Dynamic Programming

For *all* of the following problems, a proof that the algorithm you design is correct is required.

1. String A is a supersequence of string B if string B can be obtained from string A by removing letters. For example, the strings BARNYARDSNACK, YUMMYBANANAS, and BWANWANWA are supersequences of the string BANANA.

   Give a dynamic program for finding the length of the shortest string that is a supersequence of two input strings, A and B.

   In this question, be sure to define the dynamic programming table and how to fill it in as well as analyze the running time.

2. Suppose you have a subroutine $QUALITY$ that can compute the *quality* of any given string $A[1 \dots k]$ in time $O(k)$. For example, the quality of a string might be 1 if the string is a Canadianism, and 0 otherwise.

   Given an arbitrary input string $T[1 \dots n]$, we would like to break it into contiguous substrings, such that the total quality of all the substrings is as large as possible. For example, the string "Icashedmypogeyandwenttobuyamickeyof CCatthebeerparlourbutmyskidoogotstuckinthemuskegonmywaybacktotheduplex" can be decomposed into the substrings "I cashed my pogey and went to buy a mickey of CC at the beer parlour but my skidoo got stuck in the muskeg on my way back to the duplex", of which seven are Canadianisms.

   Describe an algorithm that breaks a string into substrings of maximum total quality, using the $QUALITY$ subroutine.

   based on erickson 3.5

3. There are different variations of change making problem, one of them is the following:

   Given an unlimited supply of coins of denominations $x_1, x_2, \dots, x_n$, we wish to make change for a value v using at most k coins; that is, we wish to find a set of $\leq k$ coins whose total value is v. This might not be possible: for instance, if the denominations are 5 and 10 and $k = 6$, then we can make change for 55 but not for 65. Give an efficient dynamic-programming algorithm for the following problem.

   Input: $x_1, x_2, \dots, x_n$; k; v

   Question: Is it possible to make change for v using at most k coins, of denominations $x_1, x_2, \dots, x_n$?

4. For this problem, a subtree of a binary tree means any connected subgraph. A binary tree is complete if every internal node has two children, and every leaf has exactly the same depth. Describe and analyze a dynamic programming algorithm to compute the largest complete subtree of a given binary tree. Your algorithm should return the root and the depth of this subtree.

5. Oh, no! You have been appointed as the gift czar for Giggle, Inc.'s annual mandatory holiday party! The president of the company, who is certifiably insane, has declared that every Giggle employee must receive one of three gifts: (1) an all-expenses-paid six-week vacation anywhere in the world, (2) an all-the-pancakes-you-can-eat breakfast for two at Jumping Jack Flash's Flapjack Stack Shack, or (3) a burning paper bag full of dog poop. Corporate regulations prohibit any employee from receiving the same gift as his/her direct supervisor. Any employee who receives a better gift than his/her direct supervisor will almost certainly be fired in a fit of jealousy. How do you decide what gifts everyone gets if you want to minimize the number of people that get fired? (see Fig. (**??**))

   More formally, suppose you are given a rooted tree T, representing the company hierarchy. You want to label each node in T with an integer 1, 2, or 3, such that every node has a different label from its parents. The *cost* of a labeling is the number of nodes that have smaller labels than their parents. Describe and analyze an algorithm to compute the minimum cost of any labeling of the given tree T. (Your algorithm does not have to compute the actual best labeling - just its cost.)

based on Erickson 3.20

6. *Minimum-Cost Dominating Set Problem* is specified by an undirected graph $G = (V, E)$ and costs $c(v)$ on the nodes $v \in V$. A subset $S \subset V$ is said to be a *dominating set* if all nodes $u \in V - S$ have an edge (u, v) to a node v in S. (Note the difference between dominating sets and vertex covers: in a dominating set, it is fine to have an edge (u, v) with neither u nor v in the set S as long as both u and v have neighbors in S.)
Give a polynomial time algorithm for Dominating Set Problem for the special case in which G is a tree.