ISSN No. 0976-5697

VOLUME 13 SPECIAL ISSUE 1, JUNE 2022

International Journal of Advanced Research in Computer Science

CASE STUDY

Available Online at www.ijarcs.info

A CONTRASTIVE ANALYSIS OF SORTING ALGORITHMS IN DIFFERENT LOAD SCENARIOS

Dr. Mayank Patel
Associate Professor, dept. of CSE
Geetanjali Institute of Technical Studies
Udaipur, India
mayank.patel@gits.ac.in

Sheetal Sharma*
UG Scholar, dept. of CSE
Geetanjali Institute of Technical Studies
Udaipur, India
sharmasheetal01761@gmail.com

Nitesh Kumawat
UG Scholar, dept. of CSE
Geetanjali Institute of Technical Studies
Udaipur, India
niteshkumawat0015@gmail.com

Abstract: Sorting algorithms are the very important data structure operation. As we have millions or trillions of data stored in our memories, it is very difficult of us to find a specific required data. To sort data is to arrange them in ascending or descending order so as the searching, locating or arranging of data becomes easy. Every sorting has some advantages and some disadvantages, like wise each sorting algorithm takes different time to sort the data. In this research paper we compared various sorting algorithm in respect to their execution time. The efficiency of every algorithm varies with the number of input and we have compared the efficiency of algorithm so that we can could which algorithm is best to use based on the load. The sorting algorithms are evaluated in JAVA.

Keywords: Bubble; Heap; Insertion; Merge; Selection; Sorting Algorithm Evaluation

1. Introduction

Sorting is an important aspect of data structures and algorithms, they exist in a variety of forms but the well-known among them are of 5 major types which are being focused in this paper. It is important to investigate the behaviour of these algorithms on different number of inputs as they work differently depending on number of inputs supplied [1]. This comparative study will advance the level of under sting on these algorithms. The constraint we focus here is the number and variety of input supplied to the different sorting algorithms.

2. Different Types of Sorting Algorithm

Different types of sorting are viz selection sort, Insertion sort, Bubble Sort, Merge Sort and heap sort along with their complexities are briefly described below.

2.1 Selection Sort

Selection sort is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end [2]. Initially, the sorted part is empty and the unsorted part is the entire list. The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right. This algorithm is not suitable for large data sets as its average and worst-case complexities are of O (n2), where n is the number of items.

2.2 Insertion Sort

This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted [3]. For example, the lower part of an array is maintained to be sorted. An element which is to be 'inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst-case complexity are of O (n2), where n is the number of items.

2.3 Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order [4]. This algorithm is not suitable for large data sets as its average and worst-case complexity are of O (n2) where n is the number of items.

2.4 Merge Sort

Merge sort is a sorting technique based on divide and conquer technique [5]. With worst-case time complexity being O (n log n), it is one of the most respected algorithms. Merge sort first divides the array into equal halves and then combines them in a sorted manner.

2.5 **Heap Sort**

In computer science, heap sort is a comparison-based sorting algorithm. Heap sort can be thought of as an improved selection sort: like that algorithm, it divides its input into a sorted and an unsorted region, and it iteratively shrinks the unsorted region by extracting the largest element and moving that to the sorted region [6]. The improvement consists of the use of a heap data structure rather than a linear-time search to find the maximum.

Although somewhat slower in practice on most machines than a well-implemented quicksort, it has the advantage of a more favourable worst-case O (n log n) runtime. Heap sort is an in-place algorithm, but it is not a stable sort.

3. Experimental Setup

For evaluating the sorting algorithm, we opted java programming language. Since java is a platform independent language it is most widely used nowadays, therefore we may have tons of data stored for storing and displaying in sorted form in java platforms.

We performed our analysis on five sorting algorithms that are Selection, Insertion, Bubble, Merge and Heap. The code of these algorithms is written in java and written to optimize the sorting of elements [7]. We experimented with different amount of data and observed the efficiency of the algorithms in different cases. Initially we started with applying with limited load to be sorted that is with 10, 50,100 elements. We observed that the algorithm in limited load or limited number of elements worked fine and the observed result was recorded for every algorithm. moving forward to to check the better efficiency the load of elements that is the number of elements were increased, now the algorithm was tested for moderate load that is 500,1000 elements. Like before the behavior of algorithms for different elements were different. The change in the results were observed and recorded. Likewise, we then observed the behavior and efficiency if the algorithm for high load of elements that is now the algorithm were tested for 5000, 10000 elements. Therefore, we calculated the time for sorting in different algorithm in Nano seconds. All of the above-mentioned sorting was performed in java language.

