Program Structures & Algorithms
Fall 2021
Team Project

- **Task (List down the tasks performed)**
  - Using five different algorithms, Timsort, Dual-pivot Quicksort, Huskysort, LSD radix sort, and MSD radix sort, to sort the natural language, Chinese (simplified), based on the pinyin order. At the same time, the sorting accuracy of these five algorithms is tested by unit test. Then using the benchmark from assignment 2 is used to test the sorting efficiency of the five algorithms for 250k, 500k and 1M, 2M, 4M names.
  - For Timsort, we rewrite the code to implement the Chinese sort, and write a new unit test for this method. For Huskysort, we choose the PureHuskysort to implement the Chinese sort and have made some modifications to realize sorting the pinyin order. To do that, we made some changes on the HuskySortCoderFactory either, we added ChineseUnicodeCoder in huskyEncoder. And we have wrote a new unit test for the PureHuskysory either. For the other three sort methods (Dual-pivot Quicksort, LSD radix sort, and MSD radix sort), we basically followed and used the origin code and just add the Collator to sort the Chinese. But for these sorting methods, we also write new test for related sort to test their accuracy.

- **Conclusion**

Dual-PivotQuicksort < LSDRadixSort < MSDRadixSort < PureHuskysort < Timsort

When the length of name list is less than 1M, the results between the five sort algorithms do not have obvious difference. The Dual-pivot Quick-sort performs better than other methods as the number increases. LSD radix sort and MSD radix sort followed, their growth become slower, although getting twice the number of names.

- **Evidence to support the conclusion:**

1. Output (Snapshot of Code output in the terminal)

We use five algorithms (Dual-pivot Quicksort, LSD radix sort, MSD radix sort, Pure Huskysort, Timsort) to sort the name lists with different lengths (250k, 500k and 1M,

2M, 4M). Then repeat each sorting ten times and obtain the average sorting time. The implementation and the run-time result of Benchmark as below.

