# PROGRAM STRUCTURES & ALGORITHMSFALL 2021 FINAL PROJECT

A Dissertation Presented

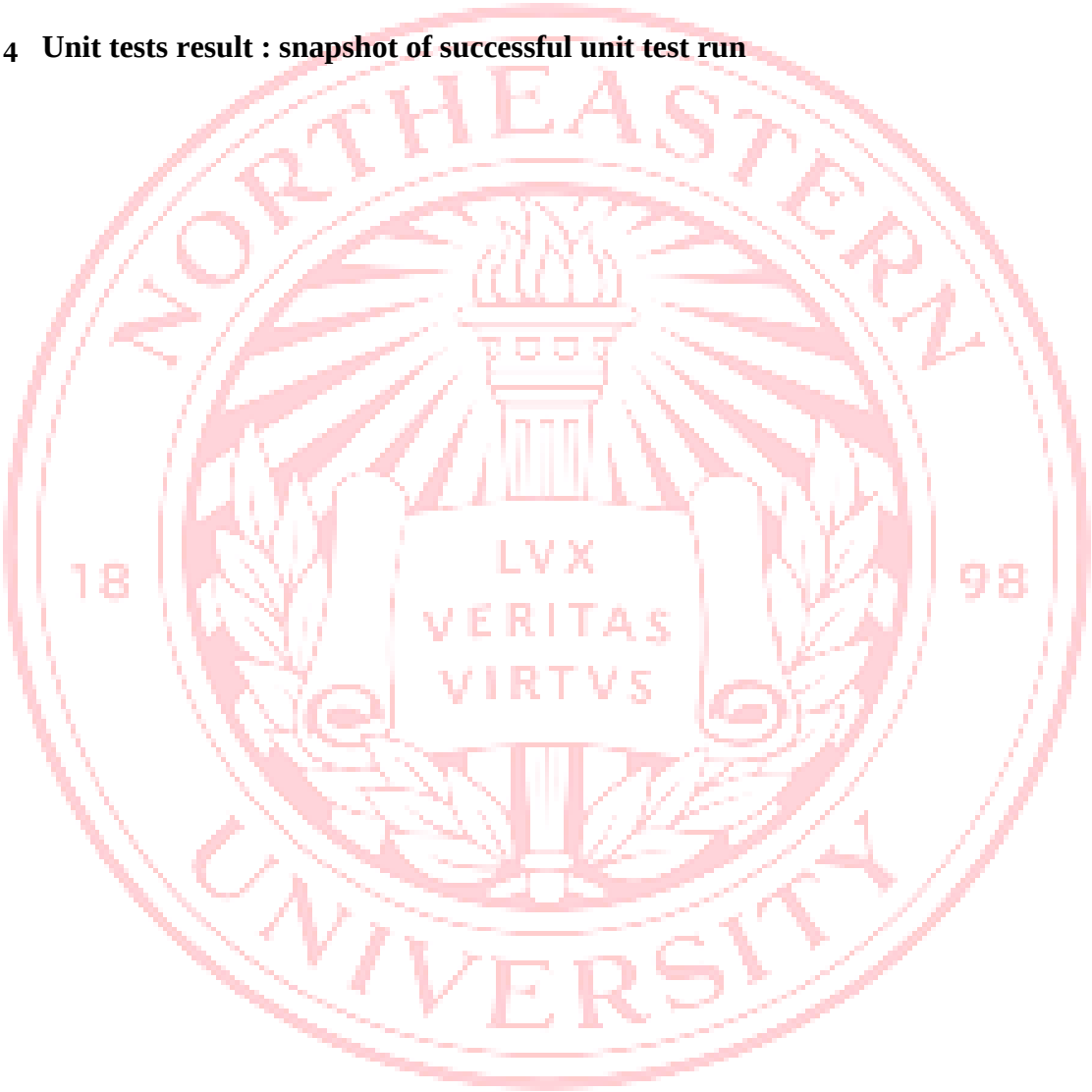
By

Yanrong Huang 001563514
Yuan Lian 001063563
Yuxin Liu 001028722

December 2021

## Contents





- **Task**

Implement MSD radix sort for a natural language which uses Unicode characters.

