International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF 2022: 7.942

Evaluation the Performance of Data Structures: A Comparative Approach

Ayshah A. Alnoshan[1], Hessah A. Alhodithy[2], Meznah S. Alquraishi[3]

[1]College of Engineering and Information Technology, Onaizah Colleges, Onaizah, Qassim, Saudi Arabia
Email: *ayshah_an[at]oc.edu.sa*

[2]College of Engineering and Information Technology, Onaizah Colleges, Onaizah, Qassim, Saudi Arabia
Email: *hassah.h[at]oc.edu.sa*

[3]College of Engineering and Information Technology, Buraydah Colleges, Buraydah, Qassim, Saudi Arabia
Email: *meznahqu[at]gmail.com*

Abstract: ***The importance of data structures makes the people to use them in many domains in real world to facilitate the dealing with the data and information. Dealing with these data structures in real worlds are possible using different operations such as insertion, deletion, searching, sorting. The effectiveness of these operations with data structures is important and can be measured in different ways. In this paper we focus on evaluate the effectiveness of operations specially searching and sorting operations on three data structure that is array, singly linked list and tree and measured based on a high range of numbers to increase the accuracy by calculate the elapsed time to execute operations and the number of comparisons and swapping in operation.***

Keywords: Data structure algorithms; sorting; searching; effectiveness; comparison; time complexity

1. Introduction

Sort and search are important operations in computer sciences. The sorting used to arrange data structure elements in a certain order. However, a sorted data structure is a benefit for searching which is considered the second operation. Hence, to carry out the data processing activities including calculations, sorting and searching use an algorithm which that is a significant fraction of the several science fields. For examples, database systems as well as searching, networking, data transmission, pattern matching, and data analytics. Furthermore, there are a large number of good sort algorithm and we will cover some of them like insertion, bubble, quick and heap [1][3].

One of the oldest sorting algorithms is insertion sort. It works by inserting each element in its correct place in the dataset. That means it removes one element from the input data and inserted into the appropriate position relative to the sorted elements. The algorithm continues doing these steps until no input elements remain. Bubble sort is the simplest sorting algorithm and the basic idea works by repeatedly comparing adjacent elements and swapping the pair of adjacent elements if their order is reversed. It repeats this step until no swaps have occurred on the data set [2][5]. Quick sort is a divide and conquers algorithm. In general, it will be applied recursively by select element as pivot element and compare the elements remain to it. So, the set was split then select another pivot from the subset and so on. The last algorithm present is heap-sort. In brief, the algorithm is continuous swap the first element in the data set with the last element [1][2].

On the other hand, the efficiency of these algorithms depends on a type of data structure which is a way to facilitate access to organized data and process it. Some of the data structures examples are array, linked list and tree. An array is a set of elements with fixed size and each item has a specifically location in the set. The second data structure type is linked list which it has a node and each of them connected by another through a specified link. For this reason, linked list has a dynamic size. Finally, there are different trees have been proposed, but we will discuss the most widely used: the binary tree and the AVL tree. Both of them represent one element as a root of the tree and each element has at most two children [1][4].

In this paper, we will perform a qualitative comparative performance evaluation of these sorting algorithms which that depend on the most representative structures by computing time complexity. The time complexity of an algorithm is the total amount of time will take to complete its execution depending on the input size. We can describe these complexities by using the commonly expressed called big-oh notation.

First, some of the related works are explained in section 2. In section 3 and 4 we present brief description of data structures and operations used and the methodology. While in section 5 we present the results and discuss these results with curves. Finally, in section 6 will conclude the paper.

2. Previous Work

Here we will talk about previous work, we have done research and we have found researches in this area, but we chose some of them.

Authors in [6] was given a comparative analysis of different types of sorting algorithm, e.g. the Quicksort, Bubble sort, Insertion sort and Heap sort. This paper is perhaps the closest to the concept of our paper. Where the aim of this

paper is counting the number of swap and comparison per algorithm by the added counter in the C programming language. However, the researchers after calculate the time complexity of algorithms, they concluded that quicksort is the fastest and bubble sort is the slowest.

