Program Structures and Algorithms (INFO6205)
Fall 2021
Final Project - Report

Akhilanand Sirra (002197798), Pramithi Jagdish (002192816), Venkata Srinivas Kompally (002137855)

TASKS:

Following is the list of Tasks performed as part of this Project

1. Implement MSD radix sort for a natural language which uses Unicode characters.
2. Choose your own language or (Simplified) Chinese.
3. Write a report comparing your method with Timsort, Dual-pivot Quicksort, Huskysort, and LSD radix sort.
4. Additionally, complete a literature survey of relevant papers while implementing Radix sort.

Introduction:

The term sorting refers to the process of arranging elements in a list in some way. Usually, numerical order, lexicographical order, or ascending or descending order are used. In our project, we aimed to sort a large collection (1 million unique strings) of Chinese and Telugu language names using MSD Radix sort and benchmark the results in comparison to various other sorting techniques and perform a comparative analysis on them.

Background:

The idea of Radix Sort is to do character by character sort starting from the least significant digit to most significant characters or the other way around. Radix sort uses counting sort as a subroutine to sort. Non-comparison Radix sort is an algorithm that sorts values instead of comparing them by sorting the elements in the values, either digits or letters. The two variations of this that we have worked with are:

MSD Radix sort - Most significant digit radix sort works with us considering characters to sort from the left most bit to the right, i.e., most significant first.

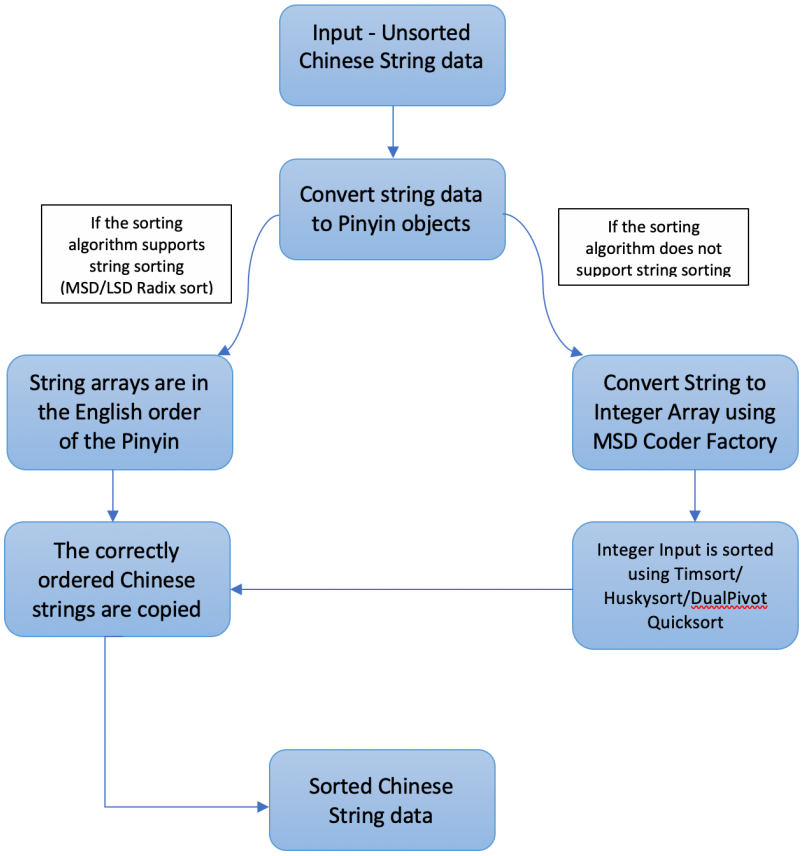
LSD Radix sort - Least significant digit radix sort is the opposite where we start from the right and move to the left, i.e., the least significant digit first.

Tim Sort - Timsort is a hybrid stable sorting algorithm derived from insertion sort and merge sort. It is designed to perform well on many kinds of real-world data. Tim Peters implemented it in 2002 for use in the Python programming language. The algorithm finds the subsequence of the already ordered data (runs) and uses them to sort the rest more efficiently.

