Comparison of Sorting Algorithms based on Input Sequences

Ashutosh Bharadwaj
Dept. of Computer Science & Engineering
BTKIT, Dwarahat
Almora, Uttarakhand

Shailendra Mishra, Ph.D
Head of Dept. of Computer Science & Engineering
BTKIT, Dwarahat
Almora, Uttarakhand

ABSTRACT
Ordering is a very important for mankind .If anything is in unordered then it will not easily understand by anyone but if it is in order then it will easily understand and used by anyone. So ordering is a very important issue in computer science also. In computer science many programming applications use ordering to solving a problem either it is in ascending or descending order. In this paper we discuss four sorting algorithms which are already existed named as Insertion Sort, Bubble Sort, Selection Sort, Merge Sort and we design a new sorting algorithm named as index sort also. In this paper we check the performance and comparison of all five sorting algorithm on the basis of increasing the no of elements in bulk. We check how much processing time is taken by all four sorting algorithms with Index Sort and compared them and finding which sorting algorithm takes less time to sort the elements like 10, 100, 1000, 10000 . If any algorithm takes less processing time it means that it sorts the element faster than others. The processing time of a sorting algorithm is based on the processing speed of a Processor as well as internal memory (RAM) used by the system.

Keywords
Insertion Sort; Bubble Sort; Selection sort; Merge Sort; Index Sort.

1. INTRODUCTION
Sorting is a fundamental operation in computer science. Sorting generally refers to the operation of ordering data in a given sequence such as increasing sequence or decreasing sequence. There are many fundamental and advance sorting algorithms. All sorting algorithm are problem oriented means they work well on some special problem and do not work well for any kind of a problems. Sorting algorithms are applied on specific kind of problems. Sorting algorithms are used for small number of elements, some sorting algorithms are used for large numbers, some sorting algorithms are used for floating number of data, and some are used for repeated values in a list. We sort data in numerical order or alphabetical order, arranging the list either in increasing order or decreasing order and alphabetical value like addressee key. In this paper we discuss four sorting algorithms which are already exist named as Insertion Sort, Bubble Sort, Selection Sort, Merge Sort and we design a new sorting algorithm named as index sort. In this paper we check the performance and comparison of all five sorting algorithm on the basis of increasing the no of elements in bulk. We check how much processing time is taken by all four sorting algorithms with Index Sort and compared them and finding which sorting algorithm takes less time to sort the elements like 10, 100, 1000, 10000 . If any algorithm takes less processing time it means that it sorts the element faster than others.

2. Sorting Algorithms

2.1 Insertion Sort
This sorting technique is used for small number of elements. The insertion sort works like playing cards in which each card is placed at its proper place while playing in hands of a person. Cards are placed in an order which is also called a sorting order .Sorting a hand of playing card is one of the real examples of insertion sort. This algorithm first sorts the first two elements of the array. It then inserts the third element in its proper place in relation to the first 2 sorted elements. This process continues until all of the remaining elements are inserted in their proper position. Insertion sort can take different running time to sort two input sequences of the same size of array depending upon how nearly they already sorted. It sorts the element of small array very fast but in sorting the elements of big array takes very long time.

Algorithm

1. For I=2 to N
2. A [I] =item, J=I-1
3. WHILE j>0 and item<A[J]
4. A[J+1]=A[J]
5. J=J-1
6. A [J+1]=item

Following are the procedure to sort a given set of element {8, 7, 1, 2}.

Data	8	7	1	2
Pass 1	7	8	1	2
Pass 2	1	7	8	2
Pass 3	1	2	7	8

2.2 BUBBLE SORT
The sorting algorithm is affectionately named the "bubble" sort. This sorting algorithm is perhaps one of the simplest sorting algorithms in terms of complexity. It makes use of a sorting method known as the exchange method. This algorithm compares pairs of adjacent elements and makes exchanges if necessary. The name comes from the fact that each element "bubbles" up to its own proper position. Here is how bubble sort would sort the integer array 8 7 1 2,It is a simplest sorting algorithm used in computer science algorithm. If we have 100 elements then the total number of

comparison is 10000. Obviously, this algorithm is rarely used except in education.

Algorithm
1. For I=1 to N-1 (for pass)

2. For k=1 to N-I (for comparison)

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

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

Following are the procedure to sort a given set of element {8, 7, 1, 2}.

Data	8	7	1	2
Pass 1	7	8	1	2
	7	1	8	2
	7	1	2	8
Pass 2	1	7	8	2
	1	7	2	8
Pass 3	1	2	7	8

Here there are 4 elements so number of comparison=8+7+1+2=19
And the number of pass=4-1=3.

2.3 SELECTION SORT

It is among the most intuitive of all sorts. In selection sort we find out the smallest elements in each pass and placed it in proper location. After that these steps are repeated until all the list of element is sorted. This is the simplest method of sorting. In this method, to sort the data in increasing order, the first element is compared with all the elements. If first element is greater than smallest element than interchanged the position of elements. So after the first pass, the smallest element is placed at the first position. The same procedure is repeated for 2nd element and so on until the element of list is sorted.

