Foundations of Algorithms, Fall 2022
Homework #1

In this course, critical thinking and problem analysis results in discovering appropriate and better, the best, algorithm for a problem; defining the algorithm in pseudocode; demonstrating (we don't usually prove) that the algorithm is correct; and determining the asymptotic runtime for the algorithm using one or more of the tools for asymptotic analysis you have studied this semester. Each algorithm you design must follow the **Pseudocode Restrictions** (Canvas page *Pseudocode Restrictions & Programming Guidelines*) when preparing your solutions to these problems.

Exercising the algorithm you design using one or more example data set(s) removes any doubt from the grader that the algorithm is structurally correct and all computations are correct. When a problem supplies data you are expected to use it to demonstrate that your algorithm is correct.

All members of the collaboration group are expected to participate fully in solving collaborative problems, and peers will assess performance at the end of the assignment. Note, however, that each student is required to write up their solutions individually. Common solution descriptions from a collaboration group will not be accepted. Furthermore, to receive credit for a collaboration problem, each student in the collaboration group must actively and substantially contribute to the collaboration. This implies that no single student should post a complete solution to any problem at the beginning of the collaboration process.

You are permitted to use Internet resources while solving problems, however complete references must be provided. Please follow the Sheridan Libraries' citation guidance at "*Citing Other Things* - HOW DO YOU CITE AN INTERVIEW, A TWEET, OR A PUBLIC WEB PAGE?" and continue to the APA Academic Writer *Sample References*. Additional example citations are provided at the Purdue University, Purdue Online Writing Lab (OWL), College of Liberal Arts *Reference List: Electronic Sources*. You can also use training provided by Victoria University Library, Melbourne Australia on *Harvard Referencing: Internet/websites*

All written assignments are expected to be typed in a word processor, the preferred submission type is PDF allowing for ease of grading. When using equations, make use of the built-in equation editor for these tools. Try to avoid scanning hand-drawn figures, if needed please ensure they are readable. Only scans from flat bed scanners or printers will be accepted; all scans from phones or phone photos will be rejected. The grader will make the determination of "readability". If the assignments are not legible, they will be returned to the student with a grade of zero.

1. [10 points] Describe the time complexity of the this linear search algorithm. Choose the tightest asymptotic representation, from Θ, O, or Ω, and argue why that is the tightest bound.

```
LINEAR-SEARCH(x, A)
1  i = 1                          //Arrays in the text begin at 1 in all but a few algorithms this semester
2  while i ≤ A.length and x ≠ A[i]
3        i = i + 1
4  if i ≤ A.length
5        location = i
6  else location = 0
7  return location
```

2. [15 points] Consider this binary search algorithm. This algorithm is different than the algorithm on page 799 of the CLRS text. Array A is a sorted list of elements. x may or may not be in A.

```
BINARY-SEARCH(x, A)
 1  i = 1                          //Arrays in the text begin at 1 in all but a few algorithms this semester
 2  j = A.length
 3  while i < j
 4        m = ⌊(i + j)/2⌋
 5        if x > A[m]
 6              i = m + 1
 7        else j = m
 8  if x = A[i]
 9        location = i
10  else location = 0
11  return location
```

Assignment continues on the next page ...

(a) [10 points] Describe the time complexity of the binary search algorithm in terms of number of comparisons used (ignore the time required to compute $m = \lfloor (i + j)/2 \rfloor$ in each iteration of the loop in the algorithm). Choose the tightest asymptotic representation, from Θ, O, or Ω, and argue why that is the tightest bound.

(b) [5 points] Make one small change to the algorithm above to improve its runtime, and give the revised tightest asymptotic representation in terms of number of comparisons used, from Θ, O, or Ω. Show your change using proper pseudocode. If the asymptotic representation changed from your answer to 2a, argue why it is different.

3. [10 points] Give asymptotic upper and lower bounds for $T(n) = 4T(n/4) + n^2$, assuming $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible. Prove that your bounds are correct.

4. [15 points] Give asymptotic upper and lower bounds for $T(n) = 3T\left(\frac{n}{3} + 1\right) + n$, assuming $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible. Prove that your bounds are correct.

5. [15 points] Give asymptotic upper and lower bounds for $T(n) = T(\sqrt{n}) + 1$, assuming $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible. Prove that your bounds are correct.

6. [35 points] ***Collaborative Problem*** –CLRS 2-1: Although merge sort runs in $\Theta(n \lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to *coarsen* the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which n/k sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

   (a) [10 points] Prove that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time.

   (b) [10 points] Prove how to merge the sublists in $\Theta(n \lg (n/k))$ worst-case time.

   (c) [10 points] Given that the modified algorithm runs in $\Theta(nk + n \lg (n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as standard merge sort, in terms of Θ-notation?

   (d) [5 points] How should we choose k in practice?

==========

Prove versus Show: When an assignment question contains "prove" you are expected to develop a formal proof. There are proof writing resources in *Support Documents*, item "Proofs". When an assignment question contains "show", or no specification at all, you may submit an example, a sketch of a proof, or an argument for your conclusion without meeting the expectations of a proof.