DualPivotQuickSort - Dual pivot quick sort involves taking two pivots, one at the left end of the array and the other at the right end. We must compare the left pivot to the right pivot and make sure the left is less than the right or swap them if necessary. We then divide the array into three parts: the first part will contain elements that are smaller than the left pivot, the second part will contain elements that are greater than the left pivot and also smaller than or equal to the right pivot, and the third part will contain elements that are larger than the right pivot.

HuskySort - Java's two sorting systems are dual-pivot quicksort (for primitives) and Timsort (for objects). Huskysort combines these two algorithms that can run significantly faster than either algorithm alone for the types of objects that are expensive to compare. This approach to sorting reduces the number of expensive comparisons in the linearithmic phase by substituting inexpensive comparisons.

Algorithm	Worst Complexity	Average Complexity	Best Complexity
MSD Radix Sort	(n*m) where m = the average length of strings.	n	n
LSD Radix Sort	n*m	n*m	n*m
Timsort	n*log(n)	n*log(n)	n
Dual Pivot Quicksort	n^2	n*log(n)	n*log(n)
Huskysort	n*log(n)	n*log(n)	n*log(n)



Implementation:

MSD radix sort can already sort English/Telugu strings, so we will need a way to convert Chinese strings to Pinyin and then use MSD radix string sort to get the conventional order for Chinese. To convert Chinese strings to Pinyin, we invoked the pinyin4j utility package. Then we sorted them by Husky sort, Tim sort, Dual-Pivot Quicksort, MSD, and LSD radix sort algorithms. All these algorithms are then benchmarked and compared based on their runtime.

Test Cases results and Analysis:

We were able to sort 1000 Telugu and Chinese words successfully. Below is the supporting evidence for the various algorithms used to sort the string input.

