KTHYUS001

Yusuf Kathrada

CSC2001F

Assignment 1 Report

Description of OO Design

Outline of classes used

To create the VaccineArrayApp, the first of the two applications, I created a Vaccine and a VaccineArray class. The VaccineArrayApp depended on the VaccineArray class and both of the before mentioned classes depended on the Vaccination class.

The second application, VaccineBSTApp, depends on the Entry class I created which is a variation of the Vaccine class used in the first application. The BinaryTreeNode, BTQueueNode, BTQueue, BinaryTree and BinarySearchTree were all given classes which were depended on by the helper Entry class I created as well as the main VaccineBSTApp application. In addition to the core of the java project files, I created an automotive script in python which created subsets for experimenting purposes. Additional information regarding the python script will be addressed later in the report.

How classes Interact

VaccineArrayApp

The VaccineArrayApp begins by creating a VaccineArray and reading through all the lines of a file and stores each line of the file as a Vaccine object in the VaccineArray. This is done by separating each line at the commas which allows the country, date and number of vaccinations to be accessed. An exception will be thrown if the file cannot be found.

After the file has been read, the user interface is invoked. The line “Enter the date:” is displayed and the user will proceed by filling in the date in the form “yyyy-mm-dd” in the line below. Next, the line “Enter the list of countries (end with an empty line):” is displayed and the user will input a list of countries entered below each other and completing their input by leaving an empty line. At this stage, the find method in VaccineArray, which invokes the compareTo method from the Vaccine class, is invoked and countries entered that contain vaccination information on the user-specified date is returned. If no vaccination information is found or the country name does not match any countries in the VaccineArray it will return “<Not Found>”. The methods testData can be invoked and runs a test with a variety of valid and invalid inputs

to test the correctness of the VaccineArrayApp. This alongside the OPPS method, which returns the number of data comparisons taken to find the object, are additional methods which provide further features for the VaccineArrayApp.

VaccineBSTApp

The VaccineBSTApp works similarly to the VaccineArrayApp with slight variations to certain methods but specifically in how the data is stored. The app begins in the same vain as the VaccineArrayApp by reading in the file “vaccinations.txt” which contains the data. However, instead of a VaccineArray being created a BinarySearchTree is made and each line read is converted into an Entry object and gets stored in the BinarySearchTree. This is done by invoking the insert method from the BinarySearchTree class. Next, the user interface is invoked and like in the VaccineArrayApp the date is entered and the countries are listed by the user. Like before, the line is separated at the commas and by using the find method invoked from the BinarySearchTree class the results will be outputted to the user. Like before, if no vaccination data can be found or if the country name entered is not in the data structure, “<Not Found>” will be returned. The class also contains an experiment method which utilizes the automated subsets of inputs to return results to a file as well as testData and an operationsCount which performs as previously discussed.

Both applications read in files and contain an interactive user interface which takes input and displays results. The defining difference between the two applications is the method of data storage. As the names suggest the VaccineArrayApp uses an Array as a data storage structure and VaccineBSTApp uses a binary search tree as a data structure. This leads us to the aim of the experiment.

Experimentation

Aim of experiment

The aim of the experiment was to compare the Binary search tree with a traditional unsorted array data structure which are both implemented in java using an application which obtained daily vaccination information for a list of countries. The comparison looked at the efficiency of the data structures using the amount of data comparisons used to compare key values as a way to quantify the efficiencies. Thus, the independent variable of this experiment would be the data structure used in each application.

Description of experiment

10 subsets of inputs were created of varying lengths which were used to compare the number of data comparisons for each data structure. The subsets were created using an automated python script I created. The same 10 subsets were used for both data structures which makes it one of the many control variables. Other control variables included the fact that both applications were coded in java, both experiments ran on the Unix operating system and on the same laptop set at the same performance settings. The experiment method was invoked for both applications and for each of the 10 subsets of data a corresponding results file was written to.

