ISSN 2347 - 3983
Volume 9. No. 5, May 2021
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter08952021.pdf
https://doi.org/10.30534/ijeter/2021/08952021

Performance Analysis of Heap Sort and Insertion Sort Algorithm

Humaira Ali[1], Haque Nawaz[2], Abdullah Maitlo[3], Inayatullah Soomro[4]
[1] Department of Computer Science, Sindh Madressatul Islam University, Karachi, Sindh, Pakistan, humaira.ali099@gmail.com
[2] Department of Computer Science, Sindh Madressatul Islam University, Karachi, Sindh, Pakistan, hnlashari@smiu.edu.pk
[3]Department of Computer Science, Shah Abdul Latif University, Khairpur Mirs, Sindh, Pakistan, abdullah.maitlo@salu.edu.pk
[4] Department of Mathematics, Shah Abdul Latif University, Khairpur Mirs, Sindh, Pakistan, inayat.soomro@salu.edu.pk

ABSTRACT

In the era of new technology, we have huge amount of data to deal with arranging the huge amount of data has remained a big challenge. This research paper includes two types of sorting algorithm, Heap Sort and Insertion Sort and also their performance analysis on the basis of running time along with their complexity. This paper includes the algorithms and their implementation in Java programming language. For the results of this research study, the comparison of these two sorting algorithms with different type of the data at running time such as Large, Average, and Small. In Large, data pass100 integers in the array. For Average data pass 50 integers in the array and for Small data pass10 integers in the array. It checks that, which sorting technique is efficient according to the input data. Then identifies the efficiency of these algorithms according to this data three cases used that is Best, Average and Worst Case. The result of this analysis is showing with the help of graphs to show that how much time both algorithms take while given the desired output.

Key words: Array, Complexity, Heap Sort, Insertion Sort, Performance, Sorting Algorithm.

1. INTRODUCTION

An algorithm can be used in both mathematical and computer science field. The term sorting algorithm means arranging data in a certain/sequential order [1]. Whereas, both Heap Sort and Insertion Sort algorithms are comparison based algorithms. Sorting algorithms are also classified as complexity of the algorithm which includes (worst, average and best case) according to the size of the input data, and check the stability how much memory is used.

The efficiency of the algorithm depends upon the time and space taken by the given data [2].

Sorting algorithm works as if the input is given like (7, 6, 4, 9, 2, 0, and 1) so it gives us output in sequential order (0, 1, 2, 4, 6, 7, and 9). For the analysis of best case numbers is organized in sequential order and in average case number is organized in unsystematic order, and in worst case number is organized in reducing order.

So it is clear that sorting is important for managing data. Due to their major role in managing data appropriately, many algorithms have been discovered for example, bubble sort, quick sort, merge sort, selection sort etc...

These all algorithms work for different domains. Some of these algorithms are useful for sorting small data, some for large and some for average data. However, depends on their performance and time complexity in programming languages like java[3]. An Insertion Sort performs with order of n^2 and Heap Sort perform the order of nlogn [4].

1.1 Insertion Sort Algorithm Working

As shown in figure 1 insertion sort divides the input data into sorted and unsorted data and then compares the unsorted part with the sorted data, if the data in the unsorted part is less than sort that data is in its accurate position[5] [6].

Figure 1: Insertion Sort Working Process

1.2 Insertion Sort Algorithm

Take Var l, j ,tmp;

For l = 1 to the length of the array

Assign temp = a[l];

k = l;

While (j > 0 && a[j - 1] > tmp)

Do Assign a[j] = a[j - 1]

Assign j=j-1

A[j] = tmp;

End of while loop

End of for loop[7]

1.3 Heap Sort Algorithm

Heap Sort is an improved sort algorithm of selection sort [8]. This is performed on the heap data and heap is basically the complete binary tree. There are 2 basic natures of heap that is max and min [9]. In maxi heap the parental node is bigger than its child and in mini heap the parental node is fewer than its child [8]. Figure 2 shows the working process of heap sort.