Algorithm
1**.** for I=1 to N-1

2**.** min=A [I]

3**.** for K=I+1 to N

4**.** if (min>A [I])

5**.** min=A [K], Loc=K

6**.** Swap (A [Loc],A[I])

7**.** Exit

Following are the procedure to sort a given set of element {8, 7, 1, 2}

Data	8	7	1	2
Pass 1	1	7	8	2
Pass 2	1	2	8	7
Pass 3	1	2	7	8

2.4 MERGE SORT

The operation that combines the element of two list in a one single sorted list is called Merging. Merging operation is used in Merge sort for sorting an array. Merge sort is based on the divide-and-conquer technique. We state each subproblem as sorting a subarray $A[p... r]$. Initially, $p = 1$ and $r = n$, but these values change as we recurse through subproblems. To sort $A[p ... r]$:

Divide Step- If a given array A has empty, simply return; then it is already sorted. Otherwise it breaks $A[p .. r]$ into two subarrays $A[p .. q]$ and $A[q + 1 .. r]$, each containing about half of the elements of $A[p .. r]$. That is, q is the halfway point of $A[p .. r]$.

Conquer Step-= Conquer by recursively sorting the two subarrays $A[p .. q]$ and $A[q + 1 .. r]$.

Combine Step- Combine the elements back in $A[p .. r]$ by merging the two sorted subarrays $A[p .. q]$ and $A[q + 1 .. r]$ into a sorted order. By executing this step, we will introduce a procedure MERGE (A, p, q, r). Note that the recursion bottoms out when the subarray has just one element, so that it is trivially sorted. Advantage of merge sort is that they are well suited for large data set. Disadvantage of merge sort is that At least twice the memory requirements than other sorts.

Algorithm
MERG-SORT (A, p, r)
1. IF $p < r$ // Check for base case
2. THEN q = FLOOR[$(p + r)/2$] // Divide step
3. MERGE (A, p, q) // Conquer step.
4. MERGE (A, $q + 1, r$) // Conquer step.
5. MERGE (A, p, q, r) // Conquer step.

MERGE (A, p, q, r)
1. $n_1 \leftarrow q - p + 1$
2. $n_2 \leftarrow r - q$
3. Create arrays L[1 ... $n_1 + 1$] and R[1 .. . $n_2 + 1$]
4. For $i \leftarrow 1$ TO n_1
5. do L[i] ← A[$p + i - 1$]
6. For $j \leftarrow 1$ TO n_2
7. do R[j] ← A[$q + j$]
8. L[$n_1 + 1$] ← ∞
9. R[$n_2 + 1$] ← ∞
10. $i \leftarrow 1$
11. $j \leftarrow 1$
12. For $k \leftarrow p$ TO r
13. do IF L[i] ≤ R[j]
14. THEN A[k] ← L[I]
15. $i \leftarrow i + 1$
16. ELSE A[k] ← R[j]
17. $j \leftarrow j + 1$

Following are the procedure to sort a given set of element {7, 4, 1, 3, 2, 5, 9, 7}.

Data	6	4	1	3	2	5	9	7
Pass 1	4	6	1	3	2	5	7	9
Pass 2	1	3	4	6	2	5	7	9
Pass 3	1	2	3	4	5	6	7	9

2.5 INDEX SORT

This sorting algorithm is the most simple and easy to use .This algorithm performs working from lower index as well as higher index and counting the elements from lower index to higher index vice versa and fixing the elements indexes.

In this sorting algorithm first element of lower index is checked with all smaller element with itself and if found and count all small elements and add the lower index in count.

Then from opposite count the sum (lower index +count) and put it swaps the value with first element if smaller elements are not found then fixed their position. While we taking element from upper index we count greater elements then we subtract from upper index and if greater element is not found the fixed their position.

Algorithm

```
arraysize= n;
MyArray = rand(arraysize,1);
MyArray= [1020, 940, 80, 370, 60, 503, 40, 330, 20, 410];
low_ind=1;
up_ind=arraysize;
counter1=0;
counter2=0;
while(low_ind<up_ind)
 z=0;
for j=low_ind+1 to up_ind
if(MyArray(low_ind)>MyArray(j))
counter1=counter1+1;
end
end
if(counter1>0)
t=MyArray(low_ind);
MyArray(low_ind)=MyArray(counter1+low_ind);
MyArray(counter1+low_ind)=t;
counter1=0;
z=1;
else
low_ind=low_ind+1;
end
j=up_ind-1;
while (j>=low_ind)
if(MyArray(up_ind)<MyArray(j))
counter2=counter2+1;
end
 j=j-1;
end
if(counter2>0)
t=MyArray(up_ind);
MyArray(up_ind)=MyArray(up_ind-counter2);
MyArray(up_ind-counter2)=t;
counter2=0;
 z=1;
else
up_ind=up_ind-1;
end
end
```

