KTHYUS001

Yusuf Kathrada

CSC2001F

Assignment 2 Report

Experiment Description

Aim of the Experiment

The aim of the experiment was to test if AVL trees really do balance nodes and whether they provide good performance regardless of the order in which data is inserted. The experiment was executed through the use of an array which, by default, stored the values of vaccination information in alphabetical order. The user inputs an integer that determines the randomization level which is then inserted into an AVL tree in that order. The experiment compares the level of randomization to the quality of performance, which is quantified by using the number of data comparisons for insertions and searching as indicators of performance.

Outline of Classes Used

The main application of the experiment was the AVLExperiment class. To create this class, I utilized the Vaccine class which I made in Assignment 1. An instance of the Vaccine class is invoked for every line of data inputted, it then splits the line at the commas (",") and stores the country, date and vaccination information. Classes that were given such as BTQueueNode, BTQueue, BinaryTreeNode, BinaryTree, AVLTreeTest and AVLTree were all used and depended on by the AVLExperiment class.

AVLExperiment

The AVLExperiment class begins by making an array of Vaccines which are originally inserted in alphabetical order. The data is read from the vaccination.txt file. The function Randomization takes an integer parameter ranging between 0 and 20 which the user inserts. This determines the level of randomization. The array is shuffled according to the level of randomization and then using iteration, is inserted into an AVL tree. More detail regarding the randomization algorithm will follow. The function randomizedFile outputs the results to a specified file and finds the worst, best and average case. The main method invokes the creation of an AVLExperiment object and allows the user to enter the randomization level. Finally, to get the results of the 20 levels of randomization, a counter-controlled for-loop was used which invoked both functions and printed each result to a file, leaving 20 files from which the results could be extracted from.