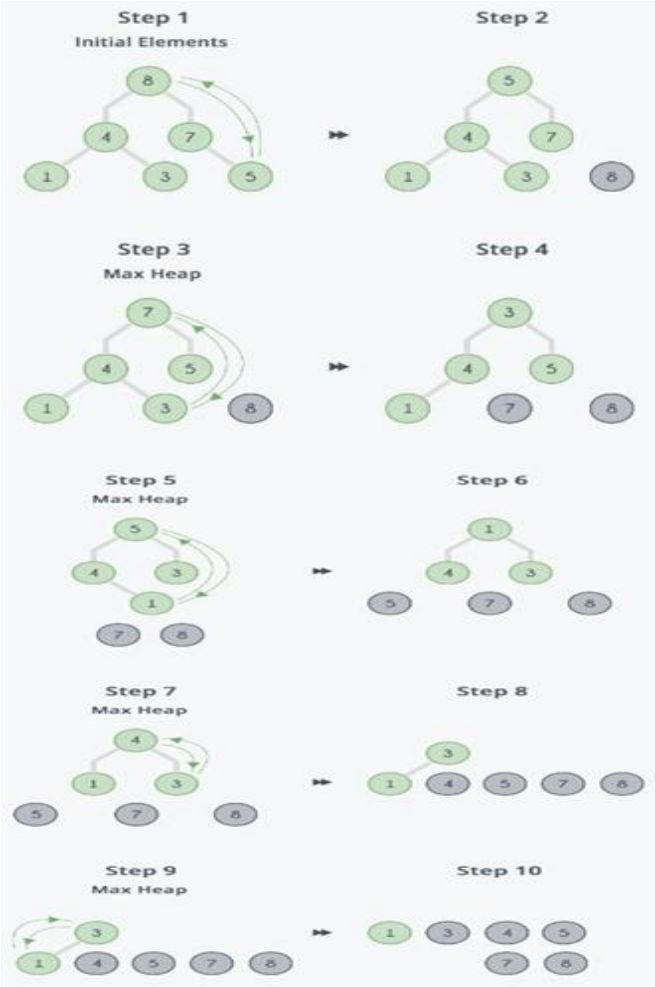

Figure 2 - Heap Sort Working Process

1.4 Heap Sort Algorithm

mx-heapfy (b, j)

l left(j)

r right(j)

if l heap-size[b] and b[l] > b[j]

then largest j

if r heap-size[b] and b[r] > b[largest]

then largest r

if largest j

then exchange b[j] b[largestl]

max-heapfy(b, largest)[7]

2. LITERATURE REVIEW

According to V.P.Kulalvaimozhi et al, while comparing an algorithm after executed, it uses the computer's Central Processing Unit (CPU) to perform operations and its memory to hold the program and data. Researcher used different sorting algorithms and then perform operations on all of them [7][10]. The algorithm which they used for the efficiency of the algorithms are bubble sort, insertion sort, shell sort, binary tree sort, heap sort, quick sort, merge sort and radix sort. In that analysis, the efficiency of the all algorithms were determined by the number of comparison that each algorithm do while running. Every algorithm has different number of comparison so the efficiency of all the algorithms may vary as it depends upon the algorithm used. Whereas, all these algorithms were implemented in C++ language[7].

Matej Hul´ın, provide a comprehensive analysis of the algorithms that include pseudo codes [11], discussion of stability and space and time analysis in the worst, average and best case. Their research is divided in different parts. In the first Part, he described all the algorithms that use to perform work on the algorithm's, the description. It includes how the algorithm works, there pseudo code, and the stability by checking the time and how much the algorithm space takes [2]. Researcher selected C++ to implement the sorting algorithms. The working environment on which these algorithms were tested. According to him the data that used for finding the efficiency of the algorithms has different sequences. Then all the algorithms were measured by dividing them into different parts and included the results of analysis in graphs[11].

Gaurav Kocher, Nikita Agrawal, provide a comparison based sorting algorithms. Perform the analysis of these comparison based algorithm for same number of data. Also mention that each and every application has an algorithm and the efficiency of the application can depend on these algorithms.

