

I approached this assignment with the following OO design in mind:

### ***For LSArrayApp***

The LSArrayApp is the driver class for LSArray, and does nothing other than start the program given certain inputs from the command line. The LSArray and LSData classes are the two main classes used in the application. LSArray includes a method called fileIn() which starts by creating an array of type LSData. After that, the fileIn() method loops through the dataset line by line. Every line is split into two objects of the LSData class, namely value1 and value2. These represent the parameters i.e (stage\_day\_startTime) and the areas respectively. The relevant index of each pair of objects is stored in the fixed array, using setValue() methods from the LSData class. The printAllAreas() method is also included in LSArray. This method iterates and prints the entire array's area values. I was slightly unsure whether or not to include the key in this method, as the question was slightly ambiguous. However, for the purposes of testing efficiency, printing the whole array would not decrease efficiency so the net outcome is the same. The printAreas() method takes in 3 arguments from the command line. The query is concatenated and stored in a string variable. The array is then searched with a for loop to find a matching key by using a built in contains method. Once the value is found, the corresponding area value is printed. I included an operation count variable, opCount(), to increment every iteration of the loop. A method called getOpCount() is then used to obtain the int value of the opCount variable. This value is sent to a writeToFile method which prints the value of the opCount variable to a file called opCount.txt in the main directory. This is the high level overview of LSArrayApp.

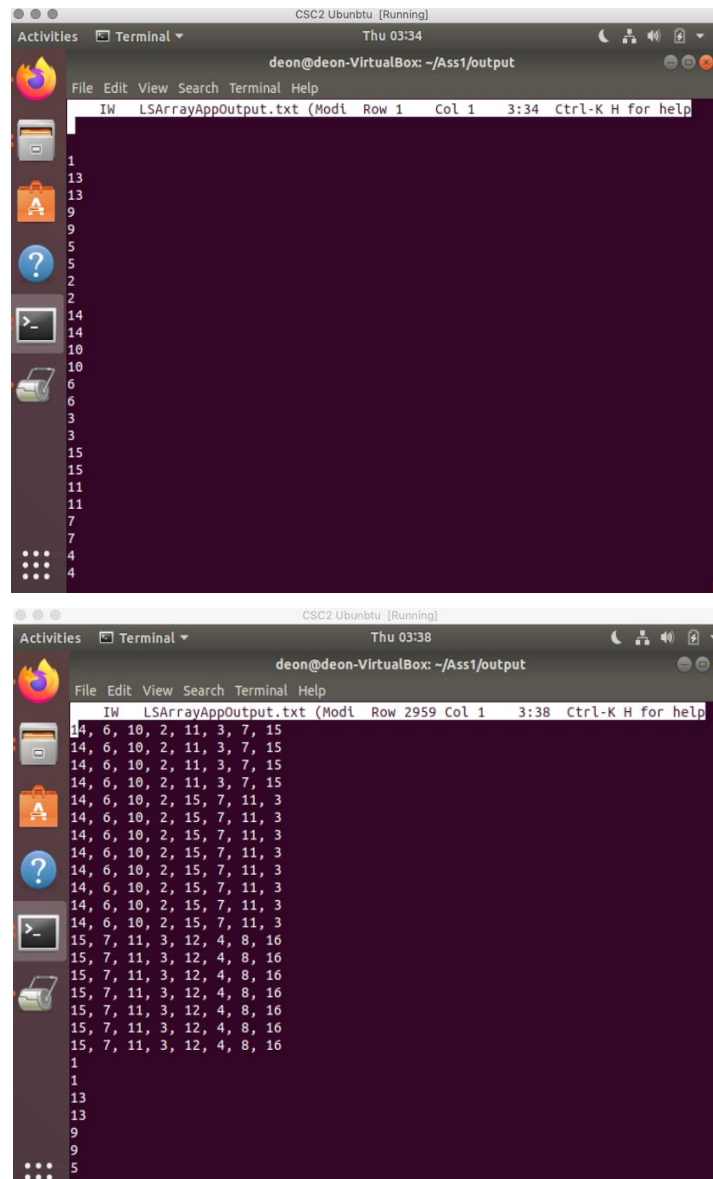
### ***For LSBSTApp***

I was quite overwhelmed by Prof Hussein's BST file given to us. I was still rusty with generics, so I decided to make my own BST. The tree is a simple Binary Search Tree class with the same printAreas and printAllAreas methods as in LSArrayApp. There is a node class called Node, and a tree class called Tree. These are the basic objects and methods used by BinarySearchTree to create a BST from the dataset. Just as in the LSArray class, a method called add() reads in the file and separates the dataset keys and values. These values are then inserted into the tree. Variables countInsert and countFind are added to keep track of the amount of operations needed for inserting and finding nodes. These variables are sent to .txt files in the main directory by methods writeToFileFind and writeToFileInsert

