Due: Wednesday, March 8, 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. Suppose you have a method for multiplying 5×5 matrices using q multiplications (without commutativity). How small does q have to be in order to beat the current asymptotically fastest matrix multiplciation algorithm which has complexity $O(n^{2.38})$. Justify your answer. [10 points]

2. Consider the following one machine scheduling problem. (Note: We already considered this problem for the special case of unit profit.) We are given n jobs $J_1, \ldots, J_n$ with $J_i = (d_i, t_i, v_i)$ where d_i is the deadline for job J_i, t_i is its processing time, and v_i is its profit. Assume all input parameters are positive integers. A schedule is a function $\sigma : \{1, \ldots, n\} \to \{0, 1, 2, \ldots\} \cup \{\infty\}$ where $\sigma(i) = \infty$ means that job J_i is not scheduled and $\sigma(i) = k$ means that job J_i begins executing at time k. A schedule is feasible if
   (1) for all $i \neq j$ if $\sigma(i) \neq \infty$ and $\sigma(j) \neq \infty$ then $[\sigma(i), \sigma(i) + t_i) \cap [\sigma(j), \sigma(j) + t_j) = \emptyset$
   (2) for all i, if $\sigma(i) \neq \infty$ then $\sigma(i) + t_i \leq d_i$.
   That is, no two scheduled jobs will overlap and every scheduled job finishes before its deadline. The optimization problem is to find a feasible schedule σ that maximizes $\sum_{\sigma(i) \neq \infty} v_i$; That is, to maximize the profit of scheduled jobs in a feasible schedule.

   (a) Using an exchange argument show that every feasible schedule can be rearranged so that $\sigma(i) < \sigma(j) \neq \infty$ implies that $d_i \leq d_j$; that is, jobs in a feasible schedule can be scheduled in order of their deadlines. [5 points]

   (b) Now consider the case that all processing times are not too large, say $t_i \leq n^2$ for all i. Describe a polynomial time dynamic programming algorithm for computing the value of an optimal solution. In particular, specify appropriate semantic and computational arrays, briefly justify that your algorithm is correct, and estimate the time complexity of your algorithm. Hint: First sort the jobs so that $d_1 \leq d_2 \ldots \leq d_n$. Observe that this problem is a generalization of the knapsack problem. The knapsack problem is the special case where all $d_i = W$ where W is the weight bound for the knapsack and the knapsack item weights are $w_i = t_i$. [20 points]

3. Consider the weighted interval covering problem where now all intervals I_i have an associated cost c_i; that is, $\mathcal{I} = \{I_i | 1 \leq i \leq n\}$ and $I_i = (s_i, f_i, c_i)$. As for the case of unit costs, we want to compute a cover $\mathcal{I}' \subseteq \mathcal{I}$ so as to minimize the cost of the cover $c(\mathcal{I}) = \sum_{I_i \in \mathcal{I}'} c_i$. Describe a polynomial time DP algorithm for this problem. In particular, specify appropriate semantic and computational arrays, briefly justify that your algorithm is correct, and estimate the time complexity of your algorithm. [20 points]

4. Extend the DP algorithm for edit distance (string alignment) so that in addition to the matching costs $\alpha_{x,y}$ and delete cost δ there is the possibility of a transposition cost τ where symbols xy can be transposed to yx at a cost of τ. [15 points]

5. Using less than or equal 200 words, write a paragraph that articulates the relation (if any) and difference (if any) between divide and conquer and DP algorithms. [15 points]

6. Suppose we are given a flow network $\mathcal{F}$ with integer edge capacities $\{c_e | e \in E\}$ and we are also given a max flow f in $\mathcal{F}$. Suppose we want to increase the max flow by one unit by increasing the capacity of certain edges by one unit each. Show how to efficiently compute a minimal set of edges $E' \subseteq E$ such that after increasing the capacity for edges $e \in E'$, the max flow f' in the resulting network $\mathcal{F}'$ has $val(f') = val(f) + 1$. Your method should take time proportional to Dikstra's shortest path algorithm. [15 points]