A study of the efficiency of different data structures by comparison of their time complexities: Array and Binary Search Tree

Abstract

Time complexity is a topic with its interests based in measuring how long an algorithm it takes complete its execution relative to the size of the data upon which it is executing [1]. The relationship between the size of a data set and the number of operations required to complete a certain algorithm on that data size is often used as a speculative measure of the time it would take for that algorithm to complete its operation. Logical reasoning can then be applied to determine if a certain algorithm is efficient or not based on its time complexity where a large time complexity would mean that the algorithm has a high time cost associated with its operation.

Introduction and Aim

Data storage and retrieval is central to computational application therefore it is imperative that reconsider how data is structured in an application and how that structure affects the efficiency of the application. It then comes naturally that we ask which data structure is most efficient in its application as this would ensure that we have the minimum time cost per instance of running that application. In this application two data structures will be investigated; that is the array and the binary search tree. The aim of this investigation is to find and compare the time complexities of is it two aforementioned data structures in order to determine which of these data structures is more efficient at data storage and retrieval. efficiency will be measured by the time complexity of each data structure with the one that has the least time complexity being the one that is more efficient.

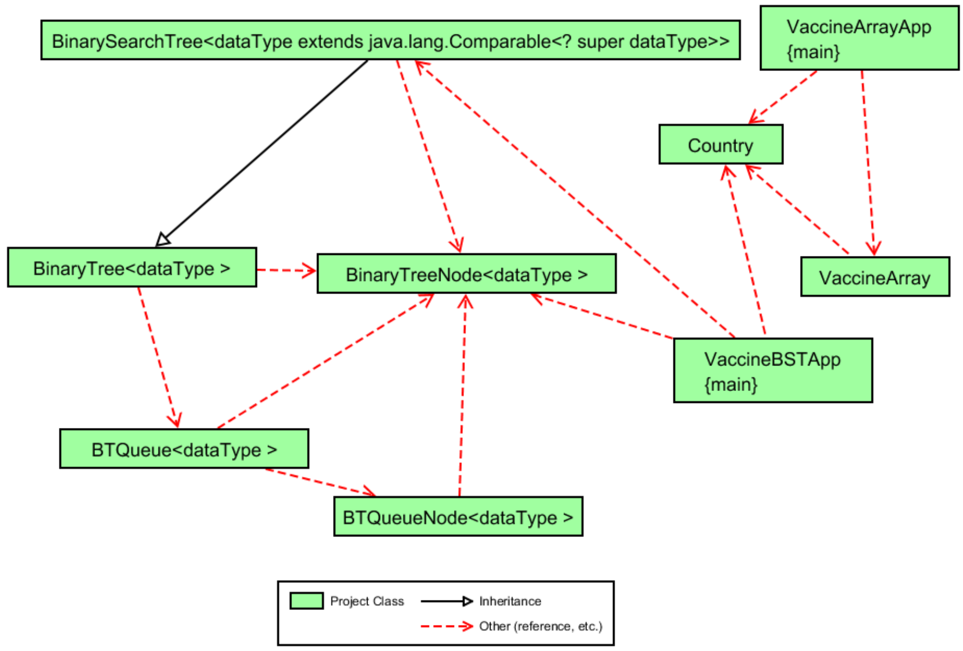
Methodology

Apparatus

- Computer (preferably a Linux based operating system)
- Java compiler
- A file with data to experiment on.

The experimental process began with the design and writing programs that would read in data from a data file and populate each data structure accordingly. An insertion algorithm, respective to each data structure, was utilized to insert these data items. The number of comparisons executed during this stage were counted, to be used as a time complexity indicator at a later stage. Data items were then queried and the number of comparisons executed during each query was also recorded. To compare the change in the number of operations per query, the initial data set was divided into subsets at a regular interval. Data items were also queried in these subsets and operations counted accordingly. accordingly. Graphs were then generated using the varying size of the subsets and their respective number of operations executed. These graphs were then analysed to verify theoretical derivations of time complexity.

