EMANUELE
ROCCIA
1967318

Algorithm Design - Midterm
21-11-2023, *Time: 120 minutes*
Prof. Stefano Leonardi, Dr. Federico Fusco

Problem 1

Federico drives his Panda from Rome to Lignano Sabbiadoro. Federico can go for at most n kilometers without having a coffee break, and Federico's map gives the distances between Autogrills on his route. Federico wishes to make as few stops as possible along the way.

(a) Give an efficient algorithm by which Federico can determine at which Autogrill he should stop.

(b) Prove that your algorithm yields an optimal solution.

(c) Argue about the algorithm's computational complexity as a function of the input size and discuss its optimality.

Problem 2

We define the Escape Problem as follows. We are given a directed graph $G = (V, E)$ (picture a network of roads). A certain collection of nodes $X \subset V$ are designated as populated nodes, and a certain other collection $S \subset V$ are designated as safe nodes. (Assume that X and S are disjoint.) In case of an emergency, we want evacuation routes from the populated nodes to the safe nodes. A set of evacuation routes is defined as a set of paths in G so that (i) each node in X is the tail of one path, (ii) the last node on each path lies in S, and (iii) the paths do not share any edges. Such a set of paths gives a way for the occupants of the populated nodes to "escape" to S, without overly congesting any edge in G.

(a) Given G, X, and S, show how to decide in polynomial time whether such a set of evacuation routes exists.

(b) Suppose we have the same problem as in (a), but we want to enforce an even stronger version of the "no congestion" condition (iii). Thus, we change (iii) to say "the paths do not share any nodes." With this new condition, show how to decide in polynomial time whether such a set of evacuation routes exists.

(c) Provide an example with the same G, X, and S, in which the answer is yes to the question in (a) but no to the question in (b).

Problem 3

Consider the problem of making change for n cents using the smallest number of coins. Assume that each coin's value is an integer.

(a) Describe a greedy algorithm to make change consisting of quarters (25 cents), dimes (10 cents), nickels (5 cents), and pennies (1 cent). Prove that your algorithm yields an optimal solution.

(b) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

(c) Give an $O(nk)$-time dynamic programming algorithm that makes change for any set of k different coin denominations using the smallest number of coins, assuming that the coin of minimum denomination is a penny.

Problem 4 (Bonus)

Suppose you are consulting for a bank concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object containing a magnetic stripe with some encrypted data corresponding to a unique bank account. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are equivalent if they correspond to the same account. It's very difficult to read the account number of a bank card directly. Still, the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent. Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them into the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the tester.

Hint: *Try to follow a divide and conquer approach.*