Linos and Marcus [7] also worked with the complexity of arrays and linked lists, the also says because the nested loops in the insertion, selection, bubble sort it takes O(n2), but the division of quicksort algorithm into sub lists it has O(nlogn). They said the bubble, insertion, and selection sort very slow or does not work very well with a huge number of unsorted elements but all of them was easy to understand and implement. On the other hand, they say the quick sort was difficult to implement, slow working in sorted elements, but it's very effective way can be used when we have huge random/unsorted elements.

Svetlana and Milo in [8] give a comparative analysis of the most popular types of trees. Additionally, they both based on the height balancing. But the difference between them is that the Red-Black tree depends on the colors in its branches. On the other hand, the evaluation in this study done by simulation with a synthetic workload model. After the authors done the experiment on the number of data set. They found that the AVL trees has the efficiency better.

In [9] the author focused on the analyzing of two types of sorting algorithms selection and quick sorting algorithms on different types of inputs by measuring their speed with many implementations. He applied these two algorithms on integers and strings types. Where the researcher show that the performance of selection sort is better than the performance of quick sort and the array of integer type is faster than the array of string type.

The ordering of items and elements is more important to make the dealing with these elements is much easier. In [10] the author focuses on the analysis of different sorting algorithms. The researchers show the implementation and results got it after applying different execution on these algorithms and show that index sort is the best with small elements, but with large elements index sort is take more time than insertion and less time than bubble sort.

Authors in [13], reviewed evaluating the available voxelisation algorithms for different geometric primitives as well as voxel data structures and they ware spilt data structures into static and dynamic grids considering the frequency to update a data structure.

In the paper [14], the researchers discuss the performance of data structures with machine language and when they are called learned data structures. As a corollary of this general analysis, learned data structures achieve outstanding practical improvements in space occupancy and time efficiency.

Yang et al. in [15] represented the various types of sorting algorithms applied to arrays. This paper describes the best and worst cases. They represent the results as the time complexity of insertion, bubble and selection sort is O(n2). On the other hand, the time complexity of quicksort is O(nlogn). The researchers also note to the time complexity of all sorting algorithms it doesn't have more difference with small input data, and it stays quicksort is the best whenever growing on input data.

Researchers in [16], proposed a methodology sorting technique by dividing the input array into a piece, sorting each of the blocks using a proposed bubble sort, and then merging all of the blocks together using a modified insertion sort. As result, they showed how various scenarios performed in addition the suggested sort outperforms traditional bubble and insertion sorting.

Roopa and Reshma in [17], conducted a comparative study on various sorting algorithms that depends on two parameters first, the time taken when is executed the data and second parameter the execution speed of the data. The summarized results were the larger and complex lists of data binary search can be used by taking small amount of time in contrast with linear search which can be used for less amount of data. Finally, quick sort it the most efficient to handle large data as compared to selection sort and bubble sort.

Authors in [18] examined nine of well-known algorithms and they found that quicksort is the best choice according to memory need when it comparing with heap sort and merge sort.

Min in [20], presented two ideas of bidirectional for bubble sort algorithm to optimize the original bubble sort algorithm and examine these two ideas of bidirectional for bubble sort algorithms and analyzing them by comparing the time complexity and space complexity of these two ideas with the original.

3. Types of Data Structures and Operations

In our paper we focus on three types of data structures which is array, linked list (singly) and tree with two types of operations: sort and search operations.

3.1 Data Structure

Array is one of data structure types which contains a set of elements grouped together with the same type such as integer, string, double and so on and have an index used to access any element in the array. Because the array has organized elements it can be used effectively in different operations such as search and sort. In addition, it can be one-dimensional or multi-dimensional array and this support in different features. Linked list another type of data structures which is a sequence of nodes where each of these nodes contains a data of any type and pointer to the next that is used to access any node in the linked list. The linked list may be one of singly linked list which it has only one pointer to the next, double linked list which it has two pointers one for the next node and another for the previous node or cycle linked list which connect all nodes together. Tree is an important type of data structures that contains a collection of nodes connected together using the edges in hierarchal form

without having any cycle. Where we dealing with the tree with a set of things such as root (the first element), children of each node and so on. Where the tree can be balanced tree such as AVL tree and unbalanced tree such as Binary Search Tree (BST) and based on these we can analyze different operations such as deletion, insertion, traversal that can be in-order, preorder or post-order and so on and how it works on each type [11].

