Readings

- [Lecture Notes Chapter 5: Running Time and Growth Functions]

Review: Runtime Analysis

When analyzing algorithms, we can analyze the best case, average case, and worst case running times:

**Best Case Analysis:** This is when we analyze the runtime of the algorithm on the set of inputs in which it performs the fastest. This isn’t super useful because algorithms can be modified to make the best case performance trivial by hardcoding the solution/what to return for a specific input. In these situations, the best case performance is effectively meaningless.

**Worst Case Analysis:** This is when we analyze the runtime of the algorithm on the set of inputs with which it performs the slowest. This is useful because the worst case running time of an algorithm gives an upper bound on the running time for any input. In other words, the worst case running time provides a guarantee that the algorithm can never take any longer. Thus, unless otherwise specified, in CIS 121, we ask you to perform worst case analysis since it is the cleanest method of analysis.

**Average Case Analysis:** This is when we analyze the runtime of the algorithm on the “average” input. This doesn’t surface too much, as what constitutes an “average” input is usually not given to us and finding the “average” input requires a probability distribution of all possible inputs to the algorithm.

Problems

**Problem 1: True or False**

1. A Big-O, Big-Θ, Big-Ω bound for an algorithm correspond to its worst-case, average-case, and best-case runtime, respectively.
2. For any two functions, \( f(n) \) and \( g(n) \), either \( f(n) \in O(g(n)) \) or \( g(n) \in O(f(n)) \).
3. \( f(n) \in O(g(n)) \) if and only if \( g(n) \in \Omega(f(n)) \).
4. If \( f(n) \in O(g(n)) \), then \( f(n) \in o(g(n)) \).
5. If \( f(n) \in o(g(n)) \), then \( f(n) \in O(g(n)) \).

**Solution**

1. **False.** Big-O (“upper bound”), Big-Θ (“tight bound”), and Big-Ω (“lower bound”) notation are definitions in asymptotic notation — all are ways to describe the asymptotic behavior and efficiency of a function as its input size (usually denoted by \( n \)) scales. In other words, under the RAM model, Big-O, Big-Θ, and Big-Ω are all approaches used to bound the running time of an algorithm as \( n \) grows. In contrast, an algorithm’s worst-case, average-case, and best-case runtime are tied to a set of inputs. For example, consider Insertion Sort. The best-case runtime can occur when the input is completely sorted, and the worst-case runtime can occur when the input is completely sorted in reverse.
order. Finding the average-case runtime for Insertion Sort requires a probability distribution on the set of inputs to determine the “average” input.

Thus, none of the definitions in asymptotic notation directly correspond to the worst-case, best-case, and average-case runtimes of an algorithm. In fact, note that one can provide a Big-$O$, Big-$Θ$, and Big-$Ω$ bound on the best-case, worst-case, and average-case runtime of an algorithm — try to find the bounds for the best-case and worst-case runtimes for Insertion Sort!

2. False. As a counterexample, consider $f(n) = \sin(n)$ and $g(n) = \cos(n)$. To prove that $f(n) \in O(g(n))$ or $g(n) \in O(f(n))$, by the definition of Big-$O$, we would need to find some positive constant $n_0$ such that for all $n \geq n_0$, either $f(n)$ is an upper-bound for $g(n)$ or $g(n)$ is an upper-bound for $f(n)$. However, since both $f(n)$ and $g(n)$ are oscillating functions, observe that this is not possible: one of these functions cannot upper-bound the other function as $n$ approaches positive infinity.

3. True. ($\Rightarrow$) First, we prove that if $f(n) \in O(g(n))$, then $g(n) \in \Omega(f(n))$. If $f(n) \in O(g(n))$, then by the definition of Big-$O$, we know there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$f(n) \leq c \cdot g(n).$$

We choose $c = c^{-1}$ and $n'_0 = n_0$. Both are positive, and observe that for all $n \geq n'_0$, we have

$$g(n) \geq c' \cdot f(n),$$

so we know that $g(n) \in \Omega(f(n))$ by the definition of Big-$Ω$.

($\Leftarrow$) Next, we prove that if $g(n) \in \Omega(f(n))$, then $f(n) \in O(g(n))$. If $g(n) \in \Omega(f(n))$, then by the definition of Big-$Ω$, we know there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$g(n) \geq c \cdot f(n).$$

We choose $c = c^{-1}$ and $n'_0 = n_0$. Both are positive, and observe that for all $n \geq n'_0$, we have

$$f(n) \leq c' \cdot g(n),$$

so we know that $f(n) \in O(g(n))$ by the definition of Big-$O$.

