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 $g(n) \in \Omega(f(n))$.

Problem 2

Prove that $3n^2 + 100n = \Theta(5n^2)$.

Problem 3

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

Problem 4

Prove that $\lg(n!) = \Theta(n \lg n)$.