3.2 Operations

3.2.1 Sorting operation

Sorting operation is one of the most important operations used to make the reorganization for a set of elements either in alphabetical order for elements of type string or character or in ascending or descending order for elements of integer type. Where sorting operation apply any algorithm of sorts on a set of elements that take it as input and return the ordered elements as output with any type of data structures. The sorting algorithms that we show it in our paper are:

Bubble sort is the simplest algorithm for sorting where it starts from first element until the number of elements-1 and compares each element with all elements and makes swapping in correct position. But it may take more time because it compares each element with all elements. Where the time complexity of bubble sort in worst case is $O(n^2)$ and in best case is O(n) [12].

Insertion sort is an inserting algorithm where it takes the first element and inserts it into the list in ordered form and continuo with each item at a time until we got the list in ordered form. The insertion sort can take less time than bubble sort because it has less comparison. Where the time complexity of insertion sort in worst case is $O(n^2)$ and in best case is O (n) [12].

Quick sort is a popular sorting algorithm based on divide and conquer procedure. Where it chooses pivot that may be first, last, middle or random elements and make partition around this pivot and every element less than pivot go left and greater than pivot go left and continue in recursively way with left and right partition until we get the list in ordered form. Where the time complexity of quick sort in worst case is $O(n^2)$ and in best case is O (nlogn) [12].

Heap sort is a sorting algorithm with comparison procedure and based on heap data structure. Where first it builds a max or min heap then take the element in the root and swapped with the element at the end and make heapify to root to rearrange heap data structure after applying change. Where the time complexity of heap sort in worst case is O(nlogn) and in best case is O(nlogn) [12].

3.2.2 Searching operation:

It is an operation used to look up for specific element in the given list. Where the searching operation can be done in many forms such as in sequential form and take the time complexity of O(n) or in binary form and take the time complexity of O(logn) [11][12].

4. Methodology

We are going to use a macOS operating system with an i5 2.5 GHz CPU. The test programs are constructed utilizing the java language for all data structures that we are focused on it in our paper, which are array, singly linked list and tree. Where we applying test on different data structures with different set of elements ranged by 50000 elements to be more accurate and measuring the results based on these sets. In considering for analyzing the results, we are using five different sets of numbers as input for each data structures in both searching and sorting operation and measure the number of comparisons and swapping for each one. Whereas the second thing use it to analyzing the results is by calculate the elapsed time to measure the performances of these operations by calling nanoTime() function for searching operation because it working fast and calling currentTimeMillis() for sorting operations because it takes too much time.

We try to running the algorithm more than one time to check the accuracy of results. On the other hand, in searching operation, we take number 30000 as an input in all cases and measure the results based on this searching. Finally, after we collecting different sets of results, we analysis them based on number of certain operations (comparisons and swapping operations) and the elapsed time for each operation in different data structures.

5. Performance Results and discussion

After we collecting all experiments and testing on different data structures with sorting and searching operations, we analysis the results based on the time elapsed to execute and certain operations which is comparisons and swapping operations and show the final results in curves and tables.

5.1 Sorting Operations

In our paper we choose different algorithms that is bubble, insertion, quick, and heap sorting algorithms to execute it on tree different data structures.

5.1.1 Array Sort

In sorting for array, all sorting algorithms in best case is better than in worst case in considering for number of comparing and swapping except the heap sorting algorithm because if it is already sorted they take more comparisons and swapping operations to build max heap again, but if it is not sorted, the number of comparing and swapping reduces because the array is already built in max heap. However, in considering for time, all sorting algorithms in best case is better than in worst case. Generally, the fastest sorting algorithm for array in best and worst case is heap sort. Thus, in best case the slowest algorithm is bubble then quick then insertion as shown in Figure 1.

