Assignment 2

AVL Trees

VMBRUM001

Table of Contents

Experiment

Goal

The goal of the experiment is to demonstrate the performance of AVL trees given different permutations of input data.

Method

The experiment was conducted as follows:
20 different levels of randomization were selected. The levels of randomization are: 5,10,15,20…100 (i.e. every number with 5 as a factor between 5 and 100 inclusive)
For each level of randomization:

1. The dataset was randomised using the method described under the section “Randomisation Method”.
2. Each element from the now randomised dataset was inserted in an AVLTree and the number of comparisons made to do so was recorded.
3. Each element from the dataset was searched for in the AVLTree and the number of comparisons made to find a result was recorded.
4. From the data collected in steps 2 and 3, the best, worst and average number of comparisons for insertions and searches was calculated and saved in a file.

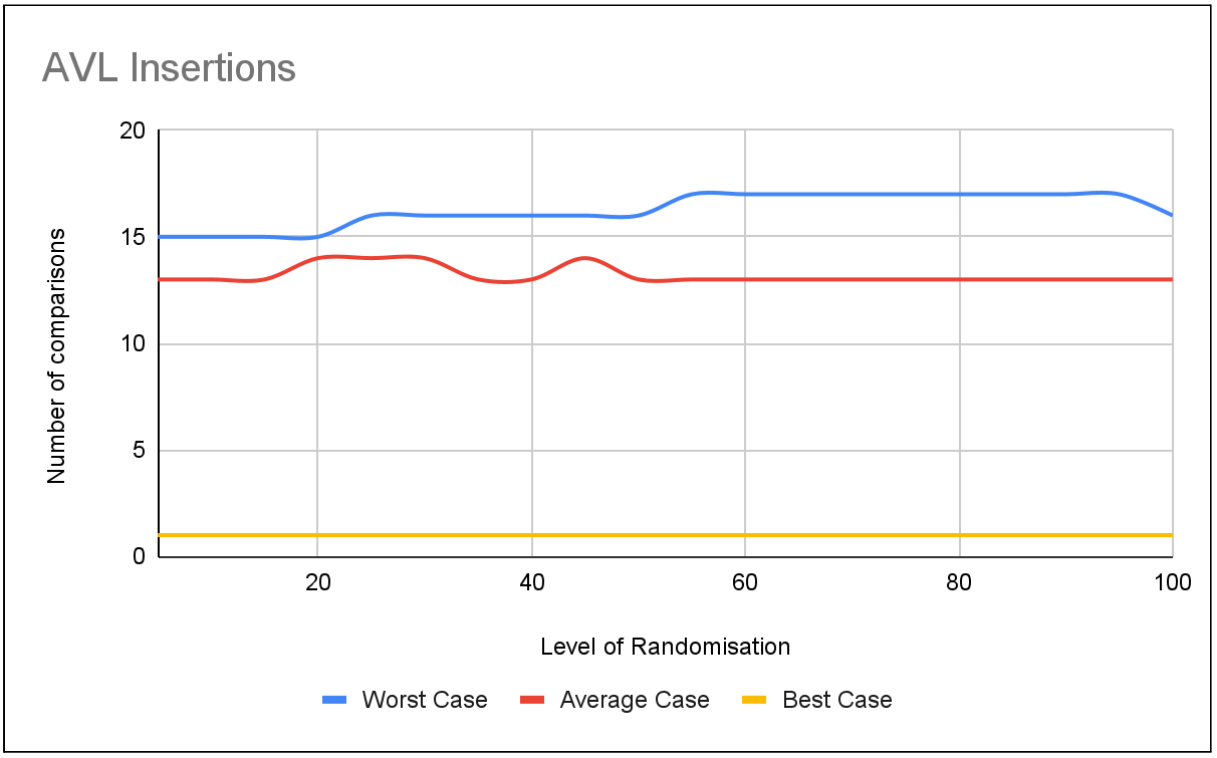
(For creativity, a BST was also used to determine the best, worst and average number of comparisons for searches as the degree of randomization changed).

