**ANALYSIS OF RADIX SORT ALGORITHMS**

INFO6205 Program Structures and Algorithms  
Course Professor: Robin Hillyard

Abhishek Ravindra Satbhai 001587689  
  
Apoorva Kethamaranahalli 001090928  
  
Sneha Ravichandran 001096251

**Table of Contents**

ABSTRACT iii  
  
INTRODUCTION iii  
  
ANALYSIS iv-v  
  
REFERENCES vi

ABSTRACT

This article is an examination of the radix sort methods: MSD Radix sort, MSD Radix Exchange sort and LSD Radix sort. From the time Radix sort has been developed by Harold H. Seward in 1954 at MIT, there have been many developments along the way on Radix sorts. Currently, radix sorts are most applied to a varied collections of strings as well as integers. This article examines three such radix sorts, and gains to reach an understanding through literature surveys done on technical papers written on these sorts in particular

INTRODUCTION

Radix sort is a type of a sorting algorithm that is non-comparative. Radix sort is a digit-by-digit sorting method that starts with the least significant digit and progresses to the most significant digit, or vice-versa. By producing and distributing items into buckets based on their radix, it eliminates the comparison. If an element has more than one significant digit, the bucketing process is repeated for each digit, keeping the previous step's ordering, until all digits have been considered. As a result, radix sort is also known as bucket sort or digital sort.

As early as 1923, radix sorting algorithms were widely used to sort punched cards. From the time Harold H. Seward developed a memory efficient way for Radix Sort, which used to identify the bucket sizes beforehand by doing a linear scan, like Harold’s counting sort. Radix sorts are most typically used today to sort collections of binary texts and numbers.

Radix sorts can be done in two ways, by beginning with the most significant digit (MSD) or the least significant digit (LSD). As MSD radix sorts employ lexicographic order, which works efficiently for sorting strings.

ANALYSIS

Analysis of “Formulation and analysis of in-place MSD radix sort algorithms” by Nasir Al-Darwish

The paper analyses a set of in place MSD radix sort sorting algorithms with varying radices. The set of algorithms employ an in-place partitioning an upgrade from the usual linked list version of radix sort, which takes O(n) space. The collective worst case running time for this type of in-place partitioning and efficient bit processing employed by the author, would be O(kn), where k is the number of bits required to encode an element value and n is the number of elements to be sorted.

These algorithms employ in-place partitioning, which is a significant improvement over the usual linked list version of radix sort, which takes up O(n) space. Of these set, one algorithm to be noted (binary) is a rewrite of the standard radix-exchange sort that emphasizes in-place partitioning and efficient bit processing operations. The worst-case order of running time for this technique is O(kn), where k is the number of bits required to encode an element value and n is the number of elements to be sorted. For the purpose of determining sorted order, shorter keys must be as long as the longest key.  
  
Along the course of the paper, we are introduced to other optimizations and evolution of the binary algorithm to handle other inputs like floating point numbers. The paper summarizes the binary algorithm to be running in comparison with Quicksort in terms of benchmarking results but provides an improvement over the usual Quicksort algorithm which tends to become slower when there is repetitive data with time complexity becoming O(n2).

The paper also reviews three partitioning methods: sequential, divide-and-conquer, and permutation-loop. For sorting English Experiments benchmarking results conclude that the generic radix sort with divide and conquer partitioning proves to be the most efficient. Also, the divide-and-conquer strategy can be improved to take use of data redundancy.

Analysis of “Radix Exchange--An Internal Sorting Method for Digital Computers”

by Paul HildeBrandt and Harold Isbitz

The paper discusses an approach for sorting data within an electronic binary digital computer's high-speed memory called Radix Exchange. The Radix Exchange method is meant for a fixed word length, internally binary machine. This restriction to binary makes the Radix Exchange feasible.

Any Radix sort method includes sorting objects into R "pockets" or areas based on the value of each digit in the key, expressed as a number to the base R using a positional representation. Unless each digit's location is known in advance, R must be large to take into account all the terms, to avoid overflows.

The sort method detailed in the paper can be summarized in three steps:   
a) A sort run on any bit

b) The bookkeeping after running a sort (not at the least significant bit)

c) The bookkeeping after running a sort on the least significant bit

After running for a number of randomly generated numbers, and sorting by two methods: insertion as well as radix, and supplying the randomly generated numbers in reverse, reshuffled order and sorted order to the algorithms respectively and running benchmarks, it was concluded that Radix Exchange works better than Inserting for any randomly generated input by a factor of 17

The paper draws conclusions from the experiments run to state that despite the fact that the Radix Exchange method was built for a certain computer logic which makes the experiments’ results biased against Inserting, Radix Exchange works best when there is a space priority. Also, when there is ample room in memory, Radix Exchange can be assumed to work faster than the other possible alternatives.

REFERENCES

1. Formulation and analysis of in-place MSD radix sort algorithms by Nasir Al-Darwish
2. Enhancing The Efficiency of Radix Sort by using Clustering Mechanism by Manju De
3. <https://en.wikipedia.org/wiki/Radix_sort>
4. <https://www.geeksforgeeks.org/radix-sort/>