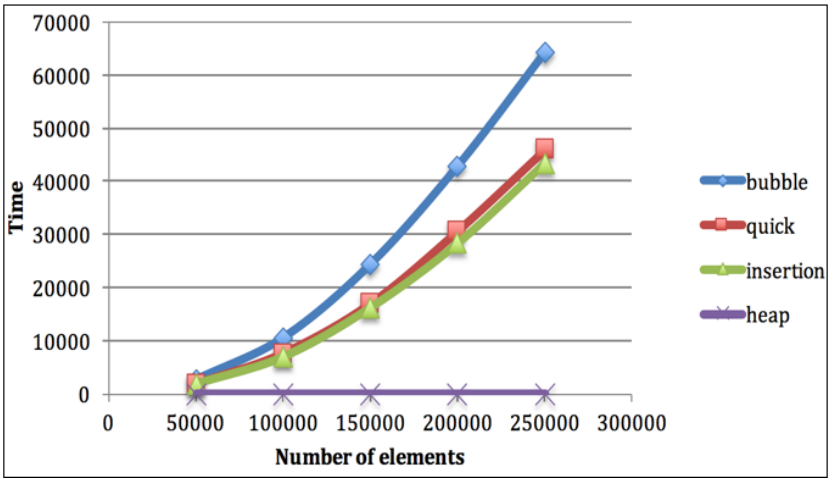

Figure 1: Graphical form of elapsed time for different sorting algorithm of array in best case.

In considering for worst case the slowest algorithm is insertion then bubble then quick sorting algorithm because the number of comparisons and swapping operations is affected on the speed of algorithm as shown Figure 2.

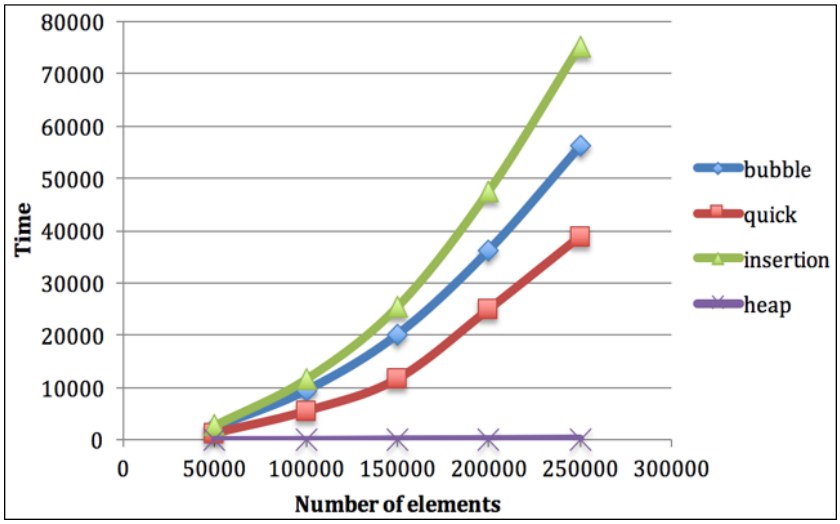

Figure 2: Graphical form of elapsed time for different sorting algorithm of array in worst case.

5.1.2 Linked List Sort

In sorting for singly linked list, all sorting algorithms in best case is better than in worst case except the heap sorting algorithm in considering for number of comparing and swapping because if it is already sorted they take more comparisons and swapping operations to build max heap again, but if it is not sorted, the number of comparing and swapping reduces because the array is already built in max heap. In addition, insertion sorting algorithm it is better in worst case because it takes numbers and directly insert it in the sorted list, but in best case it takes number and then need to check for all numbers previously inserted in the sorted list so it takes more time and comparisons.

As shown in Figure 3 and Figure 4, in considering for time, all sorting algorithms in best case are better than in worst case. In general, the fastest sorting algorithm for singly linked list in best and worst cases is insertion then bubble then quick then heap sort. Where insertion and bubble in best case is very close in the result of time so it appears in the curve one above the other. Because the using of pointer in singly linked list is affected on results as we saw in the heap and quick it takes more and more time because it dealing with pointers very dramatically.

