REVIEW ON WELL KNOWN SORTING ALGORITHMS

Praveen Verma[1], Vineet Meshram[2], Palak Keshwani[3]

[1][2]B.E Student, CSE, Kruti Institute of Technology and Engineering, Raipur, Chhattisgarh, India

[3]Assistant Professor, CSE, Kruti Institute of Technology and Engineering, Raipur, Chhattisgarh, India

E-mail: - praveenpk35851@gmail.com, vineetmesh94@gmail.com, palakeshwani@gmai.com

ABSTRACT

There are many popular problems in different practical field of computer science. One of the primary issues is ordering a list of items i.e. sorting. Its goal is to make records easier to search, insert and delete. In most cases, the efficiency of an application itself depends on usage of a sorting algorithm. To increase the performance in terms of computational complexity many of the sorting algorithms have been developed. Some factors that affect the performance are - time complexity, stability, memory space. In this paper, we have done a comparative study of some popular sorting techniques based on their time complexity or running time.

Keywords: *Bubble sort, insertion sort, selection sort, merge sort, quick sort, heap sort, time complexity.*

1.Introduction

Sorting is a technique of rearrangement of given list items either in increasing or decreasing order. In sorting, we have a number of items like a_1, a_2, a_3,.....a_n as input and we have to calculate the output as $a_1 \leq a_2 \leq a_3 \leq$.....$a_n$ and vice versa. The data elements which are to be sorted, normally stored in an array data structure but it is not necessary we can use any other linear or non-linear data structure for storing purpose according to nature of data elements. Most software applications need to sort data in some order e.g. ascending or descending. When data will be sorted then there must be used some sorting algorithm. Efficient sorting is important to optimize the use of other algorithms which require input data to be in sorted lists to work correctly. Number of reasons can participate to select a sorting algorithm to solve a specific sorting problem. In this paper, we present a comparison based analysis of some well-known sorting algorithms and explain these algorithms in a simple manner. Also present some previous comparative study of sorting algorithms of some other people which have done nicely.

2. Sorting Algorithms

A. Bubble Sort

Bubble sort is the simplest sorting algorithm that works by swapping the neighboring items again and again if they are in wrong order. The bubble sort makes multiple pass through a list until the list is arranged in either ascending or descending order. After every pass the next largest item is placed in its proper place. Primarily, each item "bubbles" up to the place where it belongs. It has O (n^2) Time complexity.

The steps in the bubble sort algorithm can be explained as follow-

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

In a list of n items, (n-1) pairs of items are compared in first pass, (n-2) in second pass, (n-3) in third and so on. Table 1 shows the number of comparisons for each pass.

Pass	Comparisons
1	(n-1)
2	(n-2)
3	(n-3)
...	…
n-1	1

Table no. 1- Number of comparisons performed in each pass

B. Insertion sort

Insertion Sort is a simple sorting technique, it can be simply explained by an example of how we sort the playing cards in our hand. Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. After every iteration, one element is removed from the input data, and placed at the location it belongs within the sorted list. At each array-position, it checks the value in the sorted list. If the value is larger, it leaves the element in place and moves to the next and if the value is smaller, it finds the correct position within the sorted list and inserts into that correct position.

The main idea of insertion sort is

- Check the first two item items of the array data, if they are ordered, then swap them.
- Now, check the next element, place it to its proper position and continue until array is sorted.

It has an O (n^2) Time complexity.

C. Selection sort

Selection sort is the simplest of sorting techniques. At every pass Selection sort places the largest item in the proper location. After the first pass, the largest item is in the correct place. After the second pass, the next largest is in place. This process continues and requires (n−1) passes to sort n items, since the final item must be in place after the $(n-1)^{st}$ pass. It works very well for small files, also It has a quite important application because each item is actually moved at most once.

The main idea of selection sort algorithm is given by

- Find the largest element in the data list.
- Put this element at last position of list.
- Find the next largest element in the list.
- Place at the second last position of the list and continue until the whole data items are sorted.

It has O (n2) time complexity, making it inefficient on large lists.

D. Merge sort

Merge sort is based on a popular problem solving technique i.e. divide and conquer technique. In this, we divide the given list into two approximately equal sub lists, then sort the sub lists recursively.

Suppose, we have a given set of data items of length n as an input, stored in an array A. The merge sort algorithm is work as-

- 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.

