## INFO 6205 Final Project

## Report on Code, Benchmarking and Findings

Trinadh Phani GInjupalli(002117493)

Anil Kumar Pothuraju(001592990)

Mohit Kumar Parvathaneni(001565574)

The task was to implement MSD radix sort for a natural language which uses Unicode characters.

## Implementation Overview

[Code Repository](https://github.com/Phani56/INFO6205project)

Sorting in natural languages is implemented for **MSD Radix Sort, LSD Radix Sort, Dual Pivot Quicksort, Tim Sort** and **Husky Sort**.

Configured all the above sorting algorithms to run for **Chinese** and **Telugu** languages.

Chinese is implemented in Pinyin ordering for all the Sorts with the help of [Pinyin4j](http://pinyin4j.sourceforge.net/) library for conversion and Telugu is implemented using the default ordering of unicode. Configured the setup to run either one of these languages *config.properties* file.

**Note**: While all the Sorting algorithms implemented Chinese sort using Pinyin4j library, HuskySort implements it using Collator.

## Findings

Firstly the following are some of the **findings/insights during code implementation** of individual sortings

### MSD Radix Sort

Initially MSD Radix sort was implemented with the basic unicode range for both CHINESE and TELUGU, but later made the following changes

* The algorithm gives a better performance when the radix range is modified to the range of natural language i.e **256 for Chinese** since we are converting to Pinyin and **128 for Telugu** (unicode from 3072 to 3200) instead of **65535**.
* Collator and Pinyin had almost the same performance for MSD.
* InsertionSortMSD improves the performance when MSD is cut off for smaller array sizes. Although Telugu didn’t have a significant difference with/without the cutoff, for Chinese it was upto 8 times faster with the cutoff.
* Chinese is slower compared to Telugu because of Pinyin conversion.

### LSD Radix Sort

Basic implementation for LSD Radix sort

* Slower than MSD Radix for all types of input types and input sizes.
* For Chinese it needs Pinyin string conversion in 2 places charAsciiVal and findMaxLength functions which adds up to the additional time.
* Chinese is slower compared to Telugu because of Pinyin conversion.

### Tim Sort

* Implementation is fairly straightforward with the system sort.
* Chinese is slower compared to Telugu because of Pinyin conversion.

### Husky Sort

* Directly adapted from HuskySort Repository.
* The output from HuskySort is different from other sortings implemented in this project. Husky uses Collator whereas other sorting algorithms.