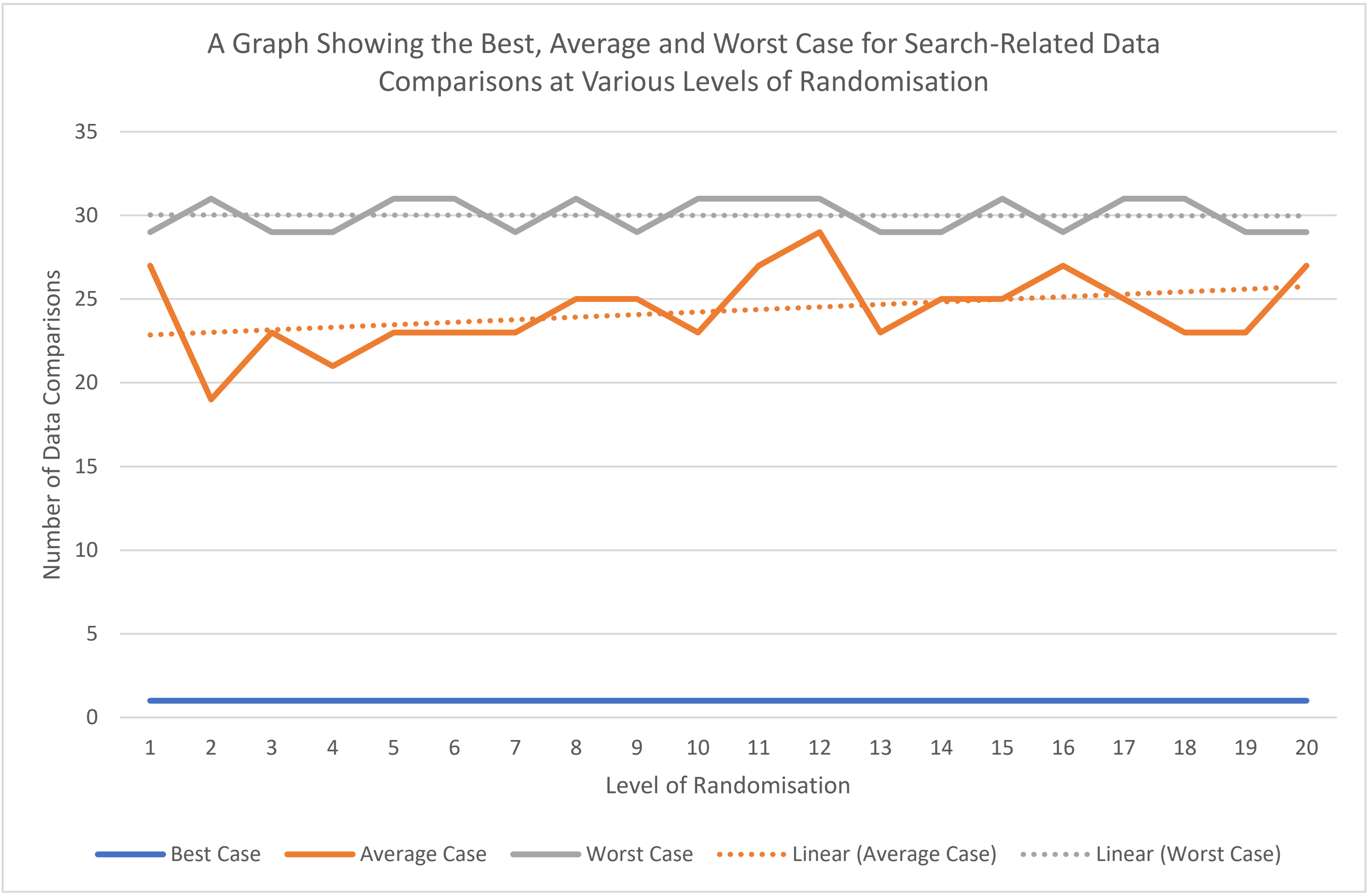
Randomization of Data

Randomizing data can easily be performed using built-in functions however, for this experiment a user inputs an integer which determines the level of randomization. I created a range between 0 and 20 for which the user can input, 0 being complete alphabetical order and 20 being the most randomized. In total there are 9919 inputs of data which I then divided by 20 to get 495 which will be used to create the ranges. The number the user inputs multiplied by 495 will give the range for which data within will be randomized. For example, if the user enters 8, of the total 9919 items of data, data items within the range 0 to 3960 will be randomized and the rest will be in the original alphabetical order.

