# 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.

#### 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></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
	balance factor,		
balanced	current node,	Node	private
	key being inserted or deleted		
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 data:="" delete="" from="" left="" node's="" subtree<="" td=""></current>			
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			

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

3.Balance tree after deletion if required

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 1, int r) {...}

Method Name	Input Arguments	Return type	Function Scope
	current node,		
Searchtwo	left limit,	ArrayList <string></string>	public
	right limit		
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	
private int height(ivode iv){}	when invoked	
	It is invoked outside the class.	
public void Insert(int item) {}	It send an internal call to function	
public void insert(int item) \(\lambda\)\	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	
	It is invoked outside avl class.	
public void Delete(int item) {}	It send an internal call to function	
public void Delete(Int Item) {}	private Node Delete(Node node,int item) {}	
	with root node to perform deletion.	
private Node successorNode(Node N){}	Find inorder successor of input node	
	It is invoked outside the class and sends	
	an internal call to the Searchtwo method.	
	It takes the output from this method (ArrayList)	
public String Search(int l, int r) {}	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){}</string>	creates a single string with comma	
private String makestring(List <string>arr){}</string>	seperated elements from a ArrayList	
	(uninvoked method)	
public void printInorder() {}	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:

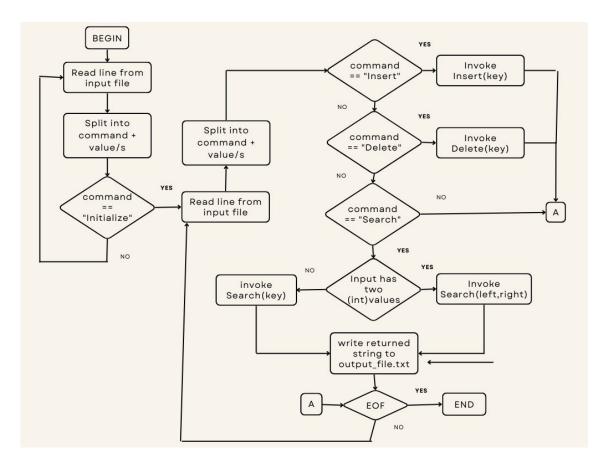


Figure 1: Flow of file avltree.java

## 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.

```
þ
                                                                                ■ output_file.txt

1 | NULL
2 116,121,212
3 NULL
4 212
                                                                                                                                                                                                                                       Initialize()
Insert(212)
Insert(10)
Insert(5)
                                                                                                                                                                                                                                                                                                                                                                              y {
BufferedReader reader = new Buff
                    J avl.java
                                                                                                                                                                                                                                       Insert(18)
Insert(4325)
Delete(108)
                                                                                                                                                                                                                                                                                                                                                                                              String[] arrofvals = line.sp
String function = arrofvals[
switch (function) {
   case "Initialize":
                                                                                                                                                                                                                                       Search(180)
Insert(121)
Insert(66)
Delete(5)

    output file.txt

                                                                                                                                                                                                                                                                                                                                                                                                       System.out.println(x: "Tr
break;
//if insert needs to be in
                                                                                                                                                                                                                                      Delete(5)
Insert(116)
Insert(23)
Search(110,250)
Insert(32)
Insert(20)
Search(36)
Search(212)
Delete(435)
Search(32)
                                                                                                                                                                                                                                                                                                                                                                                                  int new_key = Integer.
int new_key = Integer.
tree.Insert(new_key);
break;
//if delete needs to be in
case "Delete":
                                                                                                                                                                                                                                                                                                                                                                                                               int del_key = Integer
tree.Delete(del_key);
                                                                                                                                                                                                                                       Search(32)
                                                                                 PROBLEMS (2) OUTPUT DEBUG CONSOLE TERMINAL
                                                                               PS C:\Users\malvika\Domnloads\malvree> javac avltree.java
PS C:\Users\malvika\Domnloads\malvree> javac avl.java
PS C:\Users\malvika\Domnloads\malvree> java avltree.java input.txt
Tree has been initialized
PS C:\Users\malvika\Domnloads\malvree> []
                                                                                                                                                                                                                                                                                                                                                                                                                                                      ≥ powershell
≥ powershell
               > OUTLINE
```

Figure 2: Local machine

```
thunder:~> unzip avltree.zip
Archive: avltree.zip
 inflating: avltree/avl$Node.class
 inflating: avltree/avl.class
 inflating: avltree/avl.java
 inflating: avltree/avltree.class
 inflating: avltree/avltree.java
 inflating: avltree/input.txt
 inflating: avltree/makefile
 inflating: avltree/output_file.txt
thunder:~> cd avltree
thunder:~/avltree> ls
avl$Node.class' avl.java
                                               makefile
                                 avltree.java
                                 input.txt
                                                output_file.txt
avl.class
                 avltree.class
thunder:~/avltree> make
javac avl.java avltree.java
thunder:~/avltree> ls
avl$Node.class' avl.java
                                 avltree.java
                                               makefile
avl.class
                avltree.class input.txt
                                                output_file.txt
thunder:~/avltree> java avltree input.txt
Tree has been initialized
thunder:~/avltree>
```

Figure 3: Thunder server