
Lecture Notes for Chapter 4:

Recurrences

Chapter 4 overview

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

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

Examples:

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

Solution: $T(n) = n$.

- $$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n & \text{if } n \geq 2. \end{cases}$$

Solution: $T(n) = n \lg n + n$.

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

Solution: $T(n) = \lg \lg n$.

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

Solution: $T(n) = \Theta(n \lg n)$.

[The notes for this chapter are fairly brief because we teach recurrences in much greater detail in a separate discrete math course.]

Many technical issues:

- Floors and ceilings

[Floors and ceilings can easily be removed and don't affect the solution to the recurrence. They are better left to a discrete math course.]

- Exact vs. asymptotic functions
- Boundary conditions

In algorithm analysis, we usually express both the recurrence and its solution using asymptotic notation.

- Example: $T(n) = 2T(n/2) + \Theta(n)$, with solution $T(n) = \Theta(n \lg n)$.
- The boundary conditions are usually expressed as “ $T(n) = O(1)$ for sufficiently small n .”
- When we desire an exact, rather than an asymptotic, solution, we need to deal with boundary conditions.
- In practice, we just use asymptotics most of the time, and we ignore boundary conditions.

[In my course, there are only two acceptable ways of solving recurrences: the substitution method and the master method. Unless the recursion tree is carefully accounted for, I do not accept it as a proof of a solution, though I certainly accept a recursion tree as a way to generate a guess for substitution method. You may choose to allow recursion trees as proofs in your course, in which case some of the substitution proofs in the solutions for this chapter become recursion trees.

I also never use the iteration method, which had appeared in the first edition of Introduction to Algorithms. I find that it is too easy to make an error in parenthesization, and that recursion trees give a better intuitive idea than iterating the recurrence of how the recurrence progresses.]

Substitution method

1. Guess the solution.
2. Use induction to find the constants and show that the solution works.

Example:

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

1. *Guess:* $T(n) = n \lg n + n$. [Here, we have a recurrence with an exact function, rather than asymptotic notation, and the solution is also exact rather than asymptotic. We'll have to check boundary conditions and the base case.]
2. *Induction:*

Basis: $n = 1 \Rightarrow n \lg n + n = 1 = T(n)$

Inductive step: Inductive hypothesis is that $T(k) = k \lg k + k$ for all $k < n$. We'll use this inductive hypothesis for $T(n/2)$.

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \\ &= 2\left(\frac{n}{2} \lg \frac{n}{2} + \frac{n}{2}\right) + n \quad (\text{by inductive hypothesis}) \\ &= n \lg \frac{n}{2} + n + n \\ &= n(\lg n - \lg 2) + n + n \\ &= n \lg n - n + n + n \\ &= n \lg n + n. \end{aligned}$$

■

Generally, we use asymptotic notation:

- We would write $T(n) = 2T(n/2) + \Theta(n)$.
- We assume $T(n) = O(1)$ for sufficiently small n .
- We express the solution by asymptotic notation: $T(n) = \Theta(n \lg n)$.
- We don't worry about boundary cases, nor do we show base cases in the substitution proof.
 - $T(n)$ is always constant for any constant n .
 - Since we are ultimately interested in an asymptotic solution to a recurrence, it will always be possible to choose base cases that work.
 - When we want an asymptotic solution to a recurrence, we don't worry about the base cases in our proofs.
 - When we want an exact solution, then we have to deal with base cases.

For the substitution method:

- Name the constant in the additive term.
- Show the upper (O) and lower (Ω) bounds separately. Might need to use different constants for each.

Example: $T(n) = 2T(n/2) + \Theta(n)$. If we want to show an upper bound of $T(n) = 2T(n/2) + O(n)$, we write $T(n) \leq 2T(n/2) + cn$ for some positive constant c .

1. **Upper bound:**

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's OK for d to depend on c .

Substitution:

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

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

2. **Lower bound:** Write $T(n) \geq 2T(n/2) + cn$ for some positive constant c .

Guess: $T(n) \geq dn \lg n$ for some positive constant d .

Substitution:

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

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

Therefore, $T(n) = \Theta(n \lg n)$. [For this particular recurrence, we can use $d = c$ for both the upper-bound and lower-bound proofs. That won't always be the case.] ■

Make sure you show the same *exact* form when doing a substitution proof.

Consider the recurrence

$$T(n) = 8T(n/2) + \Theta(n^2).$$

For an upper bound:

$$T(n) \leq 8T(n/2) + cn^2.$$

Guess: $T(n) \leq dn^3$.

$$\begin{aligned} T(n) &\leq 8d(n/2)^3 + cn^2 \\ &= 8d(n^3/8) + cn^2 \\ &= dn^3 + cn^2 \\ &\not\leq dn^3 \quad \text{doesn't work!} \end{aligned}$$

Remedy: Subtract off a lower-order term.