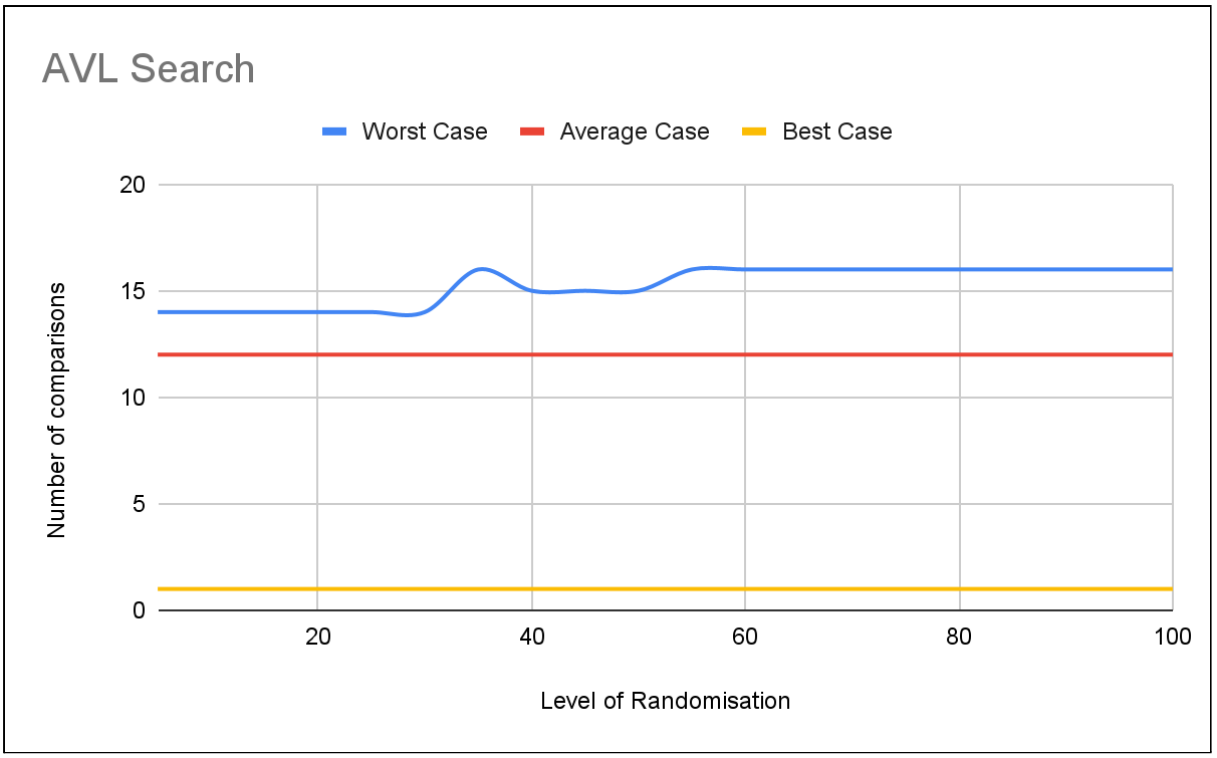
Randomisation Method

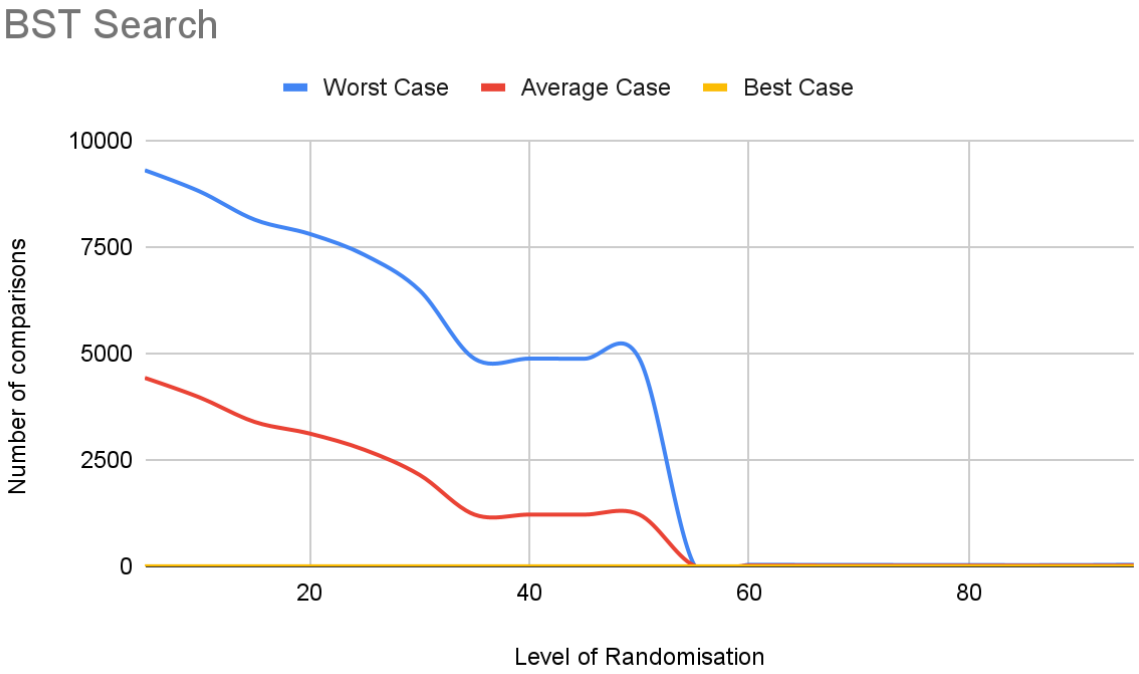
Given a level of randomization x, the first x percent of the dataset was separated from the rest of the data . The selected subsection of data was then randomised using the Collections.shuffle() method which traverses the subsection backwards, from the last element up to the second, repeatedly swapping a randomly selected element into the "current position". Elements are randomly selected from the portion of the subsection that runs from the first element to the current position, inclusive. Once the subsection of the dataset has been randomised it is combined with the rest of the dataset.

