International Journal of Students Research in Technology & Management
Vol 1 (06), October – December 2013, ISSN 2321-2543, pg 624-626

RADIX SORT WITH REDUCED TIME COMPLEXITY

[1]Saumya Lahera, [2]Nirav Shah
Department of Computer Science, [1]Shah & Anchor Kutchhi Engineering College,
[2]Rajiv Gandhi Institute of Technology, Mumbai, India
saumyanyca@yahoo.com , nrvshah10@gmail.com

Abstract
Sorting is a technique that arranges data in a specific order, so that they can be accessed easily. Radix sort is one of the techniques. Processors execute programs as per the definition of codes. Time of execution depends on the design of the program. Complex programs take more time of execution. This technical paper explains radix sort algorithm that has been made simpler to reduce time complexity.

Keywords: Reduced Time Complexity, Optimization, Main Logic, Input Phase, Radix logic

1. INTRODUCTION

Radix sort is a very sophisticated technique used for arranging integers in an appropriate order. Here, data is in the form of array and each element of the array is processed. And sorting depends on the digits of the largest number in the array. Passes are calculated from least significant digit to the most significant digit of the largest element of the array. There are many algorithms available for this sorting but complexity is also very important factor. This paper explains the refurbished algorithm, which reduces the complexity.

Now focus of this algorithm is to reduce complexity. And it is achieved by eliminating unwanted iterations and variables. Reducing iterations can be helpful in reducing the complexity. Eliminating variables can save memory. These things can avoid abdication of the resources and can also boost efficiency factor.

2. IMPLEMENTATION

This algorithm is divided into many parts and these parts are linked with each other. Initial part of any program is to get the input. Scanning part is responsible for scanning the elements and they are packed in one single array. And then elements are arranged as per the radix sort logic. In this algorithm only two loop, eight integers, one float and main logic has used four loops. And other main parts are -

A. Get Maximum Element

Maximum element is calculated to know the digits. Digits are used to indicate number of passes. So this part gets the maximum element of the array. Digits of maximum elements are used by the other part. Other part that is responsible for calculating passes is given the maximum element.

While input process is taking place, maximum element is extracted of the array. By default one variable is initialized to zero and first element of the array is checked with it, whether it is greater or not and if that element is greater than the first element than that element will be replaced by the first element. And now that variable becomes the maximum element. Similarly the variable is checked with all the elements and it will store the maximum element. Because if the variable is greater than the array element then it will be replaced by the array element or else it wont be altered.

B. Calculate Passes

This part is responsible for calculating the number of passes. Passes indicate number of digits of the maximum elements and it becomes the major part of main logic. Iterations are decided on the basis of passes. Buckets are from zero to nine and they are the main logic in every pass.
One specific variable is declared for storing the maximum value. Now from this variable value passes are calculated. Passes are calculated by calculating number of digits of the variable that stores the maximum value. So another variable is declared to store passes. Now dividing maximum element until it is less than zero can help to get the result. Loop is set and is made to do the work until the maximum element is made less than zero and while doing this passes are calculated by calculating the number of iterations and the storing it in the variable responsible for storing passes.

C. Main Logic

In this part main logic is considered and main sorting is done. Elements are arranged as per the basic radix sort rules. This part has to follow the radix sort protocols. Elements of the array are sorted from least significant digit to most significant digit. Single element is extracted and then as per the digit they are assigned to the appropriate bucket. Each pass has ten buckets. And elements are arranged in that form. And final pass is responsible for the expected answer.

Main loop repeats till the number of passes. Now one variable is multiplied by ten in every repetition of the first loop for dividing the elements for getting specific digit of the element. Another iteration is nested to scan each and every element of the array and then placed in another temporary array. Now these elements are sorted using bubble sort technique and with these array also the main array is sorted.

Second array and the main array are sorted simultaneously and then the output varies as per the iteration. After the main loop is broken then the processing is done.

3. OPTIMIZATION

A. *Input Phase*

In this phase with scanning elements maximum element is also calculated. Now there is no need to implement any other code for calculating maximum element.

And using maximum element passes are also calculated so this can be very useful. So only maximum element is to be supplied to the pass calculator and then rest is done. After passes are calculated then the control is passed to the next part that is main logic part. Figure 1 shows Input Phase.

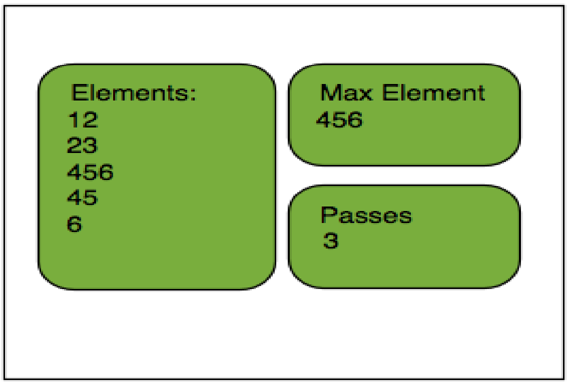

Figure 1. Input Phase

B. *Radix Logic*

Reducing loops and making it to four optimize this Logic. Increasing iterations can increase the execution time and makes the code complicated. Now over here loops are reduced and they have been put down to the saturation point. Figure 2 shows the working. It shows the arrangements of the loops and what are the uses. This logic is the main logic of the program.

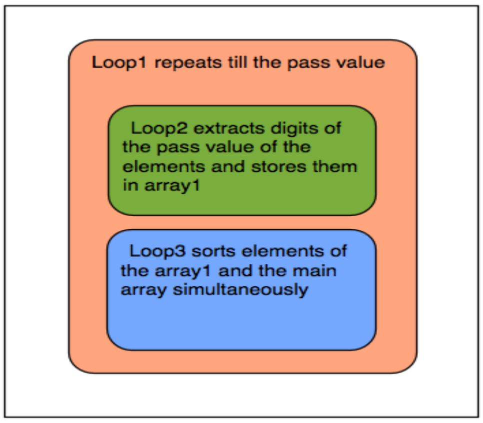
Loop1 repeats till the pass value
Loop2 extracts digits of the pass value of the elements and stores them in array1
Loop3 sorts elements of the array1 and the main array simultaneously

Figure 2. Radix Logic

4. EXPERIMENTAL RESULTS

Table 1 shows the results of the algorithm. It shows previous algorithm's results and also the results of the new algorithm.

TABLE 1: COMPARISON OF THE PREVIOUS AND THE NEW ALGORITHMS.

Average time taken by algorithms for various elements		
Elements	**Average time of the old Radix Sort Algorithm (ms)**	**Average time of the new Radix Sort Algorithm (ms)**
500	90	13
1000	98	33
2000	122	80

5. CONCLUSIONS

We explained the idea of radix sort and then explained the importance of an efficient code. We gave basic idea of the algorithm and then we mentioned the technique used for achieving the result.

Our conclusions are that when programs are made simpler then they help processors to work easily. So we have reduced the complexity by designing an efficient program.

REFERENCES

1. http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/radixSort/radix.html
2. http://www.stackoverflow.com/
3. http://en.wikipedia.org/wiki/Radix_sort
4. http://www.cprogramming.com/tutorial/computersciencetheory/radix.html
5. http://www.cs.auckland.ac.nz/software/AlgAnim/radixsort.html
6. http://c2.com/cgi/wiki?RadixSort