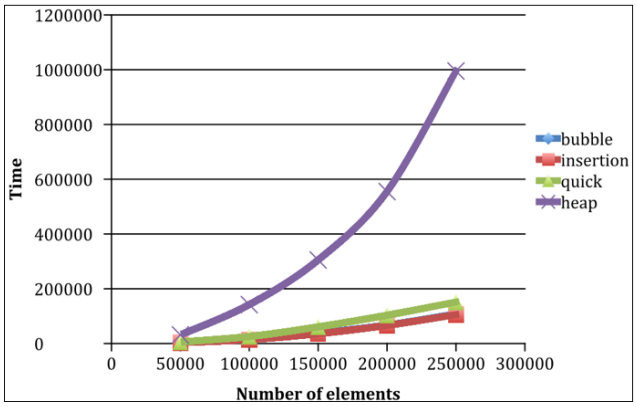

Figure 3: Graphical form of elapsed algorithms time of linked list in best case.

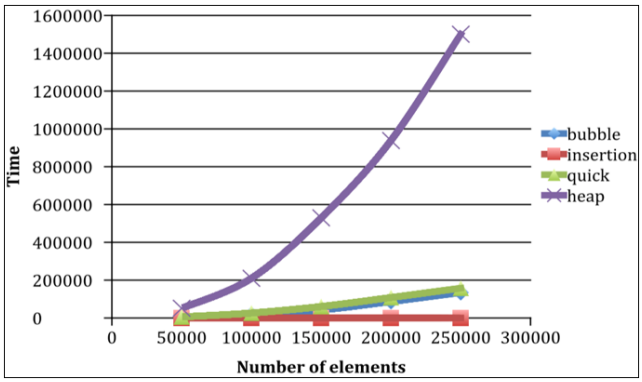

Figure 4: Graphical form of elapsed algorithms time of linked list in worst case.

5.1.3 Tree Traversal:

In this section, we analysis two types of tree: AVL and Binary Search Tree. When our experiment was run we obtained the number of visiting node and the comparison number in each type is the same for both cases. Since the AVL is a self-balancing binary search tree. In considering on traversal time, AVL traversal is better than BST traversal in all cases. Because BST in the best case or worst case is very deep that's mean all traversals in one side either be left or right. So, it takes a long time in the traversal. On the other hand, we could get the BST is the best case was much better than the worst case, but the worst case in AVL takes a little more time from the best case as shown in Figure 5 below.

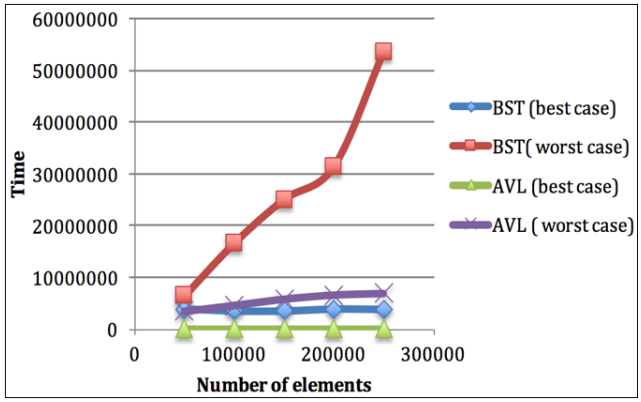

Figure 5: Elapsed time tree traversal in best and worst cases.

5.2 Searching Operations

In this section, we apply searching operation to different data structures.

5.2.1 Array Search:

We use binary and linear search and measure it in both best and worst case and found that binary search in both cases is faster than linear search since it takes less time to perform searching and less comparisons as shown in Figure 6 below.