4. Result

We performed the sorting on five algorithms. Behaviour of algorithms are recorded in tables. The table shows the time taken by each algorithm for sorting. The time is recorded in Nano seconds. The observed and recorded result of the sorting algorithms in different types of input are:

4.1 **Limited Load**

The limited load indicates the number of input elements. For this case we take 10, 50,100 elements for sorting. Table 1 shows the time in nano seconds for limited load for different sorting algorithms. The fig. 1 shows the graphical analysis of different sorting in limited load.

Table 1: Sorting algorithm's execution time for limited load.

Sorting Algorithms	Number of Elements			AVG (nSec)
	10	50	100	
Selection	0.49688	4.1402	8.436178	4.357753
Insertion	0.55034	4.170745	9.840375	4.85382
Bubble	0.30503	4.02536	7.69706	4.00915
Merge	1.211539	4.57714	6.401544	4.063408
Heap	1.241475	4.168791	7.050591	4.153619

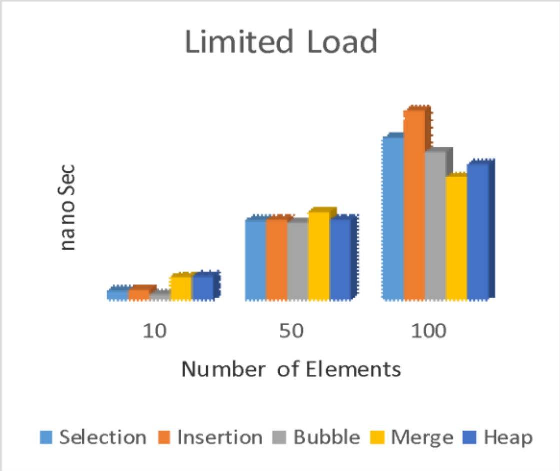

Fig 1: Graphical analysis of different sorting algorithm in limited load.

4.2 **Moderate Load**

Now for further analysis, we increase the load of the input. We compared the algorithm for 100, 500, 1000 elements. The table 2 shows the time in nano seconds for limited load for different algorithm. The fig 2 shows the graphical analysis of different sorting in moderate load.

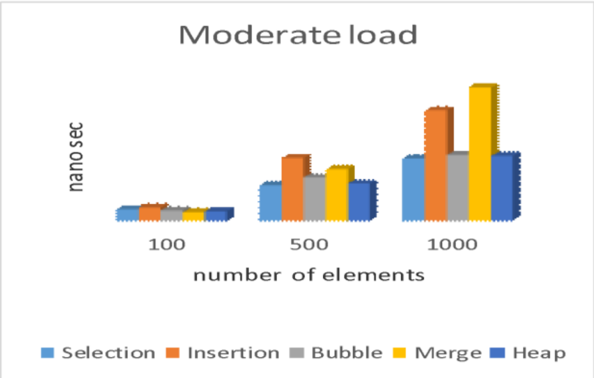

Fig 2: Graphical analysis of different sorting algorithm in moderate load.

Table 2: Sorting algorithm's execution time for moderate load

Sorting Algorithm	Number of Elements			AVG (nSec)
	100	500	1000	
Selection	8.436178	25.89322	45.23901	26.5228
Insertion	9.840375	45.63206	80.12113	45.19786
Bubble	7.69706	31.5683	47.70247	28.98928
Merge	6.401544	37.60888	96.48499	46.8318
Heap	7.050591	27.26145	47.02505	27.11236

4.3 **High Load**

For final analysis we increased the number of input and sorted for 1000,5000,10000 elements. The table (table no.3) shows the time in Nano seconds for limited load for different algorithm. The fig 3shows the graphical analysis of different sorting in high load

Table 3: Sorting algorithm's execution time for High load.

