ISSN No:-2456 –2165

A New Approach to Improve Selection Sort by the Modified Selection Sort (MSSA) and Performance Comparison

Kumar Nishant[#1], Pawan Mishra[#2], Swati[#3]
CSE Department , Uttarakhand Technical University , Dehradun, India

Abstract—Everything in this world has some advantages and disadvantages. Sorting is a data structure operation, which is used for making easy searching and arranging of element or record. There are many fundamental and advance sorting algorithms. Some sorting algorithms are problem specific means they work well on some specific problem not all the problem. Sorting algorithms help searching data quickly and in this way its saves time. Arranging the record or element in some mathematical or logical order is known as sorting. Mainly sorting may be either numerical or in alphabetical. In numerical sorting we will sort numeric value either in increasing order or decreasing order and in alphabetical sort we will sort the alphabetical value. Sorting is an algorithm that arranges all elements of an array, orderly. Sorting involves rearranging information into either ascending or descending order. In computer science and mathematics, sorting algorithm is an algorithm that puts elements of a list in a certain order, not necessarily in increasing order; it may be in decreasing order as well. Efficient sorting is important to optimizing the use of other algorithms that require sorted list to work efficiently; it is also useful for producing human-readable output. Most simple sorting algorithms involve two steps which compare two items and swap two items or copy one item. In this thesis we present a new sorting algorithm, named as Modified Selection Sort algorithm, which is faster than selection sort. After studying various sorting algorithms; I found that there are no such sorting algorithms which work on the basis of selecting two elements at a time, means selecting two elements simultaneously. We also compare Modified Selection Sort algorithm with selection sort. We have used the MATLAB for implementation. The new algorithm is analyzed, implemented & tested.

Keyword:-About Sorting, Selection Sort, Algorithm, Complexity of HSSA, Comparison between HSSA and MSSA.

I. INTRODUCTION

The whole thing in this world has some positive and negative point. The sorting is an important operation to utilize for arranging and searching of data easily in computer science. Many of researchers has done on advance and essential sorting algorithms. Few sorting algorithms are problem noticeable means they work carefully on some specific problem not completely on the problem. Sorting algorithms mainly help to searching element or data rapidly and in this way it is time saving operation. The Arrangement of the data in few numerical or logical orders is known as sorting. This operation can be alphabetical, mathematical, or in numerical. The process of selection sort start from the selecting smallest unsorted item remains in the list that after swapping it with the item in the position to be filled. The main problem in selection sorting is O (n^2).The poorest case as well as average case difficulty of Selection sort is O(n^2), where the representation of is the total number of items in the given array to be sorted and the selection sort is the unwanted step child of the n^2sorts.It produces a 60% performance enhancement over the bubble sort, but the insertion sort is almost double of the bubble sort and is just as easy to implement as the selection sort. In brief, there is not actually any reason to use the selection sort-use the insertion sort instead. The basic problems of computer science are ordering a list of elements. There is a overabundance of solutions to this problem, called as sorting algorithms. Some of the sorting algorithms are straightforward and natural, such as the bubble sort. Others, for example the quick sort are very doubtful, but generate lightning-fast out come. The classification of ordinary sorting can occur by means of complexity of their algorithms. There is a proper link among the twist of an algorithm and its proportional competence. Algorithmic complexity is normally written in a form recognized as Big-O notation, where O shows the difficulty of the algorithm and a*n* represent the amount of the set the algorithm is run nearby. The two module of sorting algorithms in this thesis has O(n^2), which include the bubble, the insertion, the selection, and the shell sorts after that O(*n* log n) which involves the heap sort, the merge sort, and the quick sort. It is not regularly likely to tell that single algorithm is bigger to another, the same as relative performance can adjust depending on the class of data being sorted. In many of the cases, data are completely mixed positive in a random sort and in other the data will be likely to be in inverse order. Many of the algorithms will run in another way due to the data being sorted. The four types of ordinary algorithms are the bubble sort, the selection sort, insertion sort and the quick sort [1, 2].

