DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4239

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A COMPARATIVE STUDY OF VARIOUS TYPES OF SORTING TECHNIQUES

Pavleen Kaur
CGC College Of Engineering
Mohali, India

Shivani Mahajan
CGC College Of Engineering,
Mohali, India

Ms. Harneet Kour
CGC College Of Engineering,
Mohali, India

Abstract: Sorting is the procedure of arranging the elements in ascending or descending order. Sorting algorithms are not only used in computer science but also in our day-to-day life to reduce complexity. For optimizing the use of other algorithms, efficient sorting is required. This paper makes a comparison between merge sort, quick sort, selection sort and insertion sort by their time complexities.

Keywords: Merge sort, quick sort, selection sort, insertion sort, analysis, time complexity.

I. INTRODUCTION

Algorithm is determined as a limited collection of consecutive steps through which the solution of a particular problem is obtained. An algorithm has to produce an output for the given set of input values. The considered number of steps in any programming language should be definite and unambiguous. Each and every instruction must be such that it can be executed easily.

Many relevant algorithms can be made by using a variety of designing procedures. The various techniques taken into account are divide-and-conquer, incremental approach, greedy method, dynamic programming, backtracking and branch and bound [1].

An algorithm that rearranges the elements of a list in a certain order, for instance ascending and descending order is referred to as a sorting algorithm. Numerical order and lexicographical order are the most-used orders. There are many sorting algorithms like merge sort, insertion sort, and quick sort [2].

The sorting algorithms efficiency depends on the number of input values, some algorithms perform best when given small number of input values while other perform best when given large number. Algorithms efficiency is judged by its time complexity, time complexity is the amount of time taken by an algorithm to solve a particular problem. It is divided into three cases, best case, worst case and average case. The best case is denoted by Ω (Lower -Bound), worst case by Big-O-notation (Upper-Bound) and average case by θ (Tight-Bound).

There are two types of sorting-Internal sorting and External Sorting. If the list to be sorted is stored in the main memory, then the sorting operation is called as Internal Sorting. On the other hand, if the list is stored in the secondary memory (floppy, hard disk, etc) then the sorting is referred to as External Sorting.

II. DESCRIPTION OF SOME EFFICIENT SORTING TECHNIQUES

A. *Merge Sort*

Merge Sort algorithm, which was invented by John von Neumann in 1945, is a comparison based sorting algorithm. Divide and conquer paradigm is used by merge sort algorithm to accomplish the task of sorting. The beginning of this algorithm includes comparing each pair of elements and interchanging them if the first element is greater than the second one. After that, each pair of pairs is merged into sorted quadruplets and then into two sorted sub arrays and finally a single sorted array list is obtained. It has O (n log n) worst case time complexity, which is quite efficient [3]. The merge sort algorithm can be represented as follows [4]:

1) Merge(LB,Mid,UB):

- Set i=LB, j=UB, k=mid+1, J=0
- Repeat steps 3&4 while(i<=mid && k<=UB)
- If(Arr[i]<=Arr[k]
  a) Set B[J]=Arr[i]
  b) Set i++
  Else
  a) Set B[J]=Arr[k]
  b) Set k++
- J=J+1
- If(i>mid) goto Step 6 else goto Step 7
- Repeat step for l=k to UB
  a) B[J]=Arr[l]
  b) Set J++
- Repeat step for l=i to mid
  a) Set B[J]=Arr[l]
  b) Set J++
- Repeat step for l=LB to UB
  - Set Arr[l]=B[l]
- Exit

MergeSort(LB,,UB,Arr):

- if(LB<UB)
  a)Mid=(LB+UB)/2
  b)MergeSort(LB,Mid,Arr)
  c)MergeSort(Mid+1,UB,Arr)
  d)Merge(LB,Mid,UB)

Fig. 1 depicts the merge sort technique [5]:

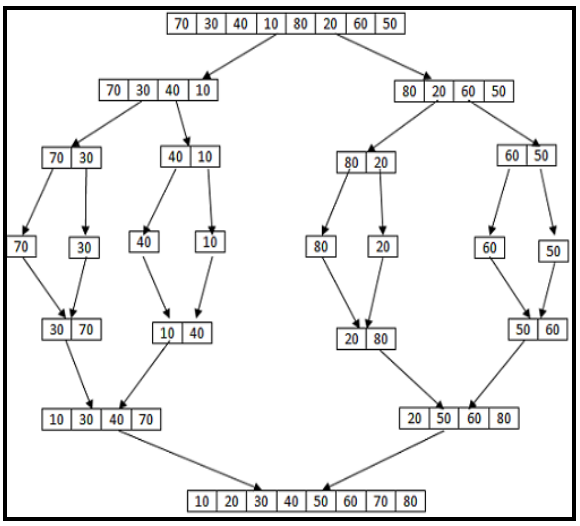

Figure 1. An example to sort an array of 8 integer values using a recursive merge sort algorithm.

