Case Analysis, Asymptotic Analysis, and Clarity

What is Case Analysis?

Ultimately, Case Analysis is an assessment of the scenarios that an algorithm (or a data structure) will encounter. Another way to think of this is what *kind of input* might this algorithm have to deal with, or how might a data structure's "shape" impact the operation we want to perform?
For example, what if we're searching for a specific value in a Binary Search Tree?

- **Best Case:** the scenario that we expect the algorithm to do really well in, or the *shape* that we expect to result in our best performance for our data structure.
- **Worst Case:** the scenario that we expect the algorithm to do poorly in, or the *shape* that we expect to result in the worst performance for our data structure.
- **Overall Case:** no matter what scenario we encounter, how should we expect the algorithm to do? Or, no matter what *shape* our data structure has, how long should we expect an operation to take?
  - Note that, no matter what, we know that whatever we're trying to do won't have any worse performance than worst we can expect in our Worst Case and it won't be any better than the best we can expect from our Best Case.

We can assess algorithms on their own, or data structures on their own, or we can talk about them in conjunction with each other. For our find operation on our BST, for example:

- Algorithm Cases:
  - Best Case: value is in root
  - Worst Case: value isn't in the tree
- Data Structure Cases:
  - Best Case: Tree is "bushy" and has the smallest height it could have
  - Worst Case: Tree is a linked list and has the greatest height it could have

What is Asymptotic Analysis then?

Asymptotic Analysis is a way to determine the order of growth for this particular algorithm *for a particular case*. How do we categorize this algorithm's efficiency in relation to the size of the input?

- **Big-O:** *in this case* what's the worst growth this algorithm could have?
- **Big-Omega:** *in this case* what's the best growth this algorithm could have?
- **Big-Theta:** *in this case* can I always expect the order of growth to be about the same? In other words, in order for Big-Theta to exist for a given case, we expect the order of growth to be the same across all inputs.

This idea is probably most easily shown in a table, so let's look again at the example of searching for a value in a BST:

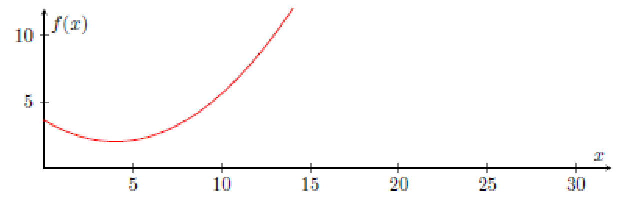
Algorithm Case	Data Structure Case	Big-O	Big-Omega	Big-Theta
Best: value in root	Best: "bushy" tree	O(1)	Ω(1)	Θ(1)
Best: value in root	Worst: "list" tree	O(1)	Ω(1)	Θ(1)
Worst: value not in tree	Best: "bushy" tree	O(log(N))	Ω(log(N))	Θ(log(N))
Worst: value not in tree	Worst: "list" tree	O(N)	Ω(N)	Θ(N)

So what about Overall? Remember that Overall is a combination of the Best and Worst cases–in fact, it will have the same Big-O as the Best Case, and the same Big-Omega as the Worst Case:

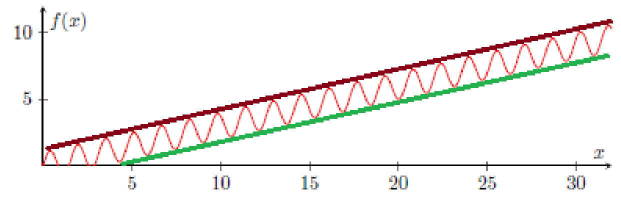
	Algorithm Overall	Data Structure Overall	Combined Overall
Big-O	O(N)	O(N)	O(N)
Big-Omega	Ω(1)	Ω(1)	Ω(1)
Big-Theta	DNE	DNE	DNE

Visual Examples from Hannah's Slides (Explained)

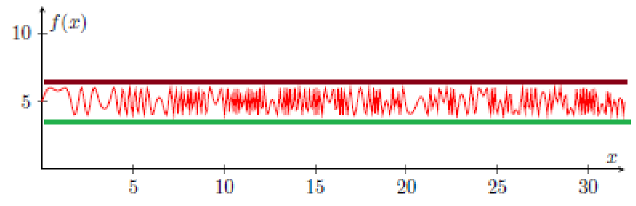
Let's assume these functions show different algorithms' behavior in the Worst Case (which is our default assumption if no case is specified):



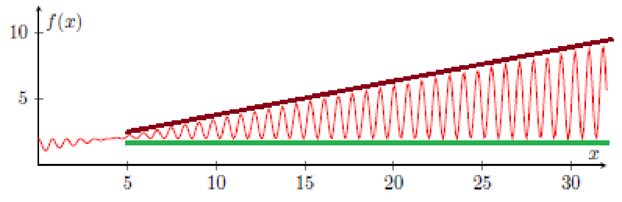
This function pretty clearly has a Big-Theta, since there's just a single line: $\mathbf{\Theta}(N^2)$



