★ 4.4: Proof of the master theorem

This section contains a proof of the master theorem (Theorem 4.1). The proof need not be understood in order to apply the theorem.

The proof is in two parts. The first part analyzes the "master" recurrence (4.5), under the simplifying assumption that $T(n)$ is defined only on exact powers of $b > 1$, that is, for $n = 1, b, b^2, \ldots$. This part gives all the intuition needed to understand why the master theorem is true. The second part shows how the analysis can be extended to all positive integers n and is merely mathematical technique applied to the problem of handling floors and ceilings.

In this section, we shall sometimes abuse our asymptotic notation slightly by using it to describe the behavior of functions that are defined only over exact powers of b. Recall that the definitions of asymptotic notations require that bounds be proved for all sufficiently large numbers, not just those that are powers of b. Since we could make new asymptotic notations that apply to the set $\{b^i : i = 0, 1,\ldots\}$, instead of the nonnegative integers, this abuse is minor.

Nevertheless, we must always be on guard when we are using asymptotic notation over a limited domain so that we do not draw improper conclusions. For example, proving that $T(n) = O(n)$ when n is an exact power of 2 does not guarantee that $T(n) = O(n)$. The function $T(n)$ could be defined as

$$T(n) = \begin{cases} n & \text{if } n = 1, 2, 4, 8, \ldots, \\ n^2 & \text{otherwise}, \end{cases}$$

in which case the best upper bound that can be proved is $T(n) = O(n^2)$. Because of this sort of drastic consequence, we shall never use asymptotic notation over a limited domain without making it absolutely clear from the context that we are doing so.

4.4.1 The proof for exact powers

The first part of the proof of the master theorem analyzes the recurrence (4.5)

$T(n) = aT(n/b) + f(n)$,

for the master method, under the assumption that n is an exact power of $b > 1$, where b need not be an integer. The analysis is broken into three lemmas. The first reduces the problem of solving the master recurrence to the problem of evaluating an expression that contains a summation. The second determines bounds on this summation. The third lemma puts the first two together to prove a version of the master theorem for the case in which n is an exact power of b.

Lemma 4.2

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. Define $T(n)$ on exact powers of b by the recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ aT(n/b) + f(n) & \text{if } n = b^i , \end{cases}$$

where i is a positive integer. Then

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) . \qquad (4.6)$$

Proof We use the recursion tree in Figure 4.3. The root of the tree has cost $f(n)$, and it has a children, each with cost $f(n/b)$. (It is convenient to think of a as being an integer, especially when visualizing the recursion tree, but the mathematics does not require it.) Each of these children has a children with cost $f(n/b^2)$, and thus there are a^2 nodes that are distance 2 from the root. In general, there are a^j nodes that are distance j from the root, and each has cost $f(n/b^j)$. The cost of each leaf is $T(1) = \Theta(1)$, and each leaf is at depth $\log_b n$, since $n/b^{\log_b n} = 1$. There are $a^{\log_b n} = n^{\log_b a}$ leaves in the tree.

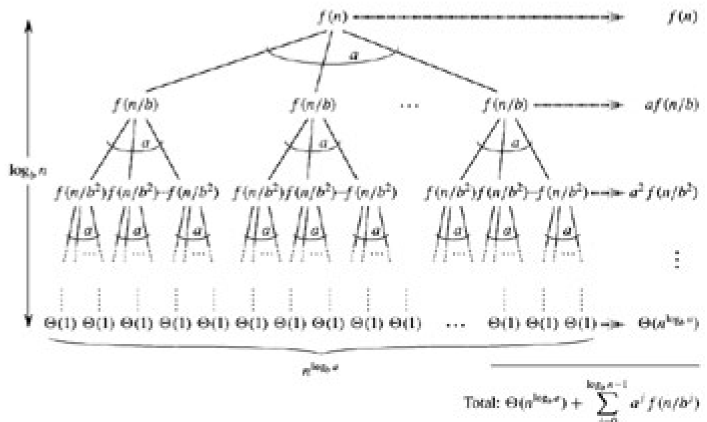


Figure 4.3: The recursion tree generated by $T(n) = aT(n/b) + f(n)$. The tree is a complete a-ary tree with $n^{\log_b a}$ leaves and height $\log_b n$. The cost of each level is shown at the right, and their sum is given in equation (4.6).

We can obtain equation (4.6) by summing the costs of each level of the tree, as shown in the figure. The cost for a level j of internal nodes is $a^j f(n/b^j)$, and so the total of all internal node levels is

$$\sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) .$$

In the underlying divide-and-conquer algorithm, this sum represents the costs of dividing problems into subproblems and then recombining the subproblems. The cost of all the leaves, which is the cost of doing all $n^{\log_b a}$ subproblems of size 1, is $\Theta(n^{\log_b a})$.

In terms of the recursion tree, the three cases of the master theorem correspond to cases in which the total cost of the tree is (1) dominated by the costs in the leaves, (2) evenly distributed across the levels of the tree, or (3) dominated by the cost of the root.

The summation in equation (4.6) describes the cost of the dividing and combining steps in the underlying divide-and-conquer algorithm. The next lemma provides asymptotic bounds on the summation's growth.

Lemma 4.3

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. A function $g(n)$ defined over exact powers of b by

$$(4.7) \quad g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

can then be bounded asymptotically for exact powers of b as follows.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $g(n) = O(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $g(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $af(n/b) \leq cf(n)$ for some constant $c < 1$ and for all $n \geq b$, then $g(n) = \Theta(f(n))$.

Proof For case 1, we have $f(n) = O(n^{\log_b a - \epsilon})$, which implies that $f(n/b^j) = O((n/b^j)^{\log_b a - \epsilon})$. Substituting into equation (4.7) yields

$$(4.8) \quad g(n) = O\left(\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a - \epsilon}\right).$$

We bound the summation within the O-notation by factoring out terms and simplifying, which leaves an increasing geometric series:

$$\begin{aligned}
\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a - \epsilon} &= n^{\log_b a - \epsilon} \sum_{j=0}^{\log_b n - 1} \left(\frac{ab^\epsilon}{b^{\log_b a}}\right)^j \\
&= n^{\log_b a - \epsilon} \sum_{j=0}^{\log_b n - 1} (b^\epsilon)^j \\
&= n^{\log_b a - \epsilon} \left(\frac{b^{\epsilon \log_b n} - 1}{b^\epsilon - 1}\right) \\
&= n^{\log_b a - \epsilon} \left(\frac{n^\epsilon - 1}{b^\epsilon - 1}\right).
\end{aligned}$$

Since b and ϵ are constants, we can rewrite the last expression as $n^{\log_b a - \epsilon} O(n^\epsilon) = O(n^{\log_b a})$. Substituting this expression for the summation in equation (4.8) yields

$$g(n) = O(n^{\log_b a}),$$

and case 1 is proved.

Under the assumption that $f(n) = \Theta(n^{\log_b a})$ for case 2, we have that $f(n/b^j) = \Theta((n/b^j)^{\log_b a})$. Substituting into equation (4.7) yields

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a}\right) . \tag{4.9}$$

We bound the summation within the Θ as in case 1, but this time we do not obtain a geometric series. Instead, we discover that every term of the summation is the same:

$$\begin{aligned}
\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} &= n^{\log_b a} \sum_{j=0}^{\log_b n-1} \left(\frac{a}{b^{\log_b a}}\right)^j \\
&= n^{\log_b a} \sum_{j=0}^{\log_b n-1} 1 \\
&= n^{\log_b a} \log_b n .
\end{aligned}$$

Substituting this expression for the summation in equation (4.9) yields

$$\begin{aligned}
g(n) &= (\Theta(n^{\log_b a} \log_b n) \\
&= (\Theta(n^{\log_b a} \lg n),
\end{aligned}$$

and case 2 is proved.

Case 3 is proved similarly. Since $f(n)$ appears in the definition (4.7) of $g(n)$ and all terms of $g(n)$ are nonnegative, we can conclude that $g(n) = \Omega(f(n))$ for exact powers of b. Under our assumption that $af(n/b) \le cf(n)$ for some constant $c < 1$ and all $n \ge b$, we have $f(n/b) \le (c/a)f(n)$. Iterating j times, we have $f(n/b^j) \le (c/a)^j f(n)$ or, equivalently, $a^j f(n/b^j) \le c^j f(n)$. Substituting into equation (4.7) and simplifying yields a geometric series, but unlike the series in case 1, this one has decreasing terms:

$$\begin{aligned}
g(n) &= \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \\
&\le \sum_{j=0}^{\log_b n-1} c^j f(n) \\
&\le f(n) \sum_{j=0}^{\infty} c^j \\
&= f(n) \left(\frac{1}{1-c}\right) \\
&= O(f(n)) ,
\end{aligned}$$

since c is constant. Thus, we can conclude that $g(n) = \Theta(f(n))$ for exact powers of b. Case 3 is proved, which completes the proof of the lemma.

We can now prove a version of the master theorem for the case in which n is an exact power of b.

Lemma 4.4

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. Define $T(n)$ on exact powers of b by the recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ aT(n/b) + f(n) & \text{if } n = b^i, \end{cases}$$

where i is a positive integer. Then $T(n)$ can be bounded asymptotically for exact powers of b as follows.

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Proof We use the bounds in Lemma 4.3 to evaluate the summation (4.6) from Lemma 4.2. For case 1, we have

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + O(n^{\log_b a}) \\ &= \Theta(n^{\log_b a}), \end{aligned}$$

and for case 2,

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \lg n) \\ &= \Theta(n^{\log_b a} \lg n). \end{aligned}$$

For case 3,

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(f(n)) \\ &= \Theta(f(n)), \end{aligned}$$

because $f(n) = \Omega(n^{\log_b a + \epsilon})$.

4.4.2 Floors and ceilings

To complete the proof of the master theorem, we must now extend our analysis to the situation in which floors and ceilings are used in the master recurrence, so that the recurrence is defined for all integers, not just exact powers of b. Obtaining a lower bound on

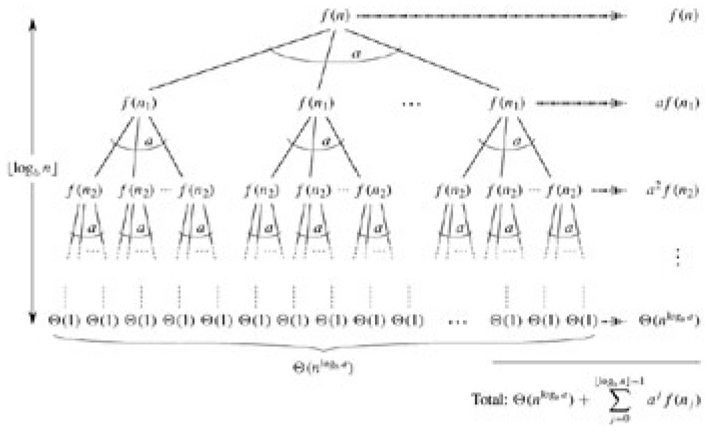
(4.10) $T(n) = aT(\lceil n/b \rceil) + f(n)$

and an upper bound on

(4.11) $T(n) = aT(\lfloor n/b \rfloor) + f(n)$

is routine, since the bound $\lceil n/b \rceil \ge n/b$ can be pushed through in the first case to yield the desired result, and the bound $\lfloor n/b \rfloor \le n/b$ can be pushed through in the second case. Lower bounding the recurrence (4.11) requires much the same technique as upper bounding the recurrence (4.10), so we shall present only this latter bound.

We modify the recursion tree of Figure 4.3 to produce the recursion tree in Figure 4.4. As we go down in the recursion tree, we obtain a sequence of recursive invocations on the arguments



$f(n)$
$f(n)$
a
$f(n_1)$
$af(n_1)$
$\lfloor \log_b n \rfloor$
$f(n_2)$
$a^2 f(n_2)$
$\Theta(1)$
$\Theta(n^{\log_b a})$
$\Theta(n^{\log_b a})$
Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$

Figure 4.4: The recursion tree generated by $T(n) = aT(\lceil n/b \rceil) + f(n)$. The recursive argument n_j is given by equation (4.12).

n ,
$\lceil n/b \rceil$,
$\lceil \lceil n/b \rceil /b \rceil$,
$\lceil \lceil \lceil n/b \rceil /b \rceil /b \rceil$,
$\vdots$

Let us denote the jth element in the sequence by n_j, where

(4.12) $n_j = \begin{cases} n & \text{if } j = 0\,, \\ \lceil n_{j-1}/b \rceil & \text{if } j > 0\,. \end{cases}$

