Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.801 – 806

RESEARCH ARTICLE

Insertion Sort with its Enhancement

Miss. Pooja K. Chhatwani
Lecturer, Department of Information Technology, Dr. N. P. Hirani Institute of Polytechnic, Maharashtra, India
pooja22.chhatwani@gmail.com

Abstract— ***An algorithm is precise specification of a sequence of instruction to be carried out in order to solve a given problem. Sorting is considered as a fundamental operation in computer science as it is used as an intermediate step in many operations. Sorting refers to the process of arranging list of elements in a particular order. The elements are arranged in increasing or decreasing order of their key values. There are many sorting algorithms like Quick sort, Heap sort, Merge sort, Insertion sort, Selection sort, Bubble sort and Shell sort. However, efforts have been made to improve the performance of the algorithm in terms of efficiency, indeed a big issue to be considered. Major Emphasis has been placed on complexity by reducing the Number of comparisons, hence reducing complexity. This research paper presents the shell sort, insertion sort and its enhancement, also gives their performance analysis with respect to time complexity.***

Keywords— Algorithm; Sorting; Insertion Sort; Shell sort; Complexity

I. INTRODUCTION

Algorithm is an unambiguous, step-by-step procedure for solving a problem, which is guaranteed to terminate after a finite number of steps. Sorting is generally understood to be the process of rearranging a given set of objects in a specific order and therefore, the analysis and design of useful sorting algorithms has remained one of the most important research areas in the field. Despite the fact that, several new sorting algorithms being introduced, the large number of programmers in the field depends on one of the comparison-based sorting algorithms like: Bubble, Insertion, Selection sort, Shell sort etc. Hence sorting is an almost universally performed and hence, considered as a fundamental activity.

The complexity of a sorting algorithm measures the running time of function in which 'n' numbers of items are sorted. The choice of which sorting method is suitable for a problem depends on various efficiency considerations for different problem. Three most important of these considerations are:

- The length of time spent by programmer in coding a particular sorting program.
- Amount of machine time necessary for running the program.
- The amount of memory necessary for running program.
- Stability-does the sort preserve the order of keys with equal values.

II. WORKING PROCEDURE OF ALGORITHMS

A. Insertion Sort

The insertion sort works just like its name suggests - it inserts each item into its proper place in the final list. In Insertion sort, the first iteration starts with comparison of 1^{st} element with 0^{th} element. In the second iteration the element is compared with 0^{th} and 1^{st} element. In general, in every iteration an element is compared with all elements. If at some point it is found the element can be inserted at a position then space is created for it by

© 2014, IJCSMC All Rights Reserved

shifting the other element one position right and inserting the element at the suitable position. This procedure is repeated for the entire element in the array.

1) Algorithm:

The algorithm for insertion sort having DATA as an array with N elements is as follows

INSERTION (DATA, N)

1. Set A[0] = -∞ [Initializes sentinel elements]
2. Repeat Steps 3 to 5 for K = 2 ,3,…N
3. Set TEMP = DATA[K] and PTR = K-1
4. Repeat while TEMP < DATA[PTR] :
   a) Set DATA[PTR+1] = A[PTR]
      [Moves element forward]
   b) Set PTR =PTR – 1
   [End of loop]
5. Set DATA[PTR+1] = TEMP
   [Inserts the elements in proper place]
   [End of step 2 loop]
6. Exit.

2) Example:

TABLE III
WORKING OF INSERTION SORT

Location	0	1	2	3	4	5	6	7
Data	77	33	44	11	88	22	66	55
1st Pass	**77**	**33**	44	11	88	22	66	55
2nd Pass	**33**	**77**	**44**	11	88	22	66	55
3rd Pass	**33**	**44**	**77**	**11**	88	22	66	55
4th Pass	**11**	**33**	**44**	**77**	**88**	22	66	55
5th Pass	**11**	**33**	**44**	**77**	**88**	**22**	66	55
6th Pass	**11**	**22**	**33**	**44**	**77**	**88**	**66**	55
7th Pass	**11**	**22**	**33**	**44**	**66**	**77**	**88**	**55**
Sorted	**11**	**22**	**33**	**44**	**55**	**66**	**77**	**88**