This method of randomisation ensures that all permutations of the portion of data to be randomised occur with approximately equal likelihood. Usage of the Collections.shuffle() method also ensures that the randomisation is performed using best/recommended practices with regards to performance and memory optimisation.

Results


AVL Insertions
Number of comparisons
20
15
10
5
0
20
40
60
80
100
Level of Randomisation
Worst Case
Average Case
Best Case


AVL Search
Worst Case
Average Case
Best Case
Number of comparisons
20
15
10
5
0
20
40
60
80
100
Level of Randomisation



See Appendix 1 for tabulated results.

Discussion

For both AVL insertions and searches, the fewest number of comparisons made i.e the best case is 1. The best case scenario for insertion occurs when the AVL Tree is empty. The best case scenario for searches occurs when the root of the tree is the item being searched for.

The worst-case and average-case number of comparisons to perform both searches and insertions does not vary significantly as the degree of randomization changes. This occurs because for both searches and insertions, the maximum number of nodes traversed to either find the location of insertion or to find the element being searched for is guaranteed to be the height of the tree ($\log_2 n$) through the use of rebalancing of the AVL after each insertion. This rebalancing ensures that the difference between the heights of a node's subtrees is no greater than 1 which prevents the formation of a tree that is effectively a Linked List.

The worst-case and average-case number of comparisons for BST searches decreases significantly as the degree of randomization changes. This occurs, because insertion of sequential elements causes an increase in the height of the tree as nodes are appended to one side of a leaf node until the order is broken. For example, for a list of size n sorted in ascending order, all nodes are appended to the right of the last inserted node creating a tree with height n). This increases the number of comparisons to be made before finding a particular element or determining it doesn't exist in the BST.