Guess: $T(n) \leq dn^3 - d'n^2$.

$$\begin{aligned} T(n) &\leq 8(d(n/2)^3 - d'(n/2)^2) + cn^2 \\ &= 8d(n^3/8) - 8d'(n^2/4) + cn^2 \\ &= dn^3 - 2d'n^2 + cn^2 \\ &= dn^3 - d'n^2 - d'n^2 + cn^2 \\ &\leq dn^3 - d'n^2 \quad \text{if } -d'n^2 + cn^2 \leq 0, \\ &\quad \quad \quad d' \geq c \end{aligned}$$

Be careful when using asymptotic notation.

The false proof for the recurrence $T(n) = 4T(n/4) + n$, that $T(n) = O(n)$:

$$\begin{aligned} T(n) &\leq 4(c(n/4)) + n \\ &\leq cn + n \\ &= O(n) \quad \text{wrong!} \end{aligned}$$

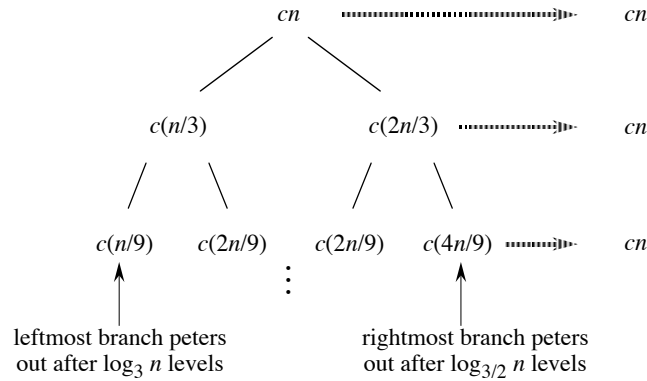
Because we haven't proven the *exact form* of our inductive hypothesis (which is that $T(n) \leq cn$), this proof is false.

Recursion trees

Use to generate a guess. Then verify by substitution method.

Example: $T(n) = T(n/3) + T(2n/3) + \Theta(n)$. For upper bound, rewrite as $T(n) \leq T(n/3) + T(2n/3) + cn$; for lower bound, as $T(n) \geq T(n/3) + T(2n/3) + cn$.

By summing across each level, the recursion tree shows the cost at each level of recursion (minus the costs of recursive calls, which appear in subtrees):



- There are $\log_3 n$ full levels, and after $\log_{3/2} n$ levels, the problem size is down to 1.
- Each level contributes $\leq cn$.
- Lower bound guess: $\geq dn \log_3 n = \Omega(n \lg n)$ for some positive constant d .
- Upper bound guess: $\leq dn \log_{3/2} n = O(n \lg n)$ for some positive constant d .
- Then *prove* by substitution.

1. **Upper bound:**

Guess: $T(n) \leq dn \lg n$.

Substitution:

$$\begin{aligned}
 T(n) &\leq T(n/3) + T(2n/3) + cn \\
 &\leq d(n/3) \lg(n/3) + d(2n/3) \lg(2n/3) + cn \\
 &= (d(n/3) \lg n - d(n/3) \lg 3) \\
 &\quad + (d(2n/3) \lg n - d(2n/3) \lg(3/2)) + cn \\
 &= dn \lg n - d((n/3) \lg 3 + (2n/3) \lg(3/2)) + cn \\
 &= dn \lg n - d((n/3) \lg 3 + (2n/3) \lg 3 - (2n/3) \lg 2) + cn \\
 &= dn \lg n - dn(\lg 3 - 2/3) + cn \\
 &\leq dn \lg n \quad \text{if } -dn(\lg 3 - 2/3) + cn \leq 0, \\
 &\quad \quad \quad d \geq \frac{c}{\lg 3 - 2/3}.
 \end{aligned}$$

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

Note: Make sure that the symbolic constants used in the recurrence (e.g., c) and the guess (e.g., d) are different.

2. **Lower bound:**

Guess: $T(n) \geq dn \lg n$.

Substitution: Same as for the upper bound, but replacing \leq by \geq . End up needing

$$0 < d \leq \frac{c}{\lg 3 - 2/3}.$$

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

Since $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$. ■

Master method

Used for many divide-and-conquer recurrences of the form

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

where $a \geq 1$, $b > 1$, and $f(n) > 0$.

Based on the **master theorem** (Theorem 4.1).

Compare $n^{\log_b a}$ vs. $f(n)$:

Case 1: $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$.

($f(n)$ is polynomially smaller than $n^{\log_b a}$.)

Solution: $T(n) = \Theta(n^{\log_b a})$.

(Intuitively: cost is dominated by leaves.)

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$.

