International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 02 | Feb-2016 www.irjet.net p-ISSN: 2395-0072

SORTING ALGORITHMS

Neelam Yadav*, Sangeeta Kumari**

**HOD in CSE Dept, DAV college, Kanina*
***Lecture in CSE Dept, DAV College, Kanina*

Abstract: *This paper presents different type of sorting that are present in data structure for example quick, insertion, heap and merge. Each algorithm tries to solve sorting problem using different formats. These four algorithms have their own pros and cons. This paper presents a detailed study of how these algorithm works and compares them on the basis of various parameters to reach a conclusion.*

Keywords: Quick Sort, Selection Sort, Insertion Sort, Bubble Sort, Shell Sort, Cocktail Sort, Comparison, Other Performance Parameters.

1. INTRODUCTION

A sorting algorithm is an algorithm that arranges the elements of a list in a certain order; the order can be increasing or decreasing. Till now many sorting algorithm has been discovered. Some of them were comparison based sort like insertion sort, selection sort, bubble sort, quick sort and merge sort while other were non comparison based sort. When we try to sort any type of list, arrays etc. we first compares element with one another then swaps or copies those elements if necessary.

It continues executing over and over until the whole array or list is sorted. Such algorithms are known as Comparison based sorting.

Sorting algorithms often classified by [1]:

Internal Sorting: if data are sorted directly in main memory

External Sorting: if data are sorted in auxiliary memory like hard disk, floppy disk, etc.

System complexity: In terms of computational. In this algorithm can be classified on basis of performance like worst case, average case, and best case.

Computational complexity: In terms of number of swaps. Each algorithm performs various numbers of swaps in order to sort.

Memory usage

Stability: stable sorting algorithms maintains the relative order if records with equal keys.

Sorting algorithms are sometimes characterized by big O notation. This notation indicates performances of algorithms and the amount of time that the algorithms take. The different cases that are popular in sorting algorithms are:-

- O(n) is fair, graph increases in the smooth path.

- O(n^2): this is inefficient because a small increase in input increases the graph tremendously.

- O(n log n): this is considered as efficient, because it shows the slower pace increase in the graph as the size of array or data is increased.

2. WORKING PROCEDURE OF ALGORTIHMS

Bubble Sort: Bubble sort is a comparison based sort. It is simplest among all comparison based sort. In this, comparison is done among adjacent elements and if the top data is greater than the data below it then they are swapped [2]. It continues doing this for each pair of adjacent elements till the end of the data set is reached. It again starts comparing first two elements, repeating until no swap occurs in pass. Unfortunately, it is a slowest sorting method as compared to other sorting algorithms.

Algorithm [3]***:-***

Here N, K is a variable whose value is element position and A is
Array of length
[1-N].

BUBBLE_SORT (A)
1. For N=1 to length[*A*]-1 (for pass)

2. For k=1 to length[*A*]-N(for comparison)

3. If A[K]>A[K+1]

4. Swap [A(K) , A(K+1)]

[End if] [End of inner loop] [End of outer loop]
5.Exit

Example:

election sort: It is a sorting algorithm that belongs to in- place comparison sorting. It has O(n^2) complexity, making it inefficient for large data sets

Algorithm:-

Here N, K,LOC is a variable whose value is a element position, A
is Array of length [1-N] and min is minimum value of array A.

SELECTION_SORT (A) [4]

1. for N=1 to length[A]-1 (finding minimum value for pass)

2. min=A [N]

3. for K=N+1 to length[A] (for comparison)

4. if (min>A [N])

5. min=A [K], Loc=K [End if] [End of inner loop]

6. Swap (A [Loc],A[N]) [End of OUTER loop]

7. Exit

Example:

	A[1]	A[2]	A[3]	A[4]	A[5]
DATA	55	33	44	11	22
	55	33	44	11	22
PASS 1	33	55	44	11	22
	33	55	44	11	22
	11	55	44	33	22
	11	55	44	33	22
	11	55	44	33	22
PASS 2	11	44	55	33	22
	11	33	55	44	22
	11	22	55	44	33
PASS 3	11	22	55	44	33
	11	22	44	55	33
	11	22	33	55	44
PASS 4	11	22	33	55	44
	11	22	33	44	55
SORTED DATA	11	22	33	44	55

Insertion Sort: Insertion sort is a naive algorithm. It belongs to the family of comparison sorting. It is efficient for sorting arrays with small size. It works by taking elements from the list one by one and inserting them at their correct position into a new sorted list.

Insertion sort is an example of an incremental algorithm. In this sorting we scan the given list from 1 to n, inserting each element into its proper position through comparison. For sorting time n-1 passes or steps are required.

Algorithm:-

Here K, J is a variable whose value is a element position and A is
Array of length [1-N].

