CSC2001F: Assignment 2 report 2021

April 26

Authored by: Kevin Chiloane CHLKEV001

AVL Search Trees

Data structures

The task of the assignment requires a written code to read a text file containing student information(student identity, student name and last name) and store two entries(student fullname and student identity) such as:
CHLKEV001 Kevin Chiloane.

Program

Created application AccessAVLApp to store and retrieve data from text file oklist, the application contains two definite methods namely, printAllStudent which outputs every entry inside data and printStudent which outputs entry by specific key
The program is supported by classes inside file AVLtree.tar.gz, Student and readFile from the course.

AccessAVLApp
Contain class AccessAVLApp.

- **Which has AVLTreeList oklist to store oklist entries.**
- **2 methods:**
  - **printAllStudent which output every data entry of the AVLTreeList oklist.**
  - **printStudent takes in a key which is the identity of the student as type Student and output a single specific data entry by comparing all keys in the AVLtreelist oklist and if found output the match as student Identity and student Full name or if not found outputs "AccessDenied!".**

Testing:

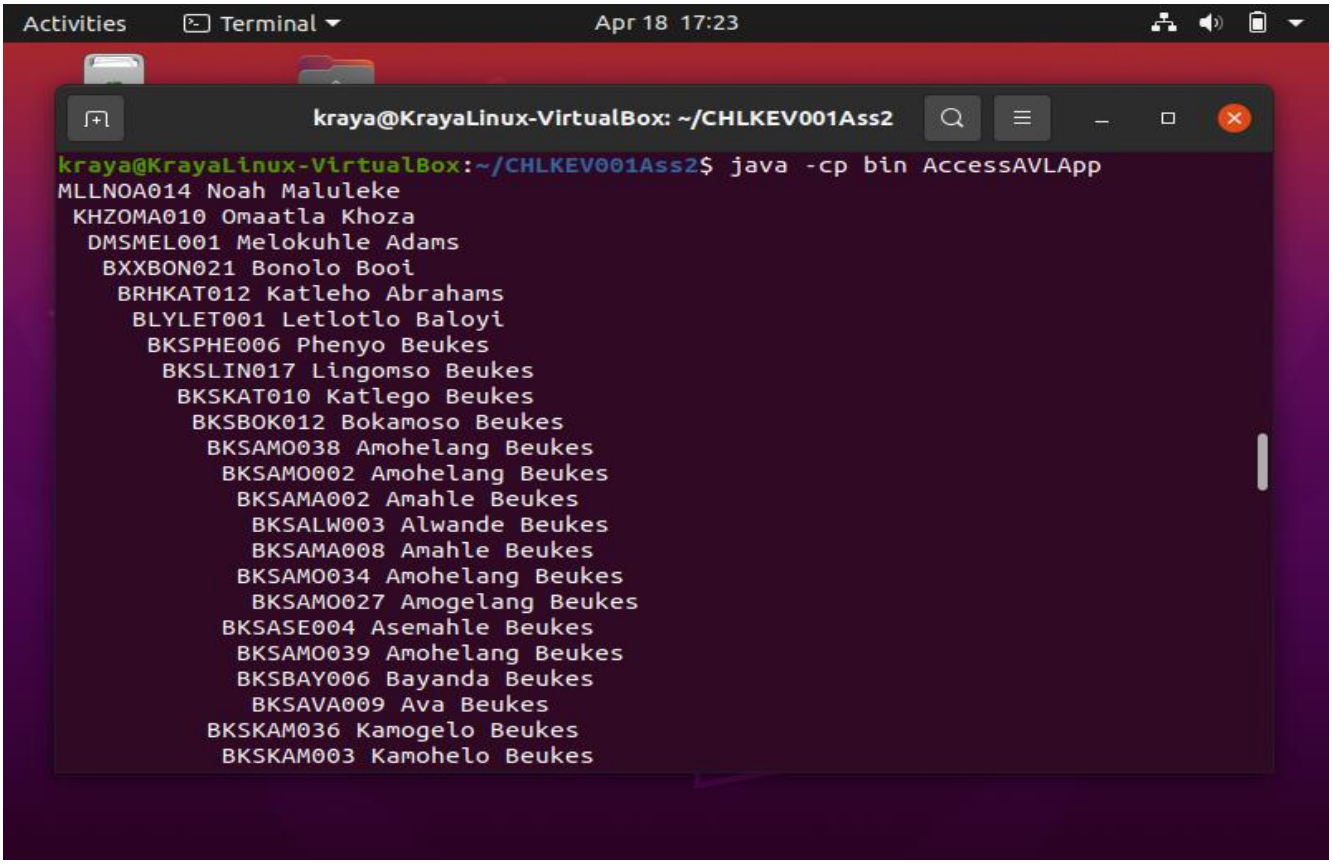
The main objective of the assignment is to test the performance of the AVL Tree, using a file containing students credentials by looping and searching the data using methods printAllStudent and printSudent.

