4 Asymptotic Analysis

You've almost certainly seen Big-O notation in a previous CS course. But you may not be familiar with the formal definitions of Big-O or its relatives Big-Ω, Big-Θ, little-o, and little-ω, and you probably don't have a lot of experience using these definitions in proofs.

In Section 4.1 we formally define Big-O notation and give some examples of using it to prove one function is Big-O of another. In Section 4.2 we will introduce related notation and show how to apply it. Finally in Section 4.3 we'll discuss some useful properties of Big-O and see some examples of more abstract proofs with these definitions.

4.1 Big-O

While you're probably familiar with using Big-O to describe the runtime of algorithms, it's actually a way to compare the growth rates of arbitrary functions. Informally, for functions f and g we say that one function $f(n) \in O(g(n))$ if $f(n)$ is upper bounded by a constant multiple of $g(n)$ for all large enough n.

Formally, for a function $g : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$ we define

$$O(g(n)) := \{f : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0} \mid \exists\, c \in \mathbb{R}^+, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, f(n) \leq cg(n)\}.$$

(Note that in a slight abuse of notation, we use "$\forall\, n \geq n_0$" to mean "$\forall\, n \in \mathbb{Z}$ such that $n \geq n_0$". The fact that n is an integer is implied by the fact that f is defined as a function on integers)

That is, $O(g(n))$ is the set of all functions f for which we can choose positive constants c and n_0 such that for every integer $n \geq n_0$ (i.e. every sufficiently large input), $f(n) \leq cg(n)$ ($f(n)$ is upper bounded by a constant multiple of $g(n)$). You can think of $O(g(n))$ as the set of all functions whose "dominant term" is less than or equal to $g(n)$'s (up to constant factors). For example, the set $O(2n^2 - 1)$ includes the following functions (among infinitely many others):

- $2n^2 - 1$ (we can choose $c = 1$ and $n_0 = 1$)
- $3n^2$ (we can choose $c = 3$ and $n_0 = 3$)
- $1000n$ (we can choose $c = 1000$ and $n_0 = 1$)

To prove that $f(n) \in O(g(n))$ for some specific functions f and g, we need to apply the definition. We need to show that f satisfies the condition for membership in $O(g(n))$, i.e. that $\exists\, c \in \mathbb{R}^+, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, f(n) \leq cg(n)$. As we saw in Section 2.3, to prove a "there exists" statement, we need to find specific values for the variables which make the statement true. In this case that means that we need to give a specific c and n_0 such that $f(n) \leq cg(n)$ for every $n \geq n_0$ (and then justify that the c and n_0 we chose actually do satisfy this predicate).

Let's see a couple examples.

Theorem 19.

$$3n^3 + 5n^2 - n + 14 \in O\left(\frac{n^3}{2}\right)$$

Proof. We need to choose positive constants c and n_0 such that $\forall\, n \geq n_0$, $3n^3 + 5n^2 - n + 14 \leq cn^3/2$. Let $c = 44$ and $n_0 = 1$. For every $n \geq 1$, we have $n^3 \geq n^2 \geq n \geq 1$, so

$$3n^3 + 5n^2 - n + 14 \leq 3n^3 + 5n^3 + 14n^3 = 22n^3 = 44 \cdot \frac{n^3}{2}.$$

□

Note that we don't need to choose the smallest possible value of c. Rather, pick whichever value of c makes the proof easiest. In the previous proof we chose $c = 44$, because it was twice the sum of the non-negative coefficients in $3n^3 + 5n^2 - n + 14$, allowing us to upper bound each term in the polynomial with a multiple of n^3. We also didn't need to explain why we chose this c in the

proof. Remember that the purpose of a proof is to convince your reader that a claim is true, not to explain how you discovered that the claim is true. Therefore a Big-O proof only needs to show that particular values of c and n_0 work, not explain how you came up with them.

Sometimes there isn't a natural inequality that we can use to directly prove the claim we want. In these cases, we may need to use induction.

Theorem 20.

$$2^n \in O(n!)$$

Proof. Let $n_0 = 1$ and $c = 2$. We will prove by induction on n that $2^n \leq 2 \cdot n!$ for every $n \geq 1$.

Base Case: As a base case, suppose $n = 1$. Then

$$2^n = 2 = 2 \cdot 1 = 2 \cdot 1! = 2 \cdot n!$$

Induction Hypothesis: Suppose that for all positive integers $k < n$, $2^k \leq 2 \cdot k!$.

