CMPS 2200 Assignment 1

Name: Jake Lehmer

In this assignment, you will learn more about asymptotic notation, parallelism, functional languages, and algorithmic cost models. As in the recitation, some of your answer will go here and some will go in `main.py`. You are welcome to edit this `assignment-01.md` file directly, or print and fill in by hand. If you do the latter, please scan to a file `assignment-01.pdf` and push to your github repository.

1. **Asymptotic notation**

(yes) –
- 1a. Is $2^{n+1} \in O(2^n)$? Why or why not?

$2^{n+1} = 2 \cdot 2^n \in O(2^n)$, since only the coefficient differs.

(no) –
- 1b. Is $2^{2^n} \in O(2^n)$? Why or why not?

$2^{2^n} \in O(2^n) \Rightarrow 2^n \in n$, which is false. Hence, $2^{2^n} \notin O(2^n)$.

(no) –
- 1c. Is $n^{1.01} \in O(\log^2 n)$?

$(\forall n)[n > (\log n)^2) \Rightarrow n^{1.01} \notin O(\log^2 n)$ [$n^{1.01} \geq n$].

(yes) –
- 1d. Is $n^{1.01} \in \Omega(\log^2 n)$?

Yes, b/c Ω is the reverse of above (i.e., best case).

(no) –
- 1e. Is $\sqrt{n} \in O((\log n)^3)$?

No, b/c where graphed, $\sqrt{n}$ always exceeds $c \cdot (\log n)^3$ for $n > n^*$.

(yes) –
- 1f. Is $\sqrt{n} \in \Omega((\log n)^3)$?

Yes, b/c Ω is the reverse of above (i.e., best case).

- 1g. Consider the definition of "Little o" notation:

$g(n) \in o(f(n))$ means that for **every** positive constant c, there exists a constant n_0 such that $g(n) \leq c \cdot f(n)$ for all $n \geq n_0$. There is an analogous definition for "little omega" $\omega(f(n))$. The distinction between $o(f(n))$ and $O(f(n))$ is that the former requires the condition to be met for **every** c, not just for some c. For example, $10x \in o(x^2)$, but $10x^2 \notin o(x^2)$.

.