CSC2001F Assignment 2 - Report

Jing Yeh
YHXJIN001

Table of Contents

To Start the Program

Two versions of the program for users' convenience are included: the UIApp.java displays a text-based interface with additional functionalities such as adding a single statement to the knowledge base. AutomatedApp.java is the version with less flexibility but allows more efficient scaling and invocation. Only the UI version of the app can be run using

```
make runUI
```

However, to view the result of experiment, please run

```
make runExperiment
```

This command invokes a python script that automate every aspect of the experiment as well as generate a beautiful graph 🤓

OOP Design

The java files can be roughly categorised into two types, one dealing with the AVL tree, the other dealing with the users.

Category 1: AVL Tree

Some of the classes from Assignment 1 have been reused, as the AVL tree can be considered as a modification of the **KnowledgeBaseTree**, a vanilla binary search tree, that we have implemented in the previous assignment, which is itself inherited from the abstract **KnowledgeBase** class. The term, sentence and score in each line have been grouped into one single data type named **Entry**, which is also the parent class of **EntryNode**, the hybrid of BinaryTreeNode and Entry. This is done to promote code reusability and to prevent over-modularisation.

Category 2: Users

As mentioned previously, two versions of the main programs have been included for different purposes. The user either can open the **UIApp** version of the program that has a text-based menu, or run the python experiment script, **experiment.py** to view the relevant data and analyses relevant to the AVL tree. An **KnowledgeBaseAppActions** class, which acts as a bridge between the app and the tree, has been constructed to promote separation of concern and code reusability.

Experiments

The goals of the experiment are to ensure that the AVL tree is correctly implemented as well as comparing the actual time complexities of this implementation to its theoretical time complexities.

The procedure is as followed:

1. Test if the AVL tree can correctly load the GenericsKB.txt file by writing all the nodes to a txt file named output.txt.
2. Use a manually constructed, toy txt file to test if the tree correctly handle statements already loaded into the data set and those that are not.
3. The performance of **searchEntry()** and **insert()** of the KnowledgeBaseAVLTree is then compared to their respective theoretically best, worst, and average performances

1. and 2. Testing the AVL Tree

To test the insert functionality of the AVL tree, the GenericKB.txt file is first loaded into the tree. Afterwards, a txt file containing 10 terms, including both of those that are in GenericKB.txt and those that are not, is loaded and searched for in the AVL tree. The result is then written into a file named output.txt, which can be found in the "output" folder.

The following terms are included in the test file:

```
insurance broker
tree
Joseph Joestar
food
Gio Gio
Marcus Aurelius
coconut
Homo habilis
australopithecus
low resource natrual language processing
plot
```

The result is consistent with the expectation:

```
Please enter the file path (Load GenericsKB-queries.txt if input is empty)
../data/Test-queries.txt
insurance broker    An insurance broker is a broker    1.0000
tree    Trees rely on photosynthesis    1.0000
Term not found: Joseph Joestar
food    Food creates chemical reaction    1.0000
Term not found: Gio Gio
Term not found: Marcus Aurelius
coconut    Coconut isa matter    1.0000
Term not found: Homo habilis
Term not found: australopithecus
Term not found: low resource natrual language processing
plot    Plots continue for years    1.0000
```

3. Experiment

The theoretical performances of the AVL tree are as follow:

	insert()	**searchEntry()**
Best	$O(1)$	$O(1)$
Average	$O(\log n)$	$O(\log n)$
Worst	$O(\log n)$	$O(\log n)$

The empirical results obtained from loading random sets of data can be roughly replicated using the python script provided. To run, simply type in

