CS577 Introduction to Algorithms

Final Exam 7/28/2021

__Notes__: (1) This is an open book exam, but you should never discuss with any classmate or seek assistance from any people. Misconduct will be reported to the school. (2) If you have questions please send a message to the instructor through Zoom chat (you can choose to use private message if you prefer). (3) You should show the major steps in your algorithm or proof. For questions that you do not have a full set of answer, partial answer or possible ideas could earn points. So normally do not leave your answer sheet blank.

Name__________________________ **Score**________________

1. **(10 points)** John was assigned a special task for completing one procedure of underwater pipeline installation. The pipeline is composed of *n* segments in a line and installation must be completed in sequence, i.e. one must complete segment k before moving to segment k+1. For the job at each segment, one tool needs to be used and the tools used at two segments could be the same or different. Since the tools are large, John can only carry three at the most with him. After finishing one segment, John will move on to the next segment under the water. If the next segment needs a tool already carried by him, he just needs to stay under the water to finish that segment (don't worry, he can stay underwater for long time since he has an oxygen tank in his back). But if the tool to be used was not carried by John, he has to swim out of the water and get the right tool. This means that he has to drop one of the current three tools to allow him carry the new one.

John's overall goal is to minimize the number of trips to swim out of water to switch a tool since it takes too much energy, and he already knows what tool is to be used at each segment before he starts working. In addition, he can only switch one tool during each trip. Now the question is, what strategy should John employ when he has to drop a tool so that he can carry the new tool to go back to work.

One example: the tool to be used in segments 1-10 are 5, 4,4,3,5,2, 5,3,1,5. Here is the series of tool switching operations as a result of John's decisions.

a. Suppose that John carried tools 3,4, and 5 at the beginning, then he has to come out of water to switch the tool at segment 6.

b. Now John (randomly) decides to drop tool 3 and have tools {2,4,5}. Then he needs to come out of water again at segment 8 to get tool 3.

c. This time John decided to drop tool 4 and have tools {2,3,5}. As a result, he needs to switch the tool again at segment 9 to get tool 1.

d. In this trip John decided to drop tool 3 and carry tools {1,2,5}. Then he does not need to come out of water again at segment 10 since he still carries tool 5. Overall, he spent 3 additional trips for fetching tools.

(1) (**2 points**) Someone proposes that John should drop the tool that he just used in the last segment. Show a counter example that this greedy strategy will not result in the least trips for switching the tools.
(2) (**1 points**) Design an optimal greedy algorithm idea to solve the problem. Note that you do not need to write out the pseudocode.
(3) (**7 points**) Prove the optimality of your algorithm using either "Exchange argument" or "Stay ahead" method.

2. **(10 points)** There are a group of n new employees in Bright Coding company like to hang out together before pandemic. For every activity (dining, watching a football game, boating etc.) they attended, the bill was always paid by a single person although everyone agreed to pay the fair share in the end. When the pandemic came and no more activities were planned, the group decided it is the time to settle up the bills. After retrieving the accounting details, they found that the money paid by employee 1,2,….n were $p_1, p_2, \ldots, p_n$. For those people who have paid less than the fair share, they need to write checks to pay those who have overpaid than their share. After the checks were written and cashed, no one should own money to any other.

However, there are two people who are supposed to write checks did not like to write more than one check, and one people who should get paid does not want more than one check. Your job is to design a network flow algorithm to find out whether the requirement of these three people can be satisfied while all payments will be settled, and if yes, find out the details of each check: the amount, and who paid who. Note that you should:

(1) **(5 points)** Construct a network flow model for this problem. Clearly state the meaning of each component (node, edge, capacity) of the flow network that you construct, draw out a flow network graph, describe your algorithm idea (you don't have to write the pseudocode), and give the computing complexity analysis of your algorithm.

(2) **(5 points)** Reason on the correctness of your algorithm. Specifically, you should:

   a. Show that a solution to the original problem will result in flows (i.e. satisfies the conservation and capacity conditions) in the flow network graph G.

   b. Show that the solution to the network flow problem you set up will give the solution for the original problem.

3. (**10 points**)
A store trying to analyze the behavior of its customers will often maintain a two-dimensional array A, where the rows correspond to its customers and the columns correspond to the products it sells. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i. Here's a tiny example of such an array A.

Name	Detergent	Beer	Juice	Diapers
Tom	0	6	0	3
Katie	2	3	0	0
Alan	0	0	0	7

We can now define the *Paradigmatic Subset* Problem as follows: Given a m X n table where position i, j indicates how many units of product j were bought by customer i, a list of products L, and an integer k, decide whether there exists a set of at most k customers that together have bought at least one unit of each product in L. Show that Paradigmatic Subset Problem is NP-complete.

Note that you should follow the template of writing proof for the NP Complete problem:
(1) (**2 points**) Prove that the *Paradigmatic Subset* problem is an NP problem (you should also analyze the computing complexity of the certifier);
(2) (**4 points**) Identify a well-known NP-Complete problem and show the construction of a *Paradigmatic Subset* problem out of the well-known NP-Complete problem.
(3) (**4 points**) Show the correctness of the reduction.