

## CSC2001F 2025 Data Structures Assignment 2

---

### [CSC2001F 2025 Data Structures Assignment 2](#)

[My Object Oriented design:](#)

[Classes Created:](#)

[How they interact:](#)

[Testing Process \(Part 1\):](#)

[Experiment \(Goal, Execution, Results, & Discussion\):](#)

[Goal:](#)

[Execution:](#)

[Results:](#)

[Discussion:](#)

[Creative aspect:](#)

[Git Usage:](#)

[GitHub link:](#)

---

### My Object Oriented design:

I created several different classes that interact with each other, making my programmes object-oriented. Object-oriented design promotes the modularity of my code and feeds well into the workflow of Java programmes, the language I used (on top of Python) to code my work. This coding approach was beneficial when considering this project's expanse compared to my previous CSC1015/16 assignments.

### Classes Created:

The **GenericsKbAVLApp** class manages the knowledge base using an AVL tree. It is an updated version of my **GenericsKbBSTApp**, with lessened functionality but heightened efficiency due to the balanced structure of the tree. It allows for loading a knowledge base from a file and storing it in a balanced tree, the searching and retrieving of terms from the knowledge base relative to a list of keys, and the parsing of said list of keys from a file.

The **Parsefile** class reads and parses the input text file, storing the terms, sentences, and confidence scores in a list of String arrays for use by the **GenericsKbAVLApp**. It is the same class as in Assignment 1.

The **AVLTree** class is an updated version of the **BinarySearchTree** class and also manages the knowledge base used with a tree, however, the insertion and search functionality was

updated to include rotations so that the tree is always balanced. It also supports efficient insertions, traversal, and searches.

The **Node** class represents an entry in the AVLTree, storing a term, sentence, confidence score, and references to its left and right child nodes. It is the same as in Assignment 1 but with an added height characteristic.

The **Tree** interface provides a blueprint for implementing tree-based knowledge base management and is the same as in Assignment 1, apart from updated insertion functionality.

The **Experiment** class tests the efficiency of my AVL Tree by parsing the dataset, running trials of the AVL tree usage for multiple sizes of  $n$  with 10 trials for every  $n$ . It does so by inserting these sized random subsets into the AVL tree and then measuring the insert and search comparisons for each trial, saving the output to a formatted text file upon completion.

The **Graphing** program loads experiment results from a file and plots my AVL tree performance metrics, including insert and search comparisons, against dataset size ( $n$ ).

### How they interact:

The **GenericsKbAVLApp** class loads the knowledge base via the **Parsefile** class, storing the data in an AVL tree ( a balanced binary tree ). The **AVLTree** class is responsible for the organization of the database into a balanced binary tree structure for more efficient searching and insertion of nodes. The **Node** class and **Tree** interface are used in **AVLTree** to represent entries and define operations, respectively. The **Experiment** class tests the efficiency of my AVL tree implementation and the results of the experiment are graphed with the **Graphing** program.

---

### Testing Process (Part 1):

(NB: I used full directories as filenames eg:/home/abrmar043/Assignment 2 - Project/textfiles/TestQueries.txt )

Test 1 : (Query set sample)
-----------------------------

```

abrmr043@MSI:~/Assignment 2 - Project$ /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /home/abrmr043/.config/Code/User/workspaceStorage/92d5408e209704adec9641f0d2700b5c/redhat.java/jdt_ws/Assignment\ 2\ -\ Project_18\ 3b3/bin GenericsKbAVLApp
Welcome to GenericsKb :)

Choose an action from the menu:
1. Load a knowledge base from a file.
2. Search for terms in the file and print to an output file.
3. Quit

Enter your choice:

1
Enter the name of the file directory:
/home/abrmr043/Assignment 2 - Project/textfiles/GenericsKB.txt
Knowledge base successfully loaded.

Choose an action from the menu:
1. Load a knowledge base from a file.
2. Search for terms in the file and print to an output file.
3. Quit

Enter your choice:

2
Please enter the name of the file of list of keys you want to parse:
/home/abrmr043/Assignment 2 - Project/textfiles/TestQueries.txt
learn disability: Learn disability affects perceptions.(1.0)
helpless kitten: Helpless kittens reach sexual maturity.(1.0)
Term not found: hemopoiesis
Term not found: young lady
drug abuse: Drug abuse is a biopsychosocial disease.(0.7294536828994751)
osmotic laxative: Osmotic laxatives increase the water content in the stool.(0.7976997494697571)
dictator: Dictators are bands.(1.0)
dilettante: A dilettante is an amateur(1.0)
pika: Pikas have (part) pedal extremities.(1.0)
bisexual: A bisexual is a sensualist(1.0)

