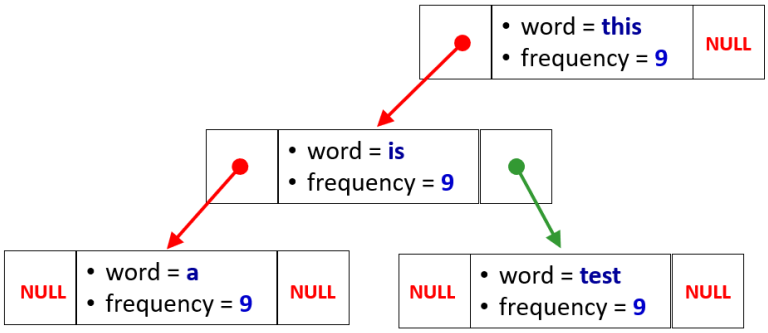
CPSC 319 – Winter 2024 Programming Assignment 3 **Binary Search Trees**	**Released:**	February 27 (Tuesday)
	Due:	March 15 (Friday) @ 11:59 PM
	# days to complete:	18 days

GOAL

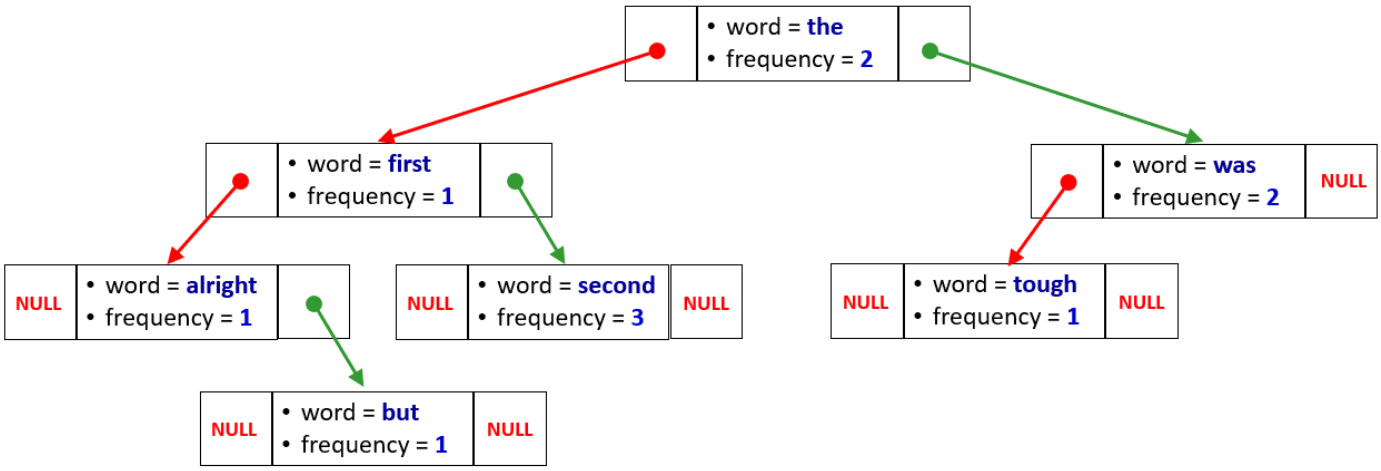
Trees are invaluable data structures in software. In this assignment, you will write the code that will create a binary search tree (BST) of unique words from a given input text file, and keep track of the frequency of the words in the text. For example:

Example 01: Input text file "example01.txt"	**Resulting BST**
this is a test this is a test this is a test this is a test this is a test this is a test this is a test this is a test this is a test	

Example 02 – Input text file "example02.txt"

the first second was alright but the second second was tough

Resulting BST



The nodes from the BST store the word itself and its frequency in the input text file. Using this structure for a tree, it is possible to find if a word occurs in the text in $O(\log_2 n)$ (i.e., if the tree is of minimum height) as well as the number of times the word occurs in the text.

Specifications for reading and processing the input text file:

- Words are delimited with spaces.

INSTRUCTIONS

Write the Java program based on the skeleton code provided (as in PA-1 and PA-2) that will do the following:

1. **Read the input text file** (available at the assignment's D2L page):
   This should be a given ASCII text files "example01" and "example02"

2. **Use the words from the input files to create the BST specified in the previous page.** You will obviously have to ignore spaces as specified above.

3. **When the tree has been created, it should supply the following information as a display output:**

(3.1) **The total number of words in the file.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal., counting the number of times you visit each node in the BST.

(3.2) **The number of unique words in the file.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal., counting the number of nodes visited containing the word with frequency = 1

(3.3) **The word which occurs most often and the number of times that it occurs.**
Use either IN-ORDER, PRE-ORDER, or POST-ORDER traversal., looking for the word(s) with the largest frequency.

Example (display screen for question #3) for example01 using POST-ORDER traversal (i.e., left, right, node):

> Total number of words in example01 = 4

> Number of unique words in example01 = 0

> The word(s) which occur(s) most often and the number of times that it/they occur(s) =
a = 9 times
test = 9 times
is = 9 times
this = 9 times

Example (display screen for question #3) for example02 using POST-ORDER traversal (i.e., left, right, node):

> Total number of words in example02 = 7

> Number of unique words in example02 = 4

> The word(s) which occur(s) most often and the number of times that it/they occur(s) = second = 3 times

3. The user should also be able to request to display the entire tree using any of the 3 traversal methods.

The result should be each word in the tree separated by a single space.

Example (display screen for question #5) for example01:

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 1
> IN-ORDER output: a is test this

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 2
> PRE-ORDER output: this is a test

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example01 ? 3
> POST-ORDER output: a test is this

Example (display screen for question #5) for example02:

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 1
> IN-ORDER output: alright but first second the tough was

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 2
> PRE-ORDER output: the first alright but second was tough

> Enter the BST traversal method (1 = IN-ORDER, 2 = PRE-ORDER, 3 = POST-ORDER) for example02 ? 3
> POST-ORDER output: but alright second first tough was the

HAND-IN

1. Cover/Grade sheet with your name and ID number only.
2. Your code.
3. Include all of the above items in a zipped folder titled (PA-3-your LAST NAME-ID).zip and submit this folder electronically to D2L dropbox with subject asgmt3-your LAST NAME-ID

LATE ASSIGNMENTS WILL NOT BE ACCEPTED

MARKING

- Source code that does not compile or produces run-time errors will receive a grade of 0%.

COLLABORATION

- The assignment must be done **individually** so you must write up the solutions *on your own* in *your own words*.
- Everything that you hand in must be your original work, except for the code copied from the textbook, lecture material (i.e., slides, notes), web, or that supplied by your TA. When someone else's code is used like this, you **must** acknowledge the source explicitly, citing the sources in a scientific way (i.e., including author(s), title, page numbers, URLs, etc.)
- Copying another student's work constitutes academic misconduct, a very serious offense that will be dealt with rigorously in all cases. Please read the sections of the University Calendar under the heading "Student Misconduct". If you are in doubt whether a certain form of aid is allowed, ask your instructor!
- Contact your TA if you have problems getting your code to work.
- Note that your code may be checked thoroughly for plagiarism by computer.

END OF THE ASSIGNMENT