CMPS 102
Introduction to Analysis of Algorithms
Recurrence Relations

When analyzing the run time of recursive algorithms we are often led to consider functions $T(n)$ defined by recurrence relations of a certain form. A typical example would be

$$T(n) = \begin{cases} c & n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + dn & n > 1 \end{cases}$$

where *c*, *d* are fixed constants. The specific values of these constants are important for determining the *explicit solution* to the recurrence. Often however we are only concerned with finding an asymptotic (upper, lower, or tight) bound on the solution. We call such a bound an *asymptotic solution* to the recurrence. In the above example the particular constants *c* and *d* have no effect on the asymptotic solution.

We will study three methods for solving recurrences.

- **Substitution Method.** This method consists of guessing an asymptotic (upper or lower) bound on the solution, and trying to prove it by induction.

- **Recursion Tree Method – Iteration Method.** These are two closely related methods for expressing $T(n)$ as a summation that can then be analyzed. A recursion tree is a graphical depiction of the entire set of recursive invocations of the function *T*. The goal of drawing a recursion tree is to obtain a guess which can then be verified by the more rigorous substitution method. Iteration is an algebraic version of the Recursion Tree Method, and consists of repeatedly substituting the recurrence into itself to obtain an iterated (i.e. summation) expression.

- **Master Method.** This is a cookbook method for determining asymptotic solutions to recurrences of a specific form.

The Substitution Method
We begin with the following example.

$$T(n) = \begin{cases} 2 & 1 \leq n < 3 \\ 3T(\lfloor n/3 \rfloor) + n & n \geq 3 \end{cases}$$

Suppose that we are able to guess somehow that $T(n) = O\big(n\log(n)\big)$. In order to prove this guess, we must find positive numbers *c* and n_0 such that $T(n) \leq cn\log(n)$ for all $n \geq n_0$. If we knew appropriate values for these constants, we could prove this inequality by induction. Our goal then, is to determine *c* and n_0 such that an induction proof can be made to work.

Observe that the recurrence itself contains two base cases: $n = 1$ and $n = 2$. This indicates that the induction proof may also require multiple base cases. However, the inequality to be proved $T(n) \leq cn\log(n)$, is actually false in the case $n = 1$, since $T(1) = 2$ and $\log(1) = 0$. Since $\log(2) \neq 0$, the same problem does not occur at $n = 2$. Indeed, for $n = 2$ we seek to show $T(2) \leq c \cdot 2 \cdot \log(2)$, which can be made true by a proper choice of c. Therefore we take the lowest base case to be $n_0 = 2$. It remains to determine the highest base case, which we will denote by n_1. We begin by mimicking the induction step

IId, with lowest base case $n = 2$ and highest base case $n = n_1$. In what follows, it will be algebraically convenient to take $\log()$ to mean $\log_3()$.

> Let $n > n_1$, and assume $T(k) \le ck\log(k)$ for all k in the range $2 \le k < n$. In particular, when $k = \lfloor n/3 \rfloor$ we have $T(\lfloor n/3 \rfloor) \le c\lfloor n/3 \rfloor \log(\lfloor n/3 \rfloor)$. We must show that $T(n) \le cn\log(n)$.
> We have

$$\begin{aligned} T(n) &= 3T(\lfloor n/3 \rfloor) + n && \text{(by the recurrence for } T\text{)} \\ &\le 3c\lfloor n/3 \rfloor \log(\lfloor n/3 \rfloor) + n && \text{(by the induction hypothesis)} \\ &\le 3c(n/3)\log(n/3) + n && \text{(since } \lfloor x \rfloor \le x \text{ for all } x\text{)} \\ &= cn(\log(n) - 1) + n \\ &= cn\log(n) - cn + n \end{aligned}$$

To obtain $T(n) \le cn\log(n)$, we need to have $-cn + n \le 0$, which will be true if $c \ge 1$. Thus as long as c is chosen to satisfy the constraint $c \ge 1$, the induction step will go through.

It remains to determine the constant n_1, which represents the highest base case. Since we only use the induction hypothesis in the case $k = \lfloor n/3 \rfloor$, we require that $2 \le \lfloor n/3 \rfloor < n$ whenever $n > n_1$. This is equivalent to $n > n_1 \Rightarrow 6 \le n$, which indicates we should choose $n_1 = 5$. The base cases are then $n = 2, 3, 4, 5$, and it is required that

$$\begin{aligned} &T(2) \le c \cdot 2 \cdot \log(2), \\ &T(3) \le c \cdot 3 \cdot \log(3), \\ &T(4) \le c \cdot 4 \cdot \log(4), \\ \text{and } &T(5) \le c \cdot 5 \cdot \log(5) \end{aligned}$$

One checks that $T(2) = 2, T(3) = 9, T(4) = 10$ and $T(5) = 11$. To satisfy all constraints then, we take c to be the maximum of the numbers $\{\, 1, 2/2\log(2), 9/3\log(3), 10/4\log(4), 11/5\log(5) \,\}$. A few comparisons reveal this maximum to be $c = 9/3\log_3(3) = 3$.

