Exercises for Tutorial 1

Carl Leonardsson    Jari Stenman
2013-09-06

Exercises

1. For each of the following claims, decide whether it is true or false, and argue for your position. Some claims are marked (*). For these claims, if you argue that they are true, then find values for the constants in the definition of $O$, $\Theta$, $\Omega$ (i.e. constants $c$, $c_1$, $c_2$, $n_0$).

(a) $f(n) = \Omega(2f(n))$
(b) $2n = O(n^2)$
(c) $n^2 + n = O(n^2)$ (*)
(d) $n^3 = O(1000n^2)$ (*)
(e) $n \log n = \Theta(n)$ (*)
(f) $n + \log n = \Theta(n)$ (*)
(g) $n \log n = O(n^2)$
(h) $f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$
(i) $f(n) = \Theta(g(n))$ and $g(n) = \Theta(f(n))$ implies $f(n) = g(n)$
(j) $f(n) = g(n)$ implies $f(n) = \Theta(g(n))$ and $g(n) = \Theta(f(n))$
(k) $f(n) = \Theta(n^k)$ assuming $f(n)$ is a $k$:th degree polynomial.

2. Can you find a function $f(x)$ such that $2 + \sin(x) = O(f(x))$? How about $2 + \sin(x) = \Theta(f(x))$ and $2 + \sin(x) = \Omega(f(x))$?

3. Use induction to prove that $T(n) = 2^n + 1$ for all $n \geq 0$ where

$$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2^n + T(n-1) & \text{if } n > 0 \end{cases}$$

4. Guess a closed form for $T(n)$, and use induction to prove that it coincides with the recursive definition for $n \geq 0$:

$$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1) + n - 1000 & \text{if } n > 0 \end{cases}$$

**Hint:** To guess a closed form, remember the induction examples from the lectures.
5. Consider sorting \( n \) numbers stored in array \( A \) by first finding the smallest element of \( A \) and exchanging it with the element in \( A[1] \). Then find the second smallest element of \( A \), and exchange it with \( A[2] \). Continue in this manner for the first \( n-1 \) elements of \( A \). Write pseudocode for this algorithm, which is known as selection sort. Why does it need to run for only the first \( n-1 \) elements, rather than for all \( n \) elements? Give the best-case and worst-case running times of selection sort in \( \Theta \)-notation. [CLRS 2.2-2 (2nd ed)]

6. What loop invariant does selection sort maintain? [CLRS 2.2-2 (2nd ed)]

7. How can we modify almost any algorithm to have a good best-case running time? [CLRS 2.2-4 (2nd ed)]

**Definitions of O, \( \Theta \), \( \Omega \) notation**

From the lecture notes:

- **O-notation**
  For a given function \( g(n) \), we denote by \( O(g(n)) \) the set of functions \( f(n) \) such that there are constants \( c > 0 \) and \( n_0 > 0 \) such that:
  \[
  0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0
  \]

- **\( \Theta \)-notation**
  For a given function \( g(n) \), we denote by \( \Theta(g(n)) \) the set of functions \( f(n) \) such that there are constants \( c_1 > 0 \), \( c_2 > 0 \), and \( n_0 > 0 \) such that:
  \[
  0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ for all } n \geq n_0
  \]

- **\( \Omega \)-notation**
  For a given function \( g(n) \), we denote by \( \Omega(g(n)) \) the set of functions \( f(n) \) such that there are constants \( c > 0 \) and \( n_0 > 0 \) such that:
  \[
  0 \leq c \cdot g(n) \leq f(n) \text{ for all } n \geq n_0
  \]

Recall that \( f(n) = O(g(n)) \) is common notation abuse for \( f(n) \in O(g(n)) \). Similarly for \( \Theta \) and \( \Omega \).