Aalto University
Department of Information and Computer Science
Pekka Orponen

T-79.4202 Principles of Algorithmic Techniques (5 cr)
Exam Thu 18 Dec 2014, 1–4 p.m.

Write down on each answer sheet:
- Your name, degree programme, and student number
- The text: "T-79.4202 Principles of Algorithmic Techniques 18.12.2014"
- The total number of answer sheets you are submitting for grading

Note: You can write down your answers in either Finnish, Swedish, or English.

1. Design an algorithm whose running time, constant factors notwithstanding, is described by the recurrence equation

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 4T(n/2)+n^2, \qquad \text{for } n = 2^k, \quad k \geq 1. \end{aligned}$$

The algorithm receives as input an n-element array $A[1..n]$, but otherwise it does not matter what the algorithm actually does. Determine the order of growth of the solution to the recurrence, when n is a power of two. *12p*

2. The *Eulerian number* $\left\langle {n \atop k} \right\rangle$ indicates how many permutations π of the set $\{1,\ldots,n\}$ contain exactly k *ascents*, i.e. positions i such that $\pi(i) < \pi(i+1)$. These numbers satisfy the recurrence:

$$\begin{aligned} \left\langle {n \atop 0} \right\rangle &= 1, \quad \text{for } n \geq 0, \\ \left\langle {n \atop k} \right\rangle &= (k+1)\left\langle {n-1 \atop k} \right\rangle + (n-k)\left\langle {n-1 \atop k-1} \right\rangle, \quad \text{for } n \geq k \geq 1, \\ \left\langle {n \atop k} \right\rangle &= 0, \quad \text{for } n < k. \end{aligned}$$

Design a reasonably efficient algorithm, based on this recurrence, for computing the number $\left\langle {n \atop k} \right\rangle$. Determine the time complexity of your algorithm. *14p*

3. Design a linear-time algorithm for the following task: given a connected undirected graph G, find a vertex v that can be removed from G without making it disconnected. (*Hint.* Think about other linear-time graph algorithms that you know.) *14p*

4. The *diameter* of a tree is the length of the longest path contained in it.[1] Design a linear-time algorithm for determining this quantity. Justify the time-complexity claim of your algorithm. (*Hint.* One possibility is to use a divide-and-conquer approach.) *14p*

[1] Recall that a *tree* is an acyclic connected graph.