It is important to realize that we have not proved anything yet. Everything we have done up to this point has been scratch work with the goal of finding appropriate values for c and n_0. It remains to present a complete induction proof of the assertion: $T(n) \le 3n\log(n)$ for all $n \ge 2$.

Proof:

I. The inequality $T(n) \le 3n\log(n)$ in the cases $n = 2, 3, 4$ and 5 becomes, respectively, $2 \le 6\log(2)$, $9 \le 9$, $10 \le 12\log(4)$ and $11 \le 15\log(5)$, which are all readily verified.

II. Let $n > 5$ and assume for all k in the range $2 \le k < n$ that $T(k) \le 3k\log(k)$. In particular, for $k = \lfloor n/3 \rfloor$ we have

$$\begin{aligned} T(n) &= 3T(\lfloor n/3 \rfloor) + n && \text{(by the recurrence for } T\text{)} \\ &\le 3 \cdot 3\lfloor n/3 \rfloor \log(\lfloor n/3 \rfloor) + n && \text{(by the induction hypothesis with } k = \lfloor n/3 \rfloor\text{)} \\ &\le 9(n/3)\log(n/3) + n && \text{(since } \lfloor x \rfloor \le x \text{ for all } x\text{)} \\ &= 3n(\log(n) - 1) + n \\ &= 3n\log(n) - 2n \\ &\le 3n\log(n) \end{aligned}$$

It follows from the 2nd PMI that $T(n) \le 3n\log(n)$ for all $n \ge 2$. ■

It is a somewhat more difficult exercise to prove by the same technique that $T(n) = \Omega(n \log(n))$, and hence $T(n) = \Theta(n \log(n))$.

Exercise Determine positive numbers c and n_0 such that $T(n) \geq cn \log(n)$ for all $n \geq n_0$ (where again log() means $\log_3()$). Hint: Use the following facts: (1) $\lfloor x \rfloor > x - 1$, and (2) $\log_3 \lfloor x \rfloor \geq \log_3(x) - 1$ for $x \geq 3/2$. (With some effort, it's possible to show that $c = 1/4$ and $n_0 = 2$ work. As before use multiple base cases with the lowest being $n_0 = 2$ and highest $n_1 = 5$)

The next example illustrates one complication that can arise in the substitution method. Define $T(n)$ by the recurrence

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor n/2 \rfloor) + 1 & n \geq 2 \end{cases}$$

We guess that $T(n) = O(n)$. To prove this guess, we attempt to find positive constant c and n_0 such that $T(n) \leq cn$ for all $n \geq n_0$. As before we proceed by mimicking the induction step. Let $n > n_0$ and assume for all k in the range $n_0 \leq k < n$ that $T(k) \leq ck$. In particular, for $k = \lfloor n/2 \rfloor$ we have $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor$, and thus

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + 1 && \text{(by the recurrence for } T(n)) \\ &\leq 2c\lfloor n/2 \rfloor + 1 && \text{(by the induction hypothesis)} \\ &\leq 2c(n/2) + 1 \\ &= cn + 1 \end{aligned}$$

Our goal is to determine c so that the induction step is feasible. We need to reach the conclusion that $T(n) \leq cn$, and to do this we must determine a number c such that $cn + 1 \leq cn$. But this inequality is obviously false for all positive c. Apparently the induction step cannot be made to work, which might lead us to believe that our guess is incorrect. Actually $T(n) = O(n)$ *is* correct (as can be seen by other methods), so we take a different approach. Our trick is to subtract a lower order term from the right side of the inequality we wish to prove. To be precise, we seek positive c, n_0, and b such that $T(n) \leq cn - b$ for all $n \geq n_0$. Observe that this inequality also implies $T(n) = O(n)$ by a result proved in the handout on asymptotic growth rates. It may seem counter-intuitive to attempt to prove an inequality that is stronger than the one that failed, but notice this strategy gives us a stronger induction hypothesis. Again let $n > n_0$ and assume for all k in the range $n_0 \leq k < n$ that $T(k) \leq ck - b$, and in particular, for $k = \lfloor n/2 \rfloor$ we have $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor - b$. Thus

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + 1 && \text{(by the recurrence for } T(n)) \\ &\leq 2(c\lfloor n/2 \rfloor - b) + 1 && \text{(by the induction hypothesis)} \\ &\leq 2(c(n/2) - b) + 1 && \text{(since } \lfloor x \rfloor \leq x) \\ &= cn - 2b + 1 \end{aligned}$$

If we take $b \geq 1$, then $T(n) \leq cn - b$, as desired. Taking $n_0 = 1$, we require $T(1) \leq c \cdot 1 - 1$, which says $c \geq T(1) + 1 = 2$. Thus we may set $n_0 = 1$, $c = 2$, and $b = 1$.

Exercise Referring to the previous example, use induction to prove that $T(n) \leq 2n - 1$ for all $n \geq 1$, whence $T(n) = 0(n)$.

