Assignment #5

Due: March 7, 2022 (11:59 pm)
(Total 40 marks)

CMPUT 204
Department of Computing Science
University of Alberta

A number of questions in this assignment are based on some problems and their solutions given in Problem Set #4. If a problem is related to some exercises of CLRS, possible solutions may also be found online. You may consult, adopt, or revise any of these solutions for your answers. If your algorithm is the same as the one given in the Problem Set, just indicate so; otherwise please specify.

Problem 1. (9 marks) Consider the following problem from Problem Set #4:

A native Australian named Oomaca wishes to cross a desert carrying only a single water bottle. He has a map that marks all the watering holes along the way. Assuming he can walk k miles on one bottle of water, design an efficient algorithm for determining where Oomaca should refill his bottle in order to make as few stops possible. Argue why your algorithm is correct.

You are asked to answer the following questions.

- What is the greedy choice in your algorithm?

  Solution: The greedy choice is for Oomaca to walk to the farthest possible watering hole $\leq k$ miles from his location

- Assume Oomaca starts with a full bottle of water and can walk 1 mile on one bottle of water. Suppose watering holes are located at the points given in the following list

  $$H = \{0.5, 1.2, 1.4, 1.8, 1.9, 2.2, 3.1, 3.5, 4.1, 4.5, 4.7, 5.0, 6.0\}$$

  measured in terms of miles from the starting point, where 6.0 also denotes the end point of the desert being crossed. Show all the location points where Oomaca refills his bottle by your algorithm.

  Solution: An algorithm using the greedy choice above will generate the array $\{0.5, 1.4, 2.2, 3.1, 4.1, 5.0, 6.0\}$

- Given a problem instance, suppose $S = \{q_1, q_2, \ldots, q_n\}$ is an optimal solution where each q_i is a location point to refill the bottle. Further assume S is sorted in ascending order, thus q_1 is the first point for a refill in S. Show that there is an optimal solution that contains your first point of refill. (**Hint:** This is to show that your first point of refill can be substituted into any optimal solution resulting in an optimal solution. **Do not write a full proof** - you are not asked to do that.)

  Solution: Let S be the solution generated by our algorithm, and let O be a known optimal solution. Set s_1, o_1 to be the first stopping point in each set. Note that, by nature of our greedy algorithm, $s_1 \geq o_1$. If they are equal, then we can substitute them trivially. If $s_1 > o_1$, then observe that the substitution can also be performed. By substituting s_1 for o_1, we only get closer to the other watering holes and therefore no other changes to the optimal solution need to be made.

Problem 2. (9 marks)

Consider the following question from Problem Set #4:

Describe an efficient greedy algorithm for making change for a specified value using a minimum number of coins, assuming there are four denominations of coins (called quarters, dimes, nickels, and pennies), with values 25, 10, 5, and 1, respectively. Argue why your algorithm is correct

In this assignment, you are asked to answer the following questions.

- Assume the change to be made is 98 cents. Show the solution generated by your algorithm.

  Solution: The solution generated by a greedy algorithm is: $\{25, 25, 25, 10, 10, 1, 1, 1\}$

- Given a problem instance, suppose $S = \{q_{25}, q_{10}, q_5, q_1\}$ is the solution generated by your algorithm, where q_i is the number of coins in denomination i. Argue why your solution is optimal.

  (**Hint:** Start by assuming $S' = \{q'_{25}, q'_{10}, q'_5, q'_1\}$ is an optimal solution and show that your greedy choices can be substituted into this optimal solution resulting in an optimal solution. In this question, you first argue why $S' = \{q_{25}, q'_{10}, q'_5, q'_1\}$ is an optimal solution, and then apply the same argument to the rest of greedy choices.)

  Solution: Note that

  - P' ≤ 4 or else 5 pennies could be replaced with a nickel to reduce the number of coins
  - N' ≤ 1; or else 2 nickels could be replaced with a dime
  - N' + D' ≤ 2 or else we have at least 2 dimes and a nickel that can be replaced with a quarter or 3 dimes which can be replaced with a quarter plus a nickel.

  Set $S' = \{q'_{25}, q'_{10}, q'_5, q'_1\}$ to be an optimal solution. Note that, by nature of our greedy algorithm, $q'_{25} \leq q_{25}$. Also, since S' is optimal it must be that $q'_{25} + q'_{10} + q'_5 + q'_1 \leq q_{25} + q_{10} + q_5 + q_1$. Suppose that $q'_{25} < q_{25}$. For this to be true, there needs to be some combination of 10, 5, and 1 cent coins which combine to form 25 cents. But this means that replacing these coins with a 25 cent coin would reduce the number of coins in the optimal solution, and so the optimal solution is not optimal. This is a contradiction, and so $q'_{25} = q_{25}$. The proof proceeds in much the same way for the other denominations of coins. If $q'_{10} < q_{10}$ then there is a combination of nickels and pennies to replace it. If $q'_5 < q_5$ then there is a combination of pennies to replace it. Thus, since $25q'_{25} + 10q'_{10} + 5q'_5 = 25q_{25} + 10q_{10} + 5q_5$, we must have that $q'_1 = q_1$. We are done.

- Give an example set of denominations of coins so that a greedy change making algorithm will not use the minimum number of coins. Your answer must include denomination 0.01 so that a solution is always guaranteed, and must also be different from the one given in Problem Set #4, where the denominations are 0.25, 0.24, 0.01.