The goal of this experiment was to compare the efficiency of searching and creating a Binary Search Tree with a standard linear array. The main objective was to compare the amount of comparison operations required for exactly the same inputs to both applications. This way we could actually see how these data structures behave in a real world example. Tests were conducted at various sizes of a randomised subset of the given dataset from  $n = 297$ ,  $594$ , ...,  $n = 2976$ . At these intervals, the average, best and worst case of both algorithms can be compared rigorously

## Part 2 Test Values:

For just running LSArraryApp without parameters, I received the following output every time:



```
deon@deon-VirtualBox: ~/Ass1/output
IW LSArraryAppOutput.txt (Modi Row 1 Col 1 3:34 Ctrl-K H for help)
1
13
13
9
9
5
5
2
2
14
14
10
10
6
6
3
3
15
15
11
11
7
7
4
4

deon@deon-VirtualBox: ~/Ass1/output
IW LSArraryAppOutput.txt (Modi Row 2959 Col 1 3:38 Ctrl-K H for help)
14, 6, 10, 2, 11, 3, 7, 15
14, 6, 10, 2, 11, 3, 7, 15
14, 6, 10, 2, 11, 3, 7, 15
14, 6, 10, 2, 11, 3, 7, 15
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
14, 6, 10, 2, 15, 7, 11, 3
15, 7, 11, 3, 12, 4, 8, 16
15, 7, 11, 3, 12, 4, 8, 16
15, 7, 11, 3, 12, 4, 8, 16
15, 7, 11, 3, 12, 4, 8, 16
15, 7, 11, 3, 12, 4, 8, 16
15, 7, 11, 3, 12, 4, 8, 16
1
1
13
13
9
9
5
5
```

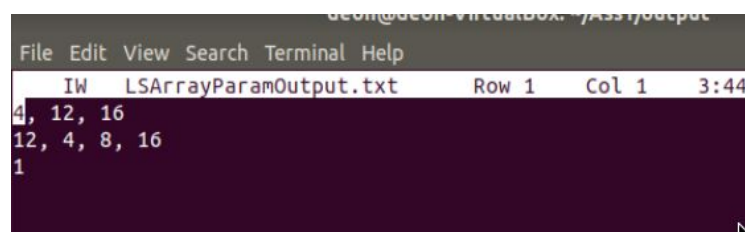
For the valid parameters, LSArraryApp produced the list of areas given the key.

For inputs: java LSArraryApp "3" "30" "08"

java LSArraryApp "4" "29" "16"

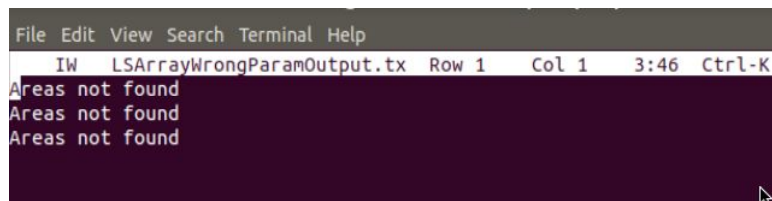
java LSArraryApp "1" "1" "00"

The following outputs were recorded:



```
deon@deon-VirtualBox: ~/Ass1/output
File Edit View Search Terminal Help
IW LSArraryParamOutput.txt Row 1 Col 1 3:44
4, 12, 16
12, 4, 8, 16
1
```

And finally for invalid parameters:

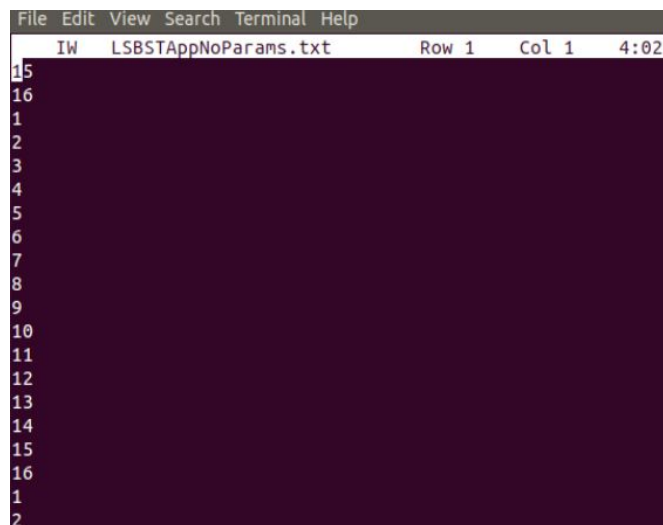


```
File Edit View Search Terminal Help
IW LSArrayWrongParamOutput.tx Row 1 Col 1 3:46 Ctrl-K
Areas not found
Areas not found
Areas not found
```