Induction Step: Suppose $n \geq 2$. Applying the induction hypothesis to $n - 1$, we have that

$$\begin{aligned}
2^n &= 2 \cdot 2^{n-1} \\
&\leq 2 \cdot 2 \cdot (n-1)! && \text{(by our IH)} \\
&\leq 2 \cdot n \cdot (n-1)! && \text{(since } n \geq 2\text{)} \\
&= 2 \cdot n! && \text{(by the definition of factorial)}
\end{aligned}$$

□

In both of the previous examples, we were able to choose $n_0 = 1$ as long as we picked a large enough c. This almost always works, however if $g(n)$ sometimes outputs 0, then we usually need to pick n_0 large enough to guarantee that $g(n) > 0$ for all $n \geq n_0$. Note that $g(n)$ cannot output negative values (for positive integer inputs) because we've defined big-O only for functions with co-domain $\mathbb{R}_{\geq 0}$ (it's possible to extend the definition to functions with negative outputs, but those functions aren't very relevant in this course, since it's generally impossible for an algorithm to use a negative amount of resources).

Theorem 21.

$$\log_2(2n + 2) \in O(\log_3(n))$$

Proof. Let $n_0 = 2$ and $c = 6$. Then we have that for every $n \geq n_0$,

$$\begin{aligned}
\log_2(2n+2) &\leq \log_2(4n) && \text{(because } n \geq 1\text{)} \\
&= \log_2(4) + \log_2(n) && \text{(by Theorem 2)} \\
&= \log_2(n) + 2 \\
&\leq 3\log_2(n) && \text{(because } n \geq 2, \text{ so } \log_2(n) \geq 1\text{)} \\
&= 3\log_2(3)\log_3(n) && \text{(by Theorem 2)} \\
&\leq 3\log_2(4)\log_3(n) \\
&= 6\log_3(n)
\end{aligned}$$

□

Note that we couldn't choose $n_0 = 1$ here, because $\log_2(2 \cdot 1 + 2) = 2$, while $\log_3(1) = 0$, and 2 is not less than $0 \cdot c$ for any value of c.

Let's look at a trickier big-O proof.

Theorem 22.

$$\log_2^2 n \in O(n)$$

Proof. Let $n_0 = 1$ and $c = 49$. We will prove that $\log_2^2 n \leq 49n$ for all $n \in \mathbb{Z}^+$ by induction on n.

Base Cases: If $1 \leq n < 128$ then

$$\log_2^2 n < \log_2^2 128 = 7^2 = 49 = 49 \cdot 1 \leq 49 \cdot n.$$

Induction Hypothesis: Suppose that for all positive integers $k < n$, $\log_2^2 k \leq 49k$.

Induction Step: Let $n \geq 128$. Then

$$\begin{aligned}
\log_2^2(n) &= (\log_2(n/2) + 1)^2 && \text{(by Theorem 2)} \\
&= \log_2^2(n/2) + 2\log_2(n/2) + 1 \\
&\leq \log_2^2(n/2) + 3\log_2(n/2) && \text{(since } n \geq 2 \text{ so } \log_2 n \geq 1\text{)} \\
&= \left(1 + \frac{3}{\log_2(n/2)}\right) \log_2^2(n/2) \\
&\leq (3/2)\log_2^2(n/2) && \text{(because } n > 128 \Rightarrow \log_2(n/2) \geq 6 \Rightarrow 3/\log_2(n/2) \leq 1/2\text{)} \\
&\leq (3/2)\log_2^2(\lceil n/2 \rceil) \\
&\leq (3/2)(49)\lceil n/2 \rceil && \text{(by our IH)} \\
&\leq 49 \cdot \frac{3}{2} \cdot \frac{n+1}{2} \\
&= 49 \cdot \frac{3n+3}{4} \\
&\leq 49n && \text{(because } n \geq 3\text{)}
\end{aligned}$$

Thus, $\log_2^2 n \leq 49n$, as desired. □

There are several things to note about this proof. The choice of $c = 49$ and the base case of $n < 128$ may seem like arbitrary magic numbers, but they actually come from thinking about the structure of our induction proof.

The starting point for developing this proof was the logic of the induction step. We need to relate $\log_2^2(n)$ to $\log_2^2(k)$ for some $k < n$. $\log_2(n)$ doesn't have a clear relationship with $\log_2(n-1)$, but we do know a nice relationship between $\log_2(n)$ and $\log_2(n/2)$ (namely $\log_2(n) = \log_2(n/2) + 1$). Unfortunately, we can't necessarily apply our induction hypothesis to $n/2$, because $n/2$ may not be an integer. That's why before applying the IH we need to upper bound $n/2$ with $\lceil n/2 \rceil$, which is an integer.

