**Final project**

Sorting Algorithms in Chinese Characters



**Team Members:**

001582498 Xingyu Huang

001528318 Zhen Luo

001048178 Haoyuan Qin

1. Project Description

Unlike English letters which encoded by ASCII, Chinese characters normally using Unicode to encode. Hence, how to sort Chinese characters in order, became a real problem we may face in the real world. So, we borrowed the idea of ‘Pinyin’, which is a kind of phonetic symbol to pronounce Chinese characters, to sort Chinese characters in order.

1.1 Problems

However, one Chinese character may have different pronunciations and one pronunciation may correspond to different Chinese characters. In order to deal with these situations, we import a library called pinyin4 which can generate pinyin from imported Chinese characters String even though there are some English letters in the String.

* 1. Sort-Algorithms

We used MSD Radix-sort, LSD Radix-sort, Dual-pivot Quicksort, Timsort and Husky Sort to sort Chinese characters. These algorithms may have different performance in sorting Chinese characters. So our goal is not only sort Chinese characters with different algorithms but also to find the best algorithms in doing such thins.

1. Project Design

This project is made up by different classes, methods and libraries. In order to sort Chinese characters, we build an Object called “MingZi” which includes “HanZi” (Store Chinese characters), “PinYin” (Store pinyin) and “longest” (Store the length of the longest String).

Before we start to sort our Strings, the TXT.java can help us to read strings from txt file and convert it to “MingZi”. So now we can sort our strings by sort the “PinYin” and then give out the result of sorting.

As we implement different sorting algorithms, the performance of each algorithm is quite different with each other and the correctness of each algorithm in this case is still a myth. So, we implement test cases and benchmark to see these results and differences.

Furthermore, in order to increase the performance of tests and benchmark to prevent over time error. Multi-threads are used in our project which can largely increase the performance and give us the best result.

1. Complexity Evaluate

Before we jump into the tests and benchmark. We’d like to analyze each sorting algorithm.

For timsort, the average time consumption is NlgN, and for quicksort it’s 2NlgN. However， the quicksort’s time consuming will be decreased if quicksort has more pivots and the best is quicksort with median of three pivots.[1] Theoretically, as number of elements increase, radix sort did not perform so well.[3] For radix sort, there are also some advantages, Fast Radix performs better than its competitors in all but two situations. First, when input sizes are very small or distributions are very narrow. Second, when inputs are large and distributed fairly uniformly across a wide range. [4]

1. Conclusion
2. Results