International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

A Novel Sorting Algorithm for Data Analysis

T. Kesava, G. Mohan Ram
Assistant Professor, Department of CSE, Shri Vishnu Engineering College for Women (A), Vishnupur, Bhimavaram, West Godavari District, Andhra Pradesh, India

ABSTRACT

Sort algorithms are a basic research domain in computer science. In last few years, many researchers proposed several sorting algorithm to enhance time complexity and space complexity. In this paper we present a mapping sorting algorithm for heterogonous values, which utilizes mapping techniques. The mapping technique will be between values of array and indexes without using any Correlation and interchanging values. The paper analyzes time complexity of proposed algorithm mathematically and empirically. The complexity of proposed algorithm is O(xlogm) + O(n) where n represents number of sorted elements and m, x<<n. Result shows that the Performance of mapping sorting algorithm is better than other sorting algorithm.

Keywords : Complexity, Sorting Algorithm

I. INTRODUCTION

Sorting is the shifting list of items into ascending or descending order. Sorting is considered as an underlying assignment previously numerous procedures like searching, data management, and, database systems, so there is a need to enhance time complexity. This article proposes a sorting algorithm depend on mapping technique which maps values of the array that being sorted to a set of categorizes indexes.

The solution of sorting is a mapping relationship between index values and their corresponding ordered positions. A perfect sorting algorithm will make us accomplish our goal via just mapping, sorting the value of elements into the function and returning us their location. This article describes a new sorting algorithm which devotes to implement the mapping sort algorithm. Assuming the mapping relationship is linear, we devised two approaches. One depends on the maximum index and the minimum index value of records.

II. ANALYSIS OF THE ALGORITHM

A. Best Case:

Our proposed algorithm is performed in two ways. Map phase is for loop from i=0 to n-1, the basic function is Performing map equation on each value of the array and adding the result to the specified index position. In this step, the time complexity is T(n)=[i=,0,1,2,…,n-1] tends to be T(n)=O (n). In sort phase, the basic function is sorted indexes on each index position. If all position contains one index, the algorithm will not use another sorting algorithm and it will be the best case. The time complexity of the best case is T(n) =O(n) . This case exists when all elements of the array are well-distribute and they exist within range equal n (e.g. n=100 and range= [0,100]) and there is no repeated numbers.

B. Worst Case:

The worst case of the algorithm occurs when elements of the array condensed in very narrow range (E.g.. n=100 and range [0, 10]).In this case all elements in the array are repeated ,the algorithm uses merge sort to sort indexes on each slot at the

second loop. The complexity of the worst case is T(N)=O(n)+O(nlogn). But the partitioning by mapping sorting algorithm will reduce competition time comparing with using merge sort only.

C. Average Case:

The complexity of the average case is T(N)=O(n)+O(p*mlogm), tends to be T(N)=O(n)+O(xlogm), which p is the number of slots and m is the length of the slot and m <<n.

III. EXPERIMENTAL EVALUATION

The demonstrates execution correlation between other sorting algorithm and mapping sort algorithm with upgrade in the average case. The analysis of n varies from 100 to 100000 and the time measured by millisecond. Mapping algorithm has presentation close to other sort.

Existing sorting algorithm	Average time complexity	Best time complexity	Worst time complexity
Bubble	$O(N^2)$	O(N)	$O(N^2)$
Insertion	$O(N^2)$	O(N)	$O(N^2)$
Selection	$O(N^2)$	$O(N^2)$	$O(N^2)$
Merge	O(NlogN)	O(NlogN)	O(NlogN)
Heap	O(NlogN)	O(NlogN)	O(NlogN)
Quick	O(NlogN)	O(NlogN)	$O(N^2)$

Figure 1. Comparison of the existing algorithms

IV. PROPOSED ALGORITHM

Input: array A[i]
Output: Max [A_i], Min[A_i]
Procedure start(max, min)
range = Max[A_i]-Min[A_i]
for i=0
if i n-1
index = [((ai-Min(A_i))*n/range]+0;
append indexes to new array object
increase i by 1;
if end;
for i=0
if i n-1
if index_size=1
insert in final array A[];
if index_size<6
append value to final array A[];
if index_size>6
call merge operation;
append value to final array A[];
end

In proposed algorithm, we are taking array A[i] as input put, and Max [A_i], Min[A_i] as the output values. Initially, it will read values from array A[i]. Our proposed algorithm, it will analyze records in the array index and sort array index values max, min process support.