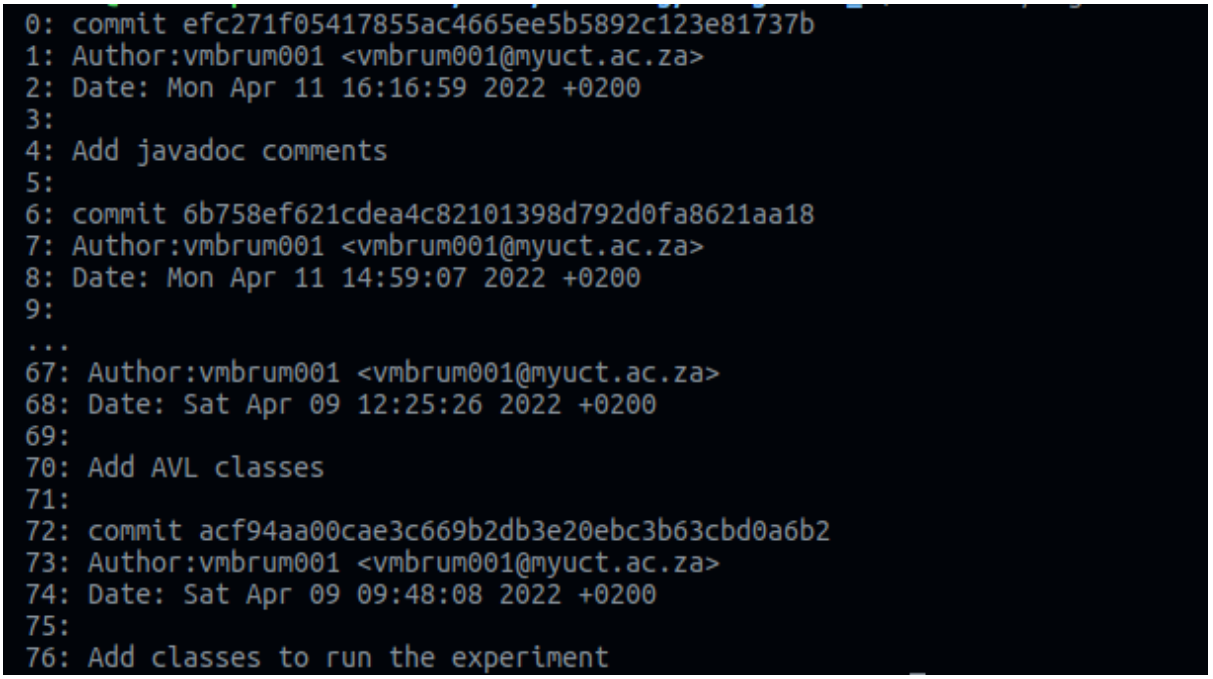
In conclusion an AVL tree guarantees O(logn) performance for both insertions and searches through maintaining balance of the tree using rebalancing.

Creativity

The following items were included to go beyond the basic requirements of the assignment:

1. Search comparisons with a (non-AVL) Binary Search Tree using the VaccineBST class to demonstrate the effectiveness of rebalancing a tree.
2. Experiment class to read the vaccination data, create subsets of data with varying levels of randomization and then run the experiment and record the results in a text file.
3. A make Experiment target to run the experiment.

Git


0: commit efc271f05417855ac4665ee5b5892c123e81737b
1: Author:vmbrum001 <vmbrum001@myuct.ac.za>
2: Date: Mon Apr 11 16:16:59 2022 +0200
3:
4: Add javadoc comments
5:
6: commit 6b758ef621cdea4c82101398d792d0fa8621aa18
7: Author:vmbrum001 <vmbrum001@myuct.ac.za>
8: Date: Mon Apr 11 14:59:07 2022 +0200
9:
...
67: Author:vmbrum001 <vmbrum001@myuct.ac.za>
68: Date: Sat Apr 09 12:25:26 2022 +0200
69:
70: Add AVL classes
71:
72: commit acf94aa00cae3c669b2db3e20ebc3b63cbd0a6b2
73: Author:vmbrum001 <vmbrum001@myuct.ac.za>
74: Date: Sat Apr 09 09:48:08 2022 +0200
75:
76: Add classes to run the experiment

Appendix 1 - Experiment Results

Table 1 - AVL Insertion Comparisons

Level of Randomisation	Worst Case	Average Case	Best Case
5	15	13	1
10	15	13	1
15	15	13	1
20	15	14	1
25	16	14	1
30	16	14	1
35	16	13	1
40	16	13	1
45	16	14	1
50	16	13	1
55	17	13	1
60	17	13	1
65	17	13	1
70	17	13	1
75	17	13	1
80	17	13	1
85	17	13	1
90	17	13	1
95	17	13	1
100	16	13	1

Table 2 - AVL Search Comparisons

Level of Randomisation	Worst Case	Average Case	Best Case
5	14	12	1
10	14	12	1
15	14	12	1
20	14	12	1
25	14	12	1

30	14	12	1
35	16	12	1
40	15	12	1
45	15	12	1
50	15	12	1
55	16	12	1
60	16	12	1
65	16	12	1
70	16	12	1
75	16	12	1
80	16	12	1
85	16	12	1
90	16	12	1
95	16	12	1
100	16	12	1

Table 3 - BST Search Comparisons

Level of Randomisation	Worst Case	Average Case	Best Case
5	9296	4423	1
10	8798	3963	1
15	8136	3389	1
20	7802	3117	1
25	7308	2734	1
30	6479	2147	1
35	4874	1214	1
40	4876	1215	1
45	4873	1214	1
50	4875	1215	1
55	29	15	1
60	36	17	1
65	34	17	1
70	34	16	1
75	30	15	1
80	30	16	1

85	28	15	1
90	30	16	1
95	37	17	1
100	29	15	1