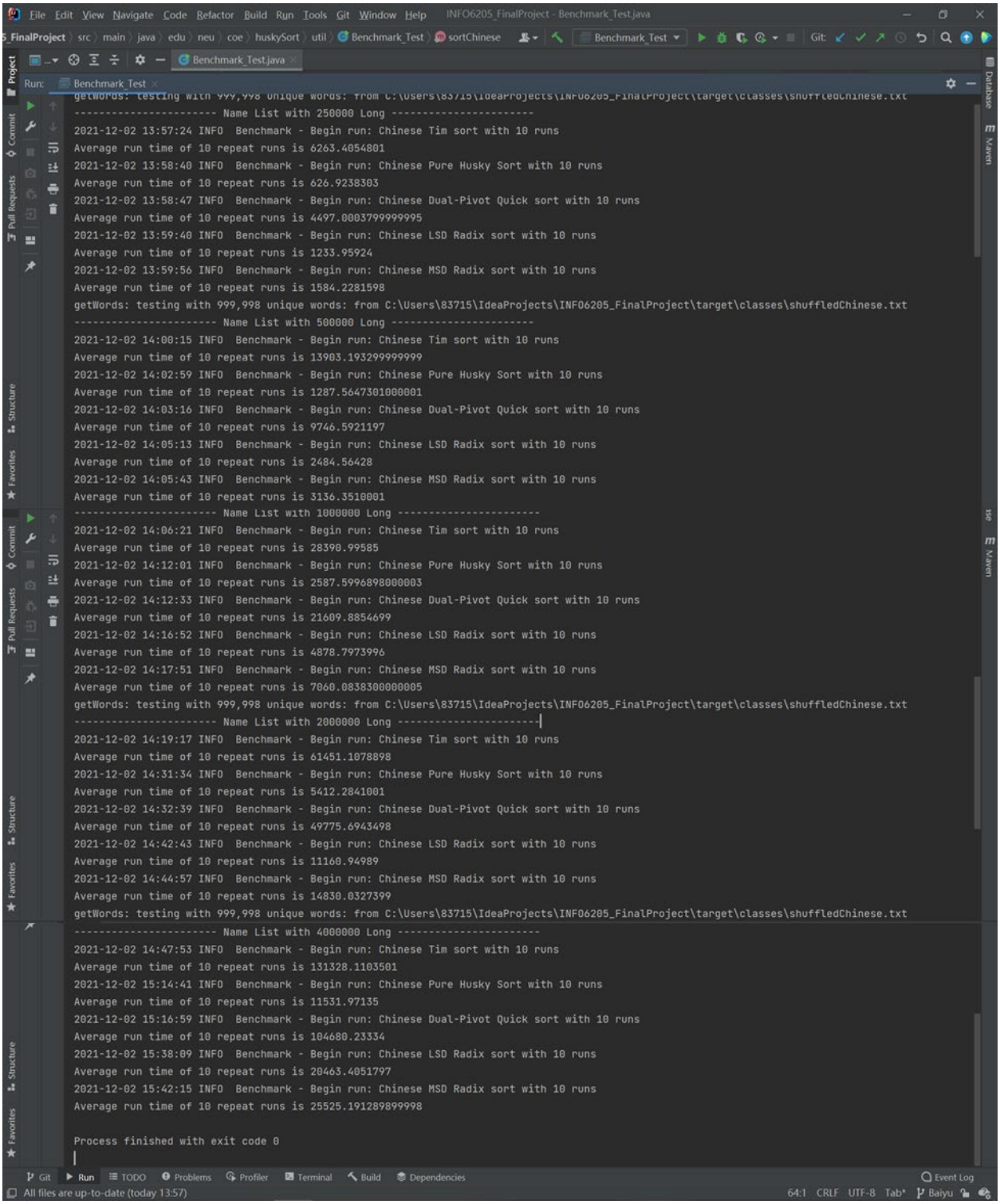

Figure 1 Benchmark Result

2. Graphical Representation (Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)

We put the average running time of each sort of name list with different length for ten times in the Excel table. And then draw the histogram and line chart for each sorting method according to the running time and the length of the name list using the data from the Excel, so that we can better observe the results.

	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

Figure 3 Benchmark Result in Excel

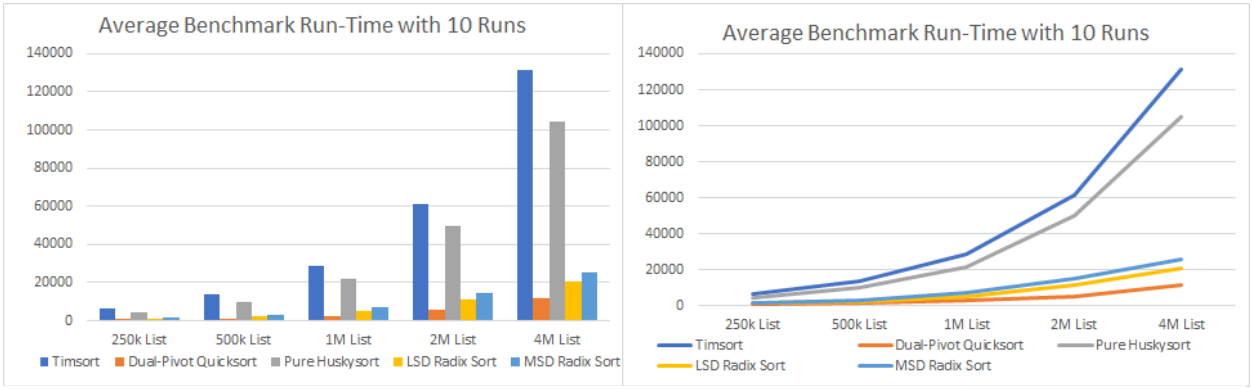

Figure 2 Histogram and Line Chart

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

We rewritten the unit tests of all sorting methods. In the test, we tested the sorting of a given array, the sorting from reading documents, and tested sort with instrumentation. For benchmark, we directly use the original unit test for testing.

