### Advanced Algorithm Homework 1
#### Result and Solution

<table>
<thead>
<tr>
<th>ID</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>Final</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>2554</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.7</td>
<td></td>
</tr>
<tr>
<td>6288</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4208</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.7</td>
<td>6</td>
</tr>
<tr>
<td>0996</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8.5</td>
</tr>
<tr>
<td>8239</td>
<td>9.5</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>9.4</td>
<td>5</td>
</tr>
<tr>
<td>9833</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9.9</td>
<td>7</td>
</tr>
<tr>
<td>7992</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.6</td>
<td></td>
</tr>
<tr>
<td>7077</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9.5</td>
<td>4</td>
</tr>
<tr>
<td>7500</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>8120</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>4216</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2169</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>8.5</td>
<td>2</td>
</tr>
<tr>
<td>0990</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>1423</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.7</td>
<td></td>
</tr>
<tr>
<td>1599</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.7</td>
<td>3</td>
</tr>
<tr>
<td>1898</td>
<td>9.5</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.6</td>
<td>5</td>
</tr>
<tr>
<td>8661</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>0993</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>0546</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.0</td>
<td>5</td>
</tr>
<tr>
<td>9919</td>
<td>9.5</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.6</td>
<td>6</td>
</tr>
<tr>
<td>7388</td>
<td>9.5</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.4</td>
<td>5</td>
</tr>
<tr>
<td>8557</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>9.7</td>
<td>6</td>
</tr>
<tr>
<td>9565</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>9.2</td>
<td>5</td>
</tr>
<tr>
<td>7978</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>9.5</td>
<td>2</td>
</tr>
<tr>
<td>7996</td>
<td>9.5</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9.8</td>
<td>6</td>
</tr>
<tr>
<td>0392</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4063</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9.4</td>
<td></td>
</tr>
<tr>
<td>4210</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>2923</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9.4</td>
<td></td>
</tr>
<tr>
<td>8383</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7887</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>8.9</td>
<td>3</td>
</tr>
<tr>
<td>Yan</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Note:**
1. Every Problem values 10 points;
2. The final grade is the average round result;
3. The late homework minus 1 point.
4. Solutions attached in the following pages.
Problem 1. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is 5, 10, 3, 12, 5, 50 and 6.

Answer: The m-table and s-table are given as follows.

According to s-table shown above, the optimal parenthesization is \((A_1A_2)((A_3A_4)(A_5A_6))\).

Problem 2: Show that a full parenthesization of an n-element expression has exactly n-1 pairs of parentheses.

Hint: The easiest and most intuitive method to prove is by induction.

Problem 3: Consider a variant of the matrix-chain multiplication problem in which the goal is to parenthesize the sequence of matrices so as to maximize, rather than minimize, the number of scalar multiplications. Does this problem exhibit optimal substructure? Explain.

Answer: Yes. It does exhibit optimal substructure.

Note: To get full credit for this problem, a reasonable explanation must be given.

Problem 4: As stated, in dynamic programming we first solve the subproblems and then choose which of them to use in an optimal solution to the problem. Professor Capulet claims that it is not always necessary to solve all the subproblems in order to find an optimal solution. She suggests that an optimal solution to the matrix-chain multiplication problem can be found by always choosing the matrix \(A_k\) at which to split the subproduct \(A_iA_{i+1} \ldots A_j\) (by selecting \(k\) to minimize the quantity \(p_{i+1}p_kp_j\)) before solving the subproblems. Find an instance of the matrix-chain multiplication problem for which this greedy approach yields a suboptimal solution.
Hints:
1. The question is to find an instance that Prof. Capulet’s algorithm does NOT yield an optimal solution;
2. Prof. Capulet’s algorithm does give an optimal solution for the instance given in Problem 1, although the s-table is a little bit different;
3. Any instance that Prof. Capulet’s algorithm gives a sub-optimal solution is fine. The following is an example sequence:

\[ P = \{5, 50, 7, 6, 5, 10, 6, 11, 8\} \]

**Problem 5:** Determine an LCS of \(<1, 0, 0, 1, 0, 1, 0, 1>\) and \(<0, 1, 0, 1, 0, 1, 1, 0>\).

**Answer:** There are several of them:

1. \{1, 0, 0, 1, 1, 0\}
2. \{1, 0, 1, 0, 1, 1\}
3. \{1, 0, 1, 1, 0, 1\}

Any one of them is fine.

**Problem 6.** Give an \(O(n^2)\) time algorithm to find the longest monotonically increasing subsequence of a sequence of \(n\) numbers.

