Assignment 3

Julien Corman, Werner Nutt
Flavio Vella

Asymptotic Complexity and Substring Matching

1. Comparison According to Asymptotic Complexity

Below you find a list of functions that could appear as functions describing the running time of algorithms:

1. $n^{3/2}$
2. $8^{\log_2 n}$
3. $2 \cdot 3^n$
4. $3 \cdot n^2$
5. $3 \cdot 2^n$
6. $3 \cdot n^2 + 2 \cdot n^3$
7. $n^{\log_2 7}$
8. $\log_2 n^n$.

Order the following functions according to their asymptotic complexity. Start with the function having the smallest asymptotic complexity and move on to the function having the next largest one. That is, write them as a sequence $f_1, f_2, \ldots, f_9$ such that $f_1(n) = O(f_2(n))$, $f_2(n) = O(f_3(n))$, etc.
Indicate as well functions f, g that are asymptotically equivalent, that is, where both $f(n) = O(g(n))$ and $g(n) = O(f(n))$ hold.

Hint: Consult the "Toolbox to Determine Big-Oh, Omega, and Theta" on OLE.

(Weight: 30% of this CW)

2. Asymptotic Equalities

Prove or disprove the following statements:

a) $8n + n \cdot \log_2 n = O(n)$

b) $(n + a)^b = \Omega(n^b)$ for all real numbers $a, b > 0$

c) $n^a = \Theta(n^b)$ if and only if $a = b$ for all real numbers $a, b > 0$

Hint: Consult the "Toolbox" also for this exercise.

(Weight: 20% of this CW)

3. Substring Matching

Write a substring matching algorithm that satisfies the following property:

> Given two character strings, string A of length n and string B of length m, the algorithm returns -1 if B is *not* a substring of A, and returns the start position s of B in A if B is a substring of A.

For example, if $A =$ "`assignment`" and $B =$ "`sign`", then the algorithm should return the number 2 (assuming that the first index of the array is 0). Develop your algorithm from first principles, by treating strings as arrays of characters, that is, use only the methods `length()` and `charAt()` to access characters in a string.
Proceed as follows:

a) Write down the idea of your algorithm.

b) Identify all relevant special input cases for your algorithm and write down examples of the input for each case and for the output you expect.

c) Develop the algorithm in pseudocode. Make sure that your algorithm is valid for the special cases you have identified.

d) Write JUnit tests for the special cases you have identified. Use as template the test available on the Codeboard project for this assignment.

e) Implement the algorithm and test the implementation with your JUnit tests.

(Weight: 50% of this CW)

Deliverables. For Question 3, submit two copies of your code (and tests):

- one via Codeboard (instructions are available here),
- one via the OLE submission page of your lab (together with the other deliverables).

The other questions must be answered in a PDF document.
Combine all deliverables into one zip file, which you submit via the OLE submission page of your lab. Please include name, student ID and email address in your submission.

Submission until Thursday, 15 November 2018, 23:55, to Codeboard and the OLE submission page of:

Lab A / Lab B