B. *Quick Sort*

Quick Sort algorithm, also known as partition-exchange sort was developed by Tony Hoare in 1959. It is an efficient, recursive, comparison based and divide and conquer sorting algorithm in which each element is placed in its proper location at every step. It first picks up a random element, called a pivot value from the list. The partitioning is done such that the elements that are less than the pivot are moved before the pivot and the elements that are greater than the pivot are moved after it. The whole procedure is applied recursively for both the obtained sub lists separately [6]. Despite of the slow worst case running time as $\theta(n^2)$,it is the most popular sorting technique because of the average running time being $\theta(n \log n)$.It is an in place sorting technique, that is it will not acquire any additional storage. The following algorithm demonstrates the quick sort technique [7].

1) QuickSort(A[],low,high):

- Set l=low, h=high, key= A[(low+high)/2]
- Repeat steps 3 to 5 while(l<=h)
- Repeat step while(A[l]<key)
  a)l++
- Repeat step while(A[h]>key)
  a)h--
- if(l<=h)
  a) Set temp=A[l], A[l]=A[h],A[h]=temp
  b)l++, h--
- if(low<h)
  a)QuickSort(A,low,h)
- if(l<high)
  a)QuickSort(A,l,high)

This approach is illustrated in Fig. 2 [8]:

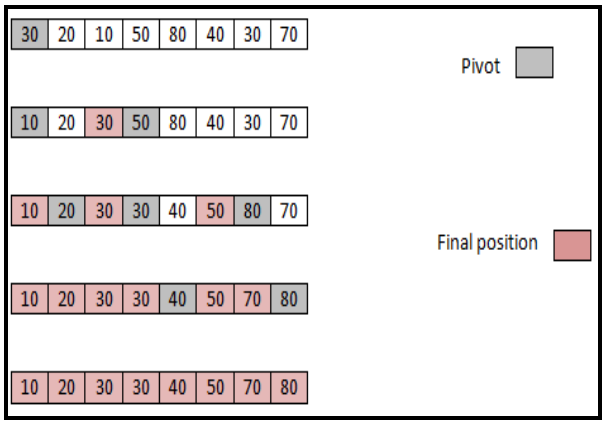

Figure 2. An example to sort an array of 8 integer values using a quick sort algorithm.

C. *Selection Sort*

Selection Sort follows incremental approach, in this if one element is positioned at its suitable place then the index is incremented. In selection sort, first the least element in the array list is found, then it is interchanged with the first element of the list, after that we again find the least element in array not considering the first element (an array without first element) and interchange with the second element and it goes on until we get the sorted list. Selection Sort has worst case time complexity of O (n^2) [9].

1) SelectionSort:

- Set j=0
- Repeat steps 3 to8 while(j<n)
- Set min=a[j]
- Set i=n-1
- Repeat step while(i>j)
- if(a[i]<min)
  a) Set min=a[i]
  b) Set loc=i
- Set k=a[j]
- if(k>min)
  a) Set a[j]=a[loc]
  b) Set a[loc]=k
- Exit

The selection sort technique is represented in Fig. 3 [10]:

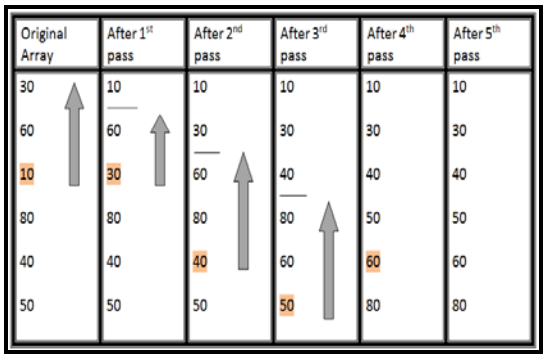

Figure 3. An example to sort an array of 6 integer values using a selection sort algorithm.

D. Insertion Sort

Insertion sort is the most effective sorting algorithm when we have to sort a small number of input values [11]. Insertion Sort is based on Bridge-Payer method. It is like arranging the playing cards when distributed, with an empty left hand and a pile of cards on the desk. We start removing one card from the desk and insert it into its appropriate place in the left hand [12].