V. RESULT ANALYSIS

N	Quick sort algorithm	Mapping sort algorithm
100	3	3
1000	15	13
10000	95	97
100000	550	490

The performance comparison between Quick sort algorithm and Mapping sort algorithm with enhancement in the average case. The value of n varies from 100 to 100000 and the time measured by millisecond. Here, Mapping sorting algorithm has performance close to quick sort or better.

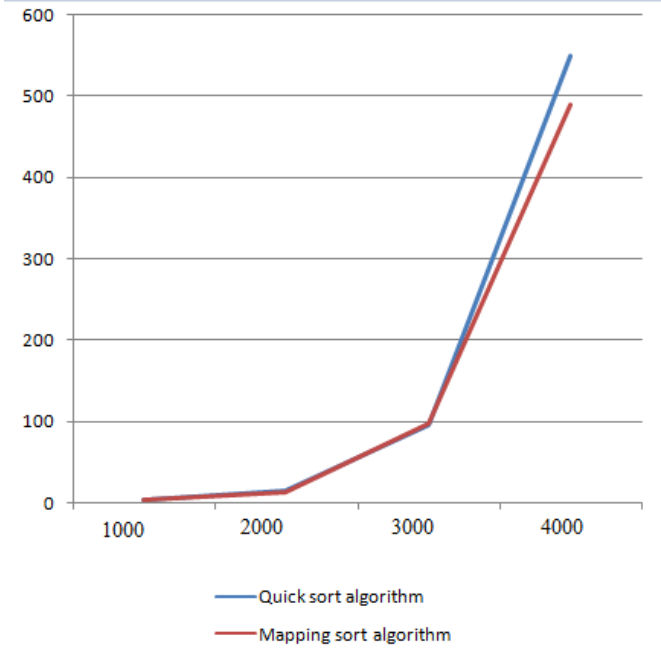

Figure 2. Result analysis between quick sort Algorithm and mapping sort Algorithm

VI. CONCLUSION AND FUTURE WORK

The paper presents mapping sorting algorithm and efficient data analysis for the algorithm. The algorithm mainly based on data mapping approach, it maps the fields of the array to their index positions. The algorithm has time complexity in average case is O(xlogm)+O(n) where n represents number of sorted fields and m, x<<n . The future work of the algorithm gives better results. In this paper, we are comparisons with other sorting algorithms showing the efficiency of the current mapping algorithm. The future scope will include improvement in the algorithm will used to decrees time and space complexity, comparing with pervious sorting algorithms.

VII. REFERENCES

[1]. D.E. Kunth, The Art of Computer Programming: Vol. 3, Sorting and Looking, second printing, Addison-Wesley, Reading, MA, 1975.

[2]. Karunanithi A., Drewes F., A Survey, Discussion and Comparison of Arranging Algorithms, Ume_a University, June 2014.

[3]. Deepak ., Setia S .,Arya M ., Sorting calculation , International Journal Of Inventive Rseearch In Technology, 2015.

[4]. Levitin A., Introduction To The Design And Analysis Of Algorithms, Villanova University,2012.

[5]. Kocher1 G., Agrawal2 N, Analysis and Review of Sorting Algorithms, IJSER, March 2014.

[6]. Puri T., Jain A., Sangwan A., Design and Analysis of Optimized Tallying Sort Algorithm, Department of Computer Science Swami Keshwanand Institute of Technology, Jaipur, INDIA

[7]. Sedgewick R., Wayne K., Algorithms, fourth Edition, June 18, 2015.

[8]. J.Alnihoud and R.Mansi, "An Enhancement of Major Sorting Calculations", The International Arab Journal of Information Innovation, Vol 7, No.1, January 2010.

[9]. Owen Astrachan, Bubble Sort: An Archeological Algorithmic Analysis,SIGCSE 2003. Accessible: http://www.cs.duke.edu/~ola/papers/bubble.pdf

[10]. Sedgewick, Algorithms in C++, pp.98-100, Addison-Wesley, 1992.

[11]. P.J. Deitel and H.M.Deitel, C++ How to Program, Sixth Edition, pp.931-932, PHI Learning Ptd, 2008, ISBN-978-81-203-3496-0.

[12]. Quick Sort HOARE, C. A.R. Quicksort. Comput. J. 5, I (1962), 10-15.

[13]. Cormen T., Leiserson C., Rivest R., and Stein C., Introduction to Calculations, McGraw Hill, 2001

[14]. http://algs4.cs.princeton.edu/addresses/23Quicksort.pdf ,2015