Also perform the analysis on both the Central Processing Unit (CPU) and Graphics Processing Unit GPU. They worked on insertion sort, bubble sort, quick sort, and heap sort and merge sort[10] [12]. They concluded that input in integer takes less time as compare to strings. According to their findings, selection sort and quick sort was not be used for large types of data[13].

Dr. I. Lakshmi, perform analysis on four different types of algorithm such as Quick sort, insertion sort, heap sort, merge sort. Researcher analyzed the time complexity [14], [15]-[18] The analysis of the study is to define which algorithm is useful when we have a confusing set of data. The analysis is depending upon best, average and worst case. In the analysis researcher used C# environment and the data was randomly used [10].

3. METHODOLOGY

This research paper includes 3 phase. The first phase includes both the algorithms implemented in java language, in the second phase finding the complexity of both algorithms is explained and the third phase compare and analyze the result of both algorithms and also implement them in java programming

The application both algorithm is implemented in java. Moreover, in second phase pass different types of data and check the complexity and showing the analysis with the help of graph. The third phase check the running time of both algorithms and then compare them on the basis of type of the data. The overall process of this study work is mentioned in three phases. Figure 3 represents the phases of study.

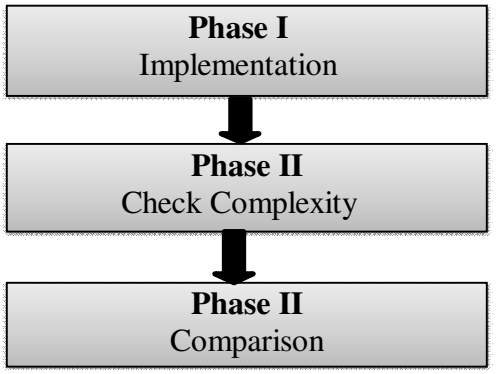

Figure 3: Phases of Study

These all three process consists on different methods and gives a result to move forward in the next step, and then find according to the graph that which is the best algorithm. Moreover, Figure 4 shows the working mechanism of both sorting algorithms.

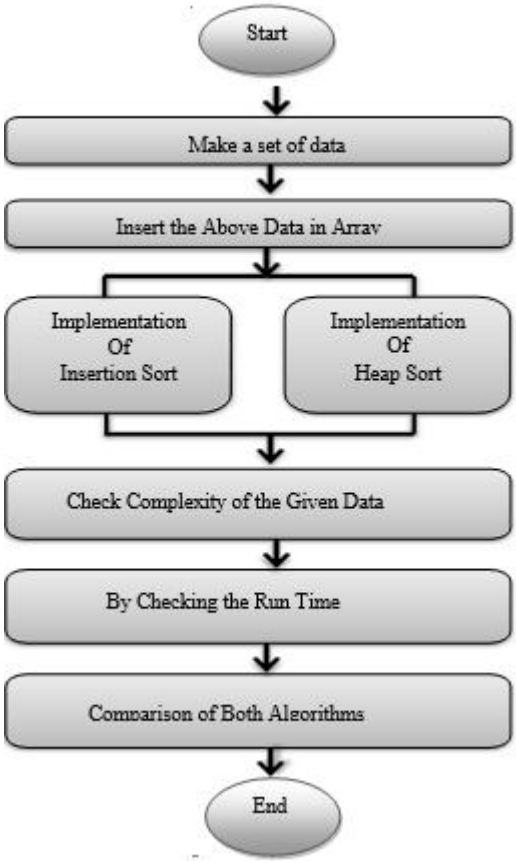

Figure 4: Flowchart shows Working Mechanism of both Sorting Algorithms

4. IMPLEMENTATION AND RESULT ANALYSIS

The analysis of this study is divided into three min phses.in the first phase the implementation is done in java language, in second phase check the complexity by using three strategies that include best, average and worst case. In the last phase the comparison of both heap sort and insertion sort is done.