4. False. Consider $f(n) = 3n$ and $g(n) = n$ as a counterexample. Note that $\lim_{n \to \infty} f(n)/g(n) = 3$, so by the limit definition of Big-$O$, we know that $3n \in O(n)$. However, observe that because the limit is nonzero, we can conclude that $3n \notin o(n)$. Alternatively, we also see that if we choose $c = 1$, there is no such $n_0$ where $3n < n$ is true as $n$ approaches positive infinity.

At a high-level, Big-$O$ notation implies an upper bound, and little-$o$ notation implies a loose upper bound. However, note that $f(n)$ being upper-bounded by $g(n)$ does not also imply that $g(n)$ is a loose upper-bound on $f(n)$ (as illustrated by the counterexample above).

5. True. By the definition of little-$o$, we know that for any positive constant $c$, there exists an $n_0$ such that for all $n \geq n_0$, $0 \leq f(n) < cg(n)$. Observe that we can choose any pair of these positive constants $c' = c$ and $n_0$, and this pair will satisfy the definition of Big-$O$ because we will still have $0 \leq f(n) \leq c'g(n)$ for all $n \geq n_0$.

At a high-level, Big-$O$ notation implies an upper bound, and little-$o$ notation implies a loose upper bound, so observe that $g(n)$ being a loose upper-bound on $f(n)$ does imply that $f(n)$ is still upper-bounded by $g(n)$.
Problem 2
Prove that $3n^2 + 100n = \Theta(5n^2)$.

Solution
First, we prove Big-O. By the definition of Big-O, we want to show that there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$3n^2 + 100n \leq c \cdot 5n^2$$

Setting $c = 1$, we want to find some positive constant $n_0$ such that for all $n \geq n_0$, we have

$$3n^2 + 100n \leq 1 \cdot 5n^2$$
$$3n + 100 \leq 5n$$
$$100 \leq 2n$$
$$50 \leq n$$

So when $c = 1$ and $n_0 = 50$, the expression holds for all $n \geq n_0$, proving that $3n^2 + 100n = O(5n^2)$.

Next, we prove Big-Ω. By the definition of Big-Ω, we want to show that there exist positive constants $c$ and $n_0$ such that for all $n \geq n_0$,

$$3n^2 + 100n \geq c \cdot 5n^2$$

Setting $c = 3/5$, we want to find some positive constant $n_0$ such that for all $n \geq n_0$, we have

$$3n^2 + 100n \geq (3/5) \cdot 5n^2$$
$$3n^2 + 100n \geq 3n^2$$
$$100n \geq 0$$
$$n \geq 0$$

So when $c = 3/5$ and $n_0 = 1$, the expression holds for all $n \geq n_0$, proving that $3n^2 + 100n = \Omega(5n^2)$.

Since $3n^2 + 100n = O(5n^2)$ and $3n^2 + 100n = \Omega(5n^2)$, we have proved that $3n^2 + 100n = \Theta(5n^2)$.

Tip: When proving an asymptotic bound, there are usually many pairs of $c$ and $n_0$ that fulfill the definitions, so remember that you can always choose values that make your math less messy. To actually find the values of $c$ and $n_0$, it can be helpful to guess and try out different values; for example, as shown above, one approach is to set $c$ to some value such as 1 and then see what values of $n_0$ could work.

Alternate Solution
Since the limit exists as shown below, we can apply the limit definition of Big-Θ to prove the claim as follows:

$$\lim_{n \to \infty} \frac{3n^2 + 100n}{5n^2} = \lim_{n \to \infty} \frac{3n + 100}{5n}$$
$$= \lim_{n \to \infty} \frac{3n}{5n}$$
$$= 3/5$$

Since the limit as $n$ approaches positive infinity is a nonzero constant, by the limit definition of Big-Θ, we know that $3n^2 + 100n = \Theta(5n^2)$. 

3
Problem 3
Prove using induction that \( n \lg n = \Omega(n) \).

Solution
We will prove that \( n \lg n \geq cn \) for all \( n \geq n_0 \) by induction on \( n \) and choosing \( c = 1 \) and \( n_0 = 4 \).

**Base Case:** From our choice of \( n_0 \), the base case is when \( n = 4 \). Since \( 4 \lg 4 \geq 1 \cdot 4 \), the base case holds.