Testing each application with 3 known parameters that work and 3 invalid parameters and without any parameters.

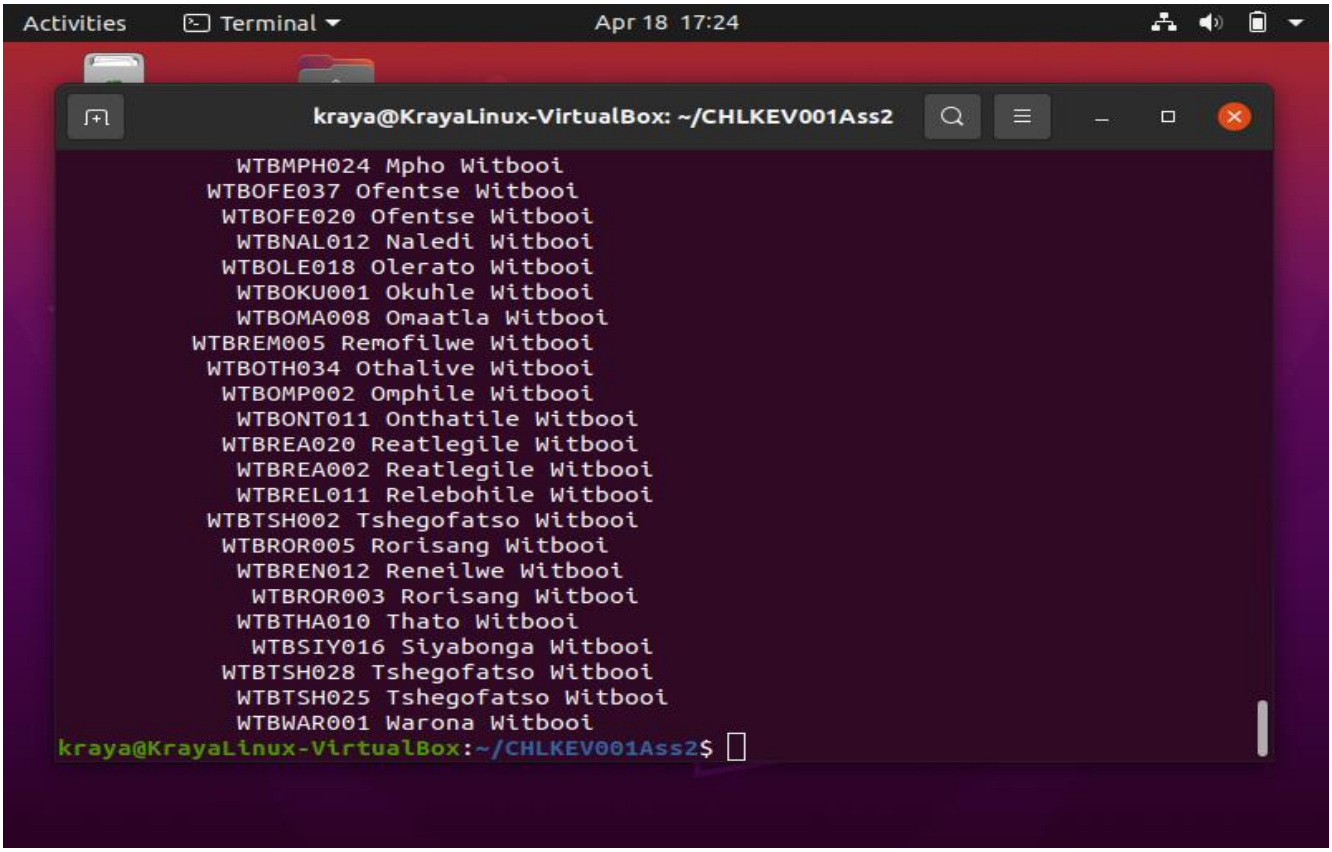
Following are snippets of the code running by testing values :