```

## Input vs Output:

```

textfiles > ≡ TestQueries.txt
1  learn disability
2  helpless kitten
3  hemopoiesis
4  young lady
5  drug abuse
6  osmotic laxative
7  dictator
8  dilettante
9  pika
10 bisexual

```

```

≡ output.txt
1  learn disability: Learn disability affects perceptions.(1.0)
2  helpless kitten: Helpless kittens reach sexual maturity.(1.0)
3  Term not found: hemopoiesis
4  Term not found: young lady
5  drug abuse: Drug abuse is a biopsychosocial disease.(0.7294536828994751)
6  osmotic laxative: Osmotic laxatives increase the water content in the stool.(0.7976997494697571)
7  dictator: Dictators are bands.(1.0)
8  dilettante: A dilettante is an amateur(1.0)
9  pika: Pikas have (part) pedal extremities.(1.0)
10 bisexual: A bisexual is a sensualist(1.0)
11

```

## Test 2: (against the entire query set)

```
depot: A depot is a station.(1.0)
Term not found: twaddle
increase: Increases are processes.(1.0)
movie camera: A movie camera is a camera.(1.0)
widespread campaign: Widespread campaigns result in development.(1.0)
gimbal lock: Gimbal lock occurs when the first and third axis of a joint line up.(0.7308312058448792)
Term not found: obituary
guerilla warfare: Guerilla warfare is actions.(1.0)
cashew: Cashews have (part) kernels.(1.0)
garnishment: A garnishment is court order(1.0)
Term not found: droop
Term not found: stadium
Term not found: catholicity
rag: Rag isa piece.(1.0)
grammatical gender: A grammatical gender is a category(1.0)
yellow iris: A yellow iris is a flag(1.0)
campground: A campground is a site(1.0)
cloth laboratory coat: Cloth laboratory coats provide protection.(1.0)
cephalosporin: Cephalosporins also inhibit the synthesis of cell walls as penicillin does.(0.7893165349960327)
female flea: Female fleas lay eggs, usually on the host, after taking a blood meal.(0.7511987686157227)
tetherball: A tetherball is an athletic game(1.0)
Term not found: new concept
odonate: Odonates hatch from the eggs as an aquatic form known as a nymph.(0.7562242150306702)
blue pyramid: Blue pyramids represent the active copper ions coordinated by five oxide ions.(0.7440516352653503)
finger millet: A finger millet is a millet(1.0)
Term not found: antler
climax: A climax is an occasion(1.0)
native tribe: Native tribes use feathers.(1.0)
tax lien: A tax lien is a lien(1.0)
exchange: Exchanges are conversations.(1.0)
hunger strike: A hunger strike is nonviolence(1.0)
raw broccoli: Raw broccolis have taste.(1.0)
ocean habitat: Ocean habitats include environments.(1.0)
Term not found: postmark
hygiene: Hygiene isa condition.(1.0)
short sale: A short sale is trading(1.0)
erythrocyte: Erythrocytes also lack all other subcellular organelles such as mitochondria.(0.7949842810630798)
teller: Tellers have (part) necks.(1.0)
thick layer: Thick layers separate layers.(1.0)
Term not found: recycle bin
harvest mouse: A harvest mouse is a mouse(1.0)
rough water: Rough water is bad weather(1.0)
Utah: Utah isa thing.(1.0)
business school: A business school is a graduate school(1.0)
few scientist: Few scientists pay attention.(1.0)
```

etc.

## Input vs Output:

```
textfiles > GenericsKB-queries.txt
~/Assignment 2 - Project/textfiles • Contains emphasized items
1  soft drink
2  maser
3  onion ring
4  concentration
5  alcoholic cirrhosis
6  nemertean worm
7  diaphragm
8  wild garlic
9  medical receptionist
10 competitor
11 funny
12 petroleum coke
13 generalized epilepsy
14 public speaker
15 golden plover
16 various chemical
17 hypotonicity
18 magnolia tree
19 fever virus
20 mineralogical factor
21 upland
22 chemical pneumonitis
23 water loss
24 binary fission
25 ordinary light
26 coastal city
27 learn disability
28 helpless kitten
```