***Fig 2.1**: DualPivotQuickSort Test Case - English, Chinese & Telugu*

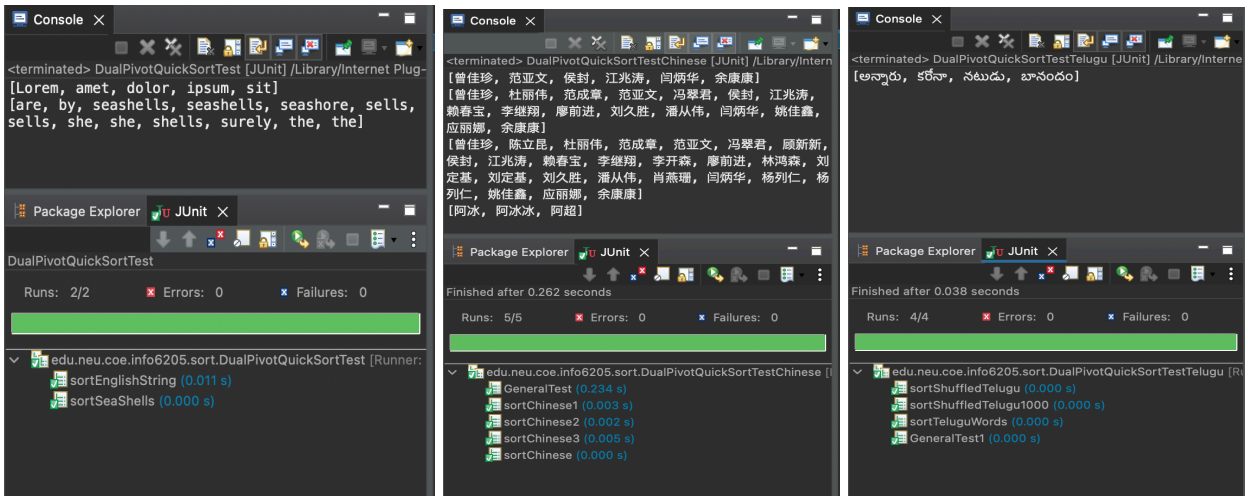


***Fig 2.2**: HuskySort Test Case - English, Chinese & Telugu*

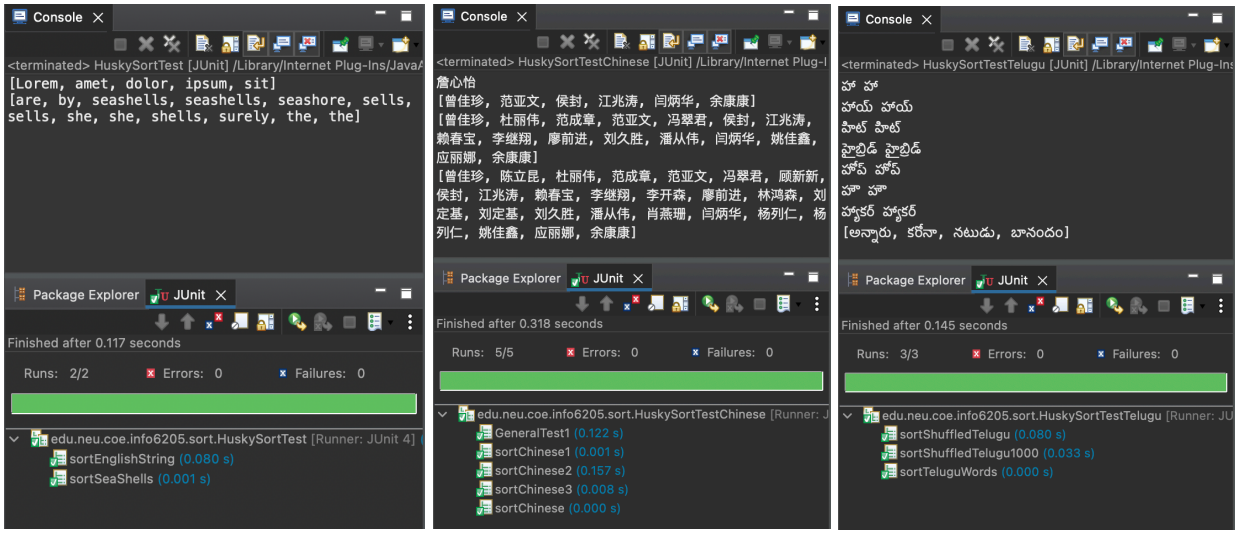