-AccessAVLApp

Front for printAllStudent

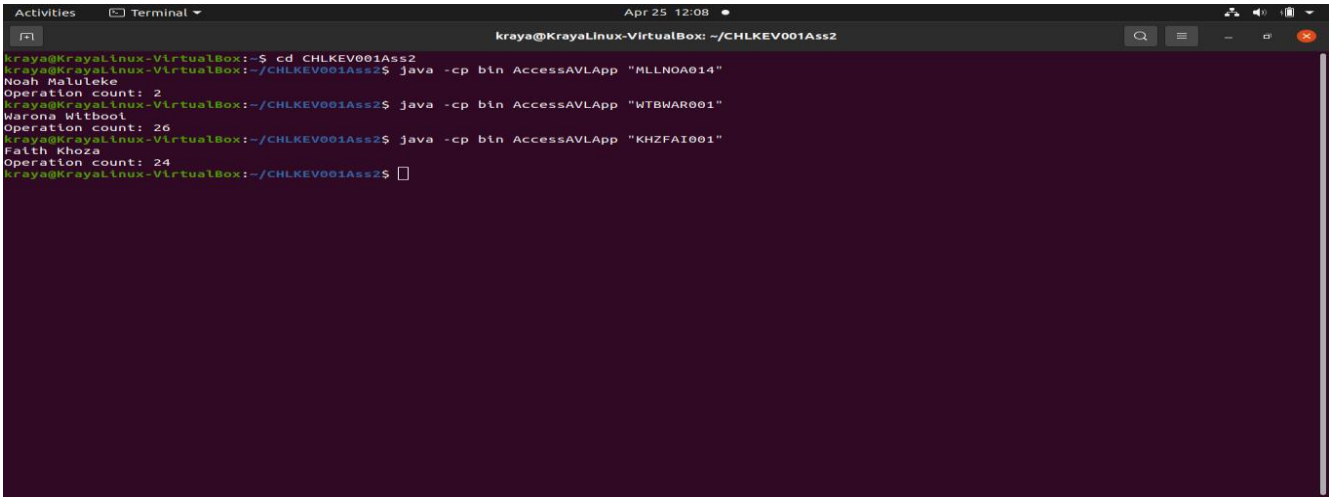


End for printAllStudent

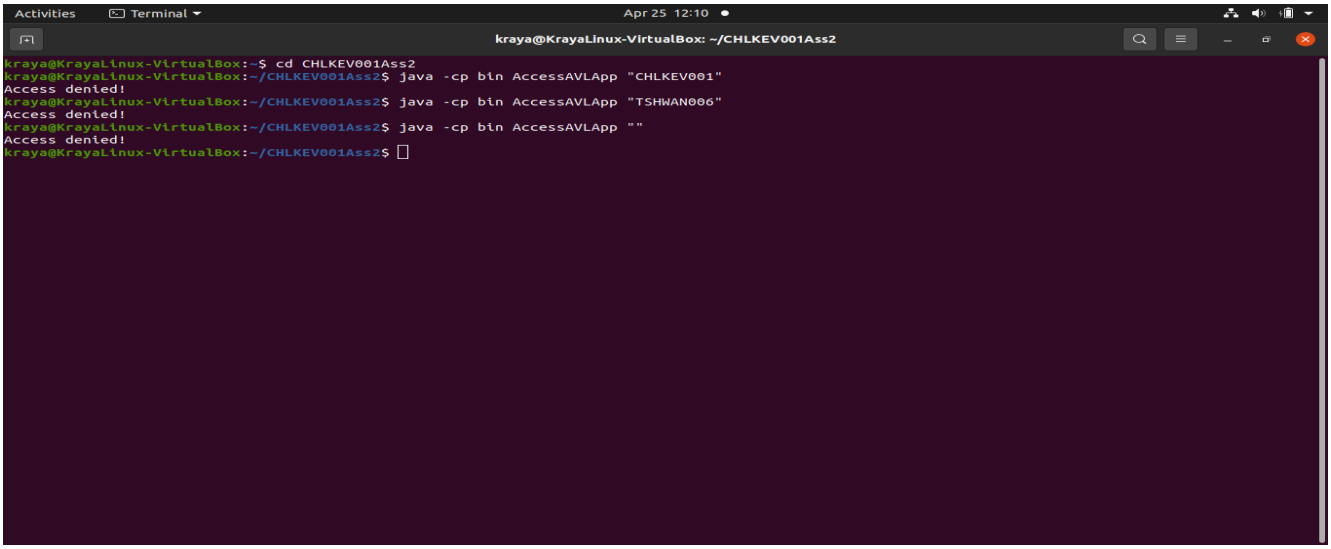


printStudent

3 valid inputs



3 invalid inputs

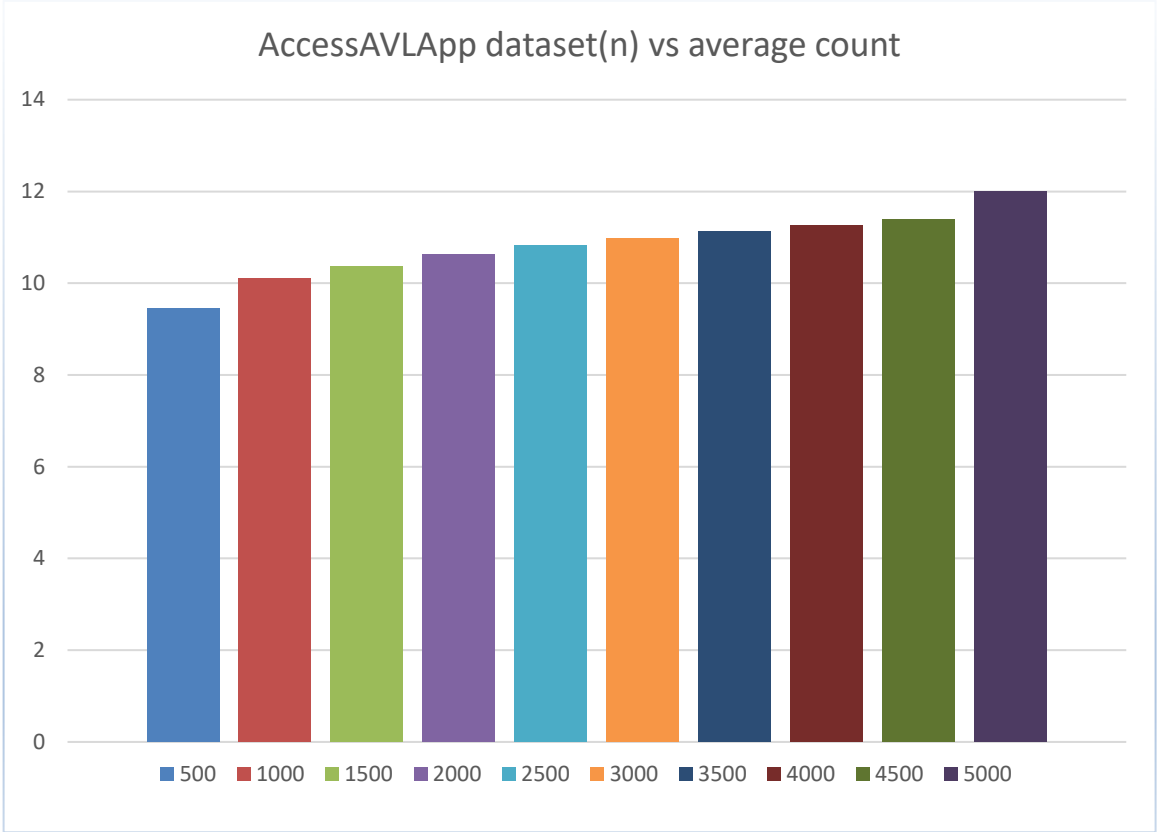


Results

table

Dataset(n)	Name of Count	Average of count
500	AccessAVLapp500	09.444
1000	AccessAVLapp1000	10.099
1500	AccessAVLapp1500	10.366
2000	AccessAVLapp2000	10.639
2500	AccessAVLapp2500	10.823
3000	AccessAVLapp3000	10.985
3500	AccessAVLapp3500	11.129
4000	AccessAVLapp4000	11.263
4500	AccessAVLapp4500	11.393
5000	AccessAVLapp5000	12

Graph



Discussion of results

The AVLTree data structure spends more time during insertion of data balancing the tree and thus affecting its performance.
The AVLTree data structure is also more efficient at searching information in the given data because the balancing of the tree is already in place to accommodate dataset n of small or big size.
The AVLTree it was able to locate the desired student identity which was at the beginning of the balanced tree by 2 operation counts which is the best case: 0(1)
The AVLTree it was able to locate the desired student identity which was at the middle of the balanced tree by 24 operation counts which is the average case: 0(logn).
