IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 11, November 2015.
www.ijiset.com
ISSN 2348 – 7968

A new one another Sorting Technique: N*3 Matrix Based Sorting Algorithm

Dr. S. Muthusundari, Associate Professor, R.M.D Engineering College

M. A. Berlin, Associate Professor, R.M.D Engineering College

Abstract

Sorting is the main key function required for many applications in today's activities. This paper proposes a new sorting algorithm based on the matrix representation. The data elements are stored in the matrix order representation, and it takes row wise and column wise comparisons to get the output data into ascending order. This propose algorithm takes O(nlgn) time complexity.

Keywords:

Matrix, row wise, column wise, compare, output

INTRODUCTION

In the field of computer science, sorting takes a vital role of ordering the elements. The basic process of sorting is the same: taking a list of items, and producing a permutation of the same list in arranged in a prescribed non-decreasing order. There are, however, varying methods or algorithms which can be used to achieve this. Amongst them are bubble sort, selection sort, insertion sort, Shell sort, and Quicksort. The purpose of this investigation is to determine which of these algorithms is the fastest to sort lists of different lengths, and to therefore determine which algorithm should be used depending on the list length. Although asymptotic analysis of the algorithms is touched upon, the main type of comparison discussed is an empirical assessment based on running each algorithm against random lists of different sizes.

There are many techniques for sortin efficiency. Implementation of particular sorting technique depends upon situation. Sorting techniques mainly depends on two parameters. First parameter is the execution time of program, which means time taken for execution of program. Second is the space, which means space taken by the program.

This paper presents a new algorithm called N* 3 matrix Sort that enhances the performance of sorting by reducing the number of swaps and the number of comparisons. The results from the implementation of the algorithm is compared with the Bubble sort and other recently used algorithms showed that the algorithm is performing better than the others in the worst case scenario.

ISSN 2348 – 7968

Proposed Algorithm

This paper proposes a new method for sorting arrange the elements into matrix representation forrmat, in order to provide a better performance than the other exisitng sorting algorithms. The proposed method has two independent phases: comparing columns and rows wise (aiming tosort column wise and row wise) and taking out minimum and maximum number from first and second row (aiming to sort the elements in row wise).

The proposed method algorithm is given below.

MATRIX Based Sorting Algorithm

Algorithm

Step 1: Get the n elements
Step 2: Arrange the elements by n * 3 matrix format representation
Step 3: Compare the elements by Column wise i.e., a[1,1] ,a[2,1] and a[3,1] to be compared, and if necessary swap it by the order smallest one be at the position a[1,1].
Step 4 Compare the elements by Row wise i.e., a[1,1] ,a[1,2] and a[1,3] to be compared, and if necessary swap it by the order smallest one be at the position a[1,1].
Step 5: After the comparison by row wise and column wise, we have to take the maximum number out for the first row in the matrix, and minimum number to be taken out from the second row.
Step 6: Now let us compare the maximum of first row and the minimum of the second row, if need to interchange in its places, and again compare it by row wise by placing the smallest to be in a[1,1] position. And continue for the second row and third row and so on until all the rows in the matrix get over.
Step 7: Repeat the steps 5 and 6 until all the elements cover in the matrix.
Step 8: Now rewrite the elements from the matrix from its positions and will give the data in sorted manner.

Fig.1. Matrix based algorithm

EXAMPLE ILLUSTRATION

Step 1: Get the n elements

24 34 56 32 67 98 09 23 11 10 05 12

Step 2: Arrange the elements by n * 3 matrix format representation

ISSN 2348 – 7968

24	34	56
32	67	98
09	23	11
10	05	12

Step 3: Compare the elements by Column wise i.e., a[1,1] ,a[2,1] and a[3,1] to be compared, and if necessary swap it by the order smallest one be at the position a[1,1].

09	05	11
10	23	12
24	34	56
32	67	98

Step 4: Compare the elements by Row wise i.e., a[1,1] ,a[1,2] and a[1,3] to be compared, and if necessary swap it by the order smallest one be at the position a[1,1].

05	09	11
10	12	23
24	34	56
32	67	98

Step 5: After the comparison by row wise and column wise, we have to take the maximum number out for the first row in the matrix, and minimum number to be taken out from the second row.

05	09	11
10	12	23
24	34	56
32	67	98

11

10 By step 6, if 11 > 10 then replace 11and 10 by its places, now we get

05	09	10
11	12	23
24	34	56
32	67	98

Then after swapping, again row wise check the elements.

23 if 23 > 24 , then replace 23 and 24 by its places,

24 otherwise no need to interchange.

ISSN 2348 – 7968

05	09	10
11	12	23
24	34	56
32	67	98

56 if 56 > 32 , then swap, or swapping is not required for 32

05	09	10
11	12	23
24	32	34
56	67	98

After swapping, again row wise compare the elements and if requires change it.

Step 8 : Now rewrite the elements from the matrix from its positions and will give the data in sorted manner.

Now , we get, the result from the matrix representation is

05 09 10 11 12 23 24 32 34 56 67 98

PERFORMANCE ANALYSIS

When compared the proposed algorithm with recently used as well as existing sorting algorithms like Bubble and Insertion sort, it has been found that the proposed method shows the better results. The following table shows the comparisons of the execution time with respect to sorting of different sorting algorithms with the proposed method.

