Assignment 1 report KPPDEA001 7 March 2022

OO design:

For my VaccineArrayApp I have created three classes to complete the required task.

The Vaccine class takes in a comma separated string of the consisting of a country, a date and a number of vaccinations. The constructor method splits these strings into an array and stores them in variables. This class also contains accessor methods for the three variables and a compareTo method.

The VaccineArray class is a class which is used for storing type Vaccine objects. The class can add records and contains a method to return the vaccinations when receiving a country along with a date.

The VaccineArrayApp class is the main class. This class reads the vaccination csv file, records each line as a type Vaccine and stores it in a VaccineArray object. The class provides an interface to the user in which it takes input of a date and countries in order to find the number of vaccinations that corresponds to each country and print it to the screen.

My VaccineBSTApp uses 3 classes in order to complete the required task.

The VaccineBSTApp makes use of the Vaccine class in the same way as the VaccineArrayApp. In place of the VaccineArray class I have used a BinarySearchTree class to store the vaccination records.

Experiment:

The goal of the experiment was to determine the best case, worst case and average case in terms of operations when inserting data as well as finding specific data in the data structures. The experiment compares the cases for these two operations for varying amounts of data and looks at the use of an array in comparison to a binary search tree.

To execute this experiment, I created an experiment method in both the VaccineArrayApp and VaccineBSTApp classes. The methods use the ReadFile methods to read in the vaccinations csv file and then makes use of nested for loops in order to add a portion of the data to the data structures at a time. As the new data is added and the number of records increase, the inner for loop performs the insert and find operations on every single piece of data. The number of operations used to perform the inserts and finds for each data item are only recorded when they are either a new best case or a new worst case and are stored in variables.

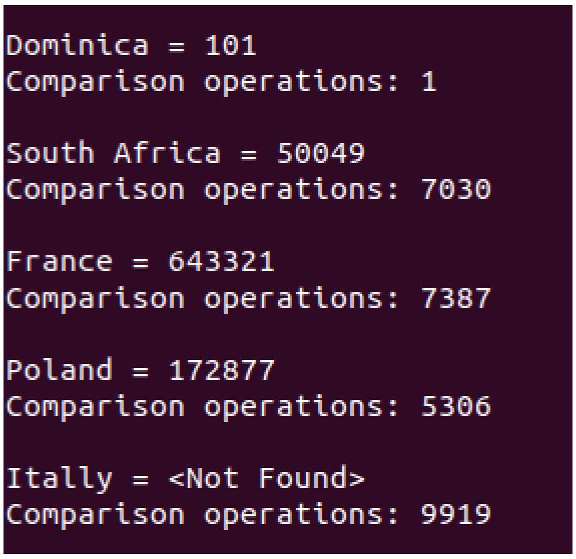
In each iteration of the outer loop, the method prints these results to its own csv file named ‘Cases’ in order to be able to read and compare the results.

Test values for trial runs:

Values for date: 2022-01-09

Countries: Dominica, South Africa, France, Poland, Itally(purposefully spelt incorrectly),

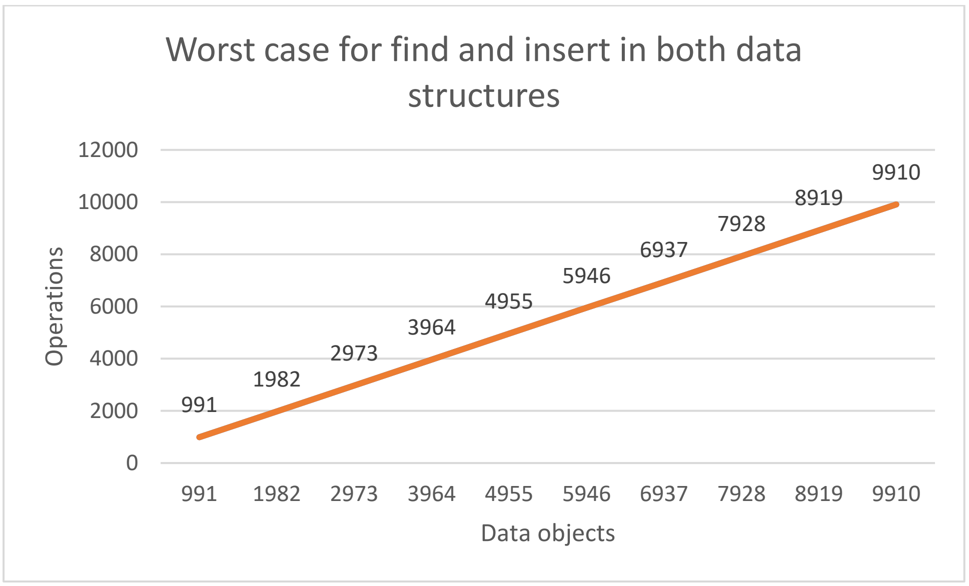
Output:

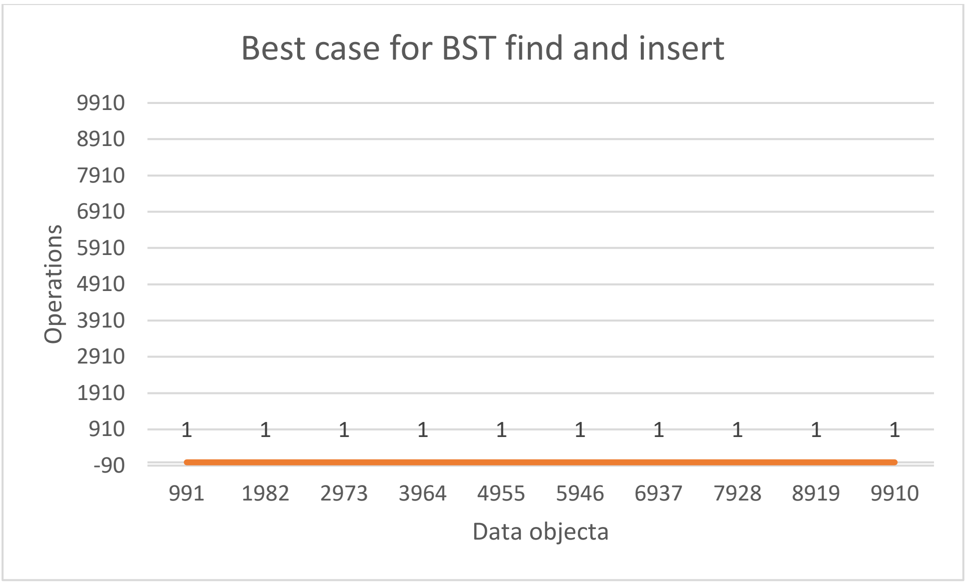


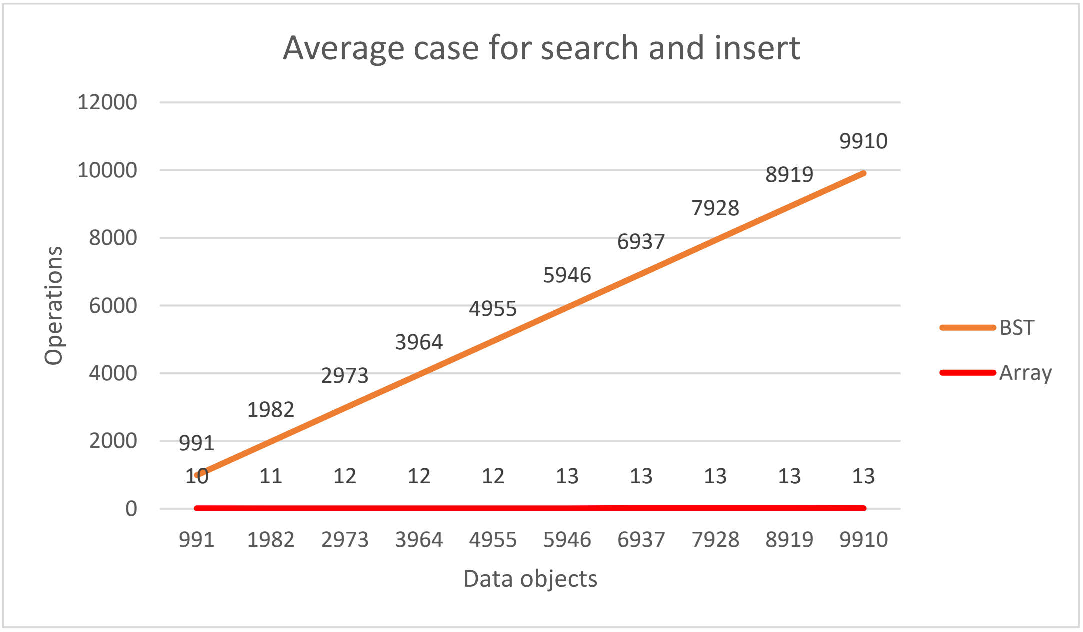
```
Dominica = 101
Comparison operations: 1

South Africa = 50049
Comparison operations: 7030

France = 643321
Comparison operations: 7387

Poland = 172877
Comparison operations: 5306

Itally = <Not Found>
Comparison operations: 9919
```