4.1 Phase I (Implementation):

Both the algorithms insertion sort and heap sort is implemented in java programming language. And the data is based on arrays. In both algorithms three types of data is tested that is (Small, Average and large data) that is the length of the array. These three types of data set are tested on three cases best, average and worst case. Table 1 presents data size.

Table 1: Data Set Size

Data	Array Size
Small	10
Average	50
Large	100

4.2 Phase II (Check Complexity):

In this phase, the complexity can be check in three ways:

1. Best Case
2. Average Case
3. Worst Case

So according to these cases in best case the data in array is arranged in sequential order, in average case the data in array is arranged in random order and in worst case the data in array is arranged in decreasing order.

Analysis for Best Case:

The running time is depending on these input data. The running time of all these three cases is different in both insertion sort and heap sort. Some data can take too much time and some data take less time. So according to the running time check the complexity of these algorithms that which algorithm is efficient for which type of data. Figure 5 presents small data in best case, Figure 6 presents average data in best case, and Figure 7 presents large data in best case.

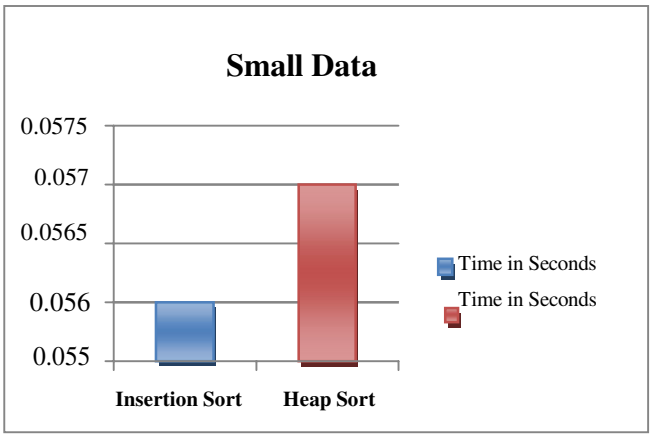

Figure 5: Represent Small Data In Best Case.

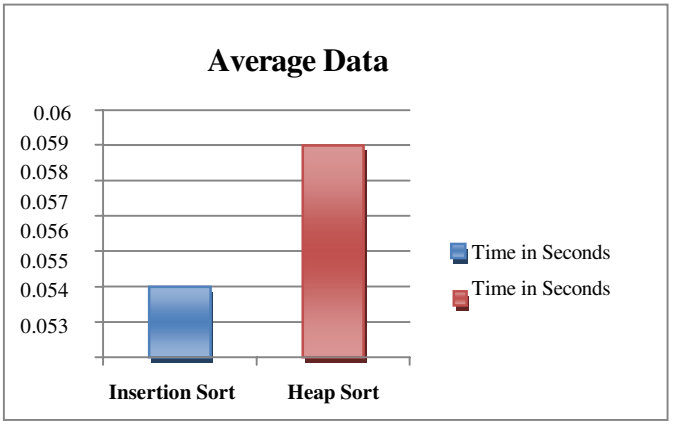

Figure 6: Represent Average Data In Best Case.

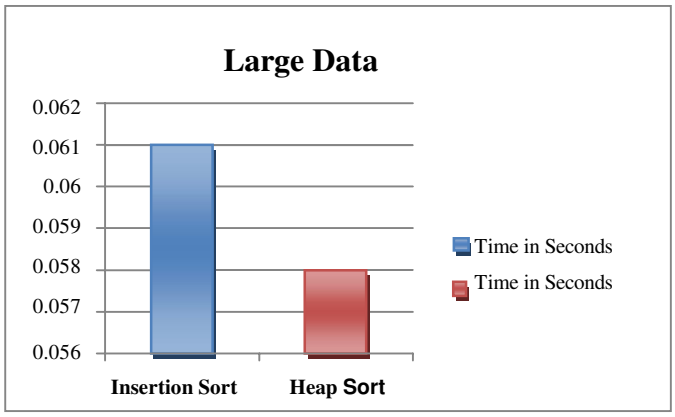

Figure 7: Represent Large Data In Best Case.

