1. Counts of Substrings

Assume all strings are over a constant-sized alphabet and are indexed starting at 0.

(a) Let $x$ be a string of length $n$. There are $O(n^2)$ substrings of $x$, but some of them may have the same sequence of characters, of course. Show how to count the number of distinct substrings of $x$ in $O(n)$ time.

**Solution:** Build the suffix tree for $x$. Take any substring that occurs in $x$. If you start from the root of the suffix tree and follow edges down the tree you will always remain inside the tree. When you stop you will be either at a node in the suffix tree, or you will be part of the way along an edge. (Recall that an edge represents a string of one or more characters.)

The number of places that you can “stop” like this is precisely the number of different substrings of $x$. So it’s just the sum of the lengths of all the edges in the suffix tree, plus 1 for the empty string.

(b) In a string $S$, a maximal repeat occurrence is a pair of positions $i$ and $j$ in $S$ and a length $k$ such that:

- $S[i, \ldots, i + k - 1] = S[j, \ldots, j + k - 1]$,
- $S[i - 1] \neq S[j - 1]$,
- $S[i + k] \neq S[j + k]$

Above, we take $S[-1]$ and $S[n]$ to be symbols distinct from every other symbol. A maximal repeat string is a string $w$ such that there is a maximal repeat occurrence $(i, j)$ with $S[i, \ldots, i + |w| - 1] = w$.

(i) Give an example of a string in which two different maximal repeat strings both start at the same position. In other words, find a string with two maximal repeat occurrences $(i, j, k)$ and $(i, j', k')$.

**Solution:** Consider the following string of length 10:

```
<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
<th>b</th>
<th>x</th>
<th>c</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<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>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
```

$(1, 5, 1)$ is a maximal repeat occurrence, and $(1, 7, 2)$ is also.

(ii) Using suffix trees, give a short, elegant proof that there are $\leq n$ distinct maximal repeat strings.

**Solution:** Consider taking a maximal repeat string $w$ and walking down the path it defines in the suffix tree. The place that it ends must have at least two continuations: One for each of its occurrences. Therefore it must end at an internal node of the
suffix tree. A suffix tree for the string \( s[0] \ldots s[n] \) has \( n \) suffixes, and at most \( n - 1 \) internal nodes. So there are at most \( n - 1 \) maximal repeat strings. Note that this does not even use the fact that the characters prior to the two starts of a maximal repeat string must differ.

2. Farthest Apart

You’re given an unrooted tree of \( n \) nodes. Some of the nodes are marked. Let \( \text{mark}(x) \) be a boolean valued function which is true if node \( x \) is marked and false otherwise. At least two nodes are marked.

(a) Give an algorithm that runs in time \( O(n) \) to compute the maximum distance between a pair of marked nodes. The distance is just the number of edges in the tree on the path between the two nodes.

**Solution:** Use “Tree DP”. At each node \( x \) in the tree we recursively compute and return two numbers: deepest, and farthest-pair.

deepest\((x)\) = the maximum distance between \( x \) and a marked node in the subtree rooted at \( x \).

furthest-pair\((x)\) = the maximum distance between a marked pair of nodes in the subtree rooted at \( x \).

deepest\((x)\) is just one plus the value of the deepest() values of all of its children.

furthest-pair\((x)\) is computed by taking the maximum of all the furthest-pair() values of all the children of \( x \). And also considering the longest path that goes through \( x \). This is just \( 2 + d_1 + d_2 \), where \( d_1 \) is the largest deepest() value among its children, and \( d_2 \) is the second largest such value. It also has to consider the possibility that \( x \) is marked. (The special cases of \( x \) having no children, or just one child also need to be considered.)

(b) Explain how to modify the algorithm in part (a) to find the pair that are farthest apart.

**Solution:** The answer can be reconstructed if we just compute, at every vertex \( x \), of the deepest node and the numbers of the pair that’s farthest apart in the subtree rooted at \( x \).