***Fig 2.3**: LSDStringSort Test Case - English, Chinese & Telugu*

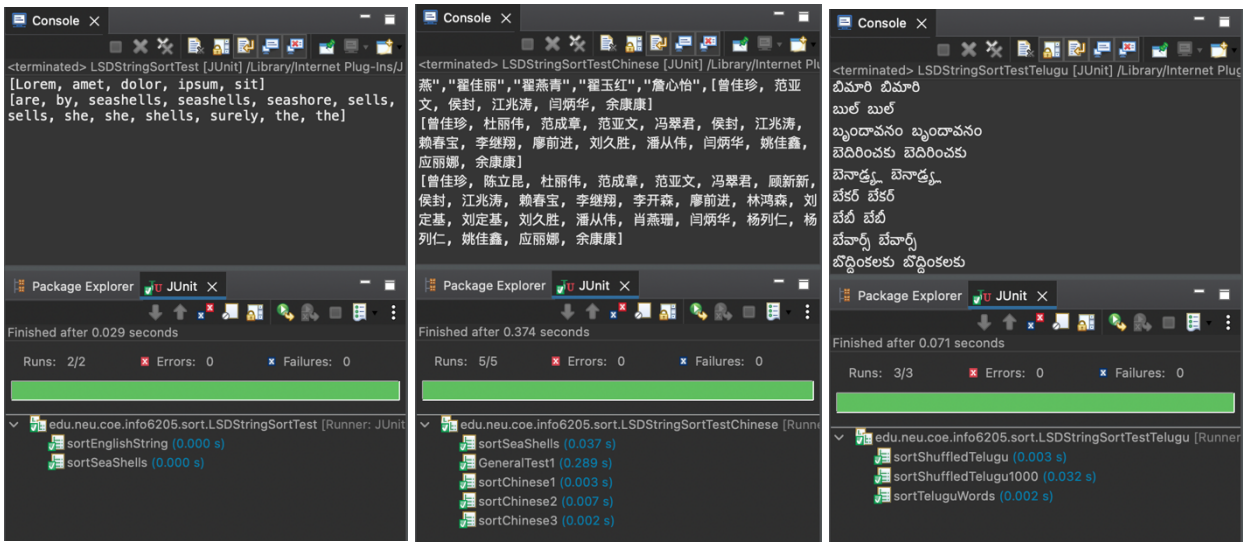


***Fig 2.4**: MSDStringSort Test Case - English, Chinese & Telugu*

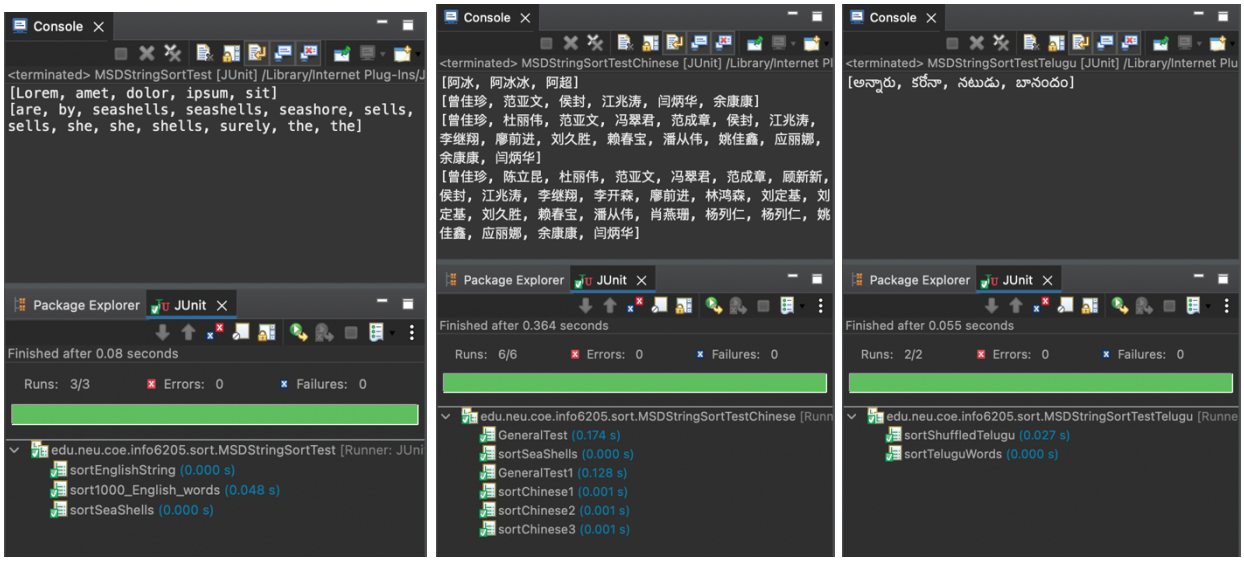
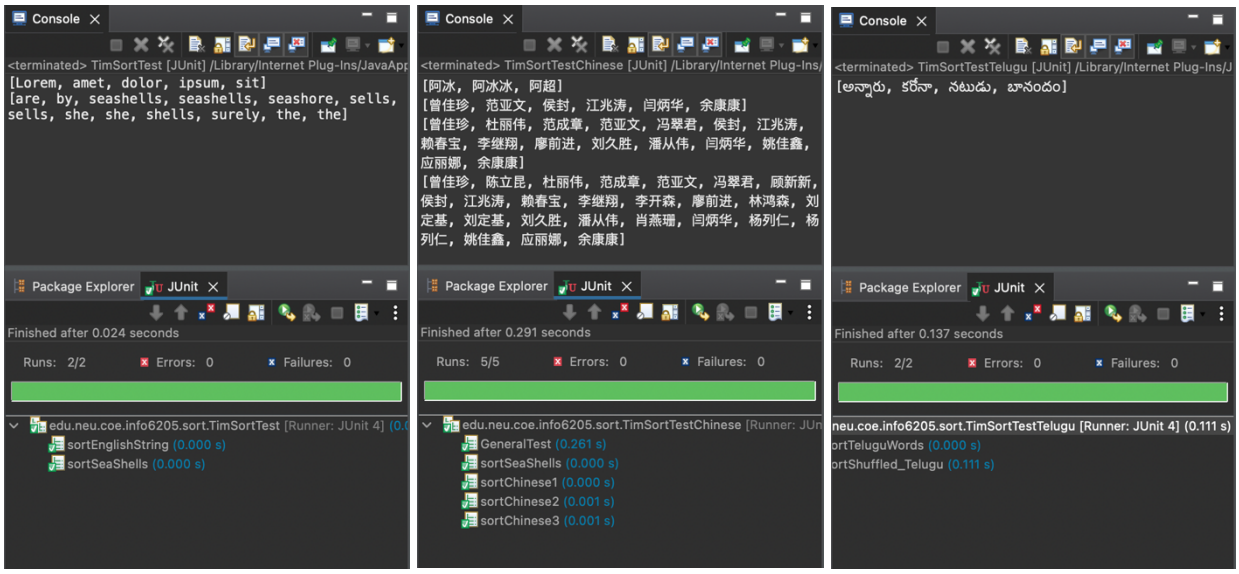


