International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Review on Various Sorting Algorithm

Sweety Patel[1]

[1]*Assistant Professor, Computer Engineering Department, Parul Institute of Technology, Vadodara, Gujarat, India*

---***---

Abstract - Sorting algorithm is one of the most basic research fields in computer science. It's goal is to make record easier to search, insert and delete. Through the description of five sort algorithms: bubble, select, insert, merger and quick, the time and space complexity was summarized

Index Terms— bubble sort, selection sort, insertion sort, merger sort and quick sort

***Key Words*: bubble sort, selection sort, insertion sort, merger sort and quick sort**

1. INTRODUCTION

Arranging or sorting things or items is not an overnight development. Its footprints can be traced back in 7th century BCE. Abdul Wahab and O.Issa (2009) state that King of Assyria used the idea of arranging clay tablets for Royal Library sorted according to their shape[1]. Sorting is not a jaguar leap but it has emerged in parallel with the development of human mind. In computer science, alphabetizing, arranging, categoring or putting data items in an ordered sequence on the basis of similar properties is called sorting. Sorting is of key importance because it optimizes the usefulness of data. We can observe plenty of sorting examples in our daily life, e.g. we can easily find required items in a shopping maal or utility store because the items are kept categorically. Finding a word from dictionary is not a tideous task because all the words are given in sorted form. Similarly, finding a telephone number, name or address from a telephone directory is also very easy due to the blessings of sorting. In computer science, sorting is one of the most important fundamental operations because of its pivotal applications. Priority scheduling and shortest job first scheduling are examples of sorting. Thomas Cormen, Charlese Ronald, Clifford Stein and Rivest Leiserson (2001) state "An algorithms is any well-defined computational procedure that takes some value, or set of values, as input and produces some value or set of values as output"[2]. A number of sorting algorithms are available with pros and cons. Alfred Aho, John Hopcroft and Jeffrey Ullman (1974) has classified algorithms on the basis of computational complexity, number of swaps, stability, usage of extra resources and recursion [3]. The items to be sorted may be in various forms i.e. random as a whole, already sorted, very small or extremely large in numer, sorted in reverse order etc. There is no algorithm which is best for sorting all types of data. We must be familiar with sorting algorithms in terms of their suitability in a particular situation. In this paper we are going to compare five (Bubble, Quick, Insertion, Selection and Merge) sorting algorithms for their CPU time consumption on a given input in best, worst and average cases.

Insertion Sort

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

In Insertion Sort algorithm 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 [5]

- 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

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.[5]

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

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.

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

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
```

COMPARISION 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. [6]

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)

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] Wahab, A., Issa, O.A. Fundamentals of Library & Information Sciences, (1st ed.). Cataloguing-in-Publication Data, Ilorin (2009).

[2] Cormen, T.H., Leiserson, C.E., & Rivest, R.L. Introduction to Algorithms (2nd ed.). Prentice Hall of India private limited, New Delhi-110001 (2001).

[3] Aho A., Hopcroft J., and Ullman J. The Design and Analysis of Computer Algorithms, Addison Wesley (1974).

[4]www.cse.iitk.ac.in/users/cs300/2014/home/~rahume/cs300A/techpaper-review/5A.pdf

[5]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.

[6]https://en.wikipedia.org/wiki/Best,_worst_and_average_case

[7]Kazim Ali, International Journal of Advanced Research in Computer Science, 8 (1), Jan-Feb 2017, 277-280

[8]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.