1. Let \( T = \text{lallilla}\$ \).

   (a) Give the suffix tree of \( T \) including suffix links.
   
   (b) Give the suffix array of \( T \) together with the LCP array.

Solution:

Suffix tree:

[Diagram of suffix tree]

Suffix and LCP array:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( SA[i] )</th>
<th>( LCP[i] )</th>
<th>( T_{SA[i]} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>\text{a}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>\text{a}$lilla$</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>\text{illa}$</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0</td>
<td>\text{la}$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
<td>\text{lilla}$</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>\text{illa}$</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>1</td>
<td>\text{illa}$</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>2</td>
<td>\text{lilla}$</td>
</tr>
</tbody>
</table>

2. The reverse of a string \( S[0..m] \) is the string \( S^R = S[m-1]\text{S}[m-2]..S[0] \). Describe an algorithm for finding the longest factor \( S \) of \( T[0..n] \) such that the reverse \( S^R \) is a factor of \( T \) too. The algorithm should work in linear time in the constant alphabet model.

Solution: \( S \) and \( S^R \) are both factors of \( T \) if and only if \( S \) is a factor of both \( T \) and \( T^R \). Therefore, \( S \) is the longest common factor of \( T \) and \( T^R \).

The longest common factor of \( T \) and \( T^R \) can be computed using the generalized suffix tree (see slide 185 in the lectures). Therefore, the problem is reduced to the computation of a suffix tree for a string \( TLT^R\$ \) which can be done in linear time using for example McCreight’s algorithm.
3. What is the number of distinct factors in the string abracadabra?

**Solution:**

The suffixes in lexicographic order are

```plaintext
a
abra
abracadabra
acadabra
adabra
bra
bracadabra
cadabra
dabra
ra
racadabra
```

The LCP array is

\[ \text{LCP} = (1, 4, 1, 0, 3, 0, 0, 2) \]

and the sum is 12. Thus the number of distinct factors is \( 11 \cdot \frac{12}{2} + 1 - 12 = 55 \).

4. Give a linear time algorithm for computing the matching statistics of \( S \) with respect to \( T \) from the generalized suffix array of \( S \) and \( T \) and the associated LCP array (without constructing the suffix tree).

**Solution:**

Let \( SA \) be the suffix array of \( T \$S \$ \) and \( LCP \) be the associated LCP array. \( SA[i] \) represents a \( T \)-suffix \( T_{SA[i]} \) if \( SA[i] \leq |T| \) and an \( S \)-suffix \( S_{SA[i]−|T|−1} \) otherwise.

Let \( i < j \) be such that \( SA[i] \) and \( SA[j] \) are \( T \)-suffixes and, for all \( k \in (i..j) \), \( SA[k] \) is an \( S \)-suffix.

Consider some \( k \in (i..j) \) and let \( k' = SA[k] − |T| − 1 \), i.e., \( S_{k'} \) is the suffix at \( SA[k] \). Let

\[
\ell_1 = \text{lcp}(T_{SA[i]}, S_{k'}) = \min\{LCP[h] \mid h \in [i+1..k]\}
\]

\[
\ell_2 = \text{lcp}(T_{SA[j]}, S_{k'}) = \min\{LCP[h] \mid h \in [k+1..j]\}
\]

where we used Lemma 4.9 in the lecture notes. Then

\[
MS[k'] = \begin{cases} 
(\ell_1, SA[i]) & \text{if } \ell_1 \geq \ell_2 \\
(\ell_2, SA[j]) & \text{otherwise}
\end{cases}
\]

We can compute \( \ell_1 \) for all \( k \in (i..j) \) in \( O(j - i) \) time with a single left-to-right scan of \( LCP[i+1..j] \) and similarly \( \ell_2 \) with a single right-to-left scan of \( LCP[i+1..j] \). Thus the total time complexity is \( O(|S| + |T|) \).
5. Let \( L = rtttrar\$ii \) be the Burrows–Wheeler transform of a text \( T \).

(a) What is \( T \)?

Solution: \( T = ritaritar\$. \)

(b) Simulate backward search on \( T \) for the pattern \( P = ari \).

Solution: The following table shows the computation in the main loop of Algorithm 4.15:

<table>
<thead>
<tr>
<th>( i )</th>
<th>old ( b )</th>
<th>old ( e )</th>
<th>( c \leftarrow P[i] )</th>
<th>( C[c] )</th>
<th>( \text{rank}(c, b) )</th>
<th>( \text{rank}(c, e) )</th>
<th>new ( b )</th>
<th>new ( e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>10</td>
<td>i</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>r</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>8</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The output is the range \([2..3)\).