Discussion About Radix Sort Algorithms

Jiao Gong

Dou Jin

Northeastern University

INFO6205 Data Structure and Algorithms

Instructor*: Robin Hillyard*

Date*: 12/05/2021*

# 

**Contents**

[Abstract iii](#_Toc89329976)

[Introduction iv](#_Toc89329977)

[Discussions iv](#_Toc89329978)

[References vi](#_Toc89329979)

# Abstract

Sorting algorithms are widely used in new word discovery, term extraction, meaningful string detection and other fields. String sorting has become a key technology that affects the speed of repetition pattern extraction in large scale corpora.

In this paper, we will study several types of radix sort algorithms, which include MSD Radix Sort, MSD Exchange Radix Sort, LSD Radix Sort, and In-place MSD Radix Sort. The last radix sort algorithm would be an improvement for the space complexity for the MSD radix sort.

And finally give conclusions and the direction of further study.

*Keywords:* Chinese string; MSD radix sort; MSD exchange radix sort; time complexity

# Introduction

String sorting is an important basic technology in natural language processing, and the sorting speed has a great influence on correlation processing. In general, sorting algorithms can be divided into two categories: comparison-based and non-comparison-based. Quicksort is a good example for the comparison-based sorting algorithm, which has a O(n log n) time complexity in the average case. But radix sort, which we will talk about in this paper is a representative of non-comparison-based sorting algorithm.

Radix sort algorithms have been divided into two major types: MSD (most significant digit) and LSD (least significant digit). LSD and MSD are both string array sorting algorithms, based on the so-called key indexed counting rather than on comparisons. Therefore, LSD and MSD have a different running time versus traditional quick sort or merge sort.

# Discussions

* 1. **MSD Radix Sort**

MSD radix sort also known as most significant digit radix sort. It starts sorting from the beginning of strings. The idea is to perform the following steps for each digit in an array. First, storing elements in different buckets according to their ith digit. Then, recursively sort each bucket that has more than one element. The best-case time complexity is O(N) and the Worst-Case time complexity is O(n\*m) where m is the average length of strings. As for the space complexity, it has O(n+ m\*B), where m is length of the longest string and B is the size of radix. MSD needs only to scan distinguishing prefixes, while all digits are scanned in LSD.

**1.2 LSD Radix Sort**

LSD Radix sorting is stable, and is different from MSD Radix sorting, because the relative order is retained after each sorting iteration. It is a branch-free algorithm. When handle with relatively short, fixed length keys, LSD radix sort algorithm has a better performance than MSD radix sort algorithm. Therefore, it’s nice to use LSD radix sort algorithm when sort key-value pairs where the key is an integer or a string, and you want to keep the original relative sequence. The time complexity is O(mn), and the space complexity is O(n + r), where r is the number of digits that represent a single position in a number.

**1.3 MSD Radix Exchange Sort**

Radix Exchange sort was first suggested for binary-alphabet but can be used with strings if bit-extraction and testing are done as low-level machine operations. The basic idea of radix exchange sort is to split in-place the data into two groups based on the most significant bit. To quote: ‘This is done using two oppositely moving pointers; the left (right) pointer skips elements having 0-bit (1-bit); otherwise, it exchanges the elements pointed to by left and right pointers. Then the process is applied recursively to each group considering the next bit’ [2].

* 1. **In-place MSD Radix Sort**

Typically, the MSD radix sort need a O(N+WR) space, where N is the length of array, and W is the length of string, and R is the size of radix. But with an improvement by using in-place partitioning, the space could be improved to O(k) and has O(kN) worst-case order of running time, where k is the number of bits needed to encode an element and N is the number of elements to be sorted. There are three partitioning methods to implement the in-place radix sort: divide-and-conquer, permutation loop, and sequential. With the experiments in the previous articles, we can put forward a conclusion that the general radix sort using divide-and-conquer partitioning has the best performance when sorting strings.

# References

1. H. Zhang and S. Shi. An Efficient Algorithm of Chinese String Sort in User-Defined Sequence. 2013 International Conference on Asian Language Processing, 2013, pp. 253-256.
2. Nasir Al-Darwish. Formulation and analysis of in-place MSD radix sort algorithms. Journal of Information Science 2005; 31; 467
3. Paul Hildebrandt, Harold Isbitz. Radix Exchange--An Internal Sorting Method for Digital Computers. [Journal of the ACM](https://dl.acm.org/toc/jacm/1959/6/2), [Volume 6](https://dl.acm.org/toc/jacm/1959/6/2), [Issue 2](https://dl.acm.org/toc/jacm/1959/6/2), April 1959, pp 156–163
4. Zhang Haijun，Ding Xiyuan，Zhu Chaoyong. Improved sort method for Chinese strings. Computer Engineering and Applications, 2010，46（19）：129-131