Algorithm Design and Complexity – Exam
2013, January 24th
SUBJECT A

TIME: 90 minutes
POINTS: 40 points + 6points bonus

A. Theory (20p)

A1. Explain which are the main reasons for using the Θ (theta) notation for expressing and for comparing the complexity of different algorithms.

A2. When can we use dynamic programming to solve a problem? Exemplify the main steps in building a dynamic programming solution for the chain matrix multiplication problem (briefly, no pseudocode is needed).

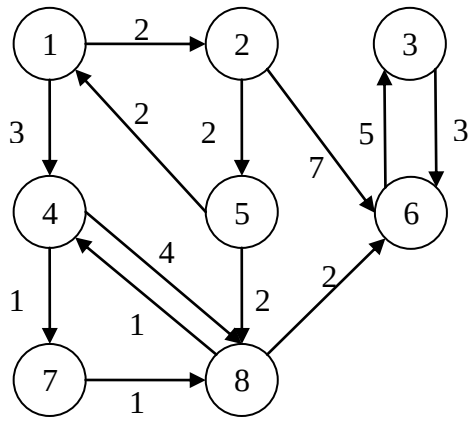
A3. Which are the most important classes of problems if we know that "P != NP" ? Explain what each class means in 1-3 lines and give an example of a problem that is part of that class.

A4. Which are the types of edges that can be identified by using a depth first search on a directed graph? Show how each type of edge can be detected using the color of the vertices!

A5. Explain how the Floyd-Warshall algorithm works by highlighting the recursive relation. What is it used for and what complexity does it have?

B. Exercices (12p + 6p bonus)
(6+3p) B1. Solve the following recurrences (any two from below, the third one is a bonus exercise):

a. $T(n) = 8 * T(n/2) + n^3 * \log n$
b. $T(n) = T(n/2) + T(n/4) + n$ *(Hint! Use recursion trees)*
c. $\Omega(n^2) + n^3 = \ldots ?$ (*Find an asymptotic margin as strong as possible*)



(6+3p) B2. Considering the graph on the left, show how the following algorithms operate (any two from below, the third one is a bonus exercise):

a. Dijkstra's algorithm from node 1, by pointing out the nodes in the queue (Q) at each step, as well as the values of the distances array d;
b. Kosaraju's or Tarjan's algorithm to determine the SCCs (for this task consider the graph **without weights**, you need to show the results of running DFS on G and on G^T);
c. Show how Kruskal's algorithm works by considering the graph **undirected** in order to build a minimum spanning tree.

C. Problem (8p)

C1. You are the king of a shire and, of course, you know all the cities and all the roads connecting them (however, you do not care about the length of these roads). You would like to find out **all the paths** that start from a city **S** and go to another city **D** and which contain **the smallest number of intermediate vertices**.
Write your idea for solving the problem, the pseudocode, the complexity and the correctness of your solution! Try to find the optimal solution!