ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Comparative Study of Sorting Algorithm Based on Their Time Complexity

Ayush Pathak*, Abhijeet Vajpayee, Deepak Agrawal
Acropolis Institute of Technology & Research, Indore, India

Abstract

The quest to develop the most memory efficient and the fastest sorting algorithm has become one of the crucial mathematical challenges of the last half century, resulting in many tried and tested algorithm available to the individual, who needs to sort the list of data. Today, the amount of data is very large, we require some sorting techniques that can arrange these data as fast as possible and also provide the best efficiency in terms of time and space. In this paper, we will discuss some of the sorting algorithms and compare their time complexities for the set of data.

Keywords: Sorting, quick, bubble, selection, insertion, merges.

Introduction

The sorting algorithm is the way to arrange the data elements in some order i.e. either ascending order or in descending order. The algorithm can also be helpful to arrange the data on the basis of certain requirements. Now, in today's era the data of any organization is in the huge amount, it is the need of any organization to arrange the data in an appropriate manner so that the fetching of data will be efficient. To fulfill the purpose, we required some better approach that not only save the time but able to sort huge amount of data efficiently. In this paper, we will focus upon some sorting algorithms like insertion, selection, bubble, quick and merge on the basis of some parameter like the approach that each of them follow, time complexity, stability and the strategy that each algorithm posses. The procedure of each algorithm is different and not a single algorithm is suitable for all type of data. We will also compare these algorithms on the basis of some parameter.

There are two types of sorting:

Internal Sorting:-Internal sorting is a process which takes place entirely in the main memory or primary memory of a computer system. Internal sorting can be used when the data to be sorted is small enough to be part of the primary memory of a computer. For a large set of data to be sorted, only a chunk of data is loaded in the memory at a time as the entire cannot fit in the main memory. The rest of the data which is not loaded is kept on secondary memory storage.

External Sorting:-On the other hand external sorting is useful to sort the data by using secondary storage like floppy disk, hard disk or with the help of any other storage peripheral which is not the part of the computer. There are two phases in external sorting that is sorting phase and merge phase. In the sorting phase the data which can fit in the main memory is read, sorted and then written in a temporary file. In merging phase the temporary files which contains sorted data is merged into a single large file.

Working procedure

Bubble sort-: Bubble sort algorithm is the type of sorting algorithm in which each element is compared with its adjacent element to check which element is larger and which is smaller. This is the basic idea of bubble sort to compare two values and interchange them if they are not in proper order.

Algorithm of **bubble sort-:**

Let us consider an Array A with N number of elements.

1. **First step is to initialize i=0**
2. **Repeat steps 3 to 5 until i<N**
3. **Set j=0**
4. **Repeat step 5 until j<N-i-1**
5. **If A(j)>A(j+1) then**
   Set temp=A[j]
   Set A[j]=A[j+1]
   Set A[j+1]=temp
   End if

6.Exit

Insertion Sort: Insertion sort is the algorithm which sorts the array by shifting its elements one by one. It sorts a set of values by inserting values into an existing sorted file. To insert a record we must first find the proper place where insertion is to be made. This sorting algorithm consists of two phases that is

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Searching and shifting. First the proper place of the element is searched after that shifting of elements takes place.

Algorithm for insertion sort:-

Let us consider an Array a with n number of elements

1. **Set k=1.**
2. **For k=1 to (n-1)**
   Set temp=a[k]
   Set j=k-1
3. **Repeat step 4 until (temp<a[j]) and (j>=0)**
4. **Set a[j+1]=a[j]**
5. **Set temp=a[j+1]**
6. **End of for loop structure**
7. **Exit**

Quick sort: This algorithm works by partitioning the array to be sorted and each partition is in turn sorted recursively. This algorithm is based on the rule of divide and conquer (also called partition exchange sort).

This algorithm divides the list into three main parts:-

1. Pivot elements
2. Elements greater than the pivot elements
3. Elements less than the pivot elements

Algorithm for Quick sort:

```
QUICK(A, p, r)
1. if p >=r then return
2. q = PARTITION(A, p, r)
3. QUICK(A, p, q - 1) Recursive call to Quick
4. QUICK(A, q +1, r)
5. Exit
PARTITION(A, p, r)
1. x = A[r]
2. i = p -1
3. for j = p to r - 1 do
 4. if A[j] <=x then
 5. i =i+1
 6. Exchange A[i] and A[j]
 END IF
7. Exchange A[i+1] and A[r]
8. return i+1
   END FOR LOOP
9. Exit
```

Selection sort:-This sorting algorithm is simplest sorting algorithm. This algorithm finds the smallest element then exchanges it with the element present at first position then finds second smallest element and exchanges it with the element at second position and continues this process until the whole array is sorted.