etc.

```

≡ output.txt
1  soft drink: Soft drinks are beverages.(1.0)
2  maser: A maser is an amplifier(1.0)
3  onion ring: Onion rings are finger food.(1.0)
4  Term not found: concentration
5  alcoholic cirrhosis: Alcoholic cirrhosis is the destruction of normal liver tissue, leaving non-functioni
6  nemertean worm: Nemertean worms are long, thin, animals without segments.(0.8558743000030518)
7  Term not found: diaphragm
8  wild garlic: A wild garlic is a bulbous plant(1.0)
9  Term not found: medical receptionist
10 competitor: Competitors are located in sporting events.(1.0)
11 funny: Funnies are located in television.(1.0)
12 petroleum coke: Petroleum coke is a mixture of carbon and hydrogen joined by nature in many different way
13 generalized epilepsy: Generalized epilepsy is epilepsy(1.0)
14 Term not found: public speaker
15 golden plover: A golden plover is a plover(1.0)
16 various chemical: Various chemicals reverse the process of blood clotting.(0.768812358379364)
17 hypotonicity: Hypotonicity is a condition of having abnormally low tension or tone, especially of the mus
18 magnolia tree: Magnolia trees produce conspicuous flowers, which contain multiple stamens and multiple pi
19 fever virus: Fever viruses cause illnesses.(1.0)
20 Term not found: mineralogical factor
21 upland: An upland is elevation(1.0)
22 chemical pneumonitis: Chemical pneumonitis occurs when aspirated material is directly toxic to the lungs.
23 water loss: Water loss is reduced when the respiratory surfaces are somehow covered.(0.7047173976898193)
24 binary fission: Binary fission is the main source of reproduction in eubacteria.(0.7857990264892578)
25 ordinary light: Ordinary light is a mixture of light of many different colors, i.e. wavelengths.(0.764071
26 coastal city: Coastal cities are cities.(1.0)
27 learn disability: Learn disability affects perceptions.(1.0)
28 helpless kitten: Helpless kittens reach sexual maturity.(1.0)
29 tellurium: Telluriums are minerals.(1.0)
30 Term not found: hemopoiesis
31 zoning: Zonings are division.(1.0)
32 twentieth century: A twentieth century is a century(1.0)
33 conjoined twin: Conjoined twins are identical twins.(1.0)
34 biological fitness: Biological fitness is measured as number of offspring.(0.7671203017234802)
35 Term not found: crimp

```

etc.

Exiting:

```

Choose an action from the menu:
1. Load a knowledge base from a file.
2. Search for terms in the file and print to an output file.
3. Quit

Enter your choice:

3

Thanks for Using GenericsKb :)abrmr043@MSI:~/Assignment 2 - Project$

```

## Experiment (Goal, Execution, Results, & Discussion):

**Goal:**

The primary objective of this experiment is to evaluate the performance of an AVL Tree in terms of insertion and search operations. Specifically, the experiment aims to:

1. **Measure Efficiency:** Quantify the number of comparisons required for inserting and searching elements in an AVL Tree.
2. **Analyze Time Complexity:** Compare the observed experimental results with the expected theoretical time complexity of AVL Trees, which is  $O(\log N)$  for both insertion and search operations.
3. **Validate AVL Balancing:** Ensure that the self-balancing property of the AVL Tree maintains optimal performance across different input sizes.
4. **Assess Scalability:** Examine how the tree performs as the dataset size increases from 5 to 50,000 elements.

### Execution:

### Structure:

Component	Description
Static Variables	Define dataset sizes (trial_sizes), number of trials (num_trials), and file path
main() Method	Controls program flow: Takes user input, loads data, runs experiments, and saves results to a formatted output file
runForSize()	Conducts trials for a given dataset size, computes statistics, and writes result
getRandomSubset()	Randomly select a subset of data for testing

### Execution Method

The experiment follows a structured approach to analyze AVL tree efficiency. The program begins by reading a dataset (*GenericsKB.txt*) and allowing the user to specify an output filename for results storage. For each dataset size ( $n = 5, 50, 500, 1000, 2500, 5000, 7500, 10000, 25000, 50000$ ), it performs **10 independent trials (iid)**, ensuring unbiased testing by randomly selecting and shuffling a subset of data before processing.

In each trial, the randomly selected data is **inserted into an AVL tree**, which maintains balance through rotations to optimize search efficiency. The program logs the total number of comparisons required for each insertion through the incrementing of an **InsertComparisonCount** variable as well as an **InsertOperationCount** variable. Next, the **search operation** is conducted using the first 100 entries (or fewer if  $n < 100$ ), recording the number of comparisons for each search through the incrementing of a **SearchComparisonCount** variable as well as a **SearchOperationCount** variable; wherever there was an operation I wanted to count, I incremented the counter.

