INFO 6205 Program Structures and Algorithms - Fall 2021

Team Project

—

Vikas Venkataraman Shastry - (001582447)

Thejas Ganjigunte Ramesh - (001586113)

Nithya Viswanathan - (001046065)

Objective

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.

Goals

1. Implement MSD Radix Sort that is capable of sorting a natural language.
2. Modify Timsort, Dual-pivot Quicksort, Huskysort, and LSD radix sort to be capable of sorting the chosen natural language.
3. Benchmark all the above mentioned sorting algorithms and compare the results.
4. Write test cases for all the implemented methods.
5. Complete a literature survey of papers relevant to the above mentioned sorts.
6. Submit sorted samples generated by the algorithms.

Study

Chinese characters do not have an alphabet like other popular languages and there is no standard way to organize Chinese characters. Chinese dictionaries come in many variations each following a way of ordering the words in them. Some of the popular methods include:

- Sorting based on the radical/root character.
- Sorting based on the number of brush strokes in the character.
- Sorting based on the pinyin representation.
- Some use a combination of the above methods.

For the purpose of the project we decided to choose the Pinyin based sorting method. In order to convert chinese words to it's pinyin format we are using the library from - http://pinyin4j.sourceforge.net/

Implementation

The following steps describe how we implemented the system to benchmark and compare all the sorting algorithms:

1. Define a new class called “Pair” that consists “ChineseName” and “PinyinName” as it’s member variables. The instances of this class (sortable objects) are comparable objects. The purpose of this design is to avoid having to maintain a lookup table in order to convert the pinyin words back to chinese words. Using this method saves a bunch of time and memory that lookup tables would have taken otherwise.
2. MSD Radix Sort and LSD Radix Sort: The class MSDStringSort and LSDStringSort (available in info6205 repo) has been modified to take an array of pairs and sort them.
3. Husky Sort: The class PureHuskySort (available in “The-repository-formerly-known-as” repo) has also been modified to take an array of Pairs and sort them.
4. Tim Sort and Dual Pivot Quick Sort: The classes TimSort and QuickSort_DualPivot are used from the “The-repository-formerly-known-as” repo to sort the comparable Pair objects.
5. Test cases have been implemented for all the modified code.
6. RandomChineseNamesGenerator: This class returns ‘x’ number of randomly ordered chinese names that are taken from the shuffled chinese words that were provided for this project.
7. ChineseToPinyinConverter: This class contains methods that help convert chinese names to their equivalent pinyin format.
8. A runner class is implemented that runs all the above mentioned sorting algorithms and times them. This class uses the Benchmark_Timer class from the info6205 repo to run the benchmarks for comparison.
9. The time taken to create an array of sortable objects (Pair[]) using the chinese name and it’s derived pinyin name is also considered in the benchmark time.

10. Something that we also tried doing but could not make it in time:
    a) Use the Husky Chinese Encoder to generate a long value for every chinese name and create sortable objects using that. (Successful)
    b) Implement MSD and LSD Radix Sorts to sort long values. (Error to be fixed)
    c) Modify PureHuskySort, QuickSort_DualPivot and TimSort to take in these husky encoded pairs and sort them. (Successful)
    d) Run Benchmarks for these sorting algorithms and compare results.

Results

Looking at the results from benchmarking the algorithms that sorted the chinese names based on their Pinyin format we can conclude the following:

MSD Radix Sort consistently performs the best when compared to Tim Sort, Husky Sort, Quick Sort Dual Pivot and LSD Radix Sort.

We believe the reason for this behavior to be in the nature of how MSD Radix Sort is designed. In just one pass the list is mostly sorted and with every pass it the sortedness of the list increases largely. The literature survey submitted along with this project also gives us some insights into why this behavior is witnessed. For a large list of strings MSD Sort seems to work the best.

