International Journal of u- and e- Service, Science and Technology
Vol.9, No. 8 (2016), pp.287-296
http://dx.doi.org/10.14257/ijunesst.2016.9.8.24

Different Sorting Algorithm's Comparison based Upon the Time Complexity

D. Rajagopal[1*] and K. Thilakavalli[2]

[1]*Assistant Professor, Department of Computer Applications, KSR College of Arts and Science(Autonomous), Tiruchengode, Namakkal Dt, Tamilnadu, India*
[2]*Assistant Professor, Department of Computer Applications, KSR College of Arts and Science(Autonomous), Tiruchengode, Namakkal Dt, Tamilnadu, India*
[1]*sakthiraj2782007@gmail.com,*[2]*thilaksathya2782007@gmail.com*

Abstract

In this paper the different sorting algorithms execution time has been examined with different number of elements. Through this experimental result concluded that the algorithm which is working best. The analysis, execution time has analyzed, tabulated in Microsoft Excel. The sorting problem has attracted a great deal of study, possibly due to the complexity of solving it proficiently despite its simple, familiar statement. Sorting algorithms are established in opening computer science classes, where the abundance of algorithms for the problem provides a gentle beginning to variety of core algorithm concepts. Objective of this paper is finding the best sorting algorithm.

Keywords: *Sorting algorithm, Time Complexity, Efficiency, Execution Time, Quick Sort, Selection Sort*

1. Introduction

Sorting algorithm is an algorithm in that the most-used orders are numerical order and lexicographical order in computer science and mathematics. Proficient sorting is vital for optimizing the utilization of other algorithms that require sorted lists to work correctly. More properly, the output must gratify two circumstances: (1) The output is in non decreasing order and (2) The output is a variation, or reordering, of the input. Since the early of computing, the sorting problem has fascinated a great deal of research, perhaps due to the complexity of solving it proficiently although its simple, familiar statement. Sorting algorithms are rife in early computer science classes, where the plenty of algorithms for the problem provides a gentle opening to a variety of core algorithm concepts, such as big O notation, divide and conquer algorithms, *etc*.

2. Lirerature Review

(Jehad Alnihoud, Rami Mansi, January-2010) They presented two sorting algorithms. They are enhanced selection sort and enhanced bubble sort. Enhanced selection sort was an enhancement on selection sort by made it slightly faster and stable sorting algorithm. Enhanced bubble sort was an enhancement on both bubble sort and selection sort algorithms with O(nlogn) complexity instead of O(n2) for bubble sort and selection sort algorithms. (Ibrahim M.Al Turani, Khalid S.Al-Kharabsheh, Abdallah M. AlTurani, 2013) They suggested a sorting algorithm, Grouping Comparison Sort (GCS) on an approach of compare-based sorting. GCS was a sort depends on separating the list of elements into assembly, each assembly consisting of three elements, each set arranged discretely. After that compared between groups even to access sorted list of elements,

* Corresponding Author

ISSN: 2005-4246 IJUNESST

results on time complexity O(n2).(Ankit R. Chadha, Rishikesh Misal, Tanaya Mokashi, Aman Chadha, April-2014) They presented the concept and implementation of ARC sort which not only reduces the running time as compared to selection sort , Insertion sort and Bubble Sort but these were also efficient in performance as it reduces the number of unnecessary comparison and swaps by making use of two dimensional arrays. (Hadi Sutopo, February-2011) He presented selection sorting algorithm visualization. Some details about implementation of selection sorting algorithm in Flash and Script programming had described. (Bilal Jan, Bartolomeo Montrucchio, Carlo Ragusa, Fiaz Gul Khan and Omar Khan, November-2012) They were tested performance of parallel bitonic, odd-even and rank sort algorithms for GPUs and comparison with their serial implementation on CPU.

3. Classification of Algorithms

1) Shell Sort method is Swap Based, in which couple of elements is being exchanged.

2) Insertion Sort method is Merge Based, Having sequence swapped with each other afterwards any element.

3) Heap Sort method is Tree Based, wherever data is stored in form of binary tree.

