Sorting Algorithms – A Comparative Study

Muhammad Idrees
Government College University Faisalabad
5 PUBLICATIONS **9** CITATIONS

Sorting Algorithms – A Comparative Study

Naeem Akhter, Muhammad Idrees, Furqan-ur-Rehman
naeemdiroffice@yahoo.com, muhammad.idrees@cs.uol.edu.pk, furqan.uca@gmail.com

Abstract – Sorting is nothing but alphabetizing, categorizing, arranging or putting items in an ordered sequence. It is a key fundamental operation in the field of computer science. It is of extreme importance because it adds usefulness to data. In this papers, we have compared five important sorting algorithms (Bubble, Quick, Selection, Insertion and Merge). We have developed a program in C# and experimented with the input values 1-150, 1-300 and 1-950. The performance and efficiency of these algorithms in terms of CPU time consumption has been recorded and presented in tabular and graphical form.

Key words: Quick Sort, Insertion Sort, Bubble Sort, Merge Sort, Selection Sort, stopwatch.

1. Introduction

Arranging or sorting things or items is not an overnight development. Its footprints can be traced back in 7th century BCE. Abdul Wahab and O.Issa (2009) state that King of Assyria used the idea of arranging clay tablets for Royal Library sorted according to their shape[1]. Sorting is not a jaguar leap but it has emerged in parallel with the development of human mind. In computer science, alphabetizing, arranging, categoring or putting data items in an ordered sequence on the basis of similar properties is called sorting. Sorting is of key importance because it optimizes the usefulness of data. We can observe plenty of sorting examples in our daily life, e.g. we can easily find required items in a shopping maal or utility store because the items are kept categorically. Finding a word from dictionary is not a tideous task because all the words are given in sorted form. Similarly, finding a telephone number, name or address from a telephone directory is also very easy due to the blessings of sorting. In computer science, sorting is one of the most important fundamental operations because of its pivotal applications. Priority scheduling and shortest job first scheduling are examples of sorting. Thomas Cormen, Charlese Ronald, Clifford Stein and Rivest Leiserson (2001) state "An algorithms is any well-defined computational procedure that takes some value, or set of values, as input and produces some value or set of values as output"[2]. A number of sorting algorithms are available with pros and cons. Alfred Aho, John Hopcroft and Jeffrey Ullman (1974) has classified algorithms on the basis of computational complexity, number of swaps, stability, usage of extra resources and recursion [3]. The items to be sorted may be in various forms i.e. random as a whole, already sorted, very small or extremely large in numer, sorted in reverse order etc. There is no algorithm which is best for sorting all types of data. We must be familiar with sorting algorithms in terms of their suitability in a particular situation. In this paper we are going to compare five (Bubble, Quick, Insertion, Selection and Merge) sorting algorithms for their CPU time consumption on a given input in best, worst and average cases. Rest of the paper comprises of: 2. Related Work 3. Methodology 4. Experiments 5. Results and Discussion
6. Conclusion 7. Future Work 8. References.

2. Related Work

Jehad Alnihoud and Rami Mansi (2010) state that sorting is graded as a fundamental problem in computer science[4]. Sonal Beniwal and Deepti Groover (2013), after comparing Bubble, Heap, Insertion, Merge and Quick sort algorithms have concluded that quality of a good sorting algorithm is not only the speed but other factors like, length, code complexity, stability, performance consistency and data type handling should also be taken into consideration[5]. Rohit Joshi etc (2013) in [6] have analyzed the time complexity of Bucket Sort, Counting Sort, Radix Sort on input 1,2,3….10 and concluded that non-comparison based algorithms are better by O(n) instead of comparison based O(n log n). Aliyu