Algorithm for selection sort:-

Let A be an array with N number of elements

1. **First of all initialize small with first array element i.e. small=A[L]**
2. **For I=L to U do{**
3. **small =A[I],pos=I**
4. **for J=I to U do{**
5. **If A[J]<small then**
6. **{ small=A[J]**
7. **pos=J}**
   J=J+1}
8. **temp=A[I]**
9. **A[I]=small**
10. **A[pos]=temp}**
11. **End**

Merge Sort: Merge sort will work on the approach called divide and conquer. First it will divide the whole list into the two partitions, sort them and then merge the complete sorted list. The merge sort is not the in place sorting algorithm, in the sense, it required extra memory to sort the elements

```
While(i<=M && j<=N)
{
        If(A[i]<B[j]){
                C[k]=A[i];
                K++;i++}
        else{
                C[k]=B[j];
                K++;j++}
}
While(i<=M)
{
C[k++]=A[i++];
}
While(j<=N)
{
C[k++]=B[j++];
}
```

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Comparative study

TABLE 3.1: Comparative Study of different Sorting Algorithms

PARAMETER	INSERTION	SELECTION	BUBBLE	QUICK	MERGE
Sorting Approach	Insertion	Selection	Exchange	Partitioning	Merging
Time Complexity Best Case Worst Case Average Case	 O(N) $O(N^2)$ $O(N^2)$	 $O(N^2)$ $O(N^2)$ $O(N^2)$	 $O(N^2)$ $O(N^2)$ $O(N^2)$	 O(NlogN) $O(N^2)$ O(NlogN)	 O(NlogN) O(NlogN) O(NlogN)
Sorting Type	Internal	Internal	Internal	Internal	Internal & External
In Place	Yes	Yes	Yes	Yes	No
Algorithm Type	Incremental	Incremental	Incremental & Exchange	Divide & Conquer	Divide & Conquer
Stability	Yes	No	Yes	Typical In Place sort is not stable	Yes
Strategy	Scan the entire previous element and place the right element at right position.	Select the min or max element and then compare it with the remaining elements for sorting the list.	Scan all the consecutive elements and bubble up the largest element.	Divide the whole data in two partition with the help of pivot element.	Divide the given list in to the two separate list sort them individually and then merge them up.

TABLE 3.2: Advantages and Disadvantages of different sorting algorithms

Sorting Type	Advantages	Disadvantages
Insertion Sort	Simple and easy to implement. Faster than bubble sort.	The main disadvantage of insertion sort is that it is inefficient for large data list.
Bubble Sort	The main advantage of bubble sort is it is simple to use and easy to implement.	It is code inefficient.
Selection Sort	Simple and easy to implement	Inefficient for large lists, so similar to the more efficient insertion sort, the insertion sort should be used in its place.
Quick Sort	Fast and efficient. Very fast and it requires very less additional space.	Quick search is not a stable search. Show unpredictable results when list is already sorted.
Merge Sort	**It can be applied to files of any size.** **Reading of the input during the run-creation step is sequential ==> Not much seeking.** **Reading through each run during merging and writing the sorted record is also sequential. The only seeking necessary is as e switch from run to run.** **If heap sort is used for the in-memory part of the merge, its operation can be overlapped with I/O** **Since I/O is largely sequential, tapes can be used.**	**If the recursion is used then it takes twice the space in memory as compared with quick sort algorithm**

Conclusion

This paper discusses and compares the five different sorting algorithms. The insertion selection and bubble sorting algorithm will have almost the same complexity in worst and average case. But if we see the time complexity of insertion sort in comparison with bubble and selection then it is a good choice. These entire three algorithms are suitable for the small amount of data. The quick sort will work upon the huge amount of data and all the data must be randomized. But if the data is small or maximum elements are already sorted then the use of the quick sort algorithm is not the good choice. On the other hand merge sort is an internal and external algorithm

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

but the only problem with the merge sort is that it required more space to sort the data.

References

1. Eshan Kapur, Parveen Kumar and Sahil Gupta,"Proposal Of A Two Way Sorting Algorithm And Performance Comparison With Existing Algorithms" International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012.
2. C.A.R. Hoare, Quicksort, Computer Journal, Vol. 5, 1, 10-15 (1962)
3. Knuth, D.E., 1988. The Art of programming-Sorting and Searching. 2nd Edn.,Addison Wesley, ISBN: 020103803X.
4. Cormen, T.H. et al. Introduction to Algorithms. 2nd Edn., 2001. ISBN: 0262032937
5. H. M. Mahmoud, R. Modarres, and R. T. Smythe. Analysis of quickselect: An algorithm for order statistics. ITA – Theoretical Informatics and Applications, 29(4):255–276, 1995.
6. 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) Volume 2,Issue 7,Pages 25-30,2013 ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805
7. Madhavi Desai, Viral Kapadiya "Performance Study of Efficient Quick Sort and Other Sorting Algorithms for Repeated Data" National Conference on Recent Trends in Engineering & Technology 13-14 May 2011
8. Ashutosh Bharadwaj, Shailendra Mishra, "Comparison of Sorting Algorithms based on Input Sequences" International Journal of Computer Applications (0975 – 8887) Volume 78 – No.14, September 2013
9. Comparison of Sorting Algorithms (On the Basis of Average Case) Pankaj Sareen.
10. A Comparison Based Analysis of Four Different Types of Sorting Algorithms in Data Structures with Their Performances.
11. Data Structures by Seymour Lipschutz and G A Vijayalakshmi Pai (Tata McGraw Hill companies), Indian adapted edition-2006,7 west patel nagar,New Delhi-110063
12. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, fifth Indian printing (Prentice Hall of India private limited), New Delhi-110001
13. Data structures using c and c++ by Yedidyah langsam,Aaron M. Tenenbaum second Indian printing (Prentice Hall of India private limited), New Delhi-110001
14. An Introduction to Data Structures with Application by Jean-paul Tremblay Tata McGraaw Hill