4) Other Categories are having an additional key for perform swap, Radix and Bucket Sort.

4. Methods of Sorting

1) Selection Sorting methods can be implemented in Heap Sort, Smooth Sort, Strand Sort, Insertion Sort and Selection Sort

2) Insertion sorting method can be implemented in Shell Sort, Tree Sort, Liberary Sort.

3) Exchange Sorting method can be implemented in Bubble Sort, Cocktail Sort, Gnome Sort, Comb Sort, and Quick Sort.

4) Merge Sorting method can be implemented in Merge Sort.

5) Non Comparison Sorts are Radix Sort, Counting Sort, and Bucket Sort

6) Other Sorting techniques are Topological Sorting, Sorting Network.

5. Comparison on the Basis of Complexity and other Factors

Table 1. Comparisons on the Basis of Complexity and Other Factors

Sorting Algorithm	Best Case	Average Case	Worst Case
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion sort	O(n)	$O(n^2)$	$O(n^2)$
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Heap Sort	O(n log n)	O(n log n)	O(n log n)
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$

5.1. Bubble Sort

It is simplest sorting algorithm that works by frequently exchange the neighboring elements if they are in wrong order. In Bubble sort putting the largest element at the highest index in the array.


```
Function BubbleSort(array, array_size)
var i, j, temp;
For i:=(array_size-1)down to 0 step-1
      For j := 1 to i do
         If array(j-1) > array(j) then
      temp := array[j-1];
      array [j-1] := array[j];
      array[j] := temp;
      End If
      End For
End For
End Function
```

Example:

First Pass:

(5 1 4 2 8) –> (1 5 4 2 8),

The first two elements compared by algorithm, until swaps 5 > 1.

(1 5 4 2 8) –> (1 4 5 2 8), Swaps until 5 > 4

(1 4 5 2 8) –> (1 4 2 5 8), Swaps until 5 > 2

(1 4 2 5 8) –> (1 4 2 5 8),

Second Pass:

(1 4 2 5 8) –> (1 4 2 5 8)

(1 4 2 5 8) –> (1 2 4 5 8), Swaps until 4 > 2

(1 2 4 5 8) –> (1 2 4 5 8)

Now, the array is already sorted, but algorithm doesn't identify if it is completed. The algorithm wants one whole pass without any swap to identify.

Third Pass:

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

(1 2 4 5 8) –> (1 2 4 5 8)

5.2. Selection Sort

The selection sort algorithm sorts an array by frequently finding the slightest element commencing unsorted part and putting at the beginning. The algorithm maintains two sub arrays in a given array.

1) The sub array which is already sorted.

2) Remaining sub array which is unsorted.

Every Iteration of selection sort, the slightest element from the unsorted secondary array is picked and moved to sorted secondary array.

```
Function SelectionSort(array, size)
var min, temp, i, j
For i := 0 to size-2 do
min := i
For j := i+1 to size-1 do
  If array(j) < array(min)
          min := j
  End If
End For j
   temp := array(i)
```

```
    array(i) := array(min)
    array(min) := temp
  End For i
  End Function
```

5.3. Insert Sort

Insertion sort is used to sort an array of numbers. Numbers are divided into sorted and unsorted. The unsorted values are putting into their suitable positions in the sorted secondary array one by one. Important characteristics of Insertion Sort are given below.

- It has simplest implementation.
- It is well-organized for minor data sets, but very incompetent for larger lists.
- Insertion Sort is reduces its total number of steps if given a partially sorted list, increased efficiency.
- It is better than Selection and Bubble Sort algorithms.
- Its space complexity is less, and it also requires a single additional memory space.
- Stable, it does not change the relative order of elements with equal keys.

