1. Merging of Two Sorted Lists, (10+10 Points)

Solve parts (a) and (b) of problem 8.6 in page 181 of the CLR textbook.

REMARKS: You may find Stirling’s approximation (p.55) for the factorial function useful. Note that to merge a sorted list of \( m \) keys with a sorted list of \( n \) items, the standard algorithm takes \( m + n - 1 \) comparisons. Parts (c) and (d) are not graded, but we ask you to attempt it (we will sketch the solution).

SOLUTION:

(a) Let the input lists be \( A = (a_1, a_2, \ldots, a_n) \) and \( B = (b_1, \ldots, b_n) \). There are \((2n)!\) ways to permute all the elements in \( A \) and \( B \). Let \( S \) be the set of these permutations. E.g., for \( n = 2 \), and the two lists are \( A = (a, a') \) and \( B = (b, b') \) then we have \( S = \{ aa'bb', aa'b'b, ab'a'b, b'a'a'b, \ldots, b'ba'\} \), and \(|S| = 4! = 24\). Now partition \( S \) into \( m = (\binom{2n}{n}) \) subsets, \( S = S_1 \cup S_2 \cup \cdots \cup S_m \), using the following rule: two permutations are put in the same set if they can be obtained from each other by permuting the \( a \)'s among themselves, and permuting the \( b \)'s among themselves. For instance, \( S_1 \) can be \{aa'bb', aa'b'b, a'abb', a'ab'b\}. Clearly, \(|S_i| = (n!)^2\) for each \( i = 1, \ldots, m \). Note the following: each set \( S_i \) has exactly one permutation which is consistent with the original input lists. Hence there are exactly \((2n)!/(n!)^2 = \binom{2n}{n}\) permutations consistent with the input ordering on \( A \) and \( B \).

(b) Let \( h \) be the height of the decision tree. By the information theoretic lower bound, \( h \geq \log \left( \frac{2n}{\sqrt{2\pi n}} \right) \). Using Stirling’s approximation (equation (3.19) of Text), we have

\[
\left( \frac{n}{e} \right)^n \sqrt{2\pi n} < n! < 1.09 \left( \frac{n}{e} \right)^n \sqrt{2\pi n}
\]

for all \( n \geq 1 \), using the fact that \( e^{1/12} = 1.0869 \ldots \). Thus

\[
\left( \frac{2n}{n} \right) > \left( \frac{\sqrt{2\pi 2n(2n/e)^{2n}}}{2\pi n(n/e)^{2n}(1.09)^2} \right)^{2n} > \frac{2^{2n}}{\sqrt{2\pi n}}
\]
Taking logs, we get \( h \geq 2n - \lg(2\pi n)/2 = 2n - o(n) \). Thus \( h \) is the worst case number of comparisons needed to merge two lists of size \( n \).

2. **LCS and Edit Distances (10 Points each part)**

Let \( X = agacgttcgtta \) and \( Y = cgactgtgta \).

- **(i)** Compute \( L(X, Y) \), which is the length of any longest common subsequence of \( X, Y \). Be sure to show a matrix containing your computation.

- **(ii)** Using the matrix you made in part (i), draw arrows from each square to indicate where each entry was derived from. Next, use these arrows to come up with a longest common subsequence of \( X \) and \( Y \).

- **(iii)** Compute the edit distance \( D(X, Y) \), which was defined in recitation to be the minimum number of Insert/Delete/Replace steps to transform \( X \) to \( Y \). Again, please show a matrix with your computation.

- **(iv)** Using the matrix you made in part (iii), draw arrows on your grid to indicate where each entry was derived from. Next, using these arrows to come up with a sequence of \( D(X, Y) \) Insert/Delete/Replace instructions to transform \( X \) into \( Y \).

- **(v)** Let \( X \) and \( Y \) be strings and \( A \) the associated matrix containing the computation record for \( L(X, Y) \) (as in part (i)). Describe an algorithm in pseudocode which, given \( (X, Y, A) \), computes a longest common subsequence of \( X \) and \( Y \).

