A study of data structure efficiency by comparison of time complexities: Array vs Binary Search Tree

Data Structures and Algorithms
South Africa

Msimamisi Sakhile Lushaba

March 8, 2022

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.

1 Introduction

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.

2 Methodology

2.1 Apparatus

- Computer (preferably a Linux based operating system e.g. Ubuntu)
- Java Development Kit (jdk)
- Data file (CSV file)

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.

3 Object-Oriented Design

4 Tables

4.1 Location keys

- Dom = Dominica
- Afgh = Afghanistan
- Jor = Jordan
- Gui = Guinea-Bissau
- Uru = Uruguay
- Syr = Syria
- New Cal = New Caledonia
- Mya = Myanmar
- Cam = Cameroon
- Sri = Sri Lanka
- Mald = Maldives
- Zim = Zimbabwe

4.2 Tables

Table 1: Array Time Complexity

Dataset Size (int)	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
Best Case	Dom	Dom	Dom	Dom	Dom	Dom	Dom	Dom	Dom	Dom
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	Jor	Gui	Uru	Syr	New Cal	Bots	Mya	Cam	Sri	Mald
insert comparisons	991	1982	2973	3964	4955	5964	6937	7928	8919	9919
search comparisons	991	1982	2973	3964	4955	5964	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 2: Array Time Complexity

Dataset Size (int)	991	1982	2973	3964	4955	5946	6937	7928	8919	9919
Best Case	Afgh	Afgh	Afgh	Afgh	Afgh	Afgh	Afgh	Afgh	Afgh	Afgh
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	Zim	Zim	Zim	Zim	Zim	Zim	Zim	Zim	Zim	Zim
insert comparisons	11609	25857	41311	57104	73633	90610	107783	125484	143235	161347
search comparisons	991	1982	2973	3964	4955	5964	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

5 Graphs

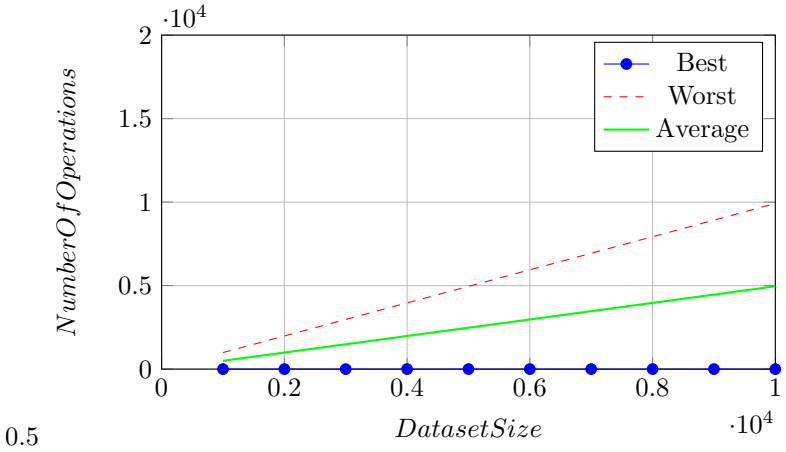

Figure 1: Array insertion time complexity (Best, Worst and Average)

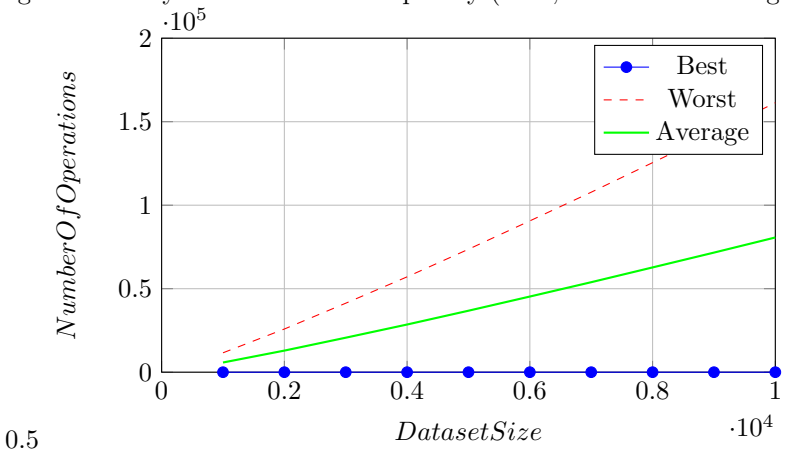

Figure 2: BST insertion time complexity (Best, Worst and Average)

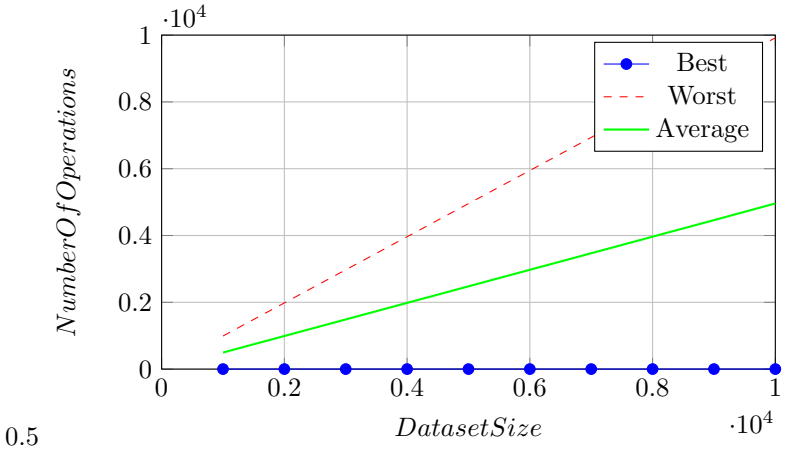

Figure 3: Array Search time complexity (Best, Worst and Average)

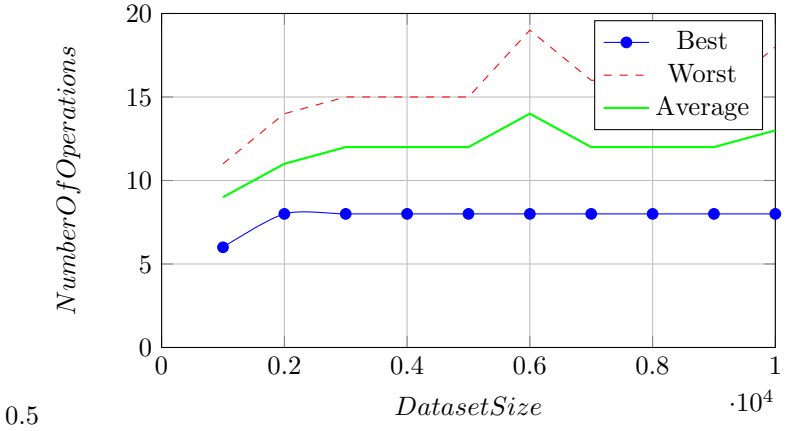

Figure 4: BST Search time complexity (Best, Worst and Average)

6 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]:

Table 3: Array Theoretical Time Complexity

	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]:

Table 4: BST Theoretical Time Complexity

	A	B	C
Insertion	O(1)	O(n)	O(log(n))
Searching	O(1)	O(n)	O(log(n))

7 Experimental Analysis

After conducting the experiment and recording the results for each data structure, the following analysis was conducted.

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

7.2 Binary Search Tree

- **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.

8 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 than the array (by a factor of approximately 600) in storing items where the search time is minimized. However, there was a trade-off 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 than to insert a new item into the array. In future, a larger data set, with a greater number of segments, may be used in this experiment in order to generate a finer graphical representation of time complexity relationships.

9 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].