Exercise Referring to the previous example, use the substitution method to show that $T(n) = \Omega(n)$, and hence $T(n) = \Theta(n)$.

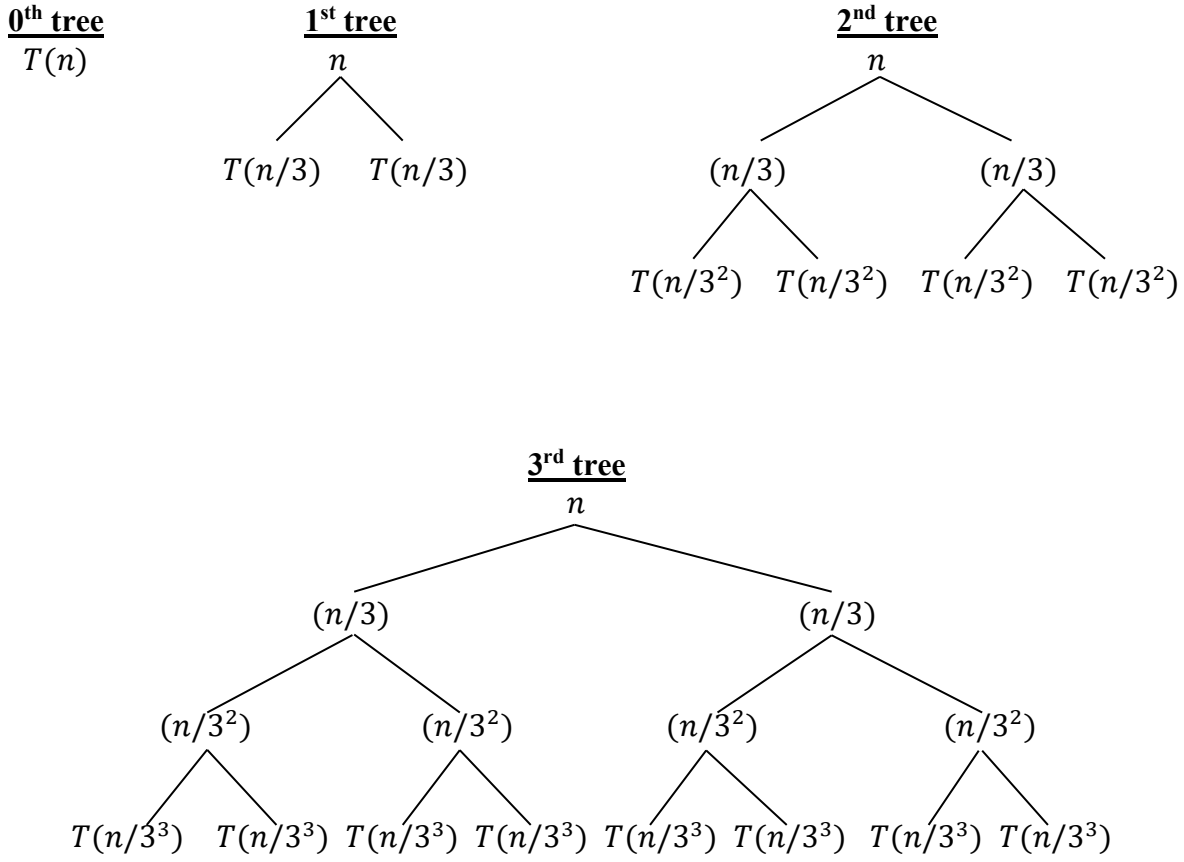
The Recursion Tree – Iteration Method

The *recursion tree method* can be used to generate a good guess for an asymptotic bound on a recurrence. This guess can then be verified by the substitution method. Since our guess will be verified, we can take some liberties in our calculations, such as dropping floors and ceilings or restricting the values of *n*.

