CSCI 432 Handout 03: Asymptotic Notation

Name(s): ____________________________________

30 August 2023

Definitions

First, let's recall the definitions:

Definition 1 (Asymptotic Notation). *Let $f, g\colon \mathbb{N} \to \mathbb{R}$. Then, we say that f* is *$O(g)$ iff: there exists $n_0 \in \mathbb{N}$ and $c \in \mathbb{R}_+$ such that for all $n \geq n_0$, the following holds:*

$$0 \leq f(n) \leq cg(n).$$

This is interpreted as "f is upper-bounded by g." In addition, if those same conditions hold, we also say that g is $\Omega(f)$. This is interpreted as "f g is lower-bounded by f."

The asymptotic tight bound is denoted by Θ. For $f, g\colon \mathbb{N} \to \mathbb{R}$, we say that f is $\Theta(g)$ iff f is $O(g)$ and f is $\Omega(g)$. This is the bound that we most often want to find!

Let's practice! We will complete some of the following problems in class. What we do not complete, please do them on your own. These will be collected for attendance, but not graded. Nonetheless, but you are expected to know how to answer these questions. If you have any questions, please reach out to the instructor for help.

1. Let $f\colon \mathbb{N} \to \mathbb{R}$ be defined by $f(n) = n^2 + 2$. Prove that $f(n)$ is $O(n^2)$.
   Try it!

2. Let $f\colon \mathbb{N} \to \mathbb{R}$ be defined by $f(n) = n^2 - 2$. Prove that $f(n)$ is $O(n^2)$.
   Try it!

3. If f is $\Theta(g)$, is it true or false that g is $\Theta(f)$? Why or why not?
   Try it!

4. Prove that $f\colon \mathbb{N} \to \mathbb{R}$ defined by $f(n) = \log_2(n)$ is $\Theta(\log_{10}(n))$.
   Try it!

5. Prove that Θ determines an equivalence relation on functions.
   Try it!

6. We often call $\Theta(1)$ constant and $\Theta(n)$ linear. We know that constant-time algorithms are 'faster' than linear-time algorithms. But, can you rank the following categories of runtimes?

$$\Theta(n), \Theta(n!), \Theta(1), \Theta(\log n), \Theta(n!), \Theta(2^n), \Theta(5^n), \Theta(n^2), \Theta(n^3), \Theta(n \log n)$$

   Try it!

There are some algorithms whose asymptotic runtimes we should know like the back of our hands. Let's see if we can remember (or lookup if we need to) the runtimes of the following:

1. Sorting an array[1]
2. Searching in a sorted array.
3. Binary search.
4. Linear search.
5. Summing n numbers.
6. Determining if a number is odd/even.
7. Finding the i-th item in an array.
8. Finding the i-th item in a linked list.
9. Bubble sort.
10. Finding all permutations of a set.
11. Finding all subsets of a set.

[1]Yes, this is a problem, not an algorithm. When I ask about the runtime of a problem, we think of the worst-case runtime of the best solution.