**Findings/insights from benchmarking process** of individual sortings

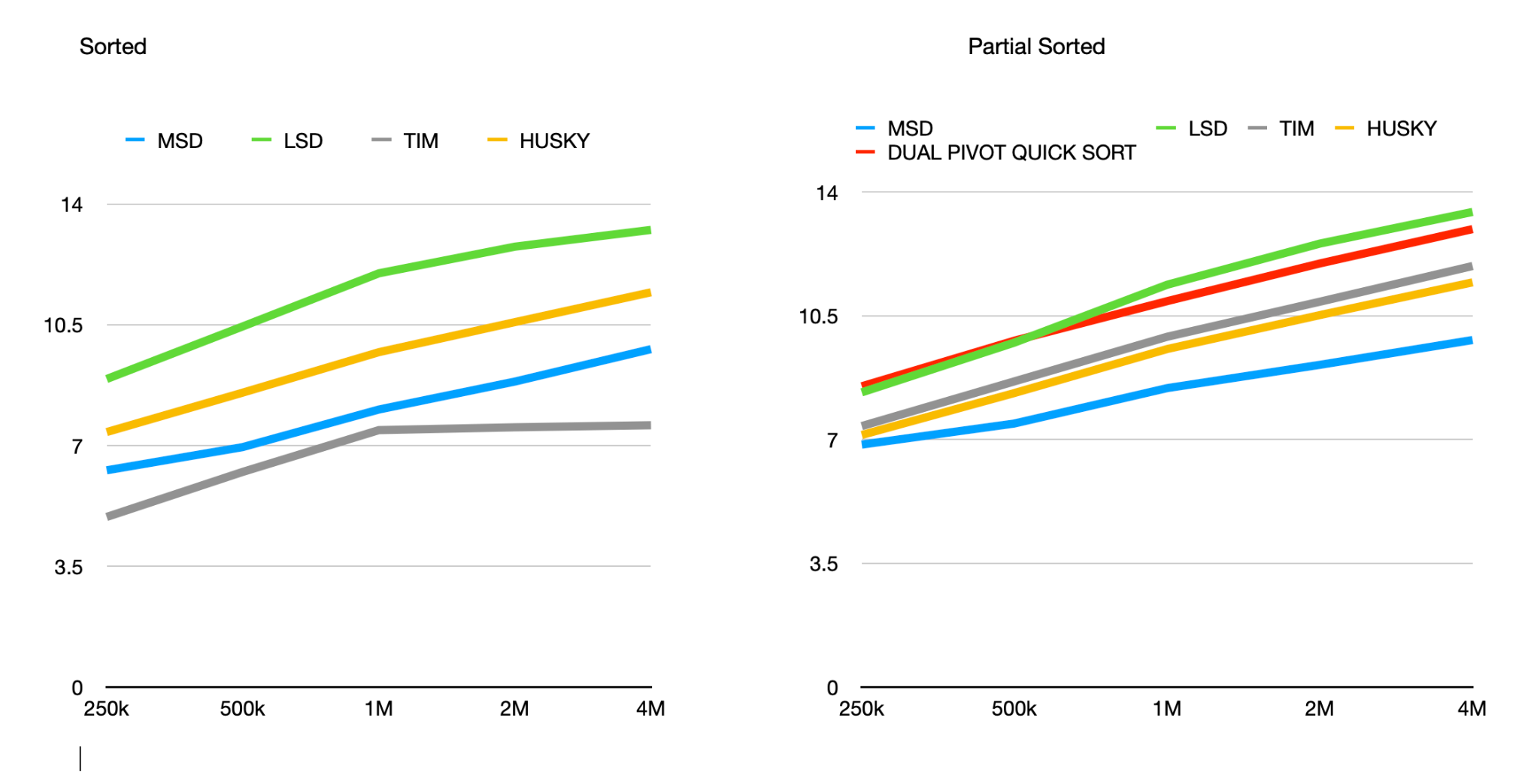
To fairly evaluate the algorithms, the benchmarking was implemented with different types of inputs, RANDOM, PARTIALLY SORTED, SORTED and REVERSE SORTED.