Fig 2.5: *TimSort Test Case - English, Chinese & Telugu*



For Chinese strings benchmark results:

Fig 3.1: *MSD Radix Sort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
MSD	250000	5243.398057
MSD	500000	9004.409708
MSD	1000000	17909.00611
MSD	2000000	38672.02019
MSD	4000000	83389.28438

Fig 3.2: *LSD Radix Sort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
LSD	250000	6046.199433
LSD	500000	10815.97081
LSD	1000000	21682.8554
LSD	2000000	43178.90998
LSD	4000000	85200.5397

Fig 3.3: *Dual Pivot Quicksort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
Dual Pivot Quicksort	250000	3934.279126
Dual Pivot Quicksort	500000	8026.686718
Dual Pivot Quicksort	1000000	17007.04191
Dual Pivot Quicksort	2000000	37396.72948
Dual Pivot Quicksort	4000000	78832.59623

Fig 3.4: *Timsort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
Timsort	250000	3783.70426
Timsort	500000	7746.835738
Timsort	1000000	16243.18195
Timsort	2000000	34210.72532
Timsort	4000000	71376.72105

Fig 3.5: *Huskysort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
Huskysort	250000	165.5118752
Huskysort	500000	334.4823602
Huskysort	1000000	703.1959474
Huskysort	2000000	1394.076129
Huskysort	4000000	2975.41815

For Telugu strings benchmark results:

Fig 3.6: *MSD Radix sort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
MSD	250000	2568.979191
MSD	500000	8259.280555
MSD	1000000	19931.32448
MSD	2000000	22006.68
MSD	4000000	25572.05127

Fig 3.7: *LSD Radix sort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
LSD	250000	769.703451
LSD	500000	1308.115778
LSD	1000000	3721.037631
LSD	2000000	7010.927644
LSD	4000000	15491.51308

Fig 3.8: *Dual Pivot Quicksort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
Dual Pivot Quicksort	250000	101.5759942
Dual Pivot Quicksort	500000	233.6251256
Dual Pivot Quicksort	1000000	496.1981794
Dual Pivot Quicksort	2000000	999.5083228
Dual Pivot Quicksort	4000000	2463.960615

Fig 3.9: *Huskysort Algorithm runtime result for array lengths ranging from 250000 to 4000000*

Algorithm	Array Length	Runtime
Huskysort	250000	89.5363006
Huskysort	500000	192.84905
Huskysort	1000000	386.763609
Huskysort	2000000	858.4698212
Huskysort	4000000	1975.045145

Conclusion:

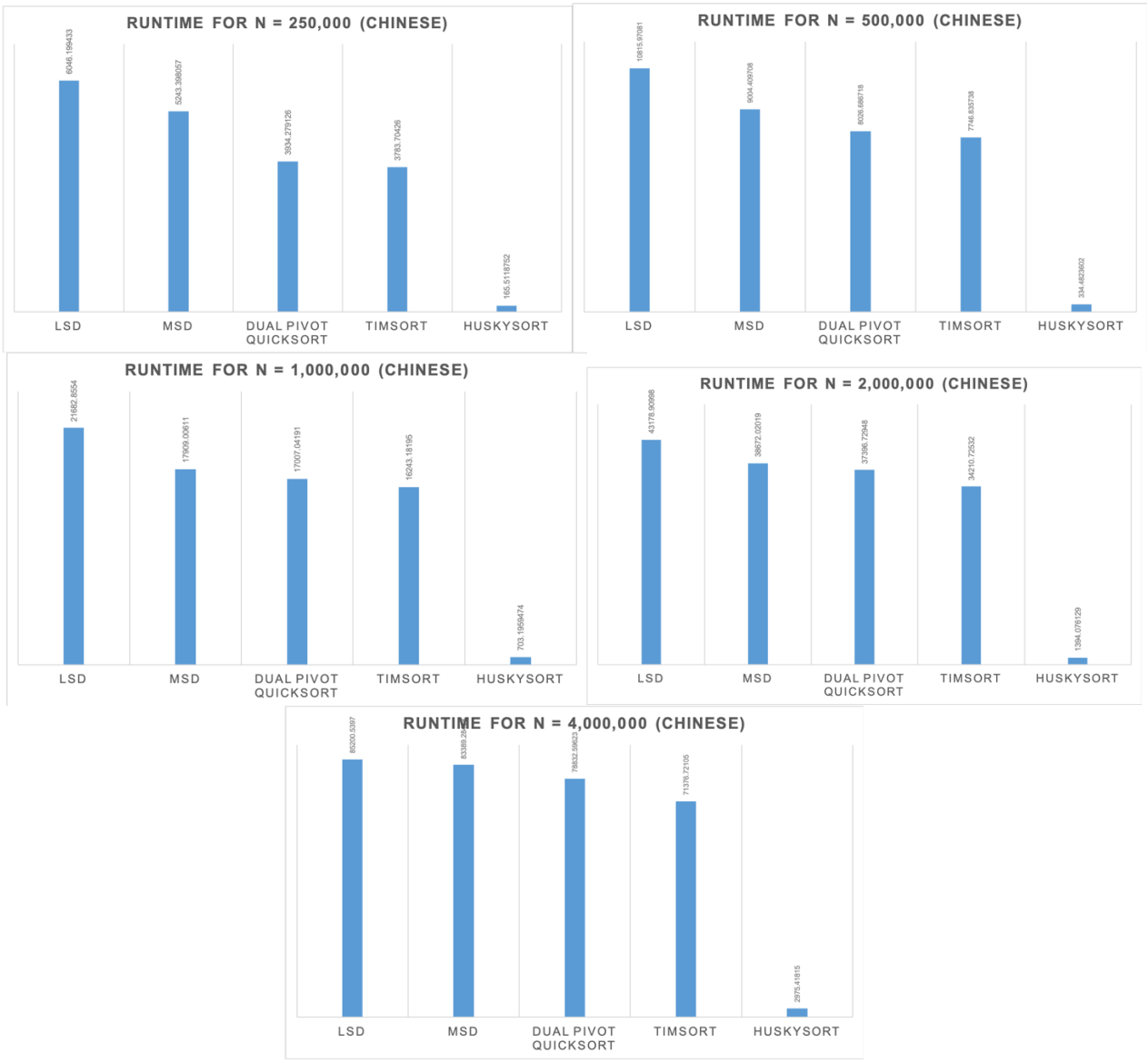
1. We have implemented 5 different algorithms - MSD Radix sort, LSD Radix sort, Dual Pivot Quicksort, Timsort and Huskysort to sort the Chinese and Telugu string data input.

