CSC375F **Problem Set 1** **Spring, 2006**

Due: Friday, February 1, beginning of tutorial

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work (but see below). Similar questions will appear on the first term test which will cover material relating to both assignment 1 and assignment 2. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. However, for problem set 1, you may work in pairs for the bonus questions. Anything else is *plagiarism*, and is subject to the University's Code of Behavior. You will receive 1/5 points for any (non bonus) question/subquestion for which you say "I do not know how to answer this question". You will receive .5/5 points if you just leave the question blank.

1. (a) Exercise 7 of chapter 4 (pages 191,192). Describe a greedy algorithm and prove that your greedy algorithm always provides an optimal schedule. [15 points]

   (b) Bonus question: Suppose that the final f_i processing time for each job had to be performed on another supercomputer (instead of a PC) and at most one job can use this supercomputer at a time. Describe a greedy algorithm and prove that your greedy algorithm always provides an optimal schedule. [20 points]

2. This problem concerns Kruskal's greedy algorithm for the MST problem. Let $G = (V, E)$ be a connected graph and $c : E \rightarrow \Re$ a cost function on the edges.

   (a) Suppose that the edge costs have the property that (when sorted) $c(e_1) = c(e_2) < c(e_3) < \ldots < c(e_m)$. That is, the two smallest cost edges have the same cost but all other edge costs are distinct. Use the analysis of Kruskal's algorithm to argue that there is a unique minimum spanning tree T. [10 points]

   (b) Suppose that we now define the cost of a tree T to be the $max_{e \in T} c_e$. Prove or disprove that Kruskals algorithm always produces an optimal solution for computing a minimum spanning tree under this new objective function. [10 points]

3. Consider the following *interval covering problem.*
   Input: A set of intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ where each interval $I_j = [s_j, f_j]$.
   Output: A *cover* $\mathcal{I}' \subseteq \mathcal{I}$ such that for all $I \in \mathcal{I}$, there exists $I' \in \mathcal{I}'$ such that $I' \cap I \neq \emptyset$. The goal is to find a cover of minimum cardinality (ie minimize $|\mathcal{I}'|$).

   (a) Show that the "most obvious greedy algorithm" (i.e. always choosing that interval which overlaps the largest number of currently uncovered intervals) does not produce a minimal cardinality cover. [5 points]

   (b) Describe an optimal greedy algorithm for this interval covering problem. Prove that your algorithm always produces an optimal cover. [15 points]

4. (a) Exercise 17 of Chapter 4 (page 197). [10 points]

(b) Bonus question: Can you provide an algorithm that works in time (close to) $O(n \log n)$? Can you asymptotically beat n^2? [? points]

Note: I am not sure if this is an open question or not. Credit for the problem will depend on the novelty/difficulty of the solution.

5. Provide a greedy algorithm that provides a 2-approximation for the problem of 2 processor interval scheduling with proportional profits. [15 points]

6. Suppose that we have an array $A[1 \dots n]$ of distinct elements which is "almost sorted", in the sense that every element in A is at most 10 positions away from its sorted position.

   (a) Write a divide-and-conquer algorithm to sort A in time $O(n)$. Indicate the recurrence used in the time analysis of your algorithm. [10 points]

   (b) Provide a more exact analysis of the number of comparisons used by your algorithm. That is, what is the smallest constant c for which you can prove that for some constant d, $T(n) \leq c \cdot n + d$ for all $n \geq 1$. Provide the recurrence for $T(n)$ and prove your bound. [5 points]