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:**

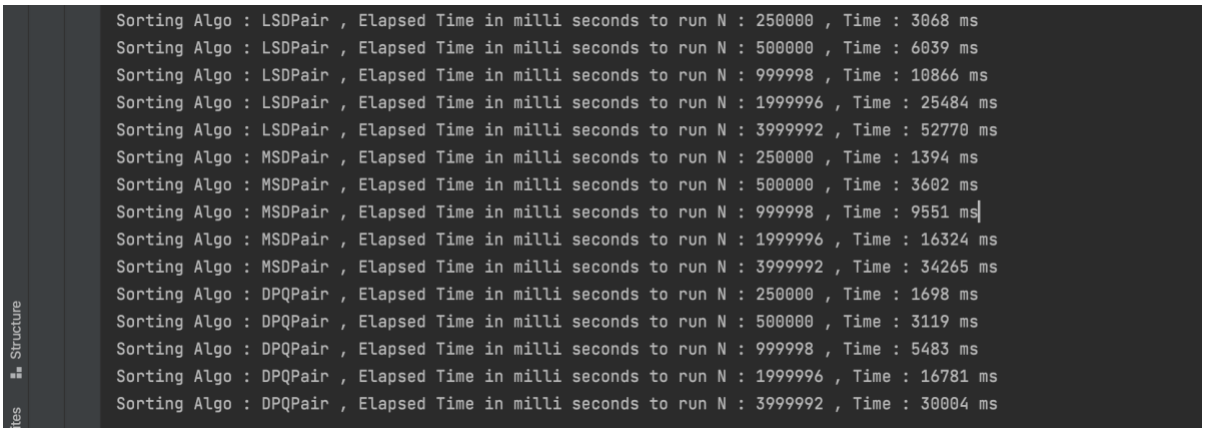
1. After running benchmarking for both our implementation approaches, on average above an input length of 500000, we conclude that the HashMap would be an ideal approach, firstly because of the time it takes (according to benchmarking results), but considering the extra space it takes, if there are use 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.

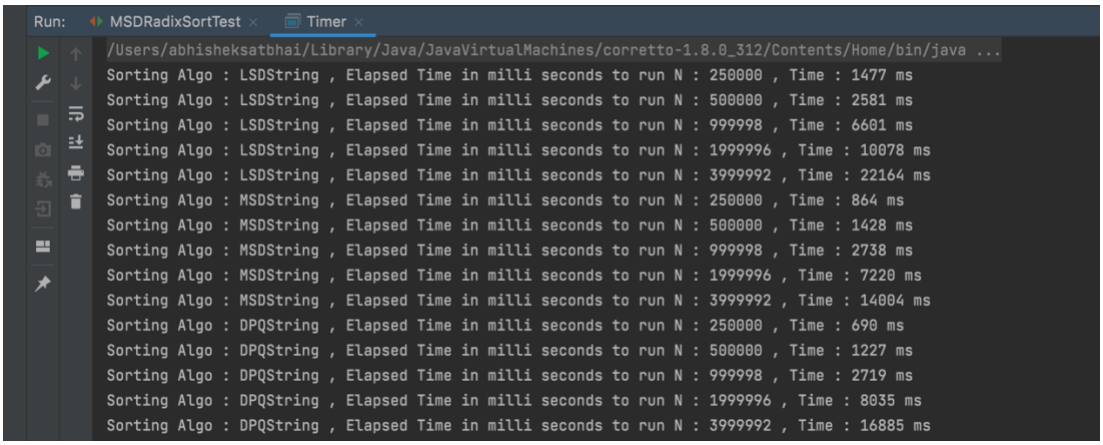
- **Output:**

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

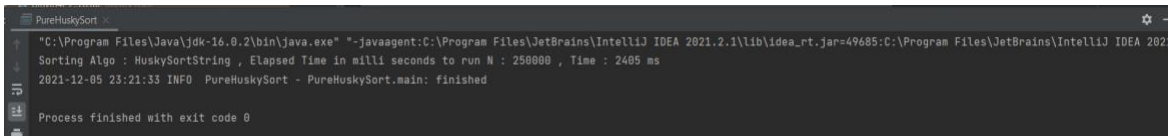


Benchmarking results: LSD Radix Sort, MSD Radix Sort, Dual Pivot Quick Sort with External Class approach for input length varying from 250000 to 4 Million and runs = 20

