COMP 3711 – Design and Analysis of Algorithms
2022 Fall Semester – Written Assignment 1
Distributed: September 9, 2022
Due: September 26, 2022, 23:59

Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:

- Please write clearly and briefly. In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.
- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page. ***You must acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- The term *Documented Pseudocode* means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.
- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.
- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.
- Submit a SOFTCOPY of your assignment to Canvas by the deadline. If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

1. (10 points) An array $A[1..n]$ of n distinct integers for some $n \geq 2$ can be sorted in decreasing order using the following algorithm. This algorithm has a main loop that iterates $n-1$ times. In each iteration, the algorithm visits the elements of A in decreasing order of indices, that is, $A[n]$, $A[n-1]$, $A[n-2]$, …, $A[2]$ in this order. Whenever the algorithm visits an element $A[i]$, it compares $A[i]$ with $A[i-1]$, and if $A[i] > A[i-1]$, the algorithm swaps the contents of $A[i-1]$ and $A[i]$.

   (a) Write a documented pseudocode for the algorithm described above. Make sure that you document your pseudocode properly.

   (b) Prove that the algorithm described above works correctly. That is, for any $n \geq 2$, $A[1..n]$ is sorted in decreasing order by the algorithm. (Hint: What happens to the array A after the first i passes?)

   (c) Analyze the asymptotic worst-case running time of one iteration of the algorithm. Use your result to derive a bound on the worst-case running time of the algorithm.

2. (10 points) Let $A[1..n]$ be an array of n distinct integers for some $n \geq 1$.

   (a) Describe a *recursive* algorithm that returns a list of all permutations of $A[1..n]$. You can either describe your algorithm in text or in a documented pseudocode. Explain your output representation. Make sure that your algorithm is recursive. (Hint: If you have all permutations of $A[2..n]$, how do you obtain all permutations of $A[1..n]$?)

   (b) Write down the recurrence for the running time of your recursive algorithm in (a) with the boundary conditions. Explain your notations. Solve your recurrence to obtain the the running time of your algorithm.

3. (5 points) The following list specifies a pair of functions in each row. For each pair of functions, tell us the relation between the two functions in terms O, Ω, and Θ. Of course, if Θ is applicable, it already subsumes O and Ω. No explanation is needed.

n^2	$n^2/\log n$
n^3	$2^{\log n}$
$\log n$	$\log\log n$
n^2	$4^{\log_2 n}$
$(\log n)^2$	$n^{1/3}$

4. (10 points) Derive asymptotic upper bounds for $T(n)$ in the following recurrences. The boundary condition in each case is $T(1) = 1$. Make your bounds as tight as possible. You may apply the master theorem or derive your solutions from scratch. In either case, show all your work steps in order to gain full credits.

   (a) $T(n) = 4T(n/2) + n$. You may assume that n is a power of 2.

(b) $T(n) = 4T(n/2) + n^3$. You may assume that n is a power of 2.

(c) $T(n) = T(n/3) + \sqrt{n}$. You may assume that n is a power of 3.

(d) $T(n) = 5T(n/4) + n \log n$. You may assume that n is a power 4.

(e) $T(n) = T(\lfloor \sqrt{n} \rfloor) + 1$.

5. (10 points) Let $f : \mathbb{R} \to \mathbb{R}$ be a strictly convex function. That is, for all $x, z \in \mathbb{R}$ and for all $\lambda \in [0, 1]$, we have $f(\lambda x + (1 - \lambda)z) < \lambda f(x) + (1 - \lambda)f(z)$. An equivalent interpretation is that for any $x < y < z$, the point $(y, f(y))$ lies strictly below the line segment with endpoints $(x, f(x))$ and $(z, f(z))$. An easy implication is that for every closed interval $[a, b]$, there is a unique minimum in $\{f(x) : x \in [a, b]\}$.

   Suppose that an array $A[1..n]$ stores n numbers such that $A[i] = f(i)$ for all $i \in [1, n]$. Describe an algorithm that finds the minimum in $A[1..n]$ in $O(\log n)$. You only know that there exists a strictly convex function f such that $A[i] = f(i)$ for all $i \in [1, n]$; you are not given any other information about the function f. Prove the correctness of your algorithm. Explain why the running time of your algorithm is $O(\log n)$.

6. (10 points) We can represent a polynomial in x of degree n using an array $A[0..n]$ such that $A[i]$ is the coefficient of the term x^i. That is, the polynomial is equal to $A[n] \cdot x^n + A[n-1] \cdot x^{n-1} + \cdots + A[1] \cdot x + A[0]$. Given two arrays $P[0..n]$ and $Q[0..n]$ that represent two polynomials $p(x)$ and $q(x)$ of degree n, respectively, describe a divide-and-conquer algorithm that computes the product $p(x) \cdot q(x)$ with a worst-case running time of $o(n^2)$. That is, the worst-case running time of your algorithm must be asymptotically less than $\Theta(n^2)$. The output of your algorithm is required to be an array $R[0..2n]$ that represents a polynomial in x of degree $2n$, i.e., $R[2n] \cdot x^{2n} + R[2n-1] \cdot x^{2n-1} + \cdots + R[1] \cdot x + R[0]$. Argue that your algorithm is correct. Derive the worst-case running time of your algorithm.