CSE431/531, Problem Set 3 Due Friday., Oct. 19, in class

Note: There are 5 problems with a total of 100 points. You are required to do all the problems.

1. (20 points) Let P be a convex polygon with n vertices. A triangulation of P is an addition of a set of non-crossing diagonals (which connect non-neighboring vertices of P) such that the interior of P is partitioned by the set of diagonals into a set of triangles. The weight of each diagonal is the Euclidean distance of the two vertices it connects. The weight of a triangulation is the total weight of its added diagonals. Design a dynamic programming algorithm to find a minimum weighted triangulation of P. You should make your running time as short as possible.

2. (20 points) Given two strings X and Y of length n and m respectively, design a dynamic programming based algorithm to find a super-string Z of minimum length such that both X and Y are subsequence of Z. Note that characters in X and Y do not need to appear consecutively in Z as long as they appear in the same order in Z as in X or Y. Design an other algorithm for solving the same problem but with three input strings, W, X, Y, i.e., finding the minimum length super-string for three strings.

3. (20 points) Consider the recurrence relation $T(0) = T(1) = 2$ and for $n \geq 1$, $T(n) = \sum_{i=1}^{n-1} T(i)T(i-1)$. Consider the problem of computing $T(n)$ from n. (a) Show that if you implement this recursion directly, the algorithm would use exponential time in n. (b) Explain how, by not recomputing the same $T(i)$ value twice, one can obtain an algorithm for this problem that uses only $O(n^2)$ time. (c) Give an algorithm for this problem that only use $O(n)$ time.

4. (20 points) Given a binary tree $T = (V, E)$ with each vertex $v \in V$ associated with a non-negative weight $w(v)$, design a dynamic programming based algorithm to partition the tree T into k or fewer subtrees by removing $k-1$ or fewer edges from T such that the maximum weight of each subtree is minimized among all possible partitions, where the weight of a tree is the sum of weights of all vertices in the tree. Make your algorithm as efficient as possible,

5. (20 points) Let $A = a_1 \cdots a_m$ and $B = b_1 \cdots b_n$ be two strings with length m and n respectively. Design a dynamic programming based algorithm to convert A into B with minimum cost using the following rules. For a cost of 3, one can delete any letter from a string. For a cost of 5, one can insert a letter in any position. For a cost of 7, one can replace any letter by any other letter. For example, you can convert $A = abcabc$ to $B = abacab$ via the following sequence of operations: abcabc with a cost of 7 can be converted to abaabc, which with a cost of 3 can be converted to ababc, which with a cost of 3 can be converted to abac, which with a cost of 5 can be converted to abacb, which with a cost of 5 can be converted to abacab. Thus the total cost for this conversion is 23 (may not be the cheapest one).