International Journal of Research in Engineering, Science and Management
Volume-1, Issue-9, September-2018
www.ijresm.com | ISSN (Online): 2581-5782

A Review on Various Sorting Algorithms

Rimple Patel
Lecturer, Department of Computer Engineering, S.B.Polytechnic, Vadodara, India

***Abstract*—In real life if the things are arranged in order than its easy to access, same way in computer terms if data is in proper order than its easy to refer in future. Sorting process help we to arrange data easily based on types of data. In this paper various sorting technology are discussed through algorithm, also comparison chart of time complexity of various algorithm is discussed for better understanding. Sorting is the operation performed to arrange the records of a table or list in some order according to some specific ordering criterion. Sorting is performed according to some key value of each record. Same data is to be sorted through different sorting technics for better understanding. In computer science, a sorting algorithm is an efficient algorithm which perform an important task that puts elements of a list in a certain order or arrange a collection of items into a particular order. Sorting data has been developed to arrange the array values in various ways for a database.**

***Index Terms*—Bubble Sort, Insertion Sort, Merge Sort Selection Sort and Quick Sort.**

I. Introduction

Arranging the data in ascending or descending order is sorting. When humans realized the importance of searching quickly sorting came into picture. The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. There are many different techniques available for sorting, differentiated by their efficiency and space requirements. There are so many things in our real life that we need to search for, like a particular record in database, roll numbers in merit list, a particular telephone number in telephone directory, a particular page in a book etc. All this would have been a mess if the data was kept unordered and unsorted, but fortunately the concept of sorting came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search. If you ask me, how I will arrange a deck of shuffled cards in order, I would say, I will start by checking every card, and making the deck as I move on. It can take me hours to arrange the deck in order, but that's how I will do it. Thank god, computers don't work like this. Since the beginning of the programming age, computer scientists have been working on solving the problem of sorting by coming up with various different algorithms to sort data. The two main criteria's to judge which algorithm is better than the other have been: Time taken to sort the given data and Memory Space required to do so. There are many different techniques available for sorting, differentiated by their efficiency and space requirements. For instance, sorting will order an array of numbers from lowest to highest or from highest to lowest, or arrange an array of strings into alphabetical order. Typically, it sorts an array into increasing or decreasing order. Most simple sorting algorithms involve two steps which are compare two items and swap two items or copy one item. It continues executing over and over until the data is sorted [1].

II. Sorting Algorithms

A. Insertion Sort

Insertion sort is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort. The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of O (n^2), where n is the number of items.

The main idea of insertion sort is [2]

- Start by considering the first two elements of the array data. If found out of order, swap them
- Consider the third element; insert it into the proper position among the first three elements.
- Consider the fourth element; insert it into the proper position among the first four elements and continue until array is sorted.

Algorithm:

Step 1 − If it is the first element, it is already sorted. Return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted

B. Selection Sort

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list. The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array

boundary by one element to the right. This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where **n** is the number of items.
The main idea of selection sort algorithm is given by

- Find the smallest element in the data list.
- Put this element at first position of list.
- Find the next smallest element in the list.
- Place at the second position of the list and continue until the whole data items are sorted.[2]

Algorithm:
Step 1 − Set MIN to location 0
Step 2 − Search the minimum element in the list
Step 3 − Swap with value at location MIN
Step 4 − Increment MIN to point to next element
Step 5 − Repeat until list is sorted

C. Merge Sort

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(n log n), it is one of the most respected algorithms. Merge sort first divides the array into equal halves and then combines them in a sorted manner.
The merge sort algorithm is work as under.

- Split array A from middle into two parts of length n/2.
- Sorts each part calling Merge Sort algorithm recursively.
- Merge the two sorting parts into a single sorted list.

Algorithm:
Step 1 − if it is only one element in the list
It is already sorted, return.
Step 2 − divide the list recursively into two halves
Until it can no more be divided.
Step 3 − merge the smaller lists into new list in
Sorted order.

D. Quick Sort

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value. Quick sort partitions an array and then calls itself recursively twice to sort the two resulting sub arrays. This algorithm is quite efficient for large-sized data sets as its average and worst case complexity are of O (n^2), where **n** is the number of items.
The partition algorithm works as follows

- A[p] = x is the pivot value.
- A [p…q - 1] contains elements less than x.
- A [q + 1…s - 1] contains the element which are greater than or equal to x.
- A[s...r] contains elements which are currently unknown

Algorithm:
Step 1 − Choose the highest index value has pivot
Step 2 − Take two variables to point left and right of the list excluding pivot
Step 3 − left points to the low index
Step 4 − right points to the high
Step 5 − while value at left is less than pivot move right
Step 6 − while value at right is greater than pivot move left
Step 7 − if both step 5 and step 6 does not match Swap left and right
Step 8 − if left ≥ right, the point where they met is new pivot

E. Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of $O(n^2)$ where n is the number of items.
The steps in the bubble sort can be described as below

- Exchange neighboring items until the largest item reaches the end of the array.
- Repeat the above step for the rest of the array.

Algorithm:

```
begin BubbleSort(list)

  for all elements of list
    if list[i] > list[i+1]
      swap(list[i], list[i+1])
    end if
  end for

  return list

end BubbleSort
```

III. Comparison among Algorithms

In computer science, best, worst, and average cases of a given algorithm express what the resource usage is at least, at most and on average, respectively. Usually the resource being considered is running time, i.e. time complexity, but it could also be memory or other resource. In real-time computing, the worst-case execution time is often of particular concern since it is important to know how much time might be needed in the worst case to guarantee that the algorithm will always finish on time [3].

TABLE I
Comparison Table for Worst Case Complexity

Algorithm	Time Complexity	Space Complexity
Quick Sort	O(n^2)	O(log(n))
Merge Sort	O(n log(n))	O(n)
Bubble Sort	O(n^2)	O(1)
Insertion Sort	O(n^2)	O(1)
Selection Sort	O(n^2)	O(1)

IV. Conclusion

In this paper, I have discussed well known sorting algorithms, depending on the type and size of data different algorithms are to be selected and compared for running time of their algorithms purely as a mathematical entity and tried to analyze as a generic point of view. The analysis of these algorithms are based on the same data and on the same computer. It has been shown that gnome sort algorithm is the quickest one for already sorted data but selection sort is quicker than gnome and bubble in unsorted data. Bubble sort and gnome sort swap the elements if required. In selection sort, however, it continues sorting even if the elements are already sorted. Doing more comparison between more different sorting algorithms is required since no specific algorithm that can solve any problem in absolute. The results show that Quick sort is efficient for both small and large integers. Quick sort is significantly faster in practice than other O(n log n) algorithms. In terms of swapping, the Bubble sort performs the greatest number of swaps because each element will only be compared to adjacent elements and exchanged if they are out of order. Insertion Sort sorts small array fast, but big array very slowly.

References

[1] www.cse.iitk.ac.in/users/cs300/2014/home/~rahume/cs300A/techpaper-review/5A.pdf
[2] T. H. Cormen, C. E. Lieserson, R. L. Rivest and S. Clifford, “Introduction to Algorith”, 3rd ed., The MIT Press Cambridge, Massachusetts London, England 2009.
[3] https://en.wikipedia.org/wiki/Best,_worst_and_average_case
[4] Kazim Ali,International Journal of Advanced Research in Computer Science, 8 (1), Jan-Feb 2017,277-280
[5] Chhajed. N, U. Imran , Simarjeet. S., B., 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, Volume 3, Issue 2, February 2013.