The AVLTree it was able to locate the desired student identity which was at the end of the balanced tree by 26 operation counts which is the worstcase: 0(logn).

The best case could be of the result that the desired student identity was at the root node and for the average and worse is the that the big 0 refers to the sub right tree being a height bigger than the left sub tree for every node.
The AVLTree data structure is very advantageous in circumstances where resource usage is rendered trivial and only time efficiency is of consideration.

Creativity
AccessAVLApp is supported by the functionality of few classes including student, readFromFile and AVLTree.
Instrumentation, operation count was added inside the AVLtree class, incremented the count on function find and a getter method was used to get the count value.
AccessAVLApp has a method to write the operation count to a file called AccessAVLApp count.
Scripty.py then reads the file AcessAVLAppCount in order to fulfill a automated testing on the dataset n size by 500 difference amounting to 10 experiments outputted in 10 different files.
The 10 file are then used in excel to the calculate the average count of each file.

Git usage log

```
Activities   Terminal   Apr 25 11:58
kraya@KrayaLinux-VirtualBox: ~/CHLKEV001Ass2
kraya@KrayaLinux-VirtualBox:~/CHLKEV001Ass2$ git log | (ln=0; while read l; do echo $ln\: $l; ln=$((ln+1)); done) | (head -35; echo ...; tail -35)
0: commit 3363ce83f5a3822d17eb3515a5480f87f669672b
1: Author: @artist kev <CHLKEV001@myuct.ac.za>
2: Date: Sun Apr 25 01:12:24 2021 +0200
3:
4: Final version
5:
6: commit e962c3f18c1178d1a9f0fb7af3b6a5a7a864e3a7
7: Author: @artist kev <CHLKEV001@myuct.ac.za>
8: Date: Sat Apr 24 12:38:56 2021 +0200
9:
10: version 3.0
11:
12: commit 8501ebde31483b39bed3d3620564332d0805c832
13: Author: @artist kev <CHLKEV001@myuct.ac.za>
14: Date: Fri Apr 23 17:12:34 2021 +0200
15:
16: version 2.5
17:
18: commit a256be23ac6ebe2ff20abee1e3bc8e9c1b5c9e71
19: Author: @artist kev <CHLKEV001@myuct.ac.za>
20: Date: Wed Apr 21 09:32:35 2021 +0200
21:
22: version 2.0
23:
24: commit 1cae4406c9f3f48c8ef861703876ddd53031ddd4
25: Author: @artist kev <CHLKEV001@myuct.ac.za>
26: Date: Fri Apr 16 03:58:49 2021 +0200
27:
28: added operation calculatorsand file to store the operations
29:
30: commit c25dab1ebd5fe68011e2c0476f5071c7d5078eb9
31: Author: @artist kev <CHLKEV001@myuct.ac.za>
32: Date: Tue Apr 13 23:57:28 2021 +0200
33:
34: version 1.0 - AVLTree
...
kraya@KrayaLinux-VirtualBox:~/CHLKEV001Ass2$
```