

Program Structure and Algorithms (INFO 6205)

Fall – 2021 Final Project

Report

Keyur Ashokbhai Barot – 001568664

I. Tasks:

Following tasks were performed in the project:

1. Implement MSD radix sort for a natural language which uses Unicode characters.
2. Sort Simplified Chinese words and compare the results of MSD radix with Timsort, Dual Pivot Quicksort, Huskysort and LSD radix sort.

II. Results:

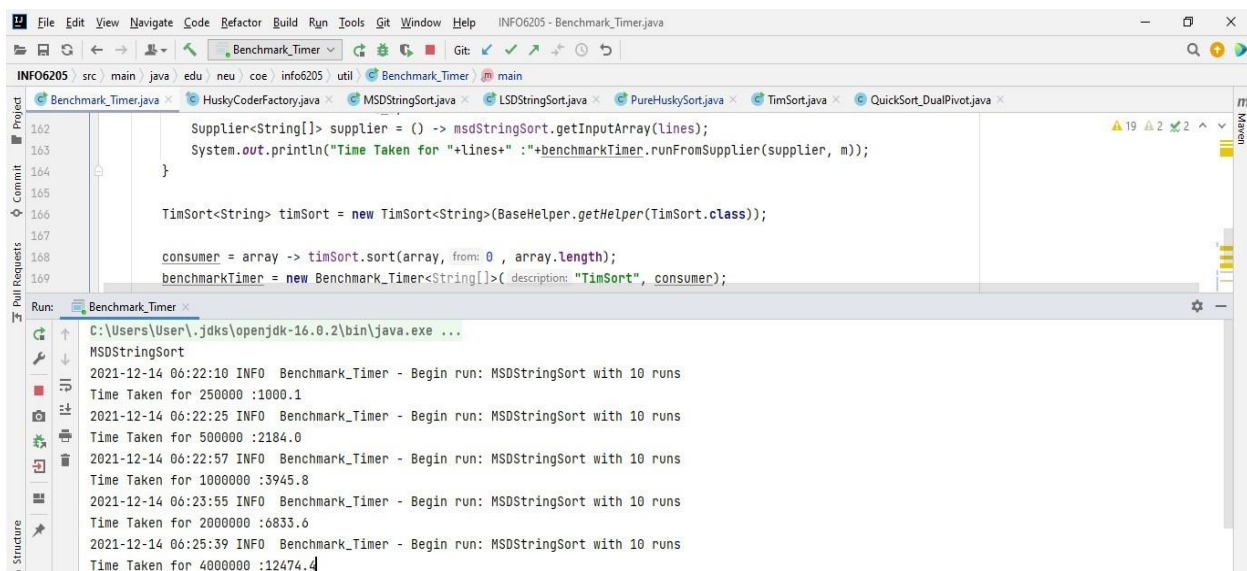
The code for the benchmarking can be found here:

https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/util/Benchmark_Timer.java

1. MSD Radix Sort:

<https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/sort/counting/MSDStringSort.java>

The benchmarking results for the MSD radix sort are as follows:



```
Supplier<String[]> supplier = () -> msdStringSort.getInputArray(lines);
System.out.println("Time Taken for "+lines+" :"+benchmarkTimer.runFromSupplier(supplier, m));
}

TimSort<String> timSort = new TimSort<String>(BaseHelper.getHelper(TimSort.class));

consumer = array -> timSort.sort(array, from: 0, array.length);
benchmarkTimer = new Benchmark_Timer<String[]>("TimSort", consumer);
```

```
Run: Benchmark_Timer
C:\Users\User\jdk\openjdk-16.0.2\bin\java.exe ...
MSDStringSort
2021-12-14 06:22:10 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 250000 :1000.1
2021-12-14 06:22:25 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 500000 :2184.0
2021-12-14 06:22:57 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 1000000 :3945.8
2021-12-14 06:23:55 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 2000000 :6833.6
2021-12-14 06:25:39 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 4000000 :12474.4
```