- **(vi)** Solve the analogous problem to part (v), but this time for the edit distance problem \( D(X, Y) \).

**SOLUTION:**

(i)

Given that:

\[
L(X, Y) = \begin{cases} 
0 & \text{if } m = 0 \text{ or } n = 0 \\
1 + L(X', Y') & \text{if } x_m = y_n \\
\max L(X', Y), L(X, Y') & \text{if } x_m \neq y_n 
\end{cases}
\]

We get the following matrix:
(ii) Arrows are as shown in previous matrix. Using these arrows, we get an acceptable longest common subsequence of \textit{gacgtgta}.

(iii) Given that:

\[
D(X,Y) = \begin{cases} 
\max |X|,|Y| & \text{if } m = 0 \text{ or } n = 0 \\
D(X',Y') & \text{if } x_m = y_n \\
1 + \min D(X',Y), D(X,Y'), D(X',Y') & \text{if } x_m \neq y_n 
\end{cases}
\]

We get the following matrix:

(iv) Arrows are as shown in previous matrix. (Note: The diagonal moves with asterisks next to them indicate when \(x_m = y_n\), which is important in generating the instructions.)

Using these arrows, we get the following instructions, which if executed from first to last, with transform \(X\) into \(Y\):

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
\text{c} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{g} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{a} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
\text{c} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{g} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{a} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\end{array}
\]
$X$.delete(10)
$X$.delete(8)
$X$.replace(6, c)
$X$.insert(t, 4, 5)
$X$.replace(1, c)

(Note: I assume that $X$.insert($t$, 4, 5) inserts a $t$ into $X$ between locations 4 and 5.)

(v)

Here is the pseudocode. Note that the “+” symbol used on strings denotes string concatenation (just like in Java).

LCS($X$, $Y$, $A$) {
    \begin{align*}
    i &= |X|; j = |Y|; s = ""; \\
    &\text{while } ((i > 0) || (j > 0)) \{ \\
    &\quad \text{if } (i == 0) \{ \\
    &\quad \quad j --; \\
    &\quad \}\text{ else if } (j == 0) \{ \\
    &\quad \quad i --; \\
    &\quad \}\text{ else if } (x_i == y_j) \{ \\
    &\quad \quad s = x_i + s; \\
    &\quad \quad i --; j --; \\
    &\quad \}\text{ else if } (\max A[i - 1, j], A[i, j - 1] == A[i - 1, j]) \{ \\
    &\quad \quad i --; \\
    &\quad \}\text{ else } \\
    &\quad \quad j --; \\
    &\quad \}
    &\text{return } s; \\
    \end{align*}
}

(vi)

Here is the pseudocode. Note that the “+” symbol used on strings denotes string concatenation, and we assume integers and characters as implicitly converted to string form when concatenated with other string (just like in Java). And in my pseudocode, we assume $s_k$ corresponds to the $k$th instruction of $s$, where each instruction is to be done in order of first to last.

ED($X$, $Y$, $A$) {
3. **Generalized LCS (15+30+10 Points)**

Recall the recurrence relation we showed for \( L(X, Y) \).

(a) Derive and prove the analogous recurrence for \( L(X, Y, Z) \) is the length of any longest common subsequence of three strings \( X, Y, Z \).
(b) Describe how you would organize the computation of \(L(X, Y, Z)\) in a systematic way.

(c) Illustrate your solution to (b) by computing \(L(X, Y, Z)\) where \(X = longest\), \(Y = lengthen\) and \(Z = elongated\).