[This formulation of Case 2 is more general than in Theorem 4.1, and it is given in Exercise 4.4-2.]

($f(n)$ is within a polylog factor of $n^{\log_b a}$, but not smaller.)

Solution: $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

(Intuitively: cost is $n^{\log_b a} \lg^k n$ at each level, and there are $\Theta(\lg n)$ levels.)

Simple case: $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$.

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $f(n)$ satisfies the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n .

($f(n)$ is polynomially greater than $n^{\log_b a}$.)

Solution: $T(n) = \Theta(f(n))$.

(Intuitively: cost is dominated by root.)

What's with the Case 3 regularity condition?

- Generally not a problem.
- It always holds whenever $f(n) = n^k$ and $f(n) = \Omega(n^{\log_b a + \epsilon})$ for constant $\epsilon > 0$. [Proving this makes a nice homework exercise. See below.] So you don't need to check it when $f(n)$ is a polynomial.

[Here's a proof that the regularity condition holds when $f(n) = n^k$ and $f(n) = \Omega(n^{\log_b a + \epsilon})$ for constant $\epsilon > 0$.

Since $f(n) = \Omega(n^{\log_b a + \epsilon})$ and $f(n) = n^k$, we have that $k > \log_b a$. Using a base of b and treating both sides as exponents, we have $b^k > b^{\log_b a} = a$, and so $a/b^k < 1$. Since a , b , and k are constants, if we let $c = a/b^k$, then c is a constant strictly less than 1. We have that $af(n/b) = a(n/b)^k = (a/b^k)n^k = cf(n)$, and so the regularity condition is satisfied.]

Examples:

- $T(n) = 5T(n/2) + \Theta(n^2)$
 $n^{\log_2 5}$ vs. n^2
 Since $\log_2 5 - \epsilon = 2$ for some constant $\epsilon > 0$, use Case 1 $\Rightarrow T(n) = \Theta(n^{\lg 5})$

- $T(n) = 27T(n/3) + \Theta(n^3 \lg n)$
 $n^{\lg_3 27} = n^3$ vs. $n^3 \lg n$
 Use Case 2 with $k = 1 \Rightarrow T(n) = \Theta(n^3 \lg^2 n)$
- $T(n) = 5T(n/2) + \Theta(n^3)$
 $n^{\lg_2 5}$ vs. n^3
 Now $\lg 5 + \epsilon = 3$ for some constant $\epsilon > 0$
 Check regularity condition (don't really need to since $f(n)$ is a polynomial):
 $af(n/b) = 5(n/2)^3 = 5n^3/8 \leq cn^3$ for $c = 5/8 < 1$
 Use Case 3 $\Rightarrow T(n) = \Theta(n^3)$
- $T(n) = 27T(n/3) + \Theta(n^3 / \lg n)$
 $n^{\lg_3 27} = n^3$ vs. $n^3 / \lg n = n^3 \lg^{-1} n \neq \Theta(n^3 \lg^k n)$ for any $k \geq 0$.
 Cannot use the master method.

[We don't prove the master theorem in our algorithms course. We sometimes prove a simplified version for recurrences of the form $T(n) = aT(n/b) + rf$. Section 4.4 of the text has the full proof of the master theorem.]

Solutions for Chapter 4: Recurrences

Solution to Exercise 4.2-2

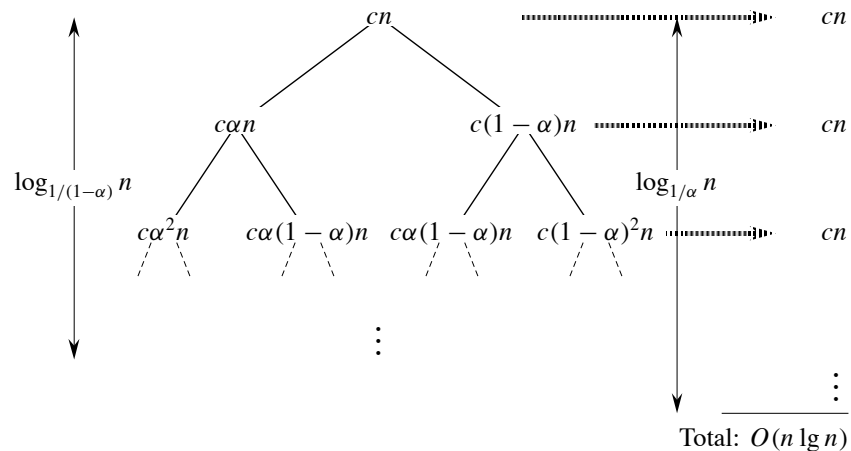
The shortest path from the root to a leaf in the recursion tree is $n \rightarrow (1/3)n \rightarrow (1/3)^2 n \rightarrow \dots \rightarrow 1$. Since $(1/3)^k n = 1$ when $k = \log_3 n$, the height of the part of the tree in which every node has two children is $\log_3 n$. Since the values at each of these levels of the tree add up to n , the solution to the recurrence is at least $n \log_3 n = \Omega(n \lg n)$.

