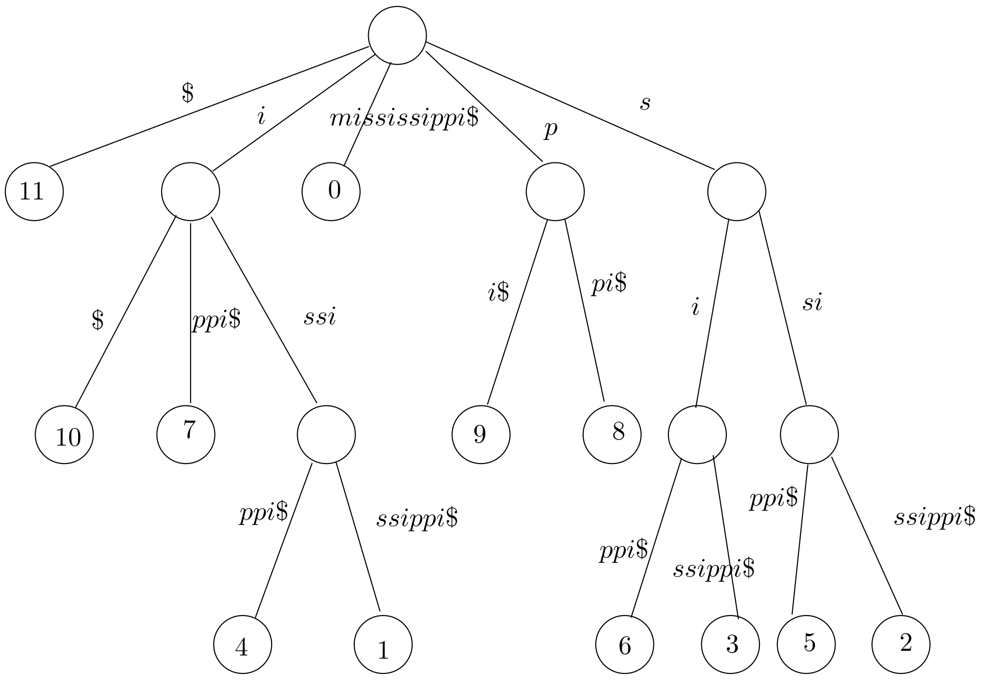
Dalhousie University
CSCI 6057/4117 — Advanced Data Structures
Winter 2022
Assignment 4 Solutions

Questions:

1. [15 marks] Draw the suffix tree for the string `mississippi`. Append a \$ (the end of file symbol) to the end of the string when drawing the suffix tree.

   The following is the figure for this suffix tree. In this figure, we assume that in the string T, the indices are $0, 1, \ldots$. It is OK if you assume the indices are $1, 2, \ldots$ instead, but then labels on the leaves would be different.



2. [10 marks] In class, we learned how to construct nearly optimal binary search trees in linear time. The running time is given by the following recursion

$$T(n) = O(\lg \min\{i, n-i+1\}) + T(i-1) + T(n-i)$$

Prove by induction that $T(n) = O(n)$.

Hint: Pages 85-86 of the CLRS book might help.

Solutions: Rewrite the iteration to

$$T(n) \leq c \lg \min\{i, n-i+1\} + T(i-1) + T(n-i) \leq c \lg \min\{i, n-i+1\} + T(i) + T(n-i+1)$$

where c is an appropriate constant.

We now prove, by induction, that $T(n) \leq dn - c \lg n - d - c$, where d is an appropriate positive constant. Base cases are easy.

We assume that this holds for $n \leq k - 1$. We now prove it for $n = k$. Then

$$\begin{aligned}
T(k) &\leq c \lg \min\{i, n-i+1\} + T(i) + T(n-i+1) \\
&\leq c \lg \min\{i, n-i+1\} + di - c \lg(i) - d - c + d(n-i+1) - c \lg(n-i+1) - d - c \\
&= dn - d + c \lg \min\{i, n-i+1\} - c \lg(i) - c \lg(n-i+1) - 2c
\end{aligned}$$

Assume without loss of generality that $i \leq n-i+1$. Then $n-i+1 \geq (n+1)/2 \geq n/2$. Then

$$\begin{aligned}
T(k) &\leq dn - d - c \lg(n-i+1) - 2c \\
&\leq dn - d - c \lg(n/2) - 2c \\
&\leq dn - d - c \lg n + c - 2c \\
&\leq dn - c \lg n - d - c
\end{aligned}$$

3. [10 marks] Let A be a string of length m over a constant-size alphabet, and B be a string of length n over the same alphabet. We wish to find the longest common (contiguous) substring of A and B, i.e., the longest string that appears as a (contiguous) substring of both A and B. Design an algorithm to solve this problem in $O(m + n)$ time. Show your analysis of the running time. You are not required to give pseudocode, but feel free to give pseudocode if it helps you explain your algorithm.

   Hint: It may be helpful to construct a suffix tree. However, it is unlikely that a suffix tree built over $A\$$ or $B\$$ will help you achieve the desired running time. Think about what else you could do.

   Solution: Let $T = A\#B\$$, where # and \$ are two different end-of-file (or terminal) symbols. Construct a suffix tree for T. We then look for the deepest branching vertex (in terms of "character depth" as described in class) in the suffix tree with one of its descendant leaves corresponding to a suffix that contains # and another descendant leaf corresponding to a suffix that does not contain #. This can be found using a depth-first search.

The suffix tree can be constructed in time linear in $|T|$, and thus it can be constructed in $O(m+n)$ time. As the suffix tree has $O(m+n)$ nodes, a depth-first search requires $O(m+n)$ time. Therefore, the total running time is $O(m+n)$.

4. [15 marks] In class, we learned the succinct data structures that can support the `rank` queries over a bit vector of length n. Among the set of structures constructed, one of them is a look-up table F. This table stores, for each possible bit vector of length $\frac{\lg n}{2}$ and each position in it, the answer to `rank`.

   Now, we construct a different table E. Table E has one entry for each possible bit vector of length $\frac{\lg n}{2}$, and it stores the number of 1s in this bit vector.

   For simplicity, we assume that $\frac{\lg n}{2}$ is an integer.

   (i) [7 marks] Show that E occupies $O(\sqrt{n}\lg\lg n)$ bits.

   Solution: As there are $2^{\frac{\lg n}{2}} = \sqrt{n}$ different bit vectors of length $\frac{\lg n}{2}$, table E has $\sqrt{n}$ entries. Each entry encodes a nonnegative integer that is at most $\frac{\lg n}{2}$, so each entry can be stored in $O(\lg\lg n)$ bits. Therefore, the total number of bits occupied by E is $O(\sqrt{n}\lg\lg n)$.

   (ii) [8 marks] In the set of data structures constructed to support `rank`, we replace table F by table E. Show how to use the resulting set of data structures to support `rank` in constant time. You are allowed to use bit operations.

   Show your analysis of the running time. You are not required to give pseudocode, but feel free to give pseudocode if it helps you explain your algorithm.

   Solution: Since we have all the other data structures except F, to support `rank`, it suffices to show, given a block, B, of $\frac{\lg n}{2}$ bits and an arbitrary index, i, of B, how to compute the number of 1 bits in $B[1..i]$ in constant time. To do this, we first right shift B by $\frac{\lg n}{2} - i$ bits and store the result in a bit vector, B', of $\frac{\lg n}{2}$ bits. we then use E to find out how many 1 bits are there in B'. Since B' contains the first i bits of B prefixed by 0's, the number of 1's in B' is the answer.

   Since both B and B' fit in a machine word, shift and table look up use $O(1)$ time, and thus the total running time is $O(1)$.