

Abhishek Ravindra Satbhai (001587689)

Apoorva Kethamaranahalli(001090928)

Sneha Ravichandran(001096251)

Program Structures & Algorithms

Fall 2021

Final Project

◉ Task

1. Implement MSD radix sort for sorting name list of lang which uses Unicode character (find relevant paper to implement it)
2. Compare your method with all other sorts: LSD Radix Sort, Dual Pivot Quick Sort and Husky Sort
3. Write unit test cases for your code
4. Use sort benchmark class and benchmark the results of all methods for 250k, 500k, 1M, 2M, 4M
5. Make graphical representation to show your results

◉ Implementation

In order to use the best implementation methods to conclude which algorithm works best for Unicode sorting, we ran benchmarks for LSD Radix Sort, MSD Radix Sort and Dual Pivot Quick Sort.

Implementation Idea 1: HashMap approach

1. Read the input from a file, convert to the pinyin format using the pinyin4j library
2. Maintain a HashMap of key being the pinyin string representation of the Chinese name, and the value being a list of Chinese names that corresponds to the respective pinyin representation. This HashMap serves as a lookup, when we have to write the sorted names back into an output file.
3. Run the sort methods, for each of the LSD Radix Sort, MSD Radix Sort, Husky Sort as well as Dual Pivot Quick Sort, with the input for all of them being an array of pinyin representation of Chinese names.
4. Write the output into an external file, by looking up the Chinese equivalent of the sorted pinyin array of names.

Implementation Idea 2: External Class approach

1. Read the input from a file, convert to the pinyin format using the pinyin4j library
2. Have an external class, consisting of two members: the pinyin representation of the Chinese name, and the Chinese name itself.
3. Run the sort methods, for each of the LSD Radix Sort, MSD Radix Sort, Husky Sort as well as Dual Pivot Quick Sort, with the input for all of them being an array of Objects of type external class that implements a comparable interface.
4. Write the output into an external file using the getter method implemented in the external class.

◉ Relationship Conclusion:

1. After running benchmarking for both our implementation approaches, on average above an input length of 500000, we conclude that the HashMap would be an ideal approach, firstly because of the time it takes (according to benchmarking results), but considering the extra space it takes, if there are use cases where space might be an important factor, we would choose to implement the External Class approach instead.
2. The conclusion of running benchmarks for varying input lengths can be given as:
MSD Radix Sort > Dual Pivot Quick Sort > Husky Sort > LSD Radix Sort for inputs less than 2.5 million
3. After around 3.5 million, both MSD Radix Sort and Husky Sort perform almost the same, with Dual Pivot Quick Sort coming after these two, and LSD Radix Sort having the least performance among them all.
4. All the sorts run linearly with respect to the input given. As the size of the input given increases, so does the time taken to sort them in a linear manner.

◎ Output:

1. Output of Benchmarking

Benchmarking results: LSD Radix Sort, MSD Radix Sort, Dual Pivot Quick Sort with HashMap approach for input length varying from 250000 to 4 Million and runs =20

Tests	Structure	Sorting Algo : LSDPair , Elapsed Time in milli seconds to run N : 250000 , Time : 3068 ms
		Sorting Algo : LSDPair , Elapsed Time in milli seconds to run N : 500000 , Time : 6039 ms
		Sorting Algo : LSDPair , Elapsed Time in milli seconds to run N : 999998 , Time : 10866 ms
		Sorting Algo : LSDPair , Elapsed Time in milli seconds to run N : 1999996 , Time : 25484 ms
		Sorting Algo : LSDPair , Elapsed Time in milli seconds to run N : 3999992 , Time : 52770 ms
		Sorting Algo : MSDPair , Elapsed Time in milli seconds to run N : 250000 , Time : 1394 ms
		Sorting Algo : MSDPair , Elapsed Time in milli seconds to run N : 500000 , Time : 3602 ms
		Sorting Algo : MSDPair , Elapsed Time in milli seconds to run N : 999998 , Time : 9551 ms
		Sorting Algo : MSDPair , Elapsed Time in milli seconds to run N : 1999996 , Time : 16324 ms
		Sorting Algo : MSDPair , Elapsed Time in milli seconds to run N : 3999992 , Time : 34265 ms
		Sorting Algo : DPQPair , Elapsed Time in milli seconds to run N : 250000 , Time : 1698 ms
		Sorting Algo : DPQPair , Elapsed Time in milli seconds to run N : 500000 , Time : 3119 ms
		Sorting Algo : DPQPair , Elapsed Time in milli seconds to run N : 999998 , Time : 5483 ms
		Sorting Algo : DPQPair , Elapsed Time in milli seconds to run N : 1999996 , Time : 16781 ms
		Sorting Algo : DPQPair , Elapsed Time in milli seconds to run N : 3999992 , Time : 30004 ms