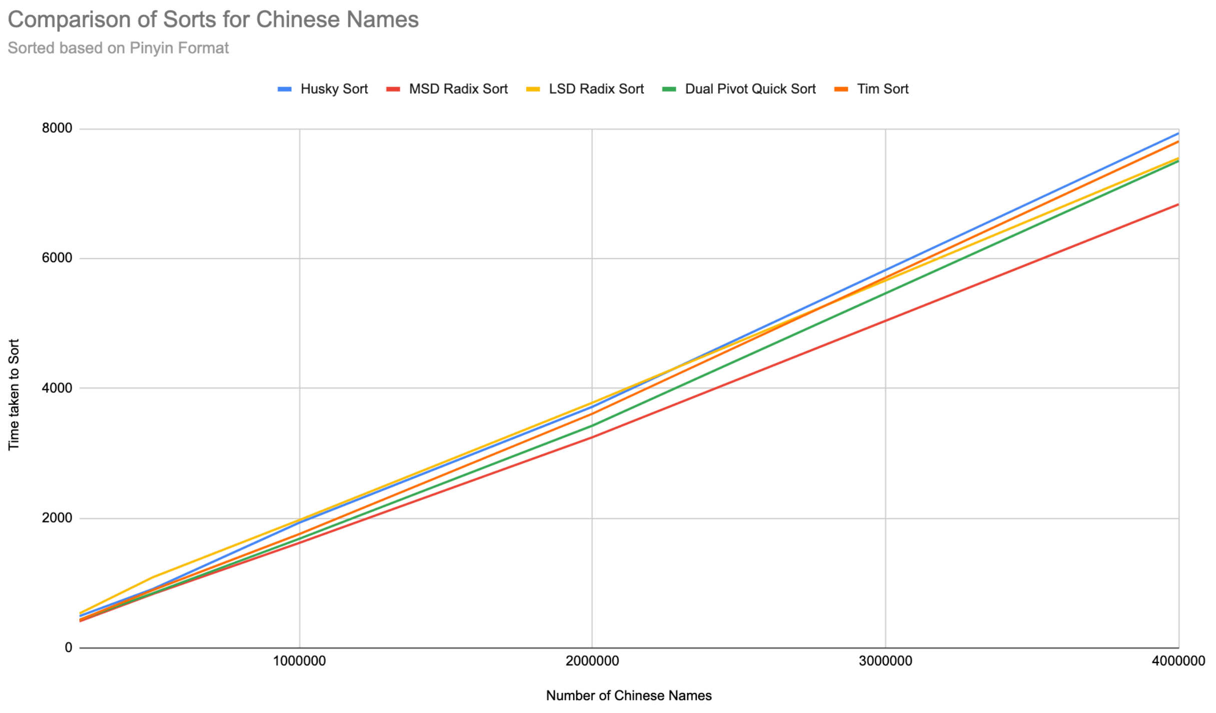
(These tests have been conducted on Macbook M1 Air)

Type of sort / No. of Names	250000	500000	1000000	2000000	4000000
Husky Sort	489.7127542	905.5744397	1930.008852	3714.76895	7929.215408
MSD Radix Sort	409.7612314	828.1087999	1617.231835	3245.125352	6835.085879
LSD Radix Sort	530.5913687	1087.741904	1969.810558	3778.594139	7544.671796
Dual Pivot Quick Sort	432.6157313	839.9439083	1680.531269	3424.173115	7502.378421
Tim Sort	430.3977853	889.9094272	1752.960694	3607.810517	7803.963752