Ahmed and Dr. Zirra (2013) in [7] have compared Insertion and Quick sort algorithms on both integer and character arrays concluded that performance of insertion short is better than quick sort. Both algorithms are for sorting small number of items. CPU time consumption while sorting integer arrays is very low as compared to character arrays. In [8] Nidhi Chhajed and Simarjeet Singh Bhatia (2013) have compared Quick, Heap and Insertion Sort algorithms in terms of time complexity and various performance factors. Random numbers between 10,000 to 30,000 have been used as input data. They have concluded that insertion sort algorithm is slower than heap and quick sort. All of them have worst case time complexity as N^2. But quick sort performed better than the other two. As the input increases insertion sort performs very poorly and shows exponential growth. In [9] Ashutosh Bharadwaj and Shailendra Mishra (2013) have compared Insertion, Bubble, Selection, Merge and Index Sort algorithms on 10, 100 and 10001000 inputs and concluded that performance of Index Sort is better on all input values. It consumes lesser CPU time than others on small input. It takes more CPU time than Selection, Merge and Bubble Sort for larger input. Jehad Hammad (2015) states in his comparative study on HornerEval, Linear Search, Towers, Binary Search, Insertion, Max, Min, MaxMin, Merge, Quick, SelectionSort, Heap, Bubble and Gnome Sorting algorithms on 5000, 10000, 20000 and 30000 input values that Gnome sorting algorithm is the quickest in best case. Selection sort is quicker than bubble sort and gnome sort on random data. He has further analyzed a drawback of selection sort which continues sorting the items it they are already arranged, while gnome and bubble sort algorithms swap the items if required[10]. In [11] Gaurav Kumar etc (2013) have compared Bubble, Insertion, Quick, Merge and Heap sorting algorithms on 100000, 300000, 500000, 700000, 1000000, 1500000 input values. After comparing the data empirically the algorithms were ranked in the following order on the basis of their speed.

1. Merge
2. Quick
3. Heap
4. Insertion
5. Bubble

Merge, Quick and Heap sort algorithms are faster than the remaining two when the input size is very large. In [12] Pankaj Sareen (2013) has compared Bubble, Insertion, Selection, Merge and Quick sort algorithms on the input values 10, 100, 1000 and 10000. The comparison was based on average case only. The average running time of all the algorithms was noted on these inputs and presented graphically. It was concluded that the most efficient algorithm was Quick Sort. In [13 & 14] several studies (Thomas Cormen, et. al. 2009; Juliana Pena Ocampo, 2008) suggest state that all algorithms perform $O(n^2)$ in their worst case, Quick Sort is the only algorithm which have average and best runtime of O(nLogn). It means that Quick Sort is better than Insertion, Bubble and Selection sort in average case, for sufficiently large input.

3. Methodology

Empirical comparison is always machine dependent. It is essential to explicitly describe the machine used for experiments in particular to facilitate the researchers who intend to reverify the results. A program developed in C# has been used for calculation of CPU time taken by each algorithm. The data set used for this purpose consists of 1-150, 1-300, and 1-950 in best, worst and average cases. The program was executed on Windows 7 (64 bit), Service Pack 1, Computer used for this purpose was CPU T7100 @ 1.80 GHz, Intel(R) Core(TM)2 Duo. Memory installed was 2.00 GB. Consumption of CPU time from all the algorithm was noted using *Stopwatch* after running the program on all the inputs. The results were calculated after tabulation and then their graphical representation was developed using MS Excl.

4. Experiments & Results

Stopwatch provides high accuracy when used for comparison of algorithms for their efficiency. But it is not 100% accurate. In [15] Thomas Maierhofer (2010) states that *Stopwatch* may provide results 25%-30% different for the same code excuted repeatedly on the same machine. We have run our program 5-times on the same input and have taken the average of 5-results to achieve better accuracy as in [16] Pankaj Sareen (2013) has run his program five times on the same input to calculate average running time.

TABLE – 1
Comparison on input 1 to 150 Best Case

Algo	Time Consumption in microseconds on input Value 1-150 (Best Case)					
	1	2	3	4	5	Average
Bubble	2260	2301	2106	2460	2377	**2300.80**
Quick	2941	3261	2897	3018	2927	**3008.80**
Selection	1845	2389	2279	2003	1865	**2076.20**
Insertion	1467	2018	1570	1370	1429	**1570.80**
Merge	3763	4525	4248	5695	4510	**4548.20**

