Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute for Theoretical Computer Science
Dr. B. Gärtner, Dr. M. Hoffmann and Marek Sulovský

18.9.2008

Computational Geometry — Exercise Set 1 — HS08

URL: http://www.ti.inf.ethz.ch/ew/courses/CG08/

Exercises

Every week we will hand out an exercise sheet with a complementary material to the lecture. You are advised to solve them and may hand them in to the assistant for corrections and suggestions.

There will be four special series of exercises, which will be obligatory and graded. Three best grades from exercises will contribute to the final grade 10% each.

Exam

There will be an oral exam of 15 minutes during the examination period. Your final grade consists to 70% of the grade for the exam and to 30% of the grades for the exercises.

Remark: Throughout the course we will be using asymptotic notation when analyzing algorithms. In this exercise we want to make sure you are familiar with it. Throughout the rest of this remark, all functions are $f, g : \mathbb{N} \to \mathbb{R}$.

We denote

$$O(g(n)) = \{f(n) \mid \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0\}.$$

and similarly

$$\Omega(g(n)) = \{f(n) \mid \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq c \cdot g(n) \leq f(n) \text{ for all } n \geq n_0\}.$$

Finally

$$\Theta(g(n)) := O(g(n)) \cap \Omega(g(n))$$

Throughout the rest of the exercise, we will denote a base 2 logarithm by log. We define functions iterated logarithm $\log^{(i)}$ (for $i \in \mathbb{N}$)

$$\log^{(i)} n := \begin{cases} n & , i = 0 \\ \log \log^{(i-1)} n & , \text{otherwise} \end{cases}$$

and

$$\log^* n := \min\{i \geq 0 \mid \log^{(i)} n \leq 1\}$$

which essentially determines, how many times does a logarithm have to be applied until we reach 1.

Exercise 1

Order the following functions by their order of growth, i.e., into a sequence $g_1, \ldots, g_{15}$ s.t. $g_i \in O(g_j)$ for $i \leq j$.

$2^{\log^* n}$	n^2	$n!$	$(\log n)!$	$\log^* \log n$
n^{72}	$\log \log^* n$	$4^{\log n}$	1	$(\log n)^{\log n}$
$(\frac{n}{3})^n$	$n^{\log \log n}$	$\log n$	e^n	$n \log n$

Exercise 2

Determine the order of magnitude of

$$\sum_{i=1}^{n} \frac{1}{i}$$

Remark: Let us have a recurrence relation for function $T(n)$ of the form

$$T(n) = aT(n/b) + f(n)$$

Then the asymptotic growth of the function $T(n)$ can be described as follows:

1. If $f(n) = O(n^{(\log_b a)-\epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.

2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.

3. If $f(n) = O(n^{(\log_b a)+\epsilon})$ for some $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

Exercise 3

Determine the order of growth of the following recursively defined functions:

1. $R(n) = 9R(n/3) + n$
2. $S(n) = S(2n/3) + 1$
3. $T(n) = 3T(n/4) + n/\log n$

Exercise 4

Consider a recursive binary search algorithm for finding a number in a sorted N-element array (compare your value to the middle element of the sorted array and accordingly search recursively in the left or right half of the subarray) depending on a parameter-passing strategy:

1. An array is passed by a pointer (time $\Theta(1)$).
2. An array is passed by copying (time $\Theta(N)$ where N is the size of the whole array).
3. An array is passed by copying the relevant part of the array (time $\Theta(q - p + 1)$ if array from index p to index q is passed).