York University
LE/EECS 3101, Fall 2023
Assignment 4

Due Date: November 24th, at 11:59pm

Beware that, when fighting monsters, you yourself do not become a monster; for when you gaze long into the abyss, the abyss gazes also into you ...

Friedrich Nietzsche

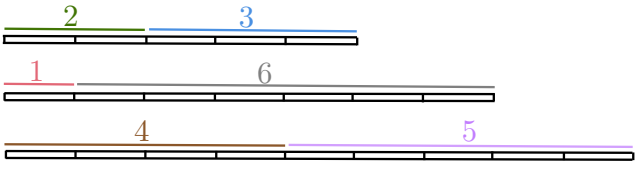
All problems are written problems; submit your solutions electronically **only via Crowdmark**. You are welcome to discuss the general idea of the problems with other students. However, you must write your answers individually and mention your peers (with whom you discussed the problems) in your solution. Please refer to the course webpage for guidelines on academic integrity.
The assignment includes a bonus question. This question is more complex and will be marked more strictly. Approach the bonus only if you have completed other questions.

Problem 1 The Rope Problem [6 marks]

We are given three ropes with lengths n_1, n_2, and n_3. Our goal is to find the smallest value k such that we can fully cover the three ropes with smaller ropes of lengths $1, 2, 3, \ldots, k$ (one rope from each length). For example, as the figure below shows, when $n_1 = 5, n_2 = 7$, and $n_3 = 9$, it is possible to cover all three ropes with smaller ropes of lengths $1, 2, 3, 4, 5, 6$, that is, the output should be $k = 6$.

Devise a dynamic-programming solution that receives the three values of n_1, n_2, and n_3 and outputs k. It suffices to show Steps 1 and 2 in the DP paradigm in your solution. In Step 1, you must specify the subproblems, and how the value of the optimal solutions for smaller subproblems can be used to describe those of large subproblems. In Step 2, you must write down a recursive formula for the minimum number of operations to reconfigure.

Hint: You may assume the value of k is guessed as k_g, and solve the decision problem that asks whether ropes of lengths n_1, n_2, n_3 can be covered by ropes of lengths $1, 2, \ldots, k_g$. Provided with an algorithm for the decision problem, you can find the best value of k using a binary search to find the smallest k_g for which the answer to the decision problem is 'yes'.



Problem 2 String Reconstruction [6 marks]

Given two strings S of length n and T of length m, we want to *reconstruct* T from S with a minimum number of the following operations: i) delete a character from S, ii) insert a character into S, or iii) swap two consecutive characters in S. For example, when $S = yorkun$ and $T = ayokru$, one can insert an 'a' at index 1 of S, swap 'r' and 'k' in S, and remove 'n' from the end of S. Therefore, we can reconstruct T from S with 3 operations.

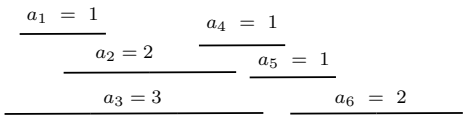
Devise a DP solution to report the minimum number of operations to reconfigure S to T. The actual operations in a minimum-cost reconfiguration are not needed. It suffices to show Steps 1 and 2 in the DP paradigm in your solution. In Step 1, you must specify the subproblems, and how the value of the optimal solutions for smaller subproblems can be used to describe those of large subproblems. In Step 2, you must write down a recursive formula for the minimum number of operations to reconfigure. Assume indices start at 1.

Problem 3 Distant Max-Product Subsequence [6 marks]

Given a sequence A formed by n positive numbers and a positive integer d, we are interested in a distant max-product subsequence (MPS) of A, which is a subsequence of A formed by elements whose indices are at least d units apart and have the maximum product. Describe a dynamic programming algorithm that reports the product of the MPS of A.
For example, if $A = [2, 10, 12, 9, 1, 3, 5]$ and $d = 2$, the output should be $10 \times 9 \times 5 = 450$. In your solution, it suffices to complete the first two steps of the DP algorithm. That is, define subproblems, describe the optimal for a larger subproblem as a function of the optimal solution for smaller subproblems, and write a recursive formula for the optimal value of a subproblem (remember to include the base case). Assume the indices start at 1.