Solution to Exercise 4.2-5

$$T(n) = T(\alpha n) + T((1 - \alpha)n) + n$$

We saw the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$ in the text. This recurrence can be similarly solved.

Without loss of generality, let $\alpha \geq 1 - \alpha$, so that $0 < 1 - \alpha \leq 1/2$ and $1/2 \leq \alpha < 1$.



The recursion tree is full for $\log_{1/(1-\alpha)} n$ levels, each contributing cn , so we guess $\Omega(n \log_{1/(1-\alpha)} n) = \Omega(n \lg n)$. It has $\log_{1/\alpha} n$ levels, each contributing $\leq cn$, so we guess $O(n \log_{1/\alpha} n) = O(n \lg n)$.

Now we show that $T(n) = \Theta(n \lg n)$ by substitution. To prove the upper bound, we need to show that $T(n) \leq dn \lg n$ for a suitable constant $d > 0$.

$$\begin{aligned}
 T(n) &= T(\alpha n) + T((1 - \alpha)n) + cn \\
 &\leq d\alpha n \lg(\alpha n) + d(1 - \alpha)n \lg((1 - \alpha)n) + cn \\
 &= d\alpha n \lg \alpha + d\alpha n \lg n + d(1 - \alpha)n \lg(1 - \alpha) + d(1 - \alpha)n \lg n + cn \\
 &= dn \lg n + dn(\alpha \lg \alpha + (1 - \alpha) \lg(1 - \alpha)) + cn \\
 &\leq dn \lg n,
 \end{aligned}$$

if $dn(\alpha \lg \alpha + (1 - \alpha) \lg(1 - \alpha)) + cn \leq 0$. This condition is equivalent to

$$d(\alpha \lg \alpha + (1 - \alpha) \lg(1 - \alpha)) \leq -c.$$

Since $1/2 \leq \alpha < 1$ and $0 < 1 - \alpha \leq 1/2$, we have that $\lg \alpha < 0$ and $\lg(1 - \alpha) < 0$. Thus, $\alpha \lg \alpha + (1 - \alpha) \lg(1 - \alpha) < 0$, so that when we multiply both sides of the inequality by this factor, we need to reverse the inequality:

$$d \geq \frac{-c}{\alpha \lg \alpha + (1 - \alpha) \lg(1 - \alpha)}$$

or

$$d \geq \frac{c}{-\alpha \lg \alpha - (1 - \alpha) \lg(1 - \alpha)}.$$

The fraction on the right-hand side is a positive constant, and so it suffices to pick any value of d that is greater than or equal to this fraction.

To prove the lower bound, we need to show that $T(n) \geq dn \lg n$ for a suitable constant $d > 0$. We can use the same proof as for the upper bound, substituting \geq for \leq , and we get the requirement that

$$0 < d \leq \frac{c}{-\alpha \lg \alpha - (1 - \alpha) \lg(1 - \alpha)}.$$

Therefore, $T(n) = \Theta(n \lg n)$.

Solution to Problem 4-1

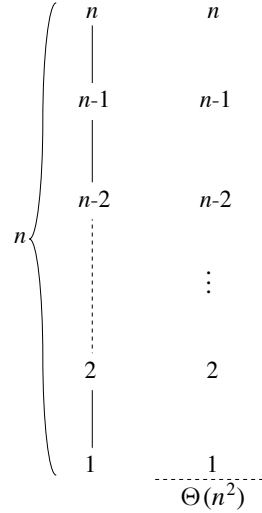
Note: In parts (a), (b), and (d) below, we are applying case 3 of the master theorem, which requires the regularity condition that $af(n/b) \leq cf(n)$ for some constant $c < 1$. In each of these parts, $f(n)$ has the form n^k . The regularity condition is satisfied because $af(n/b) = an^k/b^k = (a/b^k)n^k = (a/b^k)f(n)$, and in each of the cases below, a/b^k is a constant strictly less than 1.

a. $T(n) = 2T(n/2) + n^3 = \Theta(n^3)$. This is a divide-and-conquer recurrence with $a = 2$, $b = 2$, $f(n) = n^3$, and $n^{\log_b a} = n^{\log_2 2} = n$. Since $n^3 = \Omega(n^{\log_2 2 + 2})$ and $a/b^k = 2/2^3 = 1/4 < 1$, case 3 of the master theorem applies, and $T(n) = \Theta(n^3)$.

b. $T(n) = T(9n/10) + n = \Theta(n)$. This is a divide-and-conquer recurrence with $a = 1$, $b = 10/9$, $f(n) = n$, and $n^{\log_b a} = n^{\log_{10/9} 1} = n^0 = 1$. Since $n = \Omega(n^{\log_{10/9} 1 + 1})$ and $a/b^k = 1/(10/9)^1 = 9/10 < 1$, case 3 of the master theorem applies, and $T(n) = \Theta(n)$.