Graph – 1
Comparison on input 1 to 150 Best Case

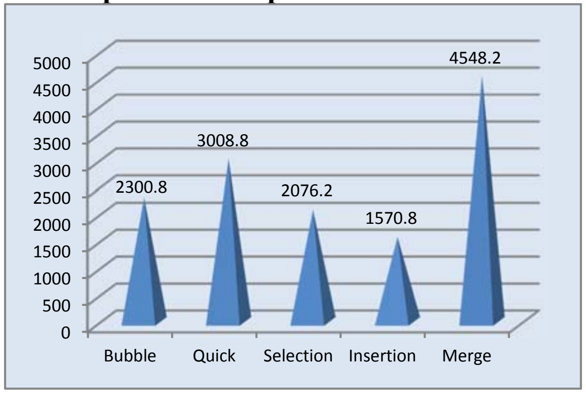


TABLE – 2
Comparison on input 1 to 150 Worst Case

Algo	Time Consumption in microseconds on input Value 1-150 (Worst Case)					
	1	2	3	4	5	Average
Bubble	2846	2876	2724	2719	2901	**2813.20**
Quick	2946	3117	3052	3467	3298	**3176.00**
Selection	1904	1909	2262	2018	2216	**2061.80**
Insertion	1744	2108	1919	1800	1766	**1867.40**
Merge	4708	3877	4077	4081	4130	**4174.60**

Graph – 2
Comparison on input 1 to 150 Worst Case

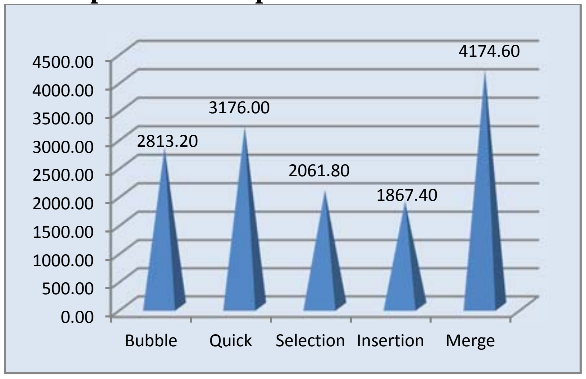
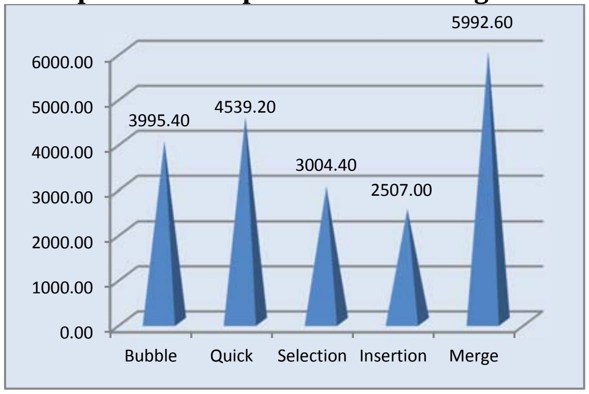


TABLE – 3
Comparison on input 1 to 150 Average Case

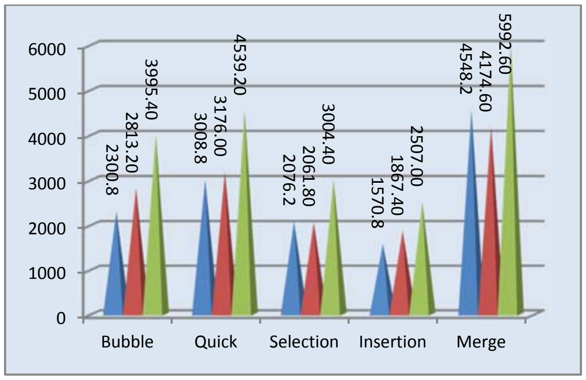
Algo	Time Consumption in microseconds on input Value 1-150 (Average Case)					
	1	2	3	4	5	Average
Bubble	4317	4070	3795	3769	4026	3995.40
Quick	4442	4311	4205	4085	5653	4539.20
Selection	2845	2910	2847	2791	3629	3004.40
Insertion	2386	2365	2565	2321	2898	2507.00
Merge	5579	5664	5734	6183	6803	5992.60

Graph – 3
Comparison on input 1 to 150 Average Case



The results are stating that on input 1 to 150 Insertion Sort algorithm performed bettern than all others in best, worst and average cases. Similarly, performance of Merge Sort was noted as worst amongst all. Performance of Bubble Sort was acceptable, it remained at No. 2 in worst and average cases. Quick Sort remained at No. 4 in all cases. Selection sort can be ranked No. 2 out of 5. Following is the comparison graph showing each algorithms' performance behavior on all three cases (best, worst and average) on input 1-150.