In best case the insertion sort performance is good for small and average data and for large data the performance of heap sort is best.

Analysis for Average Case:

In the analysis of average case the data that is tested is in unsystematic form. The analysis can be performing on large, small and average set of data. Figure 8 presents small data in average case, Figure 9 presents average data in average case, and Figure 10 presents large data in average case.

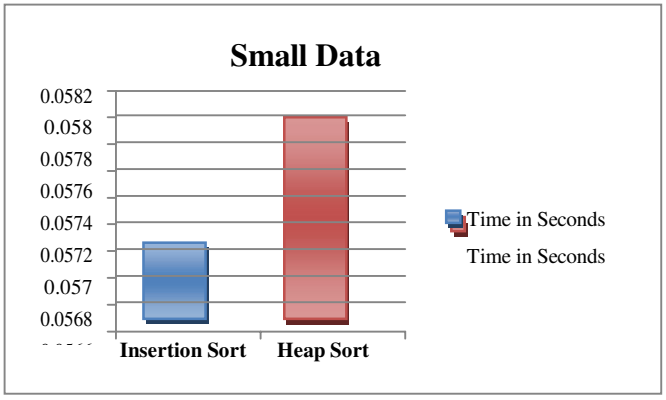

Figure 8: Represent Small Data In Average Case.

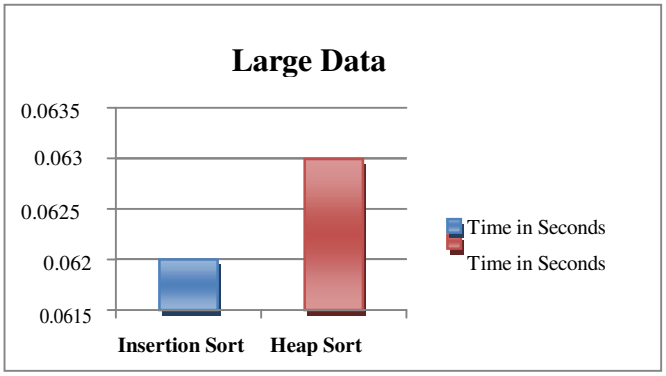

Figure 9: Represent Large Data In Average Case.

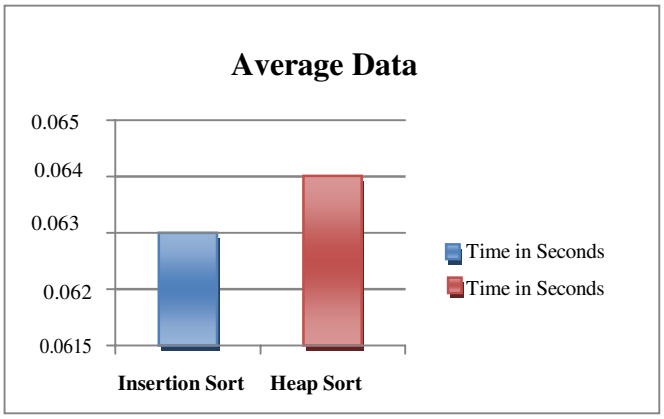

Figure 10: Represent Large Data In Average Case.

In average case the insertion sort works best for all types of data as compared to heap sort.

Analysis for Worst Case:

In the analysis of worst case the data that is tested is in decreasing form. The analysis can be performing on large, small and average set of data. Figure 11 presents small data in worst case, Figure 12 presents average data in worst case, and Figure 13 presents large data in worst case.