If we can say that $\log_2^2(n) \leq (1+\varepsilon)\log_2^2(\lceil n/2 \rceil)$, for some $\varepsilon > 0$ then we can apply the IH to get that

$$\log_2^2(n) \leq (1+\varepsilon)c\lceil n/2 \rceil \leq (1+\varepsilon)c \cdot \frac{n+1}{2}.$$

In order to upper bound this with cn, we need $\varepsilon < 1$ and n sufficiently large. We could pick any value $0 < \varepsilon < 1$ and calculate a corresponding value of c that would make the proof work, but $\varepsilon = 1/2$ is an easy choice. In order to upper bound $3/\log_2(n/2)$ with $1/2$, we need $n \geq 128$, which is why we made $1 \leq n < 128$ our base case.

Assuming $n \geq 128$, the calculations in our induction step actually go through for any choice of c. The reason we chose $c = 49$ was because $49 = \log_2^2 128$, and this makes the proof of the base case simple.

Note that we could not have chosen $n_0 = 128$ using this same proof setup. In order to assume the induction hypothesis at $n = 128$, we need the base case to have proved our claim for all $1 \leq k < 128$. In particular, our proof uses the fact that the inequality holds for $n = 64$ to prove that it holds for $n = 128$, which would be a problem if we didn't have base case covering $n = 64$ (we could validly prove $\log_2^2(n) \in O(n)$ using $n_0 = 64$ and $c = 1$ by tweaking this proof slightly, I encourage you to think about how).

One final note on notation. Often you will see $f(n) \in O(g(n))$ written as $f(n) = O(g(n))$. This notation is slightly misleading, since, unlike equality, the big-O relationship is not commutative (e.g.

$n = O(n^2)$, but $n^2 \neq O(n)$), but it's very widespread, so you should understand it (As computer scientists, you should already be used to the idea of a one-way equals, since many programming languages use = for assignment. For example, in C++, "x = 5" is valid code whereas "5 = x" is not).

4.2 Other Asymptotic Comparisons

Big-O functions sort of like a $\leq$ relation for functions. $O(g(n))$ is the set of all functions which are eventually less than or equal to $g(n)$, up to constant factors. It's natural to also define a corresponding notion of $\geq$ for functions. This is called Big-Ω (read "Big Omega"). Similarly we have a version of "=", for functions. We use Big-Θ (read "Big Theta") to denote the set of functions which grow at the same rate as $g(n)$ up to constant factors.

Formally, for a function $g : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$ we define

$$\Omega(g(n)) := \{f : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0} \mid \exists\, c \in \mathbb{R}^+, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, f(n) \geq cg(n)\}$$

(this is just the big-O definition with $f(n) \leq cg(n)$ changed to $f(n) \geq cg(n)$), and

$$\Theta(g(n)) := \{f : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0} \mid \exists\, c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

(here we need f to be bounded on both sides by constant multiples of g).

Note that the definition of big-Ω immediately implies that if $f(n) \in O(g(n))$, then $g(n) \in \Omega(f(n))$, and vice-versa (we can use the same values for c and n_0 in both definitions).

Theorem 23.

$$\forall\, a, b \in \mathbb{Z}^+ \text{ such that } a, b \geq 2, \log_a(n) \in \Theta(\log_b n)$$

Proof. Let a and b be positive integers greater than or equal to 2. Let $c_1 = c_2 = \log_a b$, and let $n_0 = 2$. Then, by Theorem 2, $\log_a n = c_1 \log_b n = c_2 \log_b n$, so for all $n \geq n_0$, we have

$$c_1 \log_b n \leq \log_a n \leq c_2 \log_b(n),$$

so $\log_a(n) \in \Theta(\log_b n)$ □

By Theorem 23, the base of a logarithm doesn't really matter from an asymptotic perspective (it only changes the value of the function by a constant factor), so when dealing with logarithms in Big-O and related notations, we often omit the base and simply write $O(\log n)$. Do be careful with this, however. If a logarithm shows up inside of an exponent, the base is relevant again, $O(\log_2 n)$ is the same as $O(\log_4 n)$, but $O(4^{\log_4 n})$ is the same as $O(n)$, while $O(4^{\log_2 n})$ is the same set as $O(n^2)$.

