Review on Sorting Algorithms A Comparative Study

Khalid Suleiman Al-Kharabsheh *khalidkh@bau.edu.jo*
Aqaba College , Balqa Applied University
Aqaba, Jordan

Ibrahim Mahmoud AlTurani *Traini111@bau.edu.jo*
Aqaba College , Balqa Applied University
Aqaba, Jordan

Abdallah Mahmoud Ibrahim AlTurani *Traini2001@yahoo.com*
IT college,Jordan University of Science and Technology
Irbid,Jordan

Nabeel Imhammed Zanoon *Dr.nabeel@bau.edu.jo*
Aqaba College , Balqa Applied University
Aqaba, Jordan

Abstract

There are many popular problems in different practical fields of computer sciences, database applications, Networks and Artificial intelligence. One of these basic operations and problems is sorting algorithm; the sorting problem has attracted a great deal of research. A lot of sorting algorithms has been developed to enhance the performance in terms of computational complexity. there are several factors that must be taken in consideration; time complexity, stability, memory space. Information growth rapidly in our world leads to increase developing sort algorithms.a stable sorting algorithms maintain the relative order of records with equal keys This paper makes a comparison between the Grouping Comparison Sort (GCS) and conventional algorithm such as Selection sort, Quick sort, Insertion sort , Merge sort and Bubble sort with respect execution time to show how this algorithm perform reduce execution time.

Keywords: Sort, Grouping Comparison Sort, Quick Sort, Merge Sort, Time Complexity.

1. INTRODUCTION

Sorting is a process of rearrangement a list of elements to the correct order since handling the elements in a certain order more efficient than handling randomize elements [1].Sorting and searching are among the most common programming processes, as an example take database applications if you want to maintain the information and ease of retrieval you must keep information in a sensible order, for example, alphabetical order, ascending/descending order and order according to names, ids, years, departments, etc.

Information growth rapidly in our world leads to increase developing sort algorithms. Developing sort algorithms through improved performance and decreasing complexity, it has attracted a great deal of research; because any effect of sorting algorithm enhancement of the current algorithms or product new algorithms that reflects to optimize other algorithms. Large number of algorithms developed to improve sorting like merge sort, bubble sort, insertion sort, quick sort ,selection sort and others, each of them has a different mechanism to reorder elements which increase the performance and efficiency of the practical applications and reduce time complexity of each one. When comparing between various sorting algorithms, there are several factors that must be taken in consideration; first of them is the time complexity, the time complexity of an algorithm determined the amount of time that can be taken by an algorithm to run [3][7][27]. This factor

different from sorting algorithm to another according to the size of data that we want to reorder, some sorting algorithm inefficient and too slow. The time complexity of an algorithm is generally written in form big O(n) notation, where the O represents the complexity of the algorithm and a value n represent the number of elementary operations performed by the algorithm [8].The second factor is the stability[26], means; algorithm keeps elements with equal values in the same relative order in the output as they were in the input. [2][3][9]. Some sorting algorithms are stable by its nature such as insertion sort, merge sort, bubble sort, while some sorting algorithms are not, such as quick sort, any given sorting algorithm which is not stable can be modified to be stable [3]. The third factor is memory space, algorithm that used recursive techniques need more copies of sorting data that affect to memory space [3][9].Many previous researches have been suggested to enhance the sorting algorithm to maintain memory and improve efficiency. Most of these algorithms are used comparative operation between the oldest algorithm and the newest one to prove that.

2. PERFORMANCE IN AVERAGE CASE BETWEEN SORTING AIGORITHMS

the following studies are previous study on the same research which make a comparative between different type of sorting algorithms:

(Pooja Adhikari,2007) The performance of any computation depends upon the performance of sorting algorithms.Like all complicated problems, there are many solutions that can achieve the same results. This paper choose two of the sorting algorithms among them selection sort and shell sort and compares the various performance factor among them.