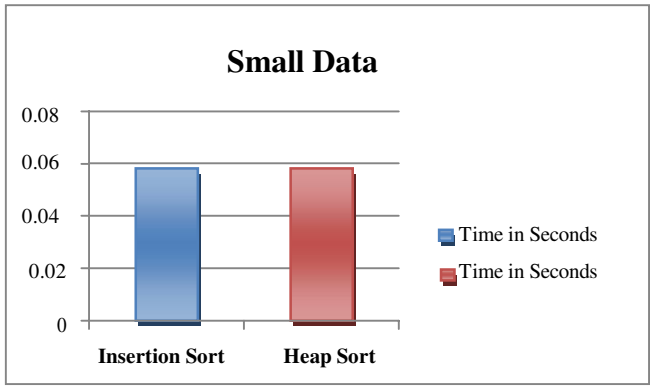

Figure 11: Represent Small Data In Worst Case.

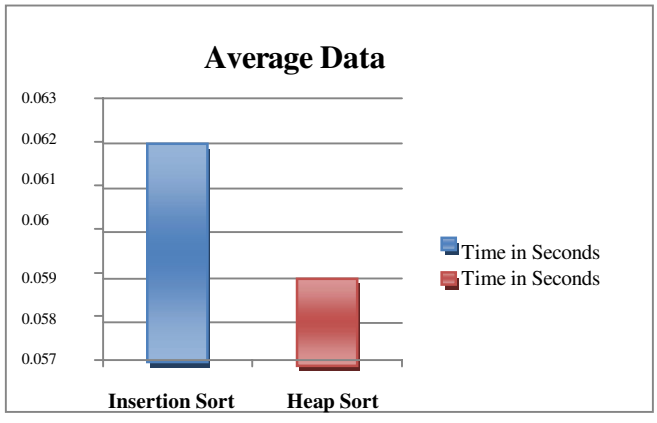

Figure 12: Represent Average Data In Worst Case.

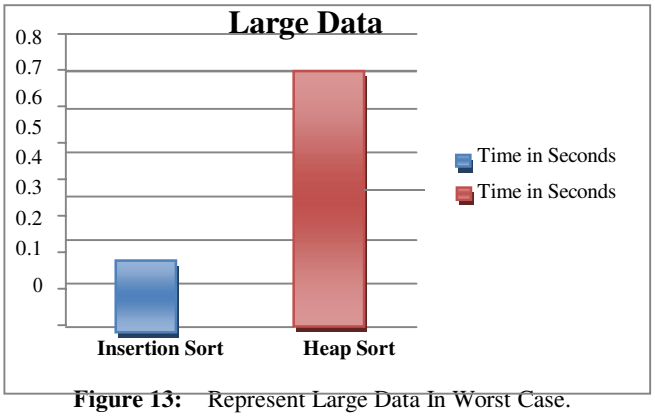

Figure 13: Represent Large Data In Worst Case.

4. 3 Phase III (Comparison):

In this phase the comparison of heap and insertion sort is performed by using the best, average and worst case on small, large and average data set. Figure 14 presents the comparison of insertion and heap sort on small data set. Figure 15 presents the comparison of insertion and heap sort on average data set, and Figure 16 presents the comparison of insertion and heap sort on large data set.

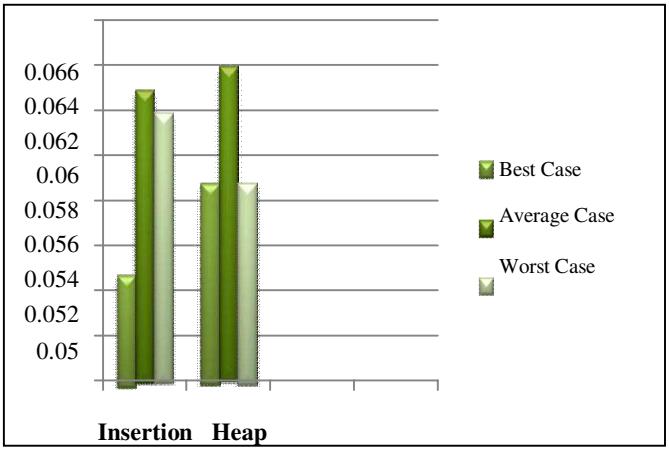

Figure 14: Represent The Comprison of Insertion and Heap Sort on Small Data Set.