Data	8	7	1	2
Pass 1	2	7	1	8
Pass 2	7	2	1	8
Pass 3	1	2	7	8

4. PERFORMANCE ANALYSIS

Bubble Sort, Selection Sort, Merge Sort, Insertion Sort Index Sort were applied in MATLAB and we test the random sequence input of length 10, 100, 1000, 10000 to check the performance. All the five sorting algorithms were executed on machine with 32-bit Operating System having Intel(R) Core 2 Duo processor @ 2.13 GHz, 2.13 GHz and installed memory (RAM) 2.00 GB. The times taken by the CPU at execution for Different inputs are shown in the table. The Plot of length of input and CPU time taken (m sec) is shown in figure. Result shows that for all small length of input elements sequence the performance of all the five techniques is all most same, but for the large input element sequence Merge sort is faster than Bubble sort, Sort and insertion Sort, Selection Sort and Index Sort. From the results it can be concluded that Index Sort algorithm is working well for all length of input values. It takes less CPU time than the Insertion Sort, Bubble Sort, Selection Sort, and Merge Sort for small inputs but for larger inputs it takes greater time then Insertion Sort, Selection Sort, Merge Sort but takes less time than in case of Bubble Sort.

Table 1: Processing time (m sec) for different lengths of input sequences

Sorting Techniques	10	100	1000	10000
Insertion Sort	.000012	.000208	.017085	1.51616
Bubble Sort	.000027	.000491	.034503	2.540110
Selection Sort	.000029	.000147	.012923	1.60680
Merge Sort	.000011	.000126	.01116	1.04148
Index Sort	.000012	.000128	.022339	2.348677

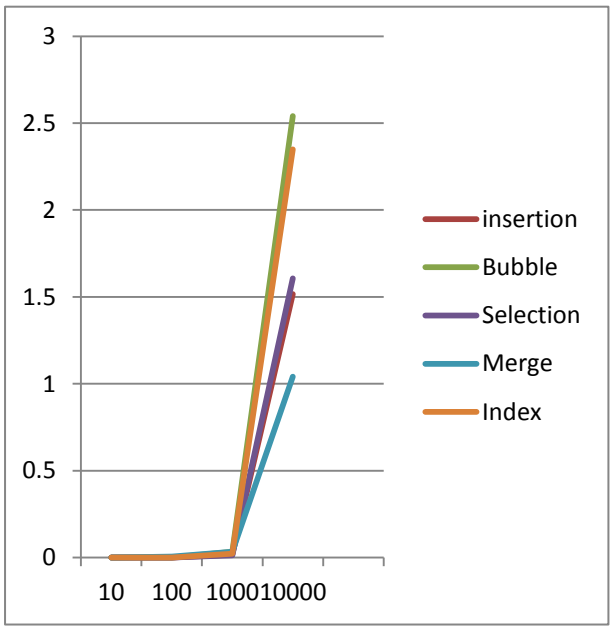

Figure1- X axis- time (m sec) Y axis input sequence (10, 100, 1000, 10000

5. CONCLUSION AND FUTURE SCOPE

From the results it can be concluded that Index Sort algorithm is working well for all length of input values. It takes less CPU time than the Insertion Sort, Bubble Sort, Selection Sort and Merge Sort for small inputs and it takes greater time in case of Insertion sort, Selection Sort Merge sort but takes less time than in case of Bubble Sort for larger inputs. In the future work more effective sorting algorithm can be proposed with running time.

6. ACKNOWLEDGMENTS

Our thanks to Dr. Shailendra Kumar Mishra (Head of Department) of Computer Science, Principal, BTKIT Dwarahat, for providing necessary infrastructure for the research work and helping me to write this paper. I would also like to thanks to all my colleagues who give me a valuable support in writing this paper.

6. REFERENCES

[1] Comparison of Sorting Algorithms (On the Basis of Average Case) Pankaj Sareen.

[2] A Comparison Based Analysis of Four Different Types of Sorting Algorithms in Data Structures with Their Performances.

[3] Min-Max Select Bubble Sorting Algorithm**.**

[4] CSCE 3110Data Structures & Algorithm Analysis.

[5] Assortment of different sorting algorithms. Amardeep Singh, Monika**,** Vandana, Sukhnandan Kaur.

[6] Robustness versus Performance in Sorting and Tournament Algorithms by Wilfried Elmenreich, Tobias Ibounig, István Fehérvári.

[7] Seymour Lipschutz (2009) Data Structure with C, Schaum Series, and Tata McGraw-Hill Education.

[8] Merge sort:- Merge sort algorithm, C. BronTechnological Univ., Eindhoven, The Netherlands, Communications of the ACM Volume 15 Issue 5, May 1972, ACM New York, NY, USA .

[9] Review on sorting algorithms A comparative study on two sorting algorithms By Pooja Adhikari.

[10] An Enhancement of Major Sorting Algorithms Jehad Alnihoud and Rami Mansi.

[11] Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, fifth Indian printing (Prentice Hall of India private limited), New Delhi-110001