Foundations of Algorithms, Fall 2022
Homework #4

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. [20 points] Consider the following algorithm for doing a postorder traversal of a binary tree with root vertex *root*.

   ```
   POSTORDER(root)
   1  if root ≠ null
   2       POSTORDER(root.left)
   3       POSTORDER(root.right)
   4       VISIT(root)
   ```

   Prove that this algorithm run in time $\Theta(n)$ when the input is an n-vertex binary tree.

2. [20 points] We define an AVL binary search tree to be a tree with the binary search tree property where, for each node in the tree, the height of its children differs by no more than 1. For this problem, assume we have a team of biologists that keep information about DNA sequences in an AVL binary search tree using the specific weight (an integer) of the DNA sequence as the key. The biologists routinely ask questions of the type, "Are there any structures in the tree with specific weight between a and b, inclusive?" and they hope to get an answer as soon as possible.

   (a) [10 points] Design an efficient algorithm that, given integers a and b, returns TRUE if there exists a key x in the tree such that $a \leq x \leq b$, and FALSE if no such key exists in the tree.

   (b) [5 points] What is the time complexity of your algorithm?

   (c) [5 points] What does your algorithm do if searches for both a and b are unsuccessful?

3. [5 points] **[CLRS 8.2-3 page 211]** Suppose that we were to rewrite the **for** loop header in line 11 of the *COUNTING−SORT* as

   ```
   11  for j = 1 to n
   ```

   Show that the algorithm still works properly, but is not stable. Then rewrite the pseudocode for counting sort so that elements with the same value are written into the output array in order of increasing index and the algorithm is stable.

(Continued on next page)

4. [10 points]**[CLRS 9.3-1 page 241]** In the algorithm SELECT, the input elements are divided into groups of 5.

   (a) [5 points] Show that the algorithm work in linear time if the input elements are divided into groups of 7 instead of 5?

   (b) [5 points] Argue that SELECT does not work in linear time if groups of 3 are used. (not in CLRS but part of this assignment)

Instructions specific to Problems 5 and 6: It is no secret that these problems are solved by dynamic programming. Your solution to each problem must address each of the four steps defined in Chapter 14, initially on page 362.

5. [20 points] Suppose you are consulting for a company that manufactures computer equipment and ships it to distributors all over the country. For each of the next n weeks, they have a weekly projected supply, ws_i, of equipment (measured in pounds) that has to be shipped by an air freight carrier. Each week's supply can be carried by one of two air freight companies, A or B.

   - Company A charges at a fixed rate r per pound (so it costs $r * ws_i$ to ship a week's supply ws_i).
   - Company B makes contracts for a fixed amount c per week, independent of weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

   A *schedule* for the computer company is a choice of air freight company (A or B) for each of the n weeks with the restriction that company B, whenever it is chosen, must be chosen for blocks of four contiguous weeks in time. The *cost* of the schedule is the total amount paid to companies A and B, according to the description above.

   Give a polynomial-time algorithm that takes a sequence of supply values $ws_1, ..., ws_n$ and returns a schedule of minimum cost. For example, suppose $r = 1$, $c = 10$, and the sequence of weekly projected supply values is

   $$11, 9, 9, 12, 11, 12, 12, 9, 9, 11.$$

   Then the optimal schedule would be to chose company A for the first three weeks, company B for the next block of four contiguous weeks, and then company A for the final three weeks.

6. [20 points] ***Collaborative Problem*** : As some of you know well, and others of you may be interested to learn, a number of languages (including Chinese and Japanese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the *word segmentation problem*–inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like "meetateight" and deciding that the best segmentation is "meet at eight" (and not "me et at eight" or "meet ate ight" or any of a huge number of even less plausible alternatives). How could we automate this process?

   A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative "quality" of its individual constituent words. Thus, suppose you are given a black box that, for any string of letters $x = x_1x_2...x_k$, we return a number $quality(x)$ in constant time, $\Theta(1)$. This number can either be positive or negative; larger numbers correspond to more plausible English words. (So $quality(\text{“}me\text{''})$ would be positive while $quality(\text{“}ight\text{''})$ would be negative.)

   Given a long string of letters $y = y_1y_2...y_n$, a segmentation of y is a partition of its letters into contiguous blocks of letters, each block corresponding to a word in the segmentation. The *total quality* of a segmentation is determined by adding up the qualities of each of its blocks. (So we would need to get the right answer above provided that $quality(\text{“}meet\text{”}) + quality(\text{“}at\text{”}) + quality(\text{“}eight\text{”})$ was greater than the total quality of any other segmentation of the string.) Give an efficient algorithm that takes a string y and computes a segmentation of maximum total quality. Prove the correctness of your algorithm and analyze its time complexity.

(Continued on next page)

7. [20 points] Suppose you are acting as a consultant for the Port Authority of a small Pacific Rim nation. They are currently doing a multi-billion-dollar business per year, and their revenue is constrained almost entirely by the rate at which they can unload ships that arrive in the port. Here is a basic sort of problem they face.

   A ship arrives with n containers of weight $w_1, w_2, ..., w_n$. Standing on the deck is a set of trucks, each of which can hold K units of weight. (You may assume that K and w_i are integers.) You can stack multiple containers in each truck, subject to the weight restrictions of K. The goal is to minimize the number of trucks that are needed to carry all the containers. This problem is NP-complete.

   A greedy algorithm you might use for this is the following. Start with an empty truck and begin piling containers 1,2,3,... onto it until you get to a container that would overflow the weight limit. (These containers might not be sorted by weight.) Now declare this truck "loaded" and send it off. Then continue the process with a fresh truck. By considering trucks one at a time, this algorithm may not achieve the most efficient way to pack the full set of containers into an available collection of trucks.

   (a) [10 points] Give an example of a set of weights and a value for K where this algorithm does not use the minimum number of trucks.

   (b) [10 points] Show that the number of trucks used by this algorithm is within a factor of two of the minimum possible number for any set of weights and any value of K.

(End of assignment)