Benchmarking results: LSD Radix Sort, MSD Radix Sort, Dual Pivot Quick Sort with External Class approach for input length varying from 250000 to 4 Million and runs = 20

```
Run: MSDRadixSortTest x Timer x
/Users/abhisheksatbhai/Library/Java/JavaVirtualMachines/corretto-1.8.0_312/Contents/Home/bin/java ...
Sorting Algo : LSDString , Elapsed Time in milli seconds to run N : 250000 , Time : 1477 ms
Sorting Algo : LSDString , Elapsed Time in milli seconds to run N : 500000 , Time : 2581 ms
Sorting Algo : LSDString , Elapsed Time in milli seconds to run N : 999998 , Time : 6601 ms
Sorting Algo : LSDString , Elapsed Time in milli seconds to run N : 1999996 , Time : 10078 ms
Sorting Algo : LSDString , Elapsed Time in milli seconds to run N : 3999992 , Time : 22164 ms
Sorting Algo : MSDString , Elapsed Time in milli seconds to run N : 250000 , Time : 864 ms
Sorting Algo : MSDString , Elapsed Time in milli seconds to run N : 500000 , Time : 1428 ms
Sorting Algo : MSDString , Elapsed Time in milli seconds to run N : 999998 , Time : 2738 ms
Sorting Algo : MSDString , Elapsed Time in milli seconds to run N : 1999996 , Time : 7220 ms
Sorting Algo : MSDString , Elapsed Time in milli seconds to run N : 3999992 , Time : 14004 ms
Sorting Algo : DPQString , Elapsed Time in milli seconds to run N : 250000 , Time : 690 ms
Sorting Algo : DPQString , Elapsed Time in milli seconds to run N : 500000 , Time : 1227 ms
Sorting Algo : DPQString , Elapsed Time in milli seconds to run N : 999998 , Time : 2719 ms
Sorting Algo : DPQString , Elapsed Time in milli seconds to run N : 1999996 , Time : 8035 ms
Sorting Algo : DPQString , Elapsed Time in milli seconds to run N : 3999992 , Time : 16885 ms
```

Benchmarking results: Husky sort with HashMap approach with runs =20

a) input length = 250000

```
PureHuskySort x
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=49685:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
Sorting Algo : HuskySortString , Elapsed Time in milli seconds to run N : 250000 , Time : 2405 ms
2021-12-05 23:21:33 INFO PureHuskySort - PureHuskySort.main: finished
Process finished with exit code 0
```

b) input length = 500000

```
PureHuskySort
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=49728:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
Sorting Algo : HuskySortString , Elapsed Time in milli seconds to run N : 500000 , Time : 3599 ms
2021-12-05 23:23:06 INFO PureHuskySort - PureHuskySort.main: finished
Process finished with exit code 0
```

c) input length = 1000000

```
Run: PureHuskySort x
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=49746:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
Sorting Algo : HuskySortString , Elapsed Time in milli seconds to run N : 1000000 , Time : 5270 ms
2021-12-05 23:24:51 INFO PureHuskySort - PureHuskySort.main: finished
Process finished with exit code 0
```

d) input length = 2000000

```

PureHuskySort
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=61951:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
Sorting Algo : HuskySortString , Elapsed Time in milli seconds to run N : 2000000 , Time : 9472 ms
2021-12-05 23:31:10 INFO PureHuskySort - PureHuskySort.main: finished
Process finished with exit code 0