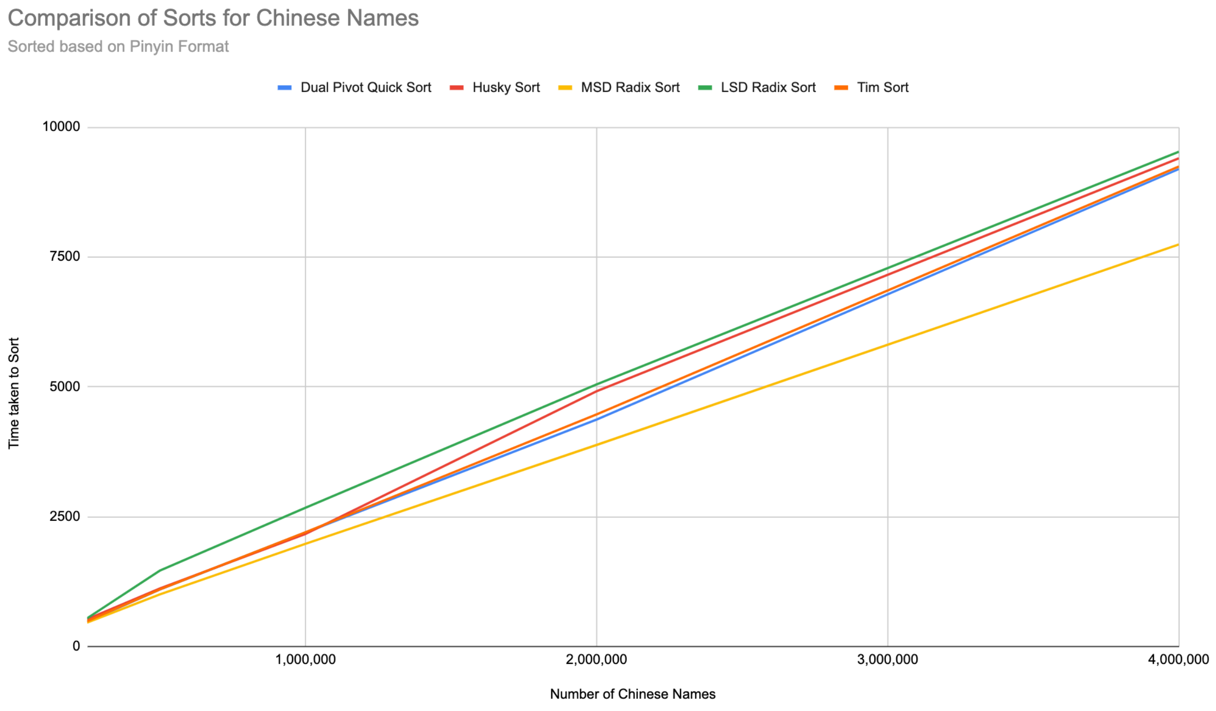
(These tests have been conducted on AMD Based laptop with 16GB Ram)

Type of sort	250,000	500,000	1,000,000	2,000,000	4,000,000
Dual Pivot Quick Sort	477.175405	1098.34119	2191.841485	4366.20205	9193.016155
Husky Sort	524.35871	1116.114395	2163.77057	4912.540615	9398.256385
MSD Radix Sort	457.59282	1000.19802	1974.222965	3879.494225	7739.49366
LSD Radix Sort	540.30326	1460.311685	2669.93896	5045.726715	9525.80362
Tim Sort	484.76438	1099.10208	2196.5876	4467.6817	9240.619685