SOLUTION:

(a) \[
L(X, Y, Z) = \begin{cases} 
0 & \text{if } m = 0 \text{ or } n = 0 \text{ or } p = 0 \\
1 + L(X', Y', Z') & \text{if } x_m = y_n = z_p \\
\max \{L(X, Y, Z), L(X', Y', Z)\} & \text{if } x_m = y_n \neq z_p \\
\max \{L(X, Y, Z), L(X', Y', Z'), L(X', Y', Z')\} & \text{if } x_m = z_p \neq y_n \\
\max \{L(X, Y, Z), L(X', Y', Z), L(X', Y', Z')\} & \text{if } y_n = z_p \neq x_m \\
\max \{L(X, Y', Z'), L(X', Y', Z'), L(X', Y', Z)\} & \text{if } x_m \neq y_n \text{ and } y_n \neq z_p \text{ and } x_m \neq z_p
\end{cases}
\]

The justification for these formulas are:

If \(X\) or \(Y\) or \(Z\) is the empty string, we know \(L(X, Y, Z) = 0\), hence we have a basis.

When \(x_m = y_n = z_p\), we know for \(x_m\) that \(L(X', Y, Z) \leq 1 + L(X', Y', Z')\), and similarly for \(y_n\) and \(z_p\). So discarding any of \(x_m\), \(y_n\), or \(z_p\) won’t give us a better answer.

When \(x_m = y_n \neq z_p\), let \(k = x_m = y_n\), and let \(l = z_p\). We note that \(L(X', Y, Z') \leq 1 + L(X', Y', Z')\), so there’s no point in keeping \(x_m\) if we discard \(y_n\). (And similarly, there’s no point in keeping \(y_n\) if we discard \(x_m\)). So our only concern is if to keep both \(x_m\) and \(y_n\) and discard \(z_p\), or if to keep \(z_p\) and to discard \(x_m\) and \(y_n\). Hence, we get \(\max L(X, Y, Z'), L(X', Y', Z)\) in this case.

The cases for \(x_m = z_p \neq y_n\) and \(y_n = z_p \neq x_m\) are similar to that of \(x_m = y_n \neq z_p\).

When \(x_m \neq y_n\) and \(y_n \neq z_p\) and \(x_m \neq z_p\), we’re deciding which of \(x_m\), \(y_n\), or \(z_p\) should be kept, hence the \(\max L(X, Y', Z'), L(X', Y, Z'), L(X', Y', Z)\) for this case. (Note to reader: If none of \(x_m\), \(y_n\), or \(z_p\) gets kept, note that \(L(X', Y', X') \leq L(X, Y', Z')\), so we’re fine with formula mentioned above...)

(b) LCS(X, Y, Z) {
    \(m = |X|; n = |Y|; p = |Z|\);
    for \((j = 0; j <= n ; j++)\) {
        for \((k = 0; k <= p ; k++)\) {
            \(A[0, j, k] = 0;\)
        }
    }
    for \((i = 0; i <= m ; i++)\) {
for \( k = 0; k <= p \); \( k++ \) \{
    A[i, 0, k] = 0;
}
}

for \( i = 0; i <= m \); \( i++ \) \{
    for \( j = 0; j <= n \); \( j++ \) \{
        A[i, j, 0] = 0;
    }
}

for \( k = 1; k <= p \); \( k++ \) \{
    for \( j = 1; j <= n \); \( j++ \) \{
        for \( i = 1; i <= m \); \( i++ \) \{
            if (((x_i == y_j) \&\&(y_j == z_k))) \{
                A[i, j, k] = 1 + A[i - 1, j - 1, k - 1];
            } else if (((x_i == y_j) \&\&(x_i == z_k))) \{
                A[i, j, k] = max(A[i, j, k - 1], A[i - 1, j - 1, k]);
            } else if (((x_i == z_k) \&\&(x_i == y_j))) \{
                A[i, j, k] = max(A[i, j - 1, k], A[i - 1, j, k - 1]);
            } else if (((y_j == z_k) \&\&(x_i == y_j))) \{
                A[i, j, k] = max(A[i - 1, j, k], A[i, j - 1, k - 1]);
            } else \{
                A[i, j, k] =
                max(A[i, j - 1, k - 1], A[i - 1, j, k - 1], A[i - 1, j - 1, k]);
            }
        }
    }
}

return A;

(c)