You may choose your own language or (Simplified) Chinese. Additionally, you will complete a literature survey of relevant papers and you will compare your method with Timsort, Dual-pivot Quicksort, Huskysort, and LSD radix sort.

- **How we realize**

1. Trans Chinese to Pinyin

Chinese characters can be stored as a unicode string or as a char. In order to sort Chinese characters in pronunciation order, all characters must be transferred to sortable Pinyins. Therefore, we implement five functions: 1) unicodeToChar, 2) charToPinyinNOTone, 3) charToPinyinWITone, 4) unicodeToPinyinNOTone, and 5) unicodeToPinyinWITone.

2.Implement Different Sorting Algorithm

We used Prof Robin's Timsort, pureHusky sort ,LSD radix sort and we rewrite the MSD radix sort and Dual-pivot Quicksort. The MSD radix sort comes from geeksforgeeks and the Dual-pivot Quicksort is from Algorism 4th.

3.Benchmark Timer

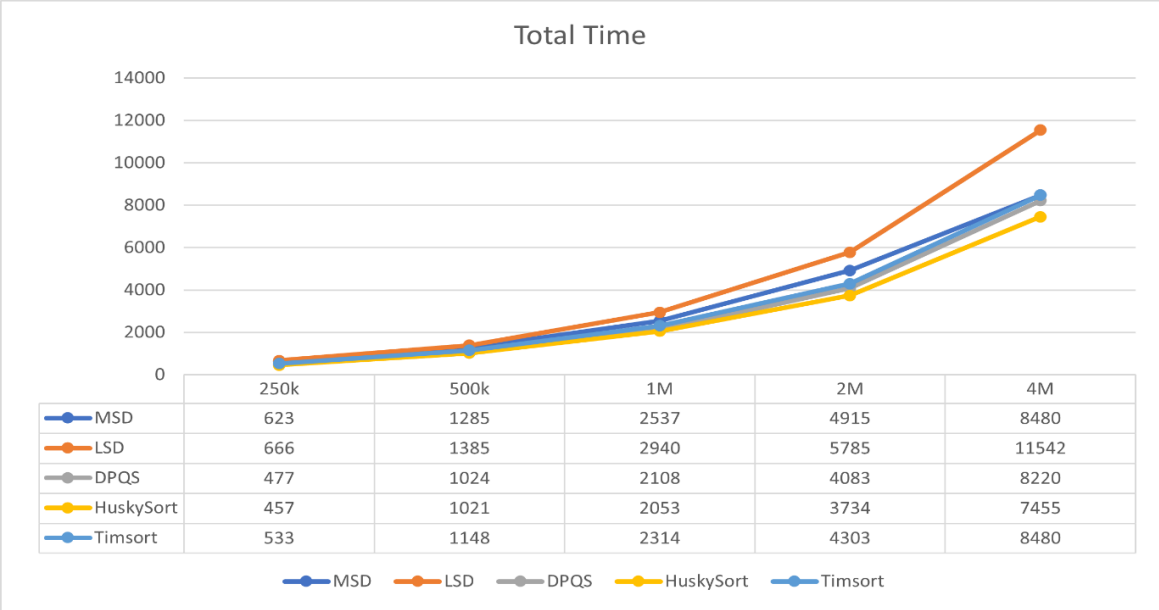
We rewrite the timer method and name it Timer. We record the sorting start system time and then record the sorting end system. And we minus this two times to count the total time of our sorting algorism. But this method can only use when the input is large. Since we count as microseconds.

