1. Consider a modification to the activity-selection problem in which each activity $a_i$ has, in addition to a start and finish time, a value $v_i$. The objective is no longer to maximize the number of activities scheduled, but instead to maximize the total value of the activities scheduled. That is, we wish to choose a set $A$ of compatible activities such that $\sum_{a_k \in A} v_k$ is maximized. Give a polynomial-time algorithm for this problem.

2. Prove that a binary tree that is not full cannot correspond to an optimal prefix code.

3. What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

   $a: 1 \quad b: 1 \quad c: 2 \quad d: 3 \quad e: 5 \quad f: 8 \quad g: 13 \quad h: 21$

   Can you generalize your answer to find the optimal code when the frequencies are the first $n$ Fibonacci numbers?

4. Generalize Huffman’s algorithm to ternary codewords (i.e., codewords using the symbols 0, 1, and 2), and prove that it yields optimal ternary codes.

5. If the set of stack operations included a MULTIPUSH, which pushes $k$ items onto the stack, would the $O(1)$ bound on the amortized cost of stack operations continue to hold?

6. Show that if a DECREMENT operation were included in the $k$-bit counter example, $n$ operations could cost as much as $\Theta(nk)$ time.

7. Suppose we perform a sequence of $n$ operations on a data structure in which the $i$th operation costs $i$ if $i$ is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.