Comparing Performance of Timsort, Dual-pivot Quicksort, Huskysort, Radix Sort on Sorting Chinese Names

Li Baiyu, Xinhui Zhao

Graduates, Northeastern University, College of Engineering

1st December 2021

Abstract

We sort Chinese using five methods: Timsort, Dual-pivot Quicksort, Huskysort, LSD radix sort and MSD radix sort and then compare their Benchmark results when the input is 1M names. And we describe two papers ——"Formulation and analysis of in-place MSD radix sort algorithms" and "Comparative of Advanced Sorting Algorithms (Quick Sort, Heap Sort, Merge Sort, Intro Sort, Radix Sort) Based on Time and Memory Usaged".

Keywords: Timsort, Dual-pivot Quicksort, Huskysort, LSD radix sort and MSD radix sort

1. Introductions about the sorting methods

a. Timsort

TimSort is a sorting algorithm based on Insertion Sort and Merge Sort, it is a stable sorting algorithm works in O(nLogn) time.

First, we divide the Array into small blocks known as "Run". We sort those "Runs" using insertion sort one by one and then using the combine function of merge sort to merge those "Runs", because insertion sort performs better for small arrays than merge sort. If the original "Run" is smaller than a minimum length, the run will be extended by insertion sort until reaching the condition. After that, merge sort is used to merge multiple runs. The size of the run may vary from 32 to 64 depending on the length of the array, and we use 32 this time.

b. Dual-pivot Quicksort

The idea of dual pivot quick sort is to take two pivots, one in the left end of the array and the second, in the right end of the array. The left pivot must be less than or equal to the right pivot, or we will swap them.

Then, we begin dividing the whole array into three parts: in the first part, all elements will be less than the left pivot, in the second part all elements will be greater or equal to the left pivot and also will be less than or equal to the right pivot, and in the third part all elements will be greater than the right pivot. Then, we shift the two pivots to their appropriate positions, and after that for these three parts, we use Quicksort recursively.

c. PureHuskySort

HuskySort is a sorting algorithm which use inexpensive comparisons to replace expensive comparisons in the linearithmic phase to as much as possible. It could indirectly reduce processing time because of reducing the total times of accessing arrays by moving work from the linearithmic phase to the linear phase.

In Java, the two system sorts are dual-pivot quicksort (for primitives) and Timsort for objects. Kandappareddigari Sai Vineeth, Liaozheng yunlu (Authors of Huskysort) demonstrate that a combination of these two algorithms can run significantly faster than either algorithm alone for the types of objects which are expensive to compare.

We use PureHuskySort this time, which set the boolean parameter "useInsertionSort" as false.

d. Radix Sort

Radix sort is an extension of Bucket sort. It doesn't need to compare between elements directly, or need to swap elements until the correct order, all we have to do is "classification".

First, provide some ordered containers, which number depends on how many different characters there are. And then cut whole strings into single characters and for each digit put strings with the same characters into one container. Finally, take strings out of containers basing on first-in, first-out principle.

Radix sort has two ways, depending on the order in which digits are compared.

---LSD radix sort
LSD Radix Sort starts sorting from the end of strings (the Least significant digit). So we need to unify all the strings into the same digit length, and zeros are added in front of the shorter digit

---MSD radix sort
MSD Radix Sort starts sorting from the beginning of strings (the Most significant digit). Sometimes we do not need to compare all the digits to get the final correct order.

2. Sorting Chinese with Pinyin

We use "Locale.CHINA" to get the collator instance, and compare two string by function "collator.compare".

The Collator class performs locale-sensitive String comparison, it is used to build searching and sorting routines for natural language text. Collator is an abstract base class. Subclasses implement specific collation strategies.

Like other locale-sensitive classes, we use the static factory method, getInstance, to obtain the appropriate Collator object for a given locale. We only need to look at the subclasses of Collator if need to understand the details of a particular collation strategy or if need to modify that strategy.

3. Benchmark and Results

Benchmark is a way to time algorithms with different inputs and see how they do. It can truly reflect the performances of sorting methods.

Below is the results of sorting 250k, 500k, 1M, 2M, 4M Chinese names by the five sorting methods. Each result is the average of running ten times.

	250k List	500k List	1M List	2M List	4M List
Timsort	6263.405	13903.19	28391	61451.11	131328.1
Dual-Pivot Quicksort	626.9238	1287.565	2587.6	5412.284	11531.97
Pure Huskysort	4497	9746.592	21609.89	49775.69	104680.2
LSD Radix Sort	1233.959	2484.564	4878.797	11160.95	20463.41
MSD Radix Sort	1584.228	3136.351	7060.084	14830.03	25525.19

4. Work of two related technical papers

a. Formulation and analysis of in-place MSD radix sort algorithms [1]

Nasir Al-Darwish presented and analyzed a number of in-place MSD radix sort algorithms, collectively referred to as ‘Matesort’ algorithms. These algorithms are evolved from the classical radix exchange sort, they use the idea of in-place partitioning which is a considerable improvement over the traditional linked list implementation of radix sort that uses O(n) space.

The binary Matesort algorithm, one of the recasts, emphasized the role of in-place partitioning and efficient implementation of bit processing operations. Experiments have shown that it has comparable speed with Quicksort for random uniformly distributed integer data. Unlike Quicksort, which becomes slower as data redundancy increases and may degenerate into O(n2), binary Matesort algorithm is unaffected and remains O(kn), where k is the element size in bits.

The binary Matesort algorithm is evolved into several "general radix Matesort" algorithms. He presented formulation and analysis for three of them(sequential, divide-and-conquer and permutation-loop). For English text, experiments have shown that the general radix Matesort using divide-and-conquer partitioning is the fastest. Moreover, the divide-and-conquer method can be optimized further to exploit data redundancy. Finally, the experimental results show that Matesort (and also Quicksort) are much faster (twice as fast in many instances) than the built-in Microsoft.

b. Comparative of Advanced Sorting Algorithms (Quick Sort, Heap Sort, Merge Sort, Intro Sort, Radix Sort) Based on Time and Memory Usage [2]

Marcellino Marcellino, Davin William Pratama, Steven Santoso Suntiarko, Kristien Margi thought that every algorithm has its own best-case as well as its worst-case scenario, so it is difficult to determine the best sort algorithm just by its Big-O. Besides, the amount of memory required also affect the algorithm's efficiency.

They used a software development method called Waterfall and chose Python as programming language to conduct a research for five advanced sort algorithms: Radix Sort, Heap Sort, Quick Sort, Merge Sort, and Introspective Sort. Their data source contains 11,000 books information such as title, author, ISBN, and others.

First, he built activity diagrams for each sort algorithm, then implemented that on a little dataset which contains 9 sample data, basing on the time and memory the little dataset needs to calculate the estimated time and memory for more datasets (contain 2500, 5000, 7500, and 11000 sample data). After doing that, these sort algorithms were directly used to sort datasets just mentioned. For each dataset and each algorithm, the test was run for up to five times as recorded.

After comparing each algorithm both on time required and on memory usage, this research show that Introspective sort is the best at time and Heap sort is the best at memory usage.

Reference

[1] Nasir Al-Darwish. Formulation and analysis of in-place MSD radix sort algorithms. Journal of Information Science, 31 (6) 2005, pp. 467–481

[2] Marcellino Marcellino, Davin William Pratama, Steven Santoso Suntiarko, Kristien Margi. Comparative of Advanced Sorting Algorithms (Quick Sort, Heap Sort, Merge Sort, Intro Sort, Radix Sort) Based on Time and Memory Usage. *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)* 28 October 2021