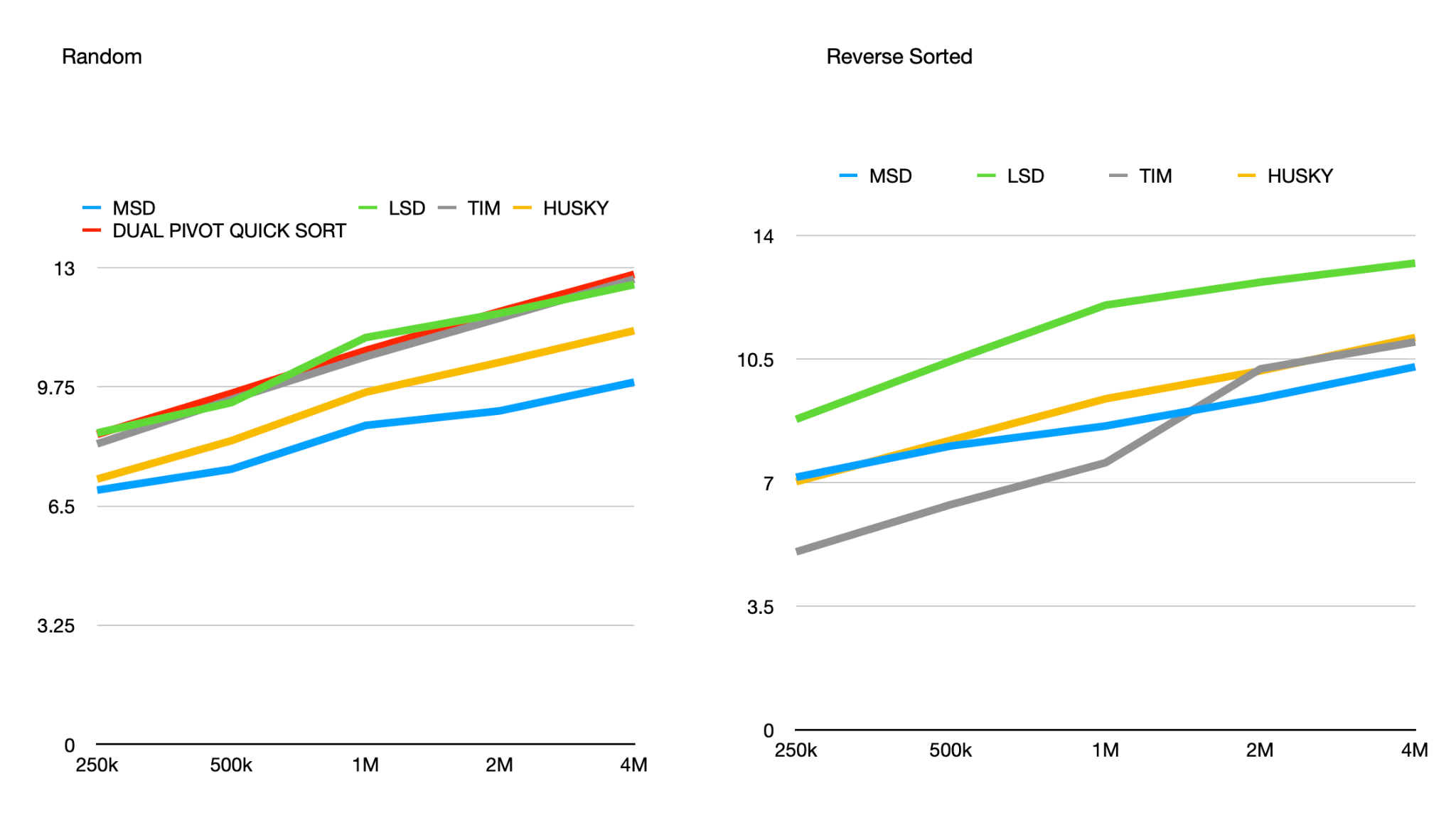
***Benchmarking*** : Taken average of 5 runs with two increments starting from 250K.