```

e) input length = 4000000

```

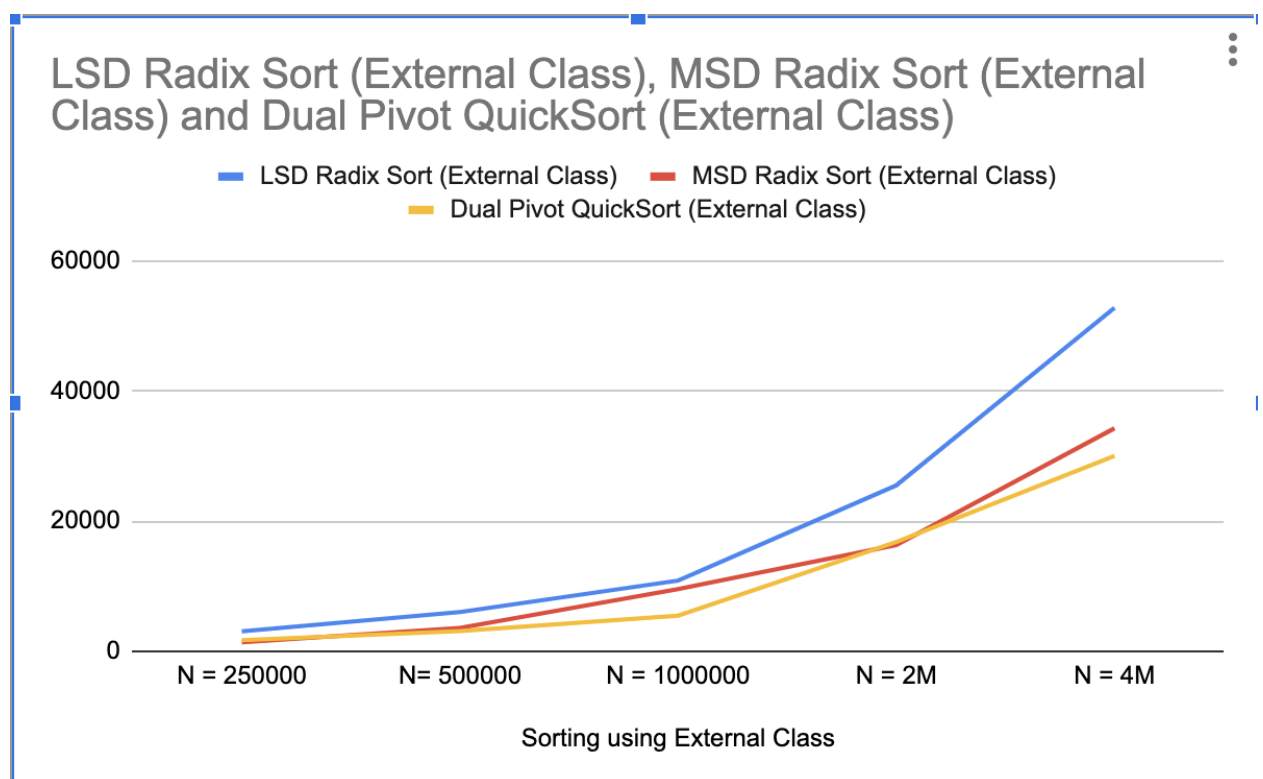
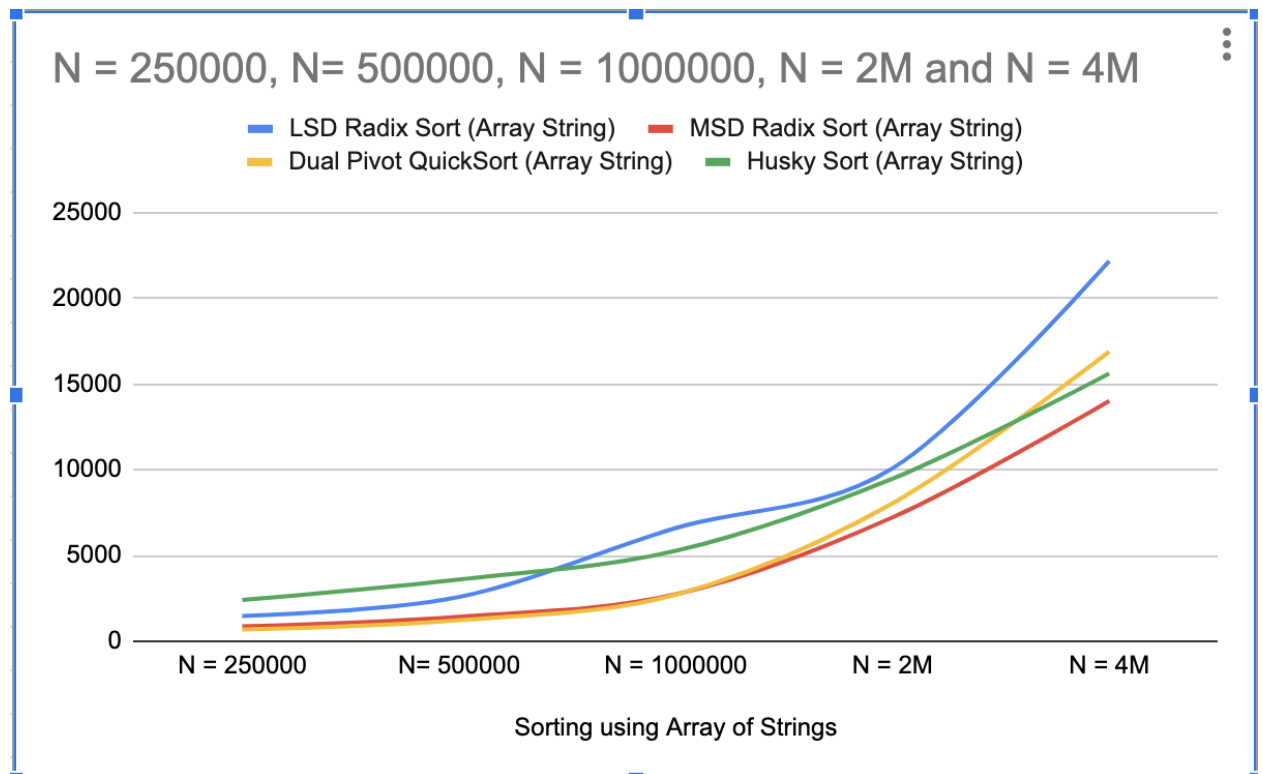
Run: PureHuskySort
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\lib\idea_rt.jar=61984:C:\Program Files\JetBrains\IntelliJ IDEA 2021.2.1\bin" -Dfile.encoding=UTF-8
Sorting Algo : HuskySortString , Elapsed Time in milli seconds to run N : 4000000 , Time : 15681 ms
2021-12-05 23:33:50 INFO PureHuskySort - PureHuskySort.main: finished
Process finished with exit code 0