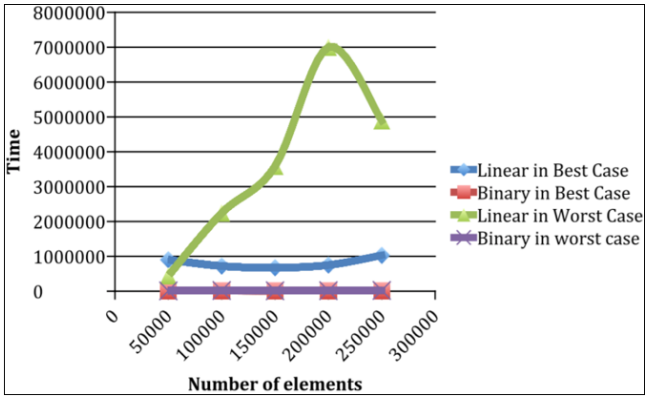

Figure 6: Elapsed time for searching array in best and worst cases

5.2.2 Linked List Search

In searching for singly linked list, there is only one way to search through it that is linear searching. Where we use it and measure it in both best and worst cases and found that in best cases is faster than in worst cases since it takes less time to perform searching and less comparisons as shown in Figure 7.

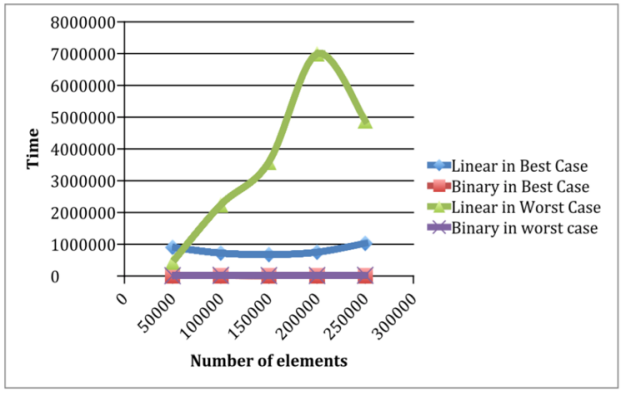

Figure 7: Graphical form of elapsed time for searching linked list in best and worst cases.

5.2.3 Tree Traversal

In tree we apply searching on Binary Search Tree (BST) and AVL tree and found that the time in AVL tree in worst case is better than BST tree as shown in the Figure 8 below.

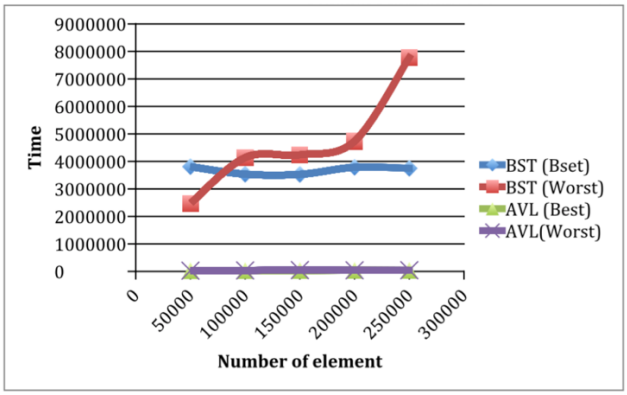

Figure 8: Elapsed time for searching tree in best and worst cases

From the results we got it, as expected, the BST tree takes the highest number of steps in each case. Nevertheless, the number of steps in the best case takes less than the worst

case. Otherwise, the minimum step number is represented in the AVL best case. Additionally, the number of steps in the AVL worst case is small and this is due to the AVL is a balanced tree.

6. Conclusion

This paper evaluates the effectiveness of sorting and searching operations for array, singly linked list and tree data structures in best and worst cases by measuring the time those operations it takes and the number of comparisons and swapping for each operation. The domain of these evaluation focuses on integer numbers with high ranges number start from 50000 to increase the accuracy of the results. The results showed that the cases of number either in worst or best case, the increasing of time and number of comparisons and swapping effect on the effectiveness and performance of operation on each data structure. Where in considering for sorting operations in three data structures the time for sorting algorithms showed exactly the performance of that algorithm and proves that with the number of comparisons or swapping. In considering for searching operation, the time elapsed and number of comparisons in binary search is better than in linear search. Also, search in balanced tree is better than in unbalanced tree. Hence, we can say that it can be there many future works to improve various sorting and searching algorithms and removing its disadvantage and also make new algorithms to keeping on with rapidly growing up of information in our world.