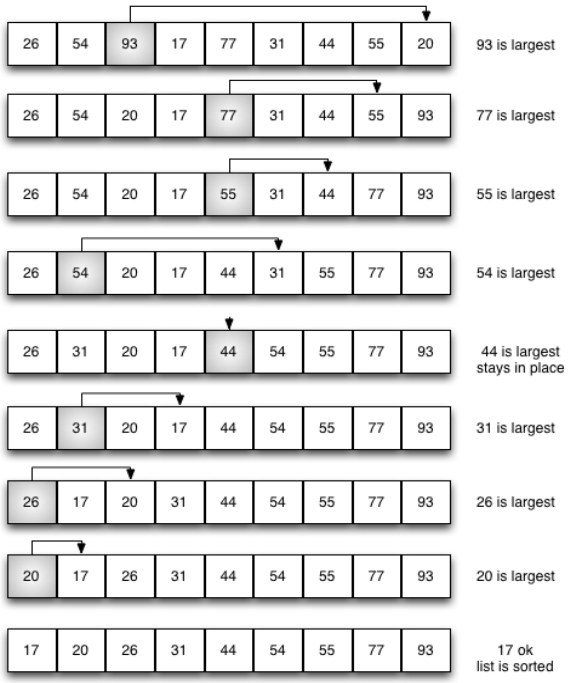
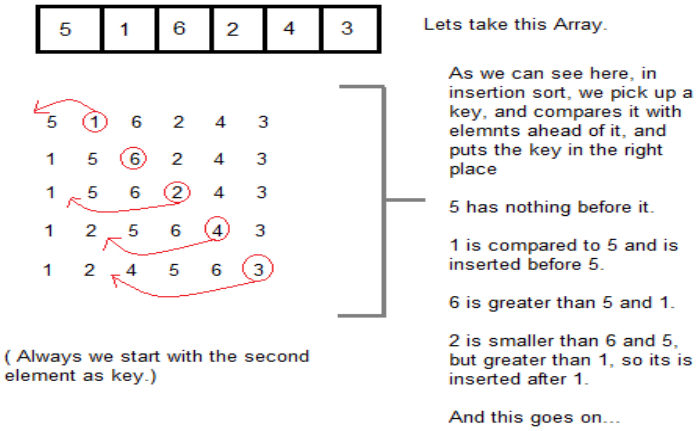


Figure 1. Example of Selection Sort

```
  For I=2 to N
  A [I] =item, J=I-1
  WHILE j>0 and item<A[J]
          A[J+1]=A[J]
          J=J-1
          A [J+1]=item
  end while
  end for
Insertion Sort working procedure
```

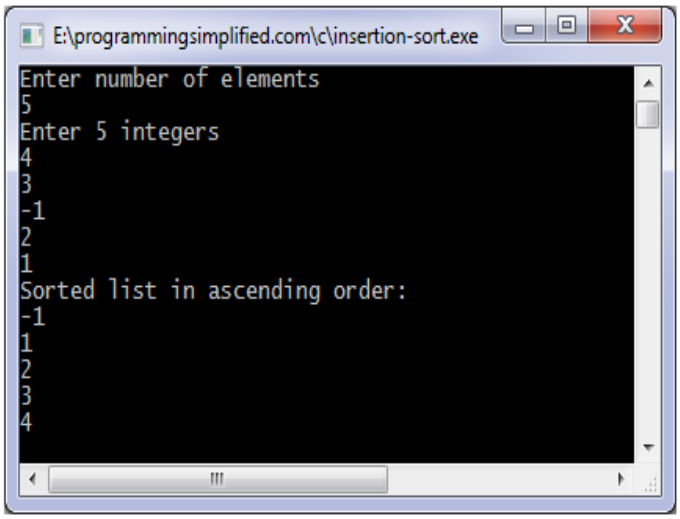


Figure 2. Output Screen for Insertion Sort

5.4. Heap Sort

First find the utmost element and place the utmost element at the end. Primarily on getting an unsorted list, the first step in heap sort is to make a Heap data organization. Once heap is built, the first element of the Heap is also largest or smallest. Heap Sort Algorithm for sorting in increasing order:

1. Build a max heap from the input data.
2. At this point, the biggest item is stored at the origin of the heap. Replace it with the final item of the heap followed by tumbling the size of heap by 1. Finally, heap the origin of tree.
3. Repeat above steps until size of heap is greater than 1.

```
heapSort(DATA, N)
 heapify(DATA, N)
 Set end = N - 1
 while end > 0 do
       a) swap(DATA[end], DATA[0])
       b) Set end = end – 1
       c) siftDown(DATA, 0, end) [End of while in Step 3]
 Exit
```

```
FUNCTION heapify(DATA,N)
 start := (N - 2) / 2
 while start ≥ 0 do
   a) siftDown(DATA, start, N-1)
   b) start := start – 1 [End of While in step 2]
```

5.5. Merge Sort

Merge Sort is worked with the basis of Divide and Conquer algorithm. It split input array in two bisects, calls itself for the two bisects and merges two sorted bisects.