```

2. Graphical Representation :

Below are the given benchmarking results for both the approaches detailed in the Implementation

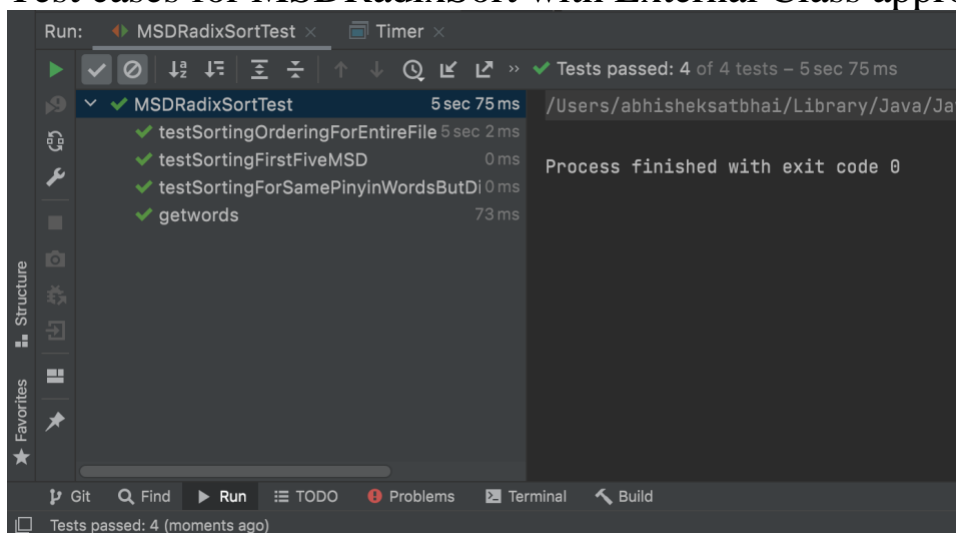
Sorting using Array of Strings	N = 250000	N= 500000	N = 1000000	N = 2M	N = 4M
LSD Radix Sort (Array String)	1477	2581	6601	10078	22164
MSD Radix Sort (Array String)	864	1428	2738	7220	14004
Dual Pivot QuickSort (Array String)	690	1227	2719	8035	16885
Husky Sort (Array String)	2405	3599	5270	9472	15601
Sorting using External Class	N = 250000	N= 500000	N = 1000000	N = 2M	N = 4M
LSD Radix Sort (External Class)	3068	6039	10866	25484	52770
MSD Radix Sort (External Class)	1394	3602	9551	16324	34265
Dual Pivot QuickSort (External Class)	1698	3119	5483	16781	30004



