CS231 Algorithms
Prof. Lyn Turbak
Wellesley College

Handout # 4
February 25, 2022
Revised March 3, 2022

Recurrence Relations

- [2022/03/03] (1) In the tree diagrams for Recurrence Example 5 and Recurrence Example 7, remove the bogus $n =$ in the lower right reported by Kayla; (2) In the table of recurrences on the last page, for all situations involving $T((n-b)/k)$, changed $b > 0$ to $b \geq 0$ because as long as $k > 0$, $(n-b)/k$ will be smaller than n even if $b = 0$ (reported by Anders & Anya).
- [2022/02/27] (1) fixed the Recurrence Example 4 bug from the Fri Feb 25 lecture by changing $T(n/2)$ to $2 \cdot T(n/2)$; (2) added solutions to the exercises in the Artihmetic Series and Geometric Series sections; and (3) completed the handout by added several new sections after the Geometric Series section.

Two-Step Strategy for Analyzing Algorithms

1. Characterize running-time (space, etc.) of algorithm by a recurrence relation.
2. Solve the recurrence relation, expressing answer in asymptotic notation.

Recurrence Relations

A recurrence relation is just a recursive function definition. It defines a function at one input in terms of its value on smaller inputs.

We use recurrence relations to characterize the running time of algorithms. $T(n)$ typically stands for the running-time (usually worst-case) of a given algorithm on an input of size n.

Recurrence relations for divide-conquer-glue algorithms typically have the form:

General Case ($n \geq 1$):

$$T(n) = \textit{divide \& glue costs} + \textit{number of subproblems} \cdot T(\textit{size of subproblems})$$

(In some cases, there can be multiple recursive calls to T that have different sizes. For example, See PS2 Problem 3d.)

Base Case ($n < 1$): $T(n) = 0$. (Because this base case is always the same, it is usually omitted when defining $T(n)$.)

The solution to a recurrence relation is usually expressed in asymptotic notation.

- For $T(n) = ...$ (a recurrence **equation**), the solution is expressed in Θ notation.
- For $T(n) < ...$, the solution is expressed in O notation (i.e., it's a upper bound).
- For $T(n) > ...$, the solution is expressed in Ω notation (i.e., it's a lower bound).

Solving Recurrence Relations: The Recursion Tree Method

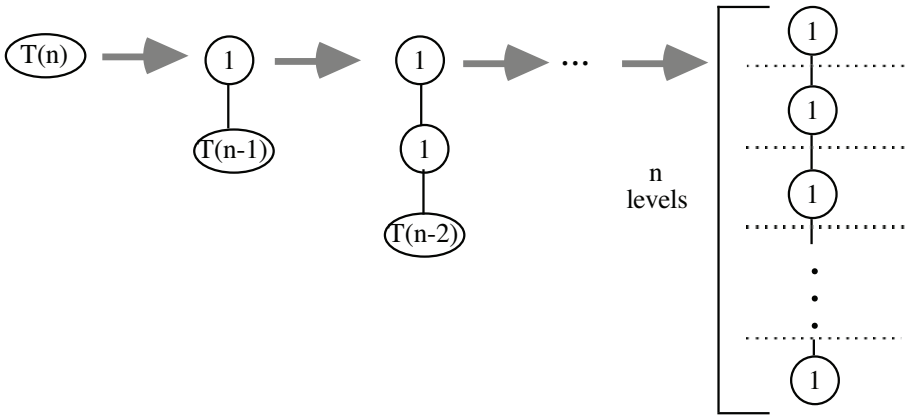
There are several methods for solving recurrence relations (see CLRS 4.3 – 4.5 for details). We will mainly use the recursion tree method. This is a two-step strategy for solving recurrence relations:

1. Construct a recursion tree by unwinding the recurrence relation.
2. Determine the cost of the entire tree by summing the costs of the nodes.

Recurrence Example 1: $T(n) = 1 + T(n-1)$

Also derivable: 2

- $T(n-1) = 1 + T(n-2)$
- $T(n-2) = 1 + T(n-3)$
- etc.



