Exam Algorithms and Networks 2015/2016

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. 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.

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

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

Important: Use separate sheets for questions 1 – 3, and for questions 4 – 7. You may use more than one sheet for a collection. Failure to do so may result in having your work graded later.

Good luck!

Question 1: Strong and weakly $\mathcal{NP}$-completeness (1 point)

Give the definitions of when a problem P is *weakly $\mathcal{NP}$-complete* and when P is *strong $\mathcal{NP}$-complete*. Explain the difference between both notions.

Question 2: $\mathcal{NP}$-completeness of Dominating Set (2 points)

Consider the following problems:

> Dominating Set
> **Given:** Graph $G = (V, E)$, integer k.
> **Question:** Does there exists a *Dominating Set* of size at most k in G?

Note that a *Dominating Set* in a graph $G = (V, E)$ is a subset $D \subseteq V$ of the vertices such that for every $v \in V$ there exists a $w \in D$ such that $v = w$ or $\{v, w\} \in E$.

> Set Cover
> **Given:** Given a set $\mathcal{U}$, a family $\mathcal{F}$ of subsets of $\mathcal{U}$, and integer k.
> **Question:** Does there exists subcollection $\mathcal{C} \subseteq \mathcal{F}$ of size at most k such that:
>
> $$\bigcup_{S \in \mathcal{C}} S = \mathcal{U}$$

Prove that the Dominating Set problem is $\mathcal{NP}$-complete.

To prove this result, you may use that the Set Cover problem is $\mathcal{NP}$-complete. You may also use any $\mathcal{NP}$-completeness results proven in the lectures.

Question 3: Exact exponential-time algorithm for Not-All-Equal 3-SAT (2 points)

The *Not-All-Equal 3-Satisfiability* problem is defined as follows:

> Not-All-Equal 3-Satisfiability
> **Given:** Given a set of boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a set of clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ 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 not all literals in the clause have the same value, i.e., they are not all set to *True*, nor are they all set to *False*.

Give an exact exponential-time algorithm for the *Not-All-Equal 3-Satisfiability* problem that runs in $\mathcal{O}^*(c^n)$ time for some $c < 2$. Prove an upper bound on the running time of your algorithm with an explicit value for c.

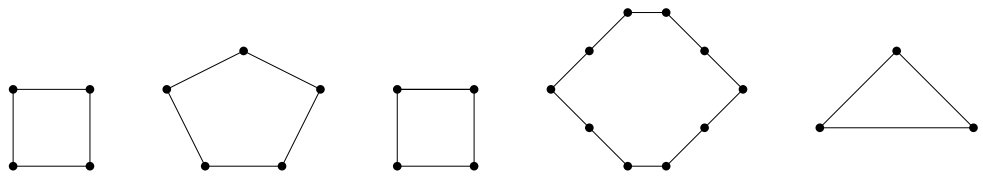
To solve this question you may (but need not!!) use the following values of the τ function given during the lectures:

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

Write the answers to questions 4 – 7 on a different set of paper and hand these in separately !

Question 4: Treewidth (1 point)

Let G be a graph such that every connected component of G is a cycle. **What is the treewidth of G?** You do not have to explain the answer.



Question 5: Isomorphism (1 point)

Give the definition of graph isomorphism.

You may assume that the graph(s) we look at are undirected and unlabelled. Give the mathematical definition; no further explanation is needed.

Question 6: Dynamic programming on trees (1.5 points)

Let $T = (V, E)$ be a rooted tree with root r. An *ABCD*-mapping is a function $f : V \to \{A, B, C, D\}$, such that

- If a vertex has label A, then all its children have label B or C, i.e., if $f(v) = A$ and w is a child of v, then $f(v) \in \{B, C\}$.
- If a vertex has label B, then all its children have label C or D, i.e., if $f(v) = B$ and w is a child of v, then $f(v) \in \{C, D\}$.
- If a vertex has label C, then all its children have label D or A, i.e., if $f(v) = C$ and w is a child of v, then $f(v) \in \{D, A\}$.
- If a vertex has label D, then all its children have label A or B, i.e., if $f(v) = D$ and w is a child of v, then $f(v) \in \{A, B\}$.

Suppose for each vertex v, we have four values: $c(A, v)$ is the cost of mapping v to A; $c(B, v)$ is the cost of mapping v to B; $c(C, v)$ is the cost of mapping v to C; $c(D, v)$ is the cost of mapping v to D. We want to solve the following problem: find an ABCD-mapping of T of minimum cost.

(I.e., we ask for an ABCD-mapping $f : V \to \{A, B, C, D\}$, fulfilling the conditions above, such that $\sum_{v \in V} c(f(v), v)$ is as small as possible.)

Show that the problem of finding an ABCD-mapping of a rooted tree of minimum cost can be solved in linear time. Briefly explain why your algorithm is correct.

Hint: use dynamic programming.

Question 7: Kernelization (1.5 points)

Consider the following problem.

> Distance from a Cycle
> **Given:** Undirected graph $G = (V, E)$, integer K
> **Parameter:** K
> **Question:** Can we delete at most K edges from G such that G forms one cycle, i.e., is there a set $F \subseteq E$ with $|F| \leq K$, such that $(V, E - F)$ is a cycle?

Note that in the resulting graph after deleting the edges, each vertex must have degree exactly two.

In the example below, the left graph is a solution when K is at least two: if we delete two edges, we get a cycle (right graph).

Show that this problem has a polynomial kernel.