Algorithm	Number of words	Time(ms)
MSDStringSort	250000	1000.1
MSDStringSort	500000	2184
MSDStringSort	1000000	3945.8
MSDStringSort	2000000	6833.6
MSDStringSort	4000000	12474.4

2. Timsort:

<https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/sort/linearithmic/TimSort.java>

The benchmarking results for Timsort are as follows:

The screenshot shows an IDE with the following code in `Benchmark_Timer.java`:

```

Supplier<String[]> supplier = () -> msdStringSort.getInputArray(lines);
System.out.println("Time Taken for "+lines+" :"+benchmarkTimer.runFromSupplier(supplier, m));
}

TimSort<String> timSort = new TimSort<String>(BaseHelper.getHelper(TimSort.class));

consumer = array -> timSort.sort(array, from: 0 , array.length);
benchmarkTimer = new Benchmark_Timer<String[]>("TimSort", consumer);

```

The Run console output shows the following results:

```

Time Taken for 2000000 :6833.6
2021-12-14 06:25:39 INFO Benchmark_Timer - Begin run: MSDStringSort with 10 runs
Time Taken for 4000000 :12474.4
TimSort
2021-12-14 06:28:52 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 250000 :191.0
2021-12-14 06:28:57 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 500000 :388.7
2021-12-14 06:29:07 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 1000000 :794.8
2021-12-14 06:29:28 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 2000000 :1604.6
2021-12-14 06:30:09 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 4000000 :3413.8

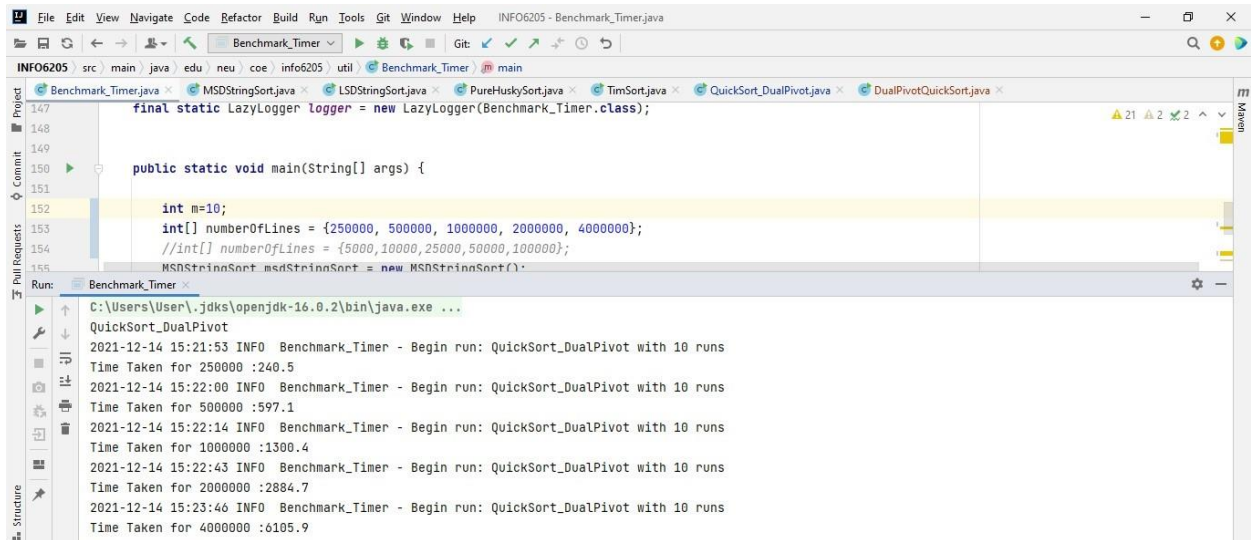
```

Algorithm	Number of words	Time(ms)
TimSort	250000	191
TimSort	500000	388.7
TimSort	1000000	794.8
TimSort	2000000	1604.6
TimSort	4000000	3413.8

3. Dual-pivot Quicksort:

<https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/sort/DualPivotQuickSort.java>

The benchmarking results for Dual-pivot Quicksort are as follows:



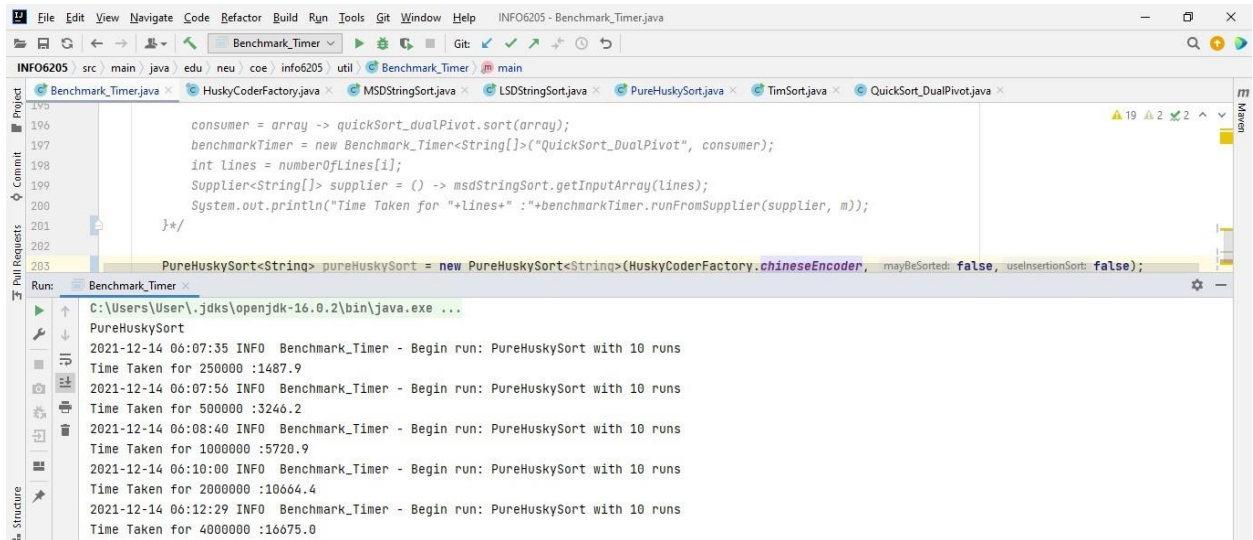
```
147 final static LazyLogger logger = new LazyLogger(Benchmark_Timer.class);
148
149
150 public static void main(String[] args) {
151
152     int m=10;
153     int[] numberOfLines = {250000, 500000, 1000000, 2000000, 4000000};
154     //int[] numberOfLines = {5000,10000,25000,50000,100000};
155     MSDStringSort msdSteinnSort = new MSDStringSort();
156
157     Run: Benchmark_Timer
158     C:\Users\User\jdk\openjdk-16.0.2\bin\java.exe ...
159     QuickSort_DualPivot
160     2021-12-14 15:21:53 INFO Benchmark_Timer - Begin run: QuickSort_DualPivot with 10 runs
161     Time Taken for 250000 :240.5
162     2021-12-14 15:22:00 INFO Benchmark_Timer - Begin run: QuickSort_DualPivot with 10 runs
163     Time Taken for 500000 :597.1
164     2021-12-14 15:22:14 INFO Benchmark_Timer - Begin run: QuickSort_DualPivot with 10 runs
165     Time Taken for 1000000 :1300.4
166     2021-12-14 15:22:43 INFO Benchmark_Timer - Begin run: QuickSort_DualPivot with 10 runs
167     Time Taken for 2000000 :2884.7
168     2021-12-14 15:23:46 INFO Benchmark_Timer - Begin run: QuickSort_DualPivot with 10 runs
169     Time Taken for 4000000 :6105.9
```

Algorithm	Number of words	Time(ms)
QuickSort_DualPivot	250000	240.5
QuickSort_DualPivot	500000	597.1
QuickSort_DualPivot	1000000	1300.4
QuickSort_DualPivot	2000000	2884.7
QuickSort_DualPivot	4000000	6105.9

4. Huskysort:

<https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/sort/huskySort/PureHuskySort.java>

