Chapter 4
Divide-and-Conquer

Recall that in divide-and-conquer, we solve a problem recursively, by applying three steps at each level of the recursion:
Divide the problem into a number of subproblems that are smaller instances of the same problem.
Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
Combine the solution to the subproblems into the solution for the original problem.

Recurrences:

A **recurrence** is a function which is defined in terms of

1. one or more base cases, and
2. itself, with smaller arguments.

Examples:

1. $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + 1 & \text{if } n > 1 \end{cases}$

   The solution for the above recurrence is given by: $T(n) = \Theta(n)$

2. $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(\frac{n}{2}) + n & \text{if } n > 1 \end{cases}$

   The solution for the above recurrence is given by: $T(n) = \Theta(n \lg n)$

3. $T(n) = \begin{cases} 0 & \text{if } n = 2 \\ T(\sqrt{n}) + 1 & \text{if } n > 2 \end{cases}$

   The solution for the above recurrence is given by: $T(n) = \Theta(\lg \lg n)$

4. $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(\frac{n}{3}) + T(\frac{2n}{3}) + n & \text{if } n > 1 \end{cases}$

   The solution for the above recurrence is given by: $T(n) = \Theta(n \lg n)$

   When dealing with recurrence functions, floors and ceiling can be removed without affecting the solution to the recurrence. There are three methods used to solve recurrence functions:

   (a) Substitution method

   (b) Recursion-tree method

(c) Master method

Substitution Method: The substitution method for solving recurrences entails two steps:

(a) Guess the solution

(b) Use induction to find the constants and show that the solution works.

Example: Let $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(\frac{n}{2}) + n & \text{if } n > 1 \end{cases}$

(a) Guess: $T(n) = n \lg n + n$

(b) Induction:
Basis: $n = 1 \Rightarrow n \lg n + n = 1 \lg 1 + 1 = 1 = T(n)$
Inductive step: Inductive hypothesis is that $T(k) = k \lg k + k$ for all $k < n$.
We will use this inductive hypothesis for $T(\frac{n}{2})$.

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + n \\ &= 2(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}) + n \\ &= n \lg \frac{n}{2} + n + n \\ &= n(\lg n - \lg 2) + n + n \\ &= n \lg n - n + n + n \\ &= n \lg n + n. \end{aligned}$$

Generally, the asymptotic notation is used when dealing with recurrence relations. For example, we could write $T(n) = 2T(\frac{n}{2}) + \Theta(n)$ for the above problem. Also, we assume $T(n) = \Theta(1)$ for sufficiently small n. In this case the solution for the above problem could be expressed as $T(n) = \Theta(n \lg n)$. Note that since we are interested only in the asymptotic solution to a recurrence, we don't worry about the base case in our proofs. We only worry about the base case if we are interested in the exact solution. One way to find the asymptotic solution is to find the upper (O) and lower (Ω) bounds separately. Let us see an example how this works
Example: Consider $T(n) = 2T(\frac{n}{2}) + \Theta(n)$.

1. **Upper bound:** If we want to show an upper bound of $T(n) = 2T(\frac{n}{2}) + \Theta(n)$, we write $T(n) \leq 2T(\frac{n}{2}) + cn$ for some positive constant c.
   Guess: $T(n) \leq dn \lg n$ for some positive constant d. We are given c in the recurrence, and we get to choose d as any positive constant. It is OK for d to depend

on c.
Substitution:

$$\begin{aligned} T(n) &\leq 2T(\frac{n}{2}) + cn \\ &= 2(d\frac{n}{2}\lg\frac{n}{2}) + cn \\ &= dn\lg\frac{n}{2} + cn \\ &= dn\lg n - dn + cn \\ &\leq dn\lg n \quad if - dn + cn \leq 0, d \geq c \end{aligned}$$