Outputs identical for both VaccineBSTApp and VaccineArrayApp.

Final Results:







The above graphs show the best, worst and average case time complexities for the binary search tree search and insert used in the VaccineBSTApp. As seen above, the best case for searching in a binary search tree will always be O(1) regardless of the size of the tree. This occurs when the desired data happens to be in the root node and only one comparison is needed. O(1) is also the best case for insertion.

The worst case for the binary tree search will always be O(n). This means that the more elements are in the tree, the worse the worst case will be. This is the same for insertion.

The average case for the binary tree search and insert is O(log n). This is shown by the graph which depicts a concave down line which flattens as the data objects increases.

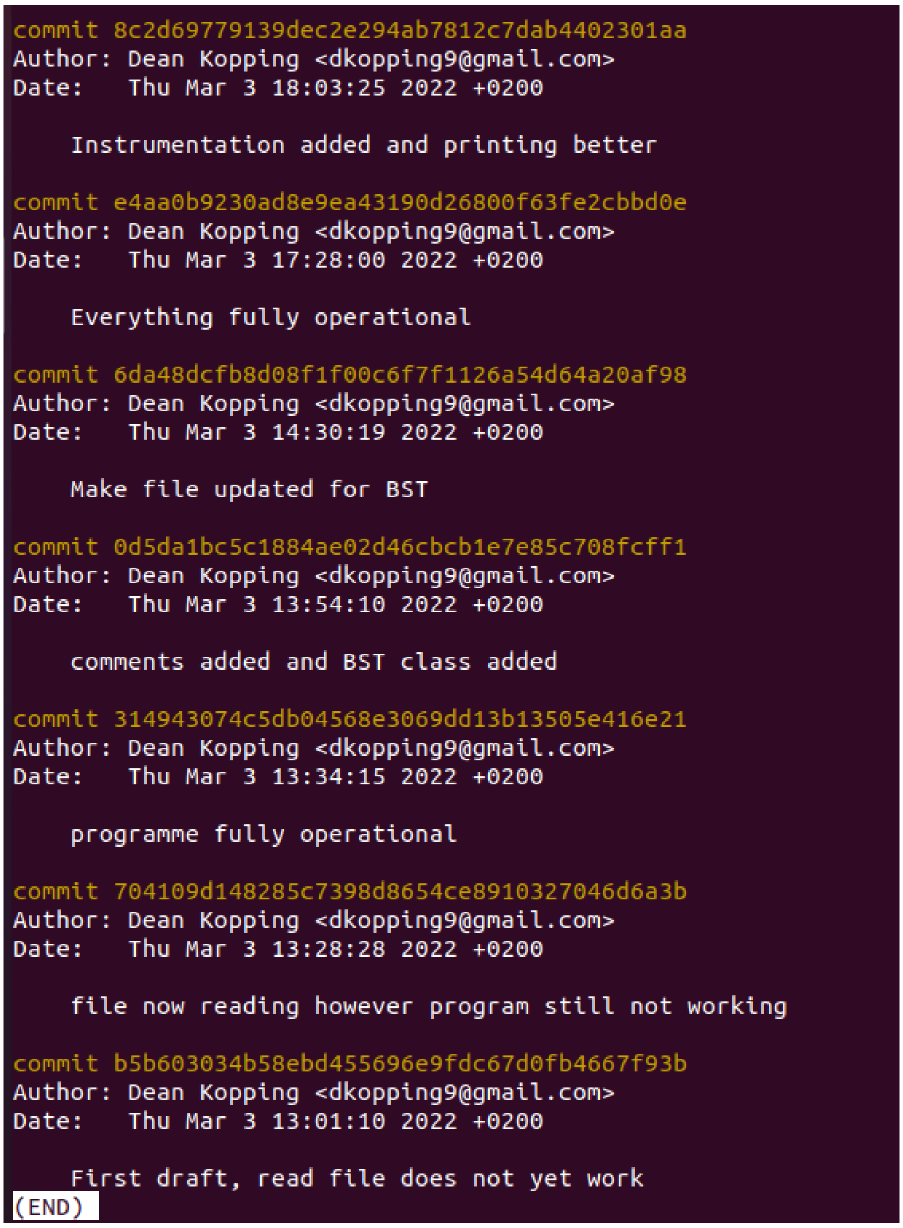
For the array, the worst case time complexity for both insert and search is O(n). Therefor the more data that is put into the array, the worse the worst case will become. The best case for these operations is O(0) which is slightly better then the binary search tree.