The benchmarking results for Huskysort are as follows:



```
consumer = array -> quickSort_dualPivot.sort(array);
benchmarkTimer = new Benchmark_Timer<String[]>("QuickSort_DualPivot", consumer);
int lines = numberOfLines[i];
Supplier<String[]> supplier = () -> msdStringSort.getInputArray(lines);
System.out.println("Time Taken for "+lines+" :"+benchmarkTimer.runFromSupplier(supplier, m));
}*/

PureHuskySort<String> pureHuskySort = new PureHuskySort<String>(HuskyCoderFactory.chineseEncoder, mayBeSorted: false, useInsertionSort: false);

C:\Users\User\jdk-16.0.2\bin\java.exe ...
PureHuskySort
2021-12-14 06:07:35 INFO Benchmark_Timer - Begin run: PureHuskySort with 10 runs
Time Taken for 250000 :1487.9
2021-12-14 06:07:56 INFO Benchmark_Timer - Begin run: PureHuskySort with 10 runs
Time Taken for 500000 :3246.2
2021-12-14 06:08:40 INFO Benchmark_Timer - Begin run: PureHuskySort with 10 runs
Time Taken for 1000000 :5720.9
2021-12-14 06:10:00 INFO Benchmark_Timer - Begin run: PureHuskySort with 10 runs
Time Taken for 2000000 :10664.4
2021-12-14 06:12:29 INFO Benchmark_Timer - Begin run: PureHuskySort with 10 runs
Time Taken for 4000000 :16675.0
```

Algorithm	Number of words	Time(ms)
PureHuskySort	250000	1487.9
PureHuskySort	500000	3246.2
PureHuskySort	1000000	5720.9
PureHuskySort	2000000	10664.4
PureHuskySort	4000000	16675

5. LSD radix sort:

<https://github.com/KeyurAshokbhaiBarot/INFO6205/blob/Fall2021/src/main/java/edu/neu/coe/info6205/sort/counting/LSDStringSort.java>

The benchmarking results of LSD radix sort are as follows:

```
Supplier<String[]> supplier = () -> msdStringSort.getInputArray(lines);
System.out.println("Time Taken for "+lines+" :"+benchmarkTimer.runFromSupplier(supplier, m));

TimSort<String> timSort = new TimSort<String>(BaseHelper.getHelper(TimSort.class));

consumer = array -> timSort.sort(array, from: 0 , array.length);
benchmarkTimer = new Benchmark_Timer<String[]>(description: "TimSort", consumer);
```

Run: Benchmark_Timer

```
Time Taken for 2000000 :1004.0
2021-12-14 06:30:09 INFO Benchmark_Timer - Begin run: TimSort with 10 runs
Time Taken for 4000000 :3413.8
LSDStringSort
2021-12-14 06:31:35 INFO Benchmark_Timer - Begin run: LSDStringSort with 10 runs
Time Taken for 250000 :342.2
2021-12-14 06:31:42 INFO Benchmark_Timer - Begin run: LSDStringSort with 10 runs
Time Taken for 500000 :538.4
2021-12-14 06:31:55 INFO Benchmark_Timer - Begin run: LSDStringSort with 10 runs
Time Taken for 1000000 :820.9
2021-12-14 06:32:19 INFO Benchmark_Timer - Begin run: LSDStringSort with 10 runs
Time Taken for 2000000 :1666.3
2021-12-14 06:33:05 INFO Benchmark_Timer - Begin run: LSDStringSort with 10 runs
Time Taken for 4000000 :2930.5
```

Algorithm	Number of words	Time(ms)
LSDStringSort	250000	342.2
LSDStringSort	500000	538.4
LSDStringSort	1000000	820.9
LSDStringSort	2000000	1666.3
LSDStringSort	4000000	2930.5

III. Conclusion:

After implementing all the five given algorithms we can say that Timsort takes the least time to sort, followed by LSD radix sort, which is followed by Dual-Pivot Quicksort, MSD radix sort and Huskysort respectively.

Evidence for the same can be seen in the following time comparisons of each algorithm.

IV. Evidence:

