Shuo Wang(001533020)

Zeen Wang(001082883)

Xiyue Suo(001348347)

Program Structures & Algorithms

Fall 2021

Final Project Report

⊙ Task (List down the tasks performed in the Assignment)

We sorted array of Chinese names into pinyin order with five kinds of sort algorithms: MSD radix sort, LSD radix sort, Dual-Pivot Quick Sort, Tim sort and Husky sort. We wrote specific unit test with Chinese names and measured the performing time with benchmarks.

We firstly converted all Chinese names to lower case pinyin and each Chinese character followed by its tone (number 1, 2, 3, or 4) and separated by a space. We added tone because that two different Chinese character may have same pinyin. For example, “王硕”was converted to “wang2 shuo4.” Then we put all Chinese names and their pinyin to a HashMap. Then we got the order of pinyin with sort algorithms, now the corresponding Chinese names listed as values in the map.

Our benchmarks counted the running time of the sort process containing the mapping and sorting process. It gave an average time of 10 runs.

Ran tests with 5 different array sizes: 0.25million, 0.5 million, 1 million, 2 million and 4 million Chinese names.

⊙ Implement

1. We finished the code of the MSD radix sort and adapted other sort to Chinese. We used the dependency which can convert the Chinese to pinyin called pinyin4j. To improve the

accuracy of the Chinese words with same pinyin, we add the tone at the end of every pinyin.

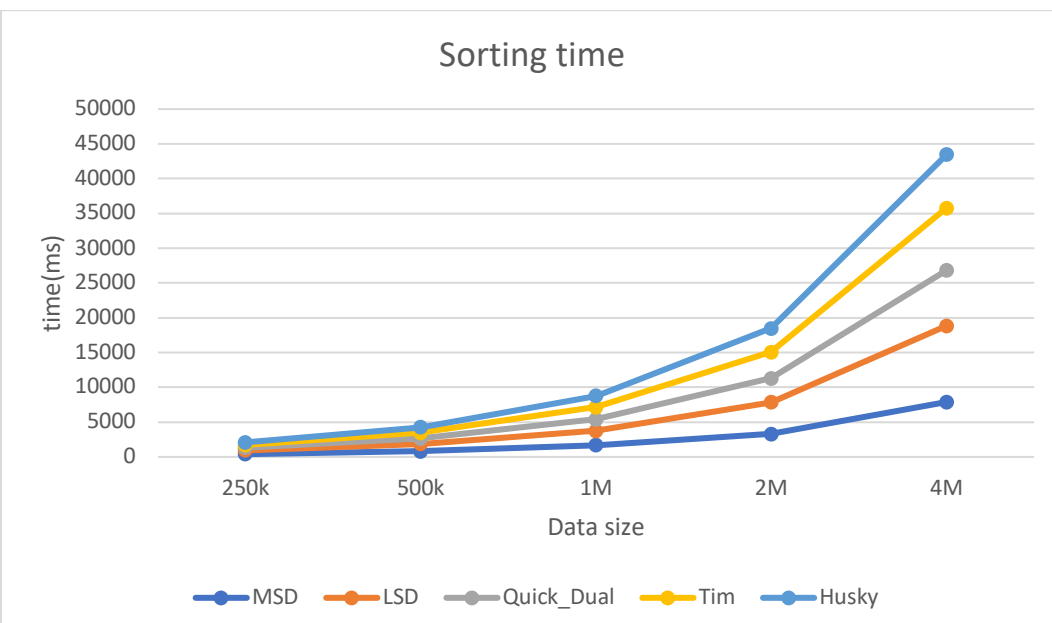
2. We used the Identity HashMap to set the different words in Chinese which have the same pinyin and tone. We did not use the array at MSD radix sort and LSD radix sort, to make the error of different sort as low as possible. Thus, we can compare the sorting efficiency without other effects.
3. And we added unique Chinese test in the unit test for every sort, to make sure our sorting method is correct.

