Assignment 2

Alessandro Artale, Werner Nutt,
Ognjen Savković, Francesco Sportelli

Recursion and Algorithms for Arrays

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") if not stated otherwise.
Please, include name, student ID, code of your lab group (A, B, or C) and email address in your submission.

1. Recursion

A palindrome is a phrase that reads the same forward and backward (examples: 'racecar', 'radar', 'noon', or 'rats live on no evil star'). By extension we call every string a palindrome that reads the same from left to right and from right to left.
Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome.
Implement your algorithm in Java. Provide the argument for the test run in a global variable.

(6 Points)

2. Maximal Length of Ascents in Arrays

Consider an array $A[1..n]$ of integers. A *subarray of* A is a contiguous segment of A. We denote the subarray from position k to position l as $A[k..l]$.
The subarray $A[k..l]$ is an *ascent* if $A[j] \leq A[j+1]$ for all j where $k \leq j < l$. In other words, an ascent is a nondecreasing segment of A.
We want to compute the maximal length of an ascent in A. For instance, for the array $A = [3, 1, 4, 2, 4, 4, 5, 3]$, the maximal length of an ascent would be 4, because the subarray $A[4..7] = [2, 4, 4, 5]$ is the longest ascent in that array.
We are interested in an *efficient* algorithm.

1. Write down pseudocode for an iterative algorithm `maxAscent`(A) that takes an array A of integers as input and returns the maximal length of an ascent in A.

2. Explain why your algorithm is correct.

3. Implement the algorithm in Java. Provide the argument for the test run in a global variable.

(10 Points)

3. Array of Averages

Design an *efficient* algorithm that achieves the following task: Given an array $A[1..n]$ of floating point numbers, it returns a two-dimensional array, say M, of size $n \times n$ in which the entry $M[i][j]$ for $i \leq j$ contains the *average* of the array entries $A[i]$ through $A[j]$. That is: if $i \leq j$, then

$$M[i][j] = \frac{A[i] + \cdots + A[j]}{j - i + 1},$$

whereas for $i > j$ we have that $M[i][j] = 0$.

1. Describe the algorithm that creates this matrix in pseudocode.

2. What is the running time of your algorithm with respect to the array size n?

3. Implement the algorithm in Java.

4. Measure the running time of your Java program for random inputs of size $n = 10$, 100, 1000, etc. Does the growth of the running time correspond to your estimates?

(14 Points)

Submission: Until Mon, 16 March 2015, 11:55 pm, to

`dsa-submissions AT inf DOT unibz DOT it`.

If you want to submit a hand-written solution, scan it and send it to the email address above.