1. Divide Step:
   If a given array A has zero or one element, simply return; if previously sorted. Otherwise, split A [p ... r] into two secondary arrays A [p ... q] and A [q + 1 ... r], each containing about partially of the elements of A [p ... r]. That is, q is the intermediate point of A [p ... r].
2. Conquer Step:
   Sorting the two secondary arrays A [p ... q] and A [q + 1 ... r] recursively by using conquer
3. Combine Step:
   Merge the elements rear in A [p ... r] by merging the two sorted secondary arrays A [p ... q] and A [q + 1 ... r] into a sorted series. To achieve this pace, will identify a procedure MERGE (A, p, q, r).

```
MERG-SORT (A, p, r)
IF (p < r) THEN
q = FLOOR [(p + r)/2]
MERGE (A, p, q)
MERGE (A, q + 1, r)
MERGE (A, p, q, r)

MERGE (A, p, q, r)
 n1 ← q − p + 1
 n2 ← r − q
 Create arrays L[1 ... n1 + 1] and R[1 .. . n2 + 1]
 For I ← 1 TO n1
 do L[I] ← A[p + I − 1]
 For J ← 1 TO n2
 do R[J] ← A[q + J ]
 L[n1 + 1] ← ∞
 R[n2 + 1] ← ∞
 I ← 1
 J ← 1
 For k ← p TO r
 do
  IF (L[I] ≤ R[J]) THEN
   A[k] ← L[I]
   I ← I + 1
  ELSE
   A[k] ← R[J]
   J← J + 1
  ENDIF
END FOR K
END FOR J
END FOR I
```

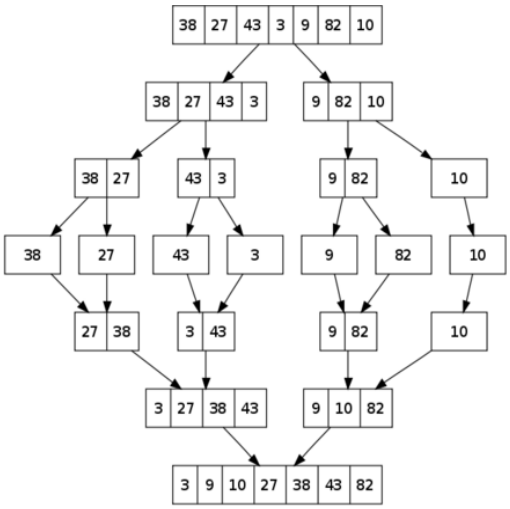



Figure 3. Example of Merge Sort

5.6. Quick Sort

It is not stable search, but it is very fast and needs very fewer supplementary spaces. Quick Sort is worked with the basis of Divide and Conquer algorithm. It selects an element as pivot and divider the given array around the picked pivot. There are many diverse versions of quick Sort that pick pivot in different ways.

- Always pick primary element as pivot.
- Always pick end of the element as pivot (implemented below)
- Pick a random element as pivot.
- Pick median as pivot.