Once insertions and searches are completed, the program performs a **statistical analysis** of insertion and search metrics. It calculates the **minimum, average, and maximum comparisons** per operation for both the Search and Insert operations and compares them to the theoretical expectation of  $\log_2(n)$ . This is done for both the total counts of the comparisons as well as the average counts as n increases so we can check for both **consistency** across the experiments despite sample size as well as **observe the trend in count as n increases**. The collected results are structured in a table format and saved in the specified output file, while raw metrics are displayed in the console for verification.

The AVL tree plays a critical role by ensuring that operations maintain a logarithmic time complexity, preventing degradation in performance as the dataset size increases. We are ultimately trying to prove this to be true through our implementation of the experiment.

Results:

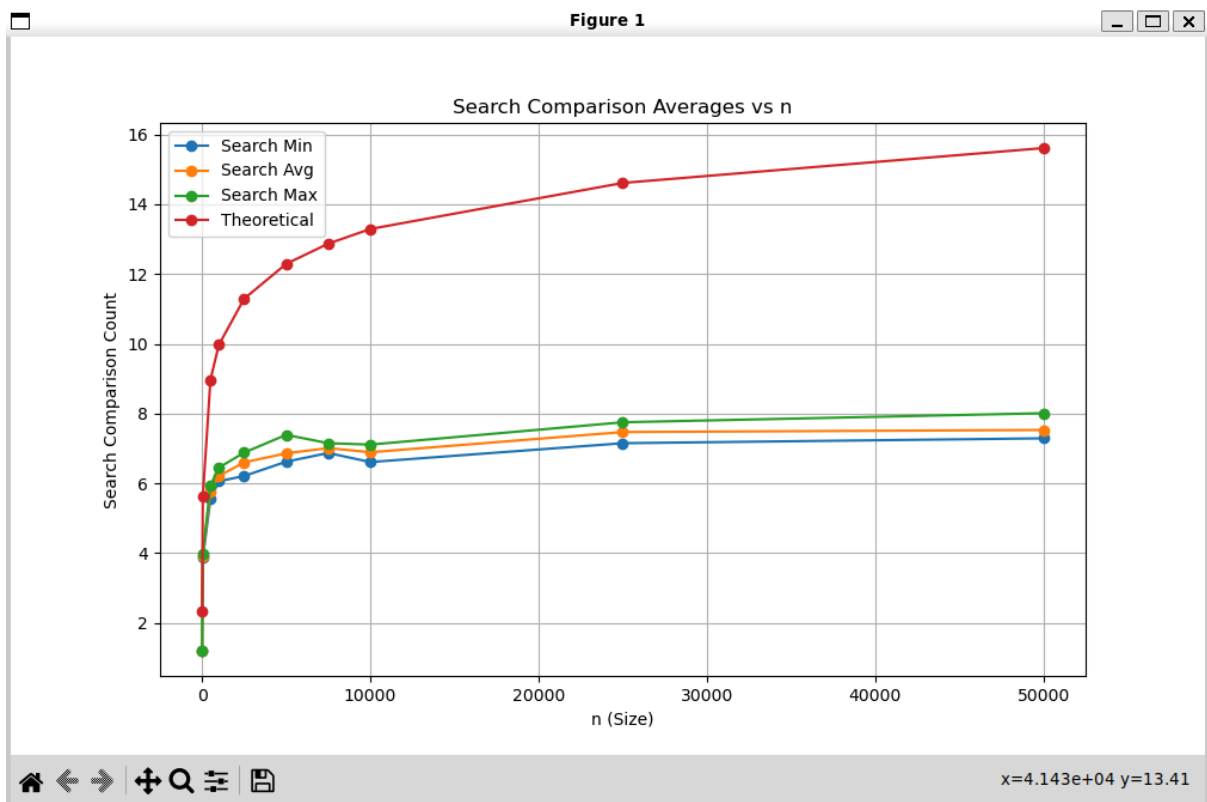
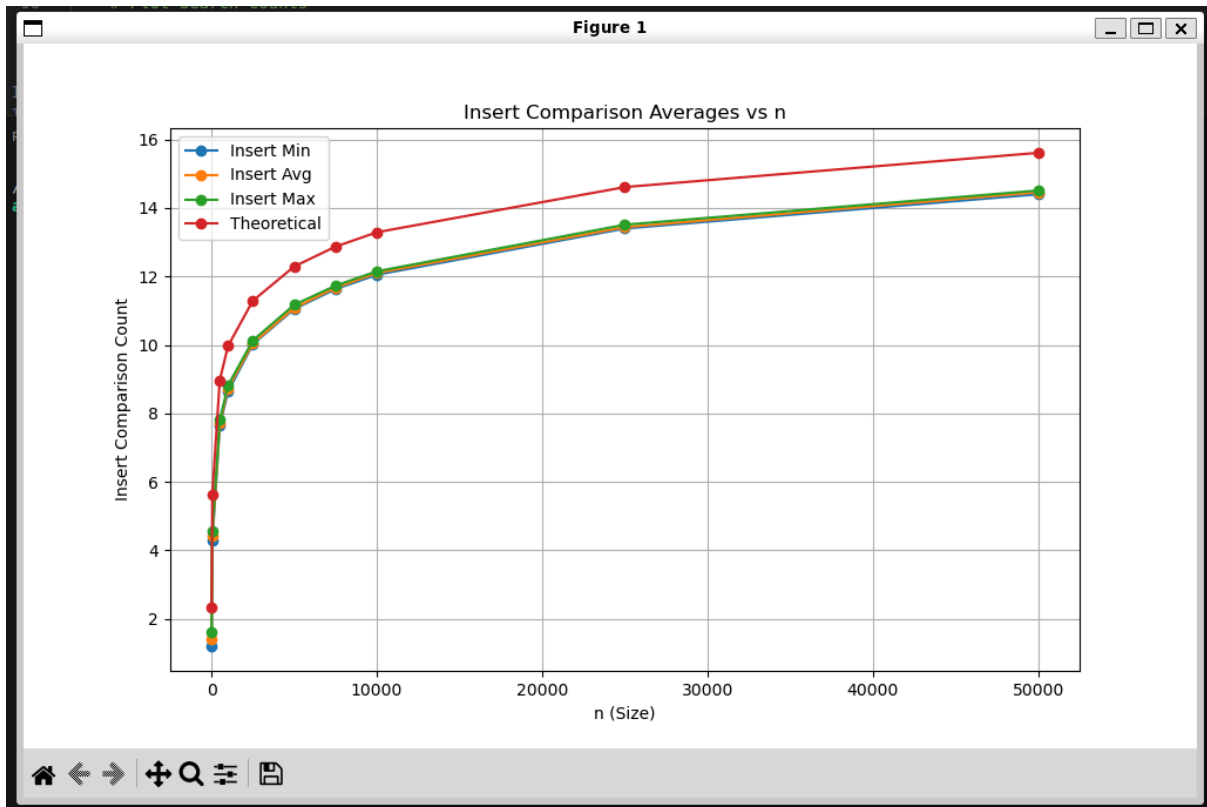
Experiment Output:

textfiles > results.txt  
You, 3 hours ago | 1 author (You)

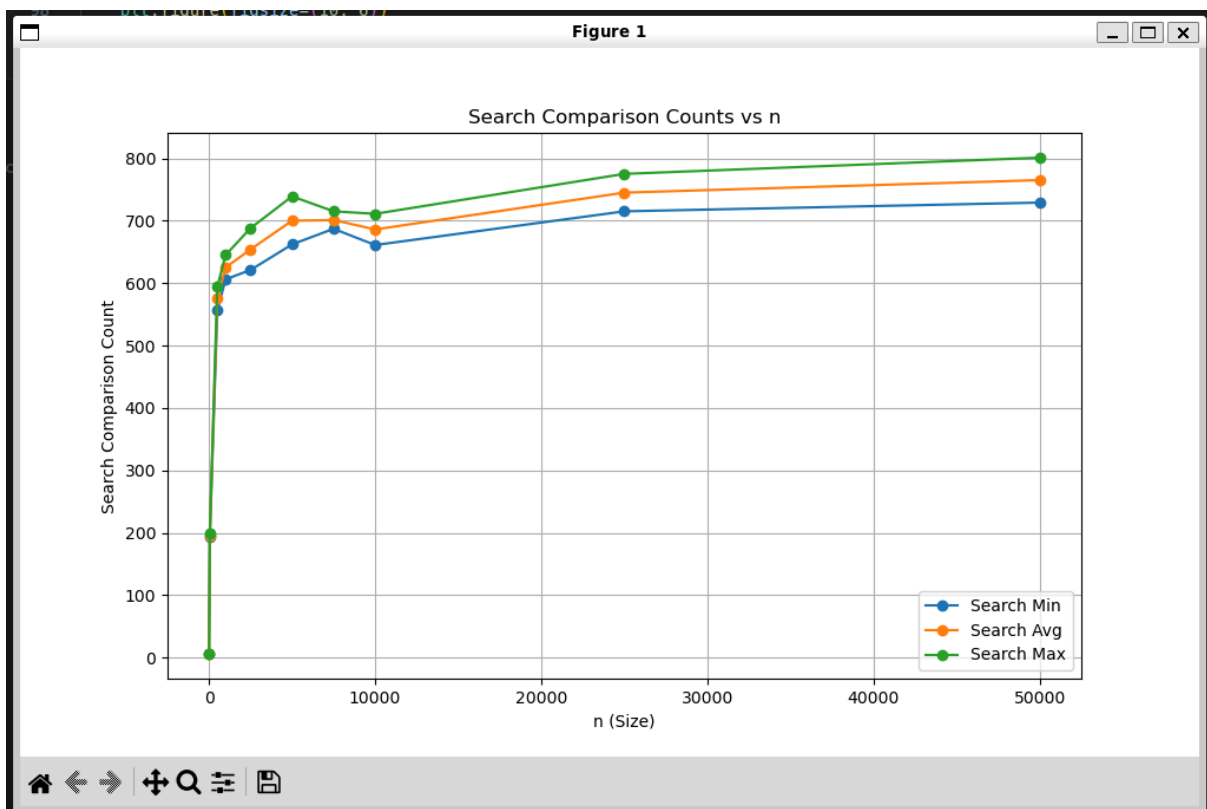
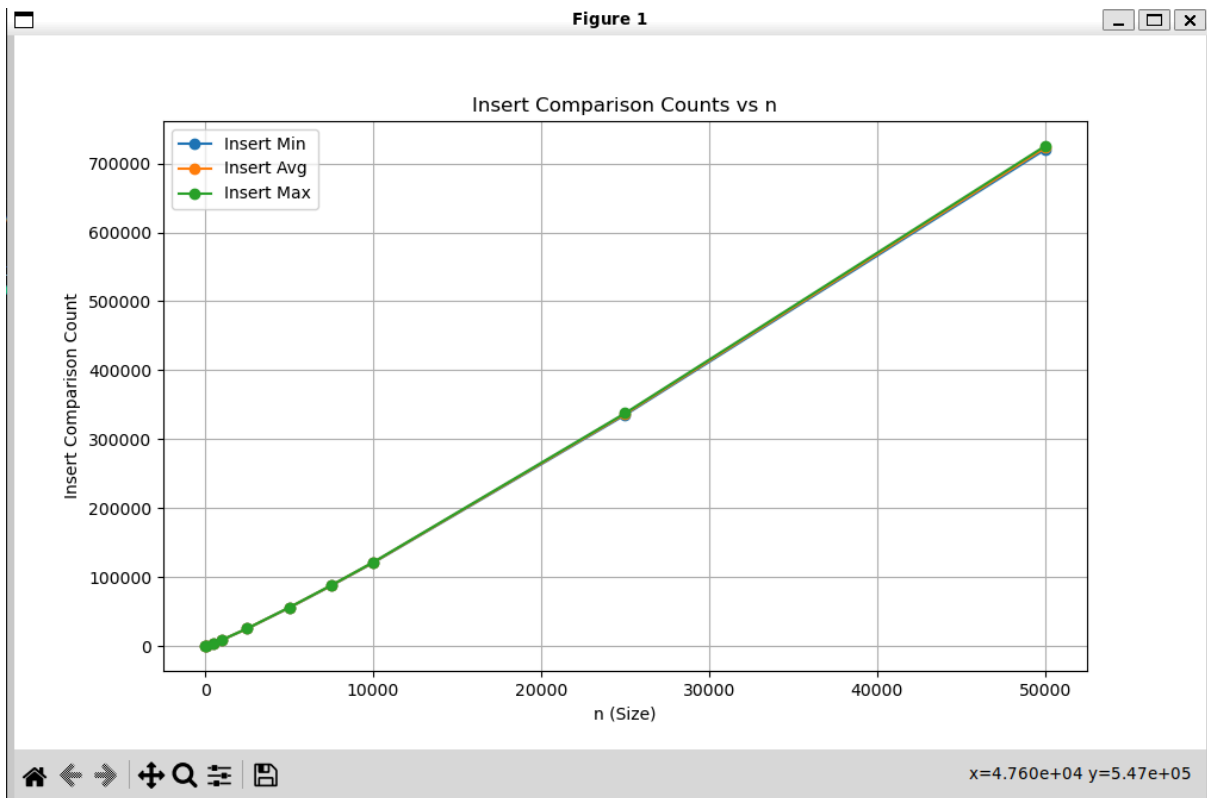
1	n	Insert Min	Count	Insert Avg	Count	Insert Max	Count	Theoretical (i)
2	5	1.20	6	1.42	7	1.60	8	2.32
3	50	4.28	214	4.42	221	4.56	228	5.64
4	500	7.65	3827	7.71	3866	7.81	3905	8.97
5	1000	8.64	8638	8.73	8730	8.82	8823	9.97
6	2500	10.02	25062	10.05	25195	10.13	25328	11.29
7	5000	11.05	55241	11.08	55539	11.17	55837	12.29
8	7500	11.63	87255	11.67	87603	11.73	87952	12.87
9	10000	12.05	120537	12.10	120996	12.15	121456	13.29
10	25000	13.40	335111	13.44	336421	13.51	337732	14.61
11	50000	14.40	720160	14.46	722727	14.51	725294	15.61

