1. 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 prices of the pieces minus the cost of making the cut.

(a) Give a dynamic-programming algorithm to solve this modified problem, including the mathematical expression for the maximum revenue and the pseudocode.

(b) Show the maximum revenue $r_j$ and the optimal size $s_j$ of the first piece to cut off, when $c = 1$ and

<table>
<thead>
<tr>
<th>length $i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>price $p_i$</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

2. Find an optimal parenthesization of a matrix-chain product $A_1A_2A_3A_4A_5A_6$, whose sequence of dimensions is $p = (5, 6, 3, 7, 5, 3, 4)$.

(a) Compute the dynamic programming $m$-table and $s$-table.

(b) Show the optimal parenthesization from computed $s$-table.

3. For the sequences $X = \langle B, C, A, A, B, A \rangle$ and $Y = \langle A, B, A, C, B \rangle$,

(a) Follow the pseudocode LCS-LENGTH to compute the dynamic programming $c$-table and $b$-table for finding the longest common subsequence (LCS) of $X$ and $Y$.

(b) Follow the computed $b$-table to list the LCS.

4. For the sequences $X = \langle B, C, A, A, B, A \rangle$ and $Y = \langle A, B, A, C, B \rangle$, compute the edit distance by showing the dynamic programming $e$-table, and the best alignment.

5. Two character strings may have many common substrings. Substrings are required to be contiguous in the original string. For example, $photograph$ and $tomography$ have several common substrings of length one (i.e., single letters), and common substrings $ph$, $to$, and $ograph$ (as well as all the substrings of $ograph$). The maximum common substring (MCS) length is 6.

Let $X = x_1x_2\cdots x_m$ and $Y = y_1y_2\cdots y_n$ be two character strings.

(a) Give a dynamic programming algorithm to find the MCS length for $X$ and $Y$.

(b) Analyze the worst-case running time and space requirements of your algorithm as functions of $n$ and $m$.

(c) Demonstrate your dynamic programming algorithm for finding the MCS length of character strings $algorithm$ and $logarithm$. by constructing the corresponding dynamic programming tables.

6. Suppose that the denominations of the coins in a country are $d_1 < d_2 < \cdots < d_n$ (e.g., 1, 5, 10, 25 for the United States). The problem to consider is:

Given an integer $M$, what is the minimum number of coins needed to make $M$ cents in change?

(a) Under what condition(s) on the denominations $d_i$ are you able to make change for any amount $M$?

(b) Give a dynamic programming algorithm to solve the problem.

(c) How fast is your algorithm?
7. The 0-1 knapsack problem. Given six items \( \{(v_i, w_i)\} \) for \( i = 1, 2, \ldots, 6 \) as follows:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( v_i )</th>
<th>( w_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

and the total weight \( W = 100 \), where \( v_i \) and \( w_i \) are the value and weight of item \( i \), respectively.

(a) Find the greedy solutions by using following strategies:

1. Greedy by value, i.e., at each step select from the remaining items the one with the highest value
2. Greedy by weight, i.e., at each step select from the remaining items the one with the least weight.
3. Greedy by value density, i.e., at each step select from the remaining items with the largest value per pound ratio \( v_i/w_i \).

Are these greedy solutions optimal? Comment your findings.

(b) Solve the preceding 0-1 knapsack problem by using dynamic programming and comment your findings.

You need to outline your DP algorithm, not just given the answer. Since the value of \( W \) is pretty large, you should write a short program to do the calculation and attach your program in your homework return.

8. (Option) Sequence alignment. When a new gene is discovered, a standard approach to understanding its function is to look through a database of known genes and find close matches. The closeness of two genes is measured by the extent to which they are aligned. To formalize this, think of a a gene as being a long string over the alphabet \( \Sigma = \{A, C, G, T\} \).

Consider two genes (strings) \( x = 'ATGCC' \) and \( y = 'TACGCA' \). An alignment of \( x \) and \( y \) is a way of matching up these two strings by writing them in columns, for instance:

\[
\begin{array}{c}
- & A & T & - & G & C & C \\
T & A & - & C & G & C & A \\
\end{array}
\]

Here the “-” indicates a “gap”. The characters of each string must appear in order, and each column must contain a character at least one of the strings. The score of an alignment is specified by a scoring matrix \( \delta \) of size \((|\Sigma| + 1) \times (|\Sigma| + 1)\), where the extra row and column are to accommodate gaps. For example the preceding alignment has the following score:

\[
\delta(-, T) + \delta(A, A) + \delta(T, -) + \delta(-, G) + \delta(G, G) + \delta(C, C) + \delta(C, A).
\]

Give a dynamic programming algorithm that takes as input two strings \( x[1 \ldots n] \) and \( y[1 \ldots m] \) and a scoring matrix \( \delta \), and returns the highest-scoring alignment. The running time should be \( O(mn) \).