* *250K, 500K, 1M, 2M, 4M values*

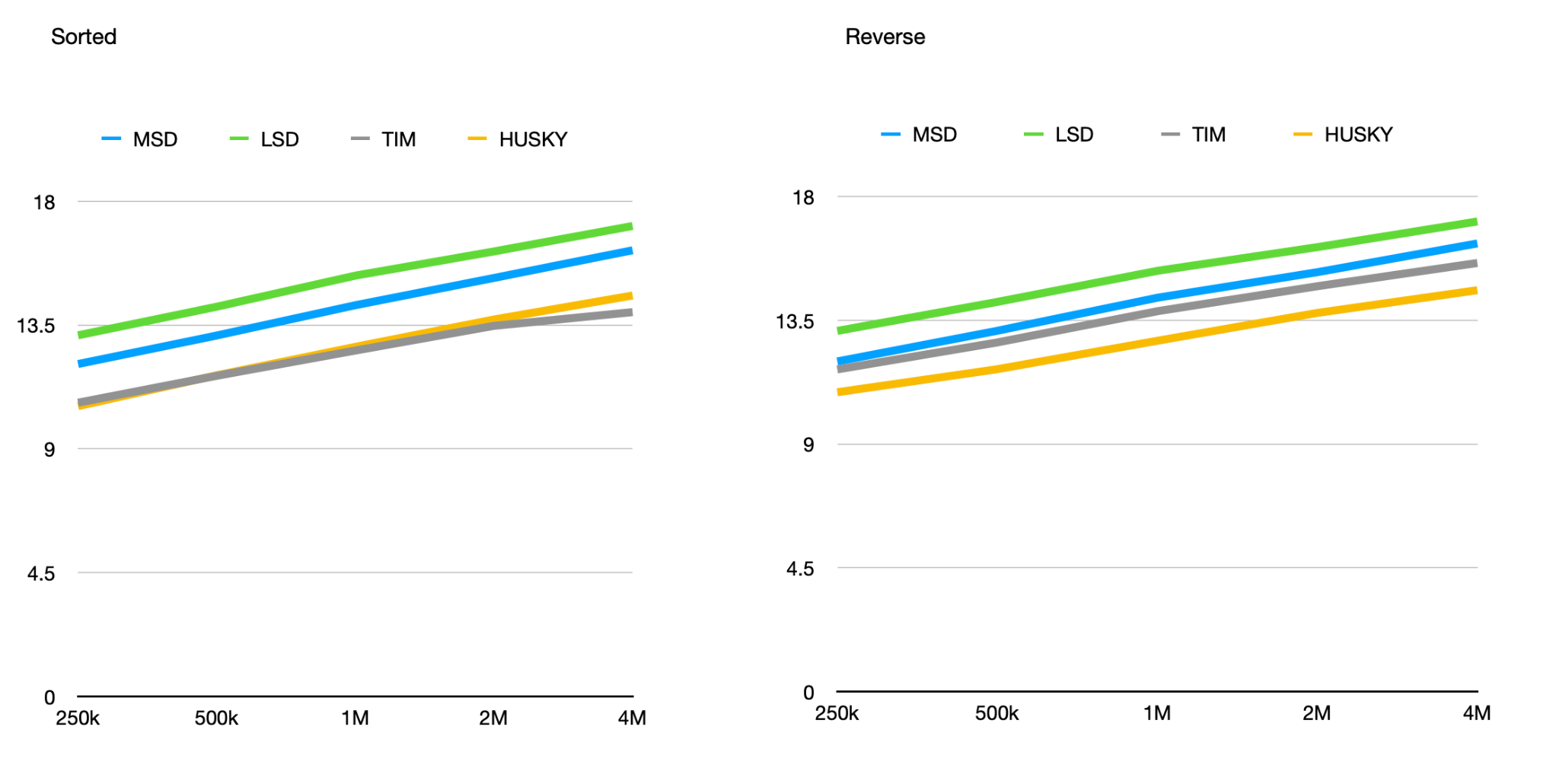
Following is the graphical representation of **LogT vs LogN** graphs. The actual values are uploaded to the repository in benchmarking\_results folder