Therefore, $T(n) = O(n\lg n)$

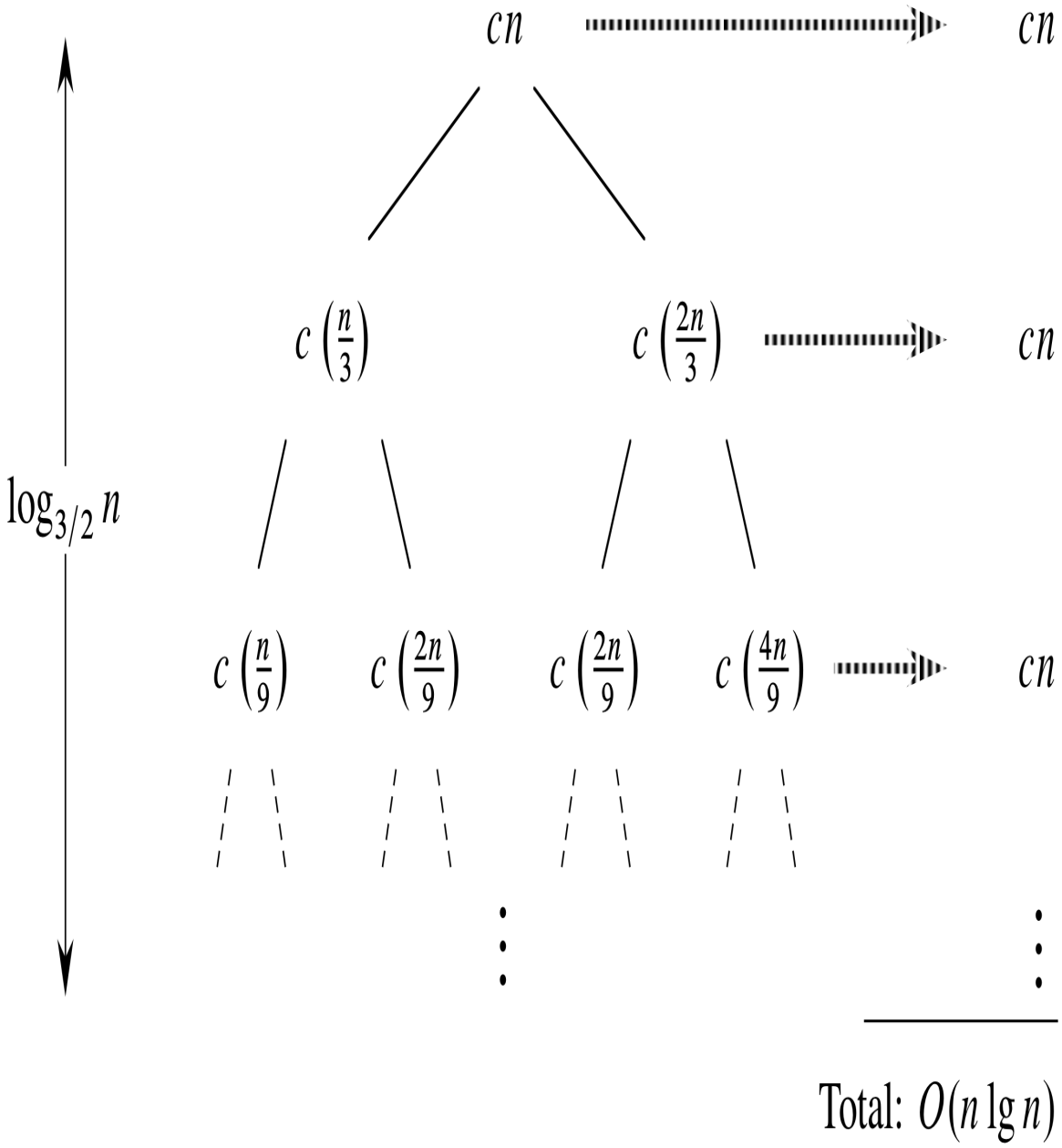
2. **Lower bound:** If we want to show a lower bound of $T(n) = 2T(\frac{n}{2}) + \Theta(n)$, we write $T(n) \geq 2T(\frac{n}{2}) + cn$ for some positive constant c.
Guess: $T(n) \geq dn\lg n$ for some positive constant d.
Substitution:

$$\begin{aligned} T(n) &\geq 2T(\frac{n}{2}) + cn \\ &= 2(d\frac{n}{2}\lg\frac{n}{2}) + cn \\ &= dn\lg\frac{n}{2} + cn \\ &= dn\lg n - dn + cn \\ &\geq dn\lg n \quad if - dn + cn \geq 0, d \leq c \end{aligned}$$

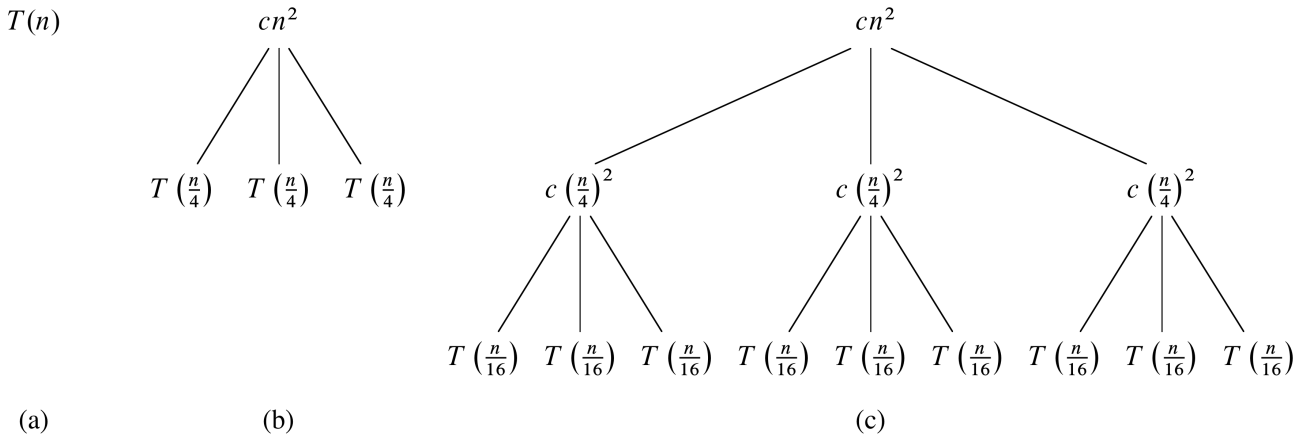
Therefore, $T(n) = \Omega(n\lg n)$

Hence, $T(n) = \Theta(n\lg n)$
Recursion Trees: are used to generate a guess. Then verify by the substitution method. In a recursion tree, each node represents the cost of a single subproblem somewhere in the set of recursive functions invocations. We sum the costs within each level of the tree to obtain a set of per-level costs, and then we sum all the per-level costs to determine the total cost of all levels of the recursion.

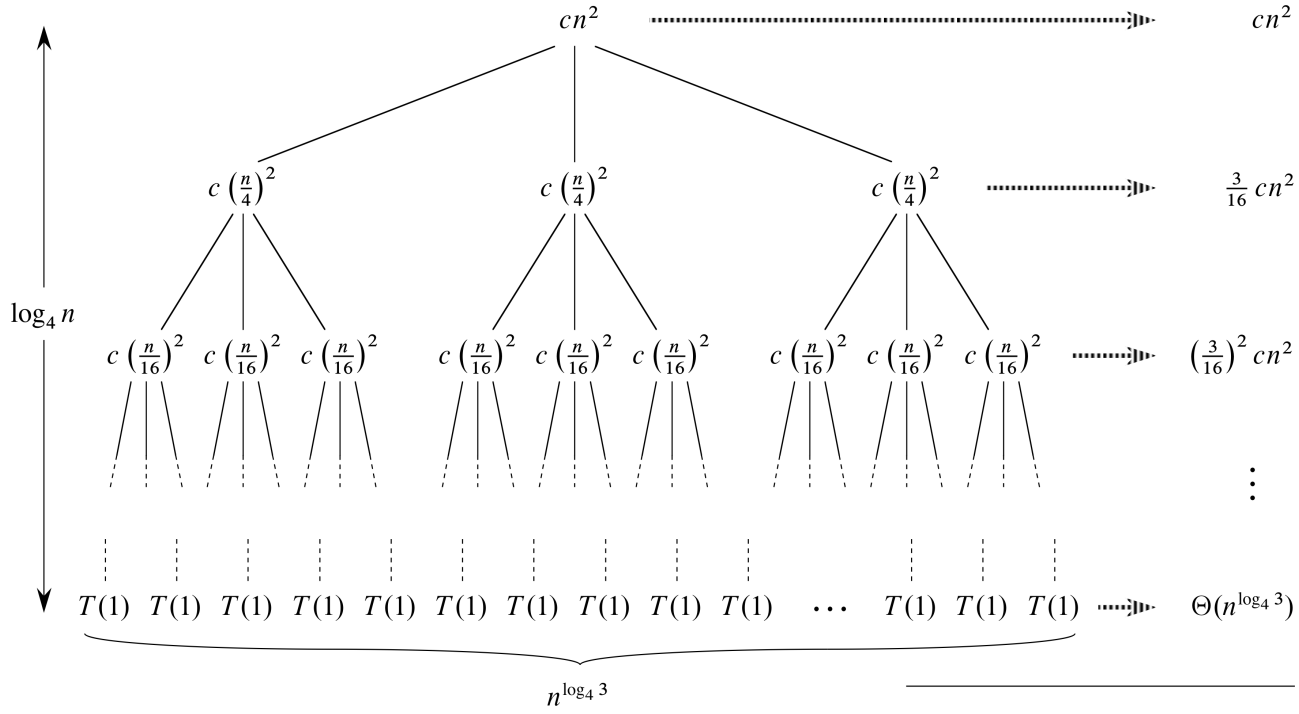
Example: Consider the recurrence $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + O(n)$. The recursion tree is given below:



Example: Consider the recurrence $T(n) = 3T(\frac{n}{4}) + \Theta(n^2)$. The recursion tree is given below:



(a) (b) (c)



(d)

Total: $O(n^2)$

Master Method: The master method is used for many divide and conquer recurrences of the form: $T(n) = aT(\frac{n}{b}) + f(n)$, where $a \geq 1, b > 1$ are constants and $f(n) > 0$.
Master Theorem (Theorem 4.1 page 94)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence
$T(n) = aT(\frac{n}{b}) + f(n)$
where we interpret $\frac{n}{b}$ to mean either $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then the solution is given by $T(n) = \Theta(n^{\lg_b a})$.

2. If $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$, then the solution is given by $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and $f(n)$ satisfies the regularity condition $af(\frac{n}{b}) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then the solution is given by $T(n) = \Theta(f(n))$

Notice in the above theorem, we compare $n^{\log_b a}$ vs. $f(n)$ and we notice the following:

1. In case one, $f(n)$ is polynomially smaller than $n^{\lg_b a}$ and the cost is dominated by the leaves.

2. In case two, $f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller and the cost is $n^{\log_b a} \lg^k n$ at each level, and there are $\Theta(\lg n)$ levels. As a simple case: Let $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$

3. In case three, $f(n)$ is polynomially greater than $n^{\log_b a}$ and the cost is dominated by root.

Examples:

1. $T(n) = 9T(\frac{n}{3}) + n$. For this recurrence, we have $a = 9, b = 3, f(n) = n$, and thus we have that $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$. Since $f(n) = O(n^{\log_3 9 - \epsilon})$, where $\epsilon = 1$, se can apply case 1 of the master theorem and conclude that the solution is $T(n) = \Theta(n^2)$

2. $T(n) = T(\frac{2n}{3}) + 1$. For this recurrence, we have $a = 1, b = \frac{3}{2}, f(n) = 1$, and $n^{\log_{\frac{3}{2}} 1} = n^0 = 1$. Case 2 applies, since $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$, and thus the solution to the recurrence is $T(n) = \Theta(\lg n)$.

3. $T(n) = 3T(\frac{n}{4}) + n \lg n$. For this recurrence, we have $a = 3, b = 4, f(n) = n \lg n$, and $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$. Since $f(n) = \Omega(n^{\log_4 3 + \epsilon})$, where $\epsilon \approx 0.2$, case 3 applies

if we can show that the regularity condition holds for $f(n)$. For sufficiently large n, we have that $af(\frac{n}{b}) = 3(\frac{n}{4})\lg(\frac{n}{4}) \leq (\frac{3}{4})n\lg n = cf(n)$ for $c = \frac{3}{4}$. Consequently, by case 3, the solution to the recurrence is $T(n) = \Theta(n \lg n)$.

4. $T(n) = 2T(n/2) + \Theta(n)$. For this recurrence, we have $a = 2, b = 2, f(n) = \Theta(n)$, and $n^{\log_b a} = n^{\log_2 2} = n$. Case 2 applies, since $f(n) = \Theta(n)$, and hence, the solution is $T(n) = \Theta(n \lg n)$

5. $T(n) = 8T(n/2) + \Theta(n^2)$. For this recurrence, we have $a = 8, b = 2, f(n) = \Theta(n^2)$, and $n^{\log_b a} = n^{\log_2 8} = n^3$. Since n^3 is polynomially larger than $f(n)(f(n) = O(n^{3-\epsilon})$ for $\epsilon = 1$), case 1 applies, and $T(n) = \Theta(n^3)$

6. $T(n) = 7T(n/2) + \Theta(n^2)$. For this recurrence, we have $a = 7, b = 2, f(n) = \Theta(n^2)$, and $n^{\log_b a} = n^{\log_2 7}$. But, $2.80 < \log_2 7 < 2.81$, we see that $f(n) = O(n^{\lg 7-\epsilon})$ for $\epsilon = 0.8$ and therefore, case 1 applies. Hence, the solution is $T(n) = \Theta(n^{\lg 7})$