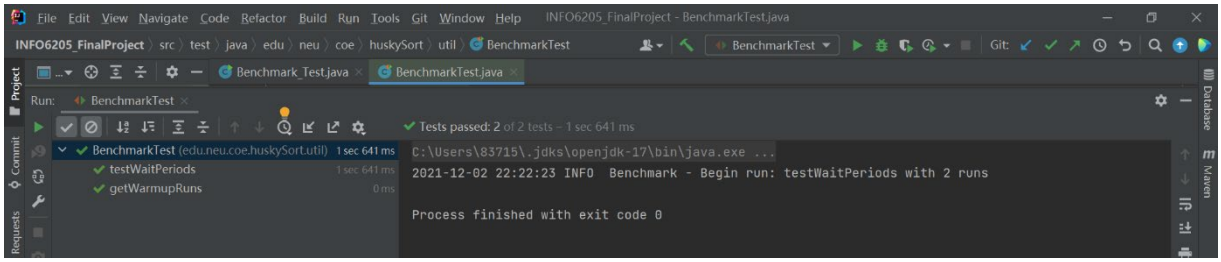

Figure 4 Benchmark Test Result

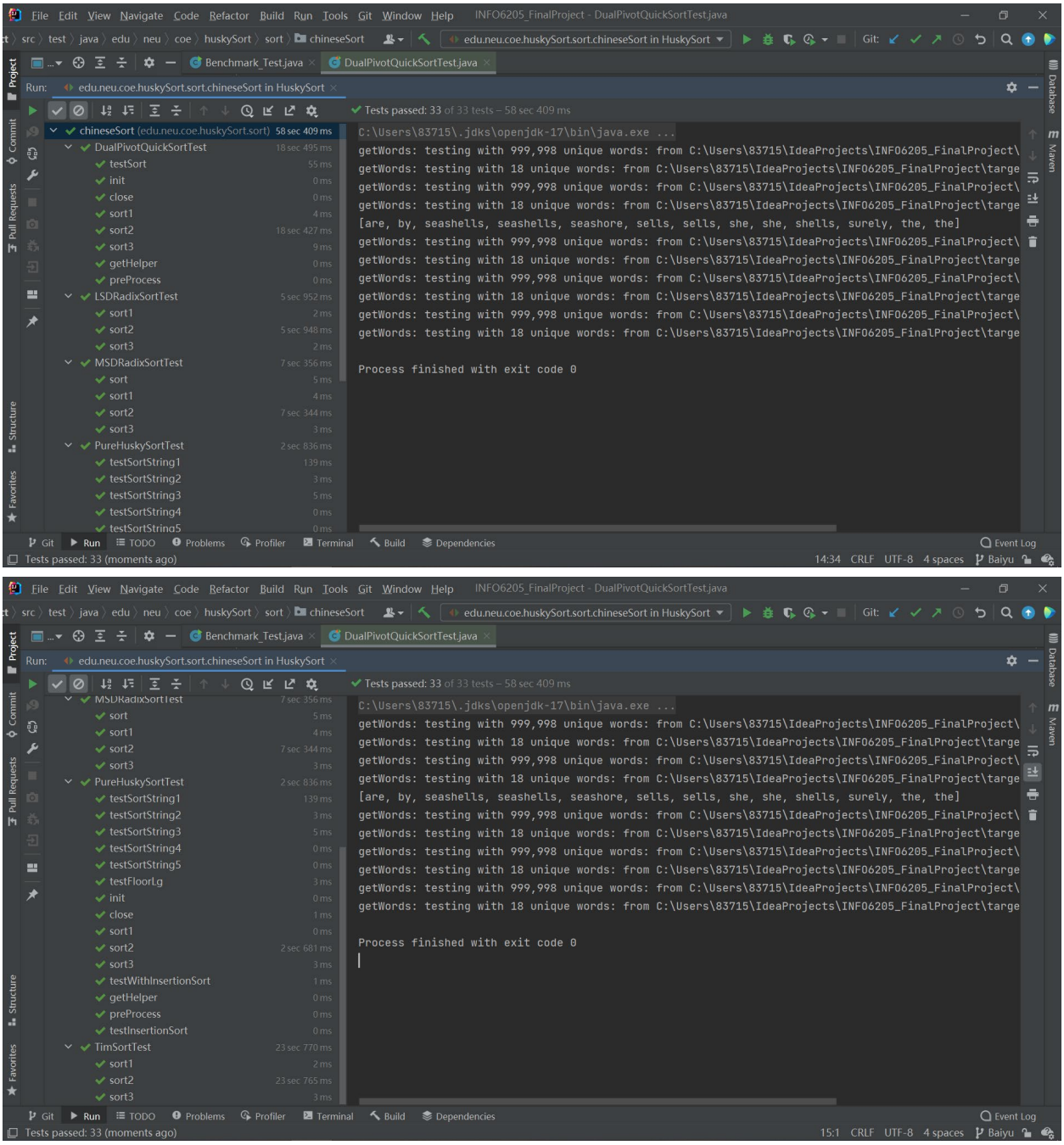

Figure 5 Sorting Methods Test Result