As the beginning of computing, the sorting crisis has paying attention a vast deal of study, may be because of the complexity of solve it capably in spite of its easy, informal statement. It is impossible to read that one sorting algorithm is improved than the other sorting algorithm. There are so many sorting algorithms which depend on its performance. Sorting is use in most of the main applications such that it has been sufficient of performance analysis. However, most of the research is based on the algorithm theoretical difficulty or their non-cached structural design. Since mainly computers today contain cache, it is important to examine them depend onto their cache performance. Quick sort was laid down to be an excellent sorting algorithm in conditions of typical theoretical difficulty and cache performance.

In computer science, sorting is a basic operation. It is used usually in a vast variety of significant applications. This Database application can be used through banks, schools, and other institutions hold sorting code. There proposed for the significance of sorting in these applications, a lot of sorting algorithms has been developed over of the decades by changing difficulty. Slow sorting methods such as the bubble sort, the insertion sort, and selection sort contains a theoretical difficulty of O (n^2).Although these algorithms are very slow in sorting large arrays, the algorithm is easy, hence they are not deficient. If an application merely requires sorting small arrays, then it is satisfactory to utilize one of the easy slow sorting algorithms as remonstrant to a faster, but very difficult sorting algorithm. On behalf of these applications, the growth in coding time and chance of coding error in using the fastest sorting algorithm is not value the go faster in implementation time. It is cleared that, if an application requires a faster sorting algorithm, there are certainly several ones available, including the quick sort, the merge sort, and the heap sort. These algorithms have a difficulty of O (n log n). These are quicker than the O (n^2) algorithms and be able to sort enormous arrays in a suitable quantity of time. Although, the cost of these speedy sorting techniques is that the algorithm is very difficult and is complex to exactly code. But the explanation of the very difficult algorithm is capable sorting method able of being used to sort enormous arrays [2].

II. LITERATURE SURVEY

In this paper we will study about the performance and comparison between Hybrid Selection Sort Algorithm (HSSA) and Modified Selection Sort Algorithm (MSSA).
Here , we will discuss about some papers which are based on the sorting.

Sanjib Palui and Somsubhra Gupta et al **[3]** introduced as chunk of acquaintance was increase day by day in the world throughout us and stipulation of few renovated operations is sorting list, the accomplished and cost dominant sorting algorithm is requisite. There are respective numbers of sorting algorithms but still now this field has tempted a fantastic deal of research, probably because of the complexity of extricating it proficiently and effectively despite of its effortless and intimate descriptions. An algorithm is selected to one's requirement with respect to space complexity and time complexity. These days, space is accessible respectively in low cost. So, time complexity is main problem for an algorithm. Therein, the offered perspective is to instate sequential time complexity applying divide and conquer rule by separating an issue into n (input size) sub problems, then these sub problems are extricated recursively. So, asymptotic capacity of this algorithm is very advanced with respect to time.

J B Hayfron-acquah, Obed Appiah and K Riverson et al [4] proposed One of the underlying issues of Computer Science is sorting a list of items. It dispatches to the regime of numerical or alphabetical or character data in statistical manner. Bubble, Insertion, Selection, Merge, and Quick sort are most general ones and they all have separate executions based on the range of the list to be sorted. As the size of a list goes up, few of the sorting algorithm spin to execute better than others and most cases programmers choose algorithms that execute well even as the size of the input data growth. As the size of dataset grows, there is always the opportunity of repeat or some form of redundancies taking place in the list. For an example, list of ages of students on a university campus is eventually to have oodles of them duplicating. A latest algorithm is introduced which can execute sorting quicker than most sorting algorithms in such cases. The developed selection sort algorithm is a correction of the current selection sort, but here the number of passes required to sort the list is not simply based on the size of the list, but the number of separate significances in the dataset. This proffer a far better execution as contrasted with the decrepit selection sort in the case where there are redundancies in the list.

Obed Appiah and Ezekiel Mensah Martey et al [5] proposed Sorting was a list of items is one underlying work in many applications applied on the computer. The word characterize the regime of a set of items in a few command to make analysis and technology very simple.

