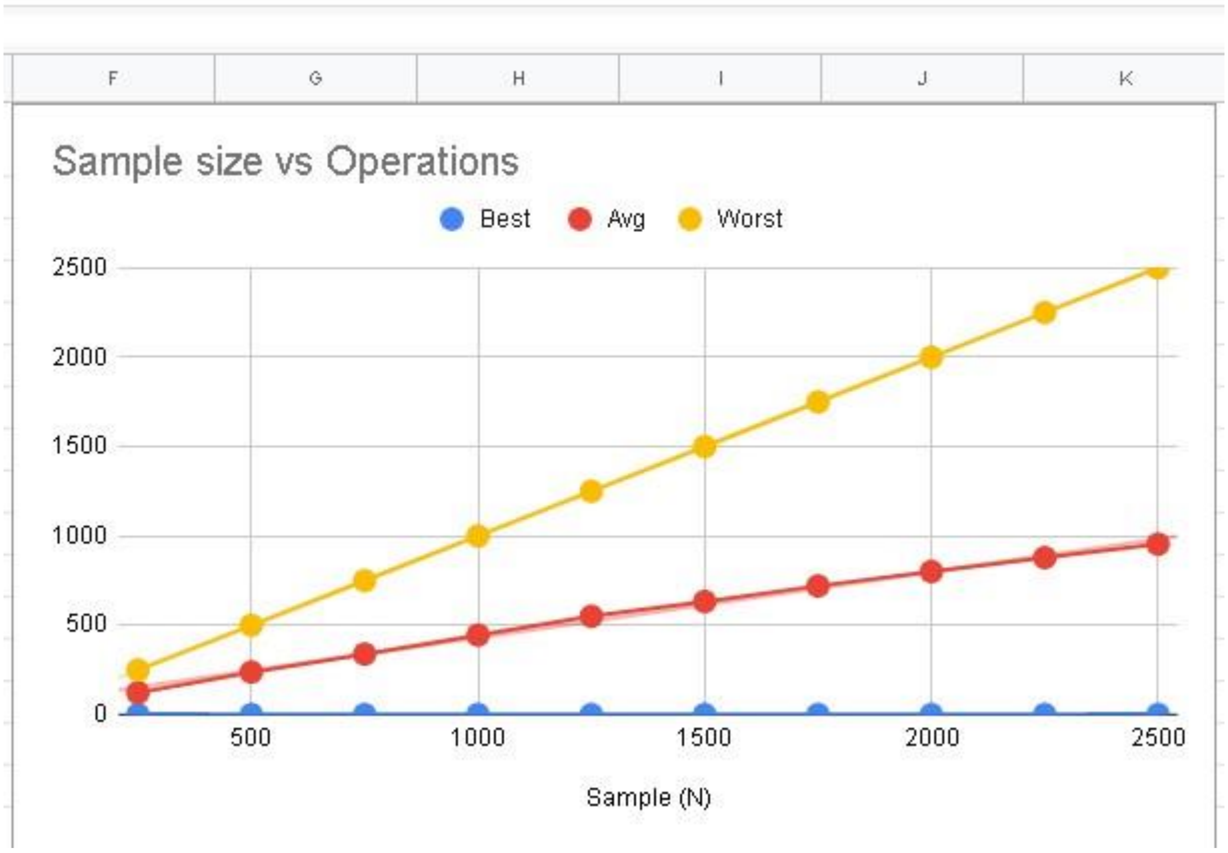
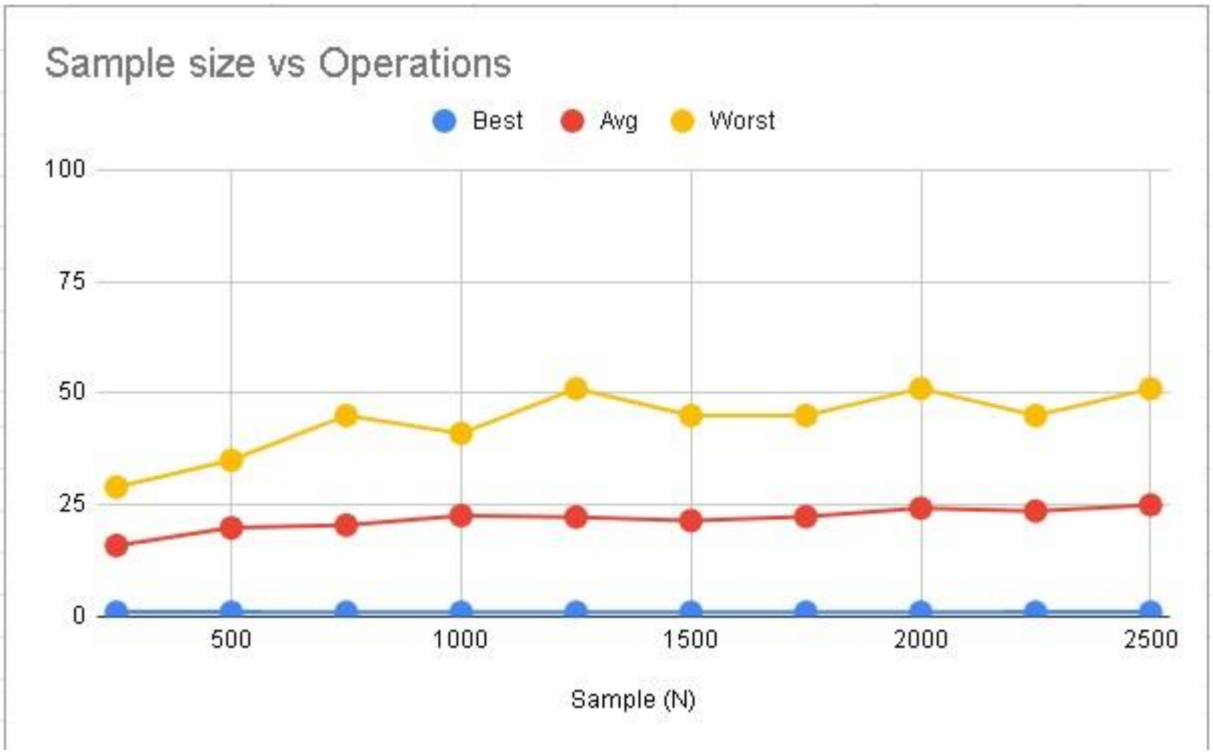
OOP Design
My classes are in the src directory which contains the following classes:LSArrayApp, BTSTApp and
LSArrayApp.class
LSArrayApp.class
This class takes the Load shedding schedule of Cape Town data from the ./Load_Shedding_All_Areas_Schedule_and_Map.clean.final.txt text file into an array of objects. The objects are the stages, date, time and areas. The class also encapsulates the printAreas function which prints out the stage, date and time a specific area entered by the user experiences loadshedding for 2 hours. The printAllAreas function prints out this same data for all areas without a specific order. OpCounts is a counter that counts the number of comparisons done by the program. The functions CreateText() and WriteText() are there for creating a text file and writing the counter value,opCount values ,respectively, whenever the program is run.
LSBTApp.class **LSBTApp.class**
This class does the same task as the above class but instead of putting it into an array, it puts the data into a binary tree. It then invokes the Binary Search tree code when a user enters a stage. The OpCount counts the number of comparisons the program does before it gets to a certain data which the Binary search tree is looking for. The CreateText() and WriteText() functions are for creating and writing the counter's values in a text file.
Experiment and Goals **Experiment and Goals**
The purpose of this exercise is to compare the implementation of a Binary search tree and an array. The comparison will be done using the different counters that are in the two respective classes, LSBTApp.class and LSArrayApp.class.
The LSArrayApp.class takes the data from ./Load_Shedding_All_Areas_Schedule_and_Map.clean.final.txt and puts it into an array of objects. The first cell of an array holds 2 mini arrays that store the stage, date and time in the first cell and the area in the second cell. A counter, OpCount was placed to monitor every comparison the program does when a user enters the data, he/she wants found and the program goes through its execution. If not found, the program displays that it doesn’t exist.