4. The whole process

We first read the input method from shuffledChinese.txt and then we convert it to pinyin. We copied this pinyin string and sort it. Finally, we use matchIndex() methods to match the origin Chinese string, pinyin String and sorted pinying string. And finally we use file writer to print output.

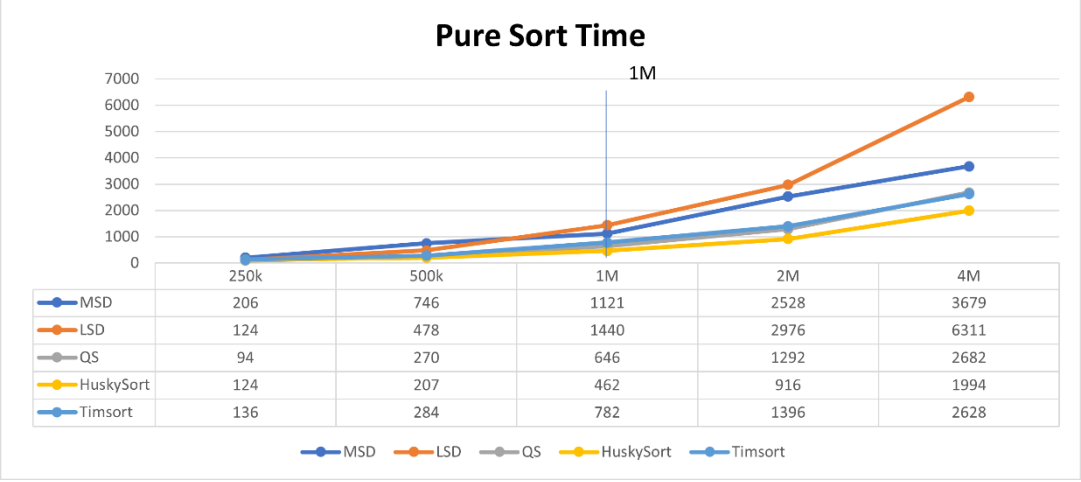
- **Output**

1. The following is a screenshot of the results of multiple algorithms running on different scales of data and only count the **total sort time**.

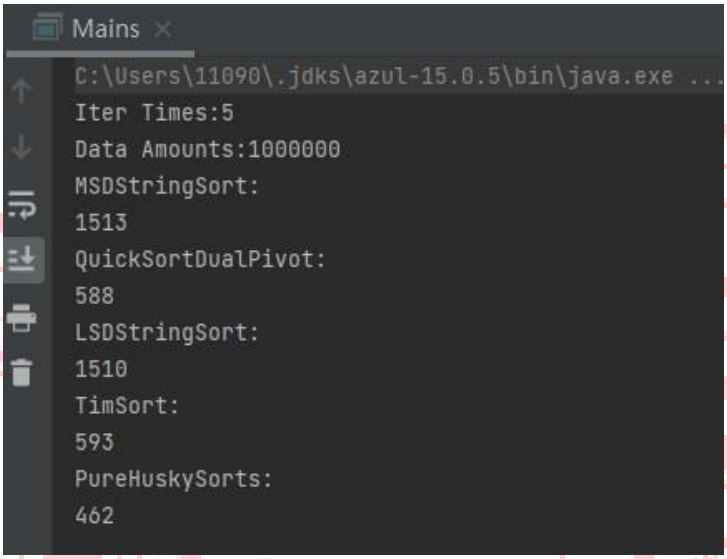



This picture is a line graph display of the running time of the same algorithm on data of different scales. See from this picture, we find that Husky sort algorithm is the most efficient algorism. The growth trend among them is similar. Dual-pivot Quicksort and Tim sort have the same speed in most of time. LSD performs better on small-scale data, but on larger data volumes, time consumption is greatly increased. The MSD algorithm performs poorly on small-scale data, but performs better on larger-scale data.

The following is a screenshot of the results of multiple algorithms running on different scales of data and only count the **pure sort time**.



