Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Desenvolvimento e Análise de Algoritmos

2019/2020 — 1º Semestre

Script 4 — Dynamic Programming

Design and test an algorithm, using the Dynamic Programming strategy, for each one of the following problems.

Afterwards, analyze the computational complexity of each one of those algorithms. In order to accomplish that:

a) Perform a sequence of experiments to count, register and analyze the number of basic operations carried out.

b) Perform a formal computational complexity analysis.

c) Compare the results of the experimental and the formal analysis.

1 – Compute the successive terms of the Fibonacci Sequence:

a) Using an array to store the computed terms.

b) Using just 3 auxiliary variables to store the required terms at each step.

2 – Compute the number of combinations $C(n,p)$:

a) Using a 2D array to store the successively computed values.

b) Using just one 1D array.

3 – The Bernstein Polynomials $B^{n,i}(t)$ can be easily defined with a recurrence, for t in [0, 1]. That recurrence can be used to design an algorithm to compute the value of a polynomial $B^{n,i}(t^*)$, for a given value t^*.

Given t^*, compute the value of polynomial $B^{n,i}(t^*)$:

a) Using a 2D array to store the successively computed values.

b) Using just one 1D array.

4 – Consider a robot that can move either 1 meter, or 2 meters or 3 meters in a certain direction. Compute the number of different ways in which the robot can travel n meters.

5 – Compute the least number of coins that represent a given change amount, given the denomination values of the coins that can be used (*"The Coin Changing Problem"*).

6 – Given a set of *n* items to be carried in a knapsack of capacity *W*, determine the (a) most valuable sub-set of items that can carried out in the knapsack (*"The 0-1 Knapsack Problem"*).

Suggestion: For some of the problems, compare the performance of the developed algorithms with the usual **recursive solutions**.