References

[1] H. K. Nievergelt J. Introduction to algorithms. The MIT Press, 2009. pp. 147-180.
[2] Z. ABBAS. COMPARISON STUDY OF SORTING TECHN IQUES IN DYNAMIC DATA STRUCTURE. Malaysia, 2016. pp. 6-18.
[3] Roy, H., Shafiuzzaman, M., Samsuddoha, M. (2019, December). SRCS: A New Proposed Counting Sort Algorithm based on Square Root Method. In 2019 22nd International Conference on Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.
[4] L. NCHENA. M. LARSSON. “SORT ALGORITHMS AND DATA STRUCTURE: AN OVERVIEW AND COMPARISSON.” International Journal of Information, Business and Management, Vol. 9, No.1, pp. 277-290, 2017.
[5] S. Jadoon, S.olehria, M.Qayum “Optimized Selection Sort Algorithm is faster than Insertion Sort Algorithm: a Comparative Study.” International Journal of Electrical and Computer Science, Vol. 11, No.02, pp. 18-23, 2011.
[6] S. Choudaiah, M. Tinn Kavitha efeld and P. Chowdary, "Evaluation of Sorting Algorithms, Mathematical and Empirical Analysis of sorting Algorithms", International Journal of Scientific & Engineering Research, vol. 8, no. 5, pp. 2229-5518, 2017.
[7] L. NCHENA and M. LARSSON. SORT ALGORITHMS AND DATA STRUCTURE: AN OVERVIEW AND COMPARISSON. International Journal of Information, Business and Management 02 / 2017. Volume. 9, Essue.1.
[8] S. Štrbac-Savić and M .Tomašević. Comparative Performance Evaluation of the AVL and Red-Black Trees. BCI’12, September 16–20, 2012, Novi Sad, Serbia.
[9] A. Aliyu and B. Zirra, "A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays", The International Journal Of Engineering And Science (IJES), vol. 2, no. 7, pp. 25-30|, 2013.
[10] A. Bharadwaj and S. Mishra, "Comparison of Sorting Algorithms based on Input Sequences", International Journal of Computer Applications, vol. 78, no. 14, pp. 7-10, 2013.
[11] A. Drozdek, Data structures and algorithms in C. England: COURSE TECHNOLOGY, 2005.
[12] Barnett, Granville, and Luca Del Tongo . " Data Structures and Algorithms : Annotated Reference with Examples." (2008).
[13] M. Aleksandrov, S. Zlatanova, and D. J. Heslop, “Voxelisation algorithms and Data Structures: A Review,” Sensors, vol. 21, no. 24, p. 8241, 2021. doi:10.3390/s21248241.
[14] P. Ferragina, F. Lillo, and G. Vinciguerra, “On the performance of Learned Data Structures,” Theoretical Computer Science, vol. 871, pp. 107–120, 2021. doi: 10.1016/j.tcs.2021.04.015
[15] Y. G. You Yang, Ping Yu. Experimental Study on the Five Sort Algorithms. International Conference on Mechanic Automation and Control Engineering (MACE), 2011
[16] T. Paul, “Enhancement of bubble and insertion sort algorithm using block partitioning,” 2022 25th International Conference on Computer and Information Technology (ICCIT), 2022. doi:10.1109/iccit57492.2022.10055404.
[17] Ms ROOPA K, Ms RESHMA J, “A Comparative Study of Sorting and Searching Algorithms” India, 2018.
[18] Kazim Ali, " A Comparative Study of Well-Known Sorting Algorithms", Lahore Leads University, Pakistan, 2017.
[19] Reyha Verma and Jasbir Singh, " A Comparative Analysis of Deterministic Sorting Algorithms based on Runtime and Count of Various Operations ", International Journal of Advanced Computer Research, V olume-5 Issue-21 December-2015.
[20] W. Min, “Analysis on Bubble Sort Algorithm Optimization”, in International Forum on Information Technology and Applications, Kunming, China, 2010.