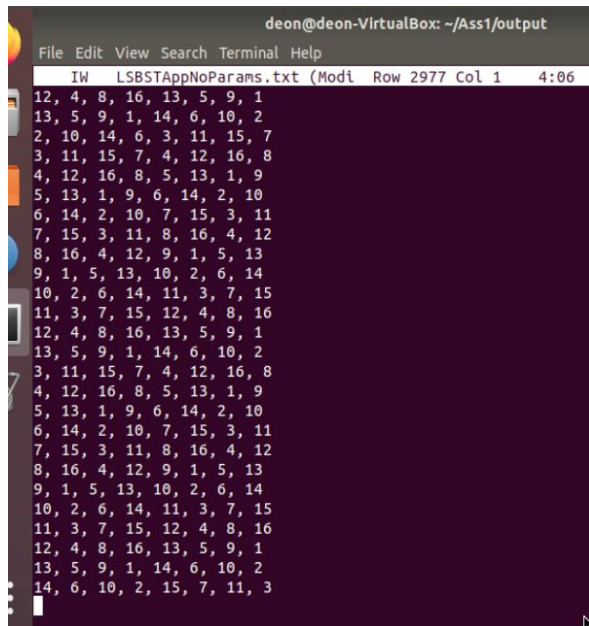
The operation counts in each case turned out to perfectly resemble what the best and worst case of linear search is:  $O(n)$

In each case, the opCount.txt file reports the number of operations performed. For LSArrayApp command without parameters, there are 0 comparisons being made. For valid parameters, the value of opCount.txt was exactly the position of the item in the array. And lastly, the worst case (incorrect parameters) resulted in 2976 comparisons being made.

Similarly, for LSBSTApp without parameters (1st 10 values):

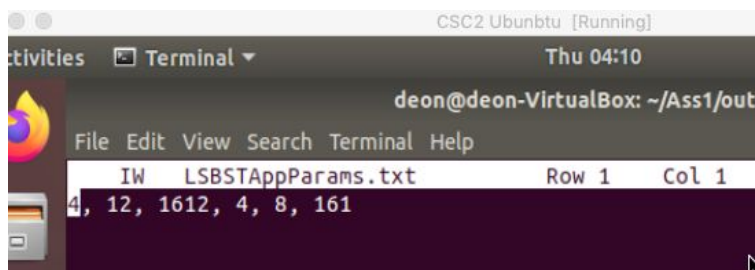


```
File Edit View Search Terminal Help
IW LSBSTAppNoParams.txt Row 1 Col 1 4:02
15
16
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
1
2
```

A terminal window titled 'deon@deon-VirtualBox: ~/Ass1/output' showing the output of a program. The output is a list of 14 rows, each containing a sequence of numbers. The first row is '12, 4, 8, 16, 13, 5, 9, 1'. The second row is '13, 5, 9, 1, 14, 6, 10, 2'. The third row is '2, 10, 14, 6, 3, 11, 15, 7'. The fourth row is '3, 11, 15, 7, 4, 12, 16, 8'. The fifth row is '4, 12, 16, 8, 5, 13, 1, 9'. The sixth row is '5, 13, 1, 9, 6, 14, 2, 10'. The seventh row is '6, 14, 2, 10, 7, 15, 3, 11'. The eighth row is '7, 15, 3, 11, 8, 16, 4, 12'. The ninth row is '8, 16, 4, 12, 9, 1, 5, 13'. The tenth row is '9, 1, 5, 13, 10, 2, 6, 14'. The eleventh row is '10, 2, 6, 14, 11, 3, 7, 15'. The twelfth row is '11, 3, 7, 15, 12, 4, 8, 16'. The thirteenth row is '12, 4, 8, 16, 13, 5, 9, 1'. The fourteenth row is '13, 5, 9, 1, 14, 6, 10, 2'. The fifteenth row is '14, 6, 10, 2, 15, 7, 11, 3'.