Comparison Graph: 1-3



Page - 3

TABLE – 4
Comparison on input 1 to 300 Best Case

Algo	Time Consumption in microseconds on input Value 1-300 (Best Case)					
	1	2	3	4	5	Average
Bubble	3741	3746	3419	3582	3642	**3626.00**
Quick	4452	4159	3977	3975	4098	**4132.20**
Selection	2956	2627	2699	2627	2782	**2738.20**
Insertion	1323	1319	1333	1324	1616	**1383.00**
Merge	4602	5014	4334	4311	4729	**4598.00**

Graph – 4
Comparison on input 1 to 300 Best Case

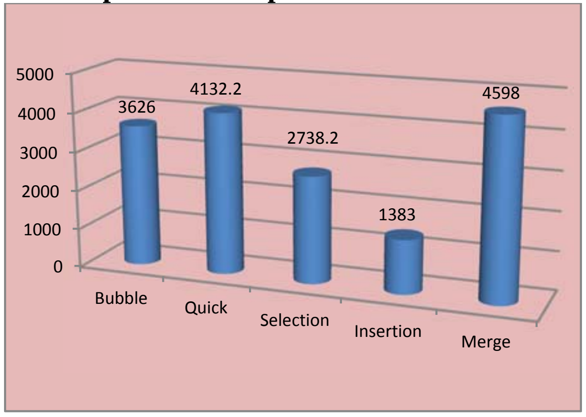


TABLE – 5
Comparison on input 1 to 300 Worst Case

Algo	Time Consumption in microseconds on input Value 1-300 (Worst Case)					
	1	2	3	4	5	Average
Bubble	5438	4959	4901	4703	4854	**4971.00**
Quick	4080	4033	5582	4155	3970	**4364.00**
Selection	2974	2975	2994	3300	2852	**3019.00**
Insertion	3418	3197	3339	3372	3043	**3273.80**
Merge	5301	4491	4835	4645	5177	**4889.80**

Graph – 5
Comparison on input 1 to 300 Worst Case

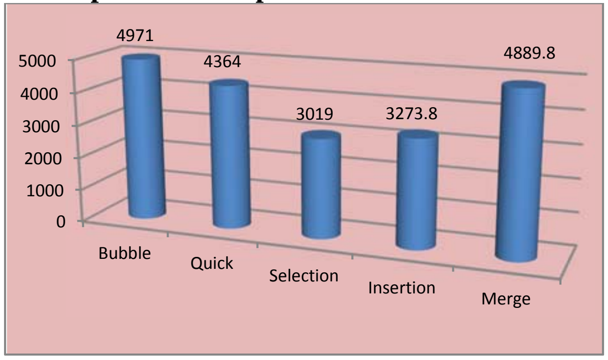
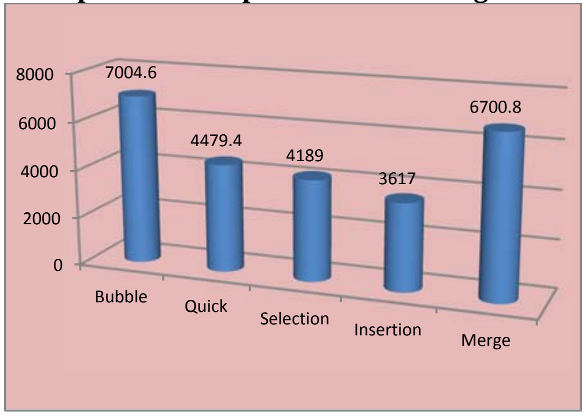


TABLE – 6
Comparison on input 1 to 300 Average Case

Algo	Time Consumption in microseconds on input Value 1-300 (Average Case)					
	1	2	3	4	5	Average
Bubble	6774	6882	7091	7341	6935	7004.60
Quick	4530	4632	4507	4513	4215	4479.40
Selection	4458	4171	4195	4021	4100	4189.00
Insertion	3852	3505	3485	3314	3929	3617.00
Merge	6571	7140	6305	6850	6638	6700.80

Graph – 6
Comparison on input 1 to 300 Average Case



The results are verifying that on the data from 1 to 300 the performance of Bubble Sort remained poorest amongst all. It consumed more time than all others for best, worst and average cases. Insertion Sort was best of the lot in best case (1383 microseconds) and average case (3617 microseconds) but Selection Sort left behind others in worst case (3019 microseconds). Quick Sort can be ranked No. 3 out of 5. Following is the comparison graph of all cases (input 1-300).