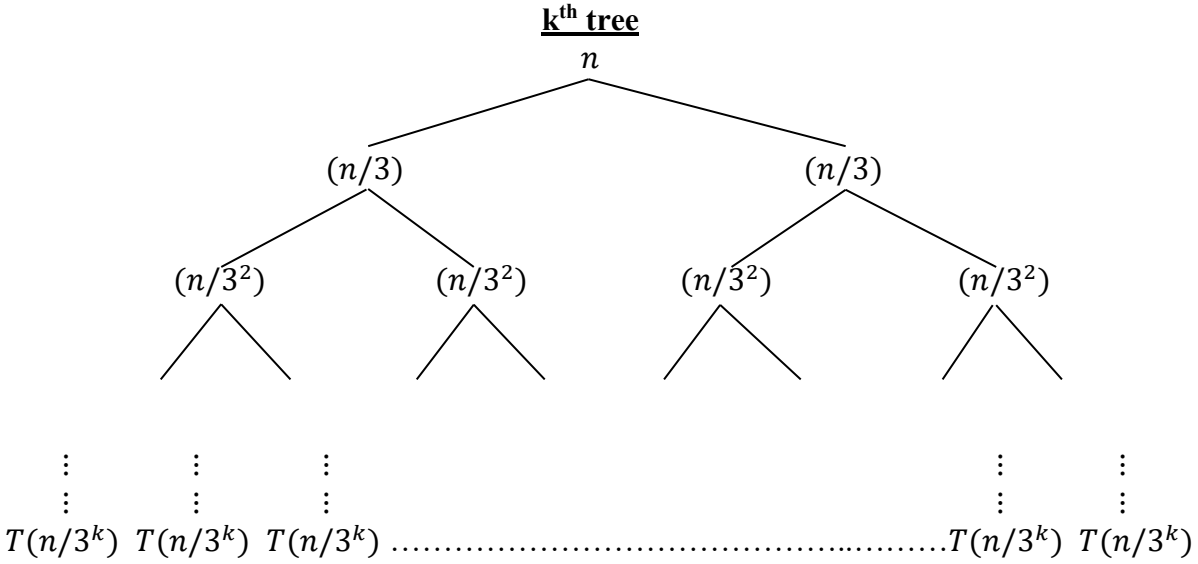
Let us begin with the example

$$T(n) = \begin{cases} 1 & 1 \le n < 3 \\ 2T(\lfloor n/3 \rfloor) + n & n \ge 3 \end{cases}$$

Each node in a recursion tree represents one term in the calculation of $T(n)$ obtained by recursively substituting the expression for *T* into itself. We construct a sequence of such trees of increasing depths.



By summing the nodes in any one of these trees, we obtain an expression for $T(n)$. After k iterations of this process we reach a tree in which all bottom level nodes are $T(n/3^k)$.



Note that there are 2^i nodes at depth i, each of which has value $n/3^i$ (for $0 \le i \le k-1$). The sequence of trees terminates when all bottom level nodes are $T(1)$, i.e. when $n/3^k = 1$, which implies $k = \log_3(n)$. The number of nodes at this bottom level is therefore $2^k = 2^{\log_3(n)} = n^{\log_3(2)}$. Summing all nodes in the final recursion tree gives us the following expression for $T(n)$.

$$\begin{aligned} T(n) &= \sum_{i=0}^{k-1} 2^i \cdot (n/3^i) \;\; + \;\; n^{\log_3(2)} \cdot T(1) \\ &= n\left(\sum_{i=0}^{k-1} (2/3)^i\right) \;\; + \;\; n^{\log_3(2)} \cdot T(1) \\ &= n\left(\frac{1-(2/3)^k}{1-(2/3)}\right) \;\; + \;\; n^{\log_3(2)} \cdot T(1) \\ &= 3n(1-(2/3)^k) \;\; + \;\; \Theta(n^{\log_3(2)}) \end{aligned}$$

If we seek an asymptotic upper bound we may drop the negative term to obtain $T(n) \le 3n + \Theta(n^{\log_3(2)})$. Since $\log_3(2) < 1$ the first term dominates, and so we guess: $T(n) = O(n)$.

Exercise Prove that this guess is correct using the substitution method.

Exercise Prove that $T(n) = \Omega(n)$ in the preceding example. Hint: examine the recurrence itself.

It is sometimes possible to push these computations a little further to obtain an asymptotic estimate directly, with no simplifying assumptions. We call this the *iteration method*. We illustrate on the very same example.

$$T(n) = \begin{cases} 1 & 1 \le n < 3 \\ 2T(\lfloor n/3 \rfloor) + n & n \ge 3 \end{cases}$$

Here we do not assume that n is an exact power of 3, and keep the floor. Substituting this expression into itself k times yields:

$$\begin{aligned}
T(n) &= n + 2T(\lfloor n/3 \rfloor) \\
&= n + 2\left(\lfloor n/3 \rfloor + 2T\left(\left\lfloor \lfloor n/3 \rfloor / 3 \right\rfloor\right)\right) \\
&= n + 2\lfloor n/3 \rfloor + 2^2 T(\lfloor n/3^2 \rfloor) \\
&= n + 2\lfloor n/3 \rfloor + 2^2\left(\lfloor n/3^2 \rfloor + 2T\left(\left\lfloor \lfloor n/3^2 \rfloor / 3 \right\rfloor\right)\right) \\
&= n + 2\lfloor n/3 \rfloor + 2^2\lfloor n/3^2 \rfloor + 2^3 T(\lfloor n/3^3 \rfloor) \\
&\vdots \\
&\vdots \\
&= \textstyle\sum_{i=0}^{k-1} 2^i \cdot \left\lfloor n/3^i \right\rfloor \;+\; 2^k T(\lfloor n/3^k \rfloor)
\end{aligned}$$

The process must terminate when k is chosen so that $1 \le \lfloor n/3^k \rfloor < 3$, which implies

$$\begin{aligned}
& 1 \le n/3^k < 3 \\
\therefore\;\; & 3^k \le n < 3^{k+1} \\
\therefore\;\; & k \le log_3(n) < k+1 \\
\therefore\;\; & k = \lfloor log_3(n) \rfloor
\end{aligned}$$

