Ejercicios en clase: Prog. Dinámica (lista en crecimiento)

Análisis y Diseño de Algoritmos

1 de enero de 2026

Exercise 1. Study the Rod cutting problem from section 15.1 of the Cormen book. After said study, execute the algorithm Bottom-Up-Cut-Rod with the following input: $p = [1; 3; 4; 5; 9; 9; 11; 12; 15]$. Show at each step how each position of the resulting array is filled. Finally, indicate the sequence of cuts that corresponds to the optimal solution obtained by the algorithm.

Exercise 2. Study the algorithm Matrix-Chain-Order (Cormen, section 15.2). After said study, execute the algorithm with the following input: [10; 20; 6; 24; 10; 100; 12]. Show the table resulting from applying the algorithm, as well as the optimal parenthesization obtained after applying it.

Exercise 3. Show the table for the Subset-sum problem seen in class for the items $[2, 3, 2, 4, 8, 9, 5, 7]$ with $W = 10$. It is only necessary to show the final resulting table.

Exercise 4. Show the table for the Min-partition problem seen in class for the array $[23, 2, 1, 2, 18, 22, 14, 21, 6, 5]$ with $k = 5$. It is only necessary to show the final resulting table.

Exercise 5. Design a dynamic programming algorithm to count the number of ways we can construct a sum n, given as input, by throwing a die one or more times. For example, if $n = 3$, there are four ways $(1 + 1 + 1, 1 + 2, 2 + 1, 3)$. You must indicate the recurrence that solves the problem and the running time of your algorithm.

Exercise 6. You have n types of coins available: $c_1, c_2, \ldots, c_n$. Furthermore, you wish to make change for an amount x such that the number of coins used is the minimum possible. For example, if $x = 11$ and you have the coins $1, 5, 7$ then you can change said sum using three coins: $11 = 1 + 5 + 5$. Indicate the recurrence that solves the problem and prove that said recurrence is correct. Based on that, design a dynamic programming algorithm with complexity $O(nx)$ for the previous problem.

Exercise 7. Consider the maximum sum subarray problem: given an array $A[1..n]$ of integers, find a subarray with the largest possible sum.

Let $OPT(i)$ be the value of the maximum sum in a subarray of the array $A[1..n]$ **that ends at** i.

- Find a recurrence for $OPT(i)$.

- Prove that your recurrence is correct using the optimal substructure property.
- Design an $O(n)$ dynamic programming algorithm based on the previous items.

Exercise 8. Consider the following problem. You receive two sequences $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$. You wish to find a sequence of indices $i_1 < i_2 < \ldots < i_k$ such that k is maximized and it holds that $a_{i_1} < a_{i_2} < \cdots a_{i_k}$ but $b_{i_1} > b_{i_2} > \cdots > b_{i_k}$. Indicate the recurrence that solves the problem and prove that said recurrence is correct. Based on that, design a dynamic programming algorithm with complexity $O(n^2)$ for the previous problem.

Exercise 9. You must cut a log into several pieces. The company's best way to do this is with the Analog Cutting Machinery (ACM), which charges according to the length of the log to be cut. Their cutting machine allows only one cut to be made at a time. If we want to make several cuts, it is easy to see that different orders of these cuts lead to many different prices. For example, consider a log 10 meters long, which must be cut at 2, 4, and 7 meters from one of its ends. There are several possibilities. We can first cut at 2 meters, then at 4, and then at 7. This order costs $10 + 8 + 6 = 24$, because the first log was 10 meters long, what remained was 8 meters long, and the last piece was of length 6. If we cut in the order 4, then 2, then 7, we would pay $10 + 4 + 6 = 20$, which is cheaper.

Design a dynamic programming algorithm that, given the length ℓ of the log and k cutting points $p_1, \ldots, p_k$, finds the minimum cost to perform these cuts. Indicate the recurrence used.

Exercise 10. Consider an $n \times n$ board of zeros and ones. A 1 indicates that I can step on said cell of the board and a 0 indicates that it is not possible to step on said cell.

A sequence of cells is considered a path if I can step on said cells and every pair of adjacent cells in the sequence shares a common side. Furthermore, consider that we can only move either to the right or downwards.

The problem consists of calculating the number of paths from cell $(1, 1)$ to cell (n, n). Write a recurrence that allows solving the problem. Prove the correctness of your recurrence. Design a dynamic programming algorithm based on the recurrence.

Exercise 11. Ferries are used to transport cars between one bank of a river and the other. They work by accommodating two lanes of cars along their entire length. Cars enter the lanes from one side of the ferry and exit, on the other bank, on the other side of the ferry.

The waiting line of cars to enter the ferry is a single line and an operator directs each car to either of the two tracks of the ferry, the left lane or the right lane, in order to balance the two tracks. Each car has a different length and an operator decides which of the two lanes each car should board. The goal is to board as many cars as possible (note that it is not possible to skip a car in the line). Design an algorithm that determines the maximum number of cars I can load onto the ferry.

More precisely, let $A[1..n]$ be an array with lengths of n cars, and let ℓ be the length of the ferry. Consider that the waiting list of cars is $A[1], A[2], \ldots, A[n]$. That is, we will first process the car with length $A[1]$, then the car with length $A[2]$, and so on. We want to determine the largest number k such that cars with lengths $A[1], \ldots, A[k]$ can be loaded onto the raft. Design a dynamic programming algorithm for this problem and indicate the recurrence used.

Exercise 12. Suppose we have a sequence $\langle x_1, \ldots, x_{2n+1} \rangle$ in which $\langle x_1, x_3, \ldots, x_{2n+1} \rangle$ are positive integers and $\langle x_2, x_4, \ldots, x_{2n} \rangle$ belong to the set $\{+, *\}$. We want to insert parentheses into the sequence such that the value of the resulting arithmetic expression is maximized. For example, if the given sequence is 2+4*1+5 then one solution is the expression $((2+4)*(1+5))$, which equals 36.

- Find a recurrence that returns the maximum value obtainable after applying parentheses to a sequence. The recurrence must allow solving the problem in $O(n^3)$.
- Prove that your recurrence is correct using optimal substructure arguments.
- Using the recurrence, write the pseudocode of a dynamic programming algorithm that allows solving the problem.

Exercise 13. Suppose you have a machine and a set of n jobs, identified by numbers $1, 2, ..., n$, to process on that machine. Each job j has a processing time t_j, a profit p_j, and a deadline d_j. The machine can only process one job at a time, and the job must be performed uninterruptedly for t_j consecutive time units. If the job is already completed by its deadline d_j, you receive a profit p_j, but if it is completed after its deadline, you receive no profit. Design a dynamic programming algorithm that finds the execution order of jobs that maximizes the sum of profits, assuming all processing times are integers between 1 and n. Indicate the recurrence used. What is the running time of your algorithm?