**Induction Hypothesis:** Assume that we have \( k \lg k \geq k \) for some integer \( k \geq 4 \).

**Induction Step:** We want to show that \( (k + 1) \lg (k + 1) \geq k + 1 \).

\[
(k + 1) \lg (k + 1) \geq (k + 1) \lg k \quad \text{(since \( \lg k \) is monotonically increasing)} \\
= k \lg k + \lg k \quad \text{(by IH)} \\
> k \lg k + 1 \quad \text{(since \( k \geq 4 \), \( \lg k \geq 2 \), so \( \lg k > 1 \))} \\
\geq k + 1 
\]

We have shown that \( (k + 1) \lg (k + 1) \geq k + 1 \), concluding our IS and thus our proof. Therefore, we have proved that \( n \lg n = \Omega(n) \).

**Tip:** When using induction to prove an asymptotic bound, it can be helpful to first probe into your Induction Step and reverse engineer values for \( c \) and \( n_0 \) that will work the best algebraically for your proof. For example, here, note how our choice of \( n_0 \) affects both our base case and IS.

Problem 4
Prove that \( \lg(n!) = \Theta(n \lg n) \).

Solution
To prove Big-\( \Theta \), we will prove Big-\( O \) and Big-\( \Omega \) separately.

First, we prove Big-\( O \). By the definition of Big-\( O \), we want to show that there exist positive constants \( c \) and \( n_0 \) such that for all \( n \geq n_0 \),

\[ \lg(n!) \leq c \cdot n \lg n \]

Manipulating the LHS, we see

\[
\lg(n!) = \lg(1 \cdot 2 \cdots n) \\
= \lg 1 + \lg 2 + \cdots + \lg n \\
= \sum_{i=1}^{n} \lg i \\
\leq n \lg n 
\]

(by log properties)

Since the expression holds for all \( n \geq n_0 \) when \( c = 1 \) and \( n_0 = 1 \), we have proved that \( \lg(n!) = O(n \lg n) \).

Next, we prove Big-\( \Omega \). By the definition of Big-\( \Omega \), we want to show that there exist positive constants \( c \) and \( n_0 \) such that for all \( n \geq n_0 \),

\[ \lg(n!) \geq c \cdot n \lg n \]
We first find a lower-bound for \( \lg(n! \)), which we do by taking a “subset” of the terms as shown below:

\[
\lg(n!) = \lg(1 \cdot 2 \cdots n) = \lg 1 + \lg 2 + \cdots + \lg \left(\frac{n}{2}\right) + \lg\left(\frac{n}{2} + 1\right) + \cdots + \lg n \quad \text{(by log properties)}
\]

\[
\geq \frac{n}{2} + \lg\left(\frac{n}{2} + 1\right) + \cdots + \lg n \quad \text{(subset with the second half of the terms)}
\]

\[
\geq \frac{n}{2} \cdot \lg \frac{n}{2} \quad \text{(since \( \lg n \) is monotonically increasing)}
\]

Setting \( c = \frac{1}{4} \) and \( n_0 = 4 \), observe that \( \frac{n}{2} \cdot \lg \frac{n}{2} \geq \frac{1}{4} \cdot n \cdot \lg n \) for all \( n \geq n_0 \) as shown below:

\[
\iff \quad \frac{n}{2} \cdot \lg \frac{n}{2} \geq \frac{n}{4} \cdot \lg n \quad \text{(by log properties)}
\]

\[
\iff \quad \frac{n}{2} \cdot (\lg n - \lg 2) \geq \frac{n}{4} \cdot \lg n
\]

\[
\iff \quad \frac{n}{2} \cdot (\lg n - 1) \geq \frac{n}{4} \cdot \lg n
\]

\[
\iff \quad \frac{n}{2} \cdot \lg n - \frac{n}{2} \geq \frac{n}{4} \cdot \lg n
\]

\[
\iff \quad \frac{n}{4} \cdot \lg n \geq \frac{n}{2}
\]

\[
\iff \quad n \cdot \lg n \geq 2n
\]

\[
\iff \quad \lg n \geq 2
\]

In summary, from the math above, we have shown that

\[
\lg(n!) \geq \frac{n}{2} \cdot \lg \frac{n}{2}
\]

\[
\geq \frac{n}{4} \cdot \lg n
\]

Since the expression holds for all \( n \geq n_0 \) when \( c = \frac{1}{4} \) and \( n_0 = 4 \), we have proved that \( \lg(n!) = \Omega(n \cdot \lg n) \).

Since \( \lg(n!) = O(n \cdot \lg n) \) and \( \lg(n!) = \Omega(n \cdot \lg n) \), we have proved that \( \lg(n!) = \Theta(n \cdot \lg n) \).

**Tip:** We can take a subset of terms when proving Big-\( \Omega \) because we are inherently proving a “lower bound” — this makes the algebraic manipulation a lot less messy in this question!