CSCI 570 - Fall 2021 - HW 3

Due September 16, 2021

1 You have N ropes each with length L_1, L_2, ..., L_N, and we want to connect the ropes into one rope. Each time, we can connect 2 ropes, and the cost is the sum of the lengths of the 2 ropes. Develop an algorithm such that we minimize the cost of connecting all the ropes.

Rubric:

10 points for correct algorithm.

-1 point if it doesn't clarify that each step involves picking TWO smallest ropes

-2 points if it's not clearly mentioned that the resultant rope length after joining goes back into the set of candidates and re-sort (the sorting is ensured if heaps or appropriate data-structures are mentioned).

2 You have a bottle that can hold L liters of liquid. There are N different types of liquid with amount L_1, L_2, ..., L_N and with value V_1, V_2, ..., V_N. Assume that mixing liquids doesn't change their values. Find an algorithm to store the most value of liquid in your bottle.

Rubric:

10 points for correct algorithm -3 points if misinterpreted the "value" to mean value per unit and thus, went greedy on V instead of V/L.

3 Suppose you were to drive from USC to Santa Monica along I-10. Your gas tank, when full, holds enough gas to go p miles, and you have a map that contains the information on the distances between gas stations along the route. Let $d_1 < d_2$ ... $< d_n$ be the locations of all the gas stations along the route where d_i is the distance from USC to the gas station. We assume that the distance between neighboring gas stations is at most p miles. Your goal is to make as few gas stops as possible along the way. Give the most efficient algorithm to determine at which gas stations you should stop and prove that your strategy yields an optimal solution. Give the time complexity of your algorithm as a function of n. (20 points)

Rubric:

10 points for correct algorithm that reaching the furthest gas stations each time.

7 points for proving that reaching the furthest gas stations is optimal using sta proof by contradiction.

3 points for pointing out that the running time is O(N).

4 (a) Consider the problem of making change for n cents using the fewest number of coins. Describe a greedy algorithm to make change consisting of quarters(25 cents), dimes(10 cents), nickels(5 cents) and pennies(1 cents). Prove that your algorithm yields an optimal solution. (Hints: consider how many pennies, nickels, dimes and dime plus nickels are taken by an optimal solution at most.) (20 points)

Rubric:

10 points for the greedy algorithm that always pick highest value of coins.

10 points for proving that the algorithm is optimal by supporting that this algorithm is optimal in each range of cents(1-5, 5-10, 10-25, 25-INF). -2 points for missing any range.

(b) For the previous problem, give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Assume that each coin's value is an integer. Your set should include a penny so that there is a solution for every value of n. (10 points)

Rubric:

10 points for any correct coin denominations that does not yield an optimal solution.

5 Solve Kleinberg and Tardos, Chapter 3, Exercise 3. (15 points)

Rubric:

10 points for correct algorithm. -2 points if the algorithm is slower than O(m + n).

5 points to indicating how this algorithm can output topological ordering or a cycle. -3 points for missing any of them.

6 Solve Kleinberg and Tardos, Chapter 4, Exercise 4. (20 points)

Rubric:

10 points for correct greedy algorithm that takes linear running time. -2 points for any slower algorithm.

7 points for proving the algorithm by induction.

3 points for correct time complexity.

7 (Not Graded) There are N tasks that need to be completed by 2 computers A and B. Each task i has 2 parts that takes time a_i(first part) and b_i (second part) to be completed. The first part must be completed before starting the second part. Computer A does the first part of all the tasks while computer B does the second part of all the tasks. Computer A can only do one task at a time, while computer B can do any amount of tasks

at the same time. Find an O(nlogn) algorithm that minimizes the time to complete all the tasks, and give a proof of why the solution is optimal.