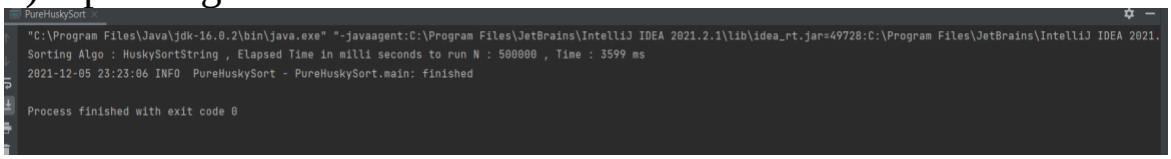


Benchmarking results: Husky sort with HashMap approach with runs =20

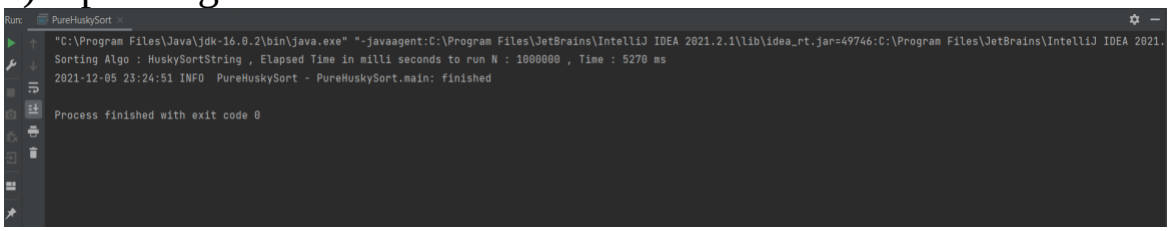
a) input length = 250000



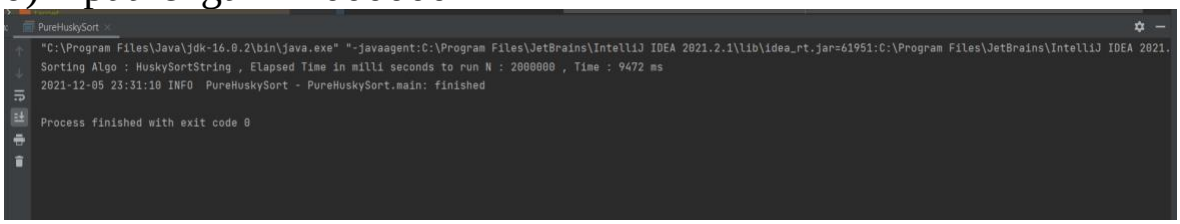
b) input length = 500000



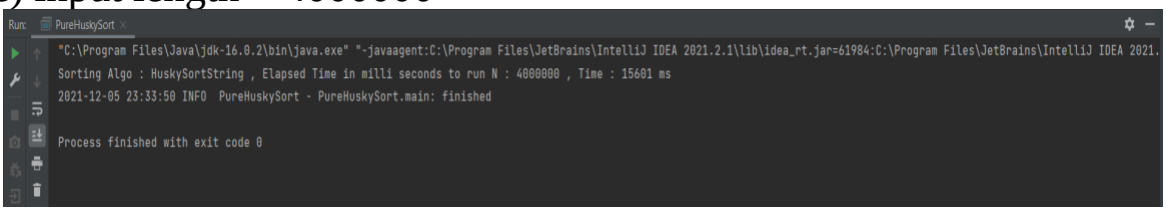
