Solution Set

CIS 1210—Data Structures and Algorithms—Spring 2025

Asymptotic Notation—Tuesday, January 28 / Wednesday, January 29

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 always useful because algorithms can be modified to make the best case performance trivial. For example, we may hardcode the solution/what to return for a specific input. In these situations, the best case performance is effectively meaningless. Note: Best case analysis is different from the *best conceivable runtime*, which is the fastest that *any* algorithm could conceivably solve a given problem.

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 than it. Thus, **unless otherwise specified, in CIS 1210, we will ask you to perform worst case analysis** since it is the cleanest and usually most useful method of analysis.

Average Case Analysis: This is when we analyze the runtime of the algorithm on the "average" input. This is less common than worst case analysis, as what constitutes an "average" input is usually not given to us and finding the "average" input requires taking an expectation over the probability distribution of all possible inputs to the algorithm.

As the input size grows, we use **asymptotic notation** to describe the asymptotic behavior and efficiency of a *function*. The definitions in asymptotic notations are as follows:

Big-O: If $f(n) \in O(g(n))$, there exist positive constants c and n_0 such that for all $n \geq n_0$,

$$0 \leq f(n) \leq c \cdot g(n)$$

$g(n)$ is an **asymptotic upper bound** for $f(n)$.

Big-Ω: If $f(n) \in \Omega(g(n))$, there exist positive constants c and n_0 such that for all $n \geq n_0$,

$$f(n) \geq c \cdot g(n) \geq 0$$

$g(n)$ is an **asymptotic lower bound** for $f(n)$.

Big-Θ: If $f(n) \in \Theta(g(n))$, there exist positive constants c_1, c_2 and n_0 such that for all $n \geq n_0$,

$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

In other words, $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.
$g(n)$ is an **asymptotic tight bound** for $f(n)$.

Asymptotic notation can also be defined in terms of **limits**:

Big-O: $f(n) \in O(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$ or a constant.

Big-Ω: $f(n) \in \Omega(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = \infty$ or a constant.

Big-Θ: $f(n) \in \Theta(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) =$ a nonzero constant.

Problems

Problem 1: True or False

1. A Big-O, Big-Θ, and 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 $g(n) \in \Omega(f(n))$.

Solution

1. **False**.

   Main Idea: Worst-case, average-case, and best-case are tied to the input instances, while Big-O, Big-Θ, and Big-Ω are bounds to the function's growth rate, which applies across all possible input instances.

   Big-O ("upper bound"), Big-Θ ("tight bound"), and Big-Ω ("lower bound") notation are definitions in asymptotic notation — they are all 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**.

   Main Idea: Oscillating functions can possibly neither upper bound nor lower bound each other.

   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**.

   Main Idea: If a $g(n)$ upper bounds $f(n)$, (above $f(n)$), then $f(n)$ must lower bound $g(n)$ (below $g(n)$).

($\Rightarrow$) 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.

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

$$\begin{aligned} 3n^2 + 100n &\leq 1 \cdot 5n^2 \\ 3n + 100 &\leq 5n \\ 100 &\leq 2n \\ 50 &\leq n \end{aligned}$$

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

$$\begin{aligned} 3n^2 + 100n &\geq (3/5) \cdot 5n^2 \\ 3n^2 + 100n &\geq 3n^2 \\ 100n &\geq 0 \\ n &\geq 0 \end{aligned}$$

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:

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

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 $n \lg n = \Omega(n)$.

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 $(k+1)\lg(k+1) \geq k+1$.

$$\begin{aligned}
(k+1)\lg(k+1) &\geq (k+1)\lg k && \text{(since } \lg k \text{ is monotonically increasing)} \\
&= k \lg k + \lg k \\
&> k \lg k + 1 && \text{(since } k \geq 4, \lg k \geq 2, \text{ so } \lg k > 1) \\
&\geq k + 1 && \text{(by IH)}
\end{aligned}$$

We have shown that $(k+1)\lg(k+1) \geq k+1$, concluding our induction step and thus the 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-Θ, we will prove Big-O and Big-Ω 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

$$\begin{aligned}
\lg(n!) &= \lg(1 \cdot 2 \cdots n) \\
&= \lg 1 + \lg 2 + \cdots + \lg n && \text{(by log properties)} \\
&= \sum_{i=1}^{n} \lg i \\
&\leq n \lg n && \text{(since } i \leq n)
\end{aligned}$$

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:

$$\begin{aligned}
\lg(n!) &= \lg(1 \cdot 2 \cdots n) \\
&= \lg 1 + \lg 2 + \cdots + \lg \frac{n}{2} + \lg(\frac{n}{2} + 1) + \cdots + \lg n && \text{(by log properties)} \\
&\geq \lg \frac{n}{2} + \lg(\frac{n}{2} + 1) + \cdots + \lg n && \text{(subset with the second half of the terms)} \\
&\geq \frac{n}{2} \cdot \lg \frac{n}{2} && \text{(since } \lg n \text{ is monotonically increasing)}
\end{aligned}$$

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

$$\begin{aligned}
& \frac{n}{2} \cdot \lg \frac{n}{2} \geq \frac{n}{4} \lg n \\
\iff \quad & \frac{n}{2} \cdot (\lg n - \lg 2) \geq \frac{n}{4} \lg n && \text{(by log properties)} \\
\iff \quad & \frac{n}{2} \cdot (\lg n - 1) \geq \frac{n}{4} \lg n \\
\iff \quad & \frac{n}{2} \lg n - \frac{n}{2} \geq \frac{n}{4} \lg n \\
\iff \quad & \frac{n}{4} \lg n \geq \frac{n}{2} \\
\iff \quad & n \lg n \geq 2n \\
\iff \quad & \lg n \geq 2
\end{aligned}$$

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

$$\begin{aligned}
\lg(n!) &\geq \frac{n}{2} \cdot \lg \frac{n}{2} \\
&\geq \frac{n}{4} \cdot \lg n
\end{aligned}$$

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

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

Tip: We can take a subset of terms when proving Big-Ω because we are inherently proving a "lower bound" — this makes the algebraic manipulation a lot less messy in this question! You will see this strategy in action more in the next week!