Big-O is represented by the darker line above the function, and Big-Omega is represented by the green line below the function. Since both lines grow at the same rate, this function has a Big-Theta: **Θ(N)**



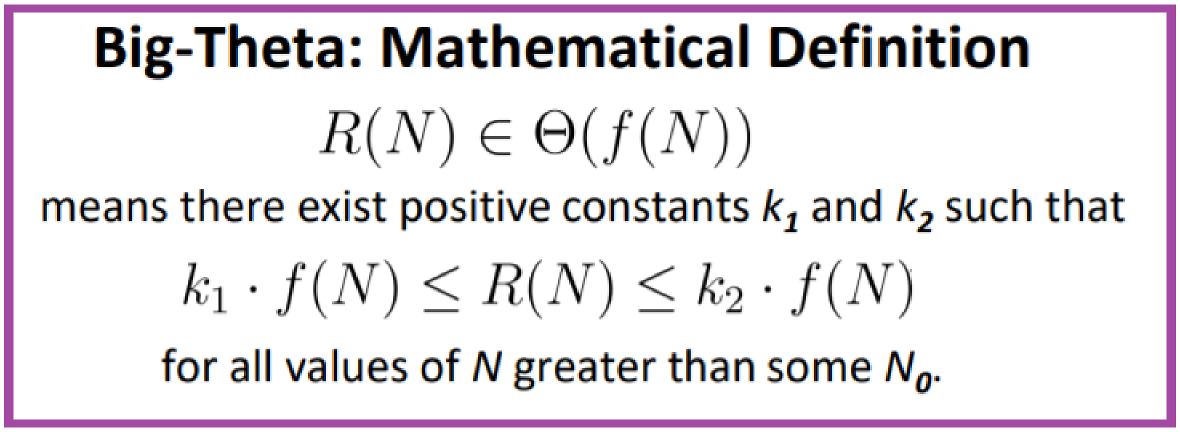
This function has the same situation as the one just above. The Big-O and Big-Omega lines have the same growth rate, so Big-Theta exists for this function: **Θ(1)**



This function is different! Notice that the Big-O line increases linearly but the Big-Omega line stays constant, so this function would be O(N) and Ω(1) in this case, and it doesnt have a Big-Theta.

In other words, theres no single function that we can use to describe the bounds of this function.

Hopefully this definition from lecture makes a little more sense now:



Big-Theta: Mathematical Definition

$$R(N) \in \Theta(f(N))$$

means there exist positive constants k_1 and k_2 such that

$$k_1 \cdot f(N) \leq R(N) \leq k_2 \cdot f(N)$$

for all values of N greater than some N_0.

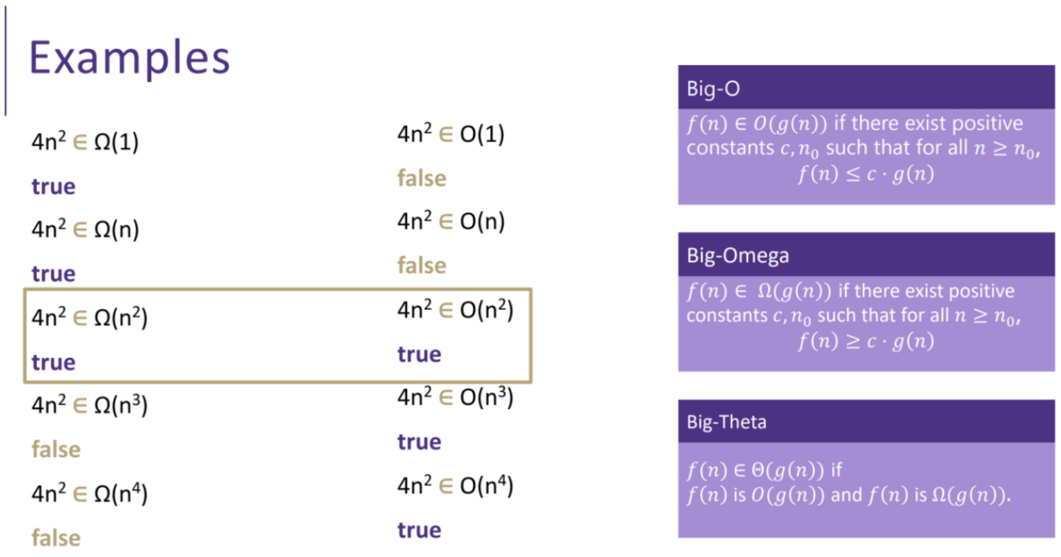
Still not sure about the formal definition for Big-Theta?