- c. $T(n) = 16T(n/4) + n^2 = \Theta(n^2 \lg n)$. This is another divide-and-conquer recurrence with $a = 16$, $b = 4$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_4 16} = n^2$. Since $n^2 = \Theta(n^{\log_4 16})$, case 2 of the master theorem applies, and $T(n) = \Theta(n^2 \lg n)$.
- d. $T(n) = 7T(n/3) + n^2 = \Theta(n^2)$. This is a divide-and-conquer recurrence with $a = 7$, $b = 3$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_3 7}$. Since $1 < \log_3 7 < 2$, we have that $n^2 = \Omega(n^{\log_3 7 + \epsilon})$ for some constant $\epsilon > 0$. We also have $a/b^k = 7/3^2 = 7/9 < 1$, so that case 3 of the master theorem applies, and $T(n) = \Theta(n^2)$.
- e. $T(n) = 7T(n/2) + n^2 = O(n^{\lg 7})$. This is a divide-and-conquer recurrence with $a = 7$, $b = 2$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_2 7}$. Since $2 < \lg 7 < 3$, we have that $n^2 = O(n^{\log_2 7 - \epsilon})$ for some constant $\epsilon > 0$. Thus, case 1 of the master theorem applies, and $T(n) = \Theta(n^{\lg 7})$.
- f. $T(n) = 2T(n/4) + \sqrt{n} = \Theta(\sqrt{n} \lg n)$. This is another divide-and-conquer recurrence with $a = 2$, $b = 4$, $f(n) = \sqrt{n}$, and $n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$. Since $\sqrt{n} = \Theta(n^{\log_4 2})$, case 2 of the master theorem applies, and $T(n) = \Theta(\sqrt{n} \lg n)$.
- g. $T(n) = T(n-1) + n$

Using the recursion tree shown below, we get a guess of $T(n) = \Theta(n^2)$.



First, we prove the $T(n) = \Omega(n^2)$ part by induction. The inductive hypothesis is $T(n) \geq cn^2$ for some constant $c > 0$.

$$\begin{aligned}
 T(n) &= T(n-1) + n \\
 &\geq c(n-1)^2 + n \\
 &= cn^2 - 2cn + c + n \\
 &\geq cn^2
 \end{aligned}$$

if $-2cn + n + c \geq 0$ or, equivalently, $n(1-2c) + c \geq 0$. This condition holds when $n \geq 0$ and $0 < c \leq 1/2$.

For the upper bound, $T(n) = O(n^2)$, we use the inductive hypothesis that $T(n) \leq cn^2$ for some constant $c > 0$. By a similar derivation, we get that

$T(n) \leq cn^2$ if $-2cn + n + c \leq 0$ or, equivalently, $n(1 - 2c) + c \leq 0$. This condition holds for $c = 1$ and $n \geq 1$.

Thus, $T(n) = \Omega(n^2)$ and $T(n) = O(n^2)$, so we conclude that $T(n) = \Theta(n^2)$.

h. $T(n) = T(\sqrt{n}) + 1$

The easy way to do this is with a change of variables, as on page 66 of the text. Let $m = \lg n$ and $S(m) = T(2^m)$. $T(2^m) = T(2^{m/2}) + 1$, so $S(m) = S(m/2) + 1$. Using the master theorem, $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ and $f(n) = 1$. Since $1 = \Theta(1)$, case 2 applies and $S(m) = \Theta(\lg m)$. Therefore, $T(n) = \Theta(\lg \lg n)$.

Solution to Problem 4-4

[This problem is solved only for parts a, c, e, f, g, h, and i.]