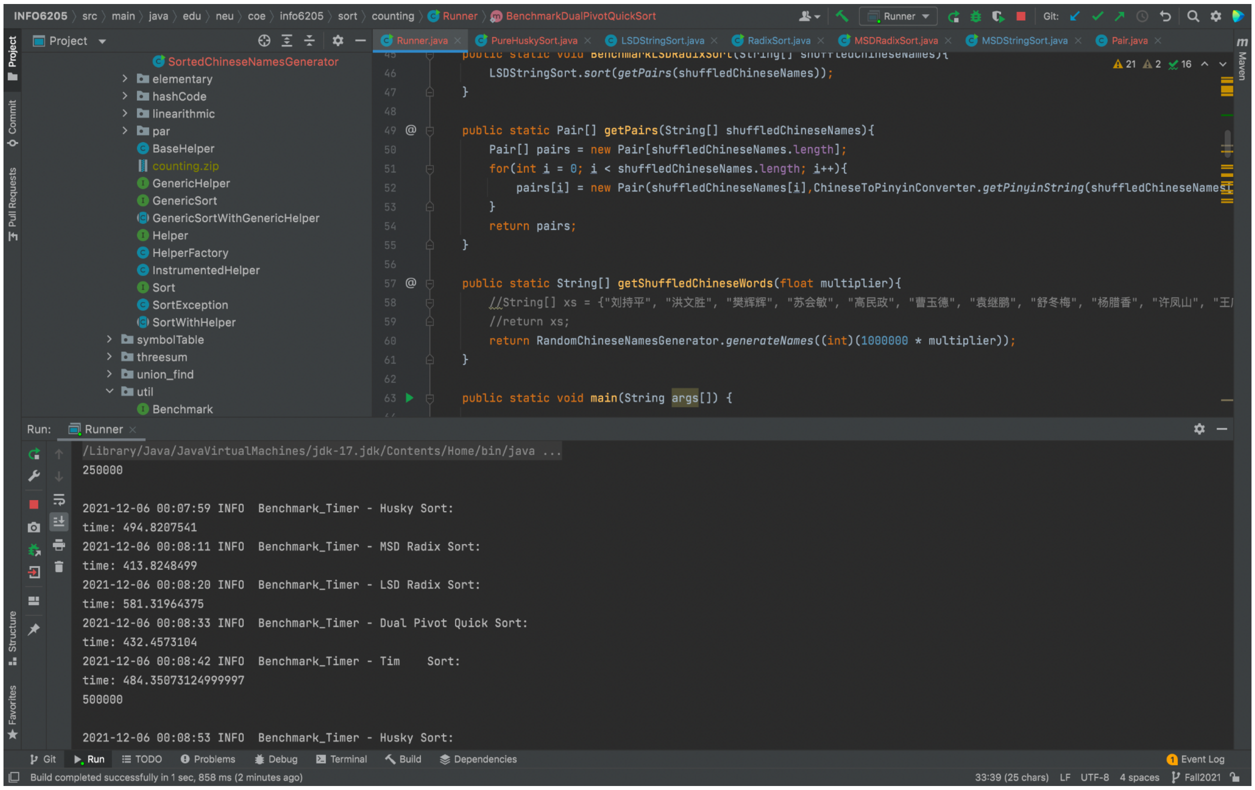
Graph for Mac Results



Graph for AMD Results



Output from Runner



```
    public static void BenchmarkLSDRadixSort(String[] shuffledChineseNames){
        LSDStringSort.sort(getPairs(shuffledChineseNames));
    }

    public static Pair[] getPairs(String[] shuffledChineseNames){
        Pair[] pairs = new Pair[shuffledChineseNames.length];
        for(int i = 0; i < shuffledChineseNames.length; i++){
            pairs[i] = new Pair(shuffledChineseNames[i],ChineseToPinyinConverter.getPinyinString(shuffledChineseNames[
        }
        return pairs;
    }

    public static String[] getShuffledChineseWords(float multiplier){
        //String[] xs = {"刘持平", "洪文胜", "樊辉辉", "苏会敏", "高民政", "曹玉德", "袁继鹏", "舒冬梅", "杨腊香", "许凤山", "王
        //return xs;
        return RandomChineseNamesGenerator.generateNames((int)(1000000 * multiplier));
    }

    public static void main(String args[]) {
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...
250000

2021-12-06 00:07:59 INFO  Benchmark_Timer - Husky Sort:
time: 494.8207541
2021-12-06 00:08:11 INFO  Benchmark_Timer - MSD Radix Sort:
time: 413.8248499
2021-12-06 00:08:20 INFO  Benchmark_Timer - LSD Radix Sort:
time: 581.31964375
2021-12-06 00:08:33 INFO  Benchmark_Timer - Dual Pivot Quick Sort:
time: 432.4573104
2021-12-06 00:08:42 INFO  Benchmark_Timer - Tim    Sort:
time: 484.35073124999997
500000

2021-12-06 00:08:53 INFO  Benchmark_Timer - Husky Sort:
```

Build completed successfully in 1 sec, 858 ms (2 minutes ago)