(Davide Pasetto Albert Akhriev,2011) In this paper we provide a qualitative and quantitative analysis of the performance of parallel sorting algorithms on modern multi-core hardware. We consider several general-purpose methods,which are widely regarded among the best algorithms available, with particular interest in sorting of database records and very large arrays (several gigabytes and more), whose size far exceed L2/L3 cache.

(ADITYA DEV MISHRA & DEEPAK GARG,2008)Many different sorting algorithms have been developed and improved to make sorting fast. As a measure of performance mainly the average number of operations or the average execution times of these algorithms have been investigated and compared. There is no one sorting method that is best for every situation. Some of the factors to be considered in choosing a sorting algorithm include the size of the list to be sorted, the programming effort, the number of words of main memory available, the size of disk or tape units, the extent to which the list is already ordered, and the distribution of values

This paper implemented of Selection sort, Quick sort, Insertion sort , Merge sort ,Bubble sort and GCS algorithms using C++ programming language, and measure the execution time of all programs with the same input data using the same computer. The built-in function (clock ()) in C++ is used to get the elapsed time of the implementing algorithms, execution time of a program is measured in milliseconds [6].The performances of GCS algorithm and a set of conventional sort algorithms are comparatively tested under average cases by using random test data from size 10000 to 30000. The result obtained is given in Table 1 to Table 6 for each Algorithm and the curves are shown in figure 1.

Selection sort

Selection sorts the simplest of sorting techniques. It's work very well for small files, also It's has a quite important application because each item is actually moved at most once [4]. It has O (n2) time complexity, making it inefficient on large lists. Selection sort has one advantage over other sort techniques[15][16]. Although it does many comparisons, it does the least amount of data moving. That means, if your data has small keys but large data area, then selection sorting may be the quickest.[8] .In Table 1 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	2227
20000	5058
30000	8254

TABLE 1: Running Time for Selection Sort.

Insertion sort

Insertion sort is very similar to selection sort. It is a simple sorting algorithm that builds the final sorted list one item at a time[18]. It has O (n2) time complexity, It is much less efficient on large lists than more advanced algorithms such as quick sort, heap sort, or merge sort. However, insertion sort provides several advantages Simple implementation and, Efficient for small data sets [10][17] .In Table 2 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	1605
20000	3678
30000	6125

TABLE 2: Running Time for Insertion Sort.

Merge sort

Merge sort is a divide and conquer algorithm .It's Divide the list into two approximately equal sub lists, Then Sort the sub lists recursively[19]. It has an O (n log n) Time complexity .merge sort is a stable sort, parallelizes better, and is more efficient at handling slow-to-access sequential media. Merge sort is often the best choice for sorting a linked list [11][20]. In Table 3 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	728
20000	1509
30000	2272

TABLE 3: Running time for merge sort

Quick sort

In this sort an element called pivot is identified and that element is fixed in its place by moving all the elements less than that to its left and all the elements greater than that to its right. Since it partitions the element sequence into left, pivot and right it is referred as a sorting by partitioning. It's an O (n log n) Time complexity in average case[21][22]. In Table 4 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	489
20000	1084
30000	1648

TABLE 4: Running Rime for Quick Sort.

Bubble sort

Bubble sort is a simple sorting algorithm that works by repeatedly; it's comparing each pair of adjacent items and swapping them if they are in the wrong order. This passing procedure is repeated until no swaps are required, indicating that the list is sorted [13][23]. It has a O (n2) Time complexity means that its efficiency decreases dramatically on lists of more than a small number of elements [12][24]. In Table 4 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	1133
20000	3103
30000	5730

TABLE 5: Running Time for Bubble Sort.

Grouping Comparison sort

In this sort we divide the list of elements into groups; each group contains three elements that compare with the first element of next groups. Performance has been decreased by GCS algorithm, mainly if the input size more than 25000 elements that returned increasing number of comparison, the performance have been improved when size of input is less than 25000 elements. It has a time complexity O (n2) [14]. In Table 6 the execution time and number of elements as follow:

Number of elements	Running time (ms)
10000	1124
20000	3374
30000	6687

TABLE 6: Running Time for Comparison Sort.