Search Min	Count	Search Avg	Count	Search Max	Count	Theoretical (s)
1.20	6	1.20	6	1.20	6	2.32
3.86	193	3.92	196	3.98	199	5.64
5.56	556	5.78	575	5.94	594	8.97
6.06	606	6.21	625	6.45	645	9.97
6.21	621	6.60	654	6.88	688	11.29
6.62	662	6.86	700	7.39	739	12.29
6.87	687	7.01	701	7.15	715	12.87
6.61	661	6.89	686	7.11	711	13.29
7.15	715	7.47	745	7.75	775	14.61
7.29	729	7.53	765	8.01	801	15.61

Graphs:







### Discussion:

The experimental results confirm that the AVL tree maintains logarithmic time complexity for both insertion and search operations. This is evident from how the number of comparisons closely follows the expected logarithmic growth pattern.

## Complexity Analysis

### □ Worst Case:

- search -  $O(\log n)$
- insert -  $O(\log n)$
- delete -  $O(\log n)$

Interestingly, insertion operations consistently performed better than the theoretical predictions. This efficiency is likely due to the tree's self-balancing property, which helps keep path lengths shorter on average.

Search operations were even more efficient, requiring only 7.53 comparisons at  $n=50,000$ —less than half the theoretical expectation. A possible explanation for this is the way searches were conducted: since they were performed on the first 100 inserted keys, these keys may have ended up in shallower levels of the tree, reducing search depth. Additionally, the AVL tree's balancing mechanism proved highly effective, as shown by the small difference between the minimum and maximum comparisons.

Overall, the experiment demonstrated strong scalability across all size variations in  $n$ . Despite increasing the input size by four orders of magnitude, insertion comparisons only grew by a factor of  $\sim 10$ , while search comparisons increased by just  $\sim 6$  when compared to expected logarithmic growth.

---

## Creative aspect:

To better analyze the time complexity, instead of only creating a graph for the Insert and Search Comparison counts, I graphed both raw comparison counts and per-operation averages so we can better observe :

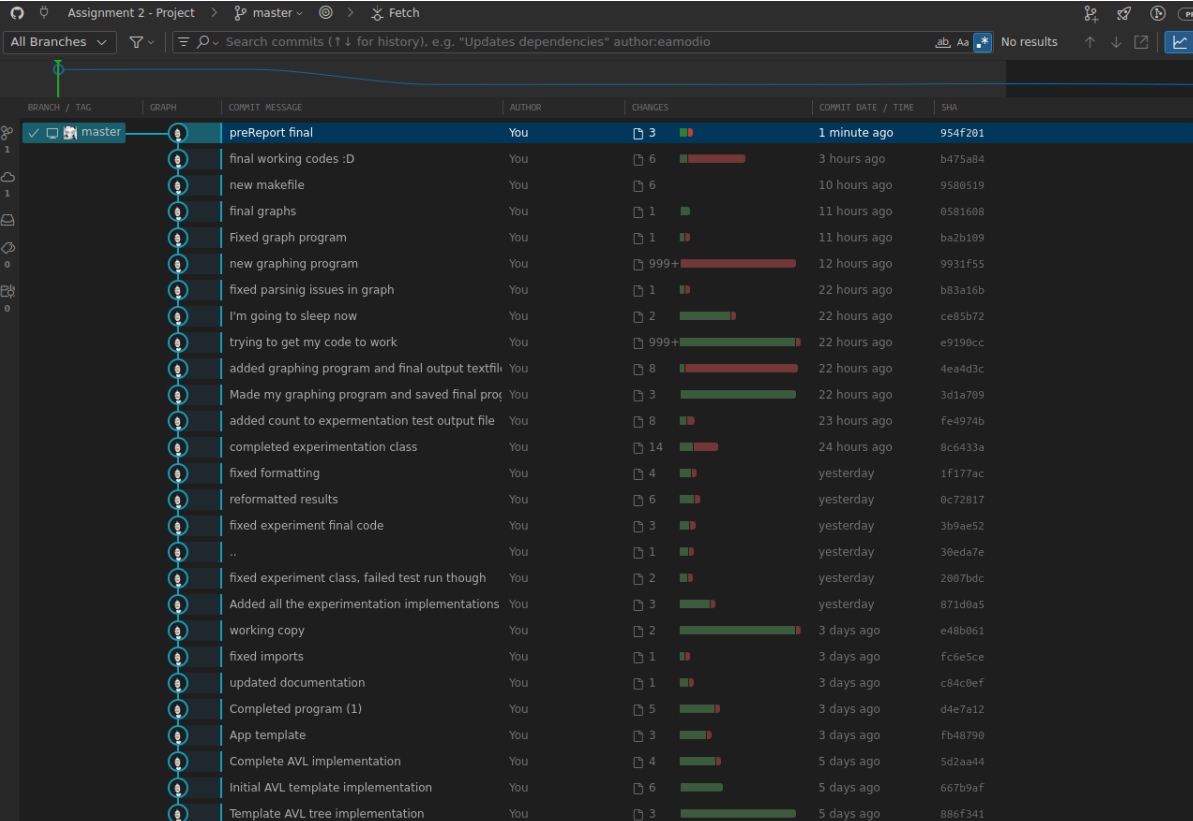
- 1) Potential values for hidden constant  $O$  ( in  $O(\log(n))$  )
- 2) Consistency across trials for all big and small  $n$
- 3) Clear scaling trends as  $n$  grows