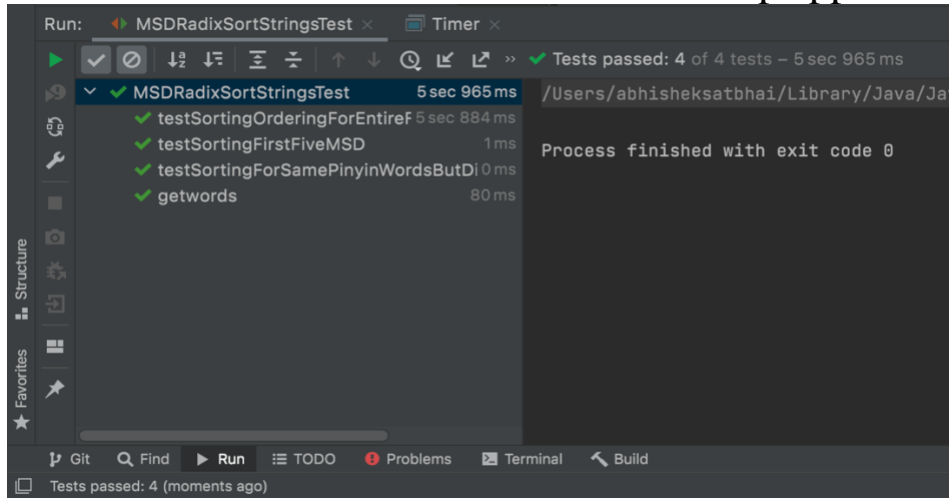
◉ Test Cases

1. For running test cases, we check if our input gets sorted according to the Arrays.sort method that java gives us, for the converted shuffled pinyin equivalents for the Chinese names *

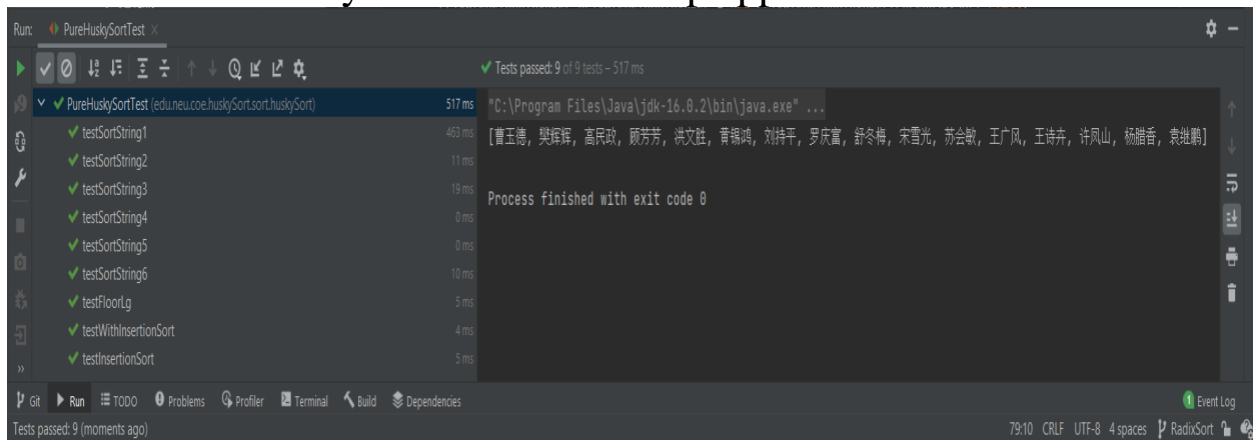
Test cases for MSDRadixSort with External Class approach



Test cases for MSDRadixSort with HashMap approach



Test cases for Husky Sort with HashMap approach:



*repository contains detailed test cases for further review.