iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** IX **Month of publication:** September 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

Comparison Among Different Sorting Techniques

Ajay Kumar*[1], Bharat Kumar*[2], Chirag Dawar*[3], Dinesh Bajaj*[4]

[1,2,3,4] *(B.Tech 3rd sem) Dept. of Computer Science and Engineering, Dronacharya College of Engineering, Gurgaon 122001, India*

Abstract: Ordering of data is very important. To understand any data it must be in ordered form, even computer cannot perform operations on randomly stored data. The process of ordering of data is known as sorting. There are varioustypes of sorting, in this research paper we are going to study 7 types of sorting techniques and we will also compare them. These are: Selection sort, Bubble sort, Insertion sort, Merge sort, Quick sort, Radix sort and Shell sort. Comparison is done on the basis of time complexity of sorting techniques. We will not use space complexity as memory problems are negligible in today's world.

Keywords: Sorting; Quick sort; Merge sort; Selection sort; Insertion sort; Bubble sort; Complexity; Space complexity; Time complexity.

I. INTRODUCTION

Sorting: The process of ordering of elements is known as sorting. It is very important in day to day life. Nor we neither computer can understand the data stored in an irregular way. Sorting of comparisons can be done on the basis of complexity.

Complexity:Complexity of an algorithm is a measure of the amount of time and/or space required by an algorithm for an input of a given size (n).there are two types of complexity:

- Space complexity
- time complexity

Space complexity measures the space used by algorithm at running time.

Time complexity for an algorithm is different for different devices as different devices have different speeds so, we measure time complexity as the no. of statements executed in different cases of inputs.

II. SORTING TECHNIQUES

1.Selection Sorting:-In selection sort we find the smallest number and place it at first position, then at second and so on.

Complexity: -

An array in sorted or unsorted form doesn't make any difference. It is same in both best & worst cases. The first pass makes (*n*-1) comparisons to find smallest number, second pass makes (*n*-2) and so on, then Time Complexity T(n) will be :

$$= (n-1)+(n-2)+(n-3)+.....+2+1$$

$$= \frac{n(n-1)}{2}$$

$$= O(n^2)$$

2.Bubble Sort: - It is also called as exchange sort. The sorting technique makes comparison between pair of consecutive element from beginning to end of list. After comparison, bigger element moves towards end. In this way, the biggest element comes at end and process repeats for (*n*-1) elements. It needs an extra variable to store data while swapping which increases space complexity by 1.

Complexity:-

Best Case: - A case when list is already sorted and in that case outer loop runs (n-1) time and it will not swap elements in inner loop.

So, Time Complexity T (n) will be

ISSN: 2321-9653

= O (n)

Worst Case :- A case when list is in decreasing order and in that case outer loop runs (n-1) time and elements get swapped (n-1) times in first pass, (n-2) times in second & so on, then Time Complexity T(n) will be :

$$T(n) \leq (n-1)+(n-2)+(n-3)+.....+2+1$$
$$\leq \frac{n(n-1)}{2}$$
$$= O(n^2)$$

3.Insertion Sort: -It takes list in two parts, sorted list and unsorted list. In this sorting technique, first element of unsorted list gets placed in previous sorted list and runs till all elements are in sorted list.

Complexity:-

Best Case: -All elements are sorted or almost sorted. Therefore, comparison occurs atleast one time in inner loop, then time Complexity T(n) will be :

$$= (n-1)+(1+1+.....)$$
$$= (n-1)+(n-1)$$
$$= 2(n-1)$$
$$= O(n)$$

Average Case: - We consider that there will be approximately (n-1)/2 comparisons in inner loop.

$$T(n) \leq \frac{1}{2}+\frac{1}{2}+.....\frac{n-1}{2}$$
$$\leq \frac{n(n-1)}{4} = O(n^2)$$

Worst Case: - In this case comparisonin innerloop is done almost one in first time, 2 times in second turn, and (n-1) times in (n-1) turns.

$$T(n) \leq 1+2+.....(n-1)$$
$$\leq \frac{n(n-1)}{2} = O(n^2)$$

4.Merge Sort :- In this sorting technique, list will be partitioned in sub arrays and then they will be merge in sorted order with the help of an additional array which is the only disadvantage.

