Computer Engineering
CSE 222/505 - Spring 2023
Homework # Report

Name: Ahmet Furkan Ekinci
Student Number: 200104004063

1. Introduction

Purpose of Document

This document is report for this project. The report is going to explain you what stages the homework was going through.

Purpose of Project

The purpose of this project is to be able to perform certain operations through java's own tree class and to implement some search algorithms on the tree.

2. Problem Solution Approach

1) Constructor and readFromFile

Since the use of arraylist is not free at the beginning of the assignment, I read the lines from the file one by one by setting the size of the string to dynamic in readFromFile method.

```
File file = new File("tree.txt");
Scanner scanner = new Scanner(file);
```

I read line by line from the file named tree.txt through the scanner object. Since the words in each line are separated by semicolons (";"), I simply separated the words from each other using the method below and added them to the two-dimensional String array.

```
String[] words = line.split(";");
```

When we come to the constructor part, I call the readFromFile method and fill the two-dimensional String directory. Since every node in the tree does not have to hava a child, I am also checking this as some elements of this string may be null. I open two nested for loops and add all the elements in the string to the tree. I use two consecutive DefaultMutableTreeNode variables named parent and

node while doing the insertion process.

```
JTree jtree=new JTree(root);
frame.add(jtree);
frame.setSize(300,600);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

As seen in the picture above, I added my root to it using a JTree and then added my JTree type variable to the JFrame and displayed an output graphically on the screen.

2) breadthFirstSearch

In this method I used queue which is data structure using FIFO(First in first out) structure. I started by putting the root variable in queue and looped a while until queue was empty. While assigning a value to the node variable using the poll method of the queue data structure, if there are children of the node variable, I add them to the tree and the loop continues until the queue is empty.

```
while (!queue.isEmpty())
{
    DefaultMutableTreeNode node = queue.poll();
    System.out.print("STEP " + counter + " -> " + node.getUserObject());
    if(node.getUserObject().equals(str))
    {
        System.out.print(" (FOUND!!)");
        flag++;
        break;
    }
    System.out.println("");
    int childCount = node.getChildCount();
    for (int i = 0; i < childCount; i++)
    {
        DefaultMutableTreeNode childNode = (DefaultMutableTreeNode) node.getChildAt(i);
        queue.add(childNode);
    }
    counter++;
}
if (flag == 0)
{
    System.out.println("");
    System.out.println("Not found!!!");
}
```

3) depthFirstSearch

I used the stack data structure because it uses the LIFO(Last in first out) structure in this method. I start a while loop by adding the root variable to the stack and this loop continues until the stack is empty. I used the pop method to empty the stack.

```
DefaultMutableTreeNode node = stack.pop();
```

If the node variable has children, I opened a for loop inside the while loop and added these children to the stack.

```
int numChildren = node.getChildCount();
for (int i = 0; i < numChildren; i++)
{
    DefaultMutableTreeNode childNode = (DefaultMutableTreeNode) node.getChildAt(i);
    stack.push(childNode);
}
```

4) postOrderTraversal

Since this method performs a search in the opposite order of the previous method, the depthFirstSearch method, I used another stack in addition to the previous method and put what I had stored in the first stack on the second stack and checked whether it exists or not. In short, I benefited from the stack data structure being in a lifo structure.

5) move and remove

If the source and target years are the same, I did not make any changes. I suppressed a warning and terminated the method. I checked whether the given resource is in my tree or not, and. If not, I terminated the method by suppressing the error message. I removed the source, if any, from the tree with the remove method. Then I added my destination year to the first index of the string and added to the tree like this from scratch.

In the remove method, I make sure that the source given to me is in the tree. If any node in the source string has no children then I delete the string at the beginning of the source string and they are all deleted. If it has children then i go to the last node of the source string and delete this node.

```
realNode.remove(findNodeIndex(realNode, words[words.length-1]));
```

I used the remove method of the DefaultMutableTreeNode class while doing the deletion.