Object-Oriented Design



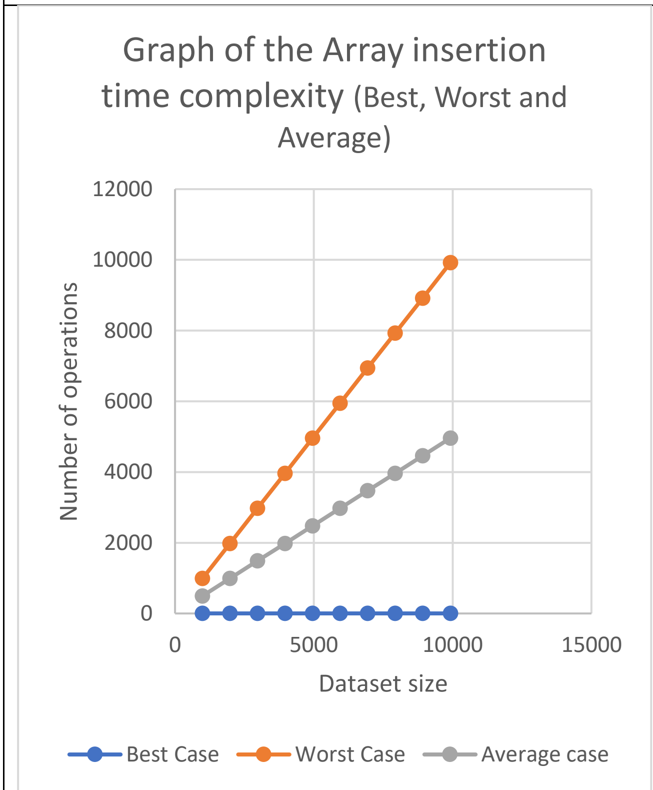
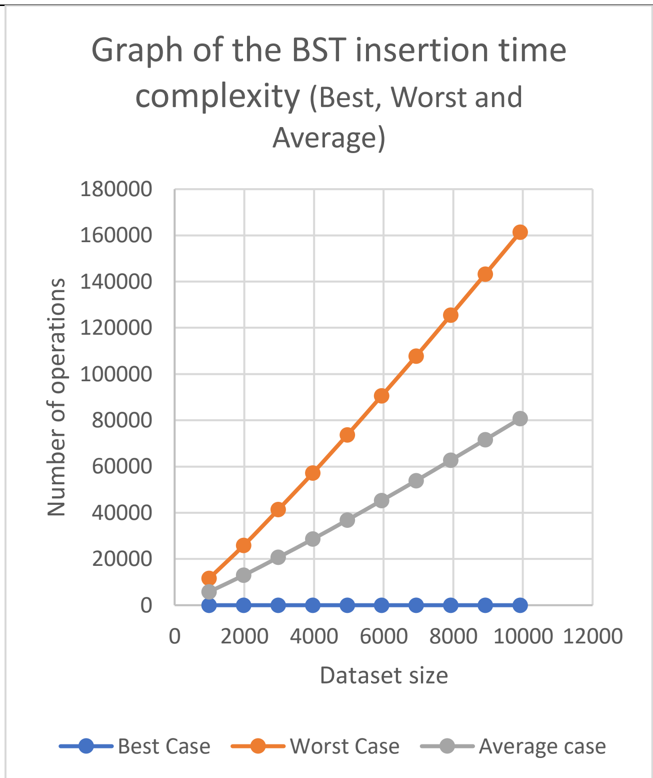
Graphs and Tables

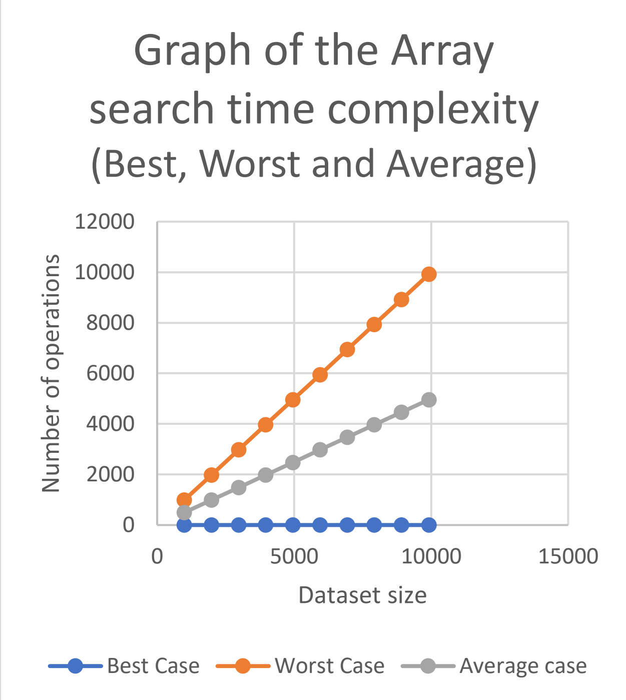
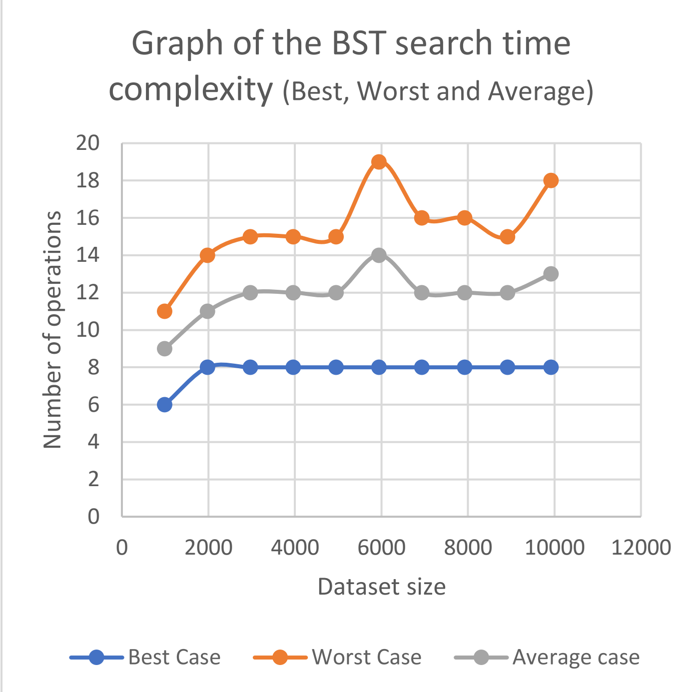
Dataset Size (int)	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
Best Case	Dominica	Dominica	Dominica	Dominica	Dominica	Dominica	Dominica	Dominica	Dominica	Dominica
	19/01/2022	19/01/2022	19/01/2024	19/01/2025	19/01/2026	19/01/2027	19/01/2028	19/01/2029	19/01/2030	19/01/2031
insert comparisons	1	1	1	1	1	1	1	1	1	1
search comparisons	1	1	1	1	1	1	1	1	1	1
Worst Case	Jordan	Guinea-Bissau	Uruguay	Syria	New Caledonia	Botswana	Myanmar	Cameroon	Sri Lanka	Maldives
	09/01/2022	20/01/2022	20/01/2022	23/01/2022	11/02/2022	24/01/2022	20/01/2022	11/01/2022	10/02/2022	11/02/2022
insert comparisons	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
search comparisons	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
Average case										
insert comparisons	496	9912	1487	1982,5	2478	2973,5	3469	3965	4460	4960
search comparisons	496	992	1487	1983	2478	2974	3469	3965	4460	4960