1) InsertionSort:

- Set j=1
- Repeat step 3 to 6 while(i<n)
- Set temp=a[j];
- Set i=j-1
- Repeat step while(i>=0 && temp<a[i])
  a) Set a[i+1]=a[i];
- Set a[i+1]=temp;
- Exit

Fig. 4 shows the insertion sort technique:

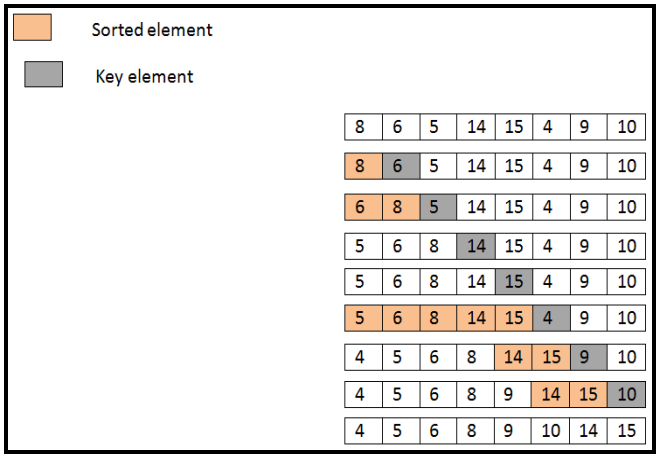

Figure 4. An example to sort an array of 8 integer values using an insertion sort algorithm.

III. A COMPARISON OF DIFFERENT ALGORITHMS

The following table gives the summary of the best-case, average-case and worst-case time complexity of the sorting techniques [13].

Table I. Summary Of Best-Case,Average-Case And Worst-Case

Sort	Time		
	Average	Best	Worst
Merge 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)
Selection Sort	O (n^2)	O (n^2)	O (n^2)
Insertion Sort	O (n^2)	O (n^2)	O (n^2)

The various advantages and disadvantages of the various sorting techniques are given in the table below [14].

Table II. Advantages And Disadvantages Of Sorting Techniques

Sort	Advantages	Disadvantages
Merge Sort	1. This recursive sort is quite fast. 2. It is appropriate for a large list.	1. Memory requirement is larger than other sorts.
Quick Sort	1. Highly fast and efficient. 2. No additional storage is required.	1. Not a stable sort. 2. For choosing some useful element, it is quite complex.
Selection Sort	1. It does not depend on the initial management of data. 2. Appropriate for small data set. 3. Advantageous when data moves are costly but comparisons are not. 4. Any additional temporary storage is not required.	1. Not suited for large lists. 2. Not a stable sort.
Insertion Sort	1. Simple and easy to implement. 2. Space requirement is less. 3. Efficient for small arrays.	1. Inefficient for large lists.

IV. CONCLUSION

In this paper, we have discussed various sorting algorithms (Quick Sort, Selection Sort, Merge Sort and Insertion Sort) and compared their best, average and worst case complexities. So, from this we conclude that Merge Sort and Quick Sort are better than Selection Sort and Insertion Sort. Also, Quick Sort is the most efficient algorithm.

V. REFERENCES

[1] Ravendra Kumar, "Review and Analysis of Sorting Techniques in Various Cases", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 1, January 2016.

[2] Gaurav Kocher and Nikita Agrawal, "Analysis and Review of Sorting Algorithms", International Journal of Scientific Engineering and Research(IJSER), Volume 2, Issue 3, March 2014.

[3] Nitin Upadhyay, The Design & Analysis Of Algorithms, 4th Edn.

[4] Ellis Horowitz, Sartaj Sahni and Sanguthevar Rajasekaran, Computer Algorithm, 2nd Edn.

[5] https://en.wikipedia.org/wiki/Merge_sort

[6] https://en.wikipedia.org/wiki/Quicksort

[7] https://www.algolist.net/Algorithms/Sorting/Quicksort

[8] https://blog.world4engineers.com/quicksort/

[9] https://en.wikipedia.org/wiki/Selection_sort

[10] https://www.studytonight.com/data-structures/selection-sorting

[11] Sonal Beniwal and Deepti Grover, "Comparison of Various Sorting Algorithms: A review", International Journal of Emerging Research in Management & Technology, Volume 2, Issue 5, May 2013.

[12] Thomas H.Coremen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Introduction to Algorithms, 2nd Edn., 2001.

[13] Pankaj Sareen, "Comparison of Sorting Algorithms (On the Basis of Average Case)", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013.

[14] Kamlesh Kumar Pandey, Rakesh Kumar Bunkar and Kamlesh Kumar Raghuvanshi, "A Comparative Study of Different Types of comparison Based Sorting Algorithms in Data Structure", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 2, February 2014.