Due: Wed, March 5, 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 any bonus questions (if given). If you wish to spend "free time" thinking about (say) bonus questions that is fine but you should not sacrifice the time needed for other courses.

1. Suppose we are given two sorted lists X^1 and X^2, each containing n distinct numbers and $X^1 \cap X^2 = \emptyset$. Sketch an $O(\log n)$ time divide and conquer algorithm for computing the median of $X^1 \cup X^2$. Define the median as the element of rank $\lceil n/2 \rceil$ so that, for example, in a list of 6 elements ordered so that $a_1 \leq a_2 \ldots \leq a_6$, a_3 is the median value. [15 points]

2. (Bonus question) We are now given n sorted lists $X^1, X^2, \ldots, X^n$, each containing n distinct numbers and $X^i \cap X^j = \emptyset$ for all $i \neq j$. Can you derive an algorithm with time complexity $o(n^2)$ for computing the median of $X^1 \cup X^2 \ldots \cup X^n$? That is, can you do asymptotically better than using a linear time algorithm for finding the median of an unordered set of k elements? [? points]

3. To simplify the discussion of some divide and conquer algorithms we made some assumptions. In the closest pair problem we assumed that all the coordinate x_i values and all th y_i values were distinct, and in the counting inversions problem we assumed that all a_i elements were distinct.

   (a) What if any changes need to be made to the nearest pair algorithm if we do not assume that all coordinate values are distinct? [5 points]

   (b) In the counting inversions problem when elements are not necessarily distinct, let us say that "$a_i = a_j$ for $i < j$" is not an inversion. What if any changes need to be made to the counting inversions algorithm? [5 points]

   (c) In the counting inversions problem when elements are not necessarily distinct, let us say that "$a_i = a_j$ for $i < j$" is an inversion. What if any changes need to be made to the counting inversions algorithm? [5 points]

4. Consider the one machine weighted interval scheduling problem. That is, the inputs are intervals $I_j = (s_j, f_j, v_j)$ where $v_j > 0$ is the positive value of an interval (if scheduled). Provide a DP algorithm for computing the number of different optimal solutions. That is, provide semantic and computational arrays for the problem (including any base cases). What is the time complexity of your algorithm? (Solution S_1 is different from solution S_2 if $S_1 \neq S_2$ as sets.)

   [20 points]

5. This problem concerns a one dimensional clustering problem. Let $X = \{x_1, \ldots, x_n\}$ be n distinct real valued points with say $x_1 < x_2 \ldots < x_n$. A k-clustering of X is a partitioning of X into k disjoint subsets (clusters) $C_1, \ldots, C_k$. The diameter $diam(C)$ of a cluster C is defined as $max\{|x - y| : x, y \in C\}$. We want to find k-clustering $C_1, \ldots, C_k$ so as to minimize $max_i diam(C_i)$.
Provide a DP algorithm for computing an optimal k-clustering. That is, provide semantic and computational arrays for the problem (including any base cases). What is the time complexity of your algorithm?

[20 points]

6. The knapsack problem is often called the $\{0, 1\}$ knapsack problem since for each item it is either taken 0 or 1 times. Suppose we consider the following two variants of the $\{0, 1\}$ knapsack problem. In both cases, the input is $\{(v_1, w_1), \ldots, (v_n, w_n); W\}$ where v_i (respectively, w_i) is the value (resp. weight) of the i^{th} item and W is the total weight limit of the knapsack. All input parameters are positive integers.

   (a) The integer knapsack problem assumes an unlimited supply of each item and the knapsack can contain any integral number of copies of an item. That is, a feasible solution is a vector $\mu = (m_1, \ldots, m_n)$ where each m_i a non-negative integer such that $\sum m_i w_i \leq W$ and the value of such a solution is $\sum m_i v_i$. Provide a DP algorithm for computing an optimal solution to the integer knapsack problem. That is, provide semantic and computational arrays for the problem (including any base cases). What is the time complexity of your algorithm assuming $W \leq n^2$?

   [10 points]

   (b) In the $\{0, 2, 3\}$ knapsack problem, every item can be taken zero, twice or three times (but not once or more than three times). That is, a feasible solution is a vector $\mu = (m_1, \ldots, m_n)$ where each $m_i \in \{0, 2, 3\}$. Provide a DP algorithm for computing an optimal solution to the integer knapsack problem. That is, provide semantic and computational arrays for the problem (including any base cases). What is the time complexity of your algorithm assuming $W \leq n^2$?

   [10 points]

7. Consider the following triangulation problem. The input is a sequence $< z_0, z_1, \ldots, z_{n-1} >$ of points in the Euclidean plane where this sequence forms a convex polygon P. A trangulation of P partitions P into $n-2$ triangles $T_0, T_1 \ldots, T_{n-3}$ where T_i has edges $z_i, z_{i+1(modn)}, z_{v(i)}$ where $v(i) \notin \{i, i+1(modn)\}$. The cost $c(T_i)$ of a triangle T_i is its perimeter, the Euclidean length of the sides of the triangle and the cost of a triangulation is the sum of the triangle costs. Show how to use dynamic programming to compute an optimal cost triangulation.

[20 points]