Testing for both VaccineArrayApp and VaccineBSTApp

Test 1 (**All Valid**):

Input:

Dominica,2022-01-09,101
Armenia,2022-01-09,4895
Sudan,2022-01-09,8915
Jamaica,2022-01-09,3811
Georgia,2022-01-09,5731

Output:

Dominica = 101
Armenia = 4895
Sudan = 8915
Jamaica = 3811
Georgia = 5731

Test 2 (**All Valid**):

Input:

Cape Verde,2022-01-13,2607
Democratic Republic of Congo,2022-01-13,5760
Ethiopia,2022-01-13,401
Senegal,2022-01-13,4472
Canada,2022-01-13,369777
Iceland,2022-01-13,3012
United Kingdom,2022-01-13,216703
Kuwait,2022-01-13,13444
Burkina Faso,2022-01-13,7399
Sint Maarten (Dutch part),2022-01-13,767
Mongolia,2022-01-13,4298
Isle of Man,2022-01-13,275
Vietnam,2022-01-13,1111945
British Virgin Islands,2022-01-13,22
Jordan,2022-01-13,35491

Output:

Cape Verde = 2607
Democratic Republic of Congo = 5760
Ethiopia = 401
Senegal = 4472
Canada = 369777
Iceland = 3012
United Kingdom = 216703
Kuwait = 13444
Burkina Faso = 7399
Sint Maarten (Dutch part) = 767
Mongolia = 4298
Isle of Man = 275
Vietnam = 1111945
British Virgin Islands = 22
Jordan = 35491

Test 3 (**Invalid Inputs**):

Input:

The Will to Live,2022-04,16
Time to relax,2022-04-17
Sleep,2022-04-17,18

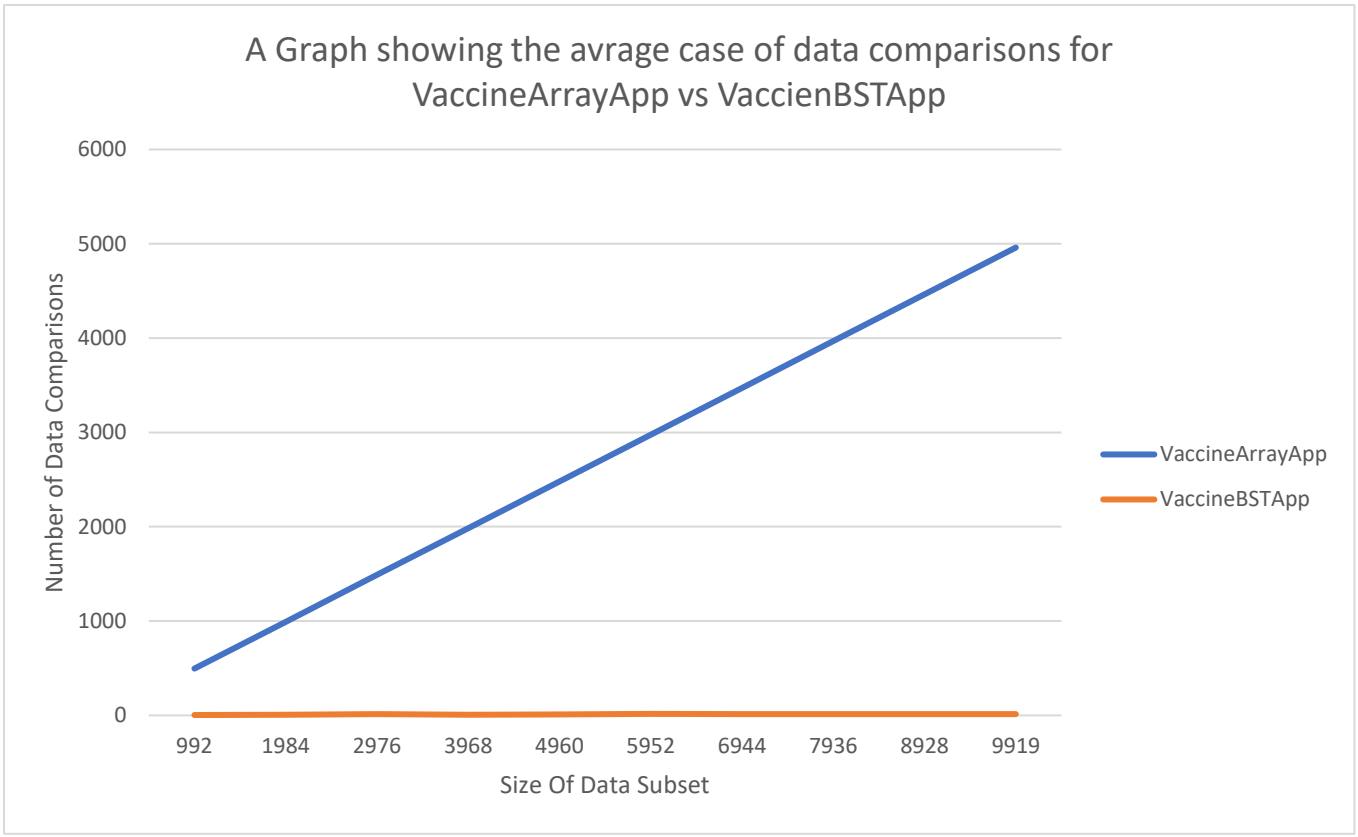
Output:

