TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Exam IN4301 Advanced Algorithms – Part I of III

October 12, 2017, 9:00-10:00

- This is a closed book examination with 4 questions worth of 20 points in total.
- Your mark for this exam part will be the number of points divided by 2.
- Use of book, readers, notes, and slides is not allowed.
- Use of (graphical) calculators is not permitted.
- Specify your name, student number and degree program, and indicate the total number of submitted pages on the first page.
- Write clearly, use correct English, and avoid verbose explanations. Giving irrelevant information may lead to a reduction in your score.
  Notice that almost all questions 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 course of the first 4 lectures, and the set of papers as described in the study guide.
- If your average for all three exam parts is at least a 5.0, and so is the average of the programming exercises, as well as the average of the homework exercises, then your final mark for this course is the average of these three marks, rounded to the nearest half of a whole number. That is, 9.7 is rounded to 9.5, and 5.8 is rounded to 6.
- The total number of pages of this exam is 1 (excluding this front page).

1. Consider the problem of deciding whether an independent set exists of size at least k in a given undirected graph G.
   (a) (3 points) Describe a search tree algorithm for this problem with an upper bound on the run time of $O^*(1.4^n)$.
   (b) (3 points) Explain how to derive a good (tight) upper bound on the run time complexity of any search tree algorithm, and do this for this specific algorithm.

2. Given a graph $G = (V, E)$ with vertex weights $w_v > 0$, we consider the problem of finding a vertex cover $S \subseteq V$ with the least total weight, i.e., $w(S) := \sum_{v \in S} w_v$ is minimized.

   Let also a *nice* tree decomposition $(Tr = (T, F), \{V_t : t \in T\})$ of G be given.
   (a) (1 point) Define for a leaf node t of Tr the minimal total weight $OPT_t(U)$ for each possible subset U of the bag V_t.
   (b) (1 point) Define for a forget node t of Tr the minimal total weight $OPT_t(U)$ for each possible subset U of the bag V_t.
   (c) (2 points) Define for an introduce node t of Tr the minimal total weight $OPT_t(U)$ for each possible subset U of the bag V_t.
   (d) (1 point) Define for a join node t of Tr the minimal total weight $OPT_t(U)$ for each possible subset U of the bag V_t.
   (e) (1 point) Suppose that for all tree nodes t and for all arguments U the values defined in the above recursive function have been stored in a table $M_t[U]$. Explain how to determine the minimal total weight of a vertex cover of G from this table.

3. In the paper by Woeginger et al. the following is written.

   > "Moon & Moser [29] have shown that a graph with n vertices contains at most $3^{n/3} \approx 1.4422^n$ maximal independent sets. By considering a collection of $n/3$ independent triangles, we see that this bound is best possible. Paull & Unger [36] designed a procedure that generates all maximal independent sets in a graph in $O(n^2)$ time per generated set."

   We will use this information to construct an algorithm to determine whether a graph is 3-colorable.

   We say a graph is k*-colorable* if and only if it is possible to assign each vertex a color (number) from $\{1, \ldots, k\}$ such that no two neighbors have the same color.
   (a) (1 point) Explain why a bi-partite graph is 2-colorable.
   (b) (3 points) Give an $O^*(1.4422^n)$ algorithm to determine whether a graph is 3-colorable and explain why this is correct.

4. (a) (2 points) Give a precise definition of when a decision problem with input (I, k) is kernelizable, where I is the problem instance and k is a parameter.
   (b) (2 points) A set $S \subseteq V$ is a *dominating set* in a graph $G = (V, E)$ if each vertex $v \in V$ is either in S or has a neighbor in S. Consider the problem of finding a dominating set of size at most k. This so-called *minimum dominating set problem* is kernelizable. Give a rule that can be used to reduce an instance of this problem.

End of test