3. COMPARATIVE STUDY AND DISCUSSION

All the six sorting algorithms (Selection Sort, Insertion sort, Merge sort, Quick sort, Bubble Sort and Comparison sort) were implemented in C++ programming languages and tested for the random sequence input of length 10000, 20000, 30000, All the six sorting algorithms were executed on machine Operating System having Intel(R) Core(TM) 2 Duo CPU E8400 @ 3.00 GHz (2 CPUs) and installed memory (RAM) 2038 MB. The Plot of length of input and CPU time taken (ms) is shown in figure 1. Result shows that for small input the performance for the six techniques is all most nearest, but for the large input Quick sort is the fastest and the selection sort the slowest. the grouping comparison sort for small input (10000) is the third sort and in the large input (30000) is the fifth sort in order between the six sorting algorithms.

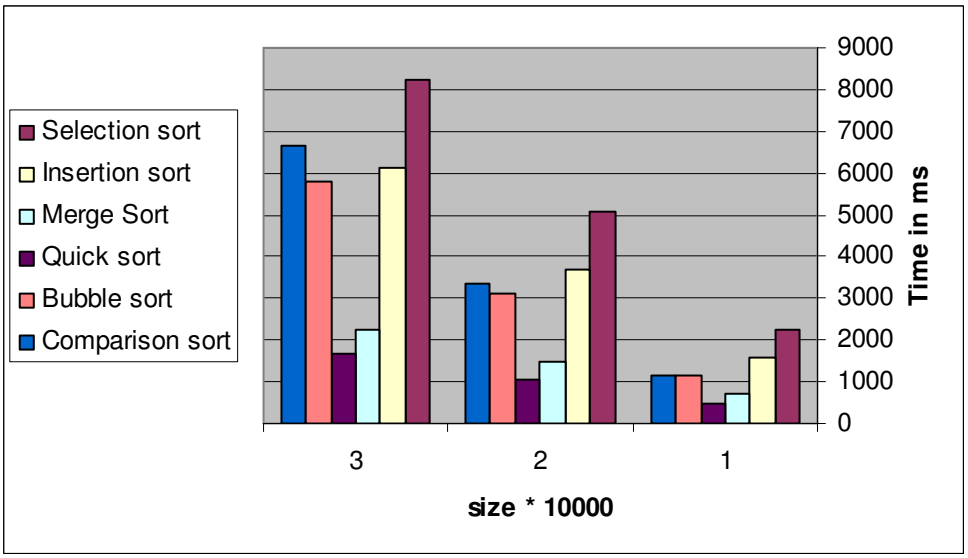

FIGURE 1 : Plot of Number of Input vs CPU.

3.1 Complexity Comparison between Typical sorting algorithms

The comparison of complexity between GCS and conventional sort algorithms are listed in table 7[5]. Table 6 determines the time complexity of new algorithm is equivalent to some conventional sort algorithms[25][28]. GCS gave an additional method to manipulate information.

Algorithm	Average case	Worst case
Selection sort	O (n2)	O (n2)
Insertion sort	O (n2)	O (n2)
Merge Sort	O (n log n)	O (n log n)
Quick sort	O (n log n)	O (n2)
Bubble sort	O (n2)	O (n2)
Comparison Sort	O (n2)	O (n2)

TABLE 7: Time Complexity of Typical Sorting Algorithms.

4. CONCLUSION AND FUTURE WORK

This paper discuss a comparison between the new suggested algorithm (GCS) and selection sort, Insertion sort, merge sort, quick sort and bubble sort. It analysis the performance of these algorithms for the same number of elements (10000, 20000, 30000). For small input the performance for the six techniques is all most nearest, but for the large input Quick sort is the fastest and the selection sort the slowest. Comparison sort in average and worst case have the same time complexity with selection, Insertion and bubble sort This research is initial step for future work; in the future we will improve our algorithms Grouping Comparison Sort algorithms (GCS) to optimize software's in searching method and retrieve data.