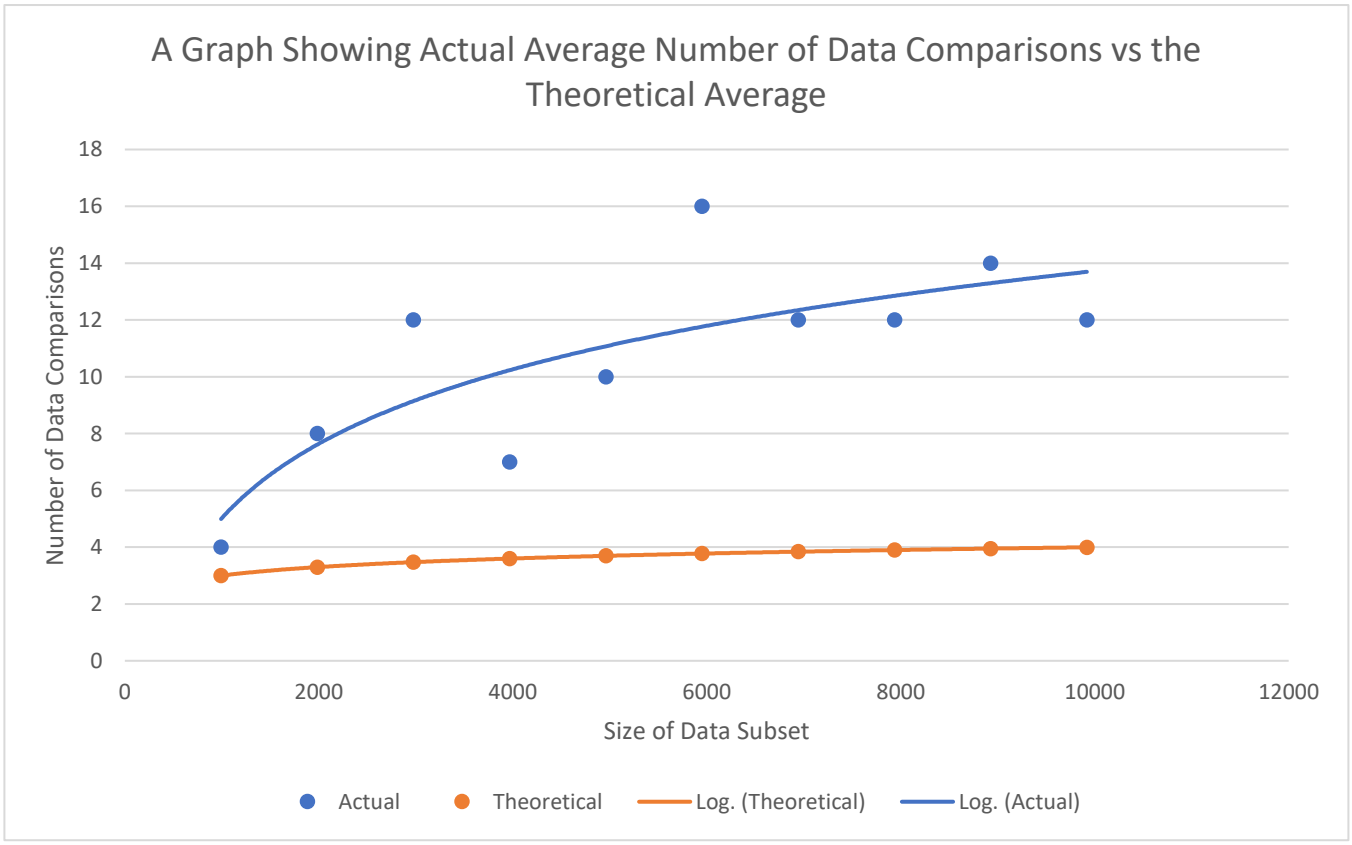
The Will to Live = <Not Found>
Time to relax = <Not Found>
Sleep = <Not Found>

