Exam II Algorithms and Networks 2018/2019

This is the exam for part II of Algorithms and Networks.

You have three hours for the exam.

You may give your answers in Dutch or in English.

Switch off your mobile phone. Use of your mobile phone during the exam is strictly forbidden.

Write clearly.

You may consult four sides of A4 with notes.

Results used in the course or exercises may be used without further proof, unless explicitly asked.

Results asked in an earlier part of an assignment may be used as without proof in a later part, even when you were unable to solve that earlier part.

There are five questions, each will give you the specified amount of points.

Some parts are harder than others: use your time well and make sure you first finish the easier parts!

$\mathbf{N} = \{0, 1, 2, 3, \ldots\}$.

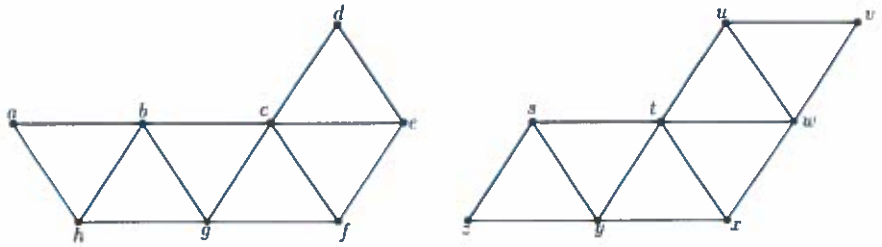
Important: Use separate sheets of paper for your answers to questions 1, 2 and 3, and for your answers to questions 4 and 5. You may use more than one sheet of paper for a collection. Failure to do so may result in having your work graded later.

Unless stated otherwise, n denotes the number of vertices of graph G, and m denotes the number of edges or arcs.

1. Graph isomorphism (1.5 points)

(a) **Are the following two graphs isomorphic? (Answer YES or NO.)**

(b) **If your answer is YES, give the isomorphism. If your answer is NO, give a compact proof why these graphs are not isomorphic.**



2. 1-outdegree orientable graphs (2.5 = 1+1+0.5 points)

Let G be an undirected graph. We say that G is *1-outdegree orientable*, if we can give each edge in G a direction, such that each vertex has outdegree at most 1. (See the pictures below. The undirected graph at the left side is 1-outdegree orientable; at the right side, we have given each edge a direction, such that each vertex has outdegree 1.)

(a) **Show that if each vertex in an undirected graph G has degree at least 3, then G is not 1-outdegree orientable.**

(b) **Show that we can decide in polynomial time if a given undirected graph $G = (V, E)$ is 1-outdegree orientable.**

Hint: look at vertices of small degree.

(c) **Show that we can decide in $O(n)$ time if a given undirected graph G is 1-outdegree orientable.**

3. List coloring on binary trees (2 points)

Suppose we are given a rooted tree $T = (V, E)$, with root r, such that each vertex has at most two children. In addition, we are given a finite set C of *colors*, and for each vertex $v \in V$, we are given a subset $L(v) \subseteq C$ of *allowed colors*. Given is also that the number of colors $|C|$ is bounded by a constant.

A *proper list-coloring* of T is a function $f : V \to C$, such that

(1) For each $v \in V$, $f(v) \in L(v)$. (Each vertex gets an allowed color.)

(2) For each edge $\{v, w\} \in E$, $f(v) \neq f(w)$.

Give an algorithm that decides in $O(n)$ time, given T and the list $L(v)$ for all vertices $v \in V$, whether there exists a proper list-coloring of T.

(You must give the algorithm explicitly here.)

4. An Exact Exponential-Time Algorithm for Exact 3-Satisfiability (2 points)

The *Exact 3-Satisfiability* problem is defined as follows:

EXACT 3-SATISFIABILITY
Given: Given a set of boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a set of clauses $C = \{C_1, C_2, \ldots, C_m\}$ using literals from the variables in X where each clause C_i has size at most 3.
Question: Does there a truth assignment to the variables in X such that for every clause in C_i we have that *exactly one* literal in the clause is set to *True*, i.e., one literal is set to *True*, and all others are set to *False*.

Give an exact exponential-time algorithm for the *Exact 3-Satisfiability* problem that runs in $O^*(c^n)$ time for some $c < 2$. Clearly state your algorithm. Analyse the running time of your algorithm by giving an explicit upper bound on the running time.

