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

Distributed Wednesday, March 9 2022.
Due at 23:59 Wednesday, March 23 2022.

Guidelines:

1. All assignments must be done individually. The solutions that you hand in must be your own work.
2. Submit a PDF file with your assignment solutions via Brightspace. We encourage you to typeset your solutions using LaTeX. However, you are free to use other software or submit scanned handwritten assignments as long as **they are legible**.
3. Working on the assignments of this course is a learning process and some questions are expected to be challenging. Start working on assignments early.
4. Whenever you are asked to prove something, give sufficient details in your proof. When it is straightforward to prove a lemma/property/observation, simply say so, but do not claim something to be straightforward when it is not.
5. We have the following late policy for course work: `http://web.cs.dal.ca/~mhe/csci6057/assignments.htm`

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.

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.

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.

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.

   (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.