Sorting Algorithms	Number of Elements			AVG (nSec)
	1000	5000	10000	
Selection	45.23901	152.8092	269.0864	155.7115
Insertion	80.12113	97.88319	244.5345	140.8463
Bubble	47.70247	195.9405	542.1678	261.9369
Merge	96.48499	90.68691	273.445	153.539
Heap	47.02505	107.8644	391.1215	182.0036

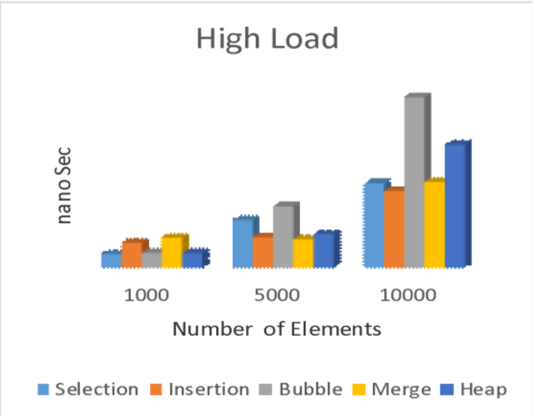

Fig 3: Graphical analysis of different sorting algorithm in high load.

4.4 **Average Case**

For the better understanding of these algorithm, we take the average case of all limited load, moderate load and high load.by these average case study we can conclude that which algorithm is better and can be choose over the other. The average case shows average of all the types of input and we can finally conclude the aim of this paper by that case. The table 4 shows the time in nano seconds for limited load for different algorithm. The fig 4 shows the graphical analysis of different sorting in average case.

Table 4: Sorting algorithm's execution time for all type of loads.

Sorting Algorithm	Limited Load	Moderate Load	High Load
Selection	4.357752667	26.5228	155.7115
Insertion	4.853819867	45.19786	140.8683
Bubble	4.00915	28.98928	261.9369
Merge	4.063407733	46.8318	153.539
'Heap	4.153619233	27.11236	182.0036

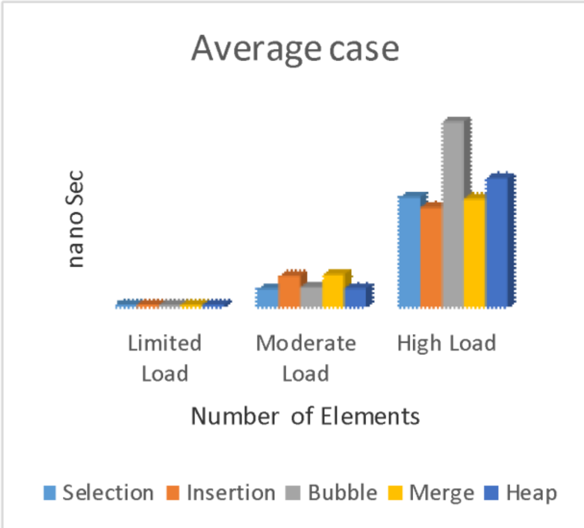

Fig 4: Graphical analysis of different sorting algorithm in all types of loads.

5. Conclusion

In this research paper we compared various sorting algorithm in respect to their execution time. The efficiency of every algorithm varies with the number of input and we have compared the execution of algorithm so that we can could which algorithm is best to use. The sorting algorithms are evaluated in JAVA. From our experimental results we identified that in limited amount of load the bubble sort behaves optimal. But as the load increases to moderate load selection sort behaves better than others. In high load elements insertion sort performs satisfactorily. Practically we also identified that recursive method to sort algorithm takes more time for execution.

6. References

[1] Yan Weimin, WuWeimin,"Data Structures" in, Beijing: Tsinghua University Press., pp. 263-278, 2000.

[2] Zhang Yiwen, Tan Ji, "Analysis and improvement on simple selection sort algorithm", *Silicon Valle*, no. 18, pp. 77-94, 2009.

[3] Weik M.H. (2000) merge sort. In: Computer Science andCommunications Dictionary. Springer, Boston, MA

[4] Kalicharan N. (2014) Advanced Sorting. In: Advanced Topics in Java. Apress, Berkeley, CA

[5] Min Wang and Yunfei Li, "Designing on a Special Algorithm of Triple Tree Based on the Analysis of Data Structure".

