COSC-211: Data Structures
HW3: Asymptotic Analysis

Due Friday, February 22 in class

Reminder regarding intellectual responsibility: This is an individual assignment, and the work you submit should be your own. Do not look at anyone else's work, and do not show anyone your work (except for me and the course TAs). If you discussed this assignment with any of your classmates, please note their names at the top of your submission.

This is a written assignment. I recommend typing up your solutions using LaTeX, which is a typesetting language that allows you to format math nicely (among many other functions). There's a lot of free software available to help you write LaTeXcode; my personal favorite is TeXstudio, available to download at https://www.texstudio.org/. You are also free to use your favorite word processor. If you write your solutions by hand, please be neat and legible!

1 Your Tasks

1. Rank the following functions from smallest to largest according to their big-O complexity. That is, order them based on their asymptotic growth rate. For example, you could write $n < n^3$ since $n \in O(n^3)$, but $n^3 \notin O(n)$. Some of the functions might be in the same big-O class. You do not need to do any formal proofs.

$$n^{100} \qquad \lg n \qquad 2^n \qquad 2^{\lg n} \qquad 4 \qquad \sqrt{n} \qquad n! \qquad 100n$$

2. Let $f(n) = 3n^2 + 5n - 12$. Prove that $f(n) \in O(n^2)$.

3. Prove that $O(n) \subseteq O(n^2)$. The notation $\subseteq$ means "is a subset of or equal to". That is, we want to show that every function in $O(n)$ must also be in $O(n^2)$. [Hint: this is a "for all" claim. Start by considering an arbitrary function $f(n)$, where all you know about $f(n)$ is that it's in $O(n)$.]

4. Java's `Stack` class provides a method called `search`. The `search` method takes an `Object` as an input parameter and returns and `int` representing the location of that `Object` within the stack (the top of the stack is 1), or -1 if the `Object` isn't in the stack.

Suppose we add a `search` method to our array-based `Stack` of `Book` objects and implemented it as follows:

```
public int search(Book b) {
   for (int i = 0; i < top; i++) {
      if (booklist[i].equals(b)) return i+1;
   }
   return -1;
```

What is the worst case big-O runtime of `search` in terms of n (the number of items in the stack)? What about the best case? Explain your answers.

2 Submit your work

Bring a hard copy of your solutions to class.

This assignment is on Friday, February 22 in class.