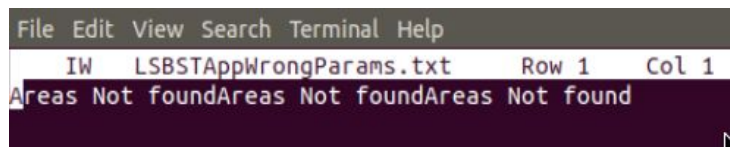
```
deon@deon-VirtualBox: ~/Ass1/output
File Edit View Search Terminal Help
IW LSBSTAppNoParams.txt (Modi Row 2977 Col 1 4:06
12, 4, 8, 16, 13, 5, 9, 1
13, 5, 9, 1, 14, 6, 10, 2
2, 10, 14, 6, 3, 11, 15, 7
3, 11, 15, 7, 4, 12, 16, 8
4, 12, 16, 8, 5, 13, 1, 9
5, 13, 1, 9, 6, 14, 2, 10
6, 14, 2, 10, 7, 15, 3, 11
7, 15, 3, 11, 8, 16, 4, 12
8, 16, 4, 12, 9, 1, 5, 13
9, 1, 5, 13, 10, 2, 6, 14
10, 2, 6, 14, 11, 3, 7, 15
11, 3, 7, 15, 12, 4, 8, 16
12, 4, 8, 16, 13, 5, 9, 1
13, 5, 9, 1, 14, 6, 10, 2
14, 6, 10, 2, 15, 7, 11, 3
```

For the same arguments given to the LSBSTApp: \* for some reason output redirection started every output on the same line

A terminal window titled 'CSC2 Ubuntu [Running]' showing the output of a program. The output is a list of 14 rows, each containing a sequence of numbers. The first row is '4, 12, 16, 12, 4, 8, 16, 1'. The second row is '12, 4, 8, 16, 13, 5, 9, 1'. The third row is '13, 5, 9, 1, 14, 6, 10, 2'. The fourth row is '2, 10, 14, 6, 3, 11, 15, 7'. The fifth row is '3, 11, 15, 7, 4, 12, 16, 8'. The sixth row is '4, 12, 16, 8, 5, 13, 1, 9'. The seventh row is '5, 13, 1, 9, 6, 14, 2, 10'. The eighth row is '6, 14, 2, 10, 7, 15, 3, 11'. The ninth row is '7, 15, 3, 11, 8, 16, 4, 12'. The tenth row is '8, 16, 4, 12, 9, 1, 5, 13'. The eleventh row is '9, 1, 5, 13, 10, 2, 6, 14'. The twelfth row is '10, 2, 6, 14, 11, 3, 7, 15'. The thirteenth row is '11, 3, 7, 15, 12, 4, 8, 16'. The fourteenth row is '12, 4, 8, 16, 13, 5, 9, 1'. The fifteenth row is '13, 5, 9, 1, 14, 6, 10, 2'. The sixteenth row is '14, 6, 10, 2, 15, 7, 11, 3'.

```
CSC2 Ubuntu [Running]
activities Terminal Thu 04:10
deon@deon-VirtualBox: ~/Ass1/outp
File Edit View Search Terminal Help
IW LSBSTAppParams.txt Row 1 Col 1
4, 12, 1612, 4, 8, 161
```

And finally, invalid parameters:

A terminal window showing the output of a program. The output is a single line: 'Areas Not foundAreas Not foundAreas Not found'.

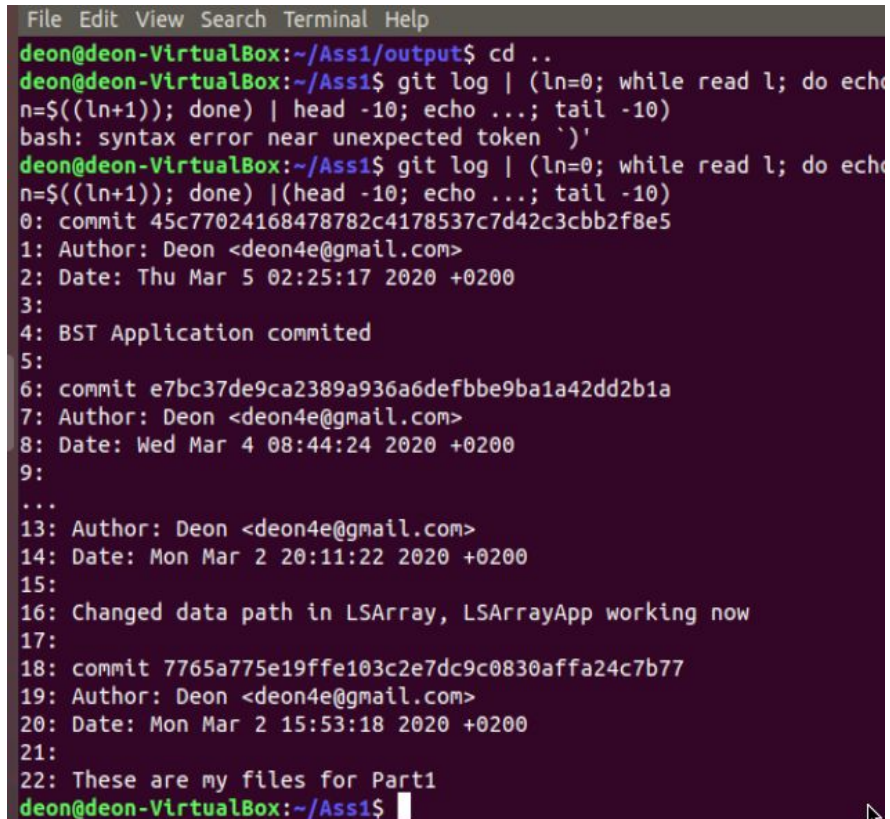
```
File Edit View Search Terminal Help
IW LSBSTAppWrongParams.txt Row 1 Col 1
Areas Not foundAreas Not foundAreas Not found
```

