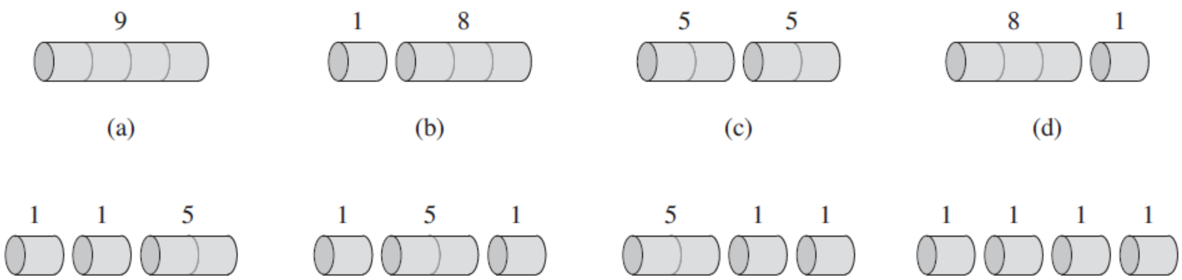
Dynamic Programming (14 points)

The rod-cutting problem is the following. Given a rod of length n inches and a table of prices p_i for $i = 1, 2, \dots, n$, determine the maximum revenue r_n obtain- able by cutting up the rod and selling the pieces. Note that if the price p_n for a rod of length n is large enough, an optimal solution may require no cutting at all.

Consider the case when $n = 4$. The figure below shows all the ways to cut up a rod of 4 inches in length, including the way with no cuts at all. We see that cutting a 4-inch rod into two 2-inch pieces produces revenue $p_2 + p_2 = 5 + 5 = 10$, which is optimal.



(a) (b) (c) (d) (e) (f) (g) (h)

1. [2 points] 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/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 $n - i$.

2. [3 points] 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.

3. [3 points] Modify MEMOIZED-CUT-ROD to return not only the value but the actual solution, too.

4. [2 points] Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 10, 3, 12, 5, 50, 6 \rangle$. What is the total cost?

5. [4 points] Give a recursive algorithm *Matrix-Chain-Multiply*(A, s, i, j) that actually performs the optimal matrix-chain multiplication, given the sequence of matrices $\langle A_1, A_2, \ldots, A_n \rangle$, the s table computed by *Matrix-Chain-Order*, and the indices i and j. (The initial call would be *Matrix-Chain-Multiply*$(A, s, 1, n)$.