Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 11 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Enhanced Novel Sorting Algorithm

By R. Srinivas
Surampalem University, India

Abstract - In computer science and mathematics; we can formulate a procedure for sorting unordered array or a file. Such procedure is always governed by an algorithm; which is called as Sorting Algorithm. Sorting is generally understood to be the process of rearranging a given set of objects in a specific order. The purpose of sorting is to facilitate the later search for members of the sorted set.

GJCST-C Classification : E.5

Strictly as per the compliance and regulations of:

Enhanced Novel Sorting Algorithm

R. Srinivas

Abstract - In computer science and mathematics; we can formulate a procedure for sorting unordered array or a file. Such procedure is always governed by an algorithm; which is called as Sorting Algorithm. Sorting is generally understood to be the process of rearranging a given set of objects in a specific order. The purpose of sorting is to facilitate the later search for members of the sorted set.

I. Introduction

The intelligent and most intuitive way to do this is to analyze the items in the array or list with each other and adapt them according to their relative order, moving an element forward or backward in the list depending on whether items next to it are greater or smaller. This is called a comparison sort.

Bubble sort algorithm[13], which is a simple sorting algorithm, works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order.

We considered three elements and move one element towards left or right and while moving this element other element moved one or two position towards opposite direction.

The main drawback of the proposed algorithm is that if smallest element is at last location then it requires n/2 iterations to move to first location. To overcome this draw back we proposed a modification to the sorting algorithm, in this for each iteration in first half the largest in the first half moved to first location of second half and in the second half, iteration start from the last element and in this iteration smallest in second half moved to the last location of first half of the elements. This procedure is repeated for n/2 times to arrange the elements in sorted order.

II. Proposed Modification

In each iteration in first half the largest in the first half moved to first location of second half and in the second half iteration start from the last element and in this second half iteration smallest in second half moved to the last location of first half of the elements. In successive iterations the smallest element moved to first half may be moved towards left if it is smaller than other elements in the first half same is true for the element moved to second half of the element. This procedure is repeated to arrange the elements in sorted order.

For example consider set of elements 9 4 6 7 8 3 9 5 2

Novel sorting algorithm

First iteration

Pass One

Element 4 compared with 9 and 6 arrange these elements in order. The order of elements after arrangement 4 6 9 7 8 3 9 5 2

Element 7 compared with 9 and 8 arrange the elements. The order of elements after arrangement 4 6 7 8 9 3 9 5 2

Element 3 compared with 9 and 9 arrange the elements. The order of elements after arrangement 4 6 7 8 3 9 9 5 2

Element 5 compared with 9 and 2 arrange the elements. The order of elements after arrangement 4 6 7 8 3 9 2 5 9

Pass Two

4 6 7 8 3 9 2 5 9
4 6 3 7 8 9 2 5 9
4 6 3 7 2 8 9 5 9
4 6 3 7 2 8 9 5 9
4 6 3 7 2 8 5 9 9

Pass Three

3 4 6 7 2 8 5 9 9
3 4 2 6 7 8 5 9 9
3 4 2 6 5 7 8 9 9

Pass Four

2 3 4 6 5 7 8 9 9
2 3 4 5 6 7 8 9 9

Using proposed modification

Consider same set of elements 9 4 6 7 8 3 9 5 2

Pass One

for first half

4 6 9 7 8 3 9 5 2
4 6 7 8 9 3 9 5 2

Second half

4 6 7 8 9 3 2 5 9
4 6 7 8 2 3 9 5 9

Pass Two

for first half

4 6 7 8 2 3 9 5 9
4 6 2 7 8 3 9 5 9

Second half

4 6 2 7 8 3 5 9 9
4 6 2 7 3 5 8 9 9

Pass three

for first half

2 4 6 7 3 5 8 9 9
2 4 3 6 7 5 8 9 9

Author : SSAIST, Surampalem, India.
E-mail : rayudu_srinivas@rediffmail.com

Second half

2 4 3 6 7 5 8 9 9
2 4 3 6 5 7 8 9 9

Pass Four
for first half

2 3 4 6 5 7 8 9 9
2 3 4 5 6 7 8 9 9

III. Algorithm

Input: List of elements a[0..n-1] where n is number of elements.

Step 1: swap=0
Step 2: repeat step 3 to 6 for j=0 to n/2 where step size=1
Step 3: repeat step 4 for i=1 to n/2 where step size=2
Step 4: compare elements a[i-1],a[i] and a[i+1]. If they are not in order arrange them in order. Set swap=1;
Step 5: repeat step 6 for k=n-1 to n/2 where step size=-2
Step 6: compare elements a[k-1], a[k] and a[k+1]. If they are not in order arrange them in order. Set swap=1;
Step 7: If swap=0 then given elements are in order break the outer loop else set swap=0.

a) Algorithm Analysis

The time for most sorting algorithms depends on the amount of data or size of the problem.[4]

Worst Case
The outer loop repeats for n/2 times.