We get the following tables:
\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
z_0 &=& 0 & 0 & 0 & 0 & 0 \\
\hline
l &=& 1 & 0 & 0 & 0 & 0 \\
\hline
e &=& 2 & 0 & 0 & 0 & 0 \\
\hline
n &=& 3 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
g &=& 4 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
t &=& 5 & 0 & 0 & 0 & 0 \\
\hline
e &=& 7 & 0 & 0 & 0 & 0 \\
\hline
n &=& 8 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{1,1,1}
&= & 0 & 1 & 2 & 3 & 4 \\
\hline
l &| o & n & g & e & s & t \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
z_1 &=& e & 0 & 0 & 0 & 0 \\
\hline
l &=& 1 & 0 & 0 & 0 & 0 \\
\hline
e &=& 2 & 0 & 0 & 0 & 0 \\
\hline
n &=& 3 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
g &=& 4 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
t &=& 5 & 0 & 0 & 0 & 0 \\
\hline
e &=& 7 & 0 & 0 & 0 & 0 \\
\hline
n &=& 8 & 0 & 0 & 0 & 0 \\
\hline
\rowcolor[rgb]{1,1,1}
&= & 0 & 1 & 2 & 3 & 4 \\
\hline
l &| o & n & g & e & s & t \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
z_2 &=& l & 0 & 0 & 0 & 0 \\
\hline
l &=& 1 & 0 & 1 & 1 & 1 \\
\hline
e &=& 2 & 0 & 1 & 1 & 1 \\
\hline
n &=& 3 & 0 & 1 & 1 & 1 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
g &=& 4 & 0 & 1 & 1 & 1 \\
\hline
\rowcolor[rgb]{0.8,0.8,0.8}
t &=& 5 & 0 & 1 & 1 & 1 \\
\hline
e &=& 7 & 0 & 1 & 1 & 1 \\
\hline
n &=& 8 & 0 & 1 & 1 & 1 \\
\hline
\rowcolor[rgb]{1,1,1}
&= & 0 & 1 & 2 & 3 & 4 \\
\hline
l &| o & n & g & e & s & t \\
\hline
\end{array}
\]
\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
e & 2 & 0 & 1 & 1 & 1 & 1 & 1 \\
n & 3 & 0 & 1 & 1 & 1 & 1 & 1 \\
g & 4 & 0 & 1 & 1 & 1 & 1 & 1 \\
t & 5 & 0 & 1 & 1 & 1 & 1 & 1 \\
h & 6 & 0 & 1 & 1 & 1 & 1 & 1 \\
e & 7 & 0 & 1 & 1 & 1 & 1 & 1 \\
n & 8 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\text{z}_3 = 0

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
l & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
e & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
g & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
t & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
h & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
e & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
\end{array}
\]

\text{z}_4 = n

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
l & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
e & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
g & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
t & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
h & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
e & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
\end{array}
\]

\text{z}_5 = g

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
l & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
e & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
g & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
t & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
h & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
e & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
n & 0 & 1 & 1 & 2 & 2 & 2 & 2 \\
\end{array}
\]
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>g</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>z_6 = a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>h</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>t o n g e s t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>g</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>z_7 = t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>h</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>t o n g e s t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>g</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>z_8 = e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>h</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>n</td>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>t o n g e s t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. **Matrix Chain Product (15 Points)**

Solve the Matrix Chain Product problem for a matrix chain with 8 matrices, with dimension:

$$(2, 4, 3, 7, 1, 9, 8, 5)$$

Please organize your work in a half-matrix as illustrated in Figure 15.3 (p.337, CLR).

When you are done, please draw the actual parenthesis structure for evaluating this chain.

**SOLUTION:**

![Matrix Chain Product Solution](image)

Note that the matrix entry $m_{ij}$ is computed as

$$
\min_{i \leq k < j} (m_{ik} + m_{k+1} + r(i) \cdot c(k) \cdot c(j))
$$
where \( r(i) \) denotes the number of rows in matrix \( i \) and \( c(j) \) denotes the number of columns in matrix \( j \). Thus, the actual parenthesis structure is

\[
(M_1 \times ((M_2 \times (M_3 \times (M_4 \times M_5)))) \times ((M_6 \times M_7) \times M_8)).
\]

5. **Treaps (10 + 50 Points)**

A treap is a binary search tree combined with a heap structure. Read the description of treaps on pages 296-297 in the CLR textbook.

