Problem 1. (20 points)

Prove (by induction on \( k \)) the following property on the \textsc{Union-Find} data structure. For all root nodes \( x \) of rank \( k \), the size of the tree rooted at \( x \) is at least \( 2^k \).

Problem 2. (30 points)

Consider a variation of the \textsc{Union-Find} data structure, with the union-by-rank heuristic but without path compression. That is, we implement \textsc{Make-Set} and \textsc{Union} as usual, but we do not reset the parent pointers in \textsc{Find-Set}. Show that there is some sequence of \( n \) calls to \textsc{Make-Set}, some number (at most \( n \)) of calls to \textsc{Union}, and \( m \) calls to \textsc{Find-Set} that require \( \Omega(m \log n) \) time from this suboptimal implementation.

Problem 3. (50 points)

In the United States, coins are minted with denominations of 1, 5, 10, 25, and 50 cents. Now consider a country whose coins are minted with denominations of \( \{d_1, \ldots, d_k\} \) units. We want to devise an algorithm that will enable us to make change of \( n \) units using the minimum number of coins.

1. The greedy algorithm for making change repeatedly uses the biggest coin smaller than the amount to be changed until it is zero. Provide a greedy algorithm for making change of \( n \) units using US denominations. Prove its correctness and analyze its time complexity.
2. Show that the greedy algorithm does not always give the minimum number of coins in a country whose denominations are \( \{1, 6, 10\} \).

3. Give an efficient dynamic programming algorithm that correctly determines the minimum number of coins needed to make change of \( n \) units using denominations \( \{d_1, \ldots, d_k\} \). Analyze its running time.