5.7. Different Sorting Algorithms Elapsed Time Analysis

Table 2. Different Sorting Algorithms Elapsed Time Analysis

No of Elements	Elapsed Time (ms)					
	Selection Sort	Insertion Sort	Bubble Sort	Quick Sort	Merge Sort	Heap Sort
1000	364	317	315	236	341	324
1500	565	482	528	438	498	562
2000	763	678	684	632	691	725
2500	865	842	795	781	844	796
5000	993	946	965	824	858	928
10000	1007	993	995	918	988	984
20000	1243	1141	1081	1040	1121	1093
30000	1364	1320	1256	1108	1278	1317

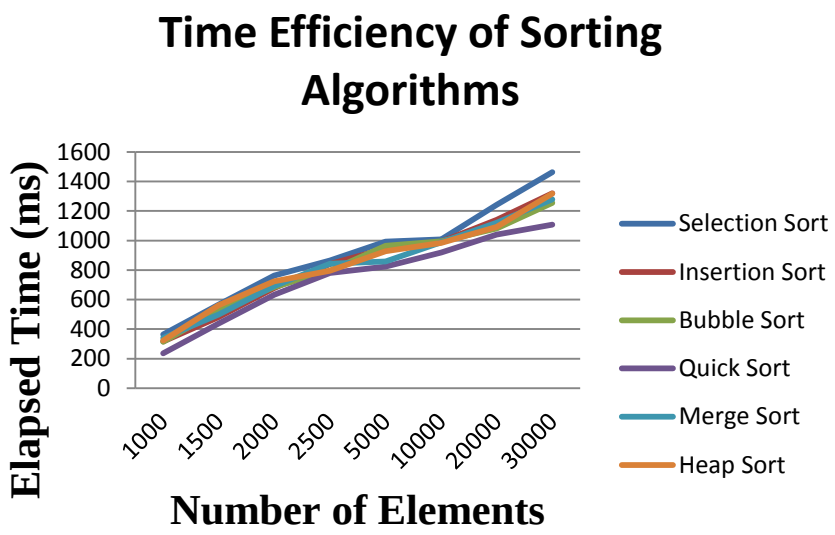


Figure 4. Time Efficiency of Sorting Algorithms

6. Conclusion

The above analysis said that, Quick Sort is the fastest algorithm but it needs enough memory. Quick sort is more often used as external sorting. On the above comparison and the resultant analysis, it is clear to use Bubble Sort for small data set whereas Quick Sort for large data set.

References

[1] J. Alnihoud and R. Mansi, "An Enhancement of Major Sorting Algorithms", The International Arab Journal of Information Technology, vol. 7, no. 1, **(2010)** January, pp. 55-62.
[2] I. M. Al-Turani, K. S. Al-Kharabsheh and A. M. AlTurani, "Grouping Comparison Sort", Australian Journal of Basic and Applied Sciences, vol. 7, no. 7, **(2013)**, pp. 470-475.
[3] A. R. Chadha, R. Misal, T. Mokashi and A. Chadha, "ARC Sort: Enhanced and Time Efficient Sorting Algorithm", International Journal of Applied Information Systems (IJAIS), vol. 7, no. 2, **(2014)** April, pp. 31-36.
[4] H. Sutopo, "Selection Sorting Algorithm Visualization Using Flash", The International Journal of Multimedia & Its Applications (IJMA) vol. 3, no. 1, **(2011)** February, pp. 22-35.
[5] B. Jan, B. Montrucchio, C. Ragusa, F. G. Khan and O. Khan, "Fast Parallel Sorting Algorithms On Gpus", International Journal of Distributed and Parallel Systems (IJDPS) vol. 3, no. 6, **(2012)** November, pp. 107-118.
[6] T. Ehsan, M. U. Ali and M. Q. Javed, "An Efficient Sorting Algorithm by Computing Randomized Sorted Sub-Sequences Based on Dynamic Programming", IJCSNS International Journal of Computer Science and Network Security, vol. 13, no. 9, **(2013)** September, pp. 51-57.
[7] S. Popli, A. Talwar and S. Gupta, "A Comparison Between Three Sorting Algorithms Based Upon The Time Complexity", International Journal of Advanced Technology in Engineering and Science, vol. 02, no. 01, **(2014)** September, pp. 643-647.
[8] R. Angrish and D. Garg, "Efficient String Sorting Algorithms: Cache-aware and Cache-Oblivious", International Journal of Soft Computing and Engineering (IJSCE), vol. 1, no. 2, **(2011)** May, pp. 12-16.
[9] Prof. (Dr.) V. N. Maurya, Dr. R. K. Bathla, D. K. Arora, Er. A. K. Maurya and R. A. Gautam, "An Alternate Efficient Sorting Algorithm Applicable for Classification of Versatile Data", International Journal of Mathematical Modeling and Applied Computing, vol. 1, no. 1, **(2013)** April, pp. 01-10.
[10] S. Singh and S. Kaur, "Freeze Sorting Algorithm Based on Even-Odd Elements", International organization of Scientific Research, vol. 04, no. 03, **(2014)** March, pp. 18-23.
[11] M. Hamdi, C. Qiao, Y. Pan and J. Tong, "Communication-Efficient Sorting Algorithms on Reconfigurable Array of Processors With Slotted Optical Buses", Journal of Parallel and Distributed Computing, vol. 57, **(1999)**, pp. 166-187.
[12] A. Shukla and A. K. Saxena, "Review of Radix Sort & Proposed Modified Radix Sort for Heterogeneous Data Set in Distributed Computing Environment", International Journal of Engineering Research and Applications (IJERA),vol. 2, no. 5, **(2012)** September- October, pp. 555-560.

