

Exam IN4301 Advanced Algorithms Part II

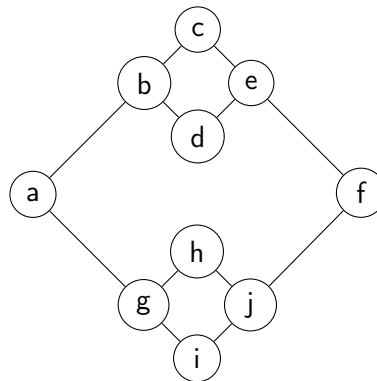
20 November 2020 | 09:00–11:00

- Start each question on a new page.
- This is an open-book individual examination with 4 questions worth 24 points in total.
- If your score is n points, then your grade for this exam will be $1 + \frac{9}{24}n$.
- Use of course books, readers, notes, and slides is permitted.
- Use of computing devices, including calculators, mobile devices and laptops, is permitted.
- Use of websites such as but not limited to Google, Stack Exchange and Wikipedia is not permitted.
- By submitting your answers this exam, you agree to the following Code of Honour pledge: “I promise that I have not used unauthorized help from people or other sources for completing my exam. I created the submitted answers all by myself during the time slot that was allocated.”

Name: _____ Signature: _____

- Oral fraud checks will occur immediately after the exam. **If you have not signed up for an oral interview, do so now, because if you forget, your exam is *invalid*.**
- Write clearly, use correct English, and avoid verbose explanations. Giving irrelevant information may lead to a reduction in your score. *Almost all question parts can be answered in a few lines!*
- This exam covers Chapters 10 of Kleinberg, J. and Tardos, E. (2005), *Algorithm Design*, all information on the slides of the Part II of the course, everything discussed in lectures of Part II, and the set of papers as described in the study guide for Part II.
- This exam assumes a familiarity with the stated background of the course.
- The total number of pages of this exam is 2 (excluding this front page).
- Exam prepared by M. de Weerd and N. Yorke-Smith. ©2020 TU Delft.

- (4 points) Give a tree decomposition of the following graph that has the lowest width you can find, explain why this is a correct tree decomposition (hint: you don't need to give the definition itself, but you may use it for your explanation).



- (6 points) We consider the traveling salesperson problem (TSP) with dependencies. Given are a set S of n cities and the distances $d_{ij} \geq 0$ between each pair of cities $i, j \in S$. In this variant of the problem, additionally we are given a set of dependencies: pairs of cities (i, j) for which city i needs to be visited before city j (it is given that city 1 has no dependencies). The problem is to find a sequence of all the cities (starting and ending in city 1) which minimizes the total distance while meeting the dependencies.

Describe how you would approach this problem using dynamic programming (i.e., provide a recursive function defining the value of the optimal solution and explain how to use it).

- (6 points) Consider the following problem. Given a bipartite graph $G = (V_S \cup V_T, E)$, find a minimum size set ('cover') $V' \subseteq V_S$ such that for each vertex $t \in V_T$ there is a neighbor $s \in V'$ (so with $(s, t) \in E$ and we may say that s covers in this case its neighbors in V_T).¹

Give two rules to reduce an instance of this problem *that are as general as you can think of* (without loss of optimality), and for each of these explain briefly why it is correct.

- If large, complex systems for which we have good models do not function properly, possibly broken components can be identified (partly) automatically.² Let us describe a system by a set of n components A . If such a system is not working correctly, a set of tests can be done. Each test i involves a subset of components $B_i \subseteq A$. Suppose that we are given m *failed* tests involving subsets B_1, \dots, B_m , and that furthermore the maximum size of each subset is less than or equal to $c \leq n$, i.e., for all i , $1 \leq i \leq m$ it holds that $|B_i| \leq c$.

We are considering the problem of deciding whether there is a subset of components H of size at most k that could explain all the failed tests, i.e., such that for all i , H has at least one component in common with B_i .

Below we ask you to give a bounded search tree algorithm for this problem with a runtime of the form $O(f(c, k) \cdot p(n, m))$, where $p(\cdot)$ is a polynomial function, and $f(\cdot)$ is an arbitrary function.

- (6 points) Express the solution to a problem of size k and sets B recursively as a combination of solutions to one or more reduced problems of size $k - 1$ (and a smaller set B') and explain why your answer is correct.

¹Note that this exact problem variant is not included in the course material.

²You may think for example of debugging large software infrastructures.

- (b) (2 points) Give the runtime of this algorithm using $O(\cdot)$ and argue why it meets the desired form given above.