[6] International Conference on Computer Education, Simulation and Modeling (CESM 2011), Proceedings, Part (Communications in Computer and Information Science),423-427.

[7] Geng Guohua, Data Structure—C Language Description, Xi'an: Xi'an Electronic Science and Technology University Press, China, (2006), 228-241.

[8] Xu Xiaokai. Simple Data Structure Tutorial. Tsinghua University Press, Beijing, (1995), 193–196.

[9] Menaria, H.K., Nagar, P., Patel, M. (2020). Tweet Sentiment Classification by Semantic and Frequency Base Features Using Hybrid Classifier. In: Luhach, A., Kosa, J., Poonia, R., Gao, XZ., Singh, D. (eds) First International Conference on Sustainable Technologies for Computational Intelligence. Advances in Intelligent Systems and Computing, vol 1045. Springer, Singapore. https://doi.org/10.1007/978-981-15-0029-9_9

[10] K. C. Giri, M. Patel, A. Sinhal and D. Gautam, "A Novel Paradigm of Melanoma Diagnosis Using Machine Learning and Information Theory," 2019 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2019, pp. 1-7, doi: 10.1109/ICACCE46606.2019.9079975.

[11] Patel, M., Badi, N., & Sinhal, A. (2019). The role of fuzzy logic in improving accuracy of phishing detection system. International Journal of Innovative Technology and Exploring Engineering, 8(8), 3162-3164.

[12] Patel, M., & Sheikh, R. (2019). Handwritten digit recognition using different dimensionality reduction techniques. International Journal of Recent Technology and Engineering, 8(2), 999-1002.

[13] H. Gupta and M. Patel, "Study of Extractive Text Summarizer Using The Elmo Embedding," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 829-834, doi: 10.1109/I-SMAC49090.2020.9243610.

[14] H. Gupta and M. Patel, "Method Of Text Summarization Using Lsa And Sentence Based Topic Modelling With Bert," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 511-517, doi: 10.1109/ICAIS50930.2021.9395976.

[15] Sen, S., Patel, M., Sharma, A.K. (2021). Software Development Life Cycle Performance Analysis. In: Mathur, R., Gupta, C.P., Katewa, V., Jat, D.S., Yadav, N. (eds) Emerging Trends in Data Driven Computing and Communications. Studies in Autonomic, Data-driven and Industrial Computing. Springer, Singapore. https://doi.org/10.1007/978-981-16-3915-9_27

[16] Ameta, U., Patel, M., Sharma, A.K. (2021). Scrum Framework Based on Agile Methodology in Software Development and Management. In: Mathur, R., Gupta, C.P., Katewa, V., Jat, D.S., Yadav, N. (eds) Emerging Trends in Data Driven Computing and Communications. Studies in Autonomic, Data-driven and Industrial Computing. Springer, Singapore. https://doi.org/10.1007/978-981-16-3915-9_28

[17] Bissa, A., Patel, M. (2021). An Adjustment to the Composition of the Techniques for Clustering and Classification to Boost Crop Classification. In: Singh Pundir, A.K., Yadav, A., Das, S. (eds) Recent Trends in Communication and Intelligent Systems. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-16-0167-5_13

[18] Taunk, Dhruvika and Patel, Mayank, Feature Extraction for an Audio Discrimination between Speech and Music for Better Human and Computer Interaction (January 20, 2021). ICICNIS 2020, Available at SSRN: https://ssrn.com/abstract=3769769 or http://dx.doi.org/10.2139/ssrn.3769769

[19] Taunk, D., Patel, M. (2021). Hybrid Restricted Boltzmann Algorithm for Audio Genre Classification. In: Sheth, A., Sinhal, A., Shrivastava, A., Pandey, A.K. (eds) Intelligent Systems. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-16-2248-9_11

[20] Min Wang, "Analysis on bubble sort algorithm optimization", *2010 International Forum on Information Technology and Applications*, July 2010.

[21] Kowalk W.P. (2011) Insertion Sort. In: Vöcking B. et al. (eds) Algorithms Unplugged. Springer, Berlin, Heidelberg

