Problem Set 1

Problem 1 (20 points, Problem 15.1-2). Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the density of a rod of length \( i \) to be \( p_i \), that is, its value per inch. The greedy strategy for a rod of length \( n \) cuts off a first piece of length \( i \), where \( 1 \leq i \leq n \), having maximum density. It then continues by applying the greedy strategy to the remaining piece of length \( n - i \).

Solution. Here is a counterexample to prove that greedy algorithm doesn’t provide an optimal solution every time: Let \( p_1 = 1, p_2 = 20, p_3 = 33, p_4 = 36 \). Let the length of the rod be 4 inches. As we run the Greedy algorithm, the first cut for the rod would be of length 3 whose density is maximum. As a result, the total price would be 34. However, if we cut the rod into two pieces of length 2, the total price would be 40 which is optimal. Hence, the greedy algorithm does not always produce an optimal solution.

Problem 2 (20 points, Problem 15.1-3). Consider a modification of the rod-cutting problem in which, in addition to a price \( p_i \) for each rod, each cut incurs a fixed cost of \( c \). The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem.

Solution. Let \( r_n \) be the max revenue for length \( n \). Then we have \( r_n = \max_{1 \leq i < n} \{ p_n, p_i + r_{n-i} - c \} \).

Pseudocode:

<table>
<thead>
<tr>
<th>Input:</th>
<th>( p, n, c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>( r_n )</td>
</tr>
</tbody>
</table>

1: let \( r_0 = 0 \);
2: for \( j = 1 \) to \( n \) do
3:   let \( q = p_j \)
4:   for \( i = 1 \) to \( j - 1 \) do
5:     let \( q = \max \{ q, p_i + r_{j-i} - c \} \)
6:   end for
7:   let \( r_j = q \)
8: end for

The time complexity of this algorithm is \( O(n^2) \).