### Words List: An Application of Binary Search Trees

### A major course output

### for the course on

### Data Structures and Algorithms

### (CCDSALG)

### Submitted by

### Villarica, Matthew James D.

### Joanna Pauline Rivera

### Teacher

### September 16, 2020

### Introduction

### /\*In paragraph form, introduce the task that is assigned and the motivation behind it.\*/

### The task assigned is to design and implement an algorithm for a program that takes a text file as input and creates a text file as output that contains the word count of each word. The program must have the non-linear data structure binary search tree as a major part of its implementation. The required binary search tree operations must be implemented correctly, and at least 1 binary search tree has to be used to store the words from the text file. Exhaustive testing must be done. The purpose of this task is to practice and gain a greater understanding of non-linear data structures, specifically binary search trees and implement them practically.

### Design and Implementation /\*Indicate the programming language that you chose. Give a brief description of how you implemented the data structure. Did you implement a single BST or multiple BST? Explain why. Give a brief description of your algorithm. You may include a flowchart for visualization if necessary. Describe how you implemented your algorithm. Depending on the programming language used, list the libraries or APIs that you used in your implementation. Indicate how to compile (if it is a complied language) your code, and how to run (execute) your program from the COMMAND LINE. Examples are shown below. Replace them accordingly. Make sure that all your group members test what you indicate below. The solution will be initially tested using the sample input text file that you have submitted. Then, another text file provided by the instructor will be used to test the data.\*/

### The Java programming language was used because of the ease of the string manipulation required to read from the text file and the built in API, such as the buffered writer and buffered reader which was used for reading text input from the text file as strings. 2 binary search trees were used to store the valid words that were scanned from the text file. One binary search tree named *allWords* contains every valid word including its duplicates. That is, if there are 14 instances of the word “the” in the text file, the binary search tree contains 14 copies of that string. The convention is to insert a string that is equal to the string in the current node to the right child during an insert operation. This binary search tree is used to determine the actual count or the number of instances of that word in the text file, with a recursive method that returns the integer count of a given string. Another binary search tree called *uniqueWords* also contains a copy of each scanned word in the text file, but it only contains 1 instance of each word without the duplicates. This binary search tree is used to do the in-order walk to print the words into the output text file and their respective counts in *allWords.*

### PSUEDO CODE:

### While there are still lines of text to be read from INPUT.txt

### Read the next line as a string with the buffered reader.

### Make a string array that splits each word if a space (“ “) Is present between them.

### Remove the unnecessary characters like “!”, “.”, “?” from every word in the string array.

### Remove the small words that have a length that is less than or equal to 3

|  |
| --- |
| Sample:Libraries:<Library 1>To compile from the command line (for compiled language only):C:\CCDSALG>gcc -Wall GROUP-1.c -o GROUP-1.exeTo run from the command line:C:\CCDSALG>GROUP-1 |

1. **Results and Analysis**  
   Discuss your test results.

### Discuss the strengths of your program. What inputs can it handle? (e.g. your solution can handle English words with hyphen.) Explain or hypothesize how it can handle such inputs.

### 

### Discuss the weaknesses of your program. What inputs can it not handle (e.g. your solution cannot handle English words with hyphen). Explain or hypothesize why it cannot handle such inputs.

### Conclusions and Recommendations

### Summarize what you did. An algorithm that … was implemented using …

### Highlight the significant and interesting findings.

### References

### Cite all resources and references. Follow the APA format.

### Appendix A: Contribution of Members

|  |  |
| --- | --- |
| Name | Contributions |
|  |  |
|  |  |
|  |  |