CP312 Algorithm Design and Analysis I **Winter 2024**

Assignment 3

Instructor: Dariush Ebrahimi *Due Date: 15-Mar-2024*

Instructions: You must submit your solutions as a single PDF file to MyLS. Make your solutions as detailed as possible by clearly stating every step in your answers. The assignment must be done individually. Any **COPYING** of solutions from external sources will result in a **ZERO** grade.

Problems

1. **Task Scheduling.** In this problem, we will revisit task scheduling, which we have solved in class using a greedy approach.

(a) (6 points) Recall that we have proved the optimality of our greedy approach using the **earliest finish time** as our greedy choice. For each of the following alternative greedy choices, **show a counterexample** to prove that this will not always succeed in outputting the optimal result (which is defined as the maximum number of scheduled tasks):

(a) Longest duration

(b) Lowest task ID

(c) Latest finish time

(b) (3 points) An alternative greedy choice is to select the task with the **latest start time** first. Explain why this greedy choice will give an optimal solution. Use the same proof technique we used for the fractional knapsack problem to prove the greedy choice property.

(c) (3 points) Consider a variant of the task scheduling problem where each activity has a cost ***c*** associated with it and the problem now is to find a set of non-conflicting activities such that **cost is minimized** (note that this is different than having the maximum number of activities). Show, using a **counterexample** that the greedy choice of earliest finish time will not yield an optimal solution for this new problem.

2. **Huffman Encoding.** You are given the following string of characters:

YPPTPQTPYYTPTQPAYPT

(a) (1 point) Compute the number of bits needed to represent the above string if using ASCII encoding.

(b) (6 points) Use Huffman coding to encode the above string. Show each step of your procedure and output the final string with the Huffman Tree.

(c) (2 points) Decode the following string using your Huffman tree:

01110001010100001101

(d) (3 points) Provide an example of a character string that, after applying Huffman encoding, will assign all characters a codeword of length 2 bits.

3. **Knapsack Problem.** In this problem, we will revisit the 0/1 and fractional knapsack problems. Suppose that you are given the following weight/value pairs for objects to be inserted in a bag of capacity **8**.

item no.	1	2	3	4
weight	3	6	4	2
value	15	24	12	16

(a) (6 points) Find the optimum solution of the 0/1 knapsack problem for the above instance using **dynamic programming**. Determine the set of items that yield the optimum solution. Show the table and all your work.

(b) (3 points) Show that a **greedy algorithm** using the **density** of items as its greedy choice will not yield the optimum solution if it is trying to solve the 0/1 knapsack problem for the above instance. Compare the value obtained by the greedy algorithm against the optimum solution.

(c) (3 points) Show the output of the greedy algorithm when solving the **fractional** knapsack problem for the above instance. State whether the result is an optimum solution or not.

4. **Max Product-Sum.** In this problem, you are given an array of n positive integers representing a sequence $X_1, X_2, \ldots, X_n$. Your goal is to create an expression out of this array by inserting either an addition operation or a multiplication operation between each contiguous pair of integers such that the value of the expression is **maximized**. However, we cannot insert **two** consecutive multiplication operations.

   For example, suppose the list of integers is:

   2, 3, 4, 1, 2

Then one possible solution is the following, which yields a value of 12:

$$(2 \times 3) + 4 + (1 \times 2)$$

Another possible solution is the following, which yields a higher value of 16 (so this solution is better):

$$2 + (3 \times 4) + (1 \times 2)$$

However, note that the following is **NOT** a valid solution (because there are two multiplications next to one another):

$$(2 \times 3) \times 4 + (1 \times 2)$$

The problem is to decide what operation (+ or ×) you should insert between each integer such that the expression value is maximized.

(a) (1 point) To test that you understand this problem, find the maximum product-sum for the following sequence:

1, 5, 2, 6, 3

(b) (3 points) Propose a **naive** solution that solves this problem for any sequence of length ***n***, and determines its asymptotic **running time** in terms of ***n***.

(c) (3 points) A friend suggests you should use a greedy approach where the greedy choice is to insert a × between the next largest integer and the larger of its two neighbors, if possible (that is if this would not cause two consecutive ×). For instance, for the above example, this would result in the following solution:

$$2 + (3 \times 4) + (1 \times 2)$$

Prove that this greedy choice will **NOT WORK** by providing a counterexample (a sequence of integers for which this greedy solution is sub-optimal).

(d) (3 points) You decide to solve this problem using dynamic programming. Let $V[j]$ be the max product for the first j elements in the sequence. The goal is to find $V[n]$. Complete the following recursive formulation for this problem.

$$V[j] = \begin{cases} 0 & \text{if } j = 0 \\ x_1 & \text{if } j = 1 \\ \max\{V[j-1] + \_\_\_\_\_ \ , \ V[j-2] + \_\_\_\_\_\} & \text{if } j \geq 2 \end{cases}$$

(a) (4 points) Write a bottom-up or top-down dynamic programming algorithm (in pseudocode) using the above recursive formulation. Find its asymptotic running time.