With this value of k we have $T(\lfloor n/3^k \rfloor) = 1$, whence $T(n) = \sum_{i=0}^{k-1} 2^i \cdot \left\lfloor n/3^i \right\rfloor + 2^k$. This expression in essence converts the computation of $T(n)$ from a recursive algorithm to an iterative algorithm, which is why we call this the *iteration method*. This summation can now be used to obtain asymptotic upper and lower bounds for $T(n)$.

$$\begin{aligned}
T(n) &= \textstyle\sum_{i=0}^{k-1} 2^i \cdot \left\lfloor n/3^i \right\rfloor \;+\; 2^k \\
&\le \textstyle\sum_{i=0}^{k-1} 2^i \cdot \left(n/3^i\right) \;+\; 2^{\log_3(n)} && \text{(since } \lfloor x \rfloor \le x\text{)} \\
&= n\textstyle\sum_{i=0}^{k-1}(2/3)^i \;+\; n^{\log_3(2)} \\
&\le n\textstyle\sum_{i=0}^{\infty}(2/3)^i \;+\; n^{\log_3(2)} \\
&= n\left(\frac{1}{1-(2/3)}\right) \;+\; n^{\log_3(2)} \\
&= 3n + n^{log_3(2)} \\
&= O(n) && \text{(since } \log_3(2) < 1\text{)}
\end{aligned}$$

Thus $T(n) = O(n)$ has been proved. Note that our proof is rigorous, since at no point have we made any simplifying assumptions (such as n being an exact power of 3.) Therefore there is no need to verify $T(n) = O(n)$ by another method, such as substitution. $T(n) = \Omega(n)$ is easy to establish since for $n \ge 3$, we have $T(n) = 2T(\lfloor n/3 \rfloor) + n \ge n = \Omega(n)$. We now have $T(n) = \Theta(n)$.

The iteration method can even be used to find *exact solutions* to some recurrences, as opposed to the *asymptotic solutions* we have been satisfied with up to now. For example define the function $T(n)$ by $T(0) = T(1) = 5$, and $T(n) = T(n-2) + n$ for $n \ge 2$. Iterating k times we find

$$\begin{aligned}
T(n) &= n + T(n-2) \\
&= n + (n-2) + T(n-4) \\
&= n + (n-2) + (n-4) + T(n-6) \\
&\vdots
\end{aligned}$$

$$
\begin{aligned}
&\vdots \\
&= \textstyle\sum_{i=0}^{k-1}(n-2i) \;+\; T(n-2k) \\
&= \textstyle\sum_{i=0}^{k-1} n \;-\; 2\sum_{i=0}^{k-1} i \;+\; T(n-2k) \\
&= kn \;-\; 2\left(\tfrac{k(k-1)}{2}\right) \;+\; T(n-2k) \\
&= kn \;-\; k(k-1) \;+\; T(n-2k)
\end{aligned}
$$

This process must terminate when k is chosen so that either $n-2k=0$ or $n-2k=1$, which is where the recurrence 'bottoms out' so to speak. This implies

$$
\begin{aligned}
&0 \le n-2k < 2 \\
\therefore\; &2k \le n < 2k+2 \\
\therefore\; &k \le \frac{n}{2} < k+1 \\
\therefore\; &k = \lfloor n/2 \rfloor
\end{aligned}
$$

Thus if $k=\lfloor n/2 \rfloor$ we have $T(n-2k)=5$, and therefore the exact solution to the above recurrence is given by

$$
T(n) = \left\lfloor \frac{n}{2} \right\rfloor \cdot n - \left\lfloor \frac{n}{2} \right\rfloor \cdot \left(\left\lfloor \frac{n}{2} \right\rfloor - 1 \right) + 5
$$

The asymptotic solution is now readily seen to be $T(n)=\Theta(n^2)$.

Recall the following example from the induction handout.

$$
T(n) = \begin{cases} 0 & n = 1 \\ T(\lfloor n/2 \rfloor) + 1 & n \ge 2 \end{cases}
$$

Applying the iteration method to this recurrence yields.

$$
\begin{aligned}
T(n) &= 1 + T(\lfloor n/2 \rfloor) \\
&= 1 + 1 + T(\lfloor \lfloor n/2 \rfloor / 2 \rfloor) = 2 + T(\lfloor n/2^2 \rfloor) \\
&= 2 + 1 + T(\lfloor \lfloor n/2^2 \rfloor / 2 \rfloor) = 3 + T(\lfloor n/2^3 \rfloor) \\
&\vdots \\
&= k + T(\lfloor n/2^k \rfloor)
\end{aligned}
$$

This process must terminate when the recursion depth k is at its maximum, i.e. when $\lfloor n/2^k \rfloor = 1$. To solve this equation for k in terms of n, we use the inequality definition of the floor function.

$$
\begin{aligned}
&1 \le n/2^k < 2 \\
\therefore\quad &2^k \le n < 2^{k+1} \\
\therefore\quad &k \le \lg(n) < k+1 \\
\therefore\quad &k = \lfloor \lg(n) \rfloor
\end{aligned}
$$

Thus for the recursion depth $k=\lfloor \lg(n) \rfloor$ we have $T(\lfloor n/2^k \rfloor) = T(1) = 0$, and hence the solution to the above recurrence is $T(n)=\lfloor \lg(n) \rfloor$. It follows that $T(n)=\Theta(\log(n))$.

Exercise
Check directly that $T(n) = \lfloor \lg(n) \rfloor$ is the solution to the above recurrence relation, i.e. check that $T(1) = 0$, and for any $n \geq 2$, that $T(n) = 1 + T(\lfloor n/2 \rfloor)$.

Exercise
Use this same technique to show that the recurrence

$$S(n) = \begin{cases} 0 & n = 1 \\ S(\lceil n/2 \rceil) + 1 & n \geq 2 \end{cases}$$

has solution $S(n) = \lceil \lg(n) \rceil$, and hence also $S(n) = \Theta(\log(n))$.

Comparing the solutions to the preceding examples, we see that replacing floor $\lfloor \ \rfloor$ by ceiling $\lceil \ \rceil$ has no effect on the *asymptotic solution* $\Theta(\log(n))$, while the *exact solutions* are different: $\lfloor \lg(n) \rfloor$ vs. $\lceil \lg(n) \rceil$. We can change other details of a recurrence without changing the asymptotic solution. The following recurrence satisfies $T(n) = \Theta(\log(n))$ for any values of the constants *c, d*, and n_0.

$$T(n) = \begin{cases} c & 1 \leq n < n_0 \\ T(\lfloor n/2 \rfloor) + d & n \geq n_0 \end{cases}$$

Using the iteration method yields:

$$\begin{aligned} T(n) &= d + T(\lfloor n/2 \rfloor) \\ &= d + d + T(\lfloor n/2^2 \rfloor) \\ &\vdots \\ &= kd + T(\lfloor n/2^k \rfloor) \end{aligned}$$

We seek the first (i.e. the smallest) integer k such that $\lfloor n/2^k \rfloor < n_0$. This is equivalent to $n/2^k < n_0$, whence

$$\frac{n}{n_0} < 2^k$$
$k - 1 \leq \lg(n/n_0) < k$ (since *k* is the least integer satisfying the previous inequality)
$k - 1 = \lfloor \lg(n/n_0) \rfloor$
$k = \lfloor \lg(n/n_0) \rfloor + 1$

Thus $T(n) = (\lfloor \lg(n/n_0) \rfloor + 1) \cdot d + c$ for $n \geq n_0$, and hence $T(n) = \Theta(\log(n))$, as claimed.

It is often difficult or impossible to determine an exact solution via the iteration method, while it *is* possible to obtain an asymptotic solution. Consider

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lfloor n/2 \rfloor) + n^2 & n \geq 2 \end{cases}$$

Upon iterating this recurrence we find $T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor^2 + 1$, where $k = \lfloor \lg n \rfloor$. We can use this expression to show that $T(n) = \Theta(n^2)$ as follows.

$$T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor^2 + 1$$

$$\leq n^2\left(\sum_{i=0}^{k-1}(1/4)^i\right) + 1 \qquad \text{(since } \lfloor x \rfloor \leq x\text{)}$$

$$\leq n^2\left(\sum_{i=0}^{\infty}(1/4)^i\right) + 1 \qquad \text{(letting } k \to \infty\text{)}$$

$$= n^2\left(\frac{1}{1-(1/4)}\right) + 1 \qquad \text{(from a well-known series formula)}$$

$$= \frac{4}{3}n^2 \quad + \quad 1$$

$$= \ O(n^2)$$

Therefore $T(n) = O(n^2)$. The lower bound $T(n) = \Omega(n^2)$ again follows directly from the recurrence since $T(\lfloor n/2 \rfloor) + n^2 \geq n^2$ for $n \geq 2$. Therefore $T(n) = \Theta(n^2)$, as claimed.

The Master Method
The Master Theorem determines (asymptotic) solutions to recurrences of the form

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

It is required that $a \geq 1$, $b > 1$, and that the function $f(n)$ be asymptotically positive. It is understood that $T(n/b)$ denotes either $T(\lfloor n/b \rfloor)$ or $T(\lceil n/b \rceil)$, and that $T(n) = \Theta(1)$ for some finite set of initial terms. It is part of the content of the theorem that these details, while they may change the exact solution, do not change the asymptotic solution. Such recurrence relations describe the run time of "divide and conquer" algorithms that divide a problem of size *n* into *a* subproblems, each of size n/b. In this context $f(n)$ represents the cost of doing the dividing and re-combining.

Master Theorem
Let $a \geq 1$, $b > 1$, $f(n)$ be asymptotically positive, and let $T(n)$ be defined by $T(n) = aT(n/b) + f(n)$. Then we have three cases:

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

Remarks In each case we compare $f(n)$ to the polynomial $n^{\log_b(a)}$, then determine which of these two functions is of a higher asymptotic order. In case (1) $n^{\log_b(a)}$ is of higher order than $f(n)$ (by a polynomial factor n^{ε}) and the solution $T(n)$ is in the class $\Theta\left(n^{\log_b(a)}\right)$. In case (3) $f(n)$ is the higher order (again by a polynomial factor n^{ε}), and an additional *regularity condition* is satisfied. The Master Theorem tells us in this case that $T(n) = \Theta\left(f(n)\right)$. In case (2) the two functions are asymptotically equivalent, and $T(n)$ is in the class $\Theta\left(n^{\log_b(a)} \cdot \log(n)\right) = \Theta\left(f(n) \cdot \log(n)\right)$.

We sometimes say, as in case (1), that $n^{\log_b(a)}$ is *polynomially larger* than $f(n)$, meaning that $f(n)$ is bounded above by a function which is smaller than $n^{\log_b(a)}$ by a polynomial factor, namely n^{ε} for some $\varepsilon > 0$.

Example $T(n) = 8T(n/2) + n^3$

Observe that $a = 8$, $b = 2$, and $\log_b(a) = 3$. Hence $f(n) = n^3 = \Theta\left(n^{\log_b(a)}\right)$, so we are in case (2). Therefore $T(n) = \Theta(n^3 \log(n))$.

Example $T(n) = 5T(n/4) + n$

In this case $a = 5$, $b = 4$, and $\log_b(a) = 1.1609\ldots > 1$. Letting $\varepsilon = \log_4(5) - 1$ we have $\varepsilon > 0$, and therefore $f(n) = n = O\left(n^{\log_4(5)-\varepsilon}\right)$. Thus case(1) applies, and $T(n) = \Theta\left(n^{\log_4(5)}\right)$.

Example $T(n) = 5T(n/4) + n^2$

Again $a = 5$, $b = 4$, so that $\log_b(a) = 1.1609\ldots < 2$. Upon setting $\varepsilon = 2 - \log_4(5)$, we have that $\varepsilon > 0$, and $f(n) = n^2 = \Omega\left(n^{\log_4(5)+\varepsilon}\right)$. If the Master Theorem applies at all then, it must be that case (3) applies. However, the regularity condition must also be checked: $5f(n/4) \leq cf(n)$ for some $0 < c < 1$ and all sufficiently large *n*. This inequality says $5(n/4)^2 \leq cn^2$, i.e. $(5/16)n^2 \leq cn^2$, which is true as long as *c* is chosen to satisfy $5/16 \leq c < 1$. By case (3) $T(n) = \Theta(n^2)$.

The conclusion reached by the Master Theorem does not change when we replace $f(n)$ by a function which is asymptotically equivalent to it. This is implied by the theorem since the hypothesis in each case refers only to the asymptotic growth rate of $f(n)$, not its actual numerical values. Thus we can, if convenient, replace $f(n)$ by any simpler asymptotically equivalent function. For instance the recurrence $T(n) = 8T(n/2) + 10n^3 + 15n^2 - n^{1.5} + n\log(n) + 1$ can be reduced to $T(n) = 8T(n/2) + n^3$ which was our first example above, and we arrive at the very same asymptotic solution. (Of course the exact solution to the recurrence would be quite different.) For this reason recurrences are sometimes specified in the form $T(n) = aT(n/b) + \Theta(f(n))$, if all that is required is an asymptotic solution. Notice that there is no mention of initial terms in the Master Theorem. It is part of the content of the theorem that the initial values of the recurrence do not effect it's asymptotic solution.

Observe that in the three preceding examples, $f(n)$ was a polynomial. This is a particularly easy setting in which to apply the Master Theorem, since to establish which case applies, one merely compares $\deg(f)$ to the number $\log_b(a)$. If they are the same, case (2) applies. If $\log_b(a)$ is larger, case (1) applies by defining $\varepsilon = \log_b(a) - \deg(f)$. When $\deg(f)$ is larger, case (3) applies upon setting $\epsilon = \deg(f) - \log_b a$, and it turns out that the regularity condition automatically holds.

Exercise: Prove that if $f(n)$ is a polynomial, and if $\deg(f) > \log_b(a)$, then case (3) of the Master Theorem applies, and that the regularity condition necessarily holds.

Checking the hypotheses can be a little more complicated if $f(n)$ is not a polynomial.

Example $T(n) = T(\lfloor n/2 \rfloor) + 2T(\lceil n/2 \rceil) + \log(n!)$

First write this as $T(n) = 3T(n/2) + n\log(n)$. Let $\varepsilon = \frac{1}{2}(\log_2(3) - 1)$, then $\varepsilon > 0$, and $1 + \varepsilon = \log_2(3) - \varepsilon$. Observe $n\log(n) = o(n^{1+\varepsilon})$, whence $n\log(n) = O(n^{1+\varepsilon}) = O(n^{\log_2(3)-\varepsilon})$. Case (1) gives $T(n) = \Theta(n^{\log_2(3)})$.