Problem 4 Activity Selection by Length [6 marks]

Consider a variant of the activity-selection problem which asks for maximizing the **product of the length** of selected activities instead of the number of selected activities. For example, for the activities in the example below, an optimal solution selects activities a_3 and a_6 with value $3 \times 2 = 6$.



a) Devise a DP program to solve this variant of the activity selection problem. It suffices to complete the first two steps of the DP algorithm. That is, define subproblems, describe the optimal value for a larger subproblem as a function of the optimal value for smaller subproblems, and write down a recursive formula for the optimal value of a subproblem (remember to include the base case). As before, assume activities are sorted by their finish time. You may use $pred(i)$ to denote the index of the last interval that finishes before a_i starts (e.g., $pred(a_6) = a_4$ in the above example). Assume indices start at 1, and let $pred(i) = 0$ if no activity ends before a_i.

b) Emma suggests the following greedy algorithm for the problem: repeatedly select the longest interval that does not overlap previously-selected intervals. Prove or disprove whether Emma's algorithm is optimal. You must either prove that the problem admits greedy-choice property or present a counter-example in which Emma's algorithm is not optimal.

Problem 5 Huffman Encoding [6 marks]

a) Apply the Huffman encoding to encode the following string over alphabet $\Sigma = \{e.h, i, o, p, \_\}$. When selecting two trees with minimum frequencies, break ties by selecting the tree whose leftmost leaf appears earlier in the above ordering of characters in the alphabet. Also, assume the tree with a smaller frequency forms the left child of the merged tree (and in case of equal frequencies, the subtree whose left-most leaf appears earlier in the alphabet forms the left child).

hippie_hipo

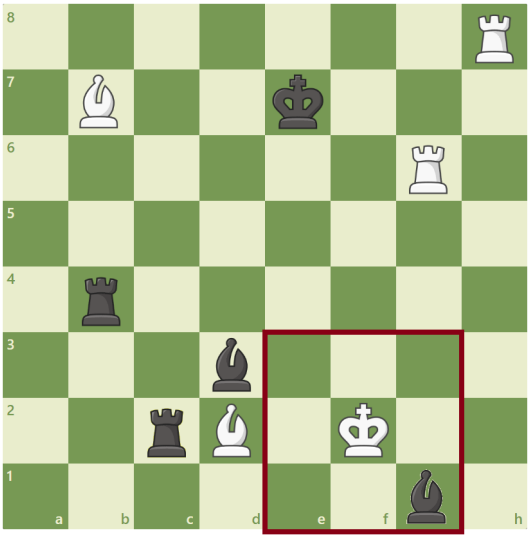
Your solution must include the (final) Huffman tree and its code for the input sequence.

b) Use the same Huffman encoding to decode the following code: 0001110001111101110

Problem 6 Bonus Question [6 marks]

In the year 1352, during the years of the Black Death, Antonius Block played a game of chess with the angle Death[1]. Antonius had white pieces and the Death had black pieces. We want to know the exact configuration of the board at the end of the 30th move. We know that both players had exactly a king, two bishops, and two rooks at that moment of the game (all other pieces were captured). Our tool is to send "queries" back in time to inquire the pieces located in a query box. In the example below, the query returns "the king of Antonios and dark-square bishop of Death". Note that the query answer does not specify the location of the pieces, only what they are.

Use a decision tree approach to derive a lower bound for the number of required to retrieve the exact configuration. Note that there is no n in this problem; your final answer must be an actual integer. Show your steps and line of thought[2].



[1]Check `https://en.wikipedia.org/wiki/The_Seventh_Seal`

[2]If you are not familiar with chess, you can find its basics here: `https://www.chess.com/learn-how-to-play-chess`.