Comparison Graph: 4-6

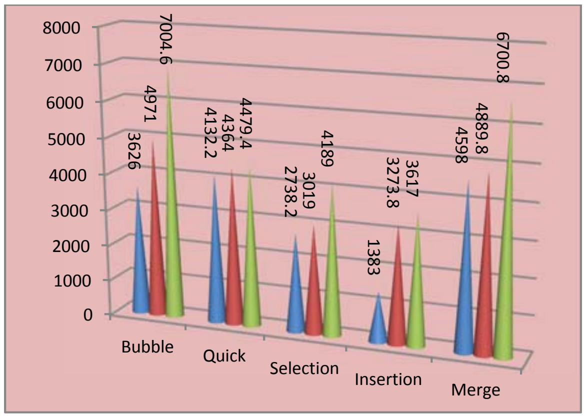


TABLE – 7
Comparison on input 1 to 950 Best Case

Algo	Time Consumption in microseconds on input Value 1-950 (Best Case)					
	1	2	3	4	5	Average
Bubble	21744	23575	18279	21566	19441	**20921**
Quick	17857	18492	17946	18682	19954	**18586**
Selection	11998	12301	12884	12054	12414	**12330**
Insertion	1369	1812	1639	1400	1465	**1537**
Merge	15137	15161	11863	15196	15399	**14551**

Graph – 7
Comparison on input 1 to 950 Best Case

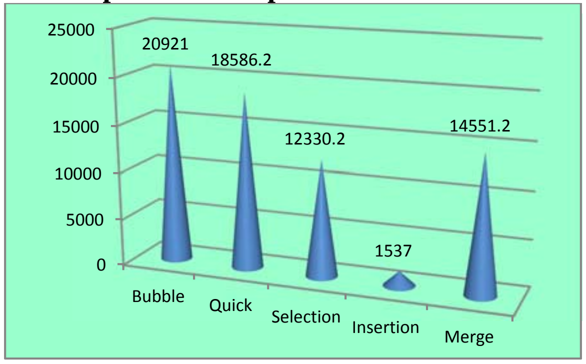


TABLE – 8
Comparison on input 1 to 950 Worst Case

Algo	Time Consumption in microseconds on input Value 1-950 (Worst Case)					
	1	2	3	4	5	Average
Bubble	34646	38896	36366	33158	39631	**36539.4**
Quick	16995	17836	18877	16575	18050	**17666.6**
Selection	15569	14590	15392	14702	14508	**14952.2**
Insertion	19747	20374	21884	19465	20996	**20493.2**
Merge	11629	13399	14876	11387	12917	**12841.6**

Graph – 8
Comparison on input 1 to 950 Worst Case

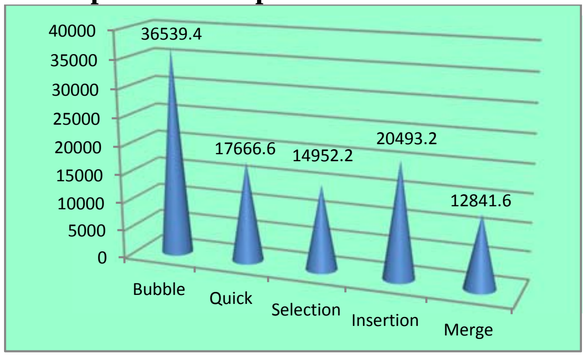
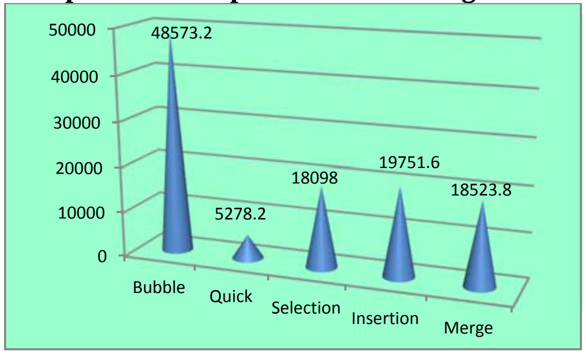