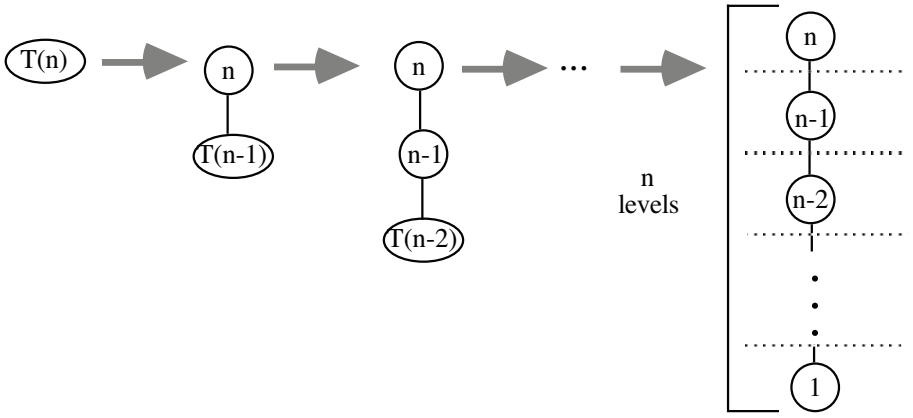
The solution to this recurrence equation is $\Theta(n)$.

Note that we simply use 1 to represent "constant costs for divide & glue", suppressing irrelevant details. This is consistent with using asymptotic notation for the solution, which also suppresses irrelevant details.

Recurrence Example 2: $T(n) = n + T(n-1)$

Also derivable:

- $T(n-1) = (n-1) + T(n-2)$
- $T(n-2) = (n-2) + T(n-3)$
- etc.

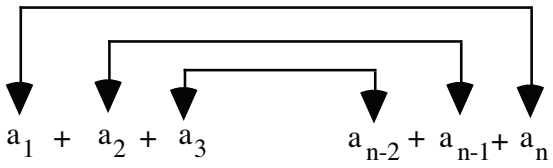


The total cost of the nodes $= 1 + 2 + 3 + ... + (n-2) + (n-1) + n = \sum_{i=1}^{n} i$ (by arithmetic series; see below). The asymptotic solution to this recurrence equation is $\Theta(n^2)$.

Arithmetic Series

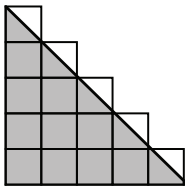
A series $a_1, a_2, a_3, \ldots$ is an **arithmetic series** iff $a_i = c + a_{i-1}$ for some constant c.

Trick for calculating this sum Sum corresponding pairs of numbers, and multiply by the number of pairs. (This works even when there are an odd number of numbers!)



$$\sum_{i=1}^{n} a_i = \frac{n}{2} \cdot (a_1 + a_n)$$

Examples



- Arithmetic Series Example 1: $1 + 2 + 3 + ... + n = \frac{n}{2} \cdot (n + 1)$

- Arithmetic Series Example 2: Sum of first n elements of series $7 + 10 + 13 + 16 + ... =$??
  Answer: 1-indexed $a_i = 4 + 3i$, so $a_n = 4 + 3n$ and sum is $\frac{n}{2}(3n + 11)$.

Examples of Recurrence Relations vs. Equation

Suppose the divide & glue costs for a single subproblem of size $(n - 1)$ is linear in some problem instances, but smaller (e.g., constant, $\lg n$, $\sqrt{n}$) in others. I.e., it's $O(n)$ but not $\Theta(n)$. Then we have an **inequality** rather than an equality:

$$T(n) < n + T(n - 1)$$

and the solution must be expressed as only an **upper bound** in big-Oh notation: $O(n^2)$

Similarly, suppose the divide & glue costs for a single subproblem of size $(n - 1)$ is linear in some problem instances, but larger (e.g., $n \cdot \lg n$, n^2) in others. I.e., it's $\Omega(n)$ but not $\Theta(n)$. Then we also have an **inequality** rather than an equality:

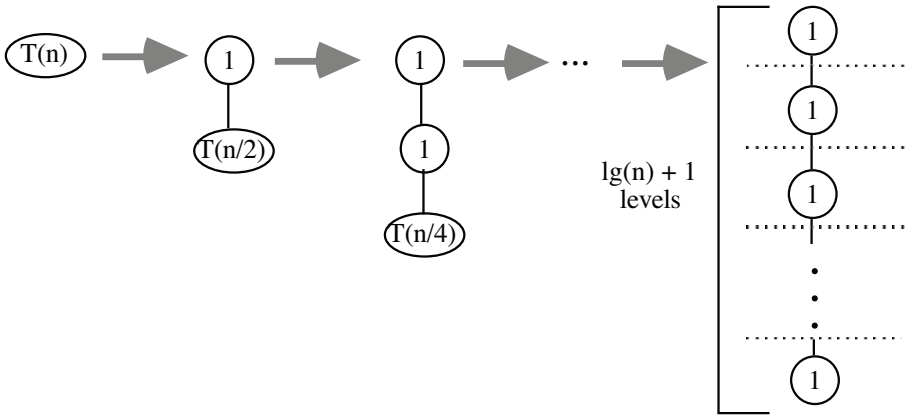
$$T(n) > n + T(n - 1)$$

and the solution must be expressed as only a **lower bound** in Ω notation: $\Omega(n^2)$