[13] A. M. Aliyu and 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, no. 7, **(2013)**, pp. 25-30.
[14] D. Roy and A. Kundu, "A Comparative Analysis of Three Different Types of Searching Algorithms in Data Structure", International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, no. 5, **(2014)** May, pp. 6626-6630.
[15] J. Singh, B. Patra and S. P. Singh, "An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System", International Journal of Advanced Computer Science and Applications, vol. 2, no. 2, **(2011)** February, pp. 31-37.
[16] A. Bharadwaj and S. Mishra, "Comparison of Sorting Algorithms based on Input Sequences", International Journal of Computer Applications, vol. 78, no. 14, **(2013)** September, pp. 7-10.
[17] N. Chhajed, I. Uddin and S. S. Bhatia, "A Comparison Based Analysis of Four Different Types of Sorting Algorithms in Data Structures with Their Performances", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 2, **(2013)** February, pp. 373-381.
[18] P. Sareen, "Comparison of Sorting Algorithms (On the Basis of Average Case)", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 3, March **(2013)**, pp. 522-532.
[19] M. Khan, S. Shaheen and F. A. Qureshi, "Comparative Analysis of five Sorting Algorithms on the basis of Best Case, Average Case, and Worst Case", International Journal of Information Technology and Electrical Engineering, vol. 3, no. 1, **(2014)** February, pp. 1-10.
[20] M. Gul, N. Amin and M. Suliman, "An Analytical Comparison of Different Sorting Algorithms in Data Structure", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 5, **(2015)** May, pp. 1289-1298.

Authors

D. Rajagopal, he has completed his Bachelor of Computer Science degree and completed his Master of Computer Applications degree in Periyar University in the year 2003 and 2006 respectively. He completed his Master of Philosophy in PRIST University in the year of 2012. He has 3 Years and 10 Months Experience in the field of Software Developing and 5 Years and 6 months Experience in the field of Teaching. He published 8 International Journal papers and a National Conference paper. He delivered more than 10 seminar & training programs for different Academic Institutions. His research area of Interest is Computer Networks (Wireless & Wired), Image Processing, Mobile Computing, Data Mining, Software Programming (OOPS). Currently he is working as an Assistant professor in the Department of Computer Applications in K.S.Rangasamy College of Arts and Science (Autonomous), Tiruchengode, Namakkal Dt, Tamilnadu, India.

K. Thilakavalli, She has completed her Bachelor of Physics degree in Bharathiar University in the year of 2006. She completed her Master of Computer Applications degree in Anna University in the year 2009. She completed her Master of Philosophy in PRIST University in the year of 2010. She has 6 Years 6 months experience in the field of Teaching. She published 8 International Journal papers, three International Conference papers, and two National Conference papers. Her research area of Interest is Image Processing, Computer Networks (Wireless & Wired), Mobile Computing, Data Mining. Currently she is working as an Assistant Professor in the Department of Computer Applications in K.S.R College of Arts and Science for women, Tiruchengode, Namakkal Dt, Tamilnadu, India.