In first pass of outer loop, the first inner loop repeats for (n/4) times and performs 3n/4 comparison operations and second inner loop repeats for (n/4) times and perform 3n/4 comparisons. The number of assignments performed depends on the order of elements. The maximum number of assignments performed is 2n.
In the second pass (n/2) in third pass (n/2)……
Inner loop repeats n/2+n/2+ ……..n/2 terms.
$=n^2/4$
Therefore the worst time complexity is $O(n^2)$

IV. Results

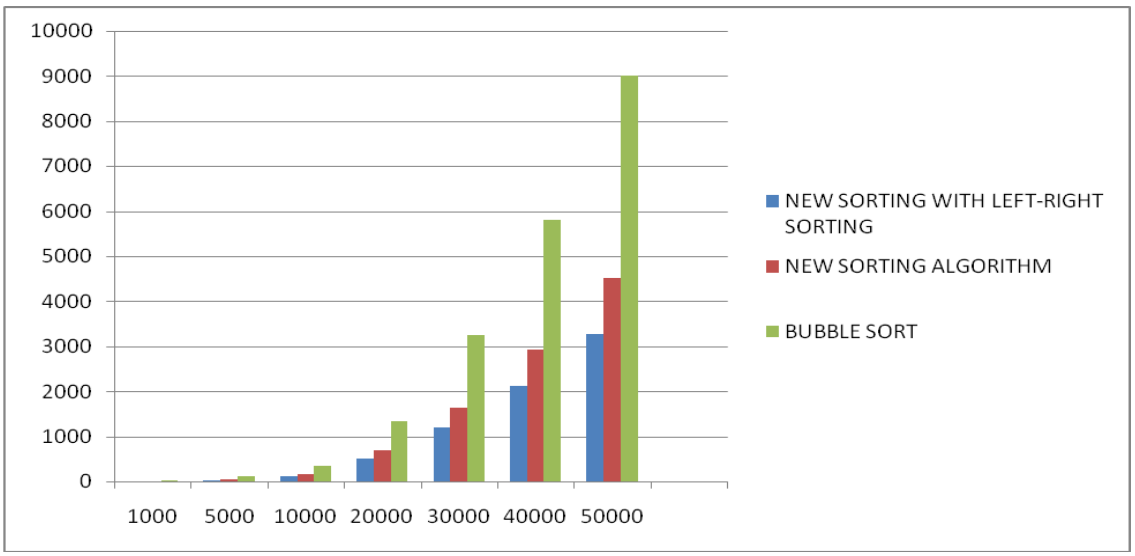
The time complexity of this algorithm in in worst case $O(n^2)$ same as bubble sort but their actual run time differ. For better understanding the actual performance we conducted some experiments.

The run times are measured on a PC, (AMD athlaon 220xd) processor and1G.B. RAM under Microsoft XP operating system. These algorithms are compiled using the sun java platform complier and run under the java interpreter. The run time shown is CPU execution time measured using object of Date class. The class Date available in java util package. The elements are generated using nextInt method of Random class. The same set of elements is used for algorithms.

Table 1 : For average execution times

	1000	5000	10000	20000	30000	40000	50000
New Sorting With Left-Right Sorting	1.58	32.2	118.22	502.52	1206.58	2114.72	3279.4
New Sorting Algorithm	2.84	45.74	167.76	692.56	1635.94	2918.8	4513.8
Bubble Sort	21.9	103.66	342.2	1345.84	3240.56	5814.6	9010.04

Graph 1 : Graph drawn for execution times



V. Conclusion

We have proposed modification to a novel sorting algorithm to sort given elements. The Proposed method uses three elements at a time to compare based on the result it arranges the elements. The proposed algorithm is easy to understand and easy to implement. We also proposed a modification of considering iterations to increase the speed of execution. The proposed novel algorithm has similarity like bubble sort that is in every phase one element moved to its correct location.

References Références Referencias

1. Krung Sinapiromsaran, "The sorted list exhibits the minimum successive difference", The Joint Conference on Computer Science and Software Engineering, November 17-18, 2005.
2. D.S. Malik, C++ Programming: Program Design Including Data Structures, Course Technology (Thomson Learning), 2002.
3. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. "Introduction to Algorithms". MIT Press, Cambridge, MA, 2nd edition, 2001.
4. Liu C. L., "Analysis of sorting algorithms", Proceedings of Switching and Automata Theory, 12th Annual Symposium, 1971, East Lansing, MI, USA, pp 207-215.
5. Francesc J.Ferri, Jesus Albert "An Analysis of selection sort using recurrence relations" Questho, vol. 20, pp-111-119 (1996).
6. Parag Bhalchandra*, Nilesh Deshmukh, Sakharam Lokhande, Santosh Phulari "A Comprehensive Note on Complexity Issues in Sorting Algorithms" Advances in Computational Research, ISSN: 0975–3273, Volume 1, Issue 2, 2009, pp-1-09.
7. Omar Khan Durrani, Shreelakshmi V, Sushma Shetty & Vinutha D C "Analysis and Determination of Asymptotic Behavior Range For Popular Sorting Algorithms" Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT) : 2231–5292, Vol.- II, Issue-1, 2.
8. Bubble Sort: An Archaeological Algorithmic Analysis Owen Astrachan *SIGCSE '03*, February 19-23, Reno, Nevada, USA. ACM 1-58113-648-X/03/0002
9. V. Estivill-Castro and D. Wood. "A Survey of Adaptive Sorting Algorithms", Computing Surveys, 24:441-476, 1992.
10. V. Estivill-Castro and D. Wood. "A Survey of Adaptive Sorting Algorithms", Computing Surveys, 24:441-476, 1992.
11. Hoffmann, J., Hofmann, M.: Amortized Resource Analysis with Polymorphic Recursion and Partial Big-Step Operational Semantics. In: 8th Asian Symp. on Prog. Langs. (APLAS'10). (2010)
12. Nadathur Satish, Mark Harris, Michael Garland," Designing Efficient Sorting Algorithms for Manycore GPUs"23rd IEEE International Parallel and Distributed Processing Symposium, May 2009.
13. Soubhik Chakraborty, Mausumi Bose, and Kumar Sushant, A Research thesis, On Why Parameters of Input Distributions Need be Taken Into Account For a More Precise Evaluation of Complexity for Certain Algorithms.