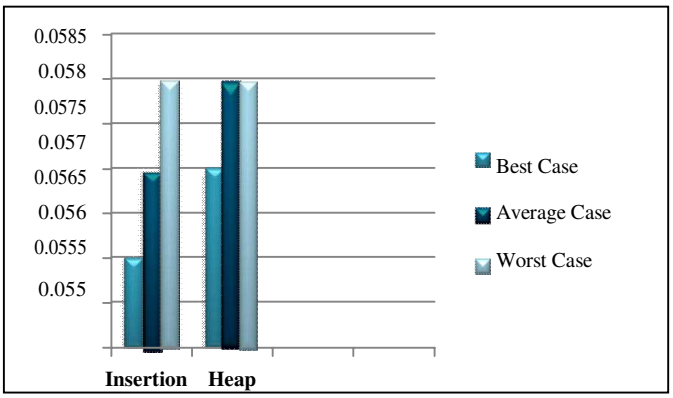

Figure 15: Represent The Comprison of Insertion and Heap Sort on Average Data Set.

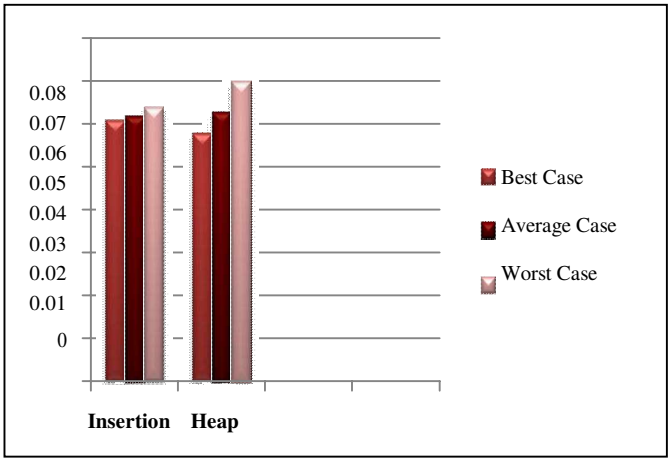

Figure 16: Represent The Comprison of Insertion and Heap Sort on Large Data Set.

5. RESULTS DISCUSSION

The analysis of this study is divided into three main phases. In the first phase the implementation is done in java language, in second phase check the complexity by using three strategies that include best, average and worst case. In the last phase the comparison of both heap sort and insertion sort is done.

Now, according to the analysis the result of the running time that insertion sort takes in small data set is less than the heap sort, and in average data the insertion sort work best but in worst case heap the performance of heap sort is better and in large data the performance of heap sort is good rather than insertion sort.

According to these conditions it is clear that insertion sort takes less time in best and average case for small and average data. But for large data the performance of heap sort is good, so for large data use heap sort algorithm to save our time and complete the tasks with in minimum time.

6. CONCLUSION

In the above analysis and comparison of both insertion sort and heap sort the insertion sort has fast running time for small and average data and for large data insertion sort takes too much time so according to this analysis insertion sort is use for small and average data. Heap sort takes less time than insertion sort in average and large data for both in average and worst case and it has smaller difference in running time for large data set so heap sort is best for large data in every case. The time complexity of insertion sort in best case is O (n) because compare each element if it is not sorted then put in the correct position. And for worst and average case the complexity is O (n^2) because when the input no in 'n' so take n comparison+ n no of movements. Then write this 2(n) that is 2(1+2+3…+n) so the sum of natural nois equal to 2n (n+1)/2 now the remaining value is n (n+1) so highest order is O (n^2). The complexity of heap sort is O (n log n) for all the cases. Because the time complexity of building a heap is O (n) and n-1 call heapify that takes O (logn) and the complete time complexity is O (n log n).

REFERENCES

[1] P. K. Chhatwani, "Insertion Sort with its Enhancement," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 3, pp. 801–806, 2014.