Numerous sorting algorithms happen though its capacity and memory space expenditure had taken place a main problem when it has to be executed. Virtually, programmers choose sort algorithms that executes well even as the size of the input data grows. A latest algorithm, Magnetic Bubble Sort Algorithm (MBS) has been introduced in this study. The MBS is an enlargement of the bubble sort algorithm which proposes a far better execution in the case where redundancies happen in the list. The run time of the MBS hinge on the number of separate significances that are found in the list to be sorted. The developed bubble sort algorithm is very easy to analysis, pondering the truth that the time complexity of the algorithm hinge on two major causes that is the size of the list (n) and number of separate significance in the list.

Mirza Abdulla et al [6] introduced sorting and searching was the most elemental issues in computer science. Sorting is applied for the most of the times to maintain in exploring. One of the most familiar sorting algorithms that are educated at communicatory computer science courses is the

traditional selection sort. While specious algorithms is simple to decode and sense at the communicatory computer science level, it is far from being an accomplished sorting technique, because it needs O(n2) time to sort a list of n numbers. It does so by frequently searching the least. We surveyed the advantage of decreasing the finding time for the least on each pass of the algorithm, and demonstrate that we can gain a worst case time bound of O(n n2) by constructing only concise modifications to the input list. Thereby our bound is a reason O(n2) of quicker than the traditional selection sort and another traditional sorts like insertion and bubble sort.

Obed Appiah and Ezekiel Mensah Martey et al [7] introduced sorting a list of items was one underlying work in different applications used on the computer. The word characterizes the regime of a set of items in a few order to make explication and technology very simple. Manifold sorting algorithms happen nevertheless its capacity and memory space expenditure become a main problem when it has to be applied. Virtually, programmers choose sort algorithms that execute well even as the size of the input data grows. A latest algorithm, Magnetic Bubble Sort Algorithm (MBS) has been introduced in this study. The MBS is an enlargement of the bubble sort algorithm which proposes a far better execution in the case where redundancies happens in the list. The run time of the MBS hinge on the number of separate significances that are found in the list to be sorted. The developed bubble sort algorithm is very easy to analysis, pondering the truth that the time complexity of the algorithm hinge on two major causes that is the size of the list (n) and number of separate significance in the list.

Partha Sarathi Dutta et al [8] proposed The sorting issue inquired us to reorganizing the items of a specified list either in scansorial, reducing or lexicographic. The items may be in any form such as numerical, alphabetical or alphanumeric. There are numerous sorting algorithm that have explored but there was no algorithm that would be the best solution in all circumstances. Few of the algorithms are good for miniature size of input, another for huge amount of input. Hence, there is always a requirement to create an accomplished sorting algorithm. An accomplished algorithm is needed for improving the use of another algorithm like finding in the database which works quick only on data that are already in sorted manner. This paper introduced a latest sorting algorithm such as - Hybrid Selection Sort Algorithm, HSS All which modulates the techniques of old selection sort algorithm with expurgation and other algorithm such as end-to-end comparisons algorithm. This HSSA has collated with the old selection sort algorithm and demonstrated that HSSA executed better by decreasing the number of proportions as well as the number of passes.

III. SYSTEM OVERVIEW

A. Selection Sort

Selection sort is among the mainly intuitive of every sort. The fundamental rule of the selection sort is to notice the slightest items in every pass and positioned it in correct position. Repetitive these steps while the list is sorted. It is the easy technique of sorting. Herein method, to sort the item in increasing order, the first item is compared with every the items. Suppose the first item is greater than the smallest item than swapped. Therefore later than the first pass, the minute's element is located at the first position. The similar process is repeated for second item and so on while the list is sorted [9].

Pass	P[1]	P[2]	P[3]	P[4]	P[5]	P[6]
K=1, LOC=4	3	4	2	1	6	5
K=2, LOC=3	1	4	2	3	6	5
K=3, LOC=4	1	2	4	3	6	5
K=4, LOC=4	1	2	3	4	6	5
K=5, LOC=6	1	2	3	4	6	5
Sorted	1	2	3	4	5	6

Table 1: Example of Selection Sort

B. Algorithm :-

1. for k'=1 to N'-1
2**.** min=a' [k']
3**.** for j'=k'+1 to N'
4**.** if (min>a' [j'])
5**.** min=a' [j'], Loc=j'
6**.** Swap (a' [Loc],a'[k'])
7**.** Exit.

