University of Konstanz
Department of Computer & Information Science
Prof. Dr. Ulrik Brandes / Dr. Sven Kosub / Dr. Sabine Cornelsen and Natalie Indlekofer

Design and Analysis of Algorithms
WS 2012/2013

Assignment 8

Post Date: 12 Dec 2012 **Due Date:** 19 Dec 2012, 14:30
You are permitted and encouraged to work in groups of two.

Problem 1: FIFO Vertex Selection Rule — 8 Points

Show that the run time of the algorithm of Goldberg & Tarjan applied to a flow network with n vertices is in $\mathcal{O}(n^3)$ if it is implemented by using the FIFO vertex selection rule as follows: First all vertices that are activated during the initialization are appended to a queue. While the queue is not empty the algorithm removes the first vertex v from the queue, performs Push-operations from v and appends newly active vertices to the queue. The algorithm examines v until it is not active any more or a Relabel-operation is performed. In the latter case v is appended again to the queue.

Hint: Partition the vertex examinations into phases. The first phase consists of examinations of vertices that become active during the initialization. The $(i+1)$st phase consists of examinations of vertices that were appended to the queue during the ith phase. Use the potential function

$$\Phi = \max_{v \text{ active}} h(v)$$

to show that there are at most $\mathcal{O}(n^2)$ phases.

Problem 2: Naive String Matching — 6 Points

(a) Suppose that all characters in the pattern are different. Show how to accelerate the naive string-matching algorithm such that it runs in $\mathcal{O}(n)$ time on an n-character text.

(b) Find a text and a pattern such that the naive string-matching algorithm needs $\Theta((n-m+1)\cdot m)$ time even if the pattern does not occur in the text.

(c) Suppose that pattern P and text T are *randomly* chosen strings of length m and n, respectively, from the d-ary alphabet $\Sigma = \{0, 1, \ldots, d-1\}$, where $d \geq 2$. Show that the *expected* number of character-to-character comparisons in every step made by the naive string-matching algorithm is

$$\frac{1-d^{-m}}{1-d^{-1}} \leq 2.$$

[please turn over]

Problem 3: Wildcards **6 Points**

Now, a pattern can contain also *wildcards* $*$. A wildcard $*$ can stand for arbitrarily many (also zero) characters.

(a) Modify the algorithms `Naive-Transition-Function` and `Finite-Automaton-Matcher` such that they also work for patterns that may contain wildcards. Explain your approach.

(b) Give the string-matching automaton for the pattern $P = aba{*}bab$ and the input alphabet $\Sigma = \{a, b, c\}$. Does this automaton find all occurrences of pattern P in a text?