Problem 9-1 (Fibonacci Meets Huffman) 10 points

Recall, Fibonacci numbers are defined by \( f_0 = f_1 = 1 \) and \( f_i = f_{i-1} + f_{i-2} \) for \( i \geq 2 \).

(a) (2 points) What is the optimal Huffman code for the following set of frequencies which are the first 8 Fibonacci numbers?

(b) (4 points) Let \( S_1 = 2 = f_0 + f_1 \) and \( S_i = S_{i-1} + f_i = \ldots = f_i + f_{i-1} + \ldots f_1 + f_0 \) (for \( i > 1 \)) be the sum of the first \( i \) Fibonacci numbers. Prove that \( S_i = f_{i+2} - 1 \) for any \( i \geq 1 \).

(c) (4 points) Generalize your solution to part (a) to find the shape of the optimal Huffman code for the first \( n \) Fibonacci numbers. Formally argue that your tree structure is correct, by using part (b).

Problem 9-2 (Frugal Tourist) 12 Points

You want to travel on a straight line from city A to city B which is \( N \) miles away from A. For concreteness, imagine a line with A being at 0 and B being at \( N \). Each day you can travel at most \( d \) miles (where \( 0 < d < N \)), after which you need to stay at an expensive hotel. There are \( n \) such hotels between 0 and \( N \), located at points \( 0 < a_1 < a_2 < \ldots < a_n = N \) (the last hotel is at B). Luckily, you know that \( |a_{i+1} - a_i| \leq d \) for any \( i \) (with \( a_0 = 0 \)), so that you can at least travel to the next hotel in one day. You goal is to complete your travel in the smallest number of days (so that you do not pay a fortune for the hotels).

Consider the following greedy algorithm: “Each day, starting at the current hotel \( a_i \), travel to the farthest hotel \( a_j \) s.t. \( |a_j - a_i| \leq d \), until eventually \( a_n = N \) is reached.” That is, if several hotels are within reach in one day from your current position, go to the one closest to your destination.

(a) (6 points) Formally argue that this algorithm is correct using the “Greedy Always Stays Ahead” method.

(Hint: Think about how to define \( F_i(Z) \). For this problem, the name of the method is really appropriate.)

(b) (6 points) Formally argue that this algorithm is correct using the “Local Swap” method. More concretely, given some hypothetical optimal solution \( Z \) of size \( k \) and the solution \( Z^* \) output by greedy, define some solution \( Z_1 \) with the following two properties: (1) \( Z_1 \) is no worse than \( Z \); (2) \( Z_1 \) agrees with greedy in the first day travel plan. After \( Z_1 \) is defined, define \( Z_2 \) s.t. (1) \( Z_2 \) is no worse than \( Z_1 \); (2) \( Z_2 \) agrees with greedy in the first two days travel plan. Continue in this fashion until you end up with the greedy solution.
Problem 9-3 (Allsafe to the Rescue) 13 Points

E-Corp has set up $n$ servers, all of which turn out to suffer from security vulnerabilities. In order to stop the leaks, E-Corp’s CTO calls in a special team of engineers from the renowned Allsafe security firm. Server $i$ takes $a_i$ hours to fix and leaks $b_i$ GB per hour. Due to the level of expertise required and personnel shortage, the Allsafe team can only work on a single server at any given time.

In this task you are to devise an algorithm that finds a repair schedule that minimizes the amount of data leaked from the beginning of Allstate’s engagement until all servers are fixed.

(a) (3 points) As a warm-up, assume there are only two servers with repair times $a_1$ and $a_2$ and leakage rates $b_1$ and $b_2$. How many GB of data are leaked (in terms of these four values) while patching up both servers if the first server is taken care of first? How many are leaked if the order is reversed?

(b) (2 points) Argue why it is better to fix the first server first if and only if $a_1/b_1 \leq a_2/b_2$.

(c) (2 points) Argue that in any optimal solution, any two servers $i$ and $j$ adjacent in the schedule must satisfy $a_i/b_i \leq a_j/b_j$.

(d) (2 points) Devise a greedy algorithm that finds an optimal scheduling (neither pseudo-code nor analysis required here).

(e) (3 points) Prove the correctness of your greedy algorithm using the “Local Swap” method and your insights above. You may use part (c) even if you did not solve it.

(f) (1 points) Analyze the running time of your algorithm.