Asymptotic Notation—Asynchronous (Monday, September 6 / Tuesday, September 7)

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.

Typically, best case performance is not really of significance; for example, algorithms can be modified to make the best-case performance trivial for certain inputs by hardcoding their solutions. In these situations, the best case performance is effectively meaningless.

Often, it is of more significance to perform worst case analysis, where we analyze the running time of the algorithm on the inputs that perform 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 never takes any longer and is thus the cleanest method of analysis.

Lastly, sometimes we can also be interested in the average case running time of an algorithm. However, this does not 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. That is, under the RAM model, Big–O, Big–Theta, and Big–Omega 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 each 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. Similarly, finding the average-case runtime for Insertion Sort requires obtaining a probability distribution on the set of inputs to the algorithm so we can find bounds for the runtime of the “average” case input.
Thus, none of the definitions in asymptotic notation directly correspond to the worst-case, best-case, and average-case runtimes of an algorithm. However, note that one can provide a Big-\(O\), Big-\(\Theta\), and Big-\(\Omega\) 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 the direction of 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-\(\Omega\).

\(\Leftarrow\) 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-\(\Omega\), 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 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 strict upper bound. However, note that \(f(n)\) being upper-bounded by \(g(n)\) does not also imply that \(g(n)\) is a strict 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 \(n \geq n_0\), \(0 \leq f(n) < c g(n)\). 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 strict upper bound, so observe that \(g(n)\) being a strict 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 $n \geq n_0$,

\[
3n^2 + 100n \leq c \cdot 5n^2
\]
\[
3n + 100 \leq 5c \cdot n
\]
\[
100 \leq (5c - 3)n
\]

Choosing $c = 1$ and $n_0 = 50$, we see

\[
100 \leq (5(1) - 3)n
\]
\[
100 \leq 2n
\]
\[
50 \leq n
\]

Since the expression holds for all $n \geq n_0$ when $c = 1$ and $n_0 = 50$, we have proved 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 $n \geq n_0$,

\[
3n^2 + 100n \geq c \cdot 5n^2
\]

Choosing $c = 3/5$ and $n_0 = 1$, we see

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

Since the expression holds for all $n \geq n_0$ when $c = 3/5$ and $n_0 = 1$, we have proved 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 $3n^2 + 100n = \Theta(5n^2)$.

Note: 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 constants that make the algebraic manipulation easier!

Alternate Solution

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

\[
\lim_{n \to \infty} \frac{3n^2 + 100n}{5n^2} = \lim_{n \to \infty} \frac{3n + 100}{5n} = \lim_{n \to \infty} \frac{3n}{5n} = \frac{3}{5}
\]

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

Problem 3

Prove using induction that $n \log n = \Omega(n)$.
Solution
We will prove that \( n \log n \geq cn \) 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 \( 4 \log 4 = 8 \geq 4 \), the base case holds.

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

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

\[
\begin{align*}
(k + 1) \log(k + 1) &\geq (k + 1) \log k \\
&= k \log k + \log k \\
&> k \log k + 1 \\
&\geq k + 1
\end{align*}
\]

(since \( \log k \) is monotonically increasing)

(by IH)

**Note:** 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.

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

**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 \( n \geq n_0 \),

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

Choosing \( c = 1 \) and \( n_0 = 1 \), we see

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

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

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

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

\[
\begin{align*}
\lg(n!) &= \lg(1 \cdot 2 \cdots n) \\
&= \lg 1 + \lg 2 + \cdots + \lg n \\
&\geq \lg \frac{n}{2} + \lg \left( \frac{n}{2} + 1 \right) + \cdots + \lg n \\
&\geq \frac{n}{2} \cdot \lg \frac{n}{2}
\end{align*}
\]

(subset with the second half of the terms)

(since \( \lg n \) is monotonically increasing)
Choosing \( c = 1/4 \) and \( n_0 = 4 \), observe that \( \frac{n}{2} \cdot \log \frac{n}{2} \geq \frac{1}{4} \cdot n \cdot \log n \) for all \( n \geq n_0 \) as shown below:

\[
\begin{align*}
\frac{n}{2} \cdot \log \frac{n}{2} & \geq \frac{n}{4} \cdot \log n \\
\frac{n}{2} \cdot \log n - \frac{n}{2} & \geq \frac{n}{4} \cdot \log n \\
(n \cdot \log n) & \geq 2n \\
\log n & \geq 2
\end{align*}
\]

(by log properties)

In summary, we have shown that

\[
\log(n!) \geq \frac{n}{2} \cdot \log \frac{n}{2} \geq \frac{n}{4} \cdot \log n
\]

Since the expression holds for all \( n \geq n_0 \) when \( c = 1/4 \) and \( n_0 = 4 \), we have proved that \( \log(n!) = \Omega(n \log n) \).

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

**Note:** 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!