < 2^k$. We can estimate $f(bn)$ above by assuming without loss of generality that $f(n)$ is nondecreasing for $n \geq n_0$:

$$f(bn) \leq f(2^k n) \leq c_2^k f(n).$$

Hence, we can use c_2^k as a required constant for this value of b to complete the proof. ■

The importance of the notions introduced above stems from the following theorem.

THEOREM 4 ***(Smoothness Rule)*** Let $T(n)$ be an eventually nondecreasing function and $f(n)$ be a smooth function. If

$$T(n) \in \Theta(f(n)) \quad \text{for values of } n \text{ that are powers of } b,$$

where $b \geq 2$, then

$$T(n) \in \Theta(f(n)).$$

(The analogous results hold for the cases of O and Ω as well.)

PROOF We will prove just the O part; the Ω part can be proved by the analogous argument. By the theorem's assumption, there exist a positive constant c and a positive integer $n_0 = b^{k_0}$ such that

$$T(b^k) \leq cf(b^k) \quad \text{for } b^k \geq n_0,$$

$T(n)$ is nondecreasing for $n \geq n_0$, and $f(bn) \leq c_b f(n)$ for $n \geq n_0$ by Theorem 3. Consider an arbitrary value of $n, n \geq n_0$. It is bracketed by two consecutive powers of b: $n_0 \leq b^k \leq n < b^{k+1}$. Therefore,

$$T(n) \leq T(b^{k+1}) \leq cf(b^{k+1}) = cf(bb^k) \leq cc_b f(b^k) \leq cc_b f(n).$$

Hence, we can use the product cc_b as a constant required by the $O(f(n))$ definition to complete the O part of the theorem's proof. ■

Theorem 4 allows us to expand the information about the order of growth established for $T(n)$ on a convenient subset of values (powers of b) to its entire domain. Here is one of the most useful assertions of this kind.

THEOREM 5 ***(Master Theorem)*** Let $T(n)$ be an eventually nondecreasing function that satisfies the recurrence

$$\begin{aligned} &T(n) = aT(n/b) + f(n) \quad \text{for } n = b^k, \;\; k = 1, 2, \ldots \\ &T(1) = c, \end{aligned}$$

where $a \geq 1$, $b \geq 2$, $c > 0$. If $f(n) \in \Theta(n^d)$ where $d \geq 0$, then

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d, \\ \Theta(n^d \log n) & \text{if } a = b^d, \\ \Theta(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

(Similar results hold for the O and Ω notations, too.)

PROOF We will prove the theorem for the principal special case of $f(n) = n^d$. (A proof of the general case is a minor technical extension of the same argument—see, e.g., [Cor09].) If $f(n) = n^d$, equality (B.15) yields for $n = b^k$, $k = 0, 1, \ldots,$

$$T(n) = n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} b^{jd}/a^j] = n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} (b^d/a)^j].$$

The sum in this formula is that of a geometric series, and therefore

$$\sum_{j=1}^{\log_b n} (b^d/a)^j = (b^d/a)\frac{(b^d/a)^{\log_b n} - 1}{(b^d/a) - 1} \quad \text{if } b^d \neq a$$

and

$$\sum_{j=1}^{\log_b n} (b^d/a)^j = \log_b n \quad \text{if } b^d = a.$$

If $a < b^d$, then $b^d/a > 1$, and therefore

$$\sum_{j=1}^{\log_b n} (b^d/a)^j = (b^d/a)\frac{(b^d/a)^{\log_b n} - 1}{(b^d/a) - 1} \in \Theta((b^d/a)^{\log_b n}).$$

Hence, in this case,

$$\begin{aligned} T(n) &= n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} (b^d/a)^j] \in n^{\log_b a}\Theta((b^d/a)^{\log_b n}) \\ &= \Theta(n^{\log_b a}(b^d/a)^{\log_b n}) = \Theta(a^{\log_b n}(b^d/a)^{\log_b n}) \\ &= \Theta(b^{d \log_b n}) = \Theta(b^{\log_b n^d}) = \Theta(n^d). \end{aligned}$$

If $a > b^d$, then $b^d/a < 1$, and therefore

$$\sum_{j=1}^{\log_b n} (b^d/a)^j = (b^d/a)\frac{(b^d/a)^{\log_b n} - 1}{(b^d/a) - 1} \in \Theta(1).$$

Hence, in this case,

$$T(n) = n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} (b^d/a)^j] \in \Theta(n^{\log_b a}).$$

If $a = b^d$, then $b^d/a = 1$, and therefore

$$\begin{aligned} T(n) &= n^{\log_b a}[T(1) + \sum_{j=1}^{\log_b n} (b^d/a)^j] = n^{\log_b a}[T(1) + \log_b n] \\ &\in \Theta(n^{\log_b a} \log_b n) = \Theta(n^{\log_b b^d} \log_b n) = \Theta(n^d \log_b n). \end{aligned}$$

Since $f(n) = n^d$ is a smooth function for any $d \geq 0$, a reference to Theorem 4 completes the proof. ■

Theorem 5 provides a very convenient tool for a quick efficiency analysis of divide-and-conquer and decrease-by-a-constant-factor algorithms. You can find examples of such applications throughout the book.