Program Structures & Algorithms
Final Project report - Fall 2021

Team Members
Akshay Singh Bayes (002956209)
Nimish Sharma (001598648)
Tanay Saxena (001586302)

Tasks Performed:

- Analysis, Implementation & unit testing of MSD radix sort.
- Analysis, Implementation & unit testing of LSD radix sort.
    - Implementation of Custom comparator for Chinese characters in pinyin order.
- Analysis, Implementation & unit testing of Husky Sort sort.
- Analysis, Implementation & unit testing of Dual-Pivot quick sort.
- Benchmarking of Husky Sort, Dual Pivot Quicksort against MSD Radix Sort & LSD Radix Sort.

Challenges:

1. **MSD & LSD Radix Sort:**
   Problem: When implemented for English characters, the frequency calculation and cumulative sum of MSD Radix sort in the worst-case scenario we need to iterate over an array of 256 characters for each recursive call in O(N) time (N = 256). However, when it comes to Chinese characters the total count is around 65K, where this iteration and array creation results in very poor performance.

   Solution: We implemented a TreeMap to store the frequency for only those Chinese characters which are in scope for the function call
   (in the range **L** to **R** only for the D^{th} digit of each string, where all the Strings in the range **[L, R]** have their **$(D-1)^{th}$** character is the same**)**.
   This TreeMap would be initialized with a suitable comparator to preserve the order of characters to be sorted. Hence in a general case we never actually encounter all the 65K characters, instead, we only iterate (in O(MlgM) time where M is the distinct characters being considered) the once we need to.

2. **String Comparison (Collator):**
   Problem: The use of Collation causes all the algorithms to slow down significantly.

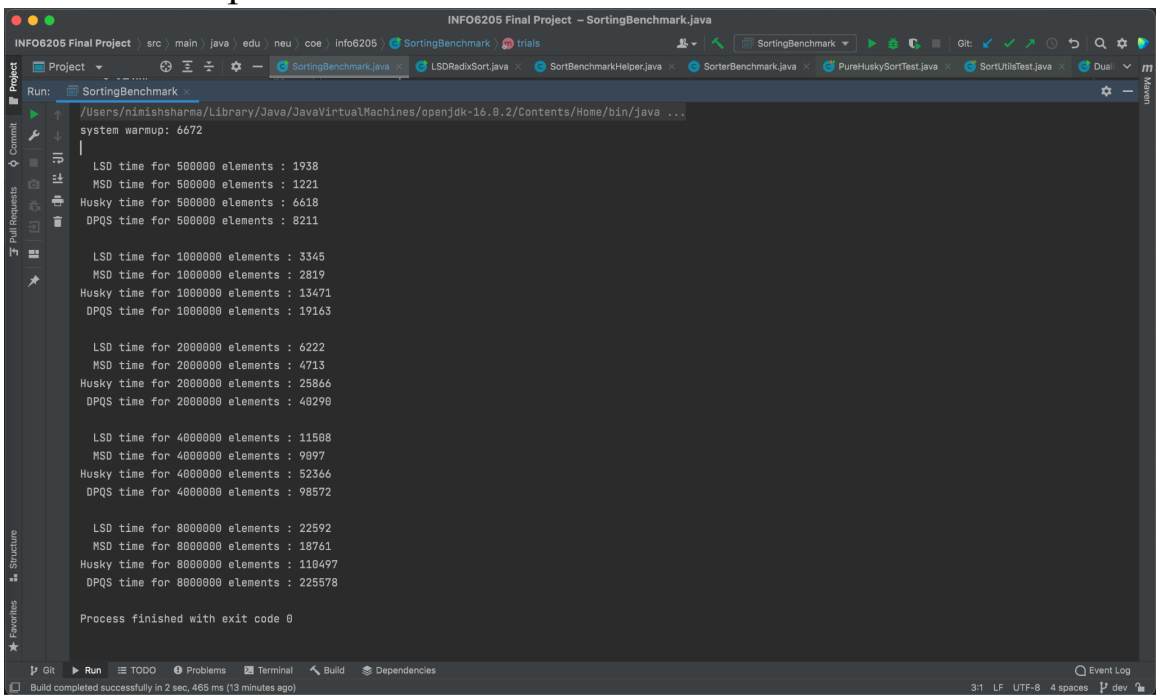
   Solution: In order to solve this problem we developed a new comparator called "OrderMaster" (.../utils/OrderMaster.java). This comparator finds the distinct characters in the corpus (in this particular case suffuledChinese.txt file), sorts them based on the described order (in this case the order given by Collator), and hashes the results with ranks of characters. These ranks are determined by the sorted order. When used in practice, this comparator works in O(1) constant time, for each subsequent call after the initial pre-processing step (done during the object initialization).

