TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

Exam IN4301 Advanced Algorithms

January 25 2010

- Use a separate sheet for each question.
- This is an closed book examination with 5 questions worth of 100 points in total.
- Your mark for this exam will be the number of points divided by 10.
- If you have at least a 5.0 for both this exam and the practical work, your final mark for this course is the average of both (or $\frac{2}{3}$ of your practical mark plus $\frac{1}{3}$ of this exam, whatever is higher).
- 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 and 11 of Kleinberg, J. and Tardos, E. (2005), *Algorithm Design*, all information on the slides of the course, and the set of papers as described in the study guide.
- The total number of pages of this exam is 3 (excluding this front page).

1. Consider the problem of placing a minimum number of (rotatable) cameras in a museum consisting only of corridors in such a way that each corridor can be observed. This problem can be modeled as the problem of finding a vertex cover of minimum size in a graph G.

   (a) (3 points) Give the definition of a vertex cover S in a graph $G = (V, E)$.

   (b) (4 points) Let $n = |V|$. To solve the problem of finding a vertex cover of size at most k more efficiently than trying all 2^n subsets, we can apply the technique of pruning the (bounded) search tree. Give an example of how to construct a search tree of size significantly less than 2^n.

   (c) (4 points) Explain how to compute the run-time of an algorithm using the technique of pruning the (bounded) search tree.

   (d) (7 points) In case G is a tree, we can efficiently solve this problem with a dynamic programming approach. Describe the main steps of this approach, illustrated by the vertex cover problem.

   (e) (4 points) Any graph $G = (V, E)$ can be modeled as a tree by a tree decomposition $(T, \{V_t\})$, which is defined as follows:

   - Every node in V belongs to at least one set V_t.
   - For every edge in E there is a set V_t that contains both end points.
   - For every three nodes t_1, t_2, t_3 such that t_2 lies on the path from t_1 to t_3, it holds that if a node v belongs to both V_{t_1} and V_{t_3}, it also belongs to V_{t_3}.

   Explain *why* a tree decomposition can be used to find a minimum vertex cover in time exponential only in the tree-width.

   (f) (3 points) When do we say that a problem of size n is fixed-parameter tractable with respect to a parameter k?

2. Consider the KNAPSACK problem, where we have a knapsack of volume V and a collection of n objects whose volumes are $v_1, \ldots, v_n$, and whose costs are $c_1, ..., c_n$. We consider the so-called *unbounded* version where there are n *types* of objects i with volume v_i and cost c_i, and an unlimited number of objects of each type. It is your task is to select objects to pack in your knapsack so that the total cost of those items is maximized, subject to the constraint that the total volume of the selected items does not exceed V. Without loss of generality, we may assume that for all $i = 1, 2, \ldots, n$ it holds that $v_i \leq V$.

   (a) (6 points) Present a greedy algorithm for the unbounded knapsack problem that selects objects in decreasing cost per unit volume.

   (b) (4 points) Give the outcome of your Greedy algorithm on the unbounded knapsack problem for the instance where we have four types of objects with volumes 3,4,6,7 and cost 3, 2, 2, 1, respectively. The total volume of the objects in the knapsack is bounded by $V = 17$. In addition, what is the optimal outcome?

   (c) (6 points (bonus)) This greedy algorithm is guaranteed to return a solution within a factor 2 of the optimum opt in the case of the unbounded knapsack problem. Prove this result.
   Hint: First, show that your greedy algorithm finds a solution with cost C where at least $|V|/2$ volume is taken by the objects that have the highest cost per volume. Then show that $opt \leq 2 \times C$.

   (d) (6 points) Give the general definition of a tight example for an approximation ratio r of an algorithm A for a problem P. Also, give a tight example for the approximation ratio of this greedy algorithm for the unbounded knapsack problem.

   The binpacking problem is the problem to store n objects of weight $w[1..n]$ in as few bins of capacity W as possible. We might assume $w[i] \leq W$ for every i.

   Assume that there exists an approximation algorithm ApproxBin$(w[1..n], W, \epsilon)$ that runs in a time polynomial in n and $\frac{1}{\epsilon}$ and is guaranteed to find an $1 + \epsilon$-approximation to the binpacking problem instance $(w[1..n], W)$.

   (e) (3 points) What is the name of the approximation complexity class where the problem resided if such an approximation algorithm would exist?

   (f) (6 points) Show that the existence of such an approximation algorithm ApproxBin$(w[1..n], W, \epsilon)$ would imply that we can solve the binpacking problem in polynomial time.
   Hint: use $\frac{1}{\epsilon} = n + 1$ and use the fact that $opt/(n + 1) < 1$ since it is always possible to store the objects in n separate bins.

3. Consider a linear program P and its dual D.
   (a) (3 points) If D is concerned with maximizing a linear fuction in k non-negative variables and there are r linear inequalities of the form <=, specify orientation of optimization, number of nonegative variables and number and form of linear inequalities of P.
   (b) (3 points) P is changed in such away that an extra non-negative variable is introduced. This variable obtains a 6 as coefficient in the objective function and also a 6 in each inequailty. What changes precisely in D?
   (c) (3 points) If one can spot a feasible solution to D with objective value 11, what can be said over P.
   (d) (3 points) It turns out that in fact the variables of P must be integer. Hence P is turned into an linear integer program P-int by adding the integrality constraints. The feasible solution of D of above turns out to be non-integer. A branch and bound search is started to solve P-int. A first relaxation run of P-int gives 10.5 as answer. Is this possibly correct?
   (e) (3 points) An integer feasible solution of P with value 13 is spotted during the branch and bound search of P-int. What changes from now in this search?
   (f) (3 points) After branching a left branch gives a relaxation value of 14 and the corresponding value of the right branch is 12.5. What can be neglected during further search?
   (g) (7 points) The variables of P now turned out to be in fact Boolean. An SDP relaxation P-Bool is considered. Describe P-Bool as detailed as possible

4. (a) (4 points) Explain the concept of Black Box Optimization (BBO).
   (b) (4 points) Explain the concept of the Estimation–of–Distribution Algorithm and give pseudo-code for its general formulation.
   (c) (4 points) Explain why EDAs are thought to be a principled approach for effective BBO.
   (d) (5 points) Given is a binary data set that contains 4 vectors of length 2:

   0 0
   0 1
   1 0
   1 1

   The Akaike Information Criterion (AIC) is used to select the probability distribution to use to represent this data set. The candidate probability distributions to choose from are $P(\vec{X}) = P(X_0)P(X_1)$ and $P(\vec{X}) = P(X_0, X_1)$. Explain which distribution will be selected.

5. (a) (4 points) Explain why it is sometimes impossible to find all Pareto-optimal solutions using weighted aggregation, i.e. by finding the optimum of the single-objective linearly weighted sum of objectives $f(\vec{x}) = \sum_{i=0}^{m-1} w_i f_i(\vec{x})$.
   (b) (4 points) Explain how an elitist archive works and why it is important to have one in a multi-objective iterative algorithm (such as a multi-objective evolutionary algorithm).

End of test