TABLE – 9
Comparison on input 1 to 950 Average Case

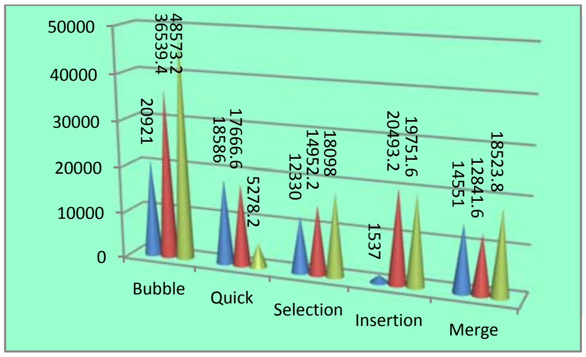
Algo	Time Consumption in microseconds on input Value 1-950 (Average Case)					
	1	2	3	4	5	Average
Bubble	44499	57172	45139	44252	51804	48573.20
Quick	5013	5609	5332	5034	5403	5278.20
Selection	17906	18356	17925	18104	18199	18098.00
Insertion	15427	25973	26047	15326	15985	19751.60
Merge	17672	23301	17500	17310	16836	18523.80

Graph – 9
Comparison on input 1 to 950 Average Case



The results are presenting the facts that on the input from 1 to 950 Bubble Sort consumed most time than all other competitors in all cases. It took 21921.20, 36539.40 and 48573.20 microseconds for best, worst and average cases respectively. Insertion Sort consumed only 1570.80 micrseconds in best case (least than all others). In worst case, Selection Sort performed best, took 14952.20 microseconds. In average case, Quick Sort was significantly the best which consumed only 5278.2 microseconds.

Comparison Graph: 7-9



5. Discussion

We have made a comparative study of five sorting algorithms (Bubble, Quick, Insertion, Selection and Merge) on input values 1-150, 1-300 and 1-950. We run our program for calculating best, worst and average cases. After performing experiments on our pre-defined data set (1-150, 1-300 and 1-950) the algorithms performed varyingly. At a small input of 1-150 Insertion Sort performed the best, Merge Sort was poorest amongst all at the same input. Performance of Bubble Sort was also not bad. It performed better than Selection, Insertion and Merge sort in best case. It also performed better than Quick and Merge Sort in worst and average cases. Bubble Sort was proved the worst algorithm when the input was increased to 1-950. Behavior of Merge Sort is almost opposite to Bubble Sort. It was worst of the lot at input 1-150, it performed reasonably well on input 1-300. It was graded best at input 1-950 with worst case. Selection Sort performed significantly well on varying inputs and turned up to remain at position 1 or 2 on all input values in all cases. There is no considerable change in the Selection Sort's performance with the smaller or larger input values. If we analyze the behavior of Quick Sort, it was ranked 4 with input 1-150. It was ranked 4 in best case and ranked 3 in worst and average cases with input 1-300. When the input was increased to 1-950 it was ranked 2 and 3 in best and worst cases respectively but significantly dominated all others in average case. It took 4.79% of the total time consumed by five algorithms in average case with input 1-950. We have thoroughly analyzed the behavior of all the algorithms on our data set in best, worst and average cases. From the results obtained after experiments, we have ranked all the algorithms according to their efficiency on our data set in all the cases(best, worst and average. Following is the ranking of all the algorithms:

Table-10: Ranking of Algorithms (input 1-150)

Input	Rank	Best Case	Worst Case	Average Case
1-150	1	Insertion	Selection	Selection
	2	Selection	Bubble	Bubble
	3	Bubble	Merge	Merge
	4	Quick	Quick	Quick
	5	Merge	Insertion	Insertion

Table-11: Ranking of Algorithms (input 1-300)

Input	Rank	Best Case	Worst Case	Average Case
1-300	1	Insertion	Selection	Insertion
	2	Selection	Insertion	Selection
	3	Bubble	Quick	Quick
	4	Quick	Merge	Merge
	5	Merge	Bubble	Bubble

Table-12: Ranking of Algorithms (input 1-950)

Input	Rank	Best Case	Worst Case	Average Case
1-950	1	Insertion	Merge	Quick
	2	Selection	Selection	Selection
	3	Merge	Quick	Merge
	4	Quick	Insertion	Insertion
	5	Bubble	Bubble	Bubble

