Foundations of Algorithms, Fall 2022
Homework #3

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. [5 points] **[CLRS 5.2-3]** Use indicator random variables to compute the expected value of the sum of n dice.

2. [5 points] **[CLRS 5.3-3]** Consider the PERMUTE-WITH-ALL [immediately following this problem in the CLRS text], which instead of swapping element $A[i]$ with a random element from the subarray $A[5 : n]$, swaps it with a random element from anywhere in the array. Does PERMUTE-WITH-ALL produce a random uniform permutation? Why or why not?

3. [10 points] The following problem is known as the *Dutch Flag Problem*. In this problem, the task is to rearrange an array of characters R, W, and B (for red, white, and blue, which are the colors of the Dutch national flag) so that all the R's come first, all the W's come next, and all the B's come last.
   Let's be real - there are SO many solutions to this problem on the Internet that it takes all the challenge out of it.

   (a) Your assignment is to find improvement(s) to the basic linear, in-place, algorithm that solves the Dutch Flag problem. You must preserve the in-place attribute of the basic algorithm.

   (b) Your solution must include a proof of the asymptotic runtime of you algorithm, and an explanation for why it remains an in-place algorithm.

4. [15 points] Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. Your solution must include:

   - a natural language description of your algorithm's operation(s),
   - an algorithm written in pseudocode in the style of the text,
   - a proof of the asymptotic runtime of you algorithm

(Continued on next page)

5. [15 points] Let $A(i)$, $1 \leq i \leq n$, be an unsorted set of positive integers. Write a **nondeterministic polynomial time algorithm** NSORT(A, n) which sorts A in linear time. Your solution must include:

   - a natural language description of your algorithm's operation(s),
   - a **nondeterministic polynomial time algorithm** written in pseudocode in the style of the text,
   - a proof of the asymptotic runtime of you algorithmthe procedure

6. [15 points]**[CLRS 6.1]** One way to build a heap is by recursively calling MAX-HEAP-INSERT to insert the elements into the heap. Consider the procedure BUILD-MAX-HEAP' [that follows this pararaph in the CLRS text]. It assumes the objects being inserted are just the heap elements.

   - Do the procedures BUILD-MAX-HEAP and BUILD-MAX-HEAP' do not always create the same heap when run on the same input array. Prove that they do, or provide a counterexample. [Your counnterexample, if provided, cannot have been used as an example or in a solution key available on the Internet. No points will be awarded for such counterexamples.]
   - Show that in the worst BUILD-MAX-HEAP' requires $\Theta(n \lg n)$ time to build an n-element heap.

7. [15 points]**[CLRS 8-3]** Answer one of the following problems:

   - You are given an array of integers, where different integers may have different numbers of digits, but the total number of digits over *all* the integers in the array is n. Show how to sort the array in time $O(n)$ time.
   - You are given an array of strings, where different strings may have different number of characters, but the totoal number of characters over all strings is n. Show how to sort the strings in $O(n)$ time. (the desired order is the standard alphabetical order: for example $a < ab < b$.)

8. [20 points] ***Collaborative Problem***: A number of *peer-to-peer systems* on the internet are based on *overlay networks*. Rather than using the physical internet as the network on which to perform computation, these systems run protocols by which nodes choose collections of virtual "neighbors" so as to define a higher-level graph whose structure may bear little or no relation to the underlying physical network. Such an overlay network is then used for sharing data and services, and it can be extremely flexible compared with a physical network, which is hard to modify in real time to adapt to changing conditions.

   Peer-to-peer networks tend to grow through the arrival of new participants who join by linking into the existing structure. This growth process has an intrinsic effect on the characteristics of the overall network. Recently, people have investigated simple abstract models for network growth that might provide insight into the way such processes behave in real networks at a qualitative level.

   Here is a simple example of such a model. The system begins with a single node v_1. Nodes then join one at a time; as each node joins, it executed a protocol whereby it forms a directed link to a single other node chosen uniformly at random from those already in the system. More concretely, if the system already contains nodes $v_1, ..., v_{k-1}$ and node v_k wishes to join, it randomly selects one of $v_1, ..., v_{k-1}$ and links to that node.

   Suppose we run this process until we have a system consisting of nodes $v_1, ..., v_n$; the random process described above will produce a directed network in which each node other than v_1 has exactly one outgoing edge. On the other hand, a node may have multiple incoming links, or none at all. The incoming links to a node v_j reflect all the other nodes whose access into the system is via v_j; so if v_j has many incoming links, this can place a large load on it. Then to keep the system load-balanced, we would like all the nodes to have a roughly comparable number of incoming links. That is unlikely to happen, however, since nodes that join earlier in the process are likely to have more incoming links than nodes that join later. Let us try to quantify this imbalance as follows.

   (a) [10 points] Given the random process described above, what is the expected number of incoming links to node v_j in the resulting network? Give an exact formula in terms of n and j, and also try to express this quantity asymptotically (via an expression without large summations) using $\Theta(\cdot)$ notation.

   (b) [10 points] Part (a) makes precise a sense in which the nodes that arrive early carry an "unfair" share of connections in the network. Another way to quantify the imbalance is to observe that, in a run of this random process, we expect many nodes to end up with no incoming links. Give a formula for the expected number of nodes with no incoming links in a network grown randomly according to this model.

(End of the assignment)