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

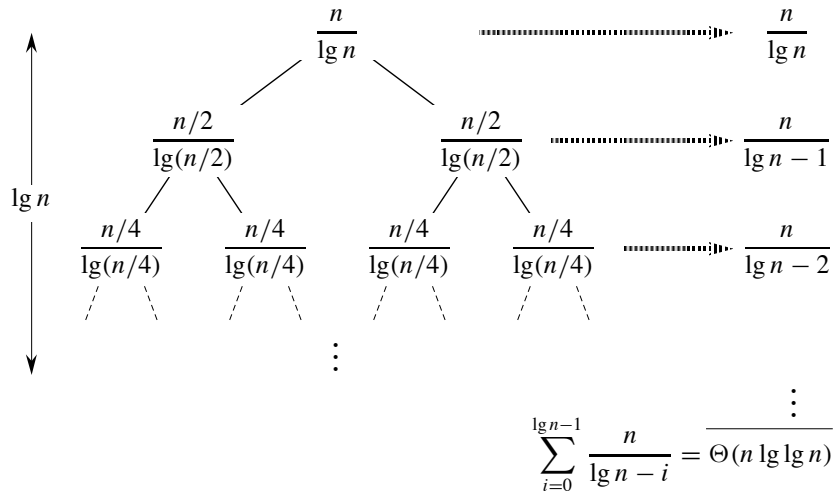
We have $f(n) = n \lg n$ and $n^{\log_b a} = n^{\lg 3} \approx n^{1.585}$. Since $n \lg n = O(n^{\lg 3 - \epsilon})$ for any $0 < \epsilon \leq 0.58$, by case 1 of the master theorem, we have $T(n) = \Theta(n^{\lg 3})$.

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

We have $f(n) = n^2 \sqrt{n} = n^{5/2}$ and $n^{\log_b a} = n^{\log_2 4} = n^{\lg 2}$. Since $n^{5/2} = \Omega(n^{\lg 2 + 3/2})$, we look at the regularity condition in case 3 of the master theorem. We have $af(n/b) = 4(n/2)^2 \sqrt{n/2} = n^{5/2}/\sqrt{2} \leq cn^{5/2}$ for $1/\sqrt{2} \leq c < 1$. Case 3 applies, and we have $T(n) = \Theta(n^2 \sqrt{n})$.

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

We can get a guess by means of a recursion tree:



We get the sum on each level by observing that at depth i , we have 2^i nodes, each with a numerator of $n/2^i$ and a denominator of $\lg(n/2^i) = \lg n - i$, so that the cost at depth i is

$$2^i \cdot \frac{n/2^i}{\lg n - i} = \frac{n}{\lg n - i}.$$

The sum for all levels is

$$\begin{aligned} \sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i} &= n \sum_{i=1}^{\lg n} \frac{1}{i} \\ &= n \sum_{i=1}^{\lg n} 1/i \\ &= n \cdot \Theta(\lg \lg n) \quad (\text{by equation (A.7), the harmonic series}) \\ &= \Theta(n \lg \lg n). \end{aligned}$$

We can use this analysis as a guess that $T(n) = \Theta(n \lg \lg n)$. If we were to do a straight substitution proof, it would be rather involved. Instead, we will show by substitution that $T(n) \leq n(1 + H_{\lfloor \lg n \rfloor})$ and $T(n) \geq n \cdot H_{\lceil \lg n \rceil}$, where H_k is the k th harmonic number: $H_k = 1/1 + 1/2 + 1/3 + \dots + 1/k$. We also define $H_0 = 0$. Since $H_k = \Theta(\lg k)$, we have that $H_{\lfloor \lg n \rfloor} = \Theta(\lg \lfloor \lg n \rfloor) = \Theta(\lg \lg n)$ and $H_{\lceil \lg n \rceil} = \Theta(\lg \lceil \lg n \rceil) = \Theta(\lg \lg n)$. Thus, we will have that $T(n) = \Theta(n \lg \lg n)$.

The base case for the proof is for $n = 1$, and we use $T(1) = 1$. Here, $\lg n = 0$, so that $\lg n = \lfloor \lg n \rfloor = \lceil \lg n \rceil$. Since $H_0 = 0$, we have $T(1) = 1 \leq 1(1 + H_0)$ and $T(1) = 1 \geq 0 = 1 \cdot H_0$.

For the upper bound of $T(n) \leq n(1 + H_{\lfloor \lg n \rfloor})$, we have

$$\begin{aligned} T(n) &= 2T(n/2) + n/\lg n \\ &\leq 2((n/2)(1 + H_{\lfloor \lg(n/2) \rfloor})) + n/\lg n \\ &= n(1 + H_{\lfloor \lg n - 1 \rfloor}) + n/\lg n \\ &= n(1 + H_{\lfloor \lg n \rfloor - 1} + 1/\lg n) \\ &\leq n(1 + H_{\lfloor \lg n \rfloor - 1} + 1/\lfloor \lg n \rfloor) \\ &= n(1 + H_{\lfloor \lg n \rfloor}), \end{aligned}$$

where the last line follows from the identity $H_k = H_{k-1} + 1/k$.