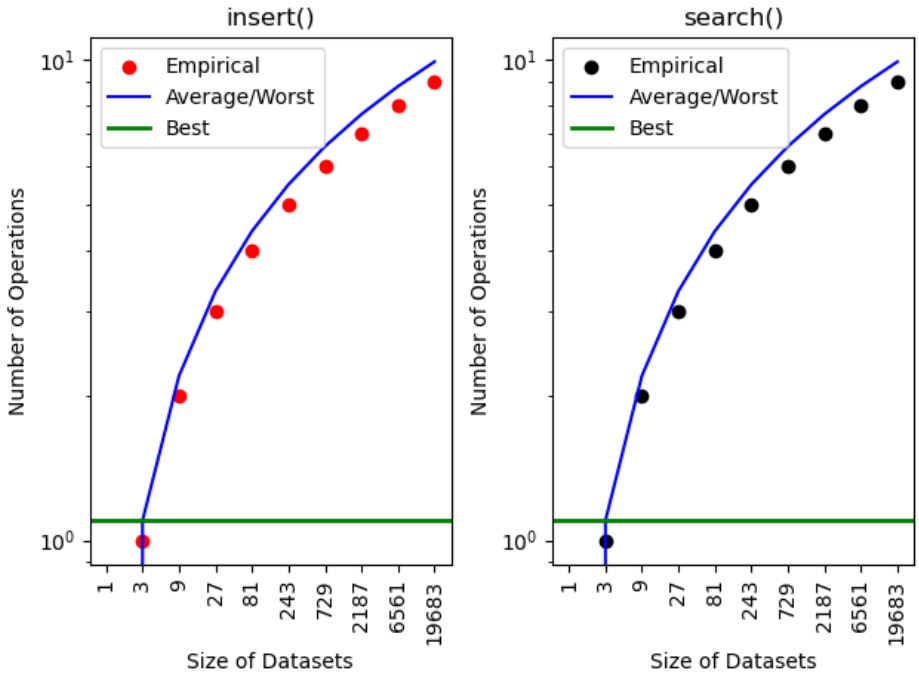
```
make runExperiment
```

The python script **experiment.py** does two things: It generates random sets of data, with sizes differ logarithmically (log_3 is used as it is the largest integer that provides 10 sets of data within the range of 50000).

These sets of data are then loaded one at the time by **AuomaticApp.java,** the version of the app that automates loading, inserting, and searching of data. The app will then write the

result of each run into a csv file, allowing the python script to access the results obtained, and graph them using **matplotlib**

The results are summarised below. We can see that the empirical results generally follow the theoretical average and worst case time complexity. Note that the time complexities for seach() differ significantly in the beginning, but eventually converge, as they are both bounded by the same function $\log n$



Creativity

A python script is included for more efficient and scalable testing. Every aspect of the experiment is fully automated, from extracting query files for testing, to generating diagrams using **matplotlib** and **numpy**. The java program is run directly inside the python script, without the need of users to manually inputing options: One can simply type in

```
make runExperiment
```

in terminal to replicate the experiment.

An UI version of the app is also included, for quick testing as well as to provide an enhanced user experience.

Git log

```
0: commit 4d6462f4fb1700a1e31be00fad77291482884f8d
1: Author: Jing Yeh <yhxjin001@myuct.ac.za>
2: Date:   Fri Mar 22 18:26:37 2024 +0200
3:
4: A2: Generate javadoc
5:
6: commit c92d2ae0bf788bbf0bbfa4a7e471fbad6c134404
7: Author: Jing Yeh <yhxjin001@myuct.ac.za>
8: Date:   Fri Mar 22 09:39:07 2024 +0200
9:
10: A2: Regorganised makefile
```

```
40: A2: Implemented rotations
41:
42: commit 15dfab58f4c07d1b7ec532b20a2e2c23923d3862
43: Author: Jing Yeh <yhxjin001@myuct.ac.za>
44: Date:   Thu Mar 21 11:50:30 2024 +0200
45:
46: A2: Set up assignment 2
47:
48: commit 2136cb69488a6bd422abf9df6f8a7626dd12275c
49: Author: Jing Yeh <yhxjin001@myuct.ac.za>
50: Date:   Sat Mar 9 23:11:08 2024 +0200
```