SOLUTIONS FOR HOMEWORK 4

15.1-2

Consider the case \( n = 3 \) with the following price table

<table>
<thead>
<tr>
<th>Length ( i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price ( p_i )</td>
<td>10</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Density ( d_i )</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

The greedy strategy will give \( p_1 + p_2 = 11 \), however our optimal solution is \( p_3 = 12 \).

15.1-3

Use the similar notations as our textbook, we have

\[
r_n = \max_{1 \leq i \leq n} (p_i - c + r_{n-i})
\]

and we have boundary condition \( r_0 = 0 \) (this is important and you will lose points if you do not explicitly write it down).

We also give the Pseudocode as Algorithm 1.

15.2-1

(Table omitted) The optimal parenthesization is \((A_1 A_2)((A_3 A_4)(A_5 A_6))\), with the minimum number of scalar multiplication 2010.

15.2-2

We borrow the implementation of \textsc{Matrix-Multiply} from our textbook, pages 371. Pseudocode is presented by Algorithm 2.
Algorithm 1: Rod-Cut($p, n, c$)

1. $r_0 \leftarrow 0$
2. for $i = 1$ to $n$
3.   $r_i \leftarrow p_i$
4.   for $j = 1$ to $i - 1$
5.     $r_i \leftarrow \max\{r_i, p_j - c + r_{i-j}\}$
6. return $r_n$

Algorithm 2: Matrix-Chain-Multiply($A, s, i, j$)

1. if $i == j$ then
2.   return $A_i$
3. else
4.   $c \leftarrow s[i,j]$
5.   $M \leftarrow$ Matrix-Chain-Multiply($A, s, i, c$)
6.   $N \leftarrow$ Matrix-Chain-Multiply($A, s, c + 1, j$)
7. return Matrix-Multiply($M, N$)