3) Analysis:

The implementation of insertion Sort shows that there are (n−1) passes to sort *n.* The iteration starts at position 1 and moves through position (n−1), as these are the elements that need to be inserted back into the sorted sub lists. The maximum number of comparisons for an insertion sort is (*n*−1) .Total numbers of comparisons are:

$(n-1)+(n-2)+\ldots+2+1=n(n-1)/2=O(n^2)$

Best Case = **O (n)**

Average Case= **O (n^2)**

Worst Case= O **(n^2)**

4) Pros and Cons:

Pros:-

- Insertion sort exhibits a good performance when dealing with a small list.
- The insertion sort is an in-place sorting algorithm so the space requirement is minimal.
- Insertion Sort algorithm is relatively simple and easy to implement.

© 2014, IJCSMC All Rights Reserved

Cons:-

- Insertion sort is useful only when sorting a list of few elements.
- The insertion sort repeatedly scans the list of elements each time inserting the elements in the unordered sequence into its correct position.

B. Shell Sort

Shell sort is also known as "Diminishing increment sort". Shell sort is a sorting algorithm, devised by **"Donald Shell"** in 1959, that is a generalization of insertion sort, which exploits the fact that insertion sort works efficiently on input that is already almost sorted. It improves on insertion sort by allowing the comparison and exchange of elements that are far apart. The last step of Shell sort is a plain insertion sort, but by then, the array of data is guaranteed to be almost sorted.

The given list of numbers are first sorted at distance 5 from each other and then resorted with distance 3 and finally insertion sort has been performed. After getting the sorted list with distance 3 from each other, simple insertion sort is performed to get the final sorted list.

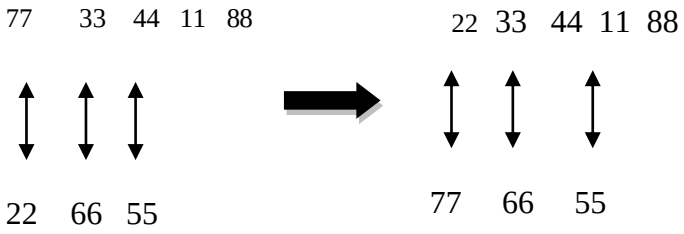
It is not mandatory to make 5, 3 and 1 as increments. Many other choices can also be made, but it should be considered that the choices like power of 2 such as 8,4,2,1 are not fruitful as the same keys compared in one pass would be compared again at the next pass. Therefore, making other choices may give better chance of obtaining new information from more of the comparisons.

1) Algorithm:

1. In the first pass or iteration, the elements are spitted with a gap to say, 5, 3 etc and are sub listed.
2. The sub list obtained after splitting is again sorted.
3. The sorted sub list is recombined.
4. The sorted list, in the second pass or iteration, is again sub listed with a gap (here it is 3)
5. Repeat through steps 2 and 3.

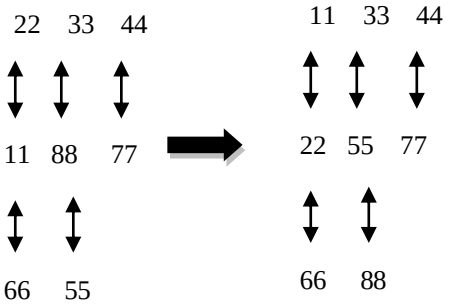
2) Example:

1. Let 77, 33,44,11,88,22,66,55 be the data sequence to be sorted. First, it is arranged in an array with distance of 5 (left), then the sub lists are sorted



Recombined list: 22, 33, 44, 11, 88, 77, 66, 55

2. In the next step, recombined list is again sub divided with distance of 3 (left) :



Recombined list: 11, 33, 44, 22, 55, 77, 66, 88

© 2014, IJCSMC All Rights Reserved

3. Now the sequence is almost completely sorted. After getting the sorted list with distance 3 from each other, now insertion sort is performed to get the final sorted list.

