Problem 1: The following problem arises in the context of DNA analysis. Let \( \Sigma \) be an alphabet. Given two strings \( X = \langle x_1, x_2, \ldots, x_n \rangle \) and \( Y = \langle y_1, \ldots, y_m \rangle \) over \( \Sigma \) we are looking for the longest common subsequence of \( X \) and \( Y \).

We say that \( Z = \langle z_1, \ldots, z_k \rangle \) is a subsequence of \( X \) if there exists a strictly increasing sequence of indices \( < i_1, \ldots, i_k > \) such that \( x_{i_j} = z_j \) for all \( j = 1, \ldots, k \). Note that a subsequence does not have to be formed of adjacent elements in the original string: for example if \( X = \langle a, b, c, b, d, a, b \rangle \) then \( Z = \langle b, c, d, b \rangle \) is a valid subsequence (the indices that correspond to \( Z \) in \( X \) are \( 2, 3, 5, 7 \)).

Given a dynamic programming algorithm to compute the longest common subsequence of \( X \) and \( Y \).

Problem 2: You are working at the cash register at the local supermarket and you have to make change for \( n \) cents. To save time (and be nice to your customers) you want to do that with the fewest number of coins.

- Describe a greedy algorithm to make change when the coins in the drawer are the usual U.S. coin denominations: quarters, dimes, nickels and pennies. Argue that the greedy algorithm yields an optimal solution.

- What if you were working in the country of Powerpolis where coins denominations are \( c^0, c^1, c^2, \ldots, c^k \) for some integer \( c > 1 \) and \( k \leq 1 \). Does the greedy algorithm still yield an optimal solution?

- Show an example of a set of coin denominations for which the greedy algorithm does not yield an optimal solution.

Problem 3: A sequence of \( n \) operations is performed on a data structure. The \( i^{th} \) operation costs \( i \) if \( i \) is a power of 2, otherwise it costs 1. Determine the total cost of the sequence of operations, and the amortized cost per operation. You can assume that \( n \) is a power of 2.