You may use the following bound on values of the τ function to analyse your algorithm. Note that there are different correct solutions to this question: for some solutions the values below will help you, in other solutions you do not need them.

$\tau(1,1) = 2$	$\tau(1,2) < 1,6181$	$\tau(1,3) < 1,4656$	$\tau(1,4) < 1,3803$	$\tau(1,5) < 1,3248$
$\tau(2,2) < 1,4143$	$\tau(2,3) < 1,3248$	$\tau(2,4) < 1,2721$	$\tau(2,5) < 1,2366$	$\tau(3,3) < 1,2600$
$\tau(3,4) < 1,2208$	$\tau(3,5) < 1,1939$	$\tau(4,4) < 1,1893$	$\tau(4,5) < 1,1674$	$\tau(5,5) < 1,1487$
$\tau(1,1,2) < 2,4143$	$\tau(1,1,3) < 2,2056$	$\tau(1,1,4) < 2,1069$	$\tau(1,2,2) = 2$	$\tau(1,2,3) < 1,8393$
$\tau(1,2,4) < 1,7549$	$\tau(1,3,3) < 1,6957$	$\tau(1,3,4) < 1,6181$	$\tau(1,4,4) < 1,5437$	$\tau(2,2,2) < 1,7321$
$\tau(2,2,3) < 1,6181$	$\tau(2,2,4) < 1,5538$	$\tau(2,3,3) < 1,5214$	$\tau(2,3,4) < 1,4656$	$\tau(2,4,4) < 1,4142$
$\tau(3,3,3) < 1,4423$	$\tau(3,3,4) < 1,3954$	$\tau(3,4,4) < 1,3533$	$\tau(4,4,4) < 1,3161$	

5. An Approximation Algorithm for Bottleneck TSP (1 + 1 point)

We are a graph $G = (V, E)$ with n vertices and an edge between every pair of vertices. Edges have non-negative lengths $\ell : E \to \mathbb{R}_+$ that satisfy the triangle inequality, i.e., for all v_i, v_j, v_k: $\ell(v_i, v_k) \leq \ell(v_i, v_j) + \ell(v_j, v_k)$).

A *bottleneck TSP tour* is a round tour that visits every vertex exactly once and starts and ends at the same vertex (just like a normal TSP tour). The cost of a bottleneck TSP tour is defined as the *maximum* length over all edges of the tour, i.e., the length of the longest edge (this we call the bottleneck). In the *bottleneck TSP* problem, we are given such a graph G and are asked to compute the minimum cost bottleneck TSP tour.

(a) **Given a minimum spanning tree of G, show that you can construct a tour 'around' the spanning tree such that each consecutive pair of nodes in the tour 'jumps over' at most two vertices of the spanning tree.** In other words, the tour contains every vertex in the graph exactly once, and for any two consecutive vertices in the tour, the path between these vertices in the spanning tree consists of at most three edges.

Hint: Give an inductive proof on the structure of any tree.

(b) **Show that there exists a polynomial time 3-approximation algorithm for the bottleneck TSP problem.**

Bonus question (1 point)

We think that this bonus question is hard. We advise to only start with it when you are ready with the other questions of the exam. The question is a twist on list coloring, now with costs, colors that are integers, on arbitrary graphs, and asks for kernelization.

Suppose we are given a graph $G = (V, E)$, an integer K, and for each vertex $v \in V$, a list $L(v) \subseteq \{0, 1, 2, \ldots, K\}$.

A *proper list-coloring* of G is a function $f : V \to \{0, 1, 2, \ldots, K\}$, such that

(1) For each $v \in V$, $f(v) \in L(v)$. (Each vertex gets an allowed color.)

(2) For each edge $\{v, w\} \in E$, $f(v) \neq f(w)$.

The *cost* of a proper list coloring f is $\sum_{v \in V} f(v)$.

Consider the following parameterized problem:

MINIMUM COST PROPER INTEGER LIST COLORING
Given: A graph $G = (V, E)$, an integer K, and for each vertex $v \in V$, a list $L(v) \subseteq \{0, 1, 2, \ldots, K\}$.
Parameter: K
Question: Does G have a proper list coloring with cost at most K?

Show that this problem has a kernel with at most $O(K^2)$ vertices.