Results

Table 1: A table showing the best, average and worst case of data comparisons for VaccineArrayApp

Size of Data Subset	Best Case		Average Case		Worst Case	
	VaccineArrayApp	VaccineBSTApp	VaccineArrayApp	VaccineBSTApp	VaccineArrayApp	VaccineBSTApp
992	1	1	496	4	992	8
1984	1	1	992	8	1984	15
2976	1	1	1488	12	2976	16
3968	1	1	1984	7	3968	12
4960	1	1	2480	10	4960	12
5952	1	1	2976	16	5952	18
6944	1	1	3472	12	6944	24
7936	1	1	3968	12	7936	27
8928	1	1	4464	14	8928	21
9919	1	1	4960	12	9919	25


A Graph showing the avrage case of data comparisons for VaccineArrayApp vs VaccienBSTApp
Number of Data Comparisons
6000
5000
4000
3000
2000
1000
0
992
1984
2976
3968
4960
5952
6944
7936
8928
9919
Size Of Data Subset
VaccineArrayApp
VaccineBSTApp


A Graph Showing Actual Average Number of Data Comparisons vs the Theoretical Average
Number of Data Comparisons
18
16
14
12
10
8
6
4
2
0
0
2000
4000
6000
8000
10000
12000
Size of Data Subset
Actual
Theoretical
Log. (Theoretical)
Log. (Actual)

Discussion on Results

When comparing the two applications I will refer to them by the data structures they employ for ease of use .i.e., VaccineArrayApp will be referred to as an Array and VaccineBSTApp as a BST. The above graph shows the linear behavior of an array which is as expected. Conceptually it is sound because when searching through an array, on average, the number of data comparisons would be half of the size of the data set. Therefore O(n) average time complexity as the coefficient of ½ is negligible. The BST on the other hand displays a very different image.

Due to the high number of comparisons the shape of the BST graph I created a second graph which compares the average number of comparisons to the theoretical average. This is confirmed by the fact that the average time complexity of a BST being O(log n). This proves that the Binary Search Tree is more efficient than the traditional unsorted array for the amount of data comparisons for searching. However, the disadvantage of the BST can be seen in the insertion portion as the average case of insertion being O(log n) as opposed to the more efficient Array having an average case of 0 as no comparison needs to take place as it is guaranteed that the next item added to an array get appended to the end.

Creativity

Creativity was displayed through the use of an automated python script which created 10 unique subsets of varied amounts of data which was used as input for the experimentation part of the assignment. This was not necessary, but it served as an interesting addition to the project as it allowed for two different programming languages to interact alongside it being a more efficient method of creating the data subsets. Other flairs of creativity can be seen in the additional methods created which provides more capabilities for the user to gain information on the applications. The testData method allows the user to input any name of a text file with the contents of the file to be used as input for testing and the outputs will be written to a file. Creativity in the Makefile came in the form of automating the generation of the javadocs as well as creating two run commands, one for each application.

