Due: Wed, March 9, beginning of lecture

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. 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. 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.
Advice: Do NOT spend an excessive amount of time on any question and especially not on a bonus question. If you wish to spend "free time" thinking about (say) bonus questions that is fine but you should not sacrifice time needed for other courses.

1. (20 points)

   Consider the following variant of the knapsack problem. The input consists of a knapsack size bound W and n sets S_i, each containing 3 items. That is, $S_i = \{I_{i,1}, I_{i,2}, I_{i,3}\}$ and $I_{i,j} = (v_{i,j}, s_{i,j})$ where $v_{i,j}$ (respectively, $s_{i,j}$) is the value (resp. size) of item $I_{i,j}$. A feasible solution S is a choice of at most one item from each set S_i so that the sum of item sizes chosen is at most W. Suppose $W \leq n^2$ and all sizes $s_{i,j}$ are integral. Provide a polynomial time dynamic programming algorithm that will compute the optimal value of a feasible solution. Provide appropriate semantic and computational (i.e. recursively defined) arrays and indicate the time complexity of your algorithm.

2. (25 points)

   Consider the following scheduling problem (which can be viewed as a generalization of the knapsack problem). The input consists of n jobs $J(1), \ldots, J(n)$ where $J(i) = [p_i, d_i, v_i]$ where (respectively) p_i is the processing time (the deadline, the value) of job $J(i)$. Jobs that complete their processing by their deadline contribute their value to the total profit of the schedule and as in interval scheduling, scheduled jobs cannot intersect. The goal is to schedule jobs so as to maximize the total profit. (Late jobs have no value and hence need not be scheduled.) Note that the knapsack problem is the special case when $d_i = W$ for all i.

   (a) (10 points) Show that if a subset S of jobs can be scheduled, then these jobs can be scheduled so that for all $J(i), J(j) \in S$, if $J(i)$ is scheduled before $J(j)$ then $d_i \leq d_j$.

   (b) (15 points)
   Suppose $V = \sum_i v_i \leq n^2$ and that all v_i are integers. Provide a polynomial time dynamic programming algorithm that will compute the optimal value of a feasible solution. Provide appropriate semantic and computational (i.e. recursively defined) arrays and indicate the time complexity of your algorithm.

3. (20 points)

   Consider the following triangulation problem. We are given a set of n points in the plane which are the vertices (in clockwise order) of a convex polygon P in the plane. We are also given a set S of m points in the interior of P. The cost of a triangle Δ is defined as the number of points in S that are located within Δ, including the boundary of Δ. We will let S_Δ denote the intersection of S and a triangle Δ. Assume that the number of points in the intersection S_Δ can be determined in time $O(|S_\Delta|)$ (without knowing what points are in S_Δ).

   Provide a dynamic programming algorithm with complexity polynomial in n and m for triangulating the polygon P so as to minimize the maximize number of points in any triangle of the triangulation.

   What is the total complexity of your algorithm in terms of m and n?

4. (20 points)

   Consider the following "more than a fourth frequency problem" We are given a set A of $n \geq 4$ elements which cannot be sorted (e.g. the elements are polynomials or matrices) but there is a test for equality of any two elements.

   Describe a divide and conquer algorithm to find all the elements x (if any) such that x occurs $n_x \geq \lceil n/4 \rceil$ times in A.

   Give a recurrence that describes the number of pairwise equality tests made by your algorithm. Solve the recurrence and derive a bound on the number of equality tests using your algorithm.
   Hint: How many such elements can A have? Also note that a frequently occurring element must occur frequently in at least one half of the input. You may want to generalize the problem so as to make it more amenable to the divide and conquer paradigm.

5. (20 points)

   (a) We discussed in class Karatsuba's method for multiplying two degree n polynomials using $O(n^{log_2 3})$ scalar arithmetic operations which is approximately, $O(n^{1.59})$. Using this result (as a subprogram), describe a divide and conquer algorithm using $O(n^{log_2 3})$ arithmetic operations which given $\{a_1, \ldots, a_n\}$ computes the polynomial
   $p(x) = \Pi_{1 \leq i \leq n}(x - a_i) = (x - a_1)(x - a_2) \cdots (x - a_n)$. Give a recurrence that describes the number of arithmetic operations made by your algorithm and briefly show why this recurrence yields the desired bound. You may assume $n = 2^k$ for some k.

   (b) Using the FFT, it is known that we can multiply two degree n polynomials (with coefficients say in the field of complex numbers) using $O(n \log n)$ scalar arithmetic operations. Using this FFT based method, how many scalar operations are needed to compute the polynomial $p(x)$ as described above? Justify your answer.