TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Exam IN4301 Advanced Algorithms mid-semester test

October 28 2008, 10.00-12.00

- This is an open book examination with 4 questions worth of 21 points in total. Your mark will be the number of points divided by 2.
- Use of book, readers and slides is allowed, but other notes or (copies of) other material is not.
- Use of (graphical) calculators is not permitted.
- Specify your name, student number and degree program, and indicate the total number of submitted pages at least on the first page.
- Write clearly and avoid verbose explanations. Giving irrelevant information may lead to a reduction in your score.
- Use a separate sheet for each question.
- This exam covers Chapters 10 and 11 of Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*, and the first five papers from the reader (pages 1–94).
- The compensation rules offered when you take part in this mid-semester test are as follows:
  - If you obtain a grade ≥ 5.0, this result will be taken into account when participating in the final examination in January. If your result is less than 5.0, this result will not be taken into account.
  - if you obtained a result ≥ 5.0 for the mid-semester test then
    * if you complete both Part I and Part II of the final examination, your final grade is the average of the maximum of the grades obtained for Part I (i.e., the result of the mid-semester test and the result of the January exam) and the grade obtained for Part II of the January exam.
    * if you complete only Part II of the final examination, your final grade is the average of the grade obtained for Part I during the mid-semester test and the grade obtained for Part II of the January exam.
  - The above compensation rules only hold for the January 2009 exam and not for the re-examination later that year.
- Total number of pages of this exam: 2.

1. Given a Boolean formula in conjunctive normal form (CNF) with m clauses (of arbitrary length) and n variables, consider the NP-complete problem of finding a truth assignment to the variables in this formula that satisfies at least k clauses (MAXSAT).

   In this exercise we assume that the given formula does not contain any clauses with both a variable and the negation of the same variable. Such clauses can never be satisfied and can thus be removed.

   (a) (1 point) Given the following CNF with 4 clauses and 2 variables: $(\neg x \vee \neg y) \wedge (x \vee y) \wedge (\neg x \vee y) \wedge (\neg y)$. Does there exist a truth assignment that satisfies at least 3 clauses? If so, give such a truth assignment. If not, explain why not.

   (b) (1 point) Suppose that $k \leq \lceil \frac{m}{2} \rceil$.[1] Give a simple linear algorithm to find a truth assignment that satisfies at least k clauses, and briefly explain why this algorithm is correct.

   (c) (1 point) Suppose that $k > \lceil \frac{m}{2} \rceil$. To arrive at an efficient kernelization, we split the set of clauses into two classes: l longer clauses that have at least k literals, and $m - l$ shorter clauses that have less than k literals. Suppose that $l \geq k$. Explain how you could easily satisfy at least k clauses.

   (d) (2 points) Suppose that there are less than k clauses with at least k literals, so $l < k$. Using the idea just mentioned, we only first need to find $k - l$ clauses to satisfy among the $m - l$ shorter clauses (with less than k literals), and can then easily satisfy the l longer clauses. Finding at least $k - l$ clauses to satisfy among the $m - l$ shorter clauses is a kernel of this problem. Bound the size of this kernel from above in terms of k (as strictly as possible).

2. The minimum-cost dominating set problem is specified by an undirected graph $G = (V, E)$ and costs $c(v)$ on the nodes $v \in V$. A subset $S \subset V$ is said to be a dominating set if all nodes $u \in V - S$ have an edge (u, v) to a node v in S. The goal is to find such a dominating subset S with minimal costs, i.e., such that $\sum_{v \in S} c(v)$ is minimized. In this exercise we only study the special case in which G is a tree.

   (a) (3 points) Give a recursive function that returns the cost of the minimum-cost dominating set for the special case in which G is a tree.

   (b) (2 points) Give an iterative polynomial-time algorithm that prints the minimum-cost dominating set for the special case in which G is a tree.

Please turn page.

[1] The square brackets denote rounding upwards.

3. The *Min Makespan Problem* is: given n jobs to schedule on m identical machines, where job j has size $s_j \in N$, schedule the jobs to minimize their makespan. Here, the makespan of a schedule is the earliest time when all machines have stopped doing work. This problem is NP-hard, as can be seen by a reduction from the Partitioning problem.

   The following algorithm, due to Ron Graham, yields an approximation algorithm:

   Algorithm 1 (Graham's List Scheduling)

   Given a set of n jobs and a set of m empty machine queues,

   1. Order the jobs arbitrarily in a list L of jobs.
   2. Until the job list L is empty, move the next job in the list to the end of the shortest machine queue.

   You have to answer the following questions:

   (a) (2 points) Prove that this algorithm is a $(2 - \frac{1}{m})$-approximation algorithm.

   (b) (1½ points) Give a tight example, proving that the $2 - \frac{1}{m}$ bound is tight.

   (c) (1½ points) Suppose that the machines $k = 1, \ldots, m$ are non-identical in the sense that a job j with size s_j needs $\lceil \frac{s_j}{r_k} \rceil$ time units to complete when executed on machine k, where $r_k \in \mathbb{N}$ and $r_k \geq 1$. Show that in this case Graham's List Schedulings algorithm does not provide a constant ratio approximation algorithm for the minimal makespan.

   (d) (1 point) (bonus) How can you improve this List Scheduling algorithm to deal with the non-identical machine case?

4. Let $M = (S, I)$ be a matroid, where $S \neq \emptyset$, let $\mathcal{B} \subseteq I$ be the set of bases of M, c a cost function $S \to \mathbb{N}$ and for every $A \subseteq S$, $c(A) = \Sigma_{a \in A} c(a)$

   (a) (½ point) Give an example of a matroid M where $|\mathcal{B}| = 1$.

   (b) (½ point) Give an example of a matroid M such that $|\mathcal{B}| = |S|$ (be careful).

   (c) (2 points) Show that the following property is a consequence of M being a matroid:

   > *For every two bases $B, B' \in \mathcal{B}$, $x \in B - B'$ implies that there exists some $y \in B' - B$ such that $(B - \{x\}) \cup \{y\} \in \mathcal{B}$.*

   (d) (2 points) Consider the following swapping algorithm:

   Algorithm 2 (Swapping Algorithm)

   1. Choose any $B \in \mathcal{B}$;
   2. while $\exists B' \in \mathcal{B}$ such that $|(B' - B) \cup (B - B')| = 2$ and $c(B') > c(B)$
      (a) $B := B'$

   Prove that, given M, the swapping algorithm finds an optimal basis B, i.e., a basis B such that $c(B)$ is maximal.

End of test