- (i) Construct a Treap out of the following (key, priority) pairs:

  (12, 2)(15, 6)(5, 1)(7, 3)(2, 4)(17, 5)

- (ii) Do parts (a), (b), (c), and (d) of problem 13-4 on pages 296-298 in the CLR textbook.

**Hints:**

- For part (b), it might help to look at section 12.4, pages 265 - 268 in the CLR book. Alternatively, you can try the following:
  Let \( H(n) \) denote the height of a binary tree with \( n \) nodes. This is a random variable (so you can take expectation, etc). If \( i \) is the number of children in the left subtree of the root, then:
  \[
  H(n) = 1 + \max_i H(i) + H(n - 1 - i). \]
  Hence,
  \[
  E[H(n)] = 1 + \frac{1}{n} \sum_{i=0}^{n-1} \max[E[H(i)], E[H(n - 1 - i)]].
  \]
  Then use real induction to prove that for some constants \( 0 < c_1 \leq c_2 \) that \( c_1 \log(n) \leq E[H(n)] \leq c_2 \log(n) \).

- For part (c), first give the English description, followed by pseudocode. If necessary, add comments to your pseudocode.

**SOLUTION:**

(i)

The node (5, 1) should be the root, with (2, 4) and (12, 2) as the left/right children of (5, 1). The nodes (7, 3) and (17, 5) should be the left/right children of (12, 2). And (15, 6) should be the left child of (17, 5).

(ii)

(a)

Let’s suppose that \( k_1 \) is the heap key for node \( x_1 \), and \( k_2 \) is heap key for node \( x_2 \), and so on... I’ll prove that for any \( n \), a treap of \( n \) nodes must be unique. I will do this by Integer Induction.

**Basis:**
If we just have a treap of one node, that treap is unique.

Induction Hypothesis:

Assume for all $k \leq n - 1$ that a treap for $k$ nodes must be unique.

Induction Step:

For the treap with $n$ nodes, consider the node $x_i$ with smallest heap key $k_i$. This node $x_i$ must always be at the root of the treap (due to minheap properties). We can then use $x_i$ to partition the remaining nodes into two sets $L$ and $R$, where:

$L = \{x_j : x_j < x_i\}$, and $R = \{x_j : x_j > x_i\}$

We know by treap property that $L$ consists of all the nodes in the left subtree of $x_i$, and $R$ consists of all the nodes in the right subtree of $x_i$.

And, by our induction hypothesis, since $|L| \leq n - 1$ and $|R| \leq n - 1$, the left and right subtrees of $x_i$ must be unique, and thus the entire treap of $n$ nodes must be unique.

(b)

By varying the heap keys for each node, we are able to obtain binary trees of any design. So, the expected height of a treap is the expected height of a randomly generated binary tree. Let $E[H(n)]$ denote the expected height of a binary tree with $n$ nodes.

First, we’ll show that $c_1 \lg n \leq E[H(n)]$ for some $c_1$. We note that when we create a complete binary tree (much like that in the answer for homework 3), the height is $\lfloor \lg n \rfloor$. And, for $n \geq 4$, we have that:

$\lfloor \lg n \rfloor \geq \lg n - 1 \geq \frac{1}{2} \lg n$

Since in the best case, the height of a binary tree is at least $\frac{1}{2} \lg n$, then we know $c_1 \lg n \leq E[H(n)]$ where $c_1 \geq \frac{1}{2}$.

Next, we’ll prove that $E[H(n)] \leq c_2 \lg n$ for some $c_2$.

