CS 3510 – Assignment 1

Due `Friday, May 27, 2022` at `11:59pm` on Canvas

- Please type your answers (LaTeX is highly recommended) and upload a single PDF file named `<Your-GT-Account>.pdf`, e.g., GBurdell3.pdf, including all your answers. You can submit multiple times. Canvas keeps track of the submissions and append a version number when you re-submit. We always grade your most recent submissions.
- Please read the policies, and do not forget to acknowledge your collaborators and cite your references.
- If you do not understand a question, please ask on Piazza or come to office hours well ahead of the due date.

Problem 1 — Asymptotic Notations (10 pts)

1. (5 pts) For each pair of functions f and g, write whether f is in $\mathbb{O}(g)$, $\Omega(g)$, or $\Theta(g)$, whichever is most accurate. Just write the asymptotic notation; no explanation is required.
   (a) $f = (n + 1000)^4$, $g = 1000n^4 - 2n^3 + 1$
   (b) $f = \log_{1000} n$, $g = \log_2 n$
   (c) $f = n^{1000}$, $g = n^2$
   (d) $f = 2^n$, $g = n!$
   (e) $f = (n + 1)^3$, $g = 4^{\log_2 n}$ (Hint: $a^{\log_b c} = c^{\log_b a}$)
2. (5 pts) Use the mathematical definition of big-O notation to prove the following additivity properties: f, g and h are functions of input size n. Prove that if $f \in \mathbb{O}(h)$ and $g \in \mathbb{O}(h)$, then $f + g \in \mathbb{O}(h)$.

Problem 2 — Divide and Conquer (20 pts)

You are given a sorted array $S = [s_1, s_2, \dots, s_n]$ with n distinct integers, i.e., $s_i < s_{i+1}$, for all $1 \leq i < n$. Design a divide-and-conquer algorithm to decide whether there exists an index k such that $S[k] = k$. If such an element exists return the index, otherwise return -1. Your algorithm should run in $O(\log n)$ time.

- Provide a description of your algorithm (in words and pseudocode), and justify its correctness.
- Discuss the running time by providing the recurrence relation and applying the Master Theorem.

Problem 3 — Divide and Conquer (10 pts)

You are given a rotated sorted array S of size n. Design a binary search algorithm to find the minimum element of this array. Your algorithm should run in $O(\log n)$ time. Provide a description of your algorithm. Runtime analysis is not required.

Def. Rotated sorted array of size n is a sorted array, where its elements are shifted k times ($0 \leq k < n$) to the right. For instance, let $S = [0, 1, 2, 3, 4, 5, 8]$ be a sorted array before rotation, then

- *After $k = 3$ rotations: $S = [4, 5, 8, 0, 1, 2, 3]$*
- *After $k = 6$ rotations: $S = [1, 2, 3, 4, 5, 8, 0]$*

Note for both examples, your algorithm should return 0 as the minimum of the array.