Complexity: - Average and Worst Case

This technique requires at most log *n* passes to merge & sort *n* elements of list, each pass requires at most *n* comparisons then T(n) for both cases will be :

$$O(n\log n)$$

5.Quick Sort: - It is based upon divide & conquers strategy. In this technique, we consider first element of array or sub-array as pivot element and provides its position in list to it and that position divides the list into parts.

Complexity:-

Worst Case: - It is a case in which one sub-array contains zero or one element and all other are in second sub-array.

Let Cn be the time required to makes position between n elements.

$$T(n) = Cn + T(0) + T(n-1)$$
$$= Cn + C(n-1) + T(n-2)$$
$$.$$
$$.$$
$$.$$
$$= Cn + C(n-1) + C(n-2) + ...C(1) + T(0)$$
$$= C[n + (n-1) + (n-2) + ...2 + 1]$$
$$= C\left[\frac{n(n+1)}{2}\right] = \frac{Cn^2}{2} + \frac{Cn}{2} = O(n^2)$$

Best Case: - It is a case in which array is divided from mid then

$$T(n) = Cn + T(n/2) + T(n/2)$$

ISSN: 2321-9653

$= \quad Cn + 2T\left(n/2\right)$

Let n is a power of 2 i.e. $n = 2^k$.

$T(2^k) = 2T\left(2^{k-1}\right) + C2^k$

$= 2\left[2T\left(2^{k-2}\right) + C.2^{k-1}\right] + C.2^k$

.

.

.

$= 2^k T\left(2^{k-k}\right) + kC2^k = nT(1) + k\log nCn$

=O (nlogn)

6.Radix Sort: - This sorting technique process individual digits of the numbers from Least Significant Digit (LSD) to Most Significant Digit for a stable sorting. In first pass it process LSD of every number & store their relative order in an array & in other pass thenext highersignificant digit and so on till the Most Significant Digits are processes and numbers get sorted. This technique is also used to sort the names alphabetically.

Complexity:-

Let be the no. of digits in greatest number then it require S passes &*d* be the no. of digits of format of number. A pass *k* will compare dik with each of the digits of format. Therefore it requires at most *d* comparisons in each pass. Therefore, total comparisons algorithm is

s.d.n

Time complexity depends on number of comparisons. Therefore

T (n) ≤ O (s, d, n)

7.Shell Sort: - This technique is mainly based on insertion sort. In a pass it sorts the numbers when are separated at equal distance. In each consecutive pass distance will be gradually decreases till the distance becomes 1. It uses insertion sort to sort elements with a little change in it.

Complexity: - Shell sort analysis is very difficult some time complexities for certain sequences of increments are known.

Base Case: - O (n)

Average Case: - $n\log^2 n$ or $n^{3/2}$

Worse Case: - It depends on gap sequence. The best known is $n\log^2 n$.

III. COMPARISON OF SORTING TECHNIQUES

Algorithm	Time Complexity			Extra space
	Best case	Worst case	Average case	
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	1
Bubble sort	O(n)	$O(n^2)$	$O(n^2)$	1
Insertion sort	O(n)	$O(n^2)$	$O(n^2)$	1
Merge sort	O(nlogn)	O(nlogn)	O(nlogn)	n(Worst case)
Quick sort	O(nlogn)	$O(n^2)$	O(nlogn)	logn(average),n(worst case)
Radix sort	O(s.d.n)	O(s.d.n)	O(s.d.n)	n
Shell sort	O(n)	$n\log^2 n$	$O(n\log^2 n)$ or $O(n^{3/2})$	1

