7.4.5 Modify the CYK algorithm to report the number of distinct parse trees for the given input, rather than just reporting membership in the language.

The Algorithm:

Associate an integer with every element of every $X_{ij}$ that appears in the CYK algorithm’s table (integers are initialized to 0). Modify the CYK algorithm’s basis to set all of the integers in the first row to 1. Modify the induction step to include the following: For every production of the form $P \rightarrow QR$ found by the CYK algorithm for $X_{ij}$, let $p$ be the current integer associated with $P$ in $X_{ij}$, let $q$ be the integer associated with $Q$ in $X_{ik}$, and let $r$ be the integer associated with $R$ in $X_{k+1,j}$, then update $p$ to be $p + q \cdot r$

If the start symbol (S) is not in $X_{1n}$ where $n$ is the length of the input string then return 0. Otherwise return the integer associated with the start symbol in $X_{1n}$

A bit of notation/terminology:

Let $N_{ij}$ represent non-terminal $N$ in $X_{ij}$. The subscript $ij$ is often referred to as a span. For example, in the table below $B_{24}$ spans $aaba$. Also let $I(N_{ij})$ represent the integer associated with $N$ in $X_{ij}$. Now we can read $I(N_{ij})$ as the number of distinct parse trees spanning $ij$ with $N$ as the start symbol.

An Example (based on 7.34 on p. 301 of the course textbook):

$$
\begin{align*}
S & \rightarrow \quad AB \mid BC \\
A & \rightarrow \quad BA \mid a \\
B & \rightarrow \quad CC \mid b \\
C & \rightarrow \quad AB \mid a \\
\end{align*}
$$

Table created by the modified CYK algorithm:

<table>
<thead>
<tr>
<th>$S:2$, $A:2$, $C:1$</th>
<th>$S:2$, $A:2$, $C:1$</th>
<th>$B:1$</th>
<th>$B:1$</th>
<th>$S:1$, $A:1$</th>
<th>$S:1$, $A:1$</th>
<th>$S:1$, $C:1$</th>
<th>$B:1$</th>
<th>$B:1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
</tr>
</tbody>
</table>

Example 1: $A_{25}$

In $X_{25}$, the CYK algorithm finds two productions that span $aaba$ with $A$ as the start symbol: $A_{25} \rightarrow B_{23} A_{45}$ and $A_{25} \rightarrow B_{24} A_{55}$. Our addition to the induction step correctly computes that $I(A_{25}) = I(B_{23}) \cdot I(A_{45}) + I(B_{24}) \cdot I(A_{55}) = 1 \cdot 1 + 1 \cdot 1 = 2$

Example 2: $A_{15}$

In $X_{15}$, the CYK algorithm finds one production that spans $baaba$ with $A$ as the start symbol: $A_{15} \rightarrow B_{11} A_{25}$, so $I(A_{15}) = I(B_{11}) \cdot I(A_{25}) = 1 \cdot 2 = 2$
Example 3: $S_{15}$

In $X_{15}$, the CYK algorithm finds two productions that span $baaba$ with $S$ as the start symbol: $S_{15} \rightarrow B_{11} C_{25}$ and $S_{15} \rightarrow A_{12} B_{35}$ so $I(S_{15}) = I(B_{11}) \cdot I(C_{25}) + I(A_{12}) \cdot I(B_{35}) = 1 \cdot 1 + 1 \cdot 1 = 2$

Additional Fun (the following is just Brain candy, and you DO NOT need to know if for this class):

Fun fact 1:

The modified CYK algorithm is known as the *inside count* algorithm in the field of Natural Language Processing (NLP). It is one of the basic algorithms used to analyze sentences, which can be highly ambiguous (there can be millions of parses for one sentence). If interested you can take CS 474 or CS 324.

Corollary to Fun fact 1:

Yes, there is an *outside count* algorithm as well (=)

Fun fact 2:

In general, the number of parse trees grows exponentially in the length of the input string. (compare the exponents in the table below)

<table>
<thead>
<tr>
<th>Length of Input String</th>
<th># of distinct parse trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^2$ ($baaa$)</td>
<td>2</td>
</tr>
<tr>
<td>$2^3$ ($abaabab$)</td>
<td>31</td>
</tr>
<tr>
<td>$2^4$ ($ababaababaabaa$)</td>
<td>2826</td>
</tr>
<tr>
<td>$2^5$ ($baaabaaababaababaabababaababaab$)</td>
<td>869846184</td>
</tr>
<tr>
<td>$2^6$ ($... ababababababab ...$)</td>
<td>$1.295 \times 10^{19}$</td>
</tr>
</tbody>
</table>

Fun fact 3:

Instead of dealing with an exponential number of parse trees, people in NLP (and probably other fields) work with a single parse *forest*, which shares the common subcomponents of the parse trees. The size of a parse forest is $O(n^3)$, where $n$ is the length of the input string. Note the similarity between the size of a parse forest and the running time of the CYK algorithm.

See the next page for graphical representations of parse forests for $baaba$ using $S$, $A$, and $C$ as the start symbols. The little numbers in boxes represent links for structure sharing. Ignore the boxes with x’s in them. Try comparing the parse forests to the CYK table above. What do the curly braces correspond to in our computation of $I$?