CS580: Foundations

Exam #3 (Final Exam)

Thursday, May 6, 2004

INSTRUCTIONS:

Be concise and clear. This is a closed-book, closed-notes exam. Answer all questions. Scratch paper is available at the back of the exam.

[As an aide to pace your timing in taking the exam, I have put the estimated time to complete each problem at the beginning of each of the 5 multi-part problems. These times total to 115 minutes; you have 120 minutes to complete the exam.]

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Max Score</th>
<th>Your Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Good luck, and have a great summer!
1. (16 points) Approximation Algorithms Performance Ratios (5 minutes)

Match the approximation algorithms we have studied with their correct performance ratio description by writing in each blank the letter of the best answer from the table below. Answers may be used more than once.

_____APPROX-TSP-TOUR (i.e., Approximate Traveling Salesman Problem with Triangle Inequality)

_____GREEDY-SET-COVER

_____APPROX-SUBSET-SUM

_____Parallel Machine Scheduling, assigning any non-scheduled job to an idle machine

_____GENERAL-TRAVELING-SALESMAN

_____APPROX-VERTEX-COVER

_____MAX-3-CNF Satisfiability

_____Bin-Packing, First-Fit heuristic

Select answers for the above algorithms from the following table:

<table>
<thead>
<tr>
<th></th>
<th>Polynomial-time 1-approximation algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Polynomial-time 7/8-approximation algorithm</td>
</tr>
<tr>
<td>B</td>
<td>Polynomial-time 8/7-approximation algorithm</td>
</tr>
<tr>
<td>C</td>
<td>Polynomial-time 2-approximation algorithm</td>
</tr>
<tr>
<td>D</td>
<td>No polynomial-time $\rho$-approximation (for $\rho \geq 1$) exists unless $P = NP$</td>
</tr>
<tr>
<td>E</td>
<td>Polynomial-time $\rho(n)$-approximation algorithm, where $\rho(n) = H(\max{</td>
</tr>
<tr>
<td>F</td>
<td>Randomized 1-approximation algorithm</td>
</tr>
<tr>
<td>G</td>
<td>Randomized 7/8-approximation algorithm</td>
</tr>
<tr>
<td>H</td>
<td>Randomized 2-approximation algorithm</td>
</tr>
<tr>
<td>I</td>
<td>Randomized $\rho(n)$-approximation algorithm</td>
</tr>
<tr>
<td>J</td>
<td>Fully polynomial-time approximation algorithm</td>
</tr>
</tbody>
</table>
2. (25 points) **Dynamic Programming** (3 parts, 35 minutes)

The 0-1 knapsack problem, as posed in your text, is as follows:

A thief robbing a store finds \( n \) items; the \( i \)th item is worth \( v_i \) dollars and weighs \( w_i \) pounds, where \( v_i \) and \( w_i \) are integers. The thief wants to take as valuable a load as possible, but he can carry at most \( W \) pounds in his knapsack, for some integer \( W \). Which items should he take?

This problem exhibits optimal substructure, as follows. Let \( i \) be the highest numbered item among \( 1, \ldots, k \) items in an optimal solution \( S \) for \( W \) with value \( v(S) \). Then \( S' = S - \{i\} \) is an optimal solution for \( W - w_i \), with value \( v(S') = v(S) - v_i \).

We can then define \( c[i, w] \) as the value of the solution for items \( 1, \ldots, i \) and maximum weight \( w \).

2a. (10 points) Express a recursive solution for the 0-1 knapsack problem, defined in terms of \( c[i, w] \). Be sure to include the proper boundary conditions.

\[
\begin{align*}
c[i, w] = & \begin{cases}
  0 & \text{if } w < 0 \\
  \max(v_i + c[i-1, w-w_i], c[i-1, w]) & \text{otherwise}
\end{cases}
\end{align*}
\]

*(The 0-1 knapsack problem is continued on the next page.)*
2b. (10 points) Write the pseudocode that implements this dynamic programming algorithm for the 0-1 knapsack problem. The input and output of your algorithm are shown below. Do not use memoization.

