1. Let \( G = (V, E) \) be a connected undirected graph in which all edges have weight either 1 or 2. Give an \( O(|V| + |E|) \) algorithm to compute a minimum spanning tree of \( G \). Justify the running time of your algorithm. (Note: You may either present a new algorithm or just show how to modify an algorithm taught in class.)

2. Give an \( O(n^2) \) time dynamic programming algorithm to find the longest monotonically increasing subsequence of a sequence of \( n \) numbers (i.e., each successive number in the subsequence is greater than or equal to its predecessor). For example, if the input sequence is \( \langle 5, 24, 8, 17, 12, 45 \rangle \), the output should be either \( \langle 5, 8, 12, 45 \rangle \) or \( \langle 5, 8, 17, 45 \rangle \).

   Hint: Let \( d[i] \) be the length of the longest increasing subsequence whose last item is item \( i \).

3. The subset sum problem is: Given a set of \( n \) positive integers, \( S = \{x_1, x_2, \ldots, x_n\} \) and an integer \( W \) determine whether there is a subset \( S' \subseteq S \) such that the sum of the elements in \( S' \) is equal to \( W \). For example, if \( S = \{4, 2, 8, 9\} \) and \( W = 11 \), then the answer is “yes” because there is a subset \( S' = \{2, 9\} \) whose elements sum to 11. Give a dynamic programming solution to the subset sum problem that runs in \( O(nW) \) time. Justify the correctness and running time of your algorithm.

4. Give an \( O(nW) \) dynamic programming algorithm for the 0-1 knapsack problem where \( n \) is the number of items and \( W \) is the max weight that can fit into the knapsack. Recall that the input is \( i \) items with given weights \( w_1, w_2, \ldots, w_n \) and associated values \( v_1, v_2, \ldots, v_n \) and the objective is to choose a set of items with weight \( \leq W \) with maximum value.

Now suppose that you are given two knapsacks with the same max weight. Give an \( O(nW^2) \) dynamic programming algorithm for finding the maximum value of items that can be carried by the two knapsacks.

5. Suppose you want to make change for \( n \) (HK) dollars using the fewest number of coins. Assume that each coin’s value is an integer.

   Give an \( O(nk) \)-time dynamic programming algorithm that makes change for any set of \( k \) different coin denominations, assuming the set always contains a 1-dollar coin (so a solution always exists).

   Let the coin denominations be \( d_1, d_2, \ldots, d_k \).