AD1 – Algorithms and Data Structures I (course 1DL210)

Assignment 1: Algorithm Analysis

Due by 23:59:59 on Friday 18 April, 2008

The objectives of this assignment are (1) to understand the notation used in algorithm complexity analysis, as well as (2) to get a sense of the steepness of various common complexity functions. (The writing and solving of recurrences will be exercised on programs in the next assignments.)

Problem 1: Understanding the Notation $(8 \cdot 3 = 24$ points)
Compute values of n_0, c_1, and c_2 that can be used to establish the following relationships:

a. $\log_2 n = \Omega(1)$

b. $\sqrt{n} = \Omega(\log_2 n)$

c. $n^{0.9} = O(n)$

d. $n \cdot \log_2 n + n = \Theta(n \cdot \log_2 n)$

e. $999 \cdot n = O(n^2)$

f. $19 \cdot n^3 - 13 \cdot n^2 + 106 \cdot n - 77 = \Theta(n^3)$

g. $2^{n+10} = \Theta(2^n)$

h. $n^{10} = O(n!)$

Exhibit your reasoning; otherwise you get zero points for correct values.

Problem 2: Waiting for Faster Hardware? $(8 \cdot (3 + 1.5) = 36$ points)
Complete Table 1, which shows the sizes of problem instances that can be solved in one hour of CPU time on three **different computers** using algorithms of the given time complexity functions. So, for example, if the algorithm has linear time complexity and one can solve a problem instance of size C on a present-day computer in one CPU hour, what is the size of the largest problem instance that one can solve in one CPU hour with a computer whose CPU is 100 or 1000 times faster? Two digits of decimal precision suffice. Exhibit your reasoning; otherwise you get zero points for correct expressions.

time complexity function	size with present-day computer	size with computer 100 times faster	size with computer 1000 times faster
1	A		
$\log_2 n$	B		
n	C		
$n \cdot \log_2 n$	D		
n^2	E		
n^3	F		
2^n	G		
$n!$	H		

Table 1: Size of largest problem instance solvable in one hour of CPU time

Problem 3: Coping with Existing Hardware? $(8 \cdot (2+2+1) = 40$ points) Complete Table 2, which shows the CPU times by the **same computer** on three problem instances of different sizes using algorithms of the given time complexity functions. Give the times in the largest unit among microseconds, milliseconds, seconds, minutes, days, years, and centuries, so that the amount is at least 1. Two digits of decimal precision suffice. Only consider years of 365 days. Assume that the CPU performs a million operations per second. Exhibit your reasoning; otherwise you get zero points for correct values.

time complexity function	$n = 10$	$n = 50$	$n = 1000$
1			
$\log_2 n$			
n			
$n \cdot \log_2 n$			
n^2			
n^3			
2^n			
$n!$			

Table 2: CPU times as the problem instance size increases

Hand in your solutions to these problems as a text, HTML, or PDF file. In a text or HTML file, write `Omega` for Ω, and `O` for O, and `Theta` for Θ, and `log_b n` for $\log_b n$, and `n^a` for n^a, and `sqrt(n)` for $\sqrt{n}$.

Reflect on the consequences of your solutions. On a given computer, the impact of a better algorithm can be seen by moving up the columns in Tables 1 and 2. The impact of faster computers can be observed by moving from left to right across the rows in Table 1. The impact of larger problem instances can be observed by moving from left to right across the rows in Table 2.

Have fun!