Zülal Karın 12622989076 **1/04/2022**
Ahmet Berkay Sarıaslan 54490639066 **Section 02**

CMPE 343 HW-5 REPORT

Problem Statement and Code Design

In this assignment, we are asked to compare the time and space complexity of data structures. To do this, we first need to create data structures one by one from the inputs in the txt files. For this, we have created a special class for each data structure. We used these classes to create data structures and to respond to queries. In the question, we use two queries to compare the performance of data structures. The first query is to find how many words have the given prefix in the input file. The second query is to find the total number of each word in the text file. To do this, we used the methods in the main and the classes in which we created the data structures. I have put a Structure Chart below for you to better understand the steps we followed.

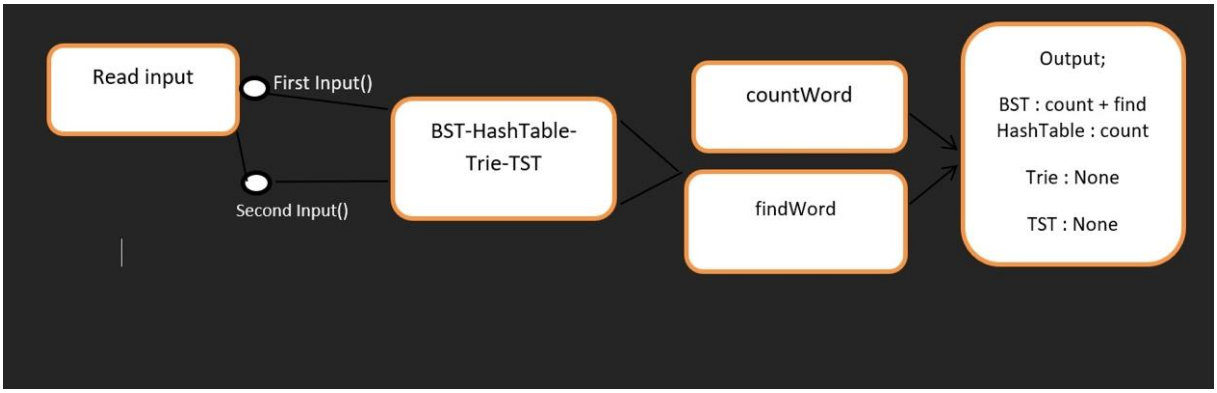

figure 1

Implementation and Functionality

For this task, mainly I created BST, HashTable, Trie, TST and Main classes. I will describe these classes and the methods within them using the Structure Diagram in figure1.

1)Read Input: We needed to read the input from the txt file. For this, we got help from java.io.File class and String methods in Main class.

2) Create the data structure: We need 4 data structure for the task,. These are: BST, Hash Table, R-way trie and TST. We created classes for each of these. We used these classes in main to create data structures. Using the constructor of each class, we created objects as in figure-2, and then added the values we read from the file to the data structures sequentially, as seen in figure-3.

```
private final BST bst = new BST();
private final TST tst = new TST();
private final HashTable ht = new HashTable();
private final Trie trie = new Trie(); //R-TRİE
```

figure 2

```
for (String s : data) {
    bst.insert(s);
    tst.insert(s,1);
    ht.put(s);
    trie.put(s);

}
```
figure 3

3)Finding output: We tried using Java's stopwatch class to find the time complexity of each data structure. For this, we started the time before calling the method and stopped the time afterwards. This way we tried to compare the time complexity of all the data structures. You can see it in more detail in Figure-4.

```
Stopwatch watch1 = new Stopwatch();   //I created stopwatch object
main.firstInput();
main.FindWord("con");
double time1 = watch1.elapsedTime();  //elapsedTime() is returns the elapsed time.
System.out.println("elapsedTime time:   " + time1 + " ms.");
```
figure 4

Testing

After reading the inputs that were given to us in this task, respectively, I can say that we got the full result from the "Binary Search Tree". Because it only Decodes by selecting and counting from words without the need for any "key" value. This way we can achieve both results. In addition, we can only get the number values from the "Hash Table" data structure. Because when searching in this data structure, we Dec need "key" values when searching. The reason Dec we get results from "trie" and "TST" is because both the search and counting algorithms directly need a "key" value. As we mentioned earlier, we cannot get any results because the "key" values are not defined for this task.

Using the BST data structure, you can see the output words starting with -con in the input file.

```
condimentum
consectetur
consequat
convallis
```
figure 5

Final Assessments

- The part that we had the most difficulty with in this task was when we decided how to compare data structures.

- The most challenging part of the task is to perform a performance analysis.

- The problem we faced was the concern about whether it directly contains a "key" value. However, we used our estimates and lecture notes for this problem. Thanks to this task we are up to.