2. The best to worst performing Algorithm in terms of run time is as follows:

For Chinese string data: ***Huskysort > Timsort > Dual Pivot Quicksort > MSD > LSD***
For Telugu string data: ***Huskysort > Dual Pivot Quicksort > LSD > MSD***

Please find evidence to support this conclusion below:

Fig. 1.1: Sorting Chinese Words Benchmark Comparison



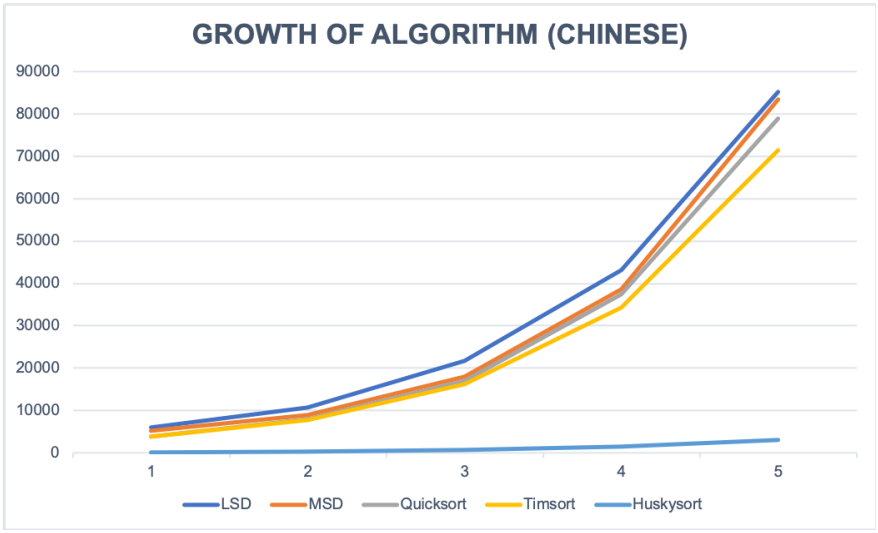
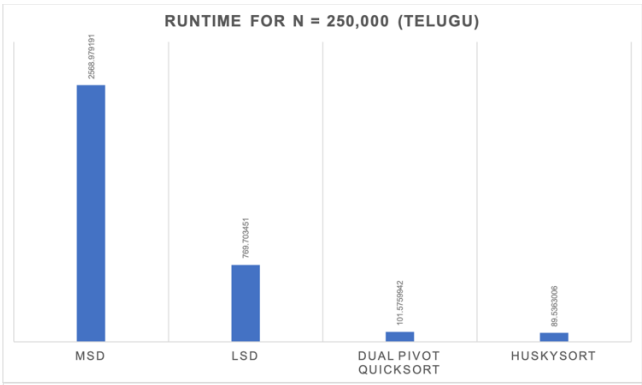
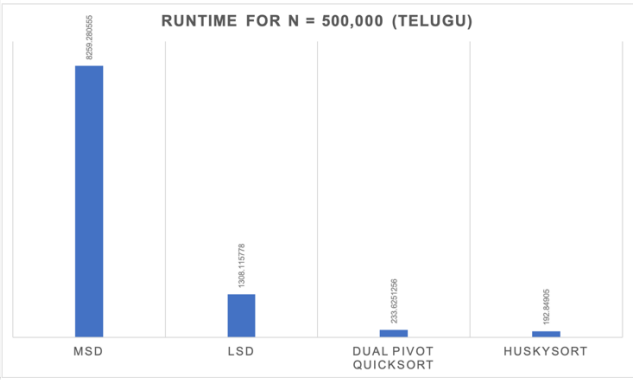
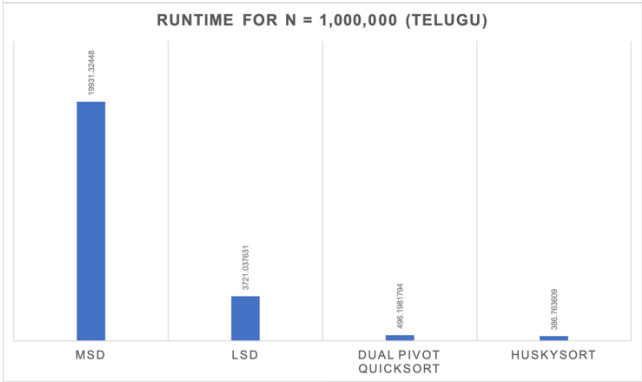
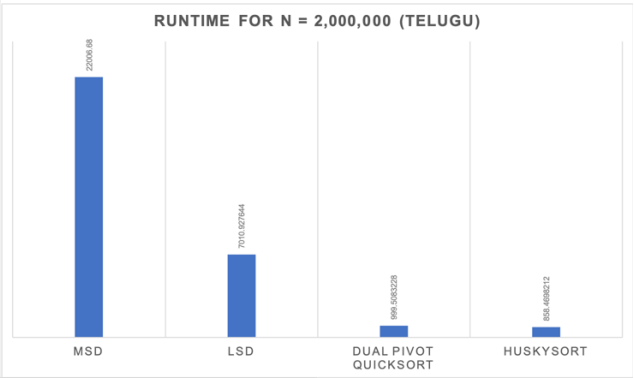


