Data Structures and Algorithms

Degree in Computer Science, Group I

First Semester Exam. February 11^{th}, 2016.

1. (3 points) Code and verify an efficient **iterative algorithm** that meets the following specification:

$$\{0 \le n \le \mathit{length}(v) \wedge \forall k : 0 \le k < n : v[k] \ge 0\}$$
$$\textbf{fun } \mathit{evenOdd}(\textbf{int } v[\,], \textbf{int } n) \textbf{ return int } p$$
$$\{p = \#\, i, j : 0 \le i < j < n : v[i]\,\%2 = 0 \wedge v[j]\,\%2 = 1\}$$

Calculate the complexity of your algorithm.

2. (3.5 points) The ISBN code of a book has 13 digits $d_1\ d_2 \ldots d_{13}$. The digit d_{13} is called *control digit* and is calculated using the following procedure: first, we must sum the result of multiplying all the digits in an even position by three and the ones in an odd position by one. The digits are numbered starting from the most significant digit:

$$d_1 * 1 + d_2 * 3 + d_3 * 1 + d_4 * 3 + d_5 * 1 + d_6 * 3 + d_7 * 1 + d_8 * 3 + d_9 * 1 + d_{10} * 3 + d_{11} * 1 + d_{12} * 3$$

The digit d_{13} is the number that, added to the result of the previous sum, results in a multiple of 10. For example, the control digit of the number 336772900480 is 9, because the result of the first sum is 81 and $81 + 9 = 90$, which is a multiple of 10.

Design and code an efficient **recursive algorithm** that calculates the control digit of an ISBN code with any number of digits. The ISBN number is represented as an integer number. For example, if the input number is 2561, the sum is $2 * 1 + 5 * 3 + 6 * 1 + 1 * 3 = 26$ and the result would be 4. Calculate the complexity of your algorithm.

3. (3.5 points) We can use a $N \times N$ matrix M to map a terrain that has been divided in cells, so that $M[i][j]$ is the average height of the cell (i, j).

We need to build a road between two points of this terrain. The road can communicate two adjacent cells (not in diagonal) if the difference between their average heights is $\le h_{max}$, i.e.:

$$\mid M[a][b] - M[c][d] \mid \le h_{max}$$

where cells (a, b) and (c, d) are adjacent.

Code a **backtracking algorithm** that, for a given matrix M, a maximum height h_{max} and two different cells of the terrain, calculates the shortest road (if it exists) that solves the problem. The length of the road is the number of cells it crosses, including both the origin and the destination.

For example, if $h_{max} = 2$ and the matrix M of the terrain is:

$$\begin{pmatrix} 0 & 3 & 4 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

the shortest road from $(0,1)$ to $(0,0)$ has length 4 and crosses the cells $(0,1)$, $(1,1)$, $(1,0)$, $(0,0)$. There are other valid roads, but they are longer, such as $(0,1)$, $(1,1)$, $(2,1)$, $(2,0)$, $(1,0)$, $(0,0)$, which has length 6. A non-valid road would be the one that goes directly from $(0,1)$ to $(0,0)$ since the difference between the heights of both cells is $> h_{max}$.