C. How to Determine Complexity

In common, the running time of a piece of code how can you determine? The response is that it's based on what types of statements are used [10].

1. Sequence of statement

2. Statement T1;

3. Statement T2;

4.……

5. Statement Tk;Total time = time (statement T1) + time(statement T2) + …… + time (statement Tk)

If every statement is "simple" (merely involves fundamental operations) then the total time is constant and the time for each statement is constant: O(1). For examples, suppose the statements are simple unless noted otherwise.

6. if-then-else statements

7. if(cond){

8. sequence of statement T1

9. }

10.else{

11. sequence of statement T2

12. }

The time of worst-case is the slower of the two cases: max (time (sequence T1) time (sequence T2)). Such as, if sequence T1 is O (N) and sequence T2 is O (1). The time of the worst-case for the total if-then-else statement would be O (N).

13. for loops

14. for (x=0; x<N; x++){

15. Sequence of statements

16. }

The loop runs N times, thus the sequence of statements besides runs N times. Since then we think the statements are O(1) and the overall time for the loop is N*O(1), which is O(N) taken as a whole.

17. Nested loops

18. for (x=0; x<N; x++){

19. for (y=0; y<M; y++){

20. sequence of statements

21. }

22. }

The external loop runs N times. All time the external loop runs, the internal loop runs M times. So the outcome, the statement in the internal loop run an overall of N*M times. Therefore, the complexity is O(N*M). In a general exceptional case where the stopping condition of the internal loop is j<N instead of j<M (i.e., the internal loop besides runs N times), the overall complexity for the loops is $O(N^2)$.

The same rule applies, when a loop is involved. For example:for (y=0; y<N; y++)g(N); has complexity (N^2). The loop performs N times and every time call g (N).

D. Comparison of Various Sorts

Here we are going to compare among various sorting algorithms on the source of various main factors. From the given comparison, we can easily find out which algorithm is best according to my problem. The various factors of the comparison are as follows- Complexity, Memory, Stability and Method. The table given below tells that all the specific feature of all sorting which we have studied. By seeing the table in one sight we can know the various important factor of the sorting like worst complexity, average complexity, stability, method, and extra memory required etc.

Name	Average	Worst	Memory	Stable	Method	Other notes
Bubble sort	$O(n^2)$	$O(n^2)$	O(1)	Yes	Responding	Older Sort
Selection sort	$O(n^2)$	$O(n^2)$	O(1)	No	Selection	Can be implemented as a constant sort
Insertion sort	$O(n^2)$	$O(n^2)$	O(1)	Yes	Insertion	Average case is also O(n + d), where d is the digit of inversion

Table 2:-Comparison of Various Sorting [11]

IV. PREVIOUS WORK

Here we are going to discuss about the previous work which is compared with selection sort and hybrid selection sort(HSSA). In this paper we will show the working of HSSA, algorithm of HSSA which is discuss in my base paper. After that we will take a size of input and we will show its comparison between selection sort and HSSA.

A. Working of HSSA

The proposed Hybrid Selection Sort algorithm (HSSA) to sort the list of elements that are stored in the array a[0..n-1] works as follows [12]:-

Step 1 (end-to-end comparisons algorithm) :-In this step of the algorithm , we start with the first element a[0] and compare with the last element a[n-1], if a[0] is larger than a[n-1], then interchange a[0] and a[n-1]. Next, compare a[1] and a[n-2] and interchange if a[1] > a[n-2].This process is repeated until there is only one element in the middle of the array a or two consecutive middle elements are compared.