Table 1: Array time complexities

Dataset Size (int)	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
Best Case	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan	Afghanistan
	06/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022	01/01/2022
insert comparisons	2	2	2	2	2	2	2	2	2	2
search comparisons	6	8	8	8	8	8	8	8	8	8
Worst Case	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe	Zimbabwe
	09/01/2022	14/02/2022	16/02/2022	16/02/2022	16/02/2022	16/02/2022	18/02/2022	18/02/2022	19/02/2022	20/02/2022
insert comparisons	11609	25857	41311	57104	73633	90610	107783	125484	143235	161347
search comparisons	11	14	15	15	15	19	16	16	15	18
Average case										
insert comparisons	5806	12930	20657	28553	36818	45306	53893	62743	71619	80675
search comparisons	9	11	12	12	12	14	12	12	12	13

Table 2: BST time complexities

Array Insertion time	BST Insertion time
	

Array Searching time	BST Searching time
	

Theoretical analysis

For an array of n items, the theoretically stated time complexity for inserting items and searching for an item in the array, is stated as follows [2]:

	Best	Worst	Average
Insertion	O(1)	O(n)	O(n)
Searching	O(1)	O(n)	O(n)

For a binary search tree of n items, the theoretically stated time complexity for inserting items and searching for an item in the binary search tree, is stated as follows [2]:

	Best	Worst	Average
Insertion	O(1)	O(n)	O(log(n))
Searching	O(1)	O(n)	O(log(n))

Experimental Results

Array

- Insertion:
  - The graph of the insertion time complexity depicts a constant relationship in the best case.
  - The worst-case time complexity shows a linearly increasing relationship with a high gradient.
  - The average time complexity also shows a linearly increasing relationship with a gentler slope.

- Searching:
  - The graph of the insertion time complexity depicts a constant relationship in the best case.
  - The worst-case time complexity shows a linearly increasing relationship with a high gradient.
  - The average time complexity also shows a linearly increasing relationship with a gentler slope.

BST

- Insertion:
  - The graph of the insertion time complexity depicts a constant relationship in the best case.
  - The worst-case time complexity shows a linearly increasing relationship with a high gradient.
  - The average time complexity also shows a linearly increasing relationship with a gentler slope.

- Searching:
  - The graph of the searching time complexity initially increased sharply and then proceeded to settle around a common level.
  - The were some outliers present in the data. However, the general relationship of the time complexity graph was logarithmic in the best case, worst case and average case.

Conclusion

After the experiment was conducted and the results were processed, it was found that the experimental results were in agreement with the theoretically derived relationships. Although some of the experimental results were noisy due to the presence of a finite and comparatively small number of outliers, the data quality was sufficient to extract valid experimental results. It was found that the binary search tree data structure was far more efficient (by multiple of approximately 600) then the array data structure. However, there was a tradeoff when it came to insertion operations where the binary search tree had a significantly greater number of operations required to insert a new item into the binary search tree then to insert a new item into the array. This tradeoff was balanced by the result which showed that these data have the same insertion time complexity. In future, a larger data set, with a greater number of segments, may be used in this experiment in order to generate and more accurate graphical representation of time complexity relationships.

References

[1] G. L. Team, "Great Learning," 5 January 2022. [Online]. Available: https://www.mygreatlearning.com/blog/why-is-time-complexity-essential/#:~:text=Time%20complexity%20is%20the%20amount,you%20understand%20time%20complexity%20clearly.. [Accessed 7 March 2022].

[2] "Know Thy Complexities!," [Online]. Available: https://www.bigocheatsheet.com/. [Accessed 7 March 2022].