Suppose we use the last example from above–the one in which Big-Theta doesn't exist. If you follow the link below, it will take you to a demo with this same function, as well as the functions for Ω(1) and O(N). There are some sliders for the k values (the constant multipliers)for you to experiment with. Hopefully this helps give you an intuitive understanding of why Big-Theta doesn't exist for this function!

Slider Demo: Visual Example

The Formal Definitions (And Another Example)

The example (below) on the right looks at the function $4n^2$ and explores possible functions for Big-O and Big-Omega. If the proposed function meets the requirements for the definition of Big-Omega (in the right column) or Big-O (in the left column) then **true** can be seen underneath. Note that the gold box is drawn around the case where the proposed function meets the requirements for *both* Big-O and Big-Omega, which means that it also meets the requirements for Big-Theta.



Examples

$4n^2 \in \Omega(1)$	**true**	$4n^2 \in O(1)$	**false**
$4n^2 \in \Omega(n)$	**true**	$4n^2 \in O(n)$	**false**
$4n^2 \in \Omega(n^2)$	**true**	$4n^2 \in O(n^2)$	**true**
$4n^2 \in \Omega(n^3)$	**false**	$4n^2 \in O(n^3)$	**true**
$4n^2 \in \Omega(n^4)$	**false**	$4n^2 \in O(n^4)$	**true**

Big-O

$f(n) \in O(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$,
$$f(n) \leq c \cdot g(n)$$

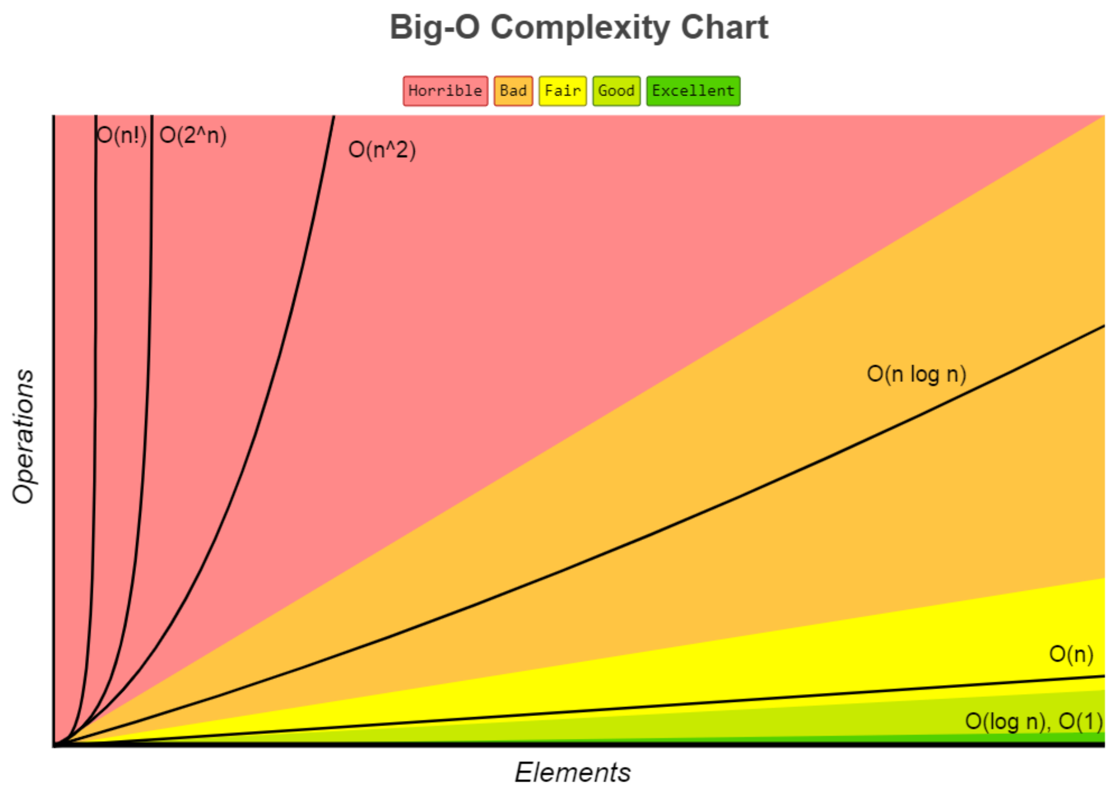
Big-Omega

$f(n) \in \Omega(g(n))$ if there exist positive constants c, n_0 such that for all $n \geq n_0$,
$$f(n) \geq c \cdot g(n)$$

Big-Theta

$f(n) \in \Theta(g(n))$ if
$f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

Visual for Comparing Orders of Growth



https://www.bigocheatsheet.com/
(Consider visiting the site where we got this image–it has lots of visuals and neat stuff!)