The use of more graphs provides deeper insight than the raw counts alone, as seen in the results above.

---

## Git Usage:

```
abrmar043@MSI:~/Assignment 2 - Project$ git log | (ln=0; while read l; do echo $ln\: $l; ln=$((ln+1)); done) | (head -10; echo ...; tail -10)
0: commit 954f20175f3163228b87dae1a8156c3bec93f87b
1: Author: Maryam Abrahams <abrmar043@myuct.ac.za>
2: Date: Fri Mar 28 21:17:18 2025 +0200
3:
4: preReport final
5:
6: commit b475a84367d472693425dc51e903eb74f1d430a3
7: Author: Maryam Abrahams <abrmar043@myuct.ac.za>
8: Date: Fri Mar 28 18:18:10 2025 +0200
9:
...
169: Author: Maryam Abrahams <abrmar043@myuct.ac.za>
170: Date: Sat Mar 22 16:33:34 2025 +0200
171:
172: initial commit
173:
174: commit 70bbe88a7699de3b57ef6c267052867f2401ef0a
175: Author: Geto Suguru <abrmar043@myuct.ac.za>
176: Date: Sat Mar 22 16:22:39 2025 +0200
177:
178: Initial commit
```



BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR	CHANGES	COMMIT DATE / TIME	SHA
✓ master		preReport final	You	3	1 minute ago	954f201
		final working codes :D	You	6	3 hours ago	b475a84
		new makefile	You	6	10 hours ago	9580519
		final graphs	You	1	11 hours ago	0581608
		Fixed graph program	You	1	11 hours ago	ba2b109
		new graphing program	You	999+	12 hours ago	9931f55
		fixed parsinig issues in graph	You	1	22 hours ago	b83a16b
		I'm going to sleep now	You	2	22 hours ago	ce85b72
		trying to get my code to work	You	999+	22 hours ago	e9190cc
		added graphing program and final output textfil	You	8	22 hours ago	4ea4d3c
		Made my graphing program and saved final proq	You	3	22 hours ago	3d1a709
		added count to experimentation test output file	You	8	23 hours ago	fe4974b
		completed experimentation class	You	14	24 hours ago	8c6433a
		fixed formatting	You	4	yesterday	1f177ac
		reformatted results	You	6	yesterday	0c72817
		fixed experiment final code	You	3	yesterday	3b9ae52
		..	You	1	yesterday	30eda7e
		fixed experiment class, failed test run though	You	2	yesterday	2007bdc
		Added all the experimentation implementations	You	3	yesterday	871d0a5
		working copy	You	2	3 days ago	e48b061
		fixed imports	You	1	3 days ago	fc6e5ce
		updated documentation	You	1	3 days ago	c84c0ef
		Completed program (1)	You	5	3 days ago	d4e7a12
		App template	You	3	3 days ago	fb48790
		Complete AVL implementation	You	4	5 days ago	5d2aa44
		Initial AVL template implementation	You	6	5 days ago	667b9af
		Template AVL tree implementation	You	3	5 days ago	886f341
		Template from Assignemnt 1	You	4	5 days ago	f96c483
		initial commit	You	1	5 days ago	82d1b5d
		Initial commit	Geto Suguru	1	6 days ago	70bbe88

## GitHub link:

<https://github.com/GetosMonkey/Assignment2-Repository.git>