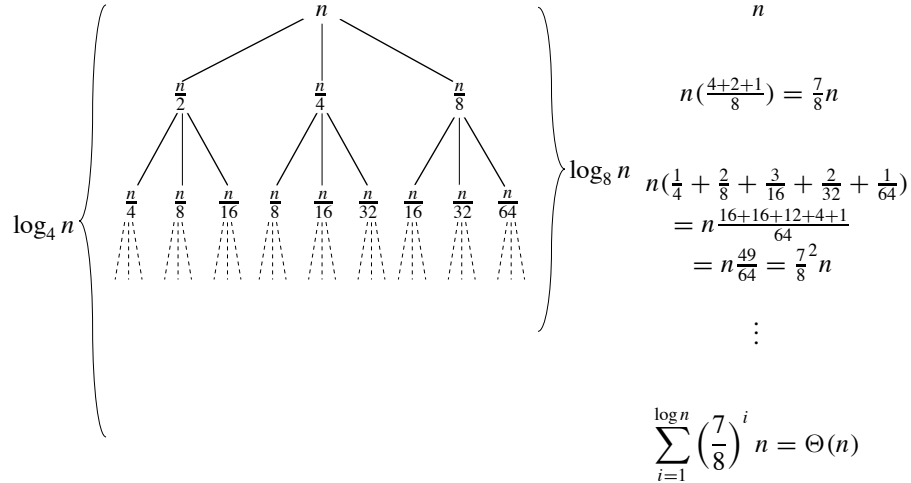
The upper bound of $T(n) \geq n \cdot H_{\lceil \lg n \rceil}$ is similar:

$$\begin{aligned} T(n) &= 2T(n/2) + n/\lg n \\ &\geq 2((n/2) \cdot H_{\lceil \lg(n/2) \rceil}) + n/\lg n \\ &= n \cdot H_{\lceil \lg n - 1 \rceil} + n/\lg n \\ &= n \cdot (H_{\lceil \lg n \rceil - 1} + 1/\lg n) \\ &\geq n \cdot (H_{\lceil \lg n \rceil - 1} + 1/\lceil \lg n \rceil) \\ &= n \cdot H_{\lceil \lg n \rceil}. \end{aligned}$$

Thus, $T(n) = \Theta(n \lg \lg n)$.

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

Using the recursion tree shown below, we get a guess of $T(n) = \Theta(n)$.



$$\begin{aligned}
 & n \\
 & n\left(\frac{4+2+1}{8}\right) = \frac{7}{8}n \\
 & n\left(\frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \frac{2}{32} + \frac{1}{64}\right) \\
 & = n \frac{16+16+12+4+1}{64} \\
 & = n \frac{49}{64} = \frac{7^2}{8}n \\
 & \vdots \\
 & \sum_{i=1}^{\log n} \left(\frac{7}{8}\right)^i n = \Theta(n)
 \end{aligned}$$

We use the substitution method to prove that $T(n) = O(n)$. Our inductive hypothesis is that $T(n) \leq cn$ for some constant $c > 0$. We have

$$\begin{aligned}
 T(n) &= T(n/2) + T(n/4) + T(n/8) + n \\
 &\leq cn/2 + cn/4 + cn/8 + n \\
 &= 7cn/8 + n \\
 &= (1 + 7c/8)n \\
 &\leq cn \quad \text{if } c \geq 8.
 \end{aligned}$$

Therefore, $T(n) = O(n)$.

Showing that $T(n) = \Omega(n)$ is easy:

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

Since $T(n) = O(n)$ and $T(n) = \Omega(n)$, we have that $T(n) = \Theta(n)$.

g. $T(n) = T(n-1) + 1/n$

This recurrence corresponds to the harmonic series, so that $T(n) = H_n$, where $H_n = 1/1 + 1/2 + 1/3 + \cdots + 1/n$. For the base case, we have $T(1) = 1 = H_1$. For the inductive step, we assume that $T(n-1) = H_{n-1}$, and we have

$$\begin{aligned}
 T(n) &= T(n-1) + 1/n \\
 &= H_{n-1} + 1/n \\
 &= H_n.
 \end{aligned}$$

Since $H_n = \Theta(\lg n)$ by equation (A.7), we have that $T(n) = \Theta(\lg n)$.

h. $T(n) = T(n-1) + \lg n$

We guess that $T(n) = \Theta(n \lg n)$. To prove the upper bound, we will show that $T(n) = O(n \lg n)$. Our inductive hypothesis is that $T(n) \leq cn \lg n$ for some constant c . We have

$$\begin{aligned}
T(n) &= T(n-1) + \lg n \\
&\leq c(n-1) \lg(n-1) + \lg n \\
&= cn \lg(n-1) - c \lg(n-1) + \lg n \\
&\leq cn \lg(n-1) - c \lg(n/2) + \lg n \\
&\quad (\text{since } \lg(n-1) \geq \lg(n/2) \text{ for } n \geq 2) \\
&= cn \lg(n-1) - c \lg n + c + \lg n \\
&< cn \lg n - c \lg n + c + \lg n \\
&\leq cn \lg n,
\end{aligned}$$

if $-c \lg n + c + \lg n \leq 0$. Equivalently,

$$\begin{aligned}
-c \lg n + c + \lg n &\leq 0 \\
c &\leq (c-1) \lg n \\
\lg n &\geq c/(c-1).
\end{aligned}$$

This works for $c = 2$ and all $n \geq 4$.

To prove the lower bound, we will show that $T(n) = \Omega(n \lg n)$. Our inductive hypothesis is that $T(n) \geq cn \lg n + dn$ for constants c and d . We have

$$\begin{aligned}
T(n) &= T(n-1) + \lg n \\
&\geq c(n-1) \lg(n-1) + d(n-1) + \lg n \\
&= cn \lg(n-1) - c \lg(n-1) + dn - d + \lg n \\
&\geq cn \lg(n/2) - c \lg(n-1) + dn - d + \lg n \\
&\quad (\text{since } \lg(n-1) \geq \lg(n/2) \text{ for } n \geq 2) \\
&= cn \lg n - cn - c \lg(n-1) + dn - d + \lg n \\
&\geq cn \lg n,
\end{aligned}$$

if $-cn - c \lg(n-1) + dn - d + \lg n \geq 0$. Since

$$\begin{aligned}
-cn - c \lg(n-1) + dn - d + \lg n &> \\
-cn - c \lg(n-1) + dn - d + \lg(n-1),
\end{aligned}$$

it suffices to find conditions in which $-cn - c \lg(n-1) + dn - d + \lg(n-1) \geq 0$. Equivalently,

$$\begin{aligned}
-cn - c \lg(n-1) + dn - d + \lg(n-1) &\geq 0 \\
(d-c)n &\geq (c-1) \lg(n-1) + d.
\end{aligned}$$

This works for $c = 1, d = 2$, and all $n \geq 2$.

Since $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$.

i. $T(n) = T(n-2) + 2 \lg n$

We guess that $T(n) = \Theta(n \lg n)$. We show the upper bound of $T(n) = O(n \lg n)$ by means of the inductive hypothesis $T(n) \leq cn \lg n$ for some constant $c > 0$. We have

$$\begin{aligned}
T(n) &= T(n-2) + 2 \lg n \\
&\leq c(n-2) \lg(n-2) + 2 \lg n \\
&\leq c(n-2) \lg n + 2 \lg n \\
&= (cn - 2c + 2) \lg n
\end{aligned}$$

$$\begin{aligned}
&= cn \lg n + (2 - 2c) \lg n \\
&\leq cn \lg n \quad \text{if } c > 1.
\end{aligned}$$

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

For the lower bound of $T(n) = \Omega(n \lg n)$, we'll show that $T(n) \geq cn \lg n + dn$, for constants $c, d > 0$ to be chosen. We assume that $n \geq 4$, which implies that

1. $\lg(n-2) \geq \lg(n/2)$,
2. $n/2 \geq \lg n$, and
3. $n/2 \geq 2$.

(We'll use these inequalities as we go along.) We have

$$\begin{aligned}
T(n) &\geq c(n-2) \lg(n-2) + d(n-2) + 2 \lg n \\
&= cn \lg(n-2) - 2c \lg(n-2) + dn - 2d + 2 \lg n \\
&> cn \lg(n-2) - 2c \lg n + dn - 2d + 2 \lg n \\
&\quad \text{(since } -\lg n < -\lg(n-2)\text{)} \\
&= cn \lg(n-2) - 2(c-1) \lg n + dn - 2d \\
&\geq cn \lg(n/2) - 2(c-1) \lg n + dn - 2d \quad \text{(by inequality (1) above)} \\
&= cn \lg n - cn - 2(c-1) \lg n + dn - 2d \\
&\geq cn \lg n,
\end{aligned}$$

if $-cn - 2(c-1) \lg n + dn - 2d \geq 0$ or, equivalently, $dn \geq cn + 2(c-1) \lg n + 2d$. Pick any constant $c > 1/2$, and then pick any constant d such that

$$d \geq 2(2c - 1).$$

(The requirement that $c > 1/2$ means that d is positive.) Then

$$d/2 \geq 2c - 1 = c + (c - 1),$$

and adding $d/2$ to both sides, we have

$$d \geq c + (c - 1) + d/2.$$

Multiplying by n yields

$$dn \geq cn + (c - 1)n + dn/2,$$

and then both multiplying and dividing the middle term by 2 gives

$$dn \geq cn + 2(c - 1)n/2 + dn/2.$$

Using inequalities (2) and (3) above, we get

$$dn \geq cn + 2(c - 1) \lg n + 2d,$$

which is what we needed to show. Thus $T(n) = \Omega(n \lg n)$. Since $T(n) = O(n \lg n)$ and $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$.

