BIL 212:Summer 2021

Assignment 4

Assigned on July 26. You should upload your program to uzak.etu.edu.trr by Friday, August 6, 23.59. Remember that your code should be fully documented. I once again remind you to read the academic honesty policy stated in the syllabus.

Overview:

In this assignment you will use a tree structure to store DNA sequences in a way that allows for an efficient search. Not only can we determine whether a specified sequence is in the tree, but we can also find any sequence that matches a sequence prefix.

We define DNA sequences to be strings on the alphabet A, C, G, and T. We will define a new tree data structure to store DNA sequences, which we will call a DNA tree. DNA trees store sequences only in their leaf nodes. Internal nodes serve only as placeholders to help direct search, they store no data. Some leaf nodes may be empty.

An internal node is a 5-way branching node, with a branch for each possible letter. In addition to the letters A, C, G, and T, we must augment the alphabet for DNA sequences to contain $ to indicate the termination of a sequence. Thus, the five branches correspond to the five letters of the augmented DNA alphabet: A, C, G, T, and $. When traversing through the tree structure to perform an operation, the first branch from an internal node corresponds to the letter A, the second branch to the letter C, and so on, with the fifth branch corresponding to $. Thus, all sequences stored that begin with A will be in the first subtree, all sequences stored that begin with C will be in the second subtree, and so on.

A leaf node is either empty, or stores a single sequence. When you want to insert a new sequence and the insert process reaches a leaf node containing a sequence, that leaf node must be converted to an internal node with 5 children. This operation is called a split.

Whenever you remove a sequence (contained in a leaf node), if possible that node will merge with its siblings. This case will happen if all siblings of the removed leaf node are empty leaf nodes.

If there are no sequences stored in the tree, the tree consists of a single empty leaf node. If there is one sequence stored in the tree, the tree consists of a single leaf node containing the sequence.

See Figure 1 (on the last page of this document) for an example DNA Tree.

Your program will read from an input directivesFile. The program will be invoked from the command line as: `java DNADatabase <directivesFile>`
DO NOT hardcode filenames within your Java code. The input directivesFile consists of a sequence of directives, each indicating an operation to be performed. **There will be one directive per line.** Each directive consists of the name of the directive followed by 0 or more arguments, separated by spaces. You may assume that only the directives given below will appear in the file, and the specified arguments will always appear. You may also assume that the sequence given is a valid DNA sequence. The output of the directives will be written to standard output.

Below are the possible directives and their meanings.

- `insert sequence`

  Insert `sequence` into the DNA tree. Print a message indicating if the insertion was successful, and if so, indicate the level of the leaf node inserted. It is an error to insert a duplicate sequence. Such an error should be reported in the output, and no changes to the tree structure should take place.

- `remove sequence`

  Remove sequence from the DNA tree if it exists. Print a suitable message if sequence is not in the tree.

- `display`

  Display the DNA tree. You should perform a preorder traversal of the tree, and print each node on a separate line in the order that it is visited by the traversal. If the node is internal, just print the letter I. If the node is an empty leaf node, just print the letter E. If the leaf node contains a sequence, print the sequence. Each node should be printed such that it is preceded by "depth (of the node in the tree" many dots. That is, the root node is printed with no dots, immediate children of the root are preceded by 1 dot, grandchildren of the root are preceded by 2 dots, and so on.

- `display-lengths` Output is identical to that of the display command, except that the length of the sequence is printed after the sequence for all sequences stored in the tree.

- `search sequenceDescriptor` Find all sequences that match sequenceDescriptor. The sequenceDescriptor can come in two forms. The first form is simply as a sequence containing letters from the alphabet A, C, G, and T. If this form is given, then print all sequences stored in the tree for which sequenceDescriptor is a prefix (including exact matches). The second form is a sequence from the letters A, C, G, and T, followed by a $ symbol. If this form is given, then only an exact match of the sequence (without the $ symbol) is to be printed. Print the number of nodes visited in the tree during the search.

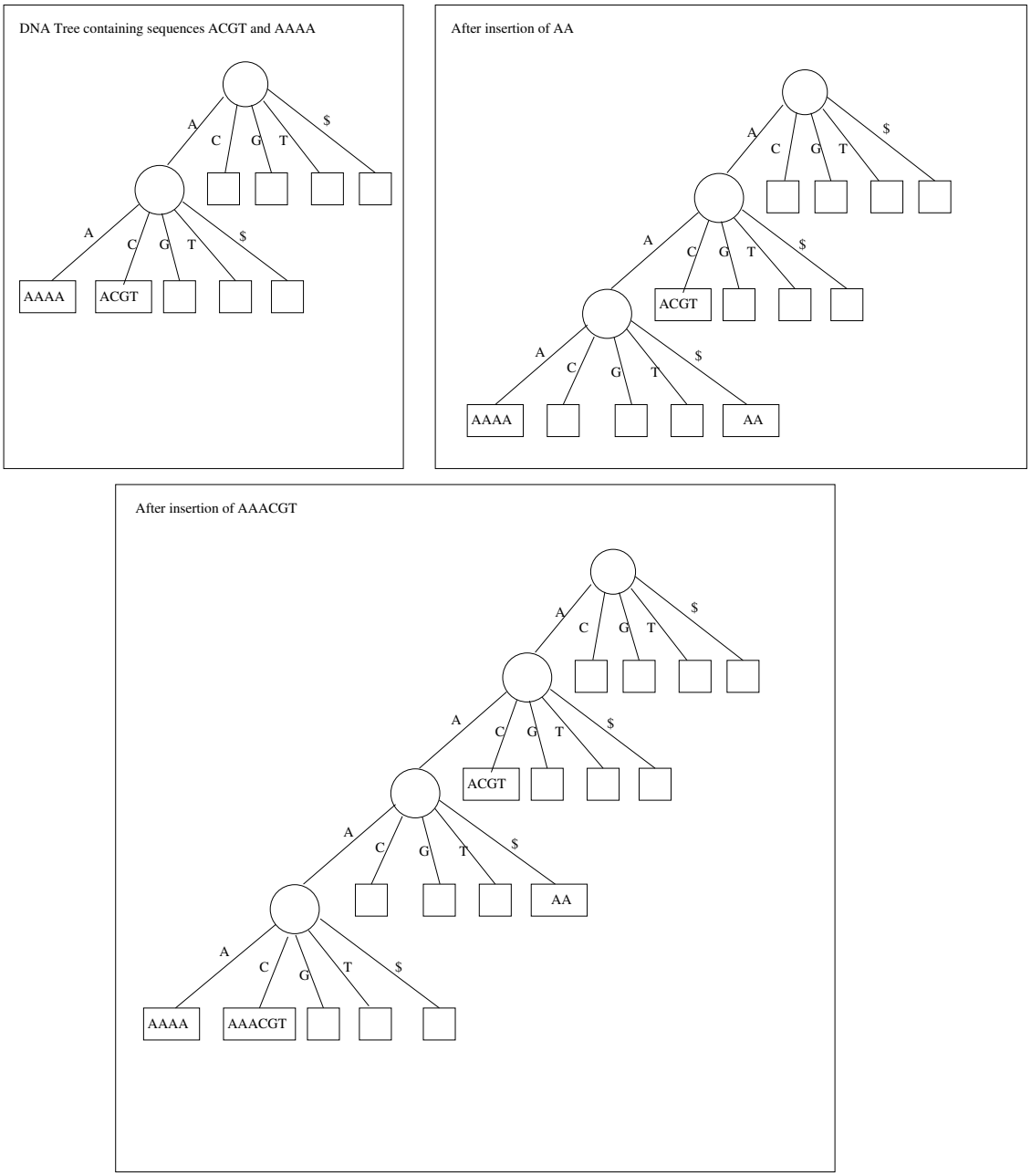

Figure 1: Example DNA Tree