The main advantage in using a binary search tree is shown in the average case for these operations. Where for a binary search tree the time complexity would be O(log n), the array would be O(n) which is significantly less efficient. This is shown in graph 3.

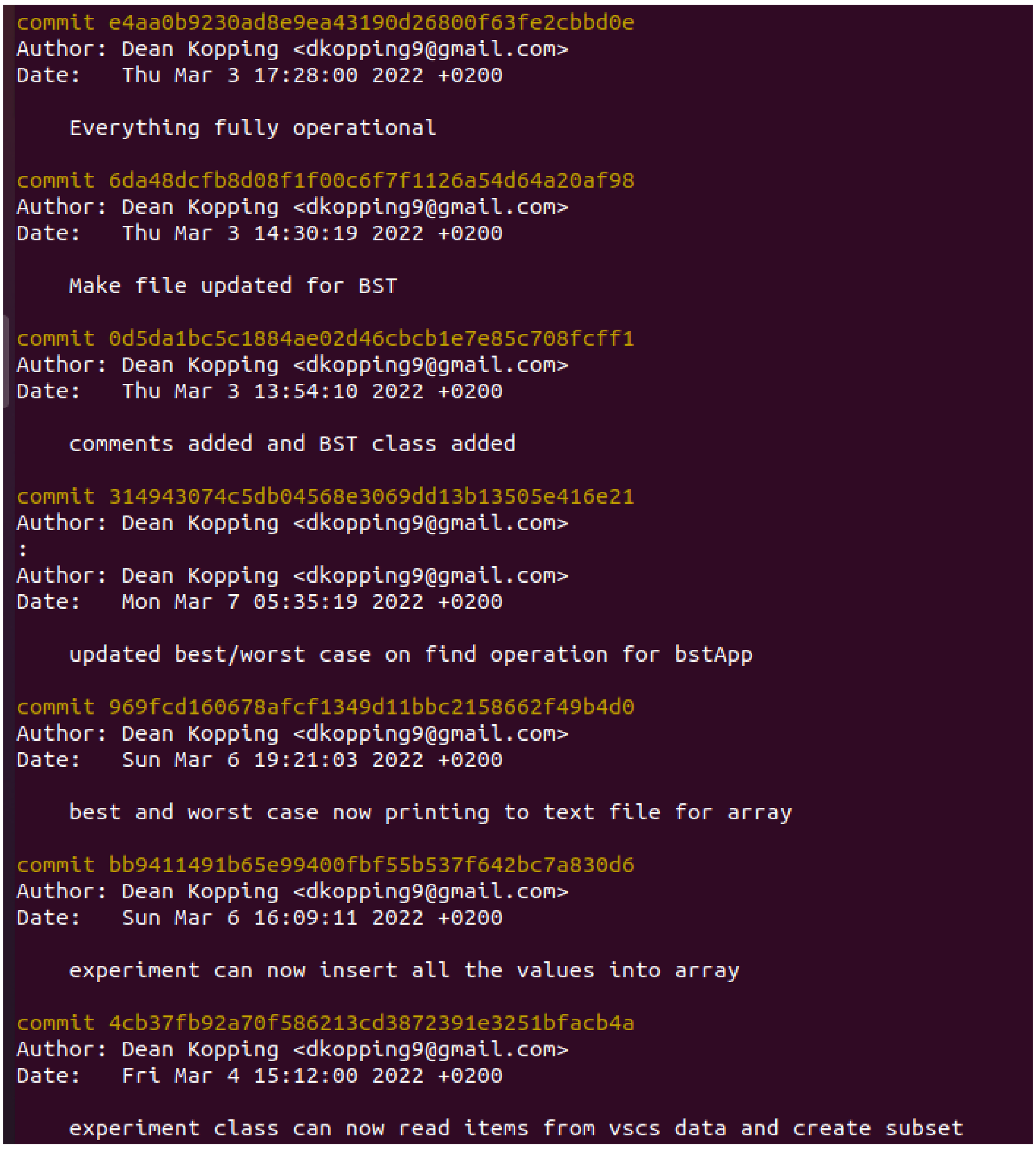
Creativity:

In order to make the assignment into one complete, simple and compact file, I have attempted to code the experiment into both the VaccineArrayApp and the VaccineBSTApp as their own methods. The user is prompted at the start of each application to select if they would like to run the user interface or to run the time complexity experiment. Instead of creating multiple csv files, I have used one that has updated itself with more vaccine data from the given csv file in a loop, the results of the experiment are printed concisely to their own csv file for easy reading.

Summary Statistics:



```
commit 8c2d69779139dec2e294ab7812c7dab4402301aa
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 18:03:25 2022 +0200

    Instrumentation added and printing better

commit e4aa0b9230ad8e9ea43190d26800f63fe2cbbd0e
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 17:28:00 2022 +0200

    Everything fully operational

commit 6da48dcfb8d08f1f00c6f7f1126a54d64a20af98
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 14:30:19 2022 +0200

    Make file updated for BST

commit 0d5da1bc5c1884ae02d46cbcb1e7e85c708fcff1
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 13:54:10 2022 +0200

    comments added and BST class added

commit 314943074c5db04568e3069dd13b13505e416e21
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 13:34:15 2022 +0200

    programme fully operational

commit 704109d148285c7398d8654ce8910327046d6a3b
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 13:28:28 2022 +0200

    file now reading however program still not working

commit b5b603034b58ebd455696e9fdc67d0fb4667f93b
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 13:01:10 2022 +0200

    First draft, read file does not yet work
(END)
```



```
commit e4aa0b9230ad8e9ea43190d26800f63fe2cbbd0e
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 17:28:00 2022 +0200

    Everything fully operational

commit 6da48dcfb8d08f1f00c6f7f1126a54d64a20af98
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 14:30:19 2022 +0200

    Make file updated for BST

commit 0d5da1bc5c1884ae02d46cbcb1e7e85c708fcff1
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Thu Mar 3 13:54:10 2022 +0200

    comments added and BST class added

commit 314943074c5db04568e3069dd13b13505e416e21
Author: Dean Kopping <dkopping9@gmail.com>
:
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Mon Mar 7 05:35:19 2022 +0200

    updated best/worst case on find operation for bstApp

commit 969fcd160678afcf1349d11bbc2158662f49b4d0
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Sun Mar 6 19:21:03 2022 +0200

    best and worst case now printing to text file for array

commit bb9411491b65e99400fbf55b537f642bc7a830d6
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Sun Mar 6 16:09:11 2022 +0200

    experiment can now insert all the values into array

commit 4cb37fb92a70f586213cd3872391e3251bfacb4a
Author: Dean Kopping <dkopping9@gmail.com>
Date:   Fri Mar 4 15:12:00 2022 +0200

    experiment class can now read items from vscs data and create subset
```