Aalto University, Department of Computer Science
Pekka Orponen

CS-E3190 Principles of Algorithmic Techniques (5 cr)
Exam Tue 19 Dec 2017, 1–4 p.m.

Write down on each answer sheet:
- Your name, degree programme, and student number
- The text: "CS-E3190 Principles of Algorithmic Techniques 19.12.2017"
- 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. Arrange the following functions according to their increasing order of growth:

$$\sqrt{n},\quad n\log n,\quad n^{1/3}+\log n,\quad \log n,\quad (2e)^n,\quad n/\log n,$$
$$n!,\quad 42,\quad \log\log n,\quad 2^{2n},\quad 1/n,\quad n^{-2}+(\log n)^2.$$

   (Notation $\log n$ denotes here logarithm in base 2.) You do not need to prove the correctness of your ordering. *12p*

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

$$\begin{aligned} T(1) &= 1, \\ T(n) &= 2T(n/2)+n\log_2 n, \quad \text{for } n=2^k,\ k=1,2,\ldots \end{aligned}$$

   The algorithm receives as input an n-element array $A[1\ldots n]$, 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. (Note that the "Master Theorem" does not apply here because of the form of the additive term, so you will need to solve the recurrence directly.) *12p*

3. A certain programming language P provides a primitive operation $cut(L,m)$, whereby a list (a linear array) $L[1\ldots n]$ of n elements is partitioned into two new lists $L_1 = L[1\ldots m]$ and $L_2 = L[(m+1)\ldots n]$, assuming that $1 \le m < n$. The lists L_1 and L_2 are created by copying the respective elements of L in two new arrays, and so the cost of an operation $cut(L,m)$ is proportional to $|L| = n$, the length of L, independent of the value of m.

   Suppose now that you want to extend P by an operation $cut(L;m_1,\ldots,m_k)$, which partitions L into $(k+1)$ pieces $L[1\ldots m_1]$, $L[(m_1+1)\ldots m_2]$, …, $L[(m_k+1)\ldots n]$. The total cost of achieving this depends on your chosen ordering of the individual two-way cuts. (Consider e.g. implementing $cut(L;3,7)$, where L is a 20-element list. Making the cuts in order $(L_1,T) \leftarrow cut(L,3)$; $(L_2,L_3) \leftarrow cut(T,4)$ has cost 20+17 = 37, whereas making them in order $(S,L_3) \leftarrow cut(L,7)$; $(L_1,L_2) \leftarrow cut(S,3)$ has cost only 20+7 = 27.)

   You of course want to be as efficient as possible, so please design a preprocessing scheme which determines the optimal ordering of the pairwise cuts to implement $cut(L;m_1,\ldots,m_k)$, where $|L| = n$, and runs in time $O(k^3)$. For the present problem it suffices to determine the *cost* of an optimal ordering, the actual ordering does not need to be presented. However, please justify the runtime of your algorithm. [*Hint:* Set $m_0 = 0$, $m_{k+1} = n$, and consider the values $C(i,j)$ = optimal cost of partitioning the sublist $L[(m_i+1)\ldots m_j]$, where $i < j$.] *15p*

4. Design an algorithm that computes the length of the shortest cycle in a given connected undirected graph $G = (V,E)$, or reports that G is acyclic. (The length of a cycle is taken to be the number of edges contained in it. This concept is also called the *girth* of the graph.) Your algorithm should run in time $O(|V|\cdot|E|)$, and please justify this. [*Hint:* Think about how you would compute shortest distances. Draw some small example graphs.] *15p*