Recurrence Example 3: $T(n) = 1 + T(n/2)$

Also derivable:

- $T(n/2) = 1 + T(n/4)$
- $T(n/4) = 1 + T(n/8)$
- etc.



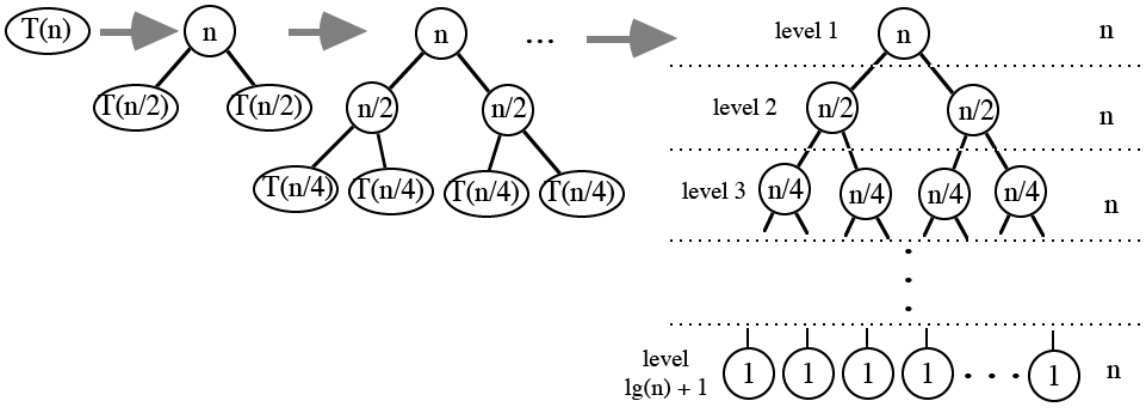
The total cost of the nodes $= (\lg n) + 1 \in \Theta(\lg n)$.

If we divide by a base b other than 2, the solution is $\Theta(\log_b(n)) = \Theta(\lg n)$.

Recurrence Example 4: $T(n) = n + 2 \cdot T(n/2)$

Also derivable:

- $T(n/2) = n/2 + 2 \cdot T(n/4)$
- $T(n/4) = n/4 + 2 \cdot T(n/8)$
- etc.



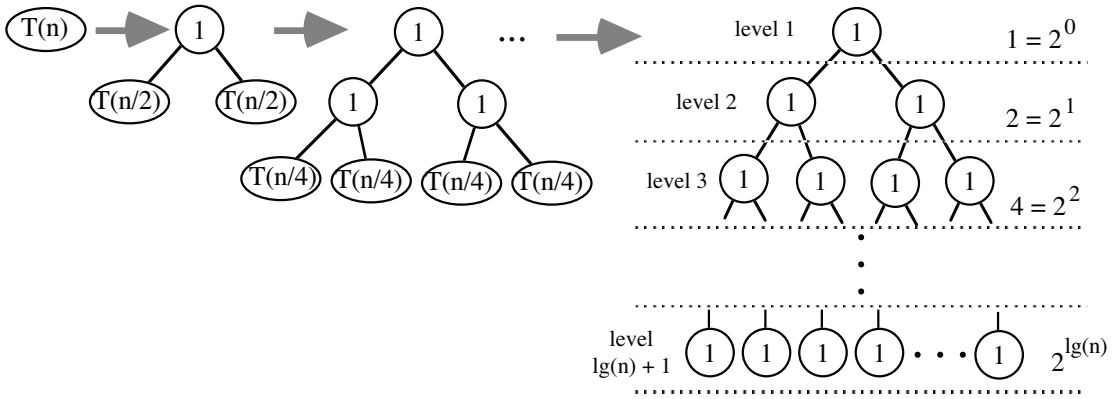
The total cost of the nodes $= n \cdot \textit{numberOfLevels} = n \cdot ((\lg n) + 1) \in \Theta(n \cdot \lg n)$.