Finally, we have notions of $<$ and $>$ for functions, which we call little-o and little-ω ("little omega"). For a function $g : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$ we define

$$o(g(n)) := \{f : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0} \mid \forall\, c \in \mathbb{R}^+, \exists\, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, f(n) \leq cg(n)\}$$

to be the functions that are "less than" $g(n)$, asymptotically. Note that we have changed the "$\exists\, c$" from the Big-O definition into a "$\forall\, c$". This definition is saying that for any possible choice of c, no matter how tiny, $f(n)$ will eventually become smaller than $cg(n)$ (and stay smaller for all large enough n).

Similarly, for a function $g : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$ we define

$$\omega(g(n)) := \{f : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0} \mid \forall\, c \in \mathbb{R}^+, \exists\, n_0 \in \mathbb{Z}^+, \forall\, n \geq n_0, f(n) \geq cg(n)\}$$

to be the functions which grow faster than $g(n)$ by more than a constant factor.

Let's see an example of using these new definitions:

Theorem 24.

$$\frac{n^3}{10} - 3n^2 + 1000 \in \omega(n^2)$$

Proof. We need to show that for every possible choice of c, there exists some n_0 such that for every $n \geq n_0$, $\frac{n^3}{10} - 3n^2 + 1000 \geq cn^2$.

Let c be an arbitrary positive real number. Choose $n_0 = \lceil 10(c+3) \rceil$, so that we know n_0 is an integer with $n_0 \geq 10(c+3)$. Then we have

$$n^3/10 - 3n^2 + 1000 > n^3/10 - 3n^2 = (n/10 - 3)n^2 \geq (n_0/10 - 3)n^2 \geq ((10(c+3))/10 - 3)n^2 = cn^2.$$

□

4.3 Properties of Big-O and Friends

In this section we'll show a variety of useful properties that will allow us to express the asymptotic behavior of new functions using the behavior of functions we've already seen.

Theorem 25. *Let $f, g : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$. Then $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.*

Proof. To prove an "if and only if" statement, we need to show both directions.

First we prove the forward direction ($f(n) \in \Theta(g(n)) \Rightarrow f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$). Suppose $f(n) \in \Theta(g(n))$. Let $c_1, c_2 \in \mathbb{R}^+$, $n_0 \in \mathbb{Z}^+$ be constants such that $\forall\, n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$ (these exist by the definition of big-Θ). Then, by the definition of big-O, using the constants c_2 and n_0, $f(n) \in O(g(n))$, and by the definition of big-Ω, using the constants c_1 and n_0, $f(n) \in \Omega(g(n))$.

Now we prove the back direction ($f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n)) \Rightarrow f(n) \in \Theta(g(n))$). Suppose $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$. Let $c_1 \in \mathbb{R}^+$, $n_1 \in \mathbb{Z}^+$ be constants such that $\forall\, n \geq n_1, f(n) \geq c_1 g(n)$ (these constants exist by the definition of big-Ω), and let $c_2 \in \mathbb{R}^+$, $n_2 \in \mathbb{Z}^+$ be constants such that $\forall\, n \geq n_2, f(n) \leq c_2 g(n)$ (these constants exist by the definition of big-O). Then, if we let $n_0 = \max(n_1, n_2)$ (so that $n \geq n_0 \Rightarrow n \geq n_1$ and $n \geq n_2$) we have that $\forall\, n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$, so $f(n) \in \Theta(g(n))$. □

Theorem 26. *If $f_1, f_2, g_1, g_2 : \mathbb{Z}^+ \to \mathbb{R}_{\geq 0}$ are functions such that $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then:*

(a) $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$

(b) $f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$

Proof. (a) Let $c_1, c_2 \in \mathbb{R}^+$, $n_1, n_2 \in \mathbb{Z}^+$ be constants such that $\forall\, n \geq n_1$, $f_1(n) \leq c_1 g_1(n)$ and $\forall\, n \geq n_2$, $f_2(n) \leq c_2 g_2(n)$ (such constants must exist by the definition of big-O). To show that $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$ we choose the constants $c = c_1 + c_2$ and $n_0 = \max(n_1, n_2)$. Then, by our choice of c_1, c_2, n_1, and n_2, we have that for every $n \geq n_0$

$$f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \leq (c_1 + c_2) g_1(n) + (c_1 + c_2) g_2(n) = (c_1 + c_2)(g_1(n) + g_2(n)).$$

So by the definition of big-O, $f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$.

