# Foundations of Algorithms, Spring 2023
# Homework #3

All members of the collaboration group are expected to participate fully in solving collaborative problems. 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.

## Problems for Grading

1. [25 points] Consider the following algorithm for doing a postorder traversal of a binary tree with root vertex $root$.

**Algorithm 1** Postorder Traversal

```
function POSTORDER($root$)
    if $root \neq$ null then
        POSTORDER($root.left$)
        POSTORDER($root.right$)
        visit $root$
    end if
end function
```

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

2. [25 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 projected $supplys_i$ 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 $rs_i$ to ship a week's supply $s_i$).
   - 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.

   We seek a way in polynomial-time to take in a sequence of supply values $s_1, ..., s_n$ and return a schedule of minimum cost. For example, suppose $r = 1$, $c = 10$, and the sequence of values is

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

   Then the optimal schedule would be to choose 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. One way to do this is set up the Bellman equation, then let a solver get the answer in polynomial time.

   For this problem, derive the Bellman equation, and talk through how you derived it. You must write down your Bellman equation, and you must sufficiently explain your rationale to get credit for this problem.

3. [25 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 $quality(x)$. This number can either be positive or negative; larger numbers correspond to more plausible English words. (So $quality(“me'')$ would be positive while $quality(“ight'')$ 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 $quality(“meet”) + quality(“at”) + quality(“eight”)$ 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.

Derive the Bellman equation for this, and prove that it is correct.

[**Hints**: Remember you are studying Chapters 9, 12, 14, 15, and 16 during Modules 7 and 8. This problem is solved by something you study in those chapters. You do not need to know the algorithm in the black box.]

4. [25 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) [12.5 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) [12.5 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 $K$.