Using our hint, this means:

$$E[H(n)] = 1 + \frac{1}{n} \sum_{i=0}^{n-1} \max E[H(i)], E[H(n - 1 - i)].$$

Substituting $F(n) = E[H(n)]$, we have:

$$F(n) = 1 + \frac{1}{n} \sum_{i=0}^{n-1} \max F(i), F(n - 1 - i).$$

Next, I will use this to show for all $n \geq 1$ that $F(n) \leq c_2 \lg n$ for some constant $c_2$. I’ll prove this by Induction. (I’m assuming $F(1) = 0$ to make things simpler...)
Basis: $F(1) = 0 \leq c_2 \cdot \lg 1 = 0$

Induction Hypothesis:
For all $k \leq n - 1$, we assume that $F(k) \leq c_2 \cdot \lg k$.

Induction Step:
(Please note that students weren’t responsible for accounting for floors and ceilings in their answers.)

Since $F(i)$ is non-decreasing, we have that:

$$F(n) = 1 + \frac{1}{n} \sum_{i=0}^{n-1} \max F(i), F(n-1-i)$$

$$= 1 + \frac{2}{n} \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n-1} F(i)$$

$$\leq 1 + \frac{2}{n} \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n-1} c_2 \cdot \lg \iota$$

$$= 1 + \frac{2c_2}{n} \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n-1} \lg i$$

From here, by splitting the summation into two parts, we get:

$$1 + \frac{2c_2}{n} \sum_{i=\lfloor \frac{n}{2} \rfloor}^{\lfloor \frac{3n}{4} \rfloor} \lg i \leq 1 + \frac{2c_2}{n} \left[ \sum_{i=\lfloor \frac{n}{2} \rfloor}^{\lfloor \frac{3n}{4} \rfloor} \lg i + \sum_{i=\lfloor \frac{3n}{4} \rfloor}^{n-1} \lg i \right]$$

$$\leq 1 + \frac{2c_2}{n} \left[ \left( \left\lfloor \frac{3n}{4} \right\rfloor - 1 - \left\lfloor \frac{n}{2} \right\rfloor \right) \lg \left( \frac{3n}{4} \right) + \left( n - 1 - \left\lfloor \frac{3n}{4} \right\rfloor \right) \lg n \right]$$

$$\leq 1 + \frac{2c_2}{n} \left[ \left( \frac{n}{4} + 1 \right) \lg \left( \frac{3n}{4} \right) + \left( \frac{n}{4} + 1 \right) \lg n \right]$$

$$= 1 + \frac{2c_2}{n} \left( \frac{n}{4} + 1 \right) \left( \lg \frac{3}{4} + \lg n \right)$$

$$= 1 + \frac{2c_2}{n} \left( \frac{n}{4} + 1 \right) \left( 2 \lg n + \lg \frac{3}{4} \right)$$

$$= 1 + \frac{2c_2}{n} \left( \frac{n}{2} \lg n + 2 \lg n + \frac{n}{4} \lg \frac{3}{4} + \lg \frac{3}{4} \right)$$

$$\leq 1 + \frac{2c_2}{n} \left( \frac{n}{2} \lg n + 2 \lg n + \frac{n}{4} \lg \frac{3}{4} \right)$$

$$= 1 + c_2 \lg n + \frac{4c_2}{n} \lg n + \frac{c_2}{2} \lg \frac{3}{4}$$

And for $n \geq 256$, $\frac{\lg n}{n} \leq \frac{8}{256} = \frac{1}{32}$, therefore:
\[ 1 + c_2 \log n + \frac{4c_2}{n} \log n + \frac{c_2}{2} \log \frac{3}{4} \leq 1 + c_2 \log n + 4c_2 \frac{1}{32} + \frac{c_2}{2} \log \frac{3}{4} \]
\[ \leq 1 + c_2 \log n + \frac{c_2}{8} \log \frac{3}{4} \]

And \( 1 + c_2 \log n + \frac{c_2}{8} \log \frac{3}{4} \leq c_2 \log n \) when:

\[ 1 + c_2 \log n + \frac{1}{8} c_2 + \frac{c_2}{2} \log \frac{3}{4} \leq c_2 \log n \]
\[ 1 + \frac{1}{8} c_2 + \frac{c_2}{2} \log \frac{3}{4} \leq 0 \]
\[ 1 + c_2 \left( \frac{1}{8} + \frac{1}{2} \log \frac{3}{4} \right) \leq 0 \]
\[ 1 \leq -c_2 \left( \frac{1}{8} + \frac{1}{2} \log \frac{3}{4} \right) \]
\[ \frac{1}{\left( \frac{1}{8} + \frac{1}{2} \log \frac{3}{4} \right)} \leq c_2 \]