Although Binary Search is a much more effective algorithm at storing large datasets, the insertion numbers obtained from countInsert.txt after insertions was very high. For a large data set, the amount of comparisons for insert is massive. In some cases, my countInsert variable had a total of more than 100k.

Some creativity that went beyond the question:

I processed the dataset before starting to make it uniformly arranged. If you look at my data set in the data directory, you will notice that there are no commas. This was done to make the two values two strings separated by a single space. This saved me a lot of headaches, since I didn't have to deal with the format changing or spaces showing up in odd places when I had to process the strings. I then converted the area string from x1-x2-x3... to x1, x2, x3... just before printing the output by using a .replace function. This made it look like a perfect and original dataset when printAreas() was evoked.

Git log summary:

A screenshot of a terminal window with a dark background and light-colored text. The terminal shows a series of commands and their outputs. The first command is 'cd ..'. The second command is 'git log | (ln=0; while read l; do echo n=\$((ln+1)); done) | head -10; echo ...; tail -10)', which results in a 'bash: syntax error near unexpected token `)''. The third command is 'git log | (ln=0; while read l; do echo n=\$((ln+1)); done) |(head -10; echo ...; tail -10)', which outputs a list of git commits. The commits are numbered 0 through 22. Commit 0 is the root commit. Commits 1 through 5 are related to the 'BST Application'. Commit 6 is a commit by 'Deon' on 'Wed Mar 4 08:44:24 2020'. Commit 13 is another commit by 'Deon' on 'Mon Mar 2 20:11:22 2020'. Commit 16 is a commit by 'Deon' on 'Mon Mar 2 15:53:18 2020'. Commit 22 is a commit by 'Deon' on 'Mon Mar 2 15:53:18 2020' with the message 'These are my files for Part1'. The terminal window has a menu bar at the top with 'File Edit View Search Terminal Help'. The prompt is 'deon@deon-VirtualBox:~/Ass1/output\$'.

```
File Edit View Search Terminal Help
deon@deon-VirtualBox:~/Ass1/output$ cd ..
deon@deon-VirtualBox:~/Ass1$ git log | (ln=0; while read l; do echo
n=$((ln+1)); done) | head -10; echo ...; tail -10)
bash: syntax error near unexpected token `)'
deon@deon-VirtualBox:~/Ass1$ git log | (ln=0; while read l; do echo
n=$((ln+1)); done) |(head -10; echo ...; tail -10)
0: commit 45c77024168478782c4178537c7d42c3cbb2f8e5
1: Author: Deon <deon4e@gmail.com>
2: Date: Thu Mar 5 02:25:17 2020 +0200
3:
4: BST Application committed
5:
6: commit e7bc37de9ca2389a936a6defbbe9ba1a42dd2b1a
7: Author: Deon <deon4e@gmail.com>
8: Date: Wed Mar 4 08:44:24 2020 +0200
9:
...
13: Author: Deon <deon4e@gmail.com>
14: Date: Mon Mar 2 20:11:22 2020 +0200
15:
16: Changed data path in LSArray, LSArrayApp working now
17:
18: commit 7765a775e19ffe103c2e7dc9c0830affa24c7b77
19: Author: Deon <deon4e@gmail.com>
20: Date: Mon Mar 2 15:53:18 2020 +0200
21:
22: These are my files for Part1
deon@deon-VirtualBox:~/Ass1$
```