Step 2 (modified selection sort algorithm) :-The modified selection sort algorithm works as follows –Suppose, the list is already in sorted or almost sorted. In this case, the straight selection sort can be modified by introducing a single Boolean variable ‗FLAG' to signal that no interchange takes place during a pass. Set the FLAG=1, before entering

an iteration and set FLAG=0, if there is an interchange as well as a[i] > a[i+1]. If FLAG=1,after any pass, then the list is already sorted and there is no need to continue. This approach reduces the number of passes significantly.

B. Algorithm

```
1. L=0      //set lower and upper bound of array a
2. U=n – 1
3. While (L < U)
4. {
5. if(a[U] < a[L])
6. Swap(a[L], a[U])        //interchange L-th and U-th
                              //position element.
7. L=L+1                   //update lower & upper
8. U=U-1                   //bound of the array a
9. }
//Modified selection sort algorithm
10. for(i=10;i<=n-2;i++)
11. {
12. k=I                    //initialize variables
13. FLAG=1                 //list is sorted order
14. min=I                  //location of minimum element
15. for(j=i+1;j<n;j++)
16. {
17. if(a[j]<min)    //search for the smallest element
18. {
19. min=j
20.FLAG=0                  //list is not sorted order
21. }
22. else
23.{
24. if(a[k]>a[k+1])
25. FLAG=0
26. k=k+1
27. }
28. }
29. Swap(a[i], a[min]) //interchange minimum and i-th
                        //element
30. if(FLAG==1) //list is in sorted order;no need to
31. break                  //continue to the next pass
32.}
```

Size Of Input	Selection Sort	HSSA
N=50	1225	925
N=99	4851	3675
N=150	11175	8400
N=499	124251	93126
N=1000	499500	376244
N=4999	12492501	9371250

Table 3:-Worst Case Analysis (on the basis of no. of comparison)[12]

V. PROPOSED WORK

There had been different creators who had tried persistent endeavours for expanding the proficiency and execution of the sorting procedure. The proposed calculation depends on choice sort. The Proposed calculation fills in as:

Start from two component and pursuits the entire rundown until it finds the minimum esteem and second slightest esteem. The sorting places minimal esteem in any case and second slightest esteem in the second place, this procedure proceeds until the entire rundown is sorted. As it were, the proposed calculation intended to decrease the quantity of passes/correlations that are performed. It works by making N/2 ignores the contracting unsorted segment of the cluster, every time selecting the minimum and second slightest esteem. Those qualities are then moved into their last sorted position in one pass.

A. Proposed Algorithm:-

```
MSSA(ARRAY, n)

for (l= 1; l <= n-1; l = l+2)

{
        Min = ARRAY [l];

        Smin = ARRAY [l+1];

        Loc_Min = l;

        Loc_Smin = l+1;

        for (m = l+1; m <= n; m++)

        {
            If (ARRAY [m] < Min)

            {
                Smin = Min;

                Loc_Smin = Loc_Min;

                Min = ARRAY [m];

                Loc_Min = m;

             }

            elseif (ARRAY [m] < Smin)

            {
                Smin = ARRAY [m];

                Loc_Smin = m;

            }
```

```
        }
        if (Loc_Min ~= l)
        {
                temp = ARRAY [l];
                ARRAY [l] = ARRAY [Loc_Min];
                ARRAY [Loc_Min] = temp;
        }
        if (Loc_Smin == l)
        {
                Loc_Smin = Loc_Min;
        }
        if (Loc_Smin ~= l+1)
        {
                 temp1 = ARRAY [l+1];
                ARRAY [l+1] = ARRAY [Loc_Smin];
                ARRAY [Loc_Smin] = temp1;
        }
}
```

B. Working of Selection Sort:-

Let assume the elements are given to sort are: 25, 17, 31, 13, 2

Data	25	17	31	13	2				
First pass	2	17	31	13	25				
Second Pass	2	13	31	17	25				
Third Pass	2	13	17	31	25				
Fourth Pass	**2**	**13**	**17**	**25**	**31**				

Table 4: Working of Selection Sort Algorithm

Data	25	17	31	13	2
First Pass	2	13	31	17	25
Second Pass	**2**	**13**	**17**	**25**	**31**

Table 5: Working of Proposed Selection Sort Algorithm

VI. RESULTS

Here in this paper it shows the worst case analysis (On the source of number of comparisons) difference between selection sort, with design and analysis of Hybrid selection sort algorithm (HSSA) and A new approach to improve selection sort by modified selection sort algorithm (MSSA) and Performance comparison.