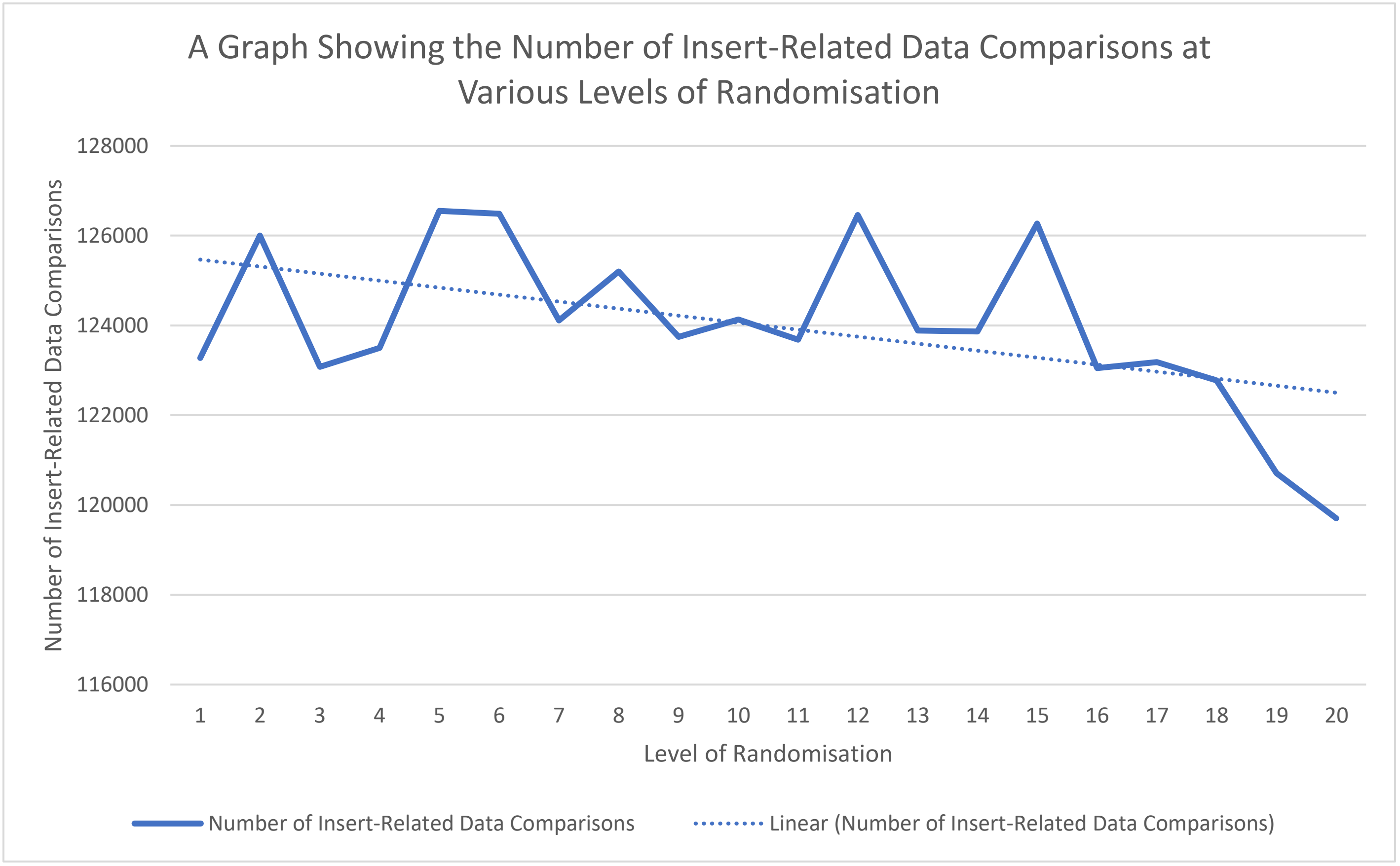
After the ranges were created, a method was needed to actually randomize the data. After doing research I found the Fisher-Yates shuffle array method and adapted it to the experiment's specifications. This method consisted of starting from the last item in the array and swapping it with a random item with the array. This was the most effective randomization algorithm as it created unbiased random permutations as well as having a respectable linear time efficiency.

A Table Showing the Number of search and insert-related data comparisons at various levels of randomization

Level Of Randomization	Number Of Search-Related Data Comparisons			Number Of Insert-Related Data Comparisons
	Best Case	Average Case	Worst Case	
1	1	27	29	123276
2	1	19	31	126000
3	1	23	29	123078
4	1	21	29	123499
5	1	23	31	126552
6	1	23	31	126489
7	1	23	29	124109
8	1	25	31	125203
9	1	25	29	123746
10	1	23	31	124131
11	1	27	31	123684
12	1	29	31	126463
13	1	23	29	123889
14	1	25	29	123867
15	1	25	31	126272
16	1	27	29	123051
17	1	25	31	123185
18	1	23	31	122776
19	1	23	29	120705
20	1	27	29	119704



A Graph Showing the Best, Average and Worst Case for Search-Related Data Comparisons at Various Levels of Randomisation
Number of Data Comparisons
Level of Randomisation
0
5
10
15
20
25
30
35
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Best Case
Average Case
Worst Case
Linear (Average Case)
Linear (Worst Case)



A Graph Showing the Number of Insert-Related Data Comparisons at Various Levels of Randomisation
Number of Insert-Related Data Comparisons
Level of Randomisation
116000
118000
120000
122000
124000
126000
128000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Number of Insert-Related Data Comparisons
Linear (Number of Insert-Related Data Comparisons)

Discussion on Results

Referring to the graph comparing the best, average and worst case for search-related data comparisons at various levels of randomization we are able to see a consistent trend across all levels of randomization. The line of best fit for the worst case is almost constant as it alternates between 29 and 31. For the average case, there is a higher degree of variation between the levels however, with the difference of only 10 between the upper and lower bounds, it still displays a very consistent trend which is illustrated via the line of best fit which is increasing slightly. The best case stays at 1 for all levels which is expected. Looking at the graph comparing the number of insert-related data comparisons at various levels we see a decreasing trend which makes sense as the data becomes less linear and more randomized. However, once again the changes are not drastic.