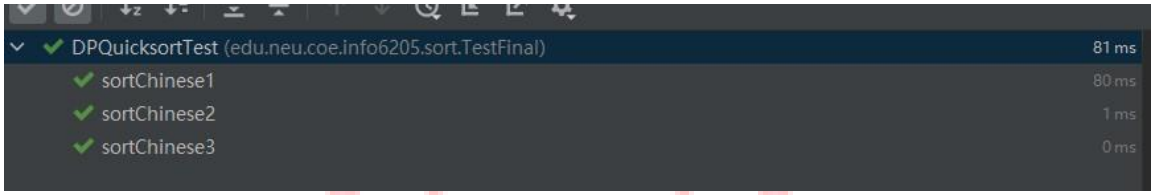
2. Meanwhile, based on 1M input as we have seen, the most efficient algorithm is the HuskySort algorithm, which runs less than the other four algorithms on the same amount of data. And at this time MSD and LSD are slowly .Dual-Pivot quicksort and Tim Sort are as quick as each other.



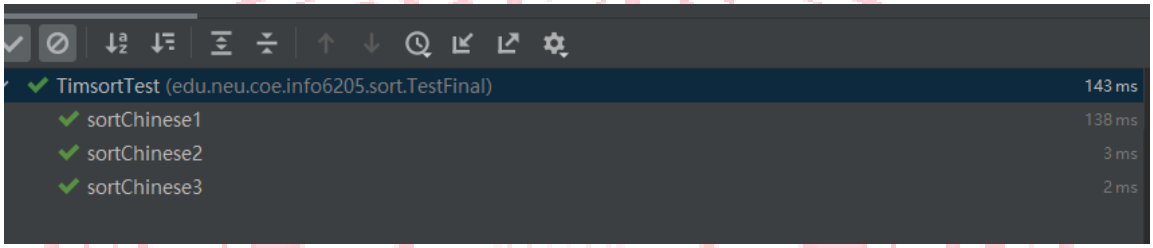

3. We also print the output as a txt file when input is 1M data.

- **Unit tests result:(Snapshot of successful unit test run)**

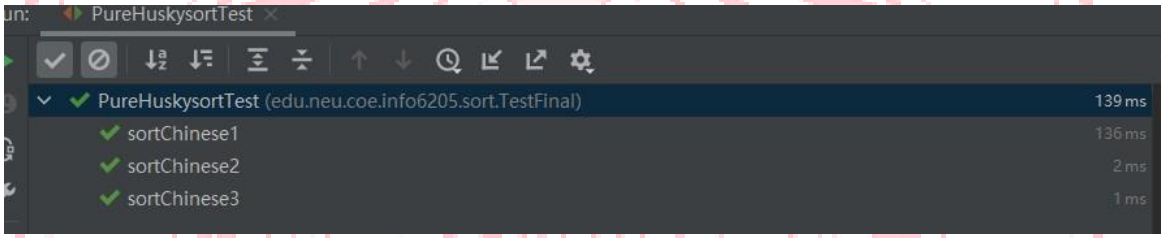
1. Dual-Pivot quicksort



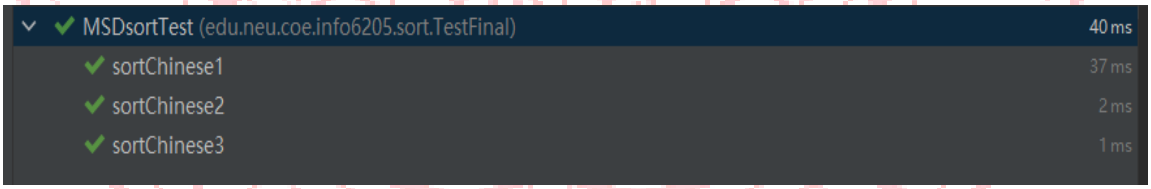

2.Tim Sort




3.PureHusky Sort




4.MSD Radix Sort




5.LSD Radix Sort