Algorithm DYNAMIC-0-1-KNAPSACK (v, w, n, W)

Input: Two sequences \( v = < v_1, \ldots, v_n > \) and \( w = < w_1, \ldots, w_n > \) (representing the values and weights, respectively), the number of items \( n \), and the maximum weight \( W \).

Output: The optimal value of the knapsack. [Note: You do NOT have to output the actual selection, just the value of the optimal solution.]

Algorithm Pseudocode:

2c. (5 points) What is the running time of your algorithm? Use asymptotic notation, and express your running time in terms of \( n \) and \( W \).
3. (22 points) Greedy Algorithms (4 parts, 30 minutes)

In the parallel machine scheduling problem, we are given $n$ jobs $J_1, J_2, \ldots, J_n$, where each job $J_k$ has an associated nonnegative processing time of $p_k$. We are also given $m$ identical machines, $M_1, M_2, \ldots, M_m$. A schedule specifies, for each job $J_k$, the machine on which it runs and the time period during which it runs. Each job $J_k$ must run on some machine $M_i$ for $p_k$ consecutive time units, and during that time period no other job may run on $M_i$.

Consider the following greedy algorithm for parallel machine scheduling: Schedule jobs one by one, and in decreasing order of processing time. Each job is scheduled on the machine on which it finishes earliest. [For example, consider 2 machines and 4 jobs, $J_1, J_2, J_3, J_4$, with processing times 7, 5, 3, 1, respectively. $J_1$ would be assigned to the first machine, $J_2$ would be assigned to the second machine (because $0+5 < 7+5$), $J_3$ would be assigned to the second machine (because $5+3 < 7+3$), and $J_4$ would be assigned to the first machine (because $7+1 < 5+3+1$).

3a. (4 points) Does this greedy algorithm guarantee that the last job to complete will finish at the earliest possible time? (Answer ‘yes’ or ‘no’.)

3b. (8 points) If your answer is ‘yes’, then argue why this is the case. If your answer is ‘no’, then give a counter-example.
3c. (5 points) Concisely define the “greedy-choice property”.

3d. (5 points) What does it mean for a problem to exhibit “optimal substructure”? (Be concise.)
4. (10 points) Computational Geometry (15 minutes)

Suppose you are given the following:

- A set \( Q \) of \( n \) points,
- The convex hull of those points, \( CH(Q) \), consisting of \( h \) vertices, and
- One additional point \( p = (x, y) \).

Give: 1) an efficient algorithm that tests whether or not \( p \) is contained inside the convex hull, and 2) the runtime complexity of your algorithm. Your solution should use cross products.
5. (27 points) Computability and Complexity (5 parts, 30 minutes)

5a. (7 points) Suppose $L$ is regular.
   Then must the set $\text{NoPre}(L) = \{x \in L \mid \text{no proper prefix of } x \text{ is in } L\}$ also be regular? Explain your answer.

5b. (7 points) Prove or disprove whether $C = \{a^i b^j \mid i \neq j; i, j \geq 0\}$ is context-free.
5c. (7 points) Let $D$ denote an algorithm for the decision version of the longest path problem. The input consists of an undirected graph $G$ along with an integer $k$. The output is “yes” if and only if $G$ has a simple path of length $k$ or more. State the corresponding optimization version of longest path, and give the algorithm that uses $D$ to generate the length of the longest path. (You don’t have to output the actual path.)
5d. (6 points) Consider the following two problems, P1 and P2:
In P1, we are given as input a set of \( n \) squares (specified by their corner points), and a number \( k \). The problem is to determine whether there is any point in the plane that is covered by \( k \) or more squares.

In P2, we are given as input an \( n \)-vertex graph, and a number \( k \). The problem is to determine whether there is a set of \( k \) mutually adjacent vertices. (For example, for \( k = 3 \), we are just looking for a triangle in the graph.)

Obviously, both problems P1 and P2 are in NP. Additionally, there exists a simple translation from P1 to P2: just make a graph vertex for each square, and add an edge between a pair of vertices if the corresponding two squares overlap.

5d.1. If P1 is NP-complete, would this translation imply that P2 is NP-complete? Explain your answer.

5d.2. If P2 is NP-complete, would this translation imply that P1 is NP-complete? Explain your answer.