

## 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  $f_1, f_2, \dots, f_9$  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 for Asymptotic Analysis” in the OLE section of this assignment.

(8 Points)

## 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  $a, b > 0$

(5 Points)

## 3. Substring Matching

Write a substring matching algorithm that satisfies the following specification:

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  $A = \text{"assignment"}$  and  $B = \text{"sign"}$ , then the algorithm should return the number 3. 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) Download the file `DSA_A3.zip` and unzip it. Implement JUnit tests for the special cases you have identified.
- e) Implement the algorithm in Java and test the implementation with your JUnit tests.

(15 Points)

**Deliverables.**

1. Write one report for all questions.
2. For Question 3, hand-in two Java files, one containing the tests and one the implementation.

Please, combine all deliverables into one zip file, which you submit via the OLE website of the course.

Submission: Until Wed, 4 April 2016, 23:55 hrs, to the OLE submission page of

Lab A / Lab B / Lab C