⊙ Results and Observations

- **Table 1 Sorting Time for Different Array Size:**

Size/Time	MSD	LSD	Quick_Dual	Tim	Husky
250k	392.865875	543.119088	382.740971	398.757271	375.495275
500k	815.630338	1041.99552	798.450225	830.409767	778.58295
1M	1681.25228	2081.67405	1630.39014	1759.8964	1604.40378
2M	3288.14192	4564.3655	3466.55106	3740.55863	3401.49164
4M	7891.54064	10954.4722	7973.06744	8977.23849	7653.35618

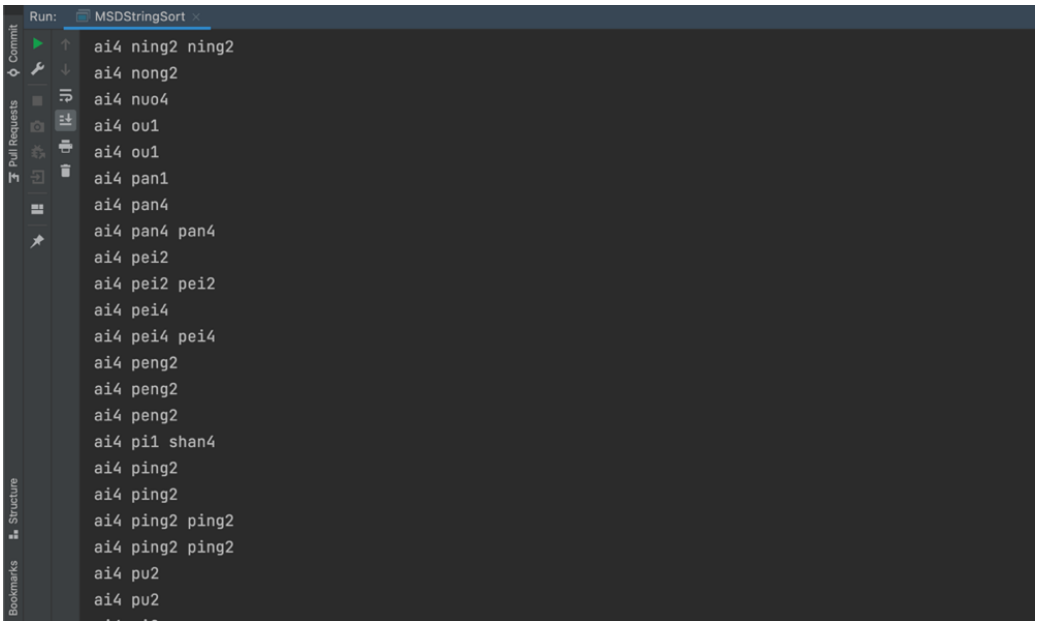
- **Chart 1 Sorting Time of 5 Sorting Algorithms**



- **Table 2 Sample Output Order**

	First 1000 Output				
	MSD	LSD	DP-QS	PureHusky	Timsort
3	阿安	阿安	阿安	阿安	阿安
4	阿斌	阿斌	阿彬	阿彬	阿斌
5	阿滨	阿滨	阿滨	阿滨	阿滨
6	阿彬	阿彬	阿斌	阿斌	阿彬
7	阿冰	阿冰	阿兵	阿兵	阿冰
8	阿兵	阿兵	阿冰	阿冰	阿兵
9	阿冰冰	阿冰冰	阿冰冰	阿冰冰	阿冰冰
10	阿婵	阿婵	阿婵	阿婵	阿婵
11	阿超	阿超	阿超	阿超	阿超
12	阿朝	阿朝	阿朝	阿朝	阿朝
13	阿琛	阿琛	阿琛	阿琛	阿琛
14	阿臣	阿臣	阿辰	阿晨	阿臣
15	阿晨	阿晨	阿臣	阿辰	阿晨
16	阿辰	阿辰	阿晨	阿臣	阿辰
17	阿称	阿称	阿称	阿称	阿称
18	阿诚	阿诚	阿诚	阿诚	阿诚
19	阿澄	阿澄	阿澄	阿澄	阿澄
20	阿弛	阿弛	阿弛	阿弛	阿弛