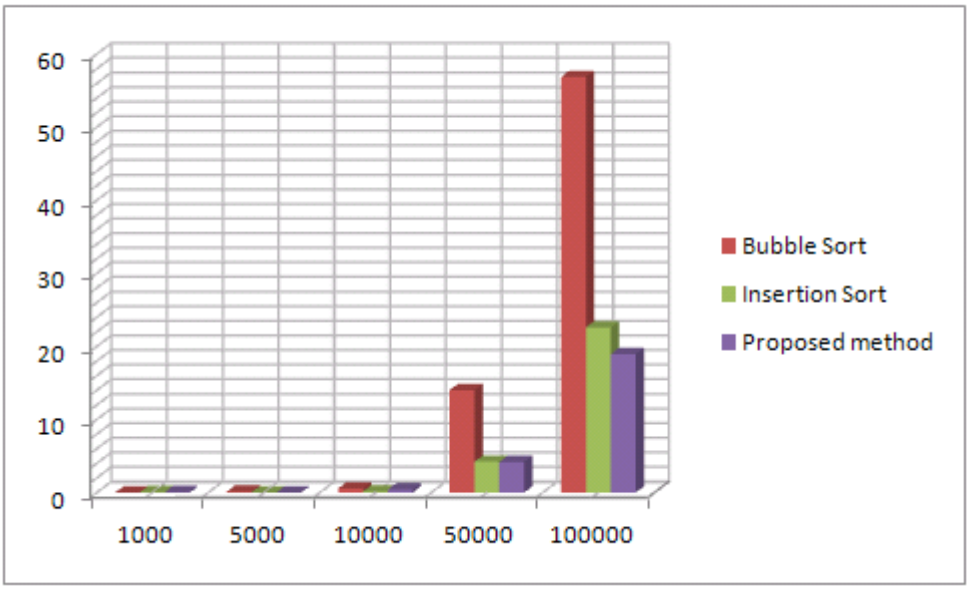
Table 1. Comparative study with proposed method

Data Size	Bubble Sort	Insertion Sort	Proposed method
1000	0.012	0.08	0.07
5000	0.145	0.045	0.016
10000	0.6	0.18	0.432
50000	14.05	4.25	4.19
100000	56.9	22.62	18.96

ISSN 2348 – 7968

RESULTS AND DISCUSSIONS

The results were derived based on the performance analysis of the above three sorting algorithms (Bubble Sort, Insertion Sort and Proposed Sort) were implemented in Java language and tested for the random sequence input of length 1000, 5000,10000, 50000 and 100000 on the average of 5 times. The proposed method is performing better performance than the exisitng sorting alogorithms. The following figure 2 shows the comparative performance analysis with the proposed method.



Fig, 2. Comparative performance analysis with proposed method

CONCLUSION

Bubble sort and Insertion sort has been described in comparative analysis. The difference between them helps in determining the best usage of the new proposed method. Furthermore, working on improve such sorting algorithm is needed in order to facilities how its applications work in more efficient way to reduce time & resources. For example, it helps in designing the memories.

The experimental result of the proposed algorithm has shown better efficient.Hence, the algorithm is recommended for all sizes of elementsto be sorted but much moreefficient as the elements to be sorted increases.

REFERENCES

[1] Thomas H. cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. (2001) Introduction to Algorithms. London: McGraw-Hill Book Company
International Journal of Computing Academic Research (IJCAR), Volume 3, Number 2, April 2014 57

ISSN 2348 – 7968

[2] Sharma, V. and Singh, S. (2008) Performance Study of Improved Heap Sort Algorithm and Other Sorting Algorithms on Different Platforms. IJCSNS International Journal of Computer Science and Network Security, VOL 8, No 4

[3] Muthusundari, S. Baboo, S. (2011) A Divide and Conquer Strategy for Improving The Efficiency and Probability Success In Sorting. International Conference on Advanced Computing, Communication and Networks.

[4] C. A. R. Hoare, "Quicksort," Computer Journal, 5, pp 10 – 15 1962.

[5] R.S. Scowen, "Algorithm 271: Quickersort," Comm. ACM 8, 11, pp 669-670, Nov. 1965..

[6] Herbert Schildt Tata McGraw-Hill [2005], “The Complete Reference C fourth Edition”.

[7] Alfred V., Aho J., Horroroft, Jeffrey D.U. (2002) Data Structures and Algorithms.

[8] Frank M.C. (2004) Data Abstraction and Problem Solving with C++. US: Pearson Education, Inc.

[9] Cormen T.H., Leiserson C.E., Rivest R.L. and Stein C. (2003) Introduction to Algorithms MIT Press, Cambridge, MA, 2nd edition.

[10] Seymour Lipschutz (2009) Data Structure with C, schaum Series, Tata McGraw-Hill Education.

[11] David Haws (2013) QUICKLEXSORT: AN EFFICIENT ALGORITHM FOR LEXICOGRAPHICALLY SORTING NESTED RESTRICTIONS OF A DATABASE, arXiv:1310.1649v1 [cs.DS] 6 Oct 2013.

[12] Li Xiao, Xiaodong Zhang, Stefan A. Kubricht, Improving Memory Performance of Sorting Algorithms, ACM Journal on Experimental Algorithmics, Vol. 5, 1-21, 2000.

[13] L. M. Busse, M. H. Chehreghani, J. M. Buhmann, The Information Content in Sorting Algorithms, IEEE International Symposium on Information Theory Proceedings (ISIT), pp. 2746-2750, 2012 [12]

[14] Srikanth Alaparthi, Sorting Binary Numbers in Hardware - A Novel Algorithm and its Implementation, 978-1-4244-3828-0/09/$25.00 ©2009 IEEE.

[15] K.E.Batcher, .Sorting networks and their applications,. in Proceedings of AFIPS Spring Joint Computer Conference, Apr 1968, pp. 307.314