

Name (NUID) 001586014 Hao Fu

Name (NUID) 001517230 Chao Yan

Program Structures & Algorithms

Fall 2021

Final Project

◉ Task

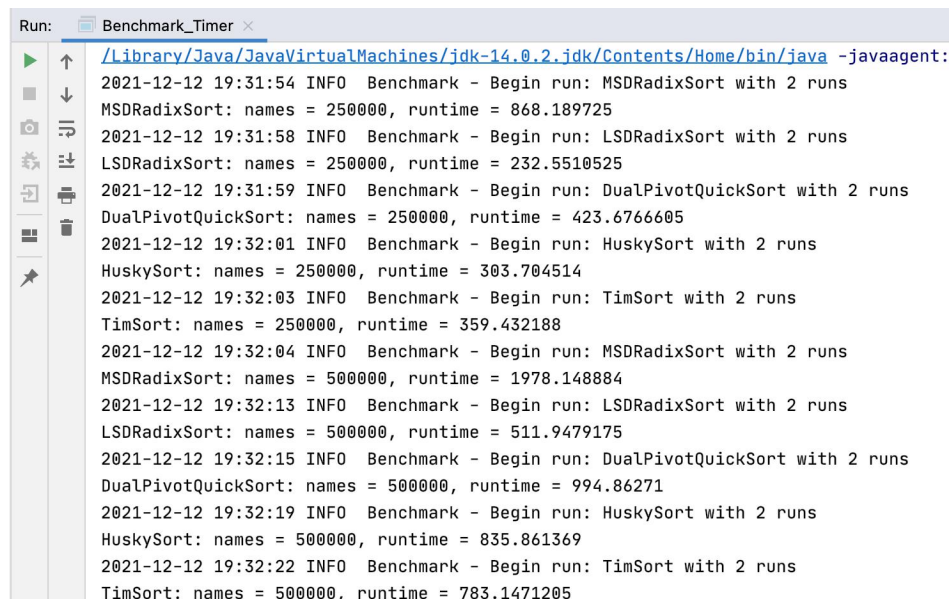
Your task is to implement MSD radix sort for a natural language which uses Unicode characters. You may choose your own language or (Simplified) Chinese. Additionally, you will complete a literature survey of relevant papers and you will compare your method with Timsort, Dual-pivot Quicksort, Huskysort, and LSD radix sort.

◉ Introduction

We use Collator class to compare Chinese words by setting new Collator.getInstance(Locale.CHINA). We use 5 algorithms to sort 1M Chinese names and analyze the 5 algorithms. The sorting results of the algorithms are placed under the resource file.

◉ Evidence to support the conclusion:

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



```
Run: Benchmark_Timer x
/Library/Java/JavaVirtualMachines/jdk-14.0.2.jdk/Contents/Home/bin/java -javaagent:
2021-12-12 19:31:54 INFO Benchmark - Begin run: MSDRadixSort with 2 runs
MSDRadixSort: names = 250000, runtime = 868.189725
2021-12-12 19:31:58 INFO Benchmark - Begin run: LSDRadixSort with 2 runs
LSDRadixSort: names = 250000, runtime = 232.5510525
2021-12-12 19:31:59 INFO Benchmark - Begin run: DualPivotQuickSort with 2 runs
DualPivotQuickSort: names = 250000, runtime = 423.6766605
2021-12-12 19:32:01 INFO Benchmark - Begin run: HuskySort with 2 runs
HuskySort: names = 250000, runtime = 303.704514
2021-12-12 19:32:03 INFO Benchmark - Begin run: TimSort with 2 runs
TimSort: names = 250000, runtime = 359.432188
2021-12-12 19:32:04 INFO Benchmark - Begin run: MSDRadixSort with 2 runs
MSDRadixSort: names = 500000, runtime = 1978.148884
2021-12-12 19:32:13 INFO Benchmark - Begin run: LSDRadixSort with 2 runs
LSDRadixSort: names = 500000, runtime = 511.9479175
2021-12-12 19:32:15 INFO Benchmark - Begin run: DualPivotQuickSort with 2 runs
DualPivotQuickSort: names = 500000, runtime = 994.86271
2021-12-12 19:32:19 INFO Benchmark - Begin run: HuskySort with 2 runs
HuskySort: names = 500000, runtime = 835.861369
2021-12-12 19:32:22 INFO Benchmark - Begin run: TimSort with 2 runs
TimSort: names = 500000, runtime = 783.1471205
```

```

2021-12-12 19:32:25 INFO Benchmark - Begin run: MSDRadixSort with 2 runs
MSDRadixSort: names = 1000000, runtime = 3935.710022
2021-12-12 19:32:40 INFO Benchmark - Begin run: LSDRadixSort with 2 runs
LSDRadixSort: names = 1000000, runtime = 1068.5325
2021-12-12 19:32:45 INFO Benchmark - Begin run: DualPivotQuickSort with 2 runs
DualPivotQuickSort: names = 1000000, runtime = 1999.3868835
2021-12-12 19:32:53 INFO Benchmark - Begin run: HuskySort with 2 runs
HuskySort: names = 1000000, runtime = 1440.9269485
2021-12-12 19:32:59 INFO Benchmark - Begin run: TimSort with 2 runs
TimSort: names = 1000000, runtime = 1833.228236
2021-12-12 19:33:06 INFO Benchmark - Begin run: MSDRadixSort with 2 runs
MSDRadixSort: names = 2000000, runtime = 5814.763113
2021-12-12 19:33:29 INFO Benchmark - Begin run: LSDRadixSort with 2 runs
LSDRadixSort: names = 2000000, runtime = 1785.3717295
2021-12-12 19:33:37 INFO Benchmark - Begin run: DualPivotQuickSort with 2 runs
DualPivotQuickSort: names = 2000000, runtime = 3516.1009845
2021-12-12 19:33:51 INFO Benchmark - Begin run: HuskySort with 2 runs
HuskySort: names = 2000000, runtime = 3022.1146975
2021-12-12 19:34:03 INFO Benchmark - Begin run: TimSort with 2 runs
TimSort: names = 2000000, runtime = 3642.9346485

2021-12-12 19:34:17 INFO Benchmark - Begin run: MSDRadixSort with 2 runs
MSDRadixSort: names = 4000000, runtime = 10938.6071585
2021-12-12 19:35:01 INFO Benchmark - Begin run: LSDRadixSort with 2 runs
LSDRadixSort: names = 4000000, runtime = 3489.423097
2021-12-12 19:35:15 INFO Benchmark - Begin run: DualPivotQuickSort with 2 runs
DualPivotQuickSort: names = 4000000, runtime = 9911.888967
2021-12-12 19:35:52 INFO Benchmark - Begin run: HuskySort with 2 runs
HuskySort: names = 4000000, runtime = 7793.566302
2021-12-12 19:36:22 INFO Benchmark - Begin run: TimSort with 2 runs
TimSort: names = 4000000, runtime = 8386.1037275

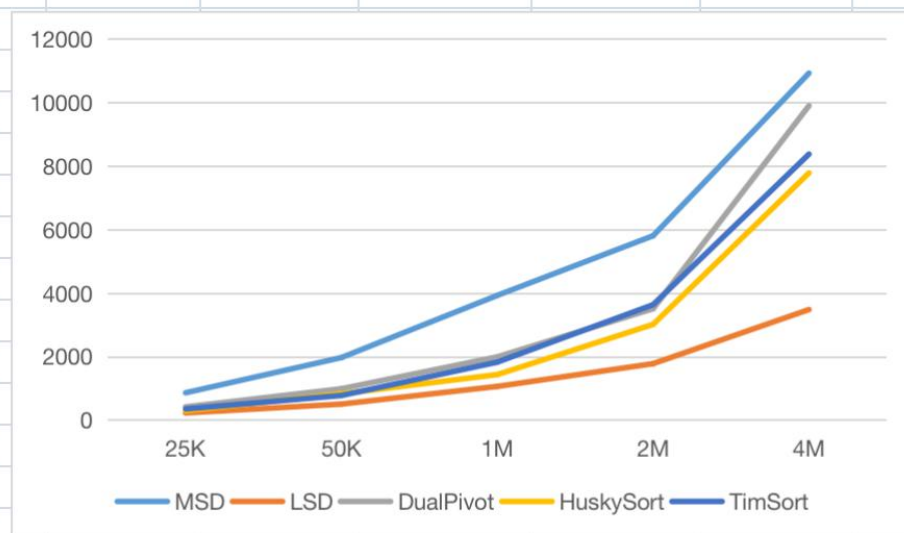
Process finished with exit code 0

```

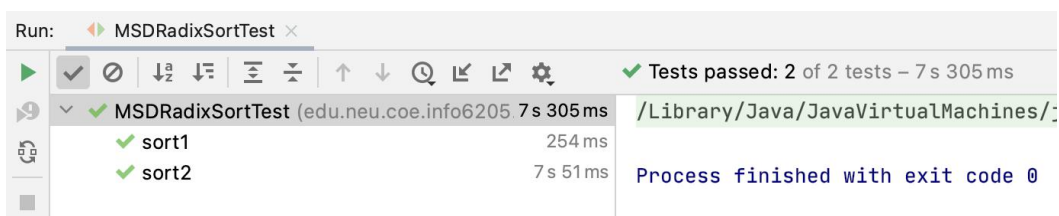
We use Benchmark_Timer class to test the running time of five algorithms for 250K, 500K, 1M, 2M, 4M names.

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

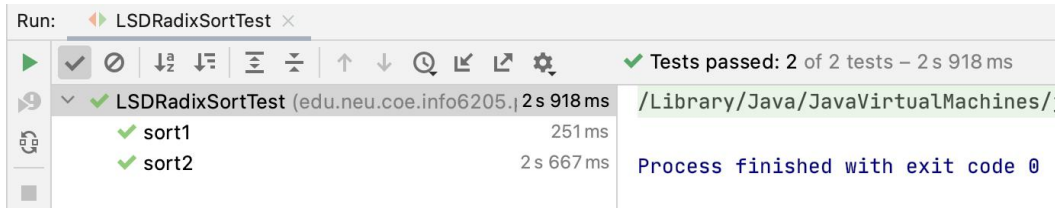
A	B	C	D	E	F	G	
	25K	50K	1M	2M	4M		
MSD	868.189	1978.149	3935.71	5814.763	10938.607		
LSD	232.551	511.948	1068.532	1785.372	3489.423		
DualPivot	423.677	994.863	1999.387	3516.1	9911.889		
HuskySort	303.705	835.861	1440.927	3022.115	7793.566		
TimSort	359.432	783.147	1833.228	3642.935	8386.104		



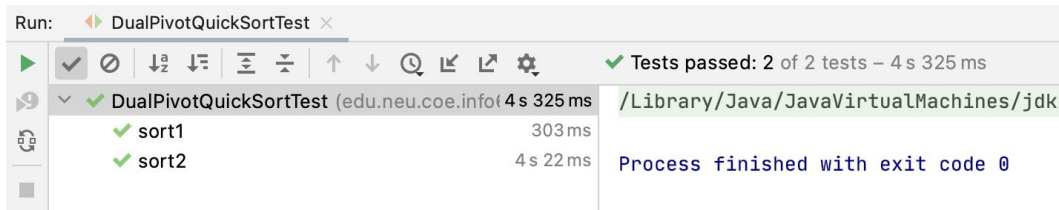
◦ Unit tests result:(Snapshot of successful unit test run)



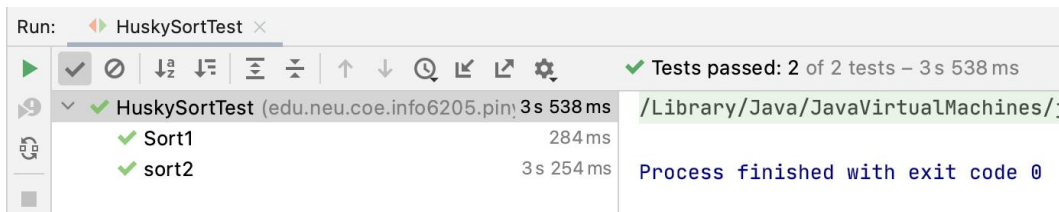
MSDRadixSort Test



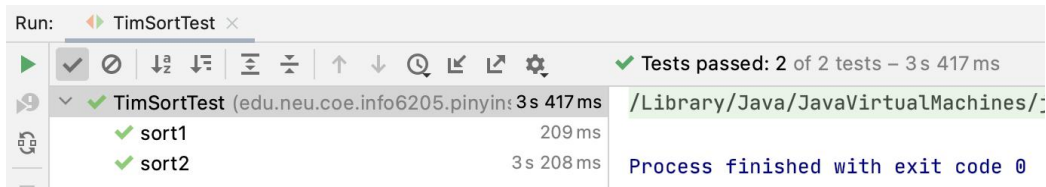
LSDRadixSort Test



DualPivotQuickSort Test



HuskySort Test



TimSort Test

Relationship Conclusion:

From the chart we can get the result of the running time:

LSDRadixSort < HuskySort < TimSort < DualPivotQuickSort < MSDRadixSort

The sorting time of five algorithms increases almost linearly with the increase of array length. MSDRadixSort has the longest sorting time and the worst performance. LSDRadixSort has the shortest sorting time and the best performance.