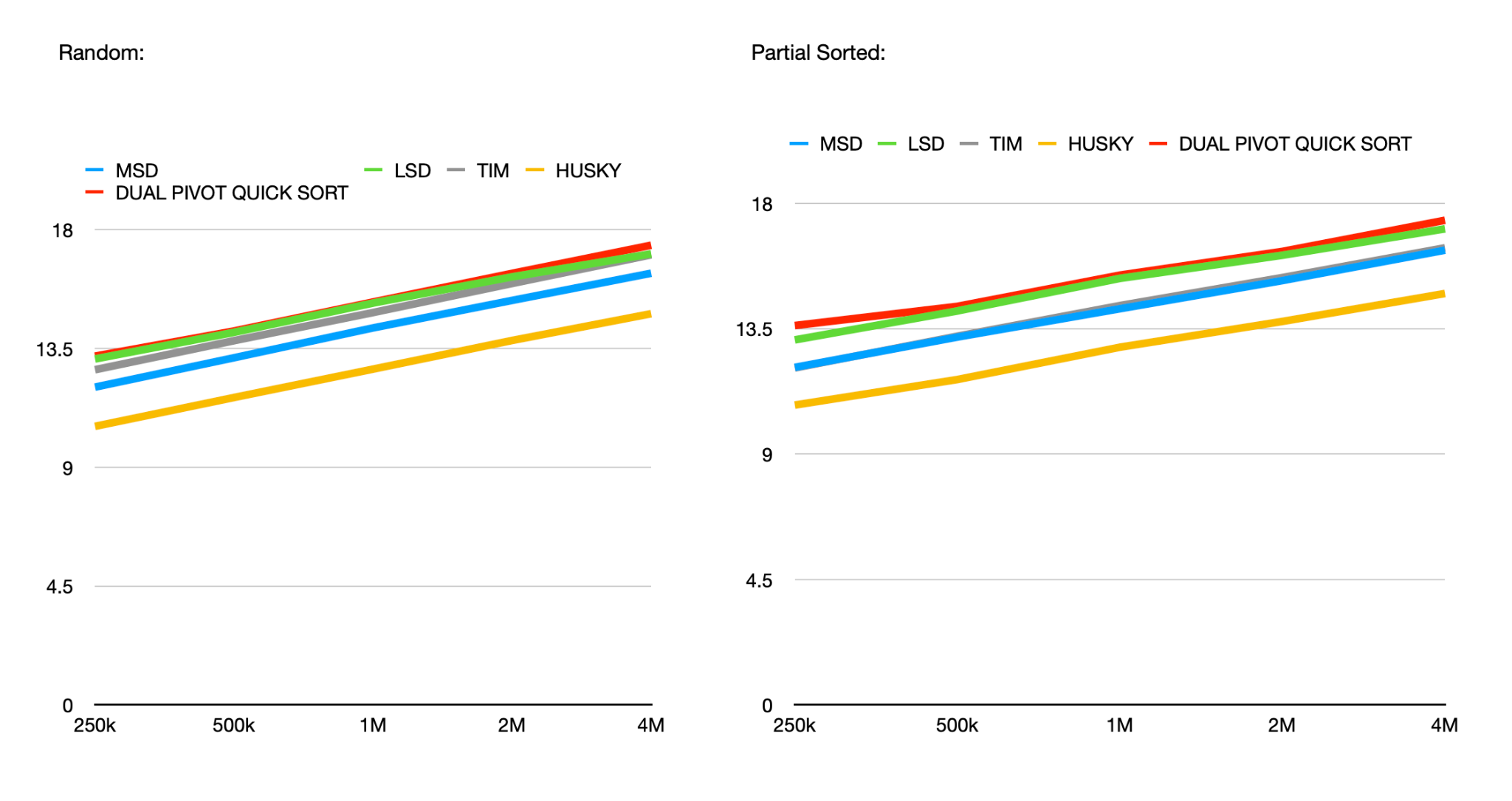
#### Graphs for Telugu words





Graphs for chinese words





* **MSD Sort** and **HuskySort** had better overall performance compared to other sorts irrespective of the input type.
* Performance of MSD, LSD does not change much based on the input while MSD had the best, LSD had the worst performance after Dual-Pivot.
* Dual Pivot quicksort had the worst performance for Sorted and Reverse sorted inputs. The tests resulted in a stackoverflow error in both the cases.

## Conclusion

Based on the benchmarking for two languages, the results for “Telugu” were more coherent and comparable than “Chinese”. This was mainly because Husky Sort used Collator where as the other sorts are implemented using Pinyin4j library

From the results of telugu words

1. MSD and LSD perform the same irrespective of input order of the strings.
2. Dual pivot performs worse for sorted and reverse sorted cases
3. This is the following order of performance for each sort

For sorted : Tim Sort > MSD Sort > Husky Sort > LSD Sort > Dual Pivot QuickSort

For random: MSD Sort > Husky Sort > Tim Sort > Dual Pivot QuickSort > LSD Sort

For partial sorted: MSD Sort > Husky Sort > Tim Sort > LSD Sort > Dual Pivot QuickSort

For reverse sorted: Tim Sort > Husky Sort > MSD Sort > LSD Sort > Dual Pivot QuickSort