Performance Comparison of Different Sorting Algorithms

Purvi Prajapati*, Nikita Bhatt#, Nirav Bhatt*

#U & P U Patel Department of Computer Engineering, CSPIT, CHARUSAT, Gujarat, India
*Department of Information Technology, CSPIT, CHARUSAT, Gujarat, India

***Abstract*— Sorting is the basic operation in most of the applications of computer science. Sorting means to arrange data in particular order inside computer. In this paper we have discussed performance of different sorting algorithms with their advantages and disadvantages. This paper also represents the application areas for different sorting algorithms. Main goal of this paper is to compare the performance of different sorting algorithms based on different parameters.**

***Keywords*— Algorithm, Time Complexity, Space Complexity**

I. INTRODUCTION

In computer application sorting is the process of arranging data in particular order. The process of sorting arranges numerical data in increasing or decreasing order and text data in alphabetical order. Now a day's big issue is to handle large amount of data in computer and for that sorting is an essential task. Sorting improves the efficiency of searching particular data in computer. There are many techniques available for sorting. The selection of the particular sorting technique depends on type of data. A Sorting is an important and widely studied issue, where the execution time and the required resources for computation is of extreme importance, especially if it is dealing with real-time data processing. So it is essential to study and to compare its performance for all the available sorting algorithms. [1, 2, 4, 6]

The importance of sorting lies in the fact that searching can be optimized to a very fast, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Following are some of the real life examples of sorting: Words in dictionary, File system in Directory, Book Index, Event calendar, Musical compact disks, Attendance sheet, etc.

II. TYPES OF SORTING TECHNIQUES

There are many categories for the sorting techniques. Depending upon the category of the algorithm we could analyze the sorting algorithm. [2]

A. Internal and external sorting

If sorting process is performed within main memory than it is referred as an internal sorting. If amount of data is so large that requires secondary memory for the soring process than it is referred as an external sorting. This technique is also referred as in-place sorting and not-in place sorting.

B. Stable and Not Stable Sorting

The sorting process maintains the same sequence of the data with same values is referred as a stable sorting. And if after Sorting process order of the data with same values is changed than it is referred as not-stable sorting.

C. Adaptive and Non-Adaptive Sorting

A sorting algorithm is said to be adaptive, if it takes advantage of already sorted data in the list that is to be sorted.

That is, while sorting if the input has some data already sorted, adaptive algorithms will take this in to account and will not apply sorting on them.

A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They apply sorting process on all the data to confirm the desired sorted data.

D. Comparison based Sorting and Distribution based Sorting

In comparison based sorting process elements are compared with other elements to find element's correct place in the sorted list. In distribution based sorting all the elements are distributed over memory space and then group the elements to get sorted list.

E. In-place Sorting and Out of place Sorting

The sorting process maintains the same input space for generating output. The input is overwritten by exchanges the elements for generating the sorted output sequences. It requires constant amount of extra storage for the sorting process. Generally in place algorithms requires O(1) memory beyond the element being sorted.

The sorting process which requires some extra storage for the output is referred as Out of place sorting algorithm.

III. PERFORMANCE OF SORTING TECHNIQUES

Algorithm is efficient based on time complexity and space complexity. Time complexity is total amount of time required to execute the algorithm and space complexity means total amount of storage required in memory by the algorithm. Below table 1 represents characteristics of different sorting techniques. And table 2 shows advantages and disadvantages of sorting techniques.

TABLE 1: PERFORMANCE CHARACTERISTICS OF SORTING TECHNIQUES [1, 2, 5, 8, 10]

Technique	**Approach**	**Data Structure**	**Time Complexity n : number of input elements**			**Worst case Space Complexity**	**Stable**	**In place**
			Best Case	Average Case	Worst Case			
Bubble Sort	Comparison Based	Array	O(n2)	O(n2)	O(n2)	O(1) auxiliary	Yes	Yes
Modified Bubble Sort	Comparison Based	Array	O(n)	O(n2)	O(n2)	O(1) auxiliary	Yes	Yes
Selection Sort	Comparison Based	Array	O(n2)	O(n2)	O(n2)	O(n) total, O(1) auxiliary	Yes	Yes
Enhanced Selection Sort	Comparison Based	Array	O(n2)	O(n2)	O(n2)	O(n) total, O(1) auxiliary	Yes	Yes
Insertion sort	Comparison Based	Array	O(n)	O(n2)	O(n2)	O(n) total, O(1) auxiliary	Yes	Yes
Merge Sort	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(nlogn)	O(n) total, O(n) auxiliary	Yes	No
Quick sort	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(n2)	O(n) auxiliary	Yes	Yes
Parallel Quick sort	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(nlogn)	O(n) auxiliary	Yes	Yes
Randomized Quick Sort	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(nlogn)	O(n) auxiliary	Yes	Yes
Hyper Quick sort	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(nlogn)	O(n) auxiliary	Yes	Yes
MQ sort (Combination of Merge and Quick Sort)	Divide and Conquer	Array	O(nlogn)	O(nlogn)	O(nlogn)	O(n) auxiliary	Yes	Yes
Radix Sort k : range of data elements	Non Comparison Based	Array, Queue	O(kN)	O(kN)	O(kN)	O(n+ k)	No	No
Counting Sort k : range of data elements	Non Comparison Based	Array	O(n+k) ≈ O(n)	O(n+k) ≈ O(n)	O(n+k) ≈ O(n)	O(n+k)	Yes	No
Heap Sort		Array, Tree	O(nlogn)	O(nlogn)	O(nlogn)	O(1) auxiliary	NO	Yes
Bucket Sort k : number of buckets	Non Comparison Based	Array	O(n+k)	O(n+k)	O(n2)	O(n.k)	Yes	No

Table 2: ADVANTAGES AND DISADVANTAGES [9, 10]

Technique	**Advantage**	**Disadvantage**
Bubble Sort	-Simple and easy for implementation -efficient when input data is sorted	-Inefficient for large volume of input data
Selection Sort	-efficient for small amount of input data.	-perform all n comparisons for sorted input data -Inefficient for large volume of input data
Enhanced Selection Sort	-number of exchanges are less compare to Selection Sort	
Insertion sort	-Simple and efficient for small input data	-Inefficient for large volume of input data
Merge Sort	-Efficient for less amount of input data	-Requires extra memory for sorting
Quick sort	-Fast and Efficient for large amount of input data	-Inefficient for sorted input data
MQ sort (Combination of Merge and Quick Sort)	-Extra memory is not required	
Radix Sort	-efficiency not depended on type and size of data -efficiently handle large amount of input data	-Occupy more memory
Counting Sort	-Uses key values as a	-Inefficient for large

k : range of data elements	indexes	amount of data -Inefficient for string data
Heap Sort	-use tree structure to represent elements	-Slower because it builds tree structure for sorting
Bucket Sort k : number of buckets	-Efficient whenever input is uniformly distributed over range.	-Inefficient for large amount of data -Requires more memory space

IV. APPLICATIONS OF SORTING TECHNIQUES

Sorted data is suitable for searching of any information or data by applying binary search algorithm. Most of the commercial organizations like government organizations, academic institutes, financial institutes, health care sectors requires information's in sorted order. Here the information's are sorted by numerical field or text field, files are arranged by name or date, transactions are managed by time or place, mail/letter to be sorted by date or address, students data to be sorted by result or id number or name so whatever the field the data must be require some sorting mechanism.[1,3]

A. Searching

Searching is the basic step in most of the applications of the computer science. Binary search is most efficient whenever we have large amount of input data. The input data must be in sorted sequence in case of binary search. So one of the most common application of sorting is searching process.

B. Kruskal's Algorithm

Kruskal's algorithm is used to generate Minimum Spanning Tree (MST) from the given graph. The first step of this algorithm is sorting according to the weights of their edges. Its running time complexity depends on the sorting process.

C. Closest Pair Problem

In this problem we need to find pair of elements that have smallest difference. Input elements are arrange in sorted sequence than closest pair will present next to each other.

D. Frequency Distribution

For given input elements, which element present largest number of times in the input sequence. So if input elements are arranged in sorted sequence than we will easily find frequency of each element.

There are different application areas according to different sorting techniques. Below given table 3 shows the applications of different sorting techniques.

Table 3: APPLICATIONS OF SORTING TECHNIQUES

Algorithm	Applications
Insertion Sort	It is more suitable for small amount of input data. Also suitable for online data sorting process. Efficient for input data which are already sorted or 99% sorted.
Quick Sort	Compare to all other sorting algorithms quick sort is fastest and no additional memory is required. It gives worst case response time for the critical applications which require guaranteed response time. Applications such as life monitoring in medical sector, aircraft controlling, monitoring of dangerous materials on industrial plants etc.
Merge Sort	In most of the e-commerce applications Merge Sort is used.
Bubble Sort	Efficient for sorted input data
Insertion Sort, Selection Sort, Bubble Sort	Not require extra memory space for sorting process.
Quick Sort	To make excellent usage of the memory hierarchy like virtual memory or caches. Well suited to modern computer architectures.
Counting Sort	Counting sort is more efficient if the range of input data is smaller (1 to k) than the number of data (1 to n) to be sorted. i.e. k<=n

V. CONCLUSION

This paper presents the survey and performance characteristics of different sorting techniques along with comparative analysis. It also presents application area and pros & cons of different sorting techniques. The selection of the efficient algorithm depends on the various factors of the problem: size of the problem, memory constraint, type of the input etc. It concludes that sorting technique is problem specific.

REFERENCES

[1]. Ali W, Islam T, Rehman Hu, Ahmad I, Khan M, Mahmood A. Comparison Of Different Sorting Algorithms. International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE). 2016 Jul 28;5(7):pp-63
[2]. Renu R, Manisha M. MQ Sort an Innovative Algorithm using Quick Sort and Merge Sort. International Journal of Computer Applications. 2015 Jul;122(21):10-4.
[3]. Alnihoud J, Mansi R. An Enhancement of Major Sorting Algorithms. Int. Arab J. Inf. Technol.. 2010 Jan 1;7(1):55-62.
[4]. Karunanithi AK. A Survey, Discussion and Comparison of Sorting Algorithms. Department of Computing Science, Umea University. 2014 Jun.
[5]. Kocher G, Agrawal N. Analysis and Review of Sorting Algorithms. IJSER. 2014 Mar.
[6]. Rajput IS, Kumar B, Singh T. Performance comparison of sequential quick sort and parallel quick sort algorithms. International Journal of Computer Applications. 2012 Jan 1;57(9).
[7]. Almutairi AH, Alruwaili AH, Alabdullatif A. Improving of Quicksort Algorithm performance by sequential thread Or parallel algorithms. Global Journal of Computer Science and Technology. 2012 Jul 16; 12(10-A).
[8]. Rajput IS, Kumar B, Singh T. Performance comparison of sequential quick sort and parallel quick sort algorithms. International Journal of Computer Applications. 2012 Jan 1;57(9)
[9]. Sareen P. Comparison of sorting algorithms (on the basis of average case). International Journal of Advanced Research in Computer Science and Software Engineering. 2013 Mar 3;3(3):522-32.
[10]. Karunanithi AK. A Survey, Discussion and Comparison of Sorting Algorithms. Department of Computing Science, Umea University. June 2014.