A classic example of this recurrence equation is merge sort, which recursively sorts two subarrays that are half the size of the original, and then uses a linear time algorithm to merge the two sorted subarrays into the sorted result.

Recurrence Example 5: $T(n) = 1 + 2 \cdot T(n/2)$

Also derivable:

- $T(n/2) = 1 + 2 \cdot T(n/4)$
- $T(n/4) = 1 + 2 \cdot T(n/8)$
- etc.



The total cost of the nodes	=	the number of nodes
	=	the sum of nodes at each level
	=	$1 + 2 + 4 + 8 + ... + 2^{\lg n}$
	=	$\sum_{i=0}^{\lg n} 2^i$
	=	$2 \cdot n - 1$ (by Geometric Series Example 2; see below)
	$\in$	$\Theta(n)$

So this recurrence describes a **linear** process!

Geometric Series

A series $a_1, a_2, a_3, \ldots$ is a **geometric series** iff $a_i = c \cdot a_{i-1}$ for some constant c.

Let $S(n)$ stand for $\sum_{i=0}^{n}(a_i \cdot c^i) = a_0 + a_0 \cdot c + a_0 \cdot c^2 + a_0 \cdot c^3 + \cdots a_0 \cdot c^n$.
(Note: $S(n)$ has $n+1$ terms, not n)

Trick for calculating this sum $S(n)$.

$$\begin{array}{rcccccccccccc}
c \cdot S(n) & = & & a_0 \cdot c & + & a_0 \cdot c^2 & + & a_0 \cdot c^3 & + & \cdots & + & a_0 \cdot c^n & + & a_0 \cdot c^{n+1} \\
-S(n) & = & -a_0 & - a_0 \cdot c & - & a_0 \cdot c^2 & - & a_0 \cdot c^3 & - & \cdots & - & a_0 \cdot c^n & & \\
\hline
c \cdot S(n) - S(n) & = & -a_0 & & & & & & & & & & & a_0 \cdot c^{n+1}
\end{array}$$

So $(c-1) \cdot S(n) = a_0 \cdot c^{n+1} - a_0$. Dividing both sides by $c-1$ yields:

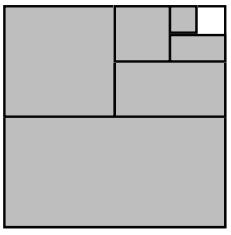
$$S(n) = \frac{a_0 \cdot (c^{n+1} - 1)}{c - 1}$$

If $0 < c < 1$ and $n \to \infty$, c^{n+1} becomes negligible and the above formula can be rewritten as:

$$\lim_{n \to \infty} S(n) = \frac{a_0}{1 - c}$$

Examples

- Geometric Series Example 1: $1 + 2 + 4 + 8 + \cdots + 2^n = \frac{1 \cdot (2^{n+1} - 1)}{2 - 1} = 2^{n+1} - 1$
- Geometric Series Example 2: $1 + 2 + 4 + 8 + \cdots + 2^{\lg n} = \frac{1 \cdot (2^{(\lg n)+1} - 1)}{2 - 1} = 2^{(\lg n)+1} - 1 = (2 \cdot 2^{\lg n}) - 1 = 2 \cdot n - 1$

