Homework 1

Please write your solutions in the LaTeX. You may use online compiler such as Overleaf or any other compiler you are comfortable with to write your solutions in the LaTeX.

Due date: Monday, Sep 18, 9:10 am EDT.

I will collect your submission when the class meets on Monday, Sep 18. Please handover a printed copy of your Homework 1 solutions (preferably written in the LaTeX). Also, please make sure that you have your full name and student ID in your submission.

You can use the LaTeX submission template I have shared along with the homework. There are two .tex files (macros.tex, and main.tex). You can upload the zipped folder directly to Overleaf, and edit main.tex to write your solutions. macros.tex is mostly for macros (predefined commands).

Handwritten solutions will also be accepted. Points will be deducted if handwritten solutions are not legible.

A quick review of **Section** 3.2 **Standard notations and common functions**, pg 53 in CLRS Third Edition, might be helpful to work on the following exercises, particularly Monotonicity, Floors and ceilings, Polynomials, Exponentials, Logarithms, & Functional iteration from Section 3.2.

Problem 1-1. [5 points] Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $8n^2$ steps, while merge sort runs in $64n \lg n$ steps. For which value of n does insertion sort beat merge sort?

Problem 1-2. Consider the ***searching problem***:

Input: A sequence of n numbers $A = \langle a_1, a_2, ..., a_n \rangle$ and a value v.
Output: An index i such that $v = A[i]$ or the special value NIL if v does not appear in A.

(a) **[10 points]** Write pseudocode for ***linear search***, which scans through the sequence, looking for v.

(b) **[15 points]** Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties (Initialization, Maintenance, & Termination).

Problem 1-3. Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element $A[1]$. Then the second smallest element of A, and exchange it with $A[2]$. Continue in this manner for the first $n - 1$ elements of A.

(a) **[10 points]** Write pseudocode for this algorithm, which is known as ***selection sort***.

(b) **[5 points]** What loop invariant does this algorithm maintain?

(c) **[5 points]** Why does it need to run for only the first $n - 1$ elements, rather than for all n elements?

(d) **[5 points]** Give the best-case and worst-case running times of selection sort in Θ-notation.

Problem 1-4. [15 points] Using the definitions of $O(n)$ and $\Omega(n)$, show that $f(n) = 2n^2 + 3n + 5$ is $f(n) = O(n^2)$. Also, show that $f(n) = \Omega(n^2)$. Once you show that $f(n) = O(n^2)$ and $f(n) = \Omega(n^2)$, use the definition of $\Theta(n)$ to show that $f(n) = \Theta(n^2)$.

Problem 1-5. [5 points] Explain why the statement, "The running time of algorithm A is at least $O(n^2)$." is meaningless.

Problem 1-6. [10 points] Prove that for any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Problem 1-7. [10 points] Show that the solution of $T(n) = T(n - 1) + n$ is $O(n^2)$ using the substitution method.

Problem 1-8. [10 points] Show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$ using substitution method. [*Hint:* You can revisit "Subtleties" section in pg. 85 in CLRS Third Edition to solve this problem.]

Problem 1-9. [15 points] Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(n/2) + n^2$. Use the substitution method to verify your answer.
[If you are using LaTeX to write solution to this, a quick way to show your recursion tree would be to use a picture of a handwritten recursion tree, and the rest of the answer in LaTeX.]

Problem 1-10. [15 points] Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use the substitution method to verify your answer.
[If you are using LaTeX to write solution to this, a quick way to show your recursion tree would be to use a picture of a handwritten recursion tree, and the rest of the answer in LaTeX.]

Extra Credit

Problem 1-11. [12 points] Use the master method to give tight asymptotic bounds of the following recurrences.

(a) $T(n) = 2T(n/4) + 1$.

(b) $T(n) = 2T(n/4) + \sqrt{n}$.

(c) $T(n) = 2T(n/4) + n^2$.

Problem 1-12. Referring back to the searching problem **1.2**, observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. The ***binary search*** algorithm repeats this procedure, halving the size of remaining portion of the sequence each time.

(a) **[10 points]** Write a recursive pseudocode for binary search.

(b) **[10 points]** Show that the worst-case running time of binary search to $\Theta(\lg n)$. Write out the running time function of binary search as a recurrence relation, and use the recurrence to show that worst-case running time.