The running time of Merge sort is Θ (nlogn).

E. Quick sort

Quick sort is a divide and conquer algorithm which relies on a partition operation. To partition an array, we choose a pivot element. Although there are many different ways to choose the pivot value, we will simply use the first item in the list.Pivot element is fixed in its place by moving all the elements less than that to its left and all the elements greater than that to its right. The actual position where the pivot value belongs in the final sorted list, commonly called the split point. We then recursively sort the lesser and greater sub lists. It is one of the fastest sorting algorithms which is the part of many sorting libraries. It is an O (n log n) Time complexity in average case. However worst case running time is Θ (n^2) but it happens rarely.

F. Heap sort

Heap sort is based on Binary Heap data structure. A heap is a left complete binary tree which satisfies the heap order. There are two types of heap. The minimum heap contains the smallest value in the root node and the maximum heap contains the largest value in the root node. Same as the selection sort, we first find the item having maximum value and place it at the end. Repeating the same process for the remaining of items. Heap sort algorithm works as follows

- Build a max heap of a given array A [1…n].
- Extract the largest value from the heap repeatedly.
- When largest element is removed then a whole is left at the root node.
- Replace with the last leaf to fix this problem.

Heap sort is a comparison-based sorting algorithm. Although somewhat slower in practice on most machines than a well-implemented quick sort, it has the advantage of a more favorable worst case O(*n* log *n*) runtime. Heap sort is an in-place algorithm, but it is not a stable sort.

The running time of heap sort is Θ (nlogn)

3. Comparison Table

This table gives the comparison of time complexity or running time of above sorting algorithms in a short and precise manner which is given as under.

Algorithm	Worst case running time	Average case running time
Bubble sort	O (n^2)	O (n^2)
Insertion sort	O (n^2)	O (n^2)
Selection sort	O (n^2)	O (n^2)
Merge sort	O (nlogn)	O (nlogn)
Quick sort	O (n^2)	O (nlogn)
Heap sort	O (nlogn)	O (nlogn)

Table no.2- Comparison of time complexity of different sorting algorithms

4.Conclusion

In this paper, we have discussed well known sorting algorithms and their running time. We have compared the running time of these algorithms purely as a mathematical entity and tried to analyze as a generic point of view. We have discussed six well known sorting algorithms and their running time which is given in the above table. From the table No.2, it is cleared that the running time of Merge Sort, Heap Sort and Quick Sort are O (nlogn) therefore these algorithms are also called O (nlogn) time algorithms. The Insertion Sort, Selection Sort and Bubble Sort take O (n^2) running time. These are called slow sorting algorithms and expensive in the sense of running time. Their use is not so much in these days. It is mathematically proved that any comparison based sorting algorithm take at least O (nlogn) running time. Therefore we should select a sorting algorithm among O (nlogn) running time algorithms when sorting is comparison based

5.Refereces

[1]. Robert Sedgewick, "Algorithms in Java", parts 1-4 3rd edition, 2003.

[2]. T. H. Cormen, C. E. Lieserson, R. L. Rivest and S. Clifford, "Introduction to Algorithm", 3rd ed., The MIT Press Cambridge, Massachusetts London, England 2009.

[3]. A. Jehad, M. Rami "An Enhancement of Major Sorting Algorithms," The International Arab Journal of Information Technology, January 2010

[4]. B. Ashutosh, M. Shailendra, Comparison of Sorting Algorithms based on Input Sequences, International Journal of Computer Applications, September 2013.

[5]. V. Mansotra, Kr. Sourabh, Implementing Bubble Sort Using a New Approach, Proceedings of the 5th National Conference; INDIACom-2011, Computing For Nation Development, March 10 –11, 2011.

[6]. R. Harish, Manisha, Runtime Bubble Sort – An Enhancement of Bubble Sort, International Journal of Computer Trends and Technology (IJCTT), August 2014.

[7]. Savina, K. Surmeet, Study of Sorting Algorithm to Optimize Search Results, International journal of emerging trends and technology in computer science, January – February 2013.

[8]. S. Pankaj, Comparison of Sorting Algorithms (On the Basis of Average Case), International Journal of Advanced Research in Computer Science and Software Engineering,, March 2013.

[9]. Gaurav Kocher, Nikita Agrawal, Analysis and Review of Sorting Algorithms, International Journal of Scientific Engineering and Research, March 2014