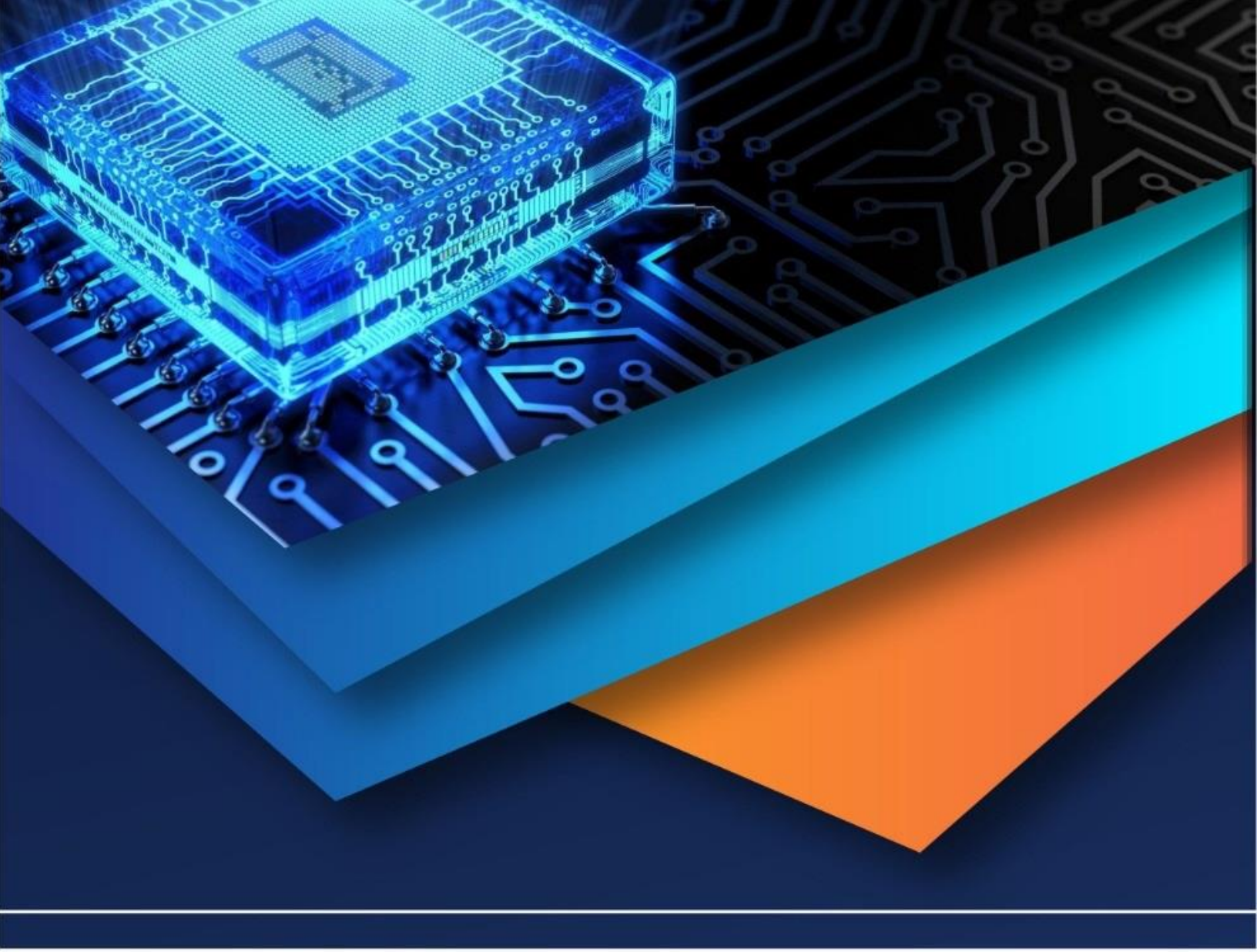
IV. CONCLUSION

Different sorting techniques have different uses according to their behaviour for different inputs, every sorting technique has its own best case and worst case according to inputs. Selection sort is useful where swapping is costly. Normally, Insertion sort is use for small data sets. For large data sets, Merge sort and Quick sort are useful. Merge sort is practically useful for

humans and Quick sort is poorly suited for humans. For restricted data (data in a fixed interval) radix sort is useful. Bubble sort is rarely used, but found in teaching and theoretical expressions. Nowadays, Shell sort is rarely used in serious applications. It does not use thestack, some implementations of the quick sort function in the Cstandard library targeted at embedded systems use it instead of quick sort. An implementation of Shell sort is present in the Linux kernel.

REFERENCES

[1]Dixit.J.B, Fundamentals of computers and programming in c, Laxmi Publication pvt.Ltd., second edition 2010.
[2]http://en.wikipedia.org/wiki/Sorting_algorithm
[3]http://en.wikipedia.org/wiki/Shellsort/Applications
[4]http://www.cprogramming.com/tutorial/computersciencetheory/sortcomp.html
[5]Seymour Lipschutz, Data Structures using C, McGraw-Hill, 2011



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY