Varun Venkatesh Gowda (002126161)
Namrata Ruchandani (002125637)
Venkteshprasad Maya Rao (001087357)

Program Structures & Algorithms

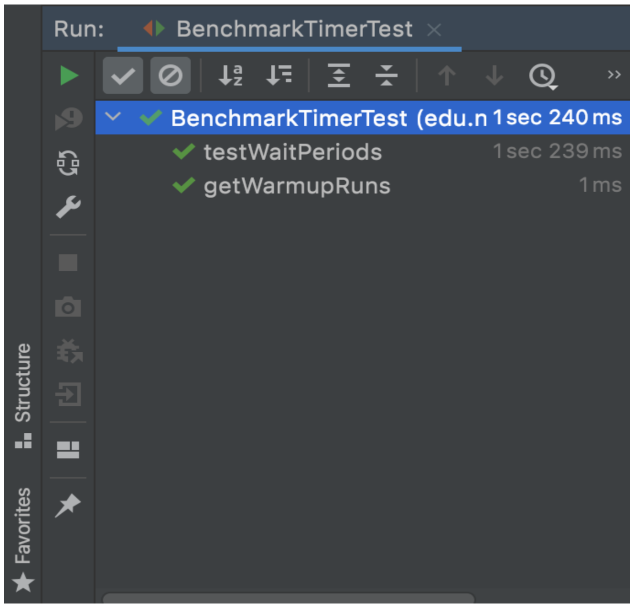
Fall 2021 Team Project

Tasks:

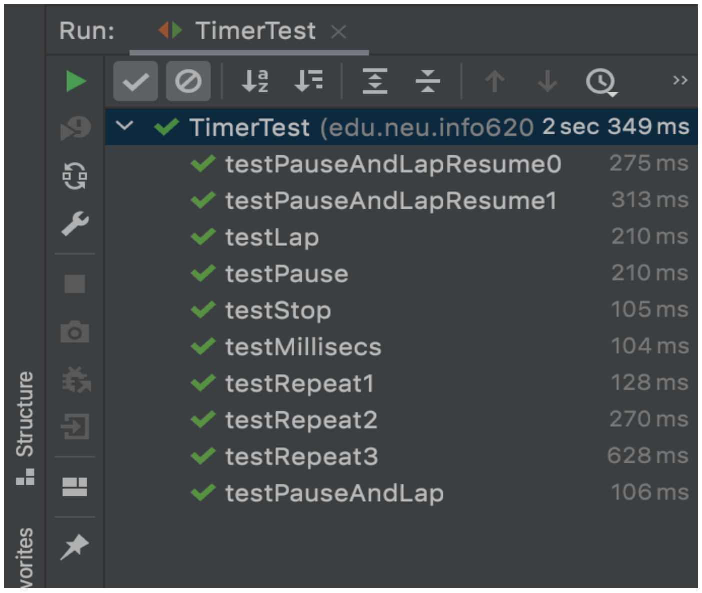
1. Implemented MSD Radix Sort, LSD Radix Sort, Dual Pivot Quick-Sort, Tim Sort and Husky Sort to sort an array of Chinese words in the order of the Pinyin.
2. Implemented Benchmarks in code by coding Benchmark timer to measure the running time of the sort algorithms.
3. Unit tested the sort algorithms and Benchmark utility using JUnit.
4. Implemented SortUtils for File Reader and string array generator, comparators, stubs for Chinese text and sorted Chinese text for unit tests and Chinese to pinyin converter.
5. Performed benchmarks for the sort algorithms with 250k, 500k, 1M, 2M and 4M Chinese names.
6. Recorded readings of benchmarks and tabulated the results to produce graphs for observation.
7. Prepared Report
8. Conducted readings of following papers for literature survey
    - Engineering Radix Sort by Peter M. Mcllroy and Keith Bostic University of California at Berkeley; and M. Douglas Mcllroy AT&T Bell Laboratories
    - PARADIS: A PARALLEL IN-PLACE RADIX SORT ALGORITHM by ROLLAND HE
    - Implementing Radixsort by Arne Andersonn, Department of Computer Science, Lund University; and Stefan Nilsson, Department of Computer Science, Helsinki University of Technology
9. Examined results and prepared conclusion

Screenshots:

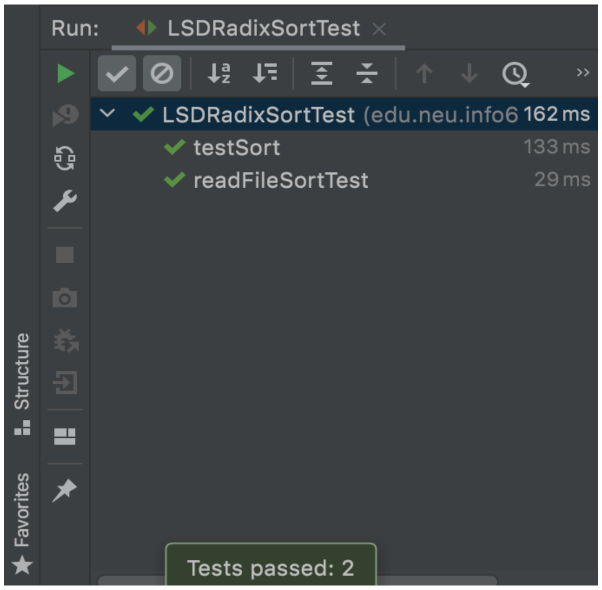
Test cases passing



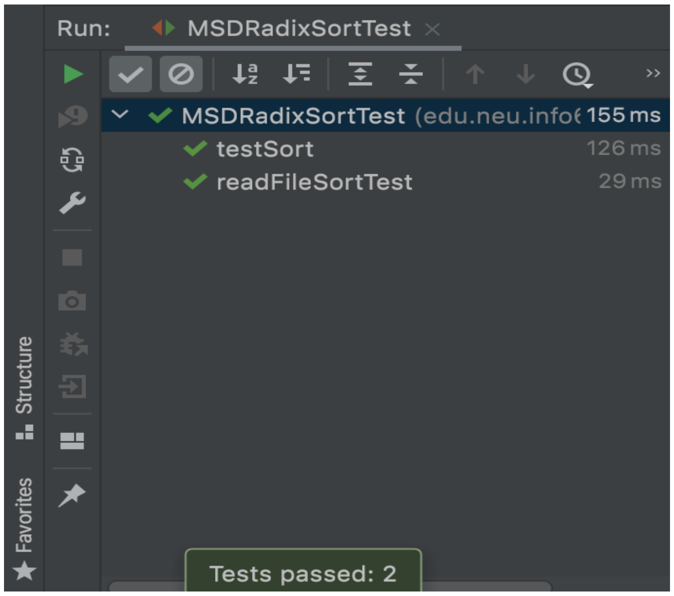
```
Run: BenchmarkTimerTest
BenchmarkTimerTest (edu.n 1 sec 240 ms
    testWaitPeriods         1 sec 239 ms
    getWarmupRuns                  1 ms
```



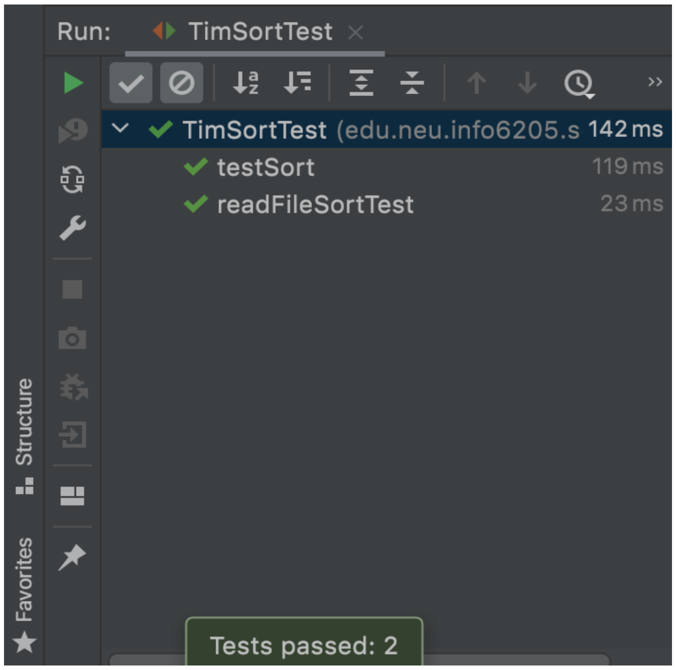
```
Run: TimerTest
TimerTest (edu.neu.info620 2 sec 349 ms
    testPauseAndLapResume0     275 ms
    testPauseAndLapResume1     313 ms
    testLap                    210 ms
    testPause                  210 ms
    testStop                   105 ms
    testMillisecs              104 ms
    testRepeat1                128 ms
    testRepeat2                270 ms
    testRepeat3                628 ms
    testPauseAndLap            106 ms
```



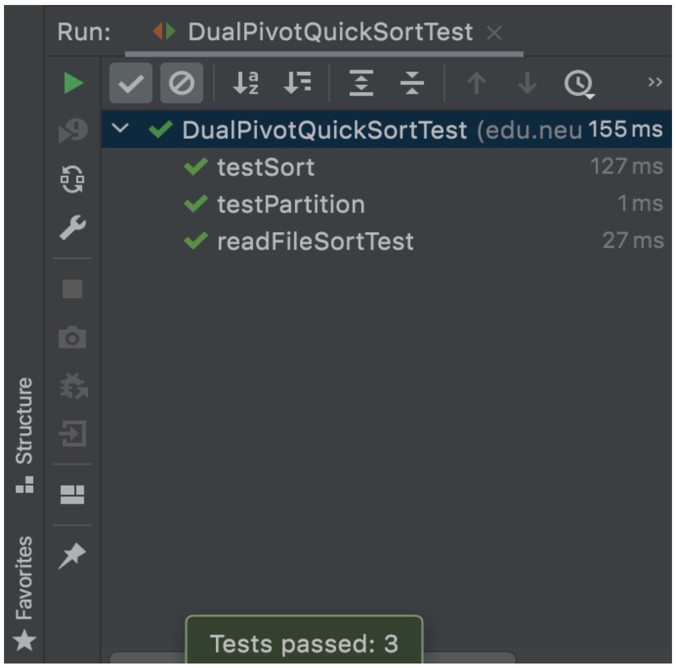
```
Run: LSDRadixSortTest
LSDRadixSortTest (edu.neu.info6 162 ms
    testSort                   133 ms
    readFileSortTest            29 ms
Tests passed: 2
```



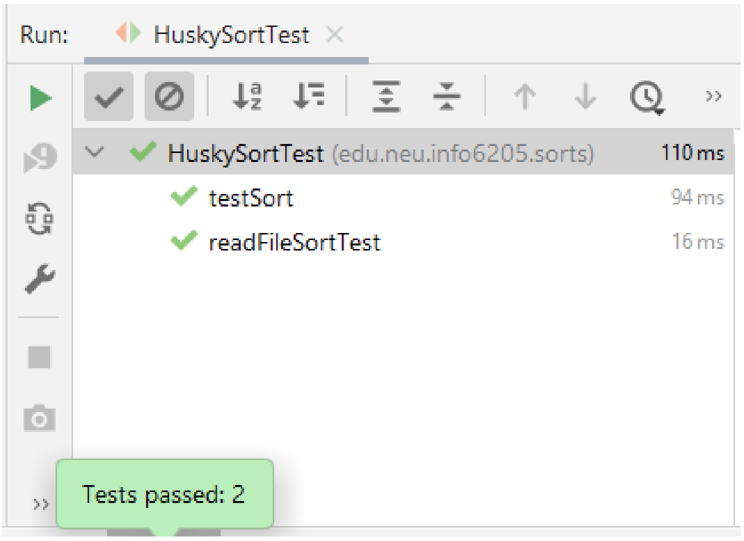
```
Run: MSDRadixSortTest
MSDRadixSortTest (edu.neu.info6 155 ms
    testSort                   126 ms
    readFileSortTest            29 ms
Tests passed: 2
```



```
Run: TimSortTest
✓ TimSortTest (edu.neu.info6205.s 142 ms
   ✓ testSort                    119 ms
   ✓ readFileSortTest             23 ms
Tests passed: 2
```

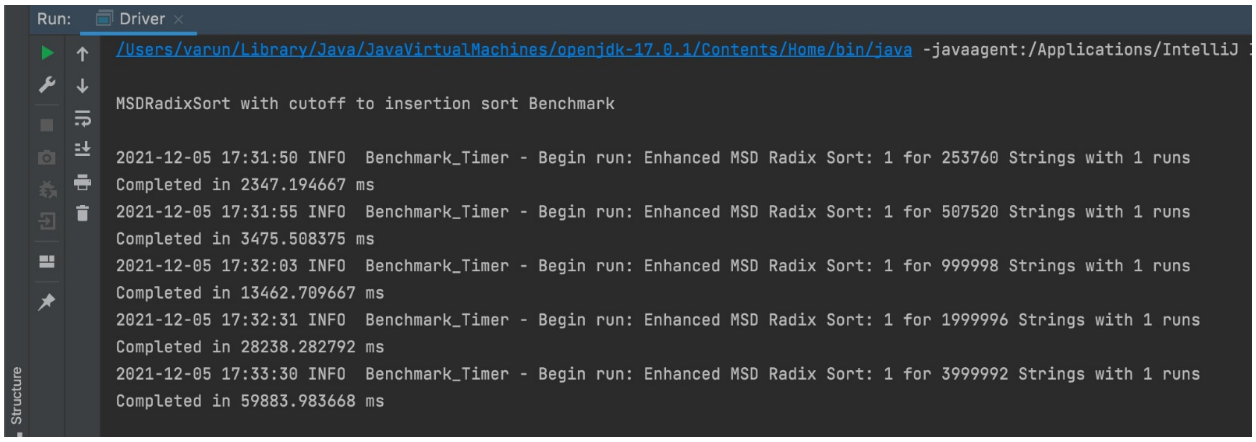


```
Run: DualPivotQuickSortTest
✓ DualPivotQuickSortTest (edu.neu 155 ms
   ✓ testSort                    127 ms
   ✓ testPartition                 1 ms
   ✓ readFileSortTest             27 ms
Tests passed: 3
```

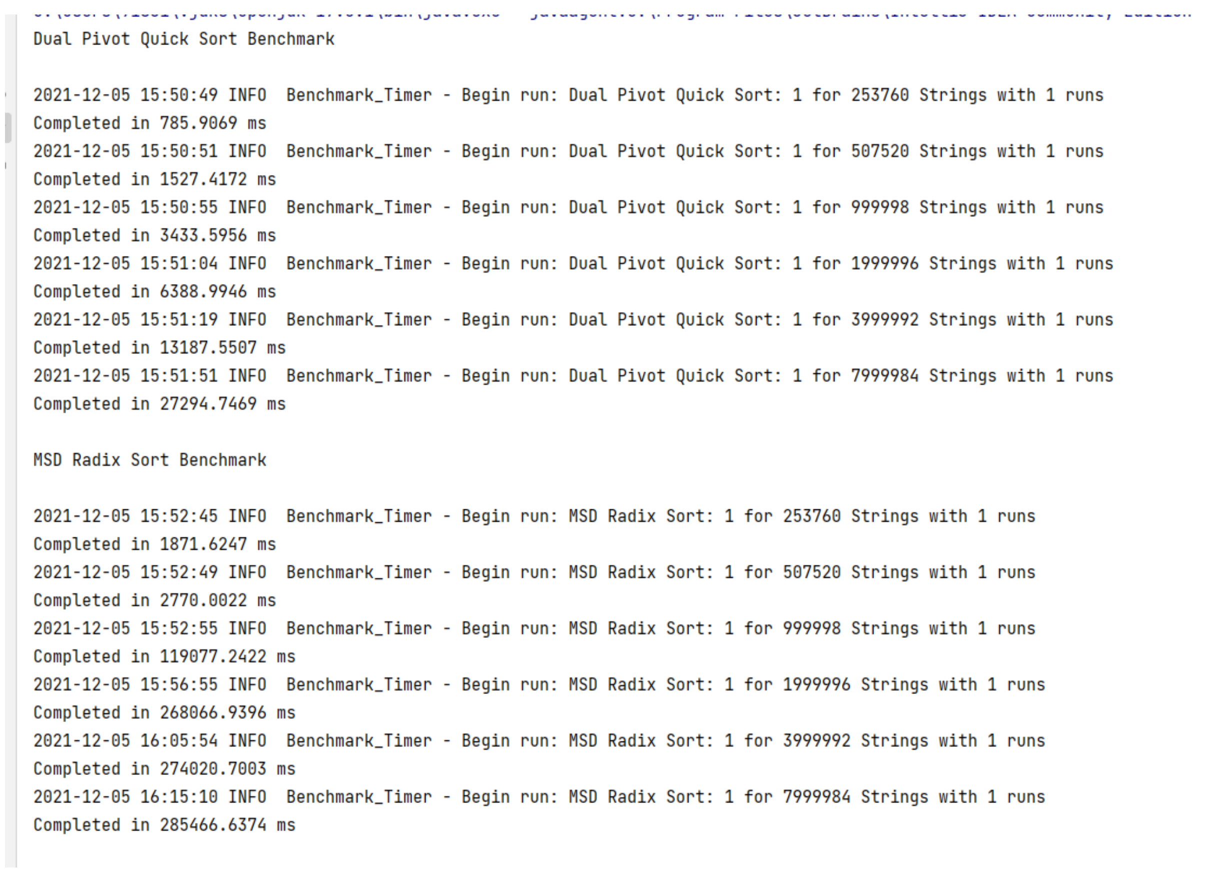


```
Run: HuskySortTest
✓ HuskySortTest (edu.neu.info6205.sorts)   110 ms
   ✓ testSort                               94 ms
   ✓ readFileSortTest                       16 ms
Tests passed: 2
```

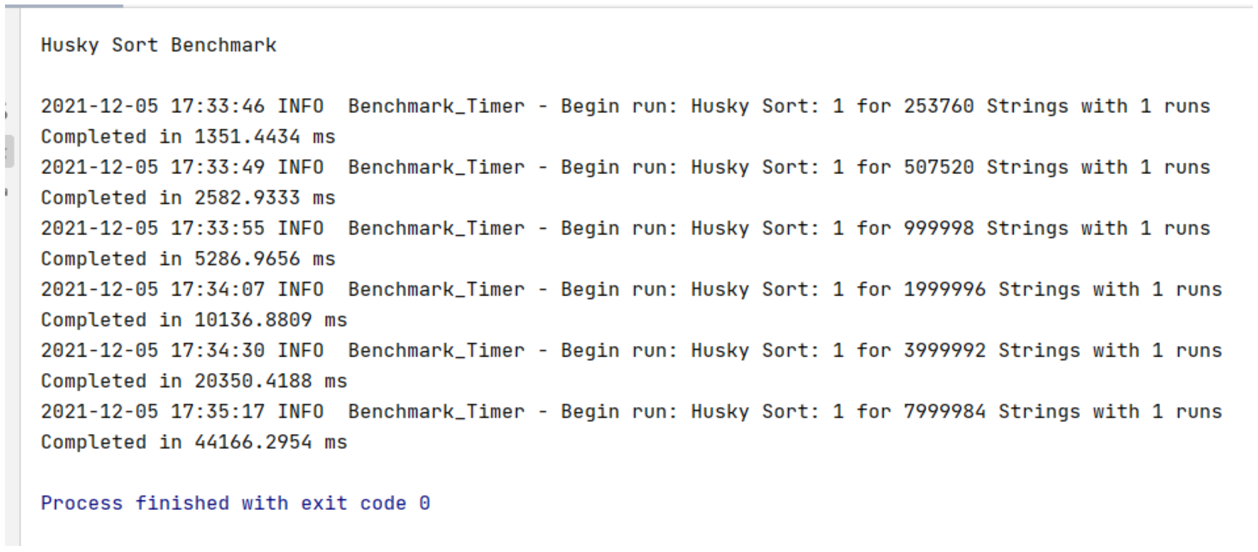
Benchmarking Results:



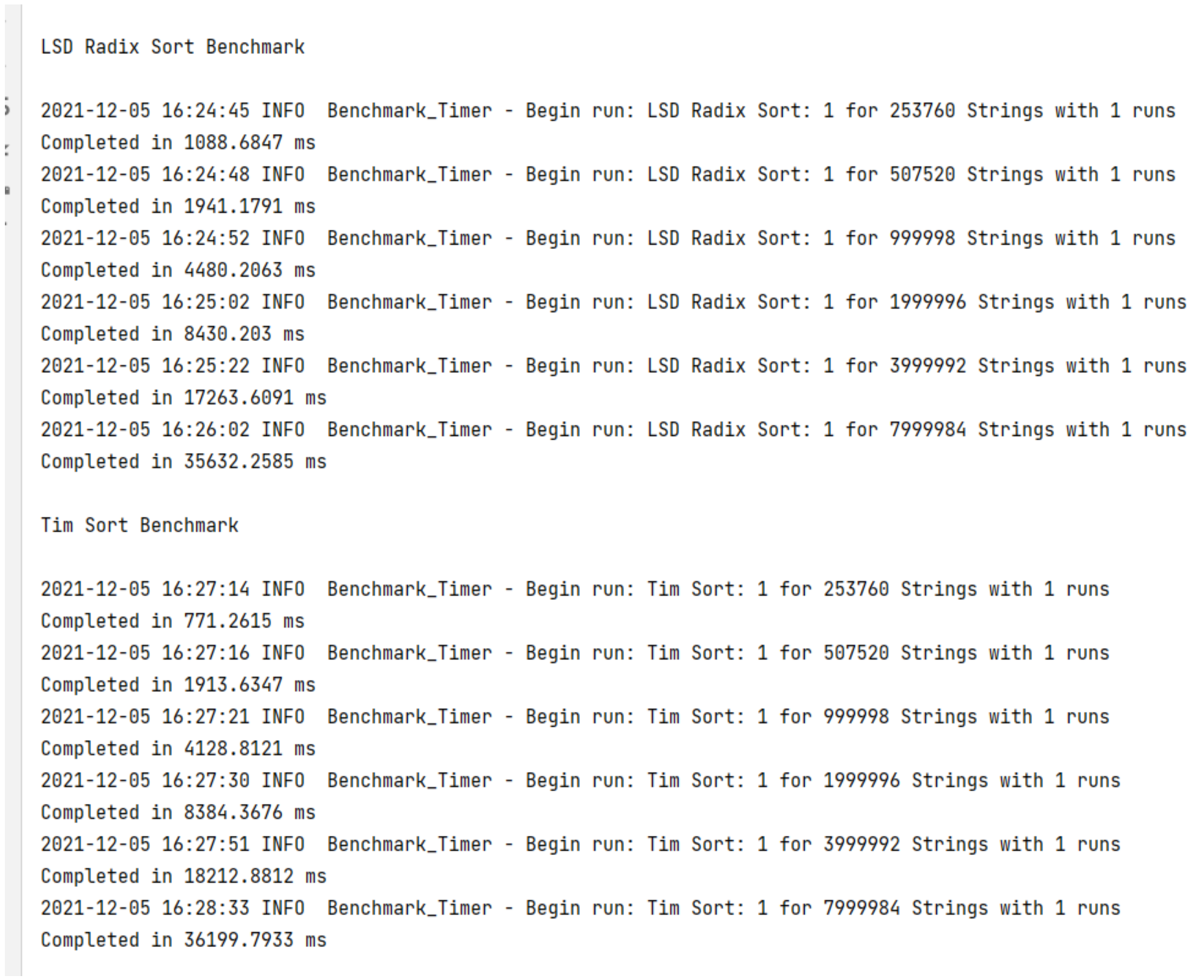
```
Run: Driver
/Users/varun/Library/Java/JavaVirtualMachines/openjdk-17.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ

MSDRadixSort with cutoff to insertion sort Benchmark

2021-12-05 17:31:50 INFO  Benchmark_Timer - Begin run: Enhanced MSD Radix Sort: 1 for 253760 Strings with 1 runs
Completed in 2347.194667 ms
2021-12-05 17:31:55 INFO  Benchmark_Timer - Begin run: Enhanced MSD Radix Sort: 1 for 507520 Strings with 1 runs
Completed in 3475.508375 ms
2021-12-05 17:32:03 INFO  Benchmark_Timer - Begin run: Enhanced MSD Radix Sort: 1 for 999998 Strings with 1 runs
Completed in 13462.709667 ms
2021-12-05 17:32:31 INFO  Benchmark_Timer - Begin run: Enhanced MSD Radix Sort: 1 for 1999996 Strings with 1 runs
Completed in 28238.282792 ms
2021-12-05 17:33:30 INFO  Benchmark_Timer - Begin run: Enhanced MSD Radix Sort: 1 for 3999992 Strings with 1 runs
Completed in 59883.983668 ms
```



```
[illegible]
Dual Pivot Quick Sort Benchmark

2021-12-05 15:50:49 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 253760 Strings with 1 runs
Completed in 785.9069 ms
2021-12-05 15:50:51 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 507520 Strings with 1 runs
Completed in 1527.4172 ms
2021-12-05 15:50:55 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 999998 Strings with 1 runs
Completed in 3433.5956 ms
2021-12-05 15:51:04 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 1999996 Strings with 1 runs
Completed in 6388.9946 ms
2021-12-05 15:51:19 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 3999992 Strings with 1 runs
Completed in 13187.5507 ms
2021-12-05 15:51:51 INFO  Benchmark_Timer - Begin run: Dual Pivot Quick Sort: 1 for 7999984 Strings with 1 runs
Completed in 27294.7469 ms

MSD Radix Sort Benchmark

2021-12-05 15:52:45 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 253760 Strings with 1 runs
Completed in 1871.6247 ms
2021-12-05 15:52:49 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 507520 Strings with 1 runs
Completed in 2770.0022 ms
2021-12-05 15:52:55 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 999998 Strings with 1 runs
Completed in 119077.2422 ms
2021-12-05 15:56:55 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 1999996 Strings with 1 runs
Completed in 268066.9396 ms
2021-12-05 16:05:54 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 3999992 Strings with 1 runs
Completed in 274020.7003 ms
2021-12-05 16:15:10 INFO  Benchmark_Timer - Begin run: MSD Radix Sort: 1 for 7999984 Strings with 1 runs
Completed in 285466.6374 ms
```

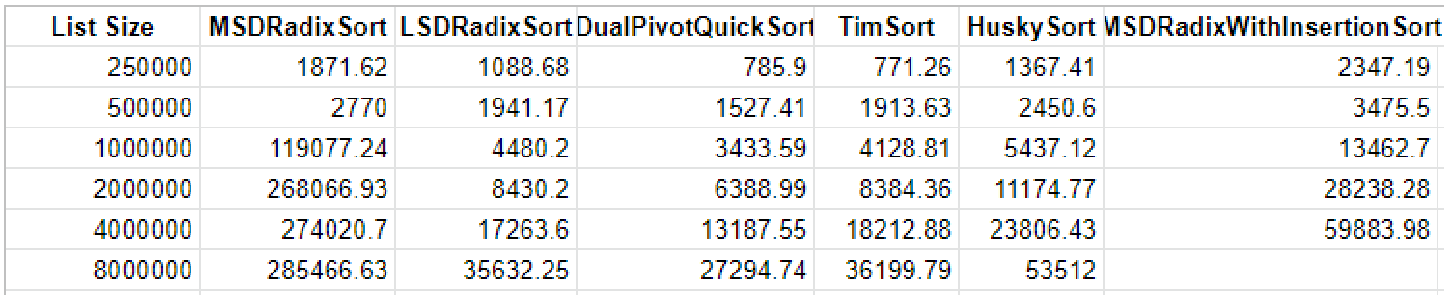


```
Husky Sort Benchmark

2021-12-05 17:33:46 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 253760 Strings with 1 runs
Completed in 1351.4434 ms
2021-12-05 17:33:49 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 507520 Strings with 1 runs
Completed in 2582.9333 ms
2021-12-05 17:33:55 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 999998 Strings with 1 runs
Completed in 5286.9656 ms
2021-12-05 17:34:07 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 1999996 Strings with 1 runs
Completed in 10136.8809 ms
2021-12-05 17:34:30 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 3999992 Strings with 1 runs
Completed in 20350.4188 ms
2021-12-05 17:35:17 INFO  Benchmark_Timer - Begin run: Husky Sort: 1 for 7999984 Strings with 1 runs
Completed in 44166.2954 ms

Process finished with exit code 0
```

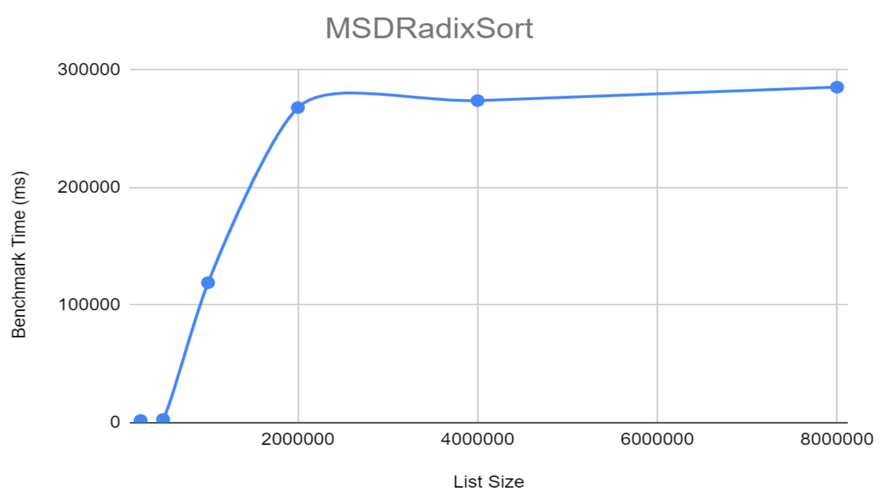


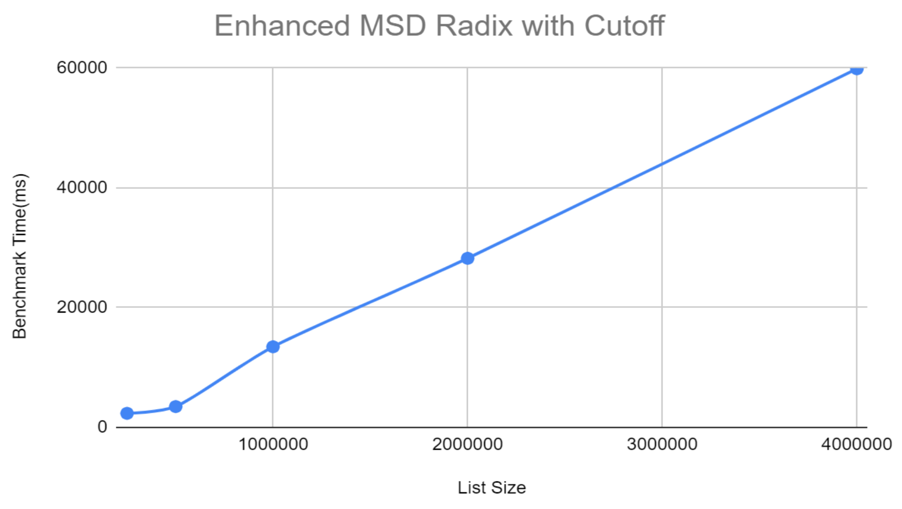
```
LSD Radix Sort Benchmark

2021-12-05 16:24:45 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 253760 Strings with 1 runs
Completed in 1088.6847 ms
2021-12-05 16:24:48 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 507520 Strings with 1 runs
Completed in 1941.1791 ms
2021-12-05 16:24:52 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 999998 Strings with 1 runs
Completed in 4480.2063 ms
2021-12-05 16:25:02 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 1999996 Strings with 1 runs
Completed in 8430.203 ms
2021-12-05 16:25:22 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 3999992 Strings with 1 runs
Completed in 17263.6091 ms
2021-12-05 16:26:02 INFO  Benchmark_Timer - Begin run: LSD Radix Sort: 1 for 7999984 Strings with 1 runs
Completed in 35632.2585 ms

Tim Sort Benchmark

2021-12-05 16:27:14 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 253760 Strings with 1 runs
Completed in 771.2615 ms
2021-12-05 16:27:16 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 507520 Strings with 1 runs
Completed in 1913.6347 ms
2021-12-05 16:27:21 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 999998 Strings with 1 runs
Completed in 4128.8121 ms
2021-12-05 16:27:30 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 1999996 Strings with 1 runs
Completed in 8384.3676 ms
2021-12-05 16:27:51 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 3999992 Strings with 1 runs
Completed in 18212.8812 ms
2021-12-05 16:28:33 INFO  Benchmark_Timer - Begin run: Tim Sort: 1 for 7999984 Strings with 1 runs
Completed in 36199.7933 ms
```

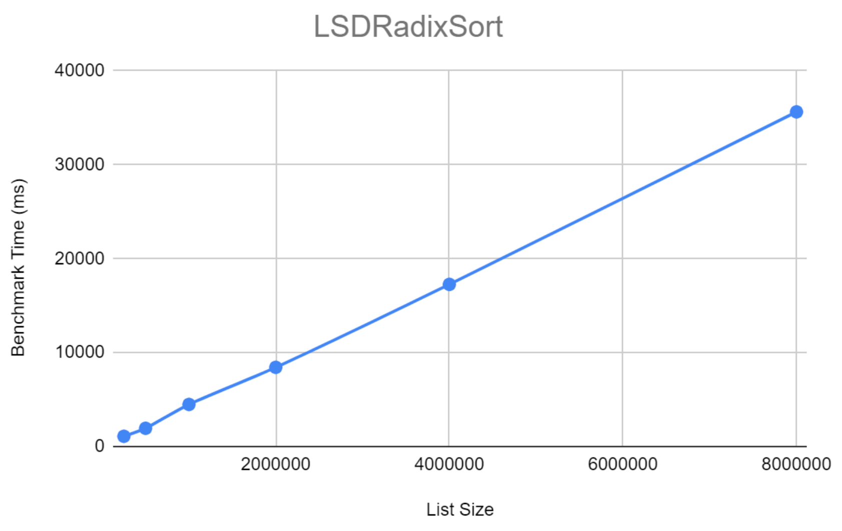
Benchmark times:

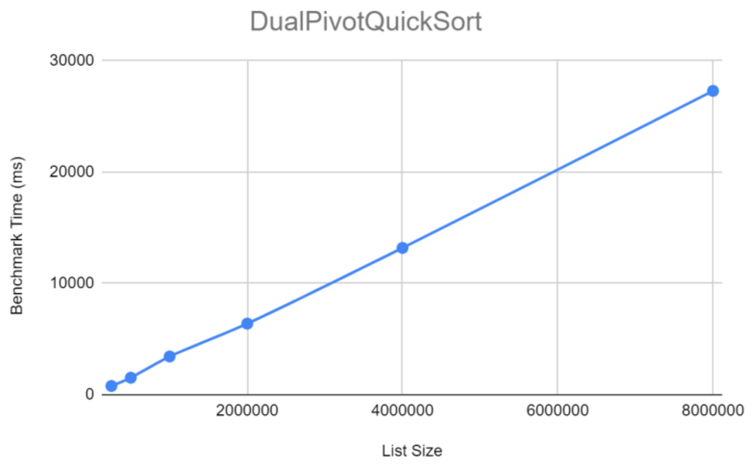


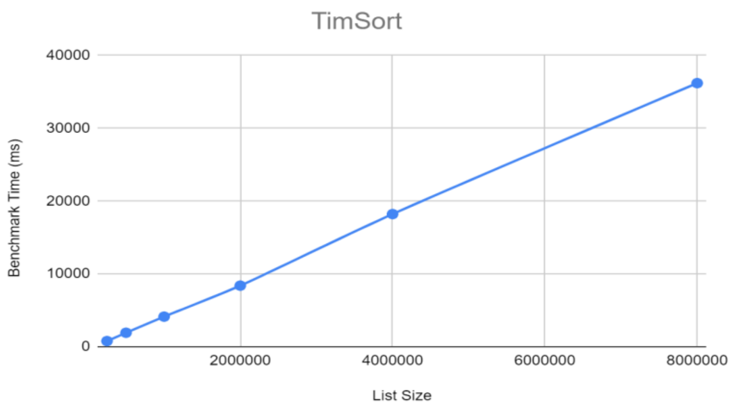
List Size	MSDRadixSort	LSDRadixSort	DualPivotQuickSor	TimSort	HuskySort	MSDRadixWithInsertionSort
250000	1871.62	1088.68	785.9	771.26	1367.41	2347.19
500000	2770	1941.17	1527.41	1913.63	2450.6	3475.5
1000000	119077.24	4480.2	3433.59	4128.81	5437.12	13462.7
2000000	268066.93	8430.2	6388.99	8384.36	11174.77	28238.28
4000000	274020.7	17263.6	13187.55	18212.88	23806.43	59883.98
8000000	285466.63	35632.25	27294.74	36199.79	53512	

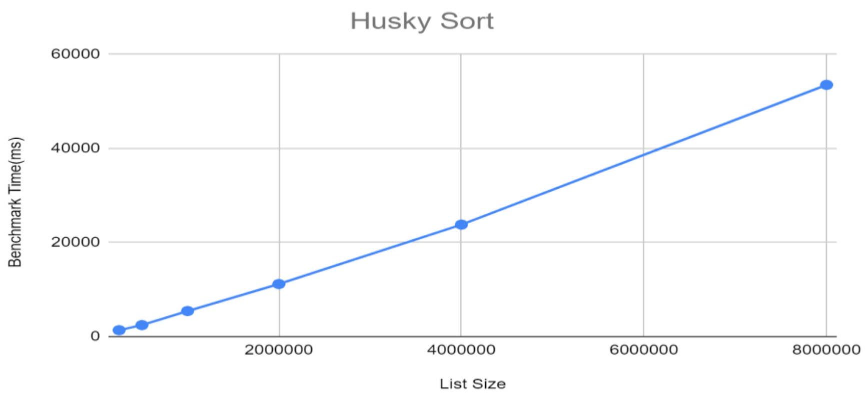

MSDRadixSort
300000
200000
100000
0
Benchmark Time (ms)
2000000
4000000
6000000
8000000
List Size


Enhanced MSD Radix with Cutoff
60000
40000
20000
0
Benchmark Time(ms)
1000000
2000000
3000000
4000000
List Size


LSDRadixSort
40000
30000
20000
10000
0
Benchmark Time (ms)
2000000
4000000
6000000
8000000
List Size







Conclusion:

The task was to implement MSD Radix Sort for comparing Chinese words and names in pinyin order, and to compare it with other sorting algorithms. In order to accomplish this, Dual Pivot Quicksort, Husky Sort, TimSort, MSD Radix Sort and LSD Radix sort were implemented.

There are instances of different Chinese characters having the same pinyin spelling (and hence the same precedence). Hence, sorting using these different algorithms produced different resultant arrays in Chinese, but their pinyin equivalents were the same. Hence, the resultant Chinese arrays were converted to pinyin to perform accuracy checks.
As per the results posted above, MSD Radix Sort performs really poorly for smaller arrays, but starts to level off as the array size reaches the 2-4 million mark. However, if we were to implement a subarray cutoff for MSD Radix sort below which insertion sort is used, performance drastically improves as shown in the screenshots.
The other sorting algorithms performed better than MSD Radix sort for the most part, with Dual Pivot Quicksort being the fastest overall. The other sorts seem to be around the same range as the DPQS, including MSD Radix with insertion sort optimization.

Credits: In addition to links provided in each file

- Prof. Robin Hillyard and contributors of the Info6205 assignments & huskysort repositories (https://github.com/rchillyard/INFO6205) https://github.com/rchillyard/The-repository-formerly-known-as)
- pinyin4j (http://pinyin4j.sourceforge.net/)
- log4j (https://logging.apache.org/log4j/2.x/)
- ini4j (http://ini4j.sourceforge.net/)
- stackoverflow and geeksforgeeks community