c) input length = 1000000



d) input length = 2000000



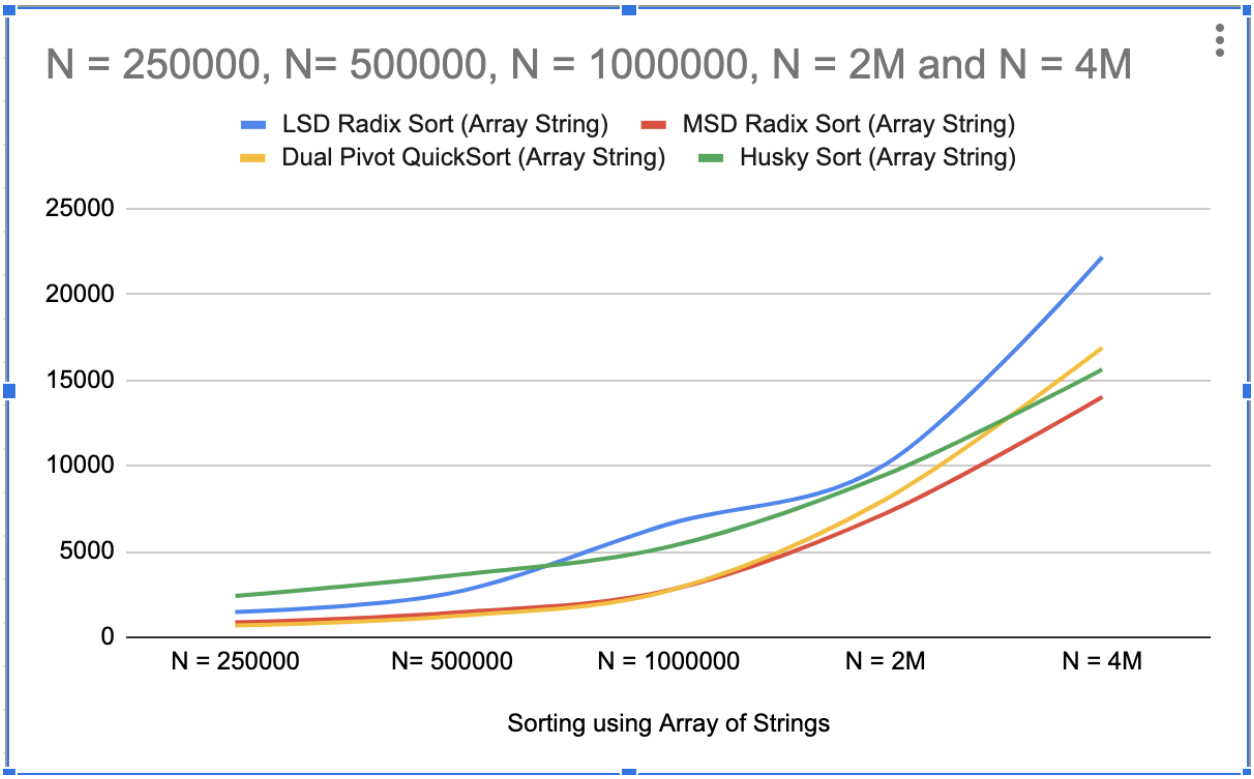
e) input length = 4000000

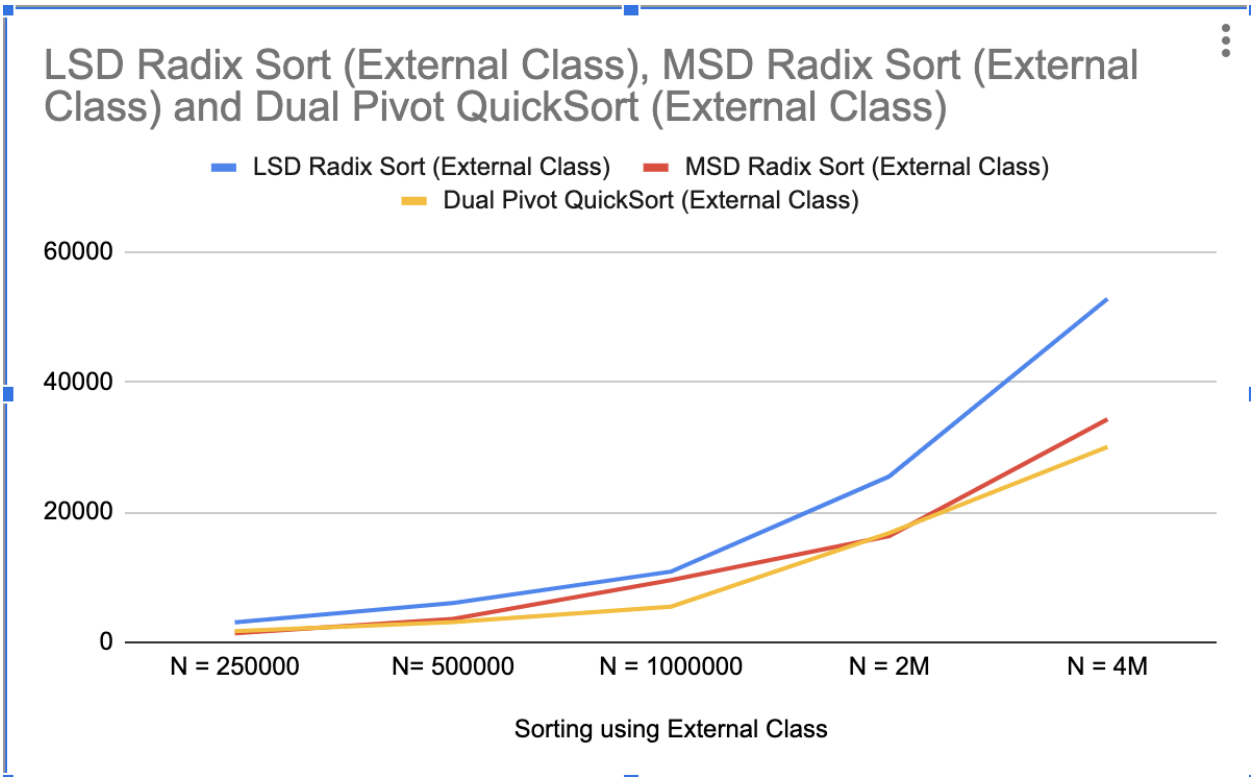


2. **Graphical Representation :**

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

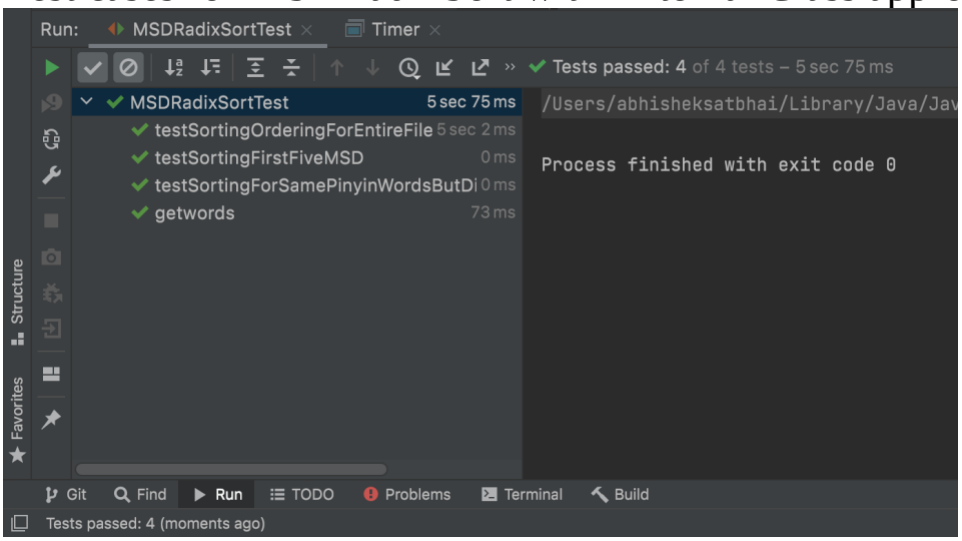

N = 250000, N= 500000, N = 1000000, N = 2M and N = 4M
LSD Radix Sort (Array String)
MSD Radix Sort (Array String)
Dual Pivot QuickSort (Array String)
Husky Sort (Array String)
25000
20000
15000
10000
5000
0
N = 250000
N= 500000
N = 1000000
N = 2M
N = 4M
Sorting using Array of Strings