(b) Let $c_1, c_2 \in \mathbb{R}^+$, $n_1, n_2 \in \mathbb{Z}^+$ be constants such that $\forall\, n \geq n_1$, $f_1(n) \leq c_1 g_1(n)$ and $\forall\, n \geq n_2$, $f_2(n) \leq c_2 g_2(n)$ (such constants must exist by the definition of big-O). To show that $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$ we choose the constants $c = c_1 \cdot c_2$ and $n_0 = \max(n_1, n_2)$. Then, by our choice of c_1, c_2, n_1, and n_2, we have that for every $n \geq n_0$

$$f_1(n) \cdot f_2(n) \leq c_1 g_1(n) \cdot c_2 g_2(n) = c(g_1(n) \cdot g_2(n)).$$

So by the definition of big-O, $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$.

□

If a problem specifically asks you to show that one function is big-O (or big-Ω, etc.) of another using the formal definition (like in the exercises for this chapter), then you should make sure to give an explicit choice of c and n_0 and justify why they work. When using Big-O to describe the runtime of your algorithms, however, you do not need to go into the full details of a big-O proof. If you've concluded that the runtime is $O(n^2) + O(n)$, you can and should say that this is $O(n^2)$ without further proof.

4.4 Comprehension Questions

1. Suppose that we want to prove that $3n^2 + 10n + 2 \in O(n^2)$ using the formal definition of big-O. Which of the following values for c and n_0 could we choose? (select only one)

   (a) $c = 10, n_0 = 1$
   (b) $c = 7, n_0 = 3$
   (c) $c = 5, n_0 = 4$
   (d) $c = 3, n_0 = 100$

2. Suppose that we want to prove that $n! \in O(n^n - 1)$ using the formal definition of big-O. Which of the following values for c and n_0 could we choose? (select only one)

   (a) $c = 0.5, n_0 = 3$
   (b) $c = 0.6, n_0 = 2$
   (c) $c = 1, n_0 = 1$
   (d) $c = 500, n_0 = 1$

3. Let f and g be functions from $\mathbb{Z}^+$ to $\mathbb{R}_{\geq 0}$ defined by $f(n) = n^2(n \bmod 2)$ and $g(n) = n^2(n \bmod 3)$. Which of the following is true? (select all that apply)

   (a) $f(n) \in o(g(n))$
   (b) $f(n) \in \Theta(g(n))$
   (c) $f(n) \in \omega(g(n))$
   (d) None of the other answers are true

4. Let f and g be functions from $\mathbb{Z}^+$ to $\mathbb{R}_{\geq 0}$ defined by $f(n) = 3n^3 + 10n$ and $g(n) = 10n^3 + 5$. Which of the following is true? (select all that apply)

   (a) $f(n) \in o(g(n))$
   (b) $f(n) \in \Theta(g(n))$
   (c) $f(n) \in \omega(g(n))$
   (d) None of the other answers are true

5. If $f(n) \in \omega(g(n))$, which of the following must also be true? (select all that apply)

   (a) $f(n) \in O(g(n))$
   (b) $f(n) \in \Omega(g(n))$
   (c) $f(n) \in \Theta(g(n))$
   (d) $f(n) \notin O(g(n))$
   (e) $f(n) \notin \Omega(g(n))$
   (f) $f(n) \notin \Theta(g(n))$

4.5 Exercises

1. Prove that $3n^5 + 4n^3 + 12 \in O(n^5)$ using the formal definition of big-O.

2. Prove that $n^3 - 10n - 99 \in \Omega(n^2)$ using the formal definition of big-Ω

3. Let $f, g : \mathbb{Z}^+ \to \mathbb{R}^+$ be functions which output only positive values, and suppose that $f(n) = O(g(n))$. Prove that there exists a constant $c \in \mathbb{R}^+$ such that for all $n \geq 1$, $0 \leq f(n) \leq cg(n)$ (i.e. show that for functions from $\mathbb{Z}^+$ to $\mathbb{R}^+$, we can always choose $n_0 = 1$ in the definition of big-O).

4. (a) Let f_1, f_2, f be functions from $\mathbb{Z}^+$ to $\mathbb{R}_{\geq 0}$ such that $f_1(n) = \Theta(f(n))$ and $f_2(n) = \Theta(f(n))$. Prove that $f_1(n) + f_2(n) = \Theta(f(n))$

   (b) Let f be a function from $\mathbb{Z}^+$ to $\mathbb{R}_{\geq 0}$ and let $f_1, f_2, f_3 \ldots$ be an infinite sequence of functions from $\mathbb{Z}^+$ to $\mathbb{R}_{\geq 0}$ such that $\forall\, i \in \mathbb{Z}^+, \quad f_i(n) = \Theta(f(n))$. Define a new function $g(n) := \sum_{i=1}^{n} f_i(n)$. Is it necessarily true that $g(n) = \Theta(nf(n))$? Determine whether or not this is true, and give either a proof or a counterexample.