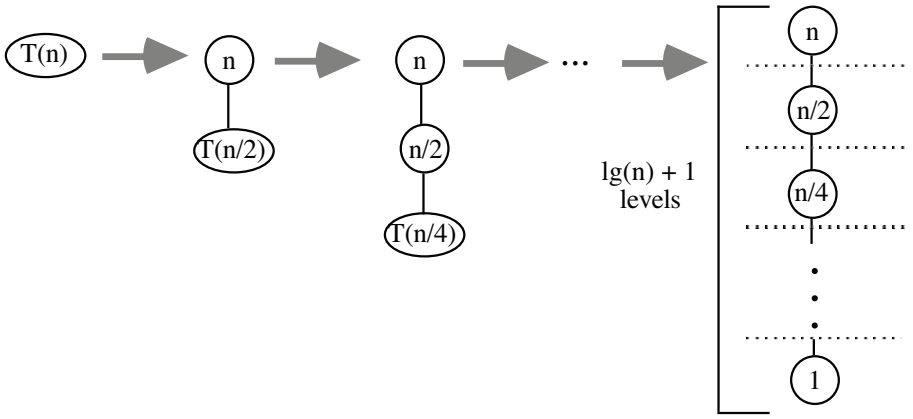


- Geometric Series Example 3: $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = \frac{\frac{1}{2}}{1 - \frac{1}{2}} = 1$
- Geometric Series Example 4: $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 2$ (i.e., 1 more than previous example).
- Geometric Series Example 5: $\frac{1}{3} + \frac{1}{9} + \frac{1}{27} + \cdots = \frac{\frac{1}{3}}{1 - \frac{1}{3}} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2}$.
  A nice way to understand this final sum intuitively is to take a piece of paper, and rip it into 3 equal-sized parts. Put two of the parts into separate piles, and rip the 3rd part into 3 equal-sized subparts. Add the two subparts to the two piles. Continue doing this. In the limit, you will end up with two piles, each of which has half of the area of the original piece of paper.

Example 6: $T(n) = n + T(n/2)$

Also derivable:

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



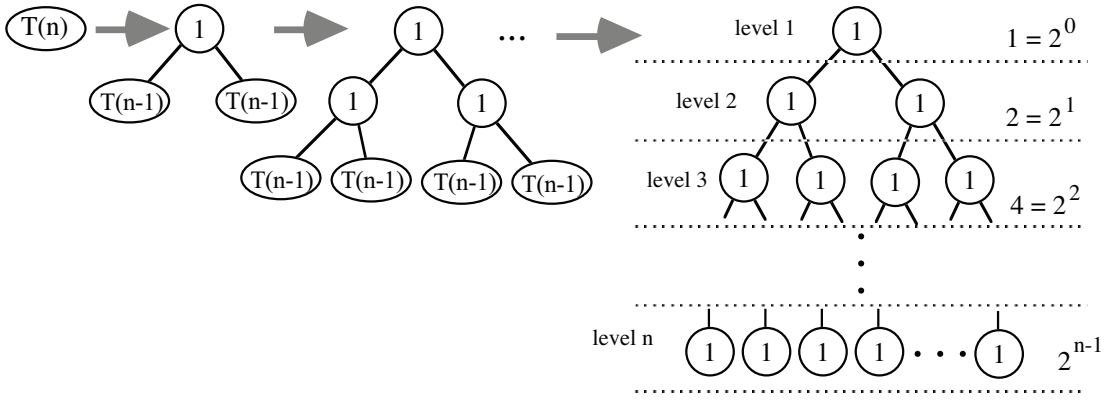
The total cost of the nodes $= n + \frac{n}{2} + \frac{n}{4} + ... + \frac{n}{2^{\lg n}} = \sum_{i=0}^{\lg n} \frac{n}{2^i} < \sum_{i=0}^{\infty} \frac{n}{2^i} = 2 \cdot n$ (by Geom. Series Ex. 4).

The asymptotic solution to this recurrence equation is $\Theta(n)$. So this is yet another recurrence equation with a **linear** solution!

Example 7: $T(n) = 1 + 2 \cdot T(n-1)$

Also derivable:

- $T(n-1) = 1 + 2 \cdot T(n-2)$
- $T(n-2) = 1 + 2 \cdot T(n-3)$
- etc.



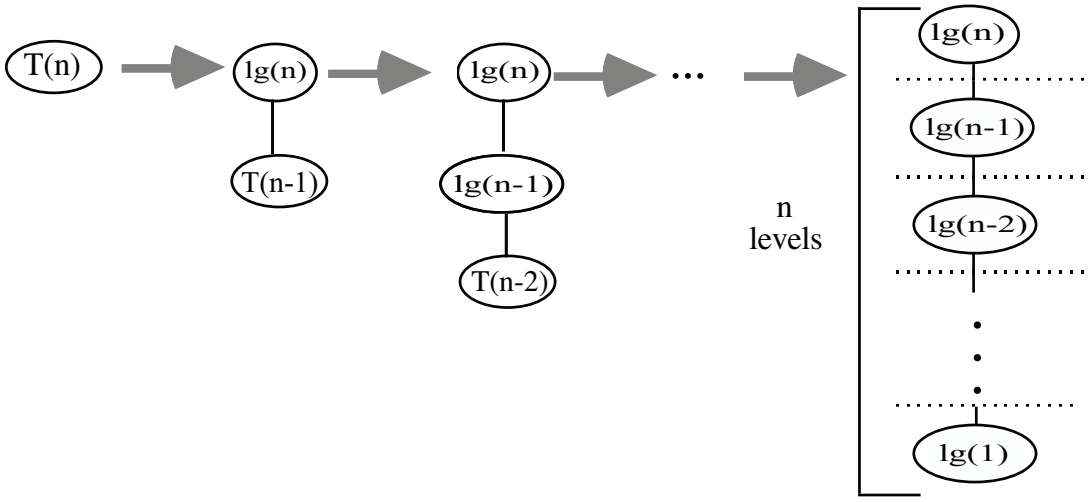
The total cost of the nodes $= 1 + 2 + 4 + 8 + \cdots + 2^{n-1} = 2^n - 1$ (by Geometric Series Example 1)