[2] E. S. Al-jaloud, H. A. Al-aqel, and G. H. Badr, "Comparative Performance Evaluation of Heap-Sort and Quick-Sort Algorithms," *Int. J. Comput. Acad. Res.*, vol. 3, no. 2, pp. 39–57, 2014.

[3] S. Grover, "Performance Analysis of Heap, Merge, and Insertion Sort," in *The 20th Winona Computer Science Undergraduate Research Symposium*, 2019, pp. 18–22.

[4] M. A. Bender, M. Farach-Colton, and M. A. Mosteiro,"Insertion Sort is O(n log n)," *Theory Comput Syst*, vol. 39, no. 3, pp. 391–397, Jun. 2006, doi: 10.1007/s00224-005-1237-z.

[5] C. C. Lwin, N. Z. Moe, and P. N. Wai, "A Study of Comparison for Sorting Algorithms Based on Data Sequences," *Univ. J. Creat. Innov. Res.*, vol. 01, no. 01, p. 4, 2020.

[6] M. A. Bender, M. Farach-Colton, and M. A. Mosteiro, "Insertion Sort is O(n log n)," *Theory Comput. Syst.*, vol. 39, no. 3, pp. 391–397, Jun. 2006.

[7] V. P. Kulalvaimozhi, M. Muthulakshmi, R. Mariselvi, G. S. Devi, C. Rajalakshmi, and C. Durai, "Performance analysis of sorting algorithm," *Int. J. Comput. Sci. Mob. Comput.*, vol. 4, no. 1, pp. 291–306, 2015.

[8] V. Sharma, S. Singh, and D. K. S. Kahlon, "Performance Study of Improved Heap Sort Algorithm and Other Sorting Algorithms on Different Platforms," *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 4, pp. 101–105, 2008.

[9] Z.-G. Zhu, "Analysis and Research of Sorting Algorithm in Data Structure Based on C Language," *J. Phys. Conf. Ser.*, vol. 1544, no. 2020, pp. 1–5, May 2020, doi: 10.1088/1742-6596/1544/1/012002.

[10] D. I. Lakshmi, "Performance Analysis of Four Different Types of Sorting Algorithms using Different Languages," *International Journal of Trend in Scientific Research and Development*, vol. Volume-2, no. Issue-2, Jan. 2018.

[11] M. Hulín, "Performance analysis of Sorting Algorithms," Thesis, Masaryk University Faculty of Informatics, Pole-Ponava, Czechia, 2017.

[12] R. Schaffer and R. Sedgewick, "The Analysis of Heapsort," *Journal of Algorithms*, vol. 15, no. 1, pp. 76–100, Jul. 1993, doi: 10.1006/jagm.1993.1031.

[13] P. Prajapati, N. Bhatt, and N. Bhatt, "Performance Comparison of Different Sorting Algorithms," p. 3, 2017.

[14] R. Schaffer and R. Sedgewick, "The Analysis of Heapsort," *Journal of Algorithms*, vol. 15, no. 1, pp. 76–100, Jul. 1993, doi: 10.1006/jagm.1993.1031.

[15] I. Ali, H. Nawaz, I. Khan, A. Maitlo, M. Ameen, and M. Malook, "Performance Comparison between Merge and Quick Sort Algorithms in Data Structure," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 11, 2018, doi: 10.14569/IJACSA.2018.091127.

[16] S. M. Aqib, H. Nawaz, and S. M. Butt, "Analysis of Merge Sort and Bubble Sort in Python, PHP, JavaScript, and C language," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 2, pp. 680–686, Apr. 2021, doi: 10.30534/ijatcse/2021/311022021.

[17] M. A. Hingoro and H. Nawaz, "A Comparative Analysis of Search Engine Ranking Algorithms," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 2, pp. 1247–1252, Apr. 2021, doi: 10.30534/ijatcse/2021/1081022021.

[18] F. A. Agha and H. Nawaz, "Comparison of Bubble and Insertion Sort in Rust and Python Language," *IJATCSE*, vol. 10, no. 2, pp. 1020–1025, Apr. 2021, doi: 10.30534/ijatcse/2021/761022021.