  Solution: Any set of coins $C = \{c_1, c_2, ..., c_n\}$ where $\forall i \in \{1, ..., n-1\}, c_i < c_{i+1}$ and $\exists i \in \{1, ..., n-1\}$ s.t. $2 * c_i > c_{i+1}$ and $0.01 \in C$ satisfies the condition in the question.

Problem 3. (9 marks) Consider the following problem from Problem Set #4:

In the art gallery guarding problem, a line L represents a long art gallery hallway. We are given a set of location points on it $X = \{x_0, x_1, \ldots, x_{n-1}\}$ of real numbers that specify the positions of the paintings along the hallway. A single guard can guard paintings standing at most 1 meter from each painting on either side. Propose an algorithm that finds the optimal number of guards to guard all the paintings along the hallway.

In this assignment, you are asked to answer the following questions.

- Given the set of location points

$$X = \{0.1, 0.5, 1.2, 1.8, 2.3, 3.1, 4.5, 6.7, 7.5\}$$

that specify the positions of the paintings along the hallway (measured in meters from the beginning of the gallery hallway), show all the position points at which your algorithm places a guard.

Solution: A greedy algorithm will place the guards at positions $\{1.1, 3.3, 5.5, 7.7\}$

- Argue that, for any given problem instance, there is an optimal solution that places the first guard at the same position point as your algorithm does.

(**Hint:** This is to show that, given any problem instance, the placement of your first guard, say at q_1, can always be substituted into any optimal solution resulting in an optimal solution. Start your argument by assuming an optimal solution $S = \{p_1, p_2, \ldots, p_k\}$. **Do not write a full proof**.)

Solution: Let S be the solution generated by our algorithm, and let O be a known optimal solution. Set s_1, o_1 to be the left-most guard in each set of guards. Note that, by nature of our greedy algorithm, $s_1 \geq o_1$. If they are equal, then we can substitute them trivially. If $s_1 > o_1$, then observe that the substitution can also be performed. This is because o_1 is the left-most guard and so moving her right does not prevent her from guarding any paintings on the left. Furthermore, any paintings on the right are not affected because we only get closer to them.

Problem 4. (13 marks) A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the ith customer has to wait $\sum_{j=0}^{i-1} t_j$ minutes, where $1 \leq i \leq n$ and $t_0 = 0$.

We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Give an efficient algorithm for computing the optimal order in which to process the customers. Prove that your algorithm gives an optimal solution.

Solution

Denote the problem input by $A = [t_0, t_1, t_2, \ldots, t_n]$. We sort A in ascending order to obtain $B = [t_{\pi_0}, t_{\pi_1}, \ldots, t_{\pi_n}]$, such that $t_{\pi_0} \leq t_{\pi_1} \leq \cdots \leq t_{\pi_n}$, where π_is are the new indices after sorting.

Then we firstly arrange customer π_1 to the server, secondly customer π_2, and so on, *i.e.*, in the order after sorting. The total waiting time of the oder π is

$$\begin{aligned} T_\pi &= \sum_{i=1}^{n} (\text{time spent waiting by customer } i) \\ &= \sum_{i=1}^{n} \sum_{j=0}^{i-1} t_{\pi_j} \\ &= \sum_{j=0}^{n-1} t_{\pi_j} \cdot (n - j) \end{aligned}$$

Now we prove the above algorithm gives an optimal solution. Suppose there is another order of arrangement $\phi : \phi_1, \phi_2, \ldots, \phi_n$, such that $\phi_1 \neq \pi_1$, and $\phi_p = \pi_1$, for some $p > 1$. We can obtain ϕ' from ϕ by exchanging ϕ_1 with ϕ_p as follows,

$$\begin{array}{lccccccl} \phi: & \phi_1 & \phi_2 & \ldots & \phi_p & \ldots & \phi_n & \\ \phi': & \phi_p & \phi_2 & \ldots & \phi_1 & \ldots & \phi_n & \text{(exchange } \phi_1 \text{ with } \phi_p\text{)} \\ \phi': & \pi_1 & \phi_2 & \ldots & \phi_1 & \ldots & \phi_n & (\phi_p = \pi_1) \end{array}$$

We want to show this exchange will not increase the total waiting time, *i.e.*, $T_{\phi'} \leq T_\phi$. We have,

$$\begin{aligned} T_\phi - T_{\phi'} &= \sum_{j=0}^{n-1} t_{\phi_j} \cdot (n-j) - \sum_{j=0}^{n-1} t_{\phi'_j} \cdot (n-j) \\ &= \sum_{j=0}^{n-1} (t_{\phi_j} - t_{\phi'_j}) \cdot (n-j) \\ &= (t_{\phi_1} - t_{\phi_p}) \cdot (n-1) + (t_{\phi_p} - t_{\phi_1}) \cdot (n-p) \qquad (\text{since } t_{\phi_j} = t_{\phi'_j} \quad \forall j \neq 1, p) \\ &= (t_{\phi_1} - t_{\phi_p}) \cdot (p-1) \\ &= (t_{\phi_1} - t_{\pi_1}) \cdot (p-1) \\ &\geq 0 \qquad\qquad (\text{since } p > 1,\ t_{\pi_1} \leq t_{\phi_1}) \end{aligned}$$

Similarly, if we exchange ϕ_2 with π_2 in the order ϕ', we still will not increase the total waiting time. Repeatedly exchanging this way for at most $n-1$ times (equivalent with the selection sort), we will finally get the order π, and $T_\pi \leq \cdots \leq T_{\phi'} \leq T_\phi$. Since ϕ can be an arbitrary permutation, we have T_π is the smallest total waiting time, *i.e.*, the algorithm gives an optimal solution.