Algorithms Middle Examination

Dec. 1, 2021 (10:10 ~ 12:20)

(Note that, if you design an algorithm, you must have pseudocode to represent your algorithm. You can put comments after your pseudocode to clarify your presentation.)

1. (15% Ch. 15) Let $A[1..n]$ be an array of n distinct integers. Give an algorithm to find the length of the longest increasing subsequence of entries in A. The subsequence is not required to be contiguous in the original sequence. For example, if the entries are 11, 17, 5, 8, 6, 4, 7, 12, 3, the longest increasing subsequence is 5, 6, 7, 12. Analyze the worst-case running time and space requirement of your algorithm.

2. (30% Ch. 15 & Ch. 16) Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

   a. (15%) Describe a greedy algorithm to make change using quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

   b. (15%) Give an $O(nk)$-time algorithm that makes change for any set of k different coins, assuming that one of the coins is a penny. (Hint: Since we have optimal substructure, dynamic programming might apply. Let us define $c[j]$ as the minimum number of coins we need to change for j cents. Let the coins be $d_1, d_2, \ldots, d_k$. Since one of the coins is a penny, there is a way to make change for any amount $j \geq 1$. Because of the optimal substructure, if we knew that an optimal solution for the problem of making change for j cents used a coin d_i, we would have $c[j] = 1 + c[j - d_i]$.)

3. (15% Ch. 16) Given a set of tasks with profits $\{p_1, p_2, \ldots, p_n\}$ and deadlines $\{d_1, d_2, \ldots, d_n\}$, determine the maximum profit you can earn. Each task occupies a one-time unit. You can only do one task at a time, and you cannot execute the task after its deadline. Which tasks will you choose to execute, and how do you schedule your time?

   a. (9%) Describe your algorithm.

   b. (3%) Prove the greedy choice property.

   c. (3%) Prove the optimal substructure property.

4. (10% Ch. 22) Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$, and two vertices s and t, and returns the number of simple paths from s to t in G. (Your algorithm only needs to count the simple paths, not list them.)

5. (10% Ch. 22) Please give an $O(|V| + |E|)$ depth-first search (DFS) tree algorithm without using recursion, where $|V|$ is the number of vertices and $|E|$ is the number of edges.

6. (10% Ch. 22) Professor Bacon claims that the algorithm for strongly connected components would be simpler if it used the original (instead of the transpose) graph in the second depth-first search and scanned the vertices to increase finishing times. Does this simpler algorithm always produce correct results? Please give a counterexample.

7. (10% Branch-and-Bound) You are given an integer array *coins* representing coins of different denominations (面額) and an integer *amount* representing a total amount of money. Return *the fewest number of coins that you need to make up that amount*. If that amount of money cannot be made up by any combination of the coins, return -1. You may assume that you have an infinite number of each kind of coin. Please use the Branch-and-Bound strategy to answer the following questions.

a. (5%) Find the formula of the lower bound of *the fewest number of coins that you need to make up that amount.* (Hint: *use nCoins* (*total number of coins taken so far*), *biggest_coin* (*the biggest coin we can choose at that recursive round*), *remaining_amount* (the *total amount we haven't taken until that recursive round*), *current_coins* (*list of possible coins at that recursive round*))

b. (5%) Given coins = [1, 2, 5] and amount = 11, use the best-first search strategy to draw the Branch-and-Bound search tree.