And, \( \frac{1}{\left( \frac{1}{8} + \frac{1}{2} \log \frac{3}{4} \right)} \leq 12.2 \), so we select \( c_2 \geq 12.2 \) and we’re done!

(Note that \( -\left( \frac{1}{8} + \frac{1}{2} \log \frac{3}{4} \right) > 0 \))

An alternative solution is as follows. If we do Real Induction, which allows us to ignore floors and ceilings, the Basis and Inductive Hypothesis are identical (where \( \delta = 1 \)), and we get as before,

\[ F(n) = 1 + \frac{2c_2}{n} \sum_{i=\frac{n}{2}}^{n-1} \log i \]

And using Integral Approximation from page 1067 of the CLR book,

\[ 1 + \frac{2c_2}{n} \sum_{i=\frac{n}{2}}^{n-1} \log i \leq 1 + \frac{2c_2}{n} \int_{\frac{n}{2}}^{n} \frac{\ln x}{\ln 2} \, dx \]
\[ = 1 + \frac{2c_2}{n \ln 2} \int_{\frac{n}{2}}^{n} \ln x \, dx \]
\[ = 1 + \frac{2c_2}{n \ln 2} [x \ln x - x]_{x=\frac{n}{2}}^{n} \]
\[ = 1 + \frac{2c_2}{n \ln 2} \left[ x \ln x - \frac{x}{\ln 2} \right]_{x=\frac{n}{2}}^{n} \]
\[ = 1 + \frac{2c_2}{n \ln 2} \left[ n \ln n + \frac{n}{2} \left( 1 - \frac{1}{\ln 2} \right) \right] \]
\[ = 1 + c_2 \log n + c_2 \left( 1 - \frac{1}{\ln 2} \right) \]
And, \(1 + c_2 \log n + c_2(1 - \frac{1}{\ln 2}) \leq c_2 \log n\) when:

\[
1 + c_2(1 - \frac{1}{\ln 2}) \leq 0 \quad \Rightarrow \quad 1 \leq c_2(\frac{1}{\ln 2} - 1) \\
\frac{1}{\ln 2 - 1} \leq c_2
\]

(Note that \(\frac{1}{\ln 2} - 1 > 0\). This might not be obvious from what I did...) And \(\frac{1}{\ln 2} - 1 \leq 2.3\), so we select \(c_2\) so that \(c_2 \geq 2.3\).

(c) To insert a node \(x\), we assign it some random value \(k\) which to be its heap key. We then place \((x, k)\) into the treap based on the value of \(x\) by regular BST insertion. Then, we rotate \((x, k)\) with its parent until \((x, k)\) is at a position where \(k\) is larger than the heap key of the \(l\), the heap key of the parent of \((x, k)\).

The rotations assure that \((x, k)\) remains in left/right order by the tree-key with respect to the other nodes.

Psuedocode is:

\[
\text{TREAP\_INSERT}(x) \{
  k = \text{random\_number};
  \text{BST\_INSERT}(x, k);
  \text{while } (((x, k).\text{parent} \neq \text{NULL}) \&\& (k < (x, k).\text{parent\_heap\_key})) \{
    \text{rotate}((x, k));
  \}
\}
\]

(d) Let’s suppose that \(h\) is the height of the treap. On average, doing BST\_INSERT takes time linearly proportional to \(h\).

And for randomly chosen \(k\), the number of rotations done on \((x, k)\) afterwards is also linearly proportional to \(h\).

Therefore on average, we take time \(\Theta(h)\).

And from part (b), we’ve proven the expected \(h = \log n\). Thus, expected average time is \(\Theta(\log n)\).