TABLE II
WORKING OF SHELL SORT

Location	0	1	2	3	4	5	6	7
Data	11	33	44	22	55	77	66	88
1st Pass	**11**	**33**	44	22	55	77	66	88
2nd Pass	**11**	**33**	**44**	22	55	77	66	88
3rd Pass	**11**	**33**	**44**	**22**	55	77	66	88
4th Pass	**11**	**22**	**33**	**44**	**55**	77	66	88
5th Pass	**11**	**22**	**33**	**44**	**55**	**77**	66	88
6th Pass	**11**	**22**	**33**	**44**	**55**	**77**	**66**	88
7th Pass	**11**	**22**	**33**	**44**	**55**	**66**	**77**	**88**
Sorted	**11**	**22**	**33**	**44**	**55**	**66**	**77**	**88**

3) Analysis:

The exact complexity of Shell Sort algorithm is still being debated. The analysis of shell sort may be quite difficult as the time requirement for specific sort depends on the number of elements in the array and their actual values. It is better if choice of increment is made of prime numbers as this guarantees that successive iterations or pass intermingle sub files so that entire file is indeed sorted and the span equals 1 on the last iteration.

Best Case = **O (n)**
Average Case= **O ($n^{3/2}$)**
Worst Case= **O n (log n)** [2]

4) Pros and Cons:

Pros:-

- Shell Sort runs faster than an insertion sort.
- It improves upon bubble sort and insertion sort by moving out of order elements more than one position at a time.
- Shell Sort is efficient for medium size list of elements.

Cons:-

- Shell Sort is that it is a complex algorithm and it's not nearly as efficient as the merge, Heap, and Quick sorts.
- Shell sort algorithm is not stable method, as its performance depends on the choice of increment sequence.

III. PROPOSED WORK

Bidirectional Insertion Sort [BIS]

There had been various authors who had made continuous efforts for increasing the efficiency and performance of the sorting process. The proposed algorithm is Optimization and Enhancements of Insertion Sort.

The proposed algorithm works in two steps:

- In first step (Pre-processing step), the first and the last element of the array is compared. If the first element is larger than the last element, then swapping of the elements is required. The position of the element from front end and element from the rear end of the array are stored in variables which are increased (front end) and decreased (rear end) as the algorithm progresses.
- In the second step, two adjacent elements from the front of the array are taken and are compared. Insertion of elements is done if required according to the order. Now similar process is carried as in Insertion sort.

© 2014, IJCSMC All Rights Reserved

1) Flowchart:

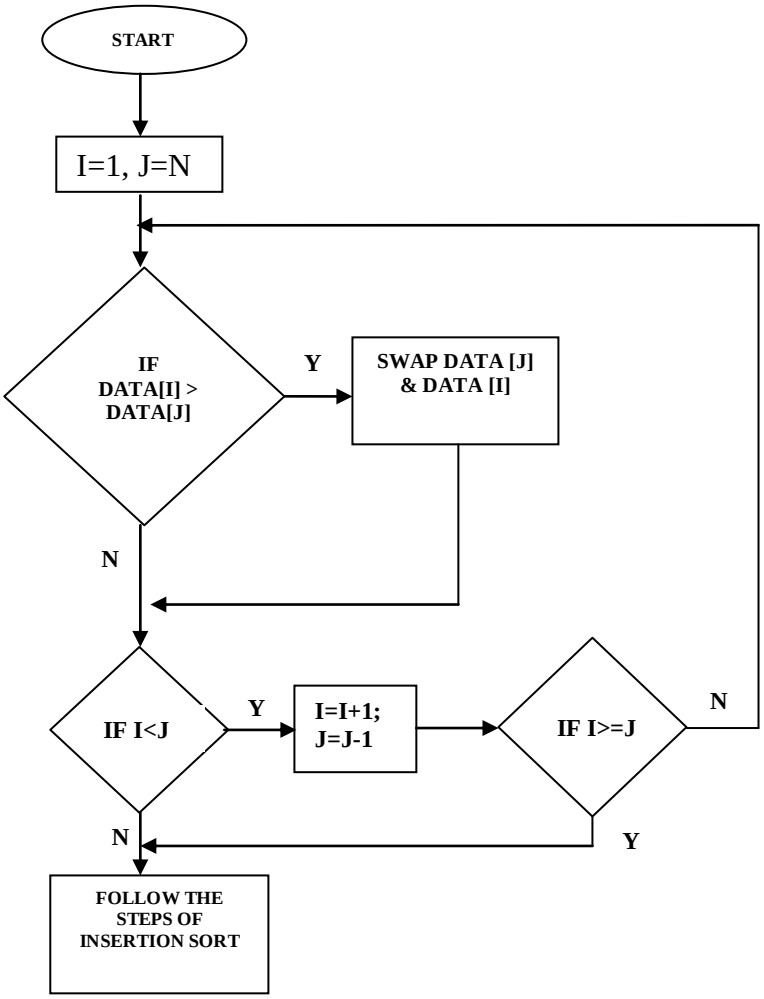


Fig 1 Flowchart of Bidirectional Insertion Sort

2) Example:

- Pre-processing step

TABLE III
WORKING OF BIDIRECTIONAL INSERTION SORT

Location	0	1	2	3	4	5	6	7
	77	33	44	11	88	22	66	**55**
	55	**33**	44	11	88	22	**66**	77
	55	33	**44**	11	88	**22**	66	77
	55	33	22	**11**	**88**	44	66	77
	55	33	22	11	88	44	66	77

- Insertion Sort

Elements	0	1	2	3	4	5	6	7
Data	55	33	22	11	88	44	66	77
1st Pass	**55**	**33**	22	11	88	44	66	77
2nd Pass	**33**	**55**	**22**	11	88	44	66	77
3rd Pass	**22**	**33**	**55**	**11**	88	44	66	77
4th Pass	**11**	**22**	**33**	**55**	**88**	44	66	77

© 2014, IJCSMC All Rights Reserved

5th Pass	**11**	**22**	**33**	**55**	**88**	**44**	66	77
6th Pass	**11**	**22**	**33**	**44**	**55**	**88**	**66**	77
7th Pass	**11**	**22**	**33**	**44**	**55**	**66**	**88**	**77**
Sorted	**11**	**22**	**33**	**44**	**55**	**66**	**77**	**88**

3) Pros and Cons:

Pros:-

- BIS algorithm is simple & easy to implement.
- It speeds up the Insertion Sort by eliminating many moves of entries.

Cons:-

- BIS algorithm is not stable

IV. CONCLUSION

This paper describes insertion sort, enhancement in insertion sort and making enhanced insertion sort more efficient for bigger list as well as smaller list.

REFERENCES

[1] Seymour Lipschutz and G A Vijayalakshmi Pai, *Data Structures*, (Tata McGraw Hill companies), Indian adapted edition 2006-07 West patel nagar,New Delhi-110063.

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

[3] Eshan Kapur, Parveen Kumar and Sahil Gupta,"Proposal Of A Two Way Sorting Algorithm And Performance Comparison With Existing Algorithms*" International Journal of Computer Science, Engineering and Applications* (IJCSEA) Vol.2, No.3, June 2012.

[4] Sultanullah Jadoon, Salman Faiz Solehria, Mubashir Qayum, "Optimized Selection Sort Algorithm is faster than Insertion Sort Algorithm: a Comparative Study*",International Journal of Electrical & Computer Sciences* IJECS-IJENS Vol: 11 No: 02.

[5] Ahmed M. Aliyu, Dr. P. B. Zirra, "A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays", *The International Journal of Engineering and Science* (IJES), volume 2, Issue-7.

[6] Tarundeep Singh Sodhi, Surmeet Kaur, Snehdeep Kaur, "Enhanced Insertion Sort Algorithm ", *International Journal of Computer Applications* (0975 – 8887) Volume 64– No.21, February 2013.

[7] G P Raja Sekhar, Lecture Notes:"Design & Analysis of Algorithms", Department of Mathematics I I T Kharagpur.

[8] Rupesh Srivastava, Tarun Tiwari Sweetesh Singh, "Bidirectional Expansion–. Insertion Algorithm for Sorting", 2009, Second *International Conference on Emerging Trend in Engineering and Technology*, ICETET-09.

[9] Aayush Agarwal, Vikas Pardesi , Namita Agarwal, "A New Approach To Sorting: Min-Max Sorting Algorithm" *International Journal of Engineering Research & Technology* (IJERT) Vol. 2 Issue 5, May – 2013.

© 2014, IJCSMC All Rights Reserved