**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.

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

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 $T(n) \le cn\log(n)$ for all $n \ge n_0$. 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 $T(n) \le cn\log(n)$, is actually false in the case $n = 1$, since $T(1) = 2$ and $\log(1) = 0$. Since $\log(2) \ne 0$, the same problem does not occur at $n = 2$. Indeed, for $n = 2$ we seek to show $T(2) \le c \cdot 2 \cdot \log(2)$, 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 $T(k) \leq ck\log(k)$ for all $k$ in the range $2 \leq k < n$. In particular, when $k = \lfloor n/3 \rfloor$ we have $T(\lfloor n/3 \rfloor) \leq c\lfloor n/3 \rfloor \log(\lfloor n/3 \rfloor)$. We must show that $T(n) \leq cn\log(n)$.
> We have

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

To obtain $T(n) \leq cn\log(n)$, we need to have $-cn + n \leq 0$, which will be true if $c \geq 1$. Thus as long as $c$ is chosen to satisfy the constraint $c \geq 1$, 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 $2 \leq \lfloor n/3 \rfloor < n$ whenever $n > n_1$. This is equivalent to $n > n_1 \Rightarrow 6 \leq n$, which indicates we should choose $n_1 = 5$. The base cases are then $n = 2, 3, 4, 5$, and it is required that

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

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: $T(n) \leq 3n\log(n)$ for all $n \geq 2$.

**Proof:**

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

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

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

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

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

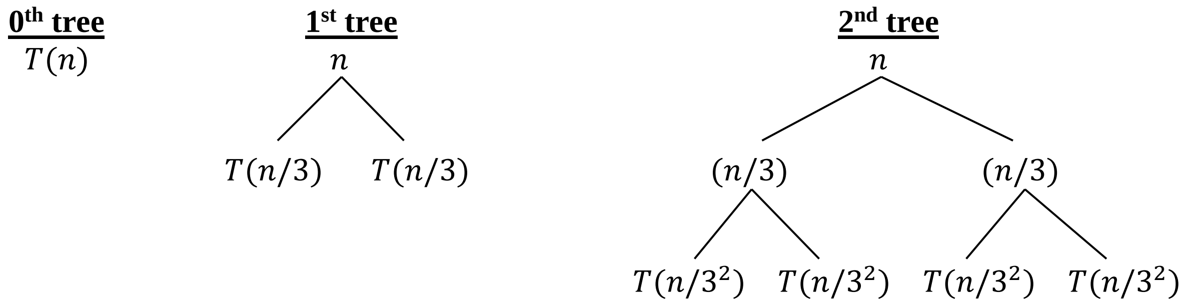
**Exercise** Referring to the previous example, use the substitution method to show that $T(n) = \Omega(n)$, and hence $T(n) = \Theta(n)$.

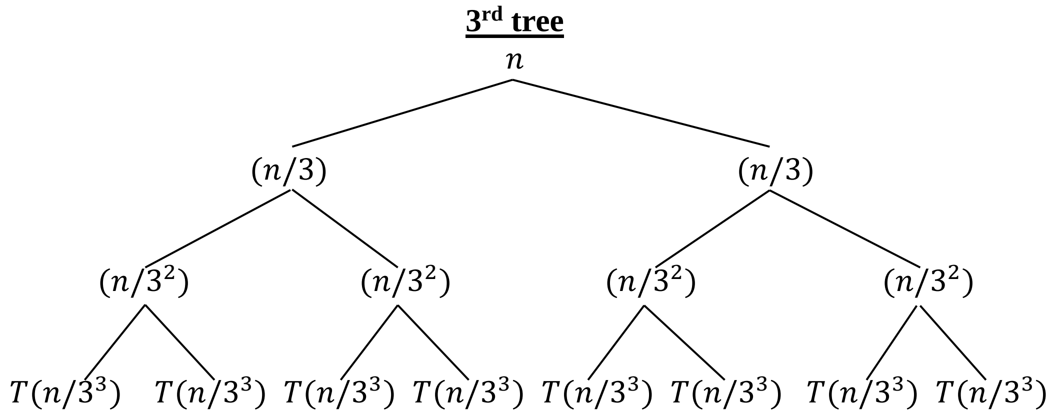
**The Recursion Tree – Iteration Method**
The *recursion tree method* can be used to generate a good guess for an asymptotic bound on a recurrence. This guess can then be verified by the substitution method. Since our guess will be verified, we can take some liberties in our calculations, such as dropping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

Let us begin with the example

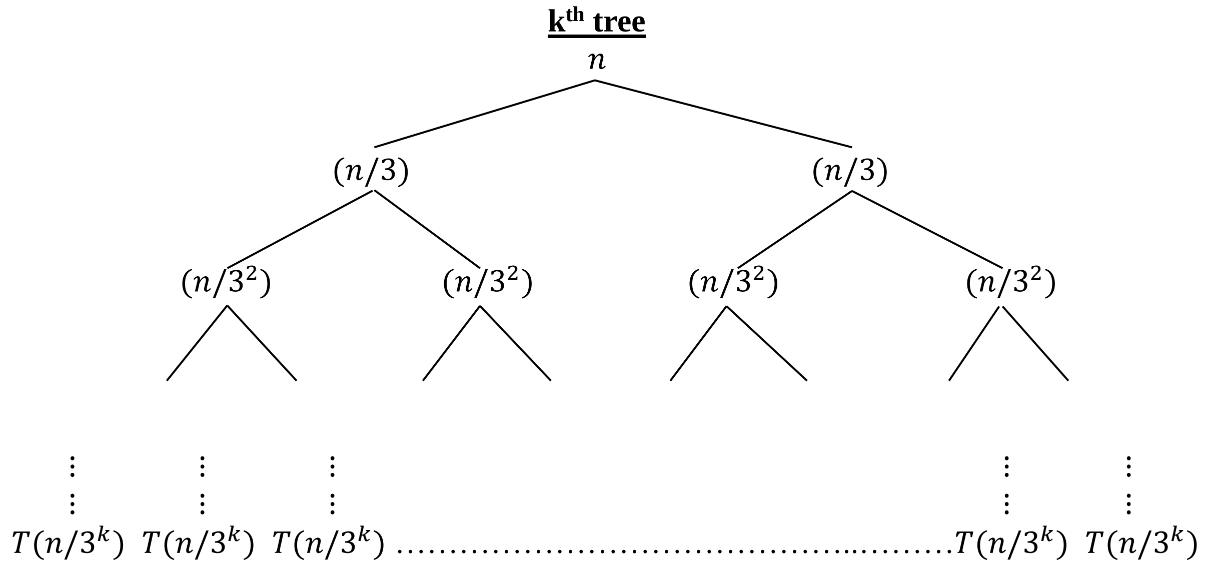
$$T(n) = \begin{cases} 1 & 1 \le n < 3 \\ 2T(\lfloor n/3 \rfloor) + n & n \ge 3 \end{cases}$$

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 $T(n/3^k)$.


kth tree
$n$
$(n/3)$
$(n/3)$
$(n/3^2)$
$(n/3^2)$
$(n/3^2)$
$(n/3^2)$
$T(n/3^k)$ $T(n/3^k)$ $T(n/3^k)$ ……………………………………..………$T(n/3^k)$ $T(n/3^k)$

Note that there are $2^i$ nodes at depth $i$, each of which has value $n/3^i$ (for $0 \leq i \leq k-1$). 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 $T(n) \leq 3n + \Theta(n^{\log_3(2)})$. 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.

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

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:

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

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 $T(n) = \sum_{i=0}^{k-1} 2^i \cdot \lfloor n/3^i \rfloor \;+\; 2^k$. 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 $T(n)$.

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

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

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

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

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 $T(n) = \Theta(n^2)$.