In spite of the name "Master Theorem" the three cases do not cover all possibilities. There is a gap between cases (1) and (2) when $n^{\log_b(a)}$ is of higher asymptotic order than $f(n)$, but not *polynomially* higher. The following example illustrates this situation.

Example $T(n) = 2T(n/2) \ + \ n/\log(n)$

Observe that $n/\log(n) = \omega(n^{1-\varepsilon})$ for any $\varepsilon > 0$, whence $n/\log(n) \neq O(n^{1-\varepsilon})$, and therefore we are not in case (1). But also $n/\log(n) = o(n)$, so that $\frac{n}{\log(n)} \neq \Theta(n)$, and neither are we in case (2). Thus the Master Theorem cannot be applied to this recurrence.

Exercise: Find an example where $f(n)$ is of higher asymptotic order than $n^{\log_b a}$, i.e. $f(n)$ "wins", but not by a polynomial factor.

Exercise: Find an example where $f(n) = \Omega\left(n^{\log_b n+\epsilon}\right)$, but the regularity condition fails to hold.

Proof of the Master Theorem
We sketch here a proof of the Master Theorem. Basically the proof is the iteration method applied with full generality. We simplify matters by ignoring all floors and ceilings in the argument. A more rigorous treatment is given in the text. Upon iteration we obtain

$$\begin{aligned} T(n) &= f(n) + aT(n/b) \\ &= f(n) + af(n/b) + a^2T(n/b^2) \\ &= f(n) + af(n/b) + a^2f(n/b^2) + a^3T(n/b^3) \\ &\ \ \vdots \\ &= \sum_{i=0}^{k-1} a^i f(n/b^i) + a^kT(n/b^k) \end{aligned}$$

The recurrence terminates when $n/b^k = 1$, and this happens when the recursion depth is $k = log_b(n)$. For this value of k we have $T(n) = \sum_{i=0}^{k-1} af(n/b^i) + a^{\log_b n} \cdot T(1)$, which yields

$$T(n) = \sum_{i=0}^{k-1} af(n/b^i) + T(1) \cdot n^{\log_b a}$$

It follows that $T(n) \geq \text{const} \cdot n^{\log_b a} = \Omega(n^{\log_b a})$, and hence $T(n) = \Omega(n^{\log_b n})$. From the recurrence itself we get $T(n) = aT(n/b) + f(n) \geq f(n) = \Omega\big(f(n)\big)$, and therefore $T(n) = \Omega\big(f(n)\big)$.

At this point we specialize to case (1) and leave the other cases to the text. We must show that $T(n) \leq O(n^{\log_b(a)})$, for then we have $T(n) = \Theta(n^{\log_b(a)})$, as required. Since we are in case (1), we have $f(n) = O\left(n^{\log_b(a)-\varepsilon}\right)$, and hence there exist positive numbers c and ε such that $f(n) \leq cn^{\log_b(a)-\varepsilon}$ if n is sufficiently large. By re-defining the function $f(n)$ on finitely many initial terms, we can make this

inequality true for *all* values of *n*. This change in $f(n)$ does not affect the asymptotic solution to the recurrence (proof omitted). Therefore

$$\sum_{i=0}^{k-1} a^i f\left(n/b^i\right) \le \sum_{i=0}^{k-1} a^i \cdot c\left(\frac{n}{b^i}\right)^{\log_b(a)-\epsilon}$$

$$= cn^{\log_b(a)-\epsilon} \cdot \sum_{i=0}^{k-1} a^i \left(\frac{1}{b^i}\right)^{\log_b(a)-\epsilon}$$

$$= cn^{\log_b(a)-\epsilon} \cdot \sum_{i=0}^{k-1} \frac{a^i}{(b^{\log_b(a)})^i} \cdot (b^\epsilon)^i$$

$$= cn^{\log_b(a)-\epsilon} \cdot \sum_{i=0}^{k-1} (b^\epsilon)^i$$

$$= \frac{cn^{\log_b(a)}}{n^\epsilon} \cdot \frac{(b^\epsilon)^k - 1}{b^\epsilon - 1}$$

$$\le \left(\frac{c}{b^\epsilon - 1}\right) \cdot \frac{n^{\log_b(a)}}{n^\epsilon} \cdot (b^\epsilon)^k$$

$$= \text{const} \cdot \frac{n^{\log_b(a)}}{n^\epsilon} \cdot \left(b^{\log_b n}\right)^\epsilon = \text{const} \cdot n^{\log_b(a)} = O\left(n^{\log_b(a)}\right),$$

and hence

$$T(n) = \sum_{i=0}^{k-1} af(n/b^i) + T(1) \cdot n^{\log_b a} = O\left(n^{\log_b(a)}\right).$$

The conclusion for case (1) follows: $T(n) = \Theta(n^{\log_b(a)})$. ■

We emphasize that the above argument was only a "sketch" and not a complete proof, owing to the fact that we ignored floors and ceilings and omitted other details. See the proofs of cases (2) and (3) in section 4.6 of the text.

Exercise Sketch a proof of the Master Theorem in cases (2) and (3).