Greedy Strategies Group Exercises

Please work the following problems in groups of 2 or 3. Use additional paper as needed, and staple the sheets together before turning them in. **ONLY TURN IN 1 WORKSHEET/ANSWERS PER TEAM.**

Assume that we have a set of coins, and we have an unlimited number of each coin. For our example we will restrict the set of coins to the US coins: \( c = \{c_1, c_2, c_3, c_4, c_5\} \) where \( c_5 \) is worth 1¢, \( c_4 \) is worth 5¢, \( c_3 \) is worth 25¢, \( c_2 \) is worth 50¢, and \( c_1 \) is worth 100¢. Note the coins are sorted in descending order: \( c_1 > c_2 > c_3 > c_4 > c_5 \). Anytime we pay for something with cash and have change returned, the goal of the cashier is to give us the fewest coins possible to make up the value of the change.

Your task is to explore this problem and the various ways to solve it, including divide and conquer, dynamic programming, and greedy strategies: Given an amount \( A \) and the set of \( n \) coins \( c = \{c_1, c_2, \ldots, c_n\} \), find the minimum number of coins needed to make the amount \( A \). Call this minimum set \( MC \). **Again, we restrict the problem to US coins.**

1. Assume we are going to solve the problem using recursion and the divide and conquer approach. Characterize the subproblem structure. Answer the following questions.
   a. What are the subproblems that are smaller instances of the overall problem?

   b. How did you create the subproblems?

   c. How many subproblems are there?

   d. Is the subproblem structure optimal? Why/why not?
2. Write the recurrence for the solution.

3a. Since we are first exploring using a divide and conquer approach, what is one potential complexity problem we might have? (Hint: Think about what the call graphs you drew on the dynamic programming worksheet for the recursive version of the rod cutting problem.)

3b. What approach could we use instead of divide and conquer to solve the problem you wrote about in 3a?

4. How can we further simplify/reduce the number of subproblems, based on the choice of the coin that we try?

5. What technique does this simplification suggest we should try?
6. We’ve already done steps 1-3 of the greedy strategy. What are these and which of the problems above addressed them?

7. What do we need to prove for the next step of the greedy strategy? What should the central idea of this argument be?

8. Write pseudo-code for a greedy algorithm to determine the minimum set of coins needed to make an amount. The coin set is \( c = \{ c_1, c_2, \ldots, c_n \} \), the amount is \( A \), and the minimum set of coins is \( MC \).

9. If you were going to code this in Python3, what data structures would you use, and for what purposes?