**Answer:** There are basically three different algorithms for this problem.

**Algorithm 1:** First, we will try to answer this question by following the steps of dynamic programming.

Suppose the sequence of \(n\) numbers is \(S(A_1, A_2, \ldots, A_n)\). Let us give the following notations first:

- **LMIS[i, j]:** The longest monotonically increasing subsequence found for the subsequence \(A_i, \ldots, A_j\);
- **max[i, j]:** The largest number in LMIS[i, j];
- **min[i, j]:** The smallest number in LMIS[i, j];
- **LEN[i, j]:** The length of LMIS[i, j].

Now we need to see if we can compute LMIS[i, j] recursively by using the optimal solutions of subproblems. First, we have the recursive definition of LMIS[i, j] as follows:

\[
LMIS[i, j] = \begin{cases} 
\{A_i\} & \text{iff } i = j; \\
\{A_i\} \cup_a LMIS[i+1, j] & \text{iff } A_i < \min[i+1,j]; \\
LMIS[i, j-1] \cup_a \{A_j\} & \text{iff } \max[i,j-1] < A_j; 
\end{cases} \quad (6.1)
\]
It can be shown that $\text{LMIS}[i, j-1]$ will be the same as $\text{LMIS}[i+1, j]$ iff $A_i < \min[i+1, j]$ and $\max[i, j-1] < A_j$. Please also note that the operation $\cup_o$ means the joint operation between two sequences; in the operation the second sequence will be appended right after the last number of the first sequence. This operation can be done in constant time by using a linked list.

Now we can have an algorithm that is quite similar to Matrix-Chain-Order. Note that the formula (6.1) can also be given by using the length of LMIS, but the algorithm will be the same.

**LMIS(S)**
1. $n \leftarrow \text{length}[S]$
2. for $i \leftarrow 1$ to $n$
   3. do $\text{LMIS}[i, i] \leftarrow \{A_i\}$;
   4. $\max[i, i] \leftarrow A_i$;
   5. $\min[i, i] \leftarrow A_i$;
6. for $l \leftarrow 2$ to $n$
   7. do for $i \leftarrow 1$ to $n - l + 1$
      8. do $j \leftarrow i + l - 1$
         9. if $A_i < \min[i+1, j]$
             then $\text{LMIS}[i, j] \leftarrow \{A_i\} \cup_o \text{LMIS}[i+1, j]$
             11. $\min[i, j] \leftarrow A_i$
             12. $\max[i, j] \leftarrow \max[i+1, j]$
         13. elseif $\max[i, j-1] < A_j$
             then $\text{LMIS}[i, j] \leftarrow \text{LMIS}[i, j-1] \cup_o \{A_j\}$
             15. $\min[i, j] \leftarrow \min[i, j-1]$
             16. $\max[i, j] \leftarrow A_j$
         17. else $\text{LMIS}[i, j] \leftarrow \text{LMIS}[i, j-1]$
             18. $\min[i, j] \leftarrow \min[i, j-1]$
             19. $\max[i, j] \leftarrow \max[i, j-1]$
20. return $\text{LMIS}[1, n]$

**Analysis:** There are two for loops nested in the above algorithm, so the time complexity will be $O(n^2)$. This algorithm can be further refined to reduce its space complexity.

**Algorithm 2: This algorithm consists of two steps:**

1) Copy the sequence $S$ into $S1$, using heapsort (with time bound of $O(n\log(n))$) or quicksort (with time bound $O(n^2)$) to sort $S1$ and make it in order.

Time complexity: $O(n^2)$ if you are using quicksort.
2). Find the LCS of S and S1.
Time complexity: \( O(n \times n) = O(n^2) \).

So the total time is \( O(n^2) + O(n^2) = O(n^2) \).

**Algorithm 3:**

\texttt{LMIS(S)}
\begin{enumerate}
\item int b[1..n]
\item int pre[1..n]
\item b[1] = 1
\item for(int i=2; i <= n; i++) do
\item \hspace{1em} b[i] = 1
\item \hspace{1em} for(int j=1; j < i; j++) do
\item \hspace{2em} if(S[j] < S[i] && b[j] >= b[i])
\item \hspace{3em} then b[i] = b[j] + 1;
\item \hspace{3em} pre[i] = j
\item return b[1..n], pre[1..n]
\end{enumerate}

The running time: \( O(n^2) \).

**Note:** You need to have enough analysis and explanation to get full credit.