Review of Radix Sort Algorithm

Abstract—***Radix sort is a linear sorting algorithm that is used to sort large numbers of integers or strings efficiently and it is an efficent sorting algorithm among many other sorting algorithms. It works by sorting the digits of numbers with counting sort from starting to sort the rightmost digit and then continue to sorting each digit until there aren't any digits left to sort. Radix sort has a time complexity of O(kn) where k is the number of digits and n is the number of elements to be sorted.***

I. INTRODUCTION

Sorting algorithms truly takes an important place in computer science since organizing data is important and they provide an effective way to do that. As it's name suggests, "Radix Sort" is a sorting algorithm among many other sorting algorithms. The word "radix" denotes to the base or the amount of digits which are used to symbolize numbers. Elements are sorted by being processed digit by digit multiple times until the biggest digit is reached and there aren't any bigger digits left to look at. This algorithm is non-comparison-based since elements aren't directly compared but rather their digits are compared. Even though, it can use different algorithms such as "Bucket Sort" or "Quick Sort", radix sort mostly uses "Counting Sort" algorithm. Radix sort is an algorithm which is efficient when large numbers of integers or strings needs to be sorted and mostly applied at situations which the elements to be sorted have a fixed length and a small range of possible values.

II. EXPLANATION OF THE ALGORİTHM

Before starting to the explanation, it is important to note that, other sorting methods can be used at radix sort as well, but since counting sort is the most commonly used algorithm with radix sort, in this explanation counting sort will be used

Numbers that needs to be sorted are stored in an array

Radix sort, starts from the least significant digit (rightmost digit) to sort and it is repeated until there aren't any digits left. At each iteration, radix sort, sorts the current digits of the numbers with using counting sort

If a number doesn't have any number at the current digit, then it is assumed that the number is going to have the number 0 as it's current digit

(Step 1 and 2 is performed to find that "how many times counting sort will applied to digits to sort the numbers?")

1. Find the biggest element of the unsorted dataset, array (maximum element)

To achieve this, array which contains the dataset is searched from beginning to end.

2. Find how many digits the maximum element has

To learn that, that number will be divided to ten until the number is equal to zero

3. Apply counting sort algorithm and continue until there aren't any number left which has a number at the current digits

For example, let's have these dataset of numbers: 173, 672, 71, 9, 143

Before sorting	1st digits sorted	2nd digits sorted	3rd digits sorted
173	071	009	009
672	672	143	071
071	173	071	143
009	143	672	173
143	009	173	672

.

III. PSEUDOCODE

Following code is the pseudocode for the radix sort algorithm:

```
RadixSort(arr[], n):
  // Step 1) Finding the maximum element
  max=arr[0]
```

```
  For (i=1 to n-1):
     If (arr[i]>max):
        max=arr[i]

  // Step 2) Finding how many times will counting sort be applied
  For (k=max to k>0):
k=k/10

  // Calling countingSort for
  // k times using For loop.
  For (div=1 to div<=k):
     countingSort(a, n, div)
     div=div*10
```

IV. ALGORITHMIC COMPLEXITY

Best Case: Ω(nk). This occurs when the data is already sorted.

Worst Case: O(nk). This occurs when the data is completely unsorted

Average Case: Θ(nk). This occurs when the data is half unsorted

All cases have the same complexity in other words has a stable time complexity because radix sort performs the same number of operations regardless of the initial order of the elements.

V. LITERATURE REVIEW

Radix sort has been studied in the literature, and has been shown to be an efficient sorting algorithm for large lists of integers or strings.

[1] In a comparison of sorting algorithms by Shukla and Saxena (2012), while for some conditions radix sort was found to be producing the same results, for some other conditions radix sort was found to be better. Also [2] an experiment with radix sort and other sorting algorithms conducted by S. Thiel, L. Thiel and G. Butler showed that radix sort performs better than it's competitors

But it also should be noted that, the studies at literature tells that "there are no algorithm which is always the best"

Also, Radix sort has usage at a lot of practical applications. [3] One example to this is searching large databases (Baeza-Yates and Navarro 1999)

VI. CONCLUSION

Radix sort is an efficient algorithm which has a linear complexity. It's complexity is O(nk) which k is the digit count of the biggest number and n is the size of the array. It is an algorithm which has a lot of advantages over the other sorting algorithms and can be beneficial in various scenarios

REFERENCES

[1] https://www.ijera.com/papers/Vol2_issue5/CR25555560.pdf
A. Shukla, A. K. Saxena," Review of Radix Sort & Proposed Modified Radix Sort for Heterogeneous Data Set in Distributed Computing Environment", International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 5, September- October 2012, pp.555-560

[2] https://users.encs.concordia.ca/~sthiel/DS/SEA2015_FastRadix.pdf
S. Thiel, L. Thiel, G. Butler, "Relaxing the Counting Requirement for Least Significant Digit Radix Sorts", Concordia University, 1455 De Maisonneuve Blvd. W., Montreal, Quebec, Canada H3G 1M8.

[3] https://users.dcc.uchile.cl/~gnavarro/ps/mds00.pdf
R. B. Yates, A. Moffat, G. Navarro, "Searching large text collections"