Our first goal is to determine the depth k such that n_k is a constant. Using the inequality $\lceil x \rceil \le x + 1$, we obtain

$$\begin{aligned} n_0 &\le n\,, \\ n_1 &\le \frac{n}{b} + 1\,, \\ n_2 &\le \frac{n}{b^2} + \frac{1}{b} + 1\,, \\ n_3 &\le \frac{n}{b^3} + \frac{1}{b^2} + \frac{1}{b} + 1\,, \\ &\vdots \end{aligned}$$

In general,

$$\begin{aligned} n_j &\le \frac{n}{b^j} + \sum_{i=0}^{j-1} \frac{1}{b^i} \\ &< \frac{n}{b^j} + \sum_{i=0}^{\infty} \frac{1}{b^i} \\ &= \frac{n}{b^j} + \frac{b}{b-1}\,. \end{aligned}$$

Letting $j = \lfloor \log_b n \rfloor$, we obtain

$$\begin{aligned} n_{\lfloor \log_b n \rfloor} &< \frac{n}{b^{\lfloor \log_b n \rfloor}} + \frac{b}{b-1} \\ &\le \frac{n}{b^{\log_b n - 1}} + \frac{b}{b-1} \\ &= \frac{n}{n/b} + \frac{b}{b-1} \\ &= b + \frac{b}{b-1} \\ &= O(1)\,, \end{aligned}$$

and thus we see that at depth $\lfloor \log_b n \rfloor$, the problem size is at most a constant.

From Figure 4.4, we see that

$$(4.13) \quad T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)\,,$$

which is much the same as equation (4.6), except that n is an arbitrary integer and not restricted to be an exact power of b.

We can now evaluate the summation

$$(4.14) \quad g(n) = \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$$

from (4.13) in a manner analogous to the proof of Lemma 4.3. Beginning with case 3, if $af(\lceil n/b \rceil) \le cf(n)$ for $n > b + b/(b - 1)$, where $c < 1$ is a constant, then it follows that $a^j f(n_j) \le c^j f(n)$. Therefore, the sum in equation (4.14) can be evaluated just as in Lemma 4.3. For case 2, we have $f(n) = \Theta(n^{\log_b a})$. If we can show that $f(n_j) = O(n^{\log_b a}/a^j) = O((n/b^j)^{\log_b a})$, then the proof for case 2 of Lemma 4.3 will go through. Observe that $j = \leq \lfloor \log_b n \rfloor$ implies $b^j/n \leq 1$. The bound $f(n) = O(n^{\log_b a})$ implies that there exists a constant $c > 0$ such that for all sufficiently large n_j,

$$\begin{aligned}
f(n_j) &\leq c\left(\frac{n}{b^j} + \frac{b}{b-1}\right)^{\log_b a} \\
&= c\left(\frac{n}{b^j}\left(1 + \frac{b^j}{n} \cdot \frac{b}{b-1}\right)\right)^{\log_b a} \\
&= c\left(\frac{n^{\log_b a}}{a^j}\right)\left(1 + \left(\frac{b^j}{n} \cdot \frac{b}{b-1}\right)\right)^{\log_b a} \\
&\leq c\left(\frac{n^{\log_b a}}{a^j}\right)\left(1 + \frac{b}{b-1}\right)^{\log_b a} \\
&= O\left(\frac{n^{\log_b a}}{a^j}\right),
\end{aligned}$$

since $c(1 + b/(b-1))^{\log_b a}$ is a constant. Thus, case 2 is proved. The proof of case 1 is almost identical. The key is to prove the bound $f(n_j) = O(n^{\log_b a - \epsilon})$, which is similar to the corresponding proof of case 2, though the algebra is more intricate.

We have now proved the upper bounds in the master theorem for all integers n. The proof of the lower bounds is similar.

Exercises 4.4-1: ★

Give a simple and exact expression for n_j in equation (4.12) for the case in which b is a positive integer instead of an arbitrary real number.

Exercises 4.4-2: ★

Show that if $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$, then the master recurrence has solution $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. For simplicity, confine your analysis to exact powers of b.

Exercises 4.4-3: ★

Show that case 3 of the master theorem is overstated, in the sense that the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ implies that there exists a constant $\in > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$.

Problems 4-1: Recurrence examples

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences.