COP 5536: Assignment 1

Malvika Ranjitsinh Jadhav
jadhav.m@ufl.edu
UFID:2802-6840

November 14, 2022

1 Problem Statement:

Implement a AVL tree with the following operations:
1. Initialize (): create a new AVL tree
2. Insert (key)
3. Delete (key)
4. Search (key): returns the key if present in the tree else NULL
5. Search (key1, key2): returns keys that are in the range key1 key key2

2 Steps to run the code:

1.Unzip the zip file 2.Specify the input file in command line as below
$java avltree filename
filename gives the input file and generated output is saved to "output_file.txt" (as per the instructions in the assignment).

3 Code Files:

avl.java
avltree.java

4 avl.java

This file contains the code to build the AVL tree and all its methods.

4.1 Node Class

Before defining the AVL tree we start by defining a private class Node that helps create a basic node structure for AVL tree.

```
private class Node {...}
```

4.1.1 Class variables:

Variable name	Data type	Function (Purpose of variable)
data	int	key at a node
left	Node	node to left of current node
right	Node	node to right of current node
height	int	height of current node

4.1.2 Class methods:

```
public Node(int data) {...}
```

Above method is constructor for Node class and sets the values of data and height variables when invoked.

4.2 avl class

Now we will look into the avl class which performs all functions related to avl tree from creation to deletion.

```
private class avl {...}
```

4.2.1 Class Variables:

Variable name	Data type	Function (Purpose of variable)
root	Node	root node of avl tree
result(static variable)	ArrayList<String>	stores result of range search query

4.2.2 Class methods:

1.Important methods

```
private Node Insert(Node node, int item) {...}
```

Method Name	Input Arguments	Return type	Function Scope
Insert	current node, key for insertion	Node	private
Function Description			
1. Creates a new node to insert the given key 2.Checks the position of insertion: if key <node's data : insert in left subtree else insert in right subtree 3.Updates height variable for newly inserted node 4.Calculates balance factor 5.Balances the AVL tree if required after new node is inserted into tree			

```
private Node balanced(int bf, Node node, int d){...}
```

Method Name	Input Arguments	Return type	Function Scope
balanced	balance factor, current node, key being inserted or deleted	Node	private
Function Description			
Checks whether one of the four cases of imbalance has occurred and takes the correcting steps to maintain AVL tree: LL case ; RR case ; LR case ; RL case ;			

```
private Node Delete(Node node, int item){...}
```

Method Name	Input Arguments	Return type	Function Scope
Delete	current node, key being deleted	Node	private
Function Description			
1. Base case: If key is not present function returns null 2.Checks the position of the key to delete the relevant node: if key<current node's data : delete from left subtree else if key>current node's data : delete from right subtree else if node is found deletion is done using one of the three cases below : Case 1: Node to be deleted had no child Case 2 : Node to be deleted has one child Case 3: Node to be deleted has two children 3.Balance tree after deletion if required			

```
public String Search(int item){...}
```

Method Name	Input Arguments	Return type	Function Scope
Search	single key to be searched	String	public
Function Description			
Traverses entire tree to check if input key is present and returns the same back			

```
public List<String> Searchtwo(Node n, int l, int r) {...}
```

Method Name	Input Arguments	Return type	Function Scope
Searchtwo	current node, left limit, right limit	ArrayList<String>	public
Function Description			
1.Perform search between range [l,r] 2.Return ArrayList of values found			

2.Helper methods

Method	Description
private int height(Node N){...}	returns height of input node object when invoked
public void Insert(int item) {...}	It is invoked outside the class. It send an internal call to function private Node Insert(Node node, int item) {...} with root node of avl tree to begin insertion.
int getBalanceFactor(Node n){...}	calculate balance factor of input node
private Node rightRotate(Node c){...}	perform right rotation
Node leftRotate(Node c){...}	perform left rotation
public void Delete(int item) {...}	It is invoked outside avl class. It send an internal call to function private Node Delete(Node node,int item) {...} with root node to perform deletion.
private Node successorNode(Node N){...}	Find inorder successor of input node
public String Search(int l, int r) {...}	It is invoked outside the class and sends an internal call to the Searchtwo method. It takes the output from this method (ArrayList) and invokes a function to get back single (with comma seperated elements) string from this output. This single string is then returned back.
private String makeString(List<String>arr){...}	creates a single string with comma seperated elements from a ArrayList
public void printInorder() {...}	(uninvoked method) For inorder printing of created AVL tree for reference

5 avltree.java

This file contains the object that invokes an object of avl class and performs the required input and output from it as per the specified format.
The overall working of this file is explained in the figure below:

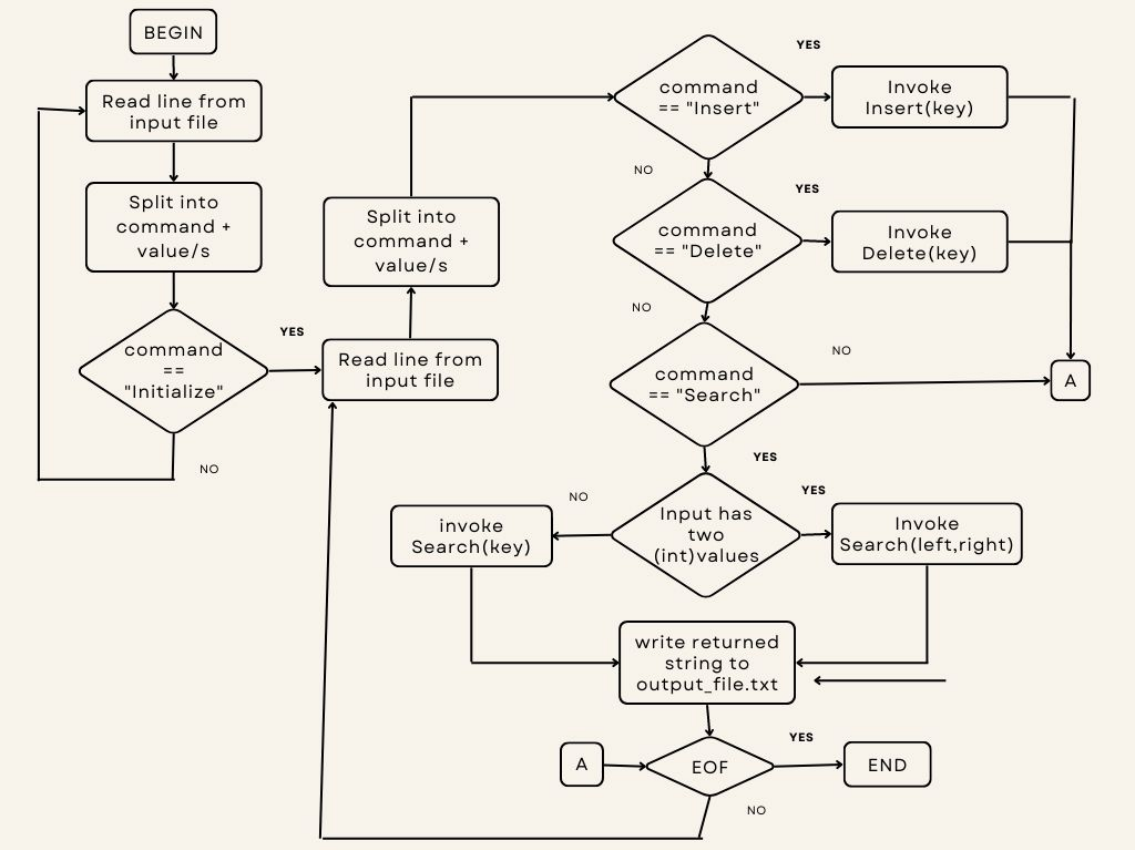

Figure 1: Flow of file avltree.java

6 Output

The output after running the code gets saved in file output_file.txt. Please find attached screenshots obtained after running the code on local machine and thunder server respectively.

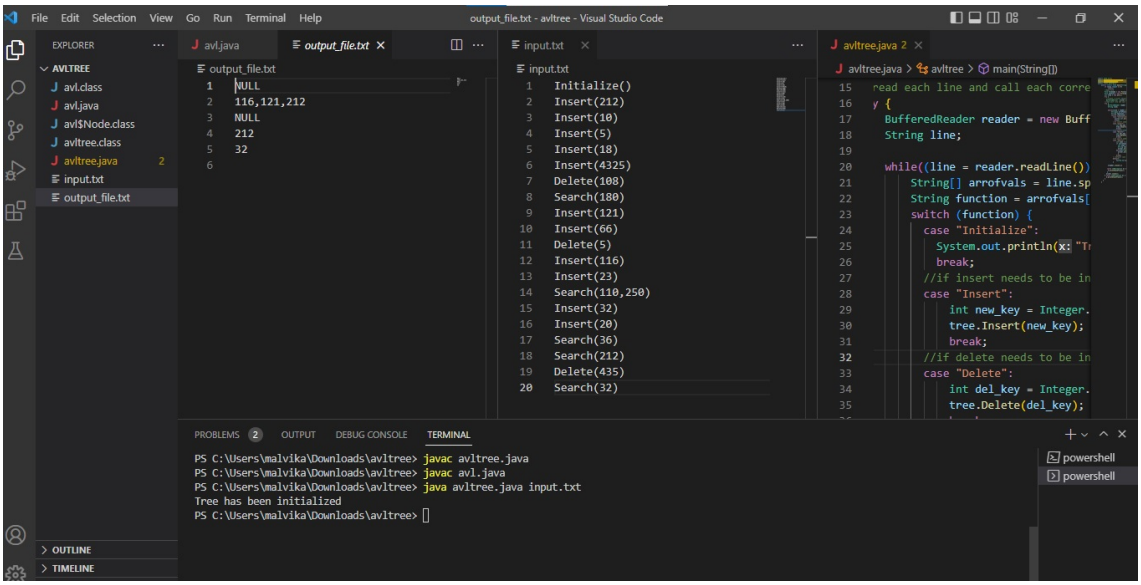

Figure 2: Local machine

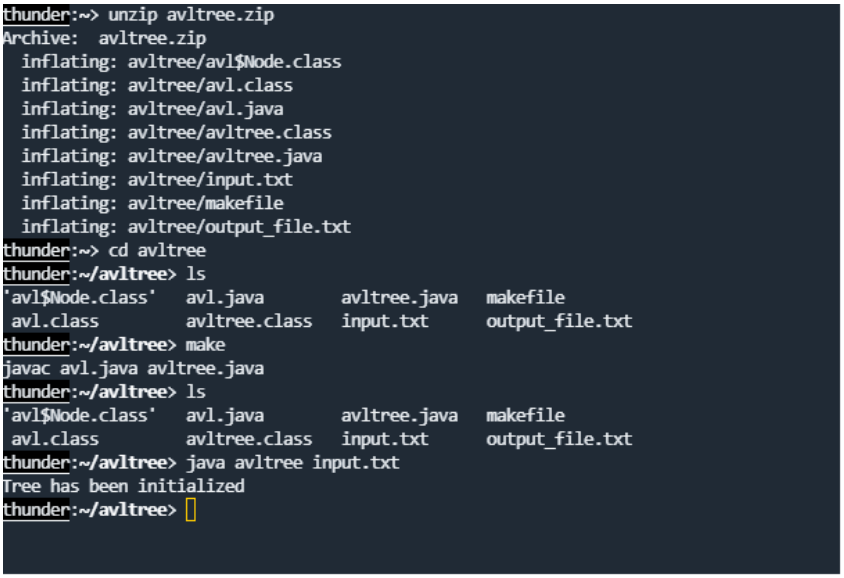

Figure 3: Thunder server