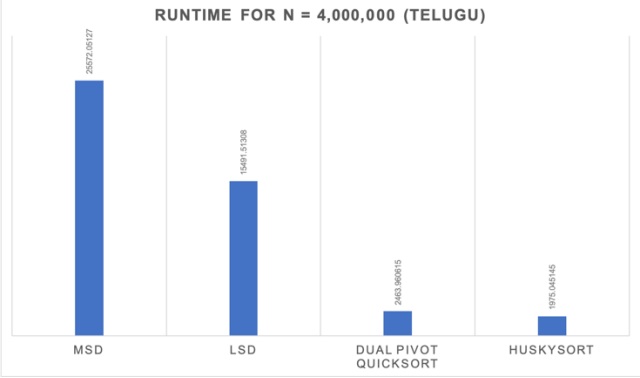
Fig 1.2: Sorting Telugu Words Benchmark Comparison

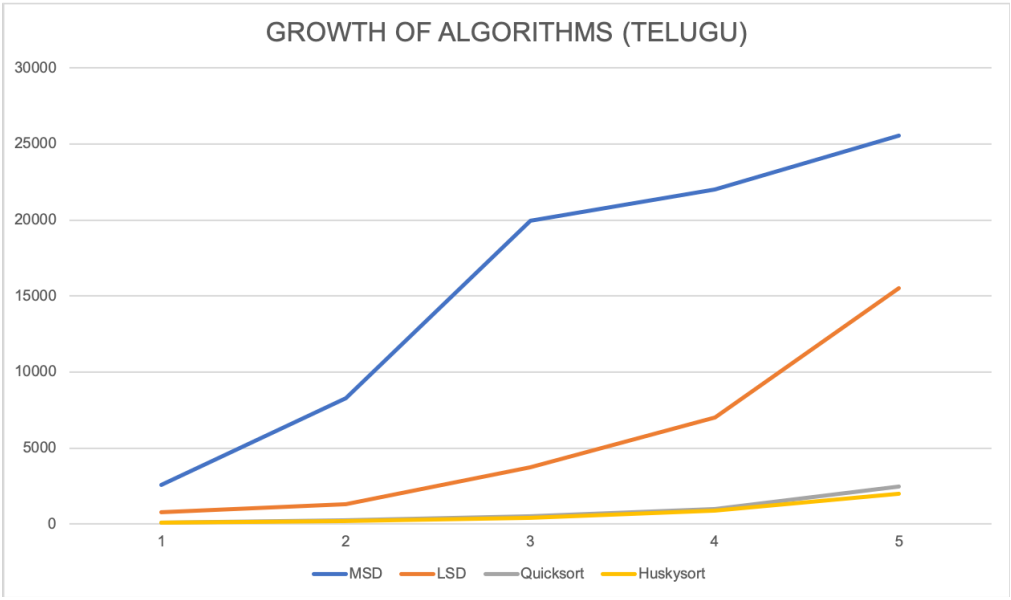












References:

1. https://arxiv.org/pdf/2012.00866.pdf
2. https://deliverypdf.ssrn.com/delivery.php
3. https://www.cs.princeton.edu/~rs/AlgsDS07/18RadixSort.pdf