The asymptotic solution to this recurrence equation is $\Theta(2^n)$, which is **exponential**.

Example 8: $T(n) = \lg n + T(n-1)$

Also derivable:

- $T(n-1) = \lg(n-1) + T(n-2)$
- $T(n-2) = \lg(n-2) + T(n-3)$
- etc.



$$\begin{aligned}
\text{The total cost of the nodes} &= \lg n + \lg(n-1) + \lg(n-2) + \cdots + \lg 3 + \lg 2 + \lg 1 \\
&= \lg(n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1) \\
&= \lg(n!) \\
&\approx \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n), \text{ by Stirlings approximation for large } n \text{ (CLRS 3.18)} \\
&= \tfrac{1}{2}\lg(2\pi n) + n \cdot \lg \tfrac{n}{e} \\
&= \tfrac{1}{2}(\lg(2\pi) + \lg n) + n \cdot ((\lg n) - (\lg e)) \\
&\in \Theta(n \cdot \lg n), \text{ because } \lg n \in o(n \cdot \lg n) \text{ and } n \in o(n \cdot \lg n)
\end{aligned}$$

So the asymptotic solution to this recurrence equation is $\Theta(n \cdot \lg n)$.

Summary

The following table summarizes and generalizes the above examples. Equivalence classes of functions are arranged in the table from "biggest" to"smallest" by asymptotic notation using Θ. That is, if function f is an element of a set in row above another set that contains function g, then $f \in \omega(g)$ and $g \in o(f)$.

Asympototic notation	**Name**	**Typical Recurrence Equations**
$\Theta(3^n)$	exponential (base = 3)	$T(n) = a + 3 \cdot T(n-b)$, where $a > 0$, $b > 0$
$\Theta(2^n)$	exponential (base = 2)	$T(n) = a + 2 \cdot T(n-b)$, where $a > 0$, $b > 0$
$\Theta(n^2)$	quadratic	$T(n) = a \cdot n + T(n-b)$, where $a > 0$, $b > 0$
$\Theta(n \cdot \log(n))$	"n log n" (base is irrelevant)	$T(n) = a \cdot n + k \cdot T((n-b)/k)$, where $a > 0$, $b \geq 0$, $k > 1$ **or** $T(n) = a \cdot \log(n) + T(n-b)$, where $a > 0$, $b > 0$
$\Theta(n)$	linear	$T(n) = a + T(n-b)$, where $a > 0$, $b > 0$ **or** $T(n) = a + k \cdot T((n-b)/k)$ where $a > 0$, $b \geq 0$, $k > 1$ **or** $T(n) = a \cdot n + T((n-b)/k)$ where $a > 0$, $b \geq 0$, $k > 1$ **or** $T(n) = a \cdot n + T(b \cdot n) + T(c \cdot n)$ where $a > 0$ and $0 < (b + c) < 1$ (see PS2 Problem 3d)
$\Theta(\log(n))$	logarithmic (base is irrelevant)	$T(n) = a + T(n/k)$, where $a > 0$, $k > 1$
$\Theta(1)$	constant	$T(n) = a$, where $a > 0$

There are many more recurrence equations and equivalence classes than are listed in the above table, but the ones in the table are the ones most commonly encountered in this course.

The above table is useful not only for **analyzing** given algorithms, but also for **designing** algorithms with a desired runtime or space usage. Note that there are **four** recurrence equations that have $\Theta(n)$ (linear) solutions and **two** recurrence equations that have $\Theta(n \cdot \log(n))$ solutions and, which indicates that there are multiple divide/conquer/glue strategies that can be considered for those asymptotic classes.