5. REFERENCES

[1] P.Adhikari, Review on Sorting Algorithms, "A comparative study on two sorting algorithm", Mississppi state university, 2007.

[2] T. Cormen, C.Leiserson, R. Rivest and C.Stein, *Introduction To Algorithms*, McGraw-Hill, Third Edition, 2009,pp.15-17.

[3] M. Goodrich and R. Tamassia, Data *Structures and Algorithms in Ja*va,John wiley & sons 4th edition, 2010,pp.241-243.

[4] R. Sedgewick and K. Wayne, *Algorithms*,Pearson Education, 4th Edition, 2011,pp.248-249.

[5] T. Cormen, C.Leiserson, R. Rivest and C.Stein,*Introduction to Algorithms*, McGraw Hill, 2001,pp.320-330.

[6] H. Deitel and P. Deitel, *C++ How to Program*, Prentice Hall, 2001,pp.150-170

[7] M. Sipser, *Introduction to the Theory of Computation*,Thomson,1996,pp.177-190.

[8] S. Jadoon , S.Solehria, S.Rehman and H.Jan.(2011,FEB). " Design and Analysis of Optimized Selection Sort Algorithm".11. (1),pp. 16-21. Available: http://www.ijens.org/IJECS%20Vol%2011%20Issue%2001.html.

[9] Aditya Dev Mishra & Deepak Garg.(2008,DEC). “Selection of Best Sorting Algorithm ", *International journal of intelligent information Processing*. II (II).pp. 363-368. Available: http://academia.edu/1976253/Selection_of_Best_Sorting_Algorithm.

[10] http://en.wikipedia.org/wiki/Insertion_sort

[11] http://en.wikipedia.org/wiki/Merge_sort

[12] http://en.wikipedia.org/wiki/Bubble_sort

[13] http://www.techopedia.com/definition/3757/bubble-sort

[14] I. trini, k. kharabsheh, A. trini, (2013,may)."Grouping Comparison Sort", *Australian Journal of Basic and Applied Sciences*.pp221-228.

[15] R. Sedgewick, *Algorithms in C++*, Addison–Wesley Longman,1998,pp 273–274.

[16] A. Levitin, *Introduction to the Design & Analysis of Algorithms*, Addison–Wesley Longman, 2007, pp 98–100.

[17] http://corewar.co.uk/assembly/insertion.htm

[18] T. Cormen, C.Leiserson, R. Rivest and C.Stein, *Introduction To Algorithms*, McGraw-Hill, Third Edition, 2009,pp.15-21

[19] Katajainen, Jyrki; Pasanen, Tomi; Teuhola, Jukka (1996,MAR). "Practical in-place mergesort". *Nordic Journal of Computing*. (3). pp. 27–40.

[20] Kronrod, M. A. (1969). "Optimal ordering algorithm without operational field". Soviet Mathematics - Doklady (10). pp. 744.

[21] T. Cormen, C.Leiserson, R. Rivest and C.Stein, *Introduction To Algorithms*, McGraw-Hill, Third Edition, 2009,pp.145-164.

[22] A. LaMarca and R. E. Ladner.(1997), "The Influence of Caches on the Performance of Sorting." Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms,. pp. 370–379.

[23] D.Knuth, The Art of Computer Programming,Addison-Wesley,Third Edition 1997,pp. 106–110 .

[24] http://www.sorting-algorithms.com/bubble-sort.

[25] T.Niemann, Sorting and Searching Algorithms:A Cookbook, 2008 , pp.11-14.

[26] H.Ahmed,H.Mahmoud, N.Alghreimil," A Stable Comparison-based Sorting Algorithm with worst Case complexity of O(n log n) Comparisons and O(n) Moves" ,*World Academy of Science, Engineering and Technology(22)*,pp.970-975.

[27] C.Cook , D.Kim. "Best sorting algorithm for nearly sorted lists*". Commun. ACM*, 23(11),pp.620-624.

[28] D.Garg," Selection O. Best Sorting Algorithm*", International Journal of Intelligent Information Processing* 2(2),pp.363-368.

,