In this paper it will shows the following steps:-

- Number of Swaps.
- Number of Passes.
- Number of Comparisons

In a sorting algorithm, Worst case refers to the situation when the given set of numbers is in decreasing order while the required set has to be in increasing order.

Size Of Input	**Selection Sort**	**HSSA**	**MSSA**	**%Improvement Over Selection Sort**
N=50	1225	925	625	48%
N=99	4851	3675	2450	49%
N=150	11175	8400	5625	49%
N=499	124251	93126	62250	49.9%
N=1000	499500	376244	250000	50%
N=4999	12492501	9371250	6247500	50.1%

Table 6: Final Result Analysis

VII. CONCLUSION

In this paper, we have learned about algorithm which has many varieties and their assessment. There are Pro and Cons in each sorting algorithm. The fundamental sorting algorithms are basic sorting algorithm and we have tried to show how this disadvantage of the fundamental sorting algorithm has removed in advance sorting algorithm. We have compared the numerous sorting algorithms on the behalf of different factor such as complexity, passes count, comparisons count, etc. After the study of variety of sorting algorithm we find that there is no such algorithm, which work in this way that to sort the elements at MSSA. So we have proposed sorting algorithm, for implementation of the proposed algorithm we have used MATLAB.

VIII. ACKNOWLEDGMENT

It gives me great sense of pleasure to present the thesis of M.Tech undertaken during M.Tech, FinalYear. I owe special date of gratitude to Mr. Pawan Kumar Mishra (Assistant Professor), Department of Computer Science & Engineering, Faculty of Technology at Uttarakhand Technical University, Dehradun for his special constant support and guidance throughout the course of work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me.

REFERENCES

[1]. M. Thorup ,"Randomized Sorting in O (n log log Time and Linear Space Using Addition, Shift, and Bit-wise Boolean Operations", Journal of Algorithms, Volume 42, Number 2, pages- 205-230, February 2002.
[2]. Y.Han "Deterministic sorting in O(nloglogn) time and linear space", Proceeding of the thirty-fourth annual ACM symposium on theory of computing, Monteral Quebec, Canada, pages-602-608, 2002.
[3]. www.google.com/sorting algorithm/wikipedia.
[4]. D. Jim´enez-Gonz´alez, J. Navarro, and J. Larriba-Pey. CC-Radix: "A Cache Conscious Sorting Based on Radix Sort". In Euromicro Conference on Parallel Distributed and Network based Processing, pages 101–108, February 2003.
[5]. Y.Han, M.Thorup, "Integer Sorting in O(nloglogn) time and linear space" proceesings of the 43rd symposium on foundations of Computer Science, pages-135-144, 2003.
[6]. M. A. Bender, M. Farach-Colton and M. A. Mosteiro, "Insertion Sort is O(n log n)", Proceedings of the Third International Conference on Fun With Algorithms (FUN), pages- 16-23, 2004.
[7]. www.z-sword.blogspot.in/2014/02/advantages-and-disadvantages-of-sorting.html.
[8]. L. Bentley and R. Sedgewick. "Fast Algorithms for Sorting and Searching Strings", ACM-SIAM SODA, pages-360–369, 2003.
[9]. Aditya Dev Mishra, Deepak Garg "Selection of Best Sorting Algorithm" International Journal of Intelligent Information Processing" Volume II Issue II ISSN 0.973-3892, p. 233-238, 2008.
[10]. Selectionsort,http://www.algolist.net/algorithms/sorting/selection_sort.
[11]. http://inimino.org/~inimino/tests.
[12]. Partha Sarathi Dutta, "Design and Analysis of Hybrid Selection Sort Algorithm",International Journal of Applied Research and Studies (IJARS) ISSN: 2278-9480 Volume 4, Issue 7, July- 2015.