LSD Radix Sort (External Class), MSD Radix Sort (External Class) and Dual Pivot QuickSort (External Class)
LSD Radix Sort (External Class)
MSD Radix Sort (External Class)
Dual Pivot QuickSort (External Class)
60000
40000
20000
0
N = 250000
N= 500000
N = 1000000
N = 2M
N = 4M
Sorting using External Class

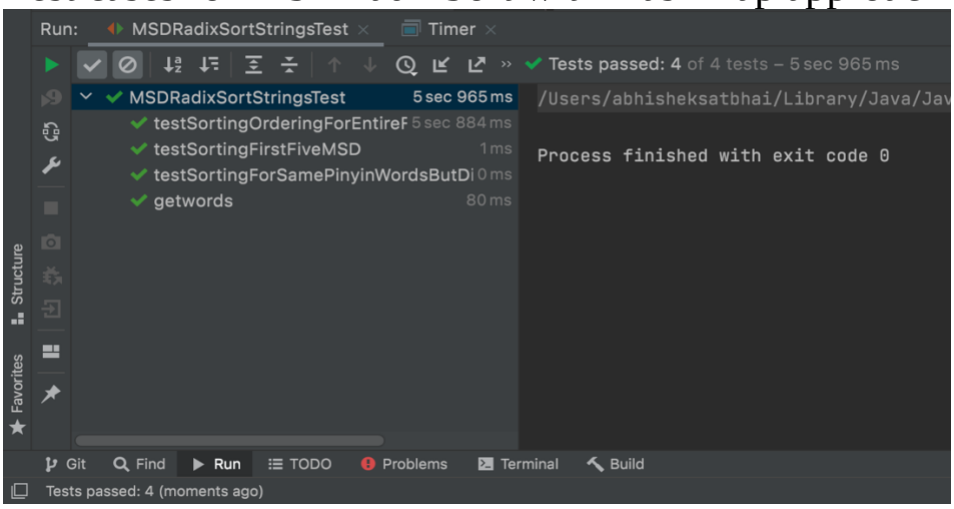
- **Test Cases**

  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 *

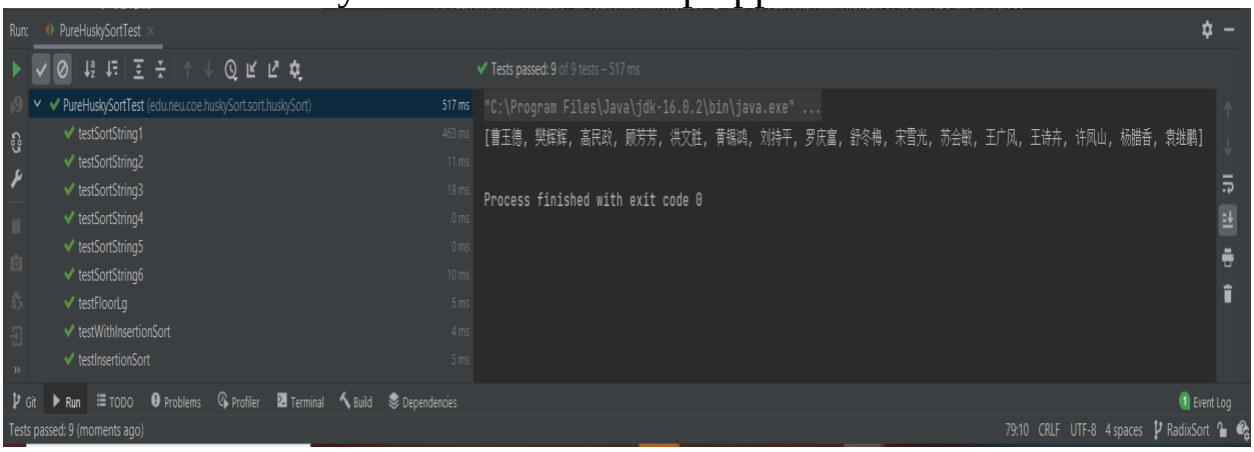
Test cases for MSDRadixSort with External Class approach



Test cases for MSDRadixSort with HashMap approach



Test cases for Husky Sort with HashMap approach:



*repository contains detailed test cases for further review.