	A[1]	A[2]	A[3]	A[4]	A[5]
DATA	55	33	44	11	22
	55	33	44	11	22
PASS 1	33	55	44	11	22
	33	55	44	11	22
	11	55	44	33	22
	11	55	44	33	22
	11	55	44	33	22
PASS 2	11	44	55	33	22
	11	33	55	44	22
	11	22	55	44	33
PASS 3	11	22	55	44	33
	11	22	44	55	33
	11	22	33	55	44
PASS 4	11	22	33	55	44
	11	22	33	44	55
SORTED DATA	11	22	33	44	55

INSERTION_SORT (A)

1. For K=2 to length [A] (for pass)

2. item= A [K], S=K-1 (for minimum number K-1 comparison)

3. WHILE S>0 and item<A[S]

4. A[S+1]=A[S]

5. S=S-1 END WHILE LOOP

6. A [S+1]=item .END FOR
LOOP

Example:

	A[1]	A[2]	A[3]	A[4]	A[5]
DATA	55	33	44	11	22
PASS 1	55	33	44	11	22
PASS 2	33	55	44	11	22
PASS 3	33	44	55	11	22
PASS 4	11	33	44	55	22
SORTED DATA	11	22	33	44	55

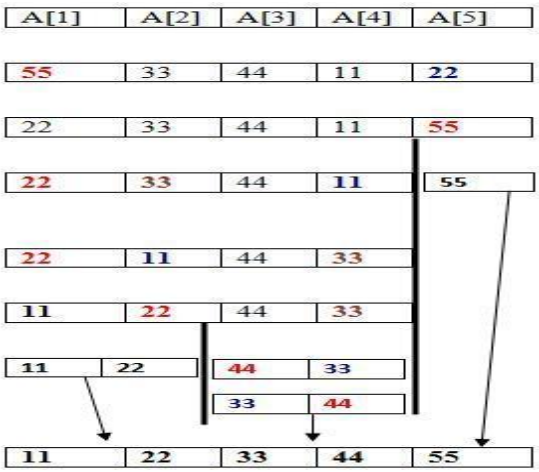
Quick Sort: Quick sort was developed by Sir Charles Antony Richard Hoare [5] (Hoare 1962). It belongs to the family of exchange sorting. Quick sort is an in-place, divide-and-conquer algorithm. It is massively recursive sort and is also termed as a partition-exchange sort.

Divide: The list is divided by choosing a partitioning element (pivot element). One list contains all element less than or equal to the partitioning element while the other list contains all element greater than the partitioning element.

Conquer: after these two lists are recursively partitioned in the same way till the resulting lists become trivially small to sort by comparison.

Combine: Lastly sorted smaller list is combined to produce the sorted list which contains the entire input element.

Example:



Shell sort: *This sort* was invented by Donald Shell in 1959. It is improvement over bubble sort and insertion sort. One way to implement is to arrange the data sequence in a two-dimensional array and then sort the columns of the array using insertion sort.

This algorithm is inefficient for large data sets but when comes to sorting smaller data set it perform well. It is one of the fastest algorithms for sorting array with smaller size.

Cocktail sort: It is also termed as bidirectional bubble sort. Cocktail shaker sort, shaker sort (which can also refer to a variant of selection sort), ripple sort, shuttle sort or happy hour sort are other name for this [6]. It is a variation of bubble sort. It is both a stable sorting algorithm and a comparison based sort. The algorithm differs from bubble sort in way where sorting is done in both directions in each pass through the list. This sorting algorithm is only marginally more difficult to implement than bubble sort, and solves the problem with so-called turtles in bubble sort.

Name	**Average Case**	**Worst Case**	**Stable**
Bubble Sort	O(n^2)	O(n^2)	yes
Insertion Sort	O(n^2)	O(n^2)	yes
Quick Sort	O(n log n)	O(n^2)	no
Cocktail Sort	-	O(n^2)	yes
Selection Sort	O(n^2)	O(n^2)	no

CONCLUSION

This paper discusses six comparison based sorting algorithms and their example. Quick Sort is faster for than Bubble Sort, Selection Sort and Insertion Sort when it comes to sort large data sets.

REFERENCES

[1] R. Lavore, Arrays, Big O Notation and simple Sorting. Data Structures and Algorithms in Java.

[2] P. K. A. S. G. Eshan Kapur, "Proposal of a two way sorting algorithm and performance with existing algorithms,"*Internationsl Journal of computer Science, Engineering and application,* vol. 2, no. 3, 2012.

[3] D. Knuth, in *The art of programming sorting and searching*,1988.

[4] T. a. Cormen, Introduction to Algorithms, 2001.

[5] J.-p. T. T. M. Hill, An introduction to Data Structure with application.

[6] J. Phongsai, "Research paper on Sorting Algorithm," 2009.