Exercise V, Algorithms 2024-2025

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. There **are many problems on this set, and it is not expected that you solve all of them during two hours**. Instead, train on the topics you find most difficult and also solve more problems at home! Also ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

1 Consider the problem of calculating the value of $f(n)$ where f is defined for non-negative integers recursively as follows:

$$f(n) = \begin{cases} 1 & \text{if } n = 0, \\ \sum_{i=0}^{n-1} f(i) & \text{if } n \geq 1. \end{cases}$$

1a Devise a dynamic programming algorithm to calculate $f(n)$ using the "top-down with memoization" approach.

1b Devise a dynamic programming algorithm to calculate $f(n)$ using the "bottom-up" method.

1c What is the running time of your algorithms?

Rod Cutting

2 (Exercise 15.1-2) Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the ***density*** of a rod of length i to be p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

3 (Based on Exercise 15.1-3) Consider a modification of the rod-cutting problem in which, in addition to a price p_i for each rod, each cut incurs a cost of c. The revenue associated with a solution is now the sum of prices of the pieces minus the costs of making the cuts.

3a Write the optimal revenue r_n of a rod of length n in terms of optimal revenues from shorter rods. In other words, give a recursive formulation of the optimal revenue.

3b Describe how to calculate r_n using the "top-down with memoization" approach. In addition, analyze asymptotically your algorithm's time and space complexity.

3c Describe how to calculate r_n using the "bottom-up" approach. In addition, analyze asymptotically your algorithm's time and space complexity.

3d How would you modify the algorithms to return not only the value but the actual solution too?

Matrix-Chain Multiplication

4 (Based on Exercise 15.2-1) Consider the dynamic programming algorithm for matrix-chain multiplication taught in class. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 10, 3, 12, 5, 50, 6\rangle$. Find the answer (and explain how the algorithm proceeds) by filling in the "memory" table in the same way as the algorithm.

5 (Exercise 15.2-6) Show that a full parenthesization of an n-element expression has exactly $n-1$ pairs of parentheses.

A product of matrices is *fully parenthesized* if it is either a single matrix or the product of two fully parenthesized matrix products, surrounded by parentheses. For example, if the chain of matrices is $\langle A_1, A_2, A_3, A_4\rangle$, then we can fully parenthesize the product $A_1A_2A_3A_4$ in five distinct ways:

$(A_1(A_2(A_3A_4)))$, $(A_1((A_2A_3)A_4))$, $((A_1A_2)(A_3A_4))$, $((A_1(A_2A_3))A_4)$, $(((A_1A_2)A_3)A_4)$.

(Hint: use induction on the number of elements)