The LSArrayApp.class takes the data from ./Load_Shedding_All_Areas_Schedule_and_Map.clean.final.txt and puts it into an array of objects. The first cell of an array holds 2 mini arrays that store the stage, date and time in the first cell and the area in the second cell. A counter, OpCount was placed to monitor every comparison the program does when a user enters the data, he/she wants found and the program goes through its execution. If not found, the program displays that it doesn’t exist.
I did the comparison of the worst/average/best case scenarios by comparing the counters. I used a python program to graph the number of comparisons against the time taken to finish for different sets of data.
I did the comparison of the worst/average/best case scenarios by comparing the counters. I used a python program to graph the number of comparisons against the time taken to finish for different sets of data.
Trial test and outputs **Trial test and output**s


F
G
H
I
J
K
Sample size vs Operations
Best
Avg
Worst
2500
2000
1500
1000
500
0
500
1000
1500
2000
2500
Sample (N)



	Best Case	Average Case	Worst Case
Dataset Size			
Number of Comparisons			

Discussion of Results **Discussion of Results** **Discussion of Results** **Discussion of Results** **Discussion of Results**

During the test I chose,

As evident from the graphs, it shows that the array takes less counts for smaller values of N, which makes it efficient in a way. But as an gets larger, the number of counts gets larger too making the array lose its efficiency as N increases. It follows a linear format.

As evident from the graphs, it shows that the array takes less counts for smaller values of N, which makes it efficient in a way. But as an gets larger, the number of counts gets larger too making the array lose its efficiency as N increases. It follows a linear format. The table shows some comparisons done

The binary search tree tells a different story. The number of comparisons is by far less when compared to the array. The graphs for the binary search tree show a nonlinear variation because the data is randomly added so the number of comparisons will be nonlinear as well. The binary tree comparisons lose efficiency more slowly that the array comparisons.

	Best case	Average Case	Worst Case
Data Size	120	500	1020
Number of comparisons	21	185	384

So, in conclusion I say the binary search tree is more efficient than the array.