- **Sample Chinese Names in Pinyin Version**



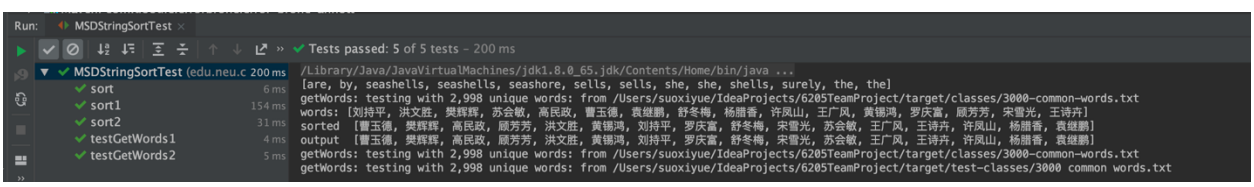
⊙ Conclusions

- From table 2 output, we compared the positions of sorted Chinese names with same pinyin and tone, such as "阿斌 577658", “阿滨 790796” and “阿彬 945627” with their original order on input, we can see MSD and LSD radix sorts, and Huskysort maintained the original order(stable), dual-pivot quick sort and PureHuskuy sort are not.
- Husky sort is faster than dual-pivot quick sort and Timsort when sorting objects because it reduces the array access, here we are sorting Strings, this is consistent with theoretical conclusion from Professor Robin’s paper Huskysort.
- Based on our benchmark results, Huskysort gives best performance, comparing to MSD/LSD radix sorts, Timsort, and dual-pivot quick sort. LSD took most time in sorting Chinese names among those 5 sorts.
- The sorting times are linearithmic. (~NlogN)

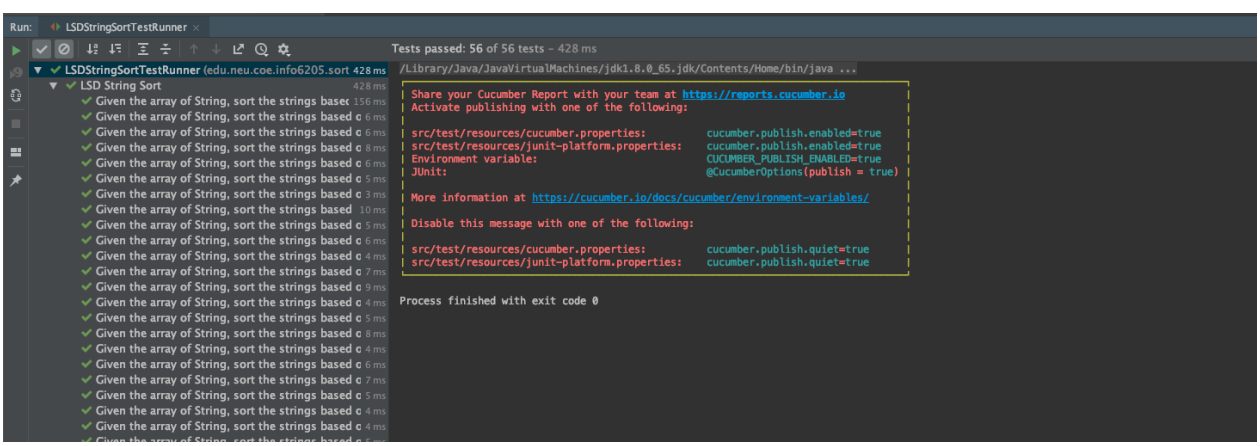
⊙ Output

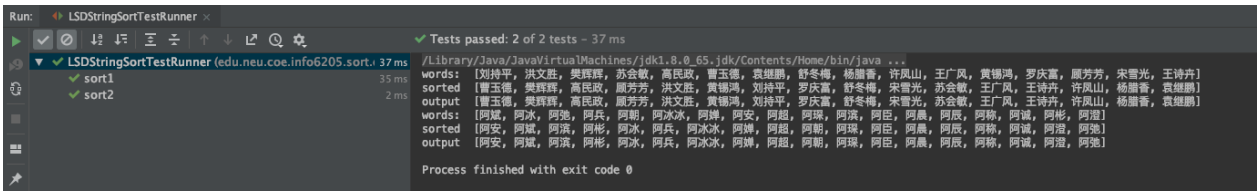
- **Unit Tests**

MSD sort:



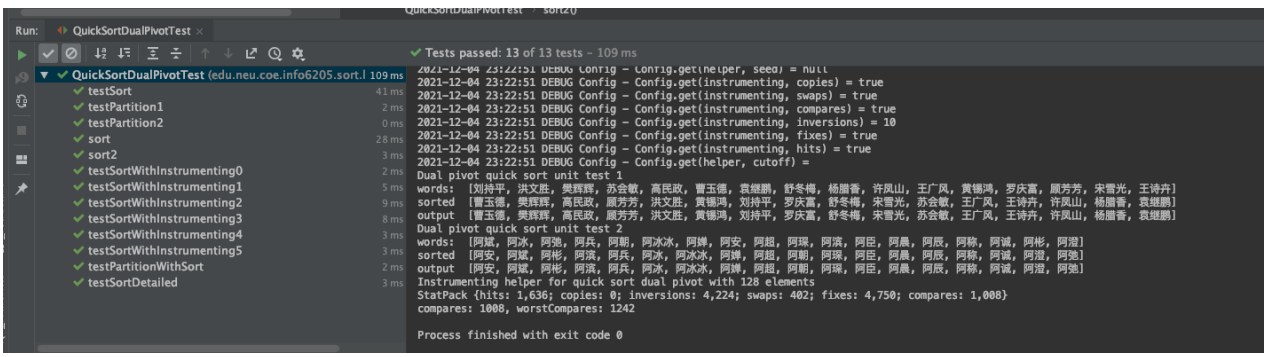
LSD sort:





Tim sort:

Dual-pivot sort:



“From the unit test 2, we notice the dual pivot have different order for Chinese names with exactly same pinyin and tone, comparing to previous sorts. Will compare which sort follows the original order, this will give which sorting algorithm is in place.”

Husky sort:

- **Benchmarks:**

Benchmarks with 1M Chinese names:

MSD sort:

```
MSDStringSort
/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/bin/java ...
2021-12-04 22:48:34 INFO  Benchmark_Timer - Begin run: MSD String Sort with 10 runs
Msd String Sort -- average time in milliseconds: 1721.3826207999998

Process finished with exit code 0
```

LSD sort:

```
LSDStringSort
/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/bin/java ...
2021-12-04 22:50:06 INFO  Benchmark_Timer - Begin run: LSD String Sort with 10 runs
Lsd String Sort -- average time in milliseconds: 2182.9351875

Process finished with exit code 0
```

Tim sort:

```
Run: TimSort
/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/bin/java ...
2021-12-04 22:51:48 INFO  Benchmark_Timer - Begin run: Tim String Sort with 10 runs
Tim String Sort -- average time in milliseconds: 1780.881396

Process finished with exit code 0
```

Dual-pivot sort:

```
Run:  QuickSort_DualPivot
/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/bin/java ...
2021-12-04 22:51:05 DEBUG Config - Config.get(helper, instrument) = false
2021-12-04 22:51:05 INFO  Benchmark_Timer - Begin run: QCD String Sort with 10 runs
QCD String Sort -- average time in milliseconds: 1628.8370123999998

Process finished with exit code 0
```

Husky sort:

```
Run:  PureHuskySort
/Library/Java/JavaVirtualMachines/zulu-11.jdk/Contents/Home/bin/java ...
2021-12-05 14:07:19 INFO  Benchmark_Timer - Begin run: PHusky String Sort with 10 runs
PHusky String Sort -- average time in milliseconds: 1539.3529376000001

Process finished with exit code 0
Git  Run  TODO  Problems  Profiler  Terminal  Endpoints  Build  Dependencies
Build completed successfully in 1 sec, 25 ms (moments ago)
```