Problem 1.

We consider the algorithm presented in problem 1 of problem set #4.

- If Oomaca can make it to the next location before his bottle is empty, he will not stop. This means he will go as far as possible with each bottle.
- He stops at
  1. Location 1 (0.5),
  2. Location 3 (1.4),
  3. Location 6 (2.2),
  4. Location 7 (3.1),
  5. Location 9 (4.1), and
  6. Location 12 (5.0).
- Let $S = \{q_1, q_2, \ldots, q_n\}$ be an optimal solution and $S' = \{x_1, x_2, \ldots, x_{n'}\}$ be a solution generated by our algorithm. We wish to show the first greedy choice x_1 in S' can be used to replace one or more choices in S so the altered solution of S is still optimal.

  Since x_1 is the furthest hole Oomaca can go with one bottle, $q_1 \leq x_1$. We also know he only needs one bottle to go to x_1 so there is no disadvantage to stopping at x_1 instead of q_1. This implies stopping at x_1 instead of q_1 doesn't result in $q_2, q_3, \ldots, q_n$ needing to change.

Problem 2.

We consider the algorithm presented in problem 2 of problem set #4.

- We would have 3 quarters, 2 dimes, and 3 pennies.
- Assume $S' = \{q'_{25}, q'_{10}, q'_5, q'_1\}$ is an optimal solution and let $S = \{q_{25}, q_{10}, q_5, q_1\}$ is our greedy solution. We show our greedy choices can be substituted into S' and still result in an optimal solution.

  S' is an optimal solution because

  1. $q'_1 < 5$ since a nickel could be used otherwise,
  2. $q'_5 < 2$ since a dime could be used otherwise, and
  3. $q'_5 + q'_{10} < 3$ since a quarter could be used otherwise (we either have 3 dimes which can be optimized to 1 nickel and 1 quarter, or we have 2 dimes and a nickel which can be optimized to 1 quarter).

  The above implies

  1. There are at most 4 pennies,
  2. There are at most 9 pennies and nickels, and
  3. There are at most 24 pennies, nickels, and dimes.

  Since our greedy solution always maximizes q_{25}, we know $q'_{25} \leq q_{25}$. In fact, we can say $q'_{25} = q_{25}$: if $q'_{25} < q_{25}$ then $q_{25} - q'_{25} > 0$ meaning dimes, nickels, and pennies are being used instead of $q_{25} - q'_{25}$ quarters which contradicts what makes S' an optimal solution.

  Similarly, our greedy solution always maximizes q_{10} so $q'_{10} \leq q_{10}$. In fact, we can say $q'_{10} = q_{10}$: if $q'_{10} < q_{10}$ then $q_{10} - q'_{10} > 0$ meaning nickels and pennies are being used instead of $q_{10} - q'_{10}$ dimes which contradicts what makes S' an optimal solution.

  We also maximize q_5 so $q'_5 \leq q_5$. In fact, we can say $q'_5 = q_5$: if $q'_5 < q_5$ then $q_5 - q'_5 > 0$ meaning pennies are being used instead of $q_5 - q'_5$ nickels which contradicts what makes S' an optimal solution.

  We finally conclude that $q'_1 = q_1$ since $25q'_{25} + 10q'_{10} + 5q'_5 + q'_1 = 25q_{25} + 10q_{10} + 5q_5 + q_1$. Thus, our greedy solution finds the optimum number of coins.
- If the denominations are 0.50, 0.40, and 0.01, then a greedy algorithm for making change for 80 cents would give 1 50¢ coin and 30 pennies for a total of 31 coins. However, the optimal solution is simply 2 40¢ coins for a total of 2 coins.

Problem 3.

We consider the algorithm presented in problem 3 of problem set #4.

- Guards are placed at
  1. $p_0 = 1.1$ (covers 0.1, 0.5, 1.2, 1.8),
  2. $p_1 = 3.3$ (covers 2.3, 3.1),
  3. $p_2 = 5.5$ (covers 4.5), and
  4. $p_3 = 7.7$ (covers 6.7, 7.5).
- Let $S = \{p_1, p_2, \ldots, p_k\}$ be an optimal solution (uses the minimum number of guards) and $S' = \{q_1, q_2, \ldots, q_k\}$ be a solution generated by our algorithm. We need to show that the following two conditions hold:
  1. $x_1, \ldots, x_p$ such that $x_p \leq q_1$ must be guarded, and
  2. $p_1 = q_1$ given any problem instance.

  x_1 is the first unguarded painting. Our algorithm places a guard at $q_1 = x_1 + 1$, satisfying the first condition ($x_1, \ldots, x_p$ where $x_p \leq q_1$ will be guarded by placing a guard at $x_1 + 1$). This implies that the first guard's location in an optimal solution p_1 cannot be placed to the right of $q_1 = x_1 + 1$ since placing p_1 any further to the right means x_1 will be unguarded. We can therefore say $p_1 \leq q_1$. We also notice we can replace p_1 with q_1 since this would not affect the coverage of x_1 and would not expose paintings protected by a guard at q_1. Therefore we also satisfy the second condition.

Problem 4.

An efficient algorithm for computing the optimal order in which to process the customers is to serve the customers in increasing order of t_i. Using a sorting algorithm such as quicksort or mergesort, the algorithm would run in $O(n \log(n))$ time.

Given the problem input $A = [t_0, t_1, \ldots, t_n]$, let $B = [t_{\pi_0}, t_{\pi_1}, \ldots, t_{\pi_n}]$ such that $t_{\pi_0} \leq t_{\pi_1} \leq \ldots \leq t_{\pi_n}$ be the result of running our algorithm where each π_i is the new index after sorting. We now prove the algorithm computes the optimal order via proof by contradiction.

Suppose that our solution B is not optimal. This means there is some solution $C = [t_{\alpha_0}, t_{\alpha_1}, \ldots, t_{\alpha_n}]$ that results in a lower total waiting time. Let j be the position at which B and C first differ. Since $t_{\pi_j} \neq t_{\alpha_j}$, we have two cases:

- $t_{\pi_j} > t_{\alpha_j}$: this scenario is impossible. Since B is sorted and position j is the first position where B and C differ, any t_α smaller than t_{π_j} must come before position j.
- $t_{\pi_j} < t_{\alpha_j}$: solution C results in a higher total waiting time than solution B. If $t_{\pi_j} < t_{\alpha_j}$, the total wait time will increase at least $t_{\alpha_j} - t_{\pi_j}$ minutes.

We see that solution C cannot be an optimal solution, contradicting the original assumption. So our solution B must be optimal.