6. Conclusion

We concluded that Bubble sort is not a bad choice when input size is small (less than 100) because it is a simple-to-implement algorithm. But as the input size increases, more efficiency is required for sorting. Bubble sort is easily dominated by more efficienct algorithms on larger input size. Merge Sort algorithm is more complex than Bubble Sort. It performs poorly on small input but enhances its performance in parallel with the increment in input size. Insertion Sort algorithm was graded the best of the lot with small inputs in all cases even with large input in best case. Selection Sort was least effected by the variation in input values. Quick Sort is complex to implement, shows poor results on small inputs (less than 100), average results on medium-sized input (more than 100 but less than 500) but signicantly dominates other algorithms in average case with large inputs (more than 500).

7. Future Work

1. More sorting algorithms like Merge Insertion sort, Odd-Even sort, Heap sort, Cycle sort, Cocktail sort, Timsort and Smoothsort can also be included in the comparison using the same input data with the outcome of detailed comparative analysis.
2. In this paper, we compared all sorting algorithms which are comparison-based. Comaparative study of the algorithms which use comparison technique and those which don't use this technique (Counting sort, LSD Radix sort,

MSD Radix sort and Bucket sort, etc) can be carried out.

3. Performance comaprison of the sorting algorithms can be made on the same input with integers and characters separately. All of them can be ranked according to their efficiency on both type of inputs separately with analysis of their behavior on varying input size.

8. References:

[1] Wahab, A., Issa, O.A. Fundamentals of Library & Information Sciences, (1st ed.). Cataloguing-in-Publication Data, Ilorin (2009).

[2] Cormen, T.H., Leiserson, C.E., & Rivest, R.L. Introduction to Algorithms (2nd ed.). Prentice Hall of India private limited, New Delhi-110001 (2001).

[3] Aho A., Hopcroft J., and Ullman J. The Design and Analysis of Computer Algorithms, Addison Wesley (1974).

[4] Alnihoud, J., & Rami, R. An Enhancement of Major Sorting Algorithms, The Intational Arab Journal of Information Technology, Vol. 7, No. 1, January (2010).

[5] Ben, S. & Deepti, Gr. Comparison of Various Sorting Algorithms: A Review, International Journal of Emerging Research in Management & Technology, Vol., 2, Issue 5 (2013).

[6] Joshi, R., Panwar, G.P., Pathak, P. (2013). Analysis of Non-Comparison Based Sorting Algorithms: A Review, International Journal of Emerging Research in Management & Technology.

[7] Ahmed M. Aliyu, Dr. P.B. Zirra. A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays, The International Journal of Engineering and Science (IJES), Vol 2, Issue 7 (2013).

[8] Chhajed, N., & Bhatia, S.S. A Comparison Based Analysis of Different Types of Sorting Algorithms with their Performances, Indian Journal of Research Paripex, Vol. 2, Issue. 3 (2013).

[9] Bharadwaj, A., & Mishra, S. Comparison of Sorting Algorithms based in Input Sequences, International Journal of Computer Applications, Vol. 78, No. 14 (2013).

[10] Hammad, J. A Comparative Study between Various Sorting Algorithms, International Journal of Computer Science and Network Security (IJCSNS), Vol 15, No. 3 (2015).

[11] Kumar, G. & Ghugh, H. Empirical Study of Complexity Graphs for Sorting Algorithms, International Journal of Computer, Communication and Information Technology (IJCCIT), Vol. 1, No. 1 (2013).

[12] Sareen, P. Comparison of Sorting Algorithms (On the Basis of Average Time), International Journal of Advanced Research in Computer Science & Software Engineering, Vol. 3, Issue 3 (2013).

[13] Cormen, T. H., Leiserson, C.E., & Rivest, R.L. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill (2009).

[14] Pena, J. O. An empirical comparison of the runtime of five sorting algorithms, International Baccalaureate Extended Essay (2008).

[15] Thomas Maierhofer Performance Tests: Precise Run Time Measurements with System.Diagnostics.Stopwatch. (2010),

Retrieved from http://www.codeproject.com/Articles/61964/Performance-Tests-Precise-Run-Time-Measurements-wi

[16] Sareen, P. Comparison of Sorting Algorithms (On the Basis of Average Time), International Journal of Advanced Research in Computer Science & Software Engineering, Vol. 3, Issue 3 (2013).