In conclusion, the consistent trends allude to the fact that AVL trees have the innate ability to balance nodes well irrespective of the order in which data is inserted. There are slight, almost negligible variations in performance which align with the idea that AVL trees produce good, consistent performance despite the order of data insertion.

Creativity

Creativity was displayed in the main method where I created a system which automated the writing of the results to files, all the user enters is the generic file name and it will print all 20 files with the generic name plus the level of randomization next to it for convenience.

Git Log

```
1. yusuf@yusuf-
VirtualBox:~/Documents/CSC2001F/Assignments/Assi
gnment2/data$ git log

commit
78521fe2bb82bbabc04a41bfb61efb8a343c8f8e
(HEAD -> master)

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:27:38 2022 +0200


   "Added Results10-20"


2. commit
311347e5ad075f22d024d0e249a938df7e570930

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:26:41 2022 +0200


   "Added Results9"


3. commit
5a439a23916af67dd279e9231d9d87ff1f1e01dc

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:25:34 2022 +0200


   "Added Results8"


4. commit
3896398e2a3eadf87c11cde842c94373b749bfb8

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:25:18 2022 +0200


   "Added Results7"


5. commit
b4fd4ada4d355334429d5e3e96affdb094f5e1ea

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:24:24 2022 +0200


   "Added Results6"


6. commit
51481b792ca05bd17564fb1377dff002381d5efa

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:23:35 2022 +0200


   "Added Results5"


7. commit
0a62eb2456faa416b60b25788525709912026e68

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:09:39 2022 +0200


   "Added Results4"


8. commit
b4922e1f522e0f8b9b3e53b1dac26a57d0938f09

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 18:09:07 2022 +0200


   "Added Results3"


9. commit
fbc7a7f5c7266faad3e6ee5cbee6c9c9baa49a59

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 17:59:45 2022 +0200


   "Added Results2"


10. commit
90da13b4856d51343ccb5c7c993cf8ad02d53449

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 17:58:53 2022 +0200


   "Added Results 1"


11. commit
885a9544a86a5ff9cf6ecb0519444c65e58f5902

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 17:57:46 2022 +0200


   "Added Random =VaccinationList"


12. commit
f190f83e6abc9e522983ae8efbb1752ed0d11e61

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Thu Mar 24 17:53:56 2022 +0200


   Added Makefile


13. commit
8735a4261bb2a4d33c6b1b1847d3cd9aeeb805d5

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:35:25 2022 +0200


   "Added AVLExperiment which is the class of the
final application"


14. commit
7b9fa45badf31189ac35aef58ab817e575fe0f36

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:34:51 2022 +0200


   "Added AVLTree, from given classes"


15 commit
d3b083140b2ff0d763106e4dad20b2291a23426f

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:34:00 2022 +0200


   "Added AVLTreeTest to test methods and functions
which can be used in final application"


16. commit
2516092d996b3f98da2134339c10fe71739516dc

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:33:09 2022 +0200


   "Added BinaryTree, from given classes"


17. commit
41cddcc1573027377371da7a5770c3dff510c784
```

```
Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:32:01 2022 +0200


    "Added BinaryTreeNode, from given classes"


18. commit
90366dd2e1d1ee52a780db9963de31aa7f3f2930

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:31:04 2022 +0200


    "Added BTQueue, from given classes"


19. commit
f1aa77f88f0cad3bc0ca72988d6ec4615d11dc0c

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:27:01 2022 +0200


    "Added BTQueueNode, a given class"


20. commit
36bd62e77793e217aef938054a487eaefaef1625

Author: Yusuf Kathrada <kthyus001@myuct.ac.za>

Date:   Mon Mar 21 00:25:22 2022 +0200


    "Added Vaccine file from previous Assignment"
```