Conclusion

In brief, we are able to conclude that the binary search tree data structure is far more efficient at searching than the unsorted array. However, the unsorted array is more efficient at insertion as no data comparisons need to take place unlike the binary search tree which has to make comparisons. This was demonstrated through the use of two applications and comparing the number of data comparisons of each.

Git Log

1) yusuf@yusuf-VirtualBox:~/CSC2001F/Assignments/Assignment 1/src$ git log

commit b6b3d46a15d9a22f8a12a4723a14102fc743a6ca (HEAD -> master)

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

Date: Mon Mar 7 23:56:46 2022 +0200

Added javadocs and comments

2) commit bea183ad4169967d4a1b07c064b7b0c347b0bbb0

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

Date: Sun Mar 6 22:42:12 2022 +0200

Added clearOpCount method

3) commit 6e60174bd5e63e1b89e5c6596f93450e4000dd3e

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

Date: Sun Mar 6 22:39:58 2022 +0200

Added method to automate experiment

4) commit b7984739d9f07a7e18b195260d1c56ad925629d2

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

Date: Sun Mar 6 21:30:49 2022 +0200

Added method to write results to file

5) commit 60a518da8308188bdf1e06df2b296d106a13b17c

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

Date: Sun Mar 6 15:21:47 2022 +0200

Added VaccineBSTApp

6) commit 7c3df7980b148dc171f1957870d6862914c62c9b

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

Date: Sun Mar 6 15:09:35 2022 +0200

Added Entry

7) commit 153297bef520369ae5d0547d393dc2270ea57876

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

Date: Sun Mar 6 15:06:38 2022 +0200

Added BinarySearchTree

8) commit 81d288847df629afe6ac87ee2980100748616522

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

Date: Sun Mar 6 15:06:06 2022 +0200

Added BinaryTree

9) commit b11fbce4010b4321a98c293df059f1804ae9c798

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

Date: Sun Mar 6 15:05:22 2022 +0200

Added BinaryTreeNode

10) commit 78192b41849ea03182182e0aa89f0e1c335a968e

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

Date: Sun Mar 6 15:04:45 2022 +0200

Added BTQueue

//--

....

//--

16) commit 6e60174bd5e63e1b89e5c6596f93450e4000dd3e

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

Date: Sun Mar 6 22:39:58 2022 +0200

Added method to automate experiment

17) commit b7984739d9f07a7e18b195260d1c56ad925629d2

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

Date: Sun Mar 6 21:30:49 2022 +0200

Added method to write results to file

18) commit 60a518da8308188bdf1e06df2b296d106a13b17c

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

Date: Sun Mar 6 15:21:47 2022 +0200

Added VaccineBSTApp

19) commit 7c3df7980b148dc171f1957870d6862914c62c9b

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

Date: Sun Mar 6 15:09:35 2022 +0200

Added Entry

20) commit 153297bef520369ae5d0547d393dc2270ea57876

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

Date: Sun Mar 6 15:06:38 2022 +0200

Added BinarySearchTree

21) commit 81d288847df629afe6ac87ee2980100748616522

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

Date: Sun Mar 6 15:06:06 2022 +0200

Added BinaryTree

22) commit b11fbce4010b4321a98c293df059f1804ae9c798

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

Date: Sun Mar 6 15:05:22 2022 +0200

Added BinaryTreeNode

23) commit 78192b41849ea03182182e0aa89f0e1c335a968e

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

Date: Sun Mar 6 15:04:45 2022 +0200

Added BTQueue

24) commit 6b31f4db6f8a7f4b344ea51cabee6c216405bcc9

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

Date: Sun Mar 6 15:03:03 2022 +0200

Added BTQueueNode

25) commit e5e60e2ba9873a3e6be038d11b299eb01a465f89

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

Date: Sun Mar 6 14:59:17 2022 +0200

Added VaccineArrayApp

26) commit a5fcdd49723bab8b65ca5020a340277177d2b0fa

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

Date: Sun Mar 6 14:49:43 2022 +0200

Completed Vaccine and VaccineArray