Conclusion:

- Based on the benchmarking results, it can be safely concluded that the enhanced implementation of MSD and LSD Radix sorts with TreeMap and comparator, outperform the pure husky sort and dual-pivot quicksort by at least three folds (for Chinese characters).

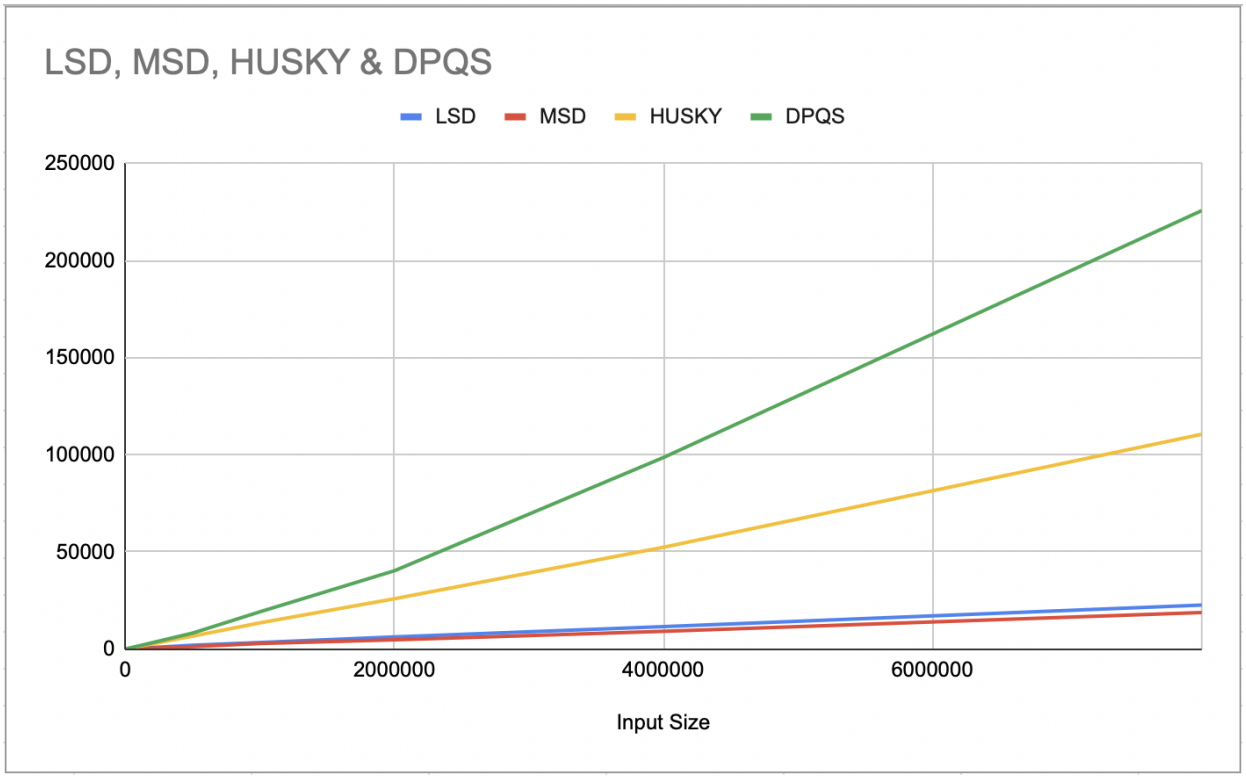
Evidence to support the conclusion:

- Terminal Output:



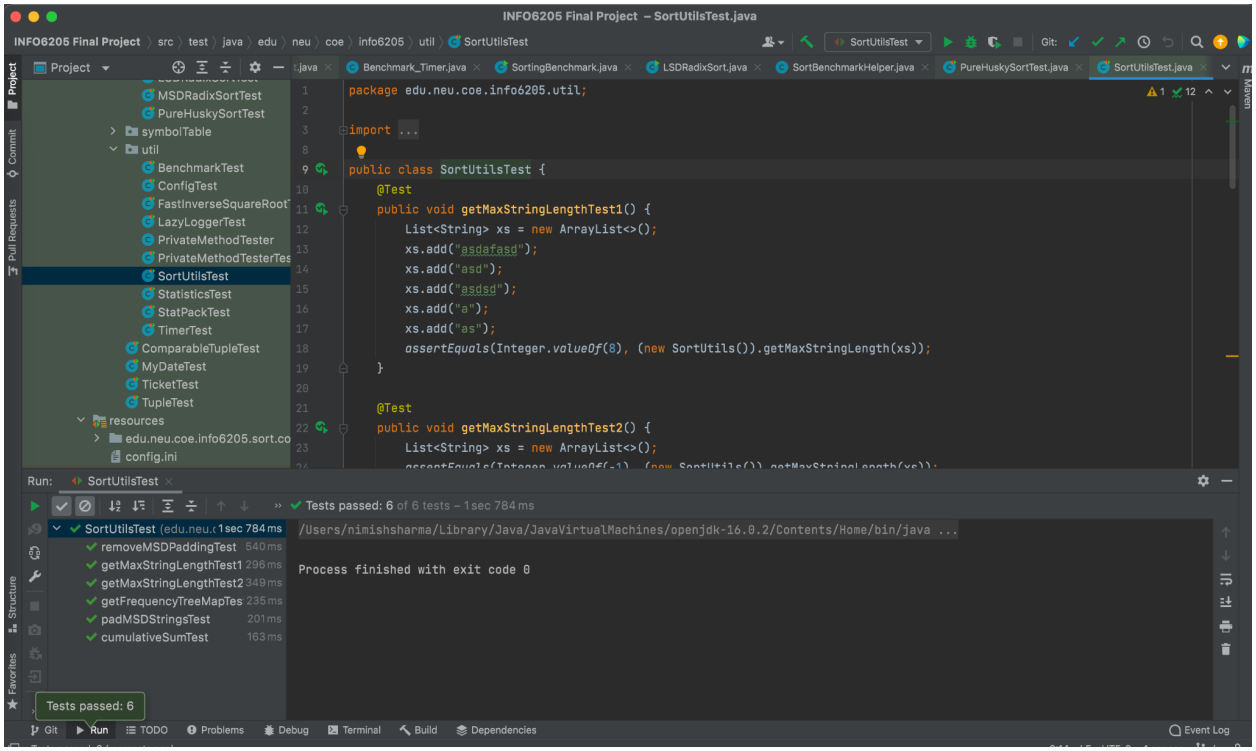
- Graphical Representation (Observations):

Input Size	LSD	MSD	HUSKY	DPQS
0	0	0	0	0
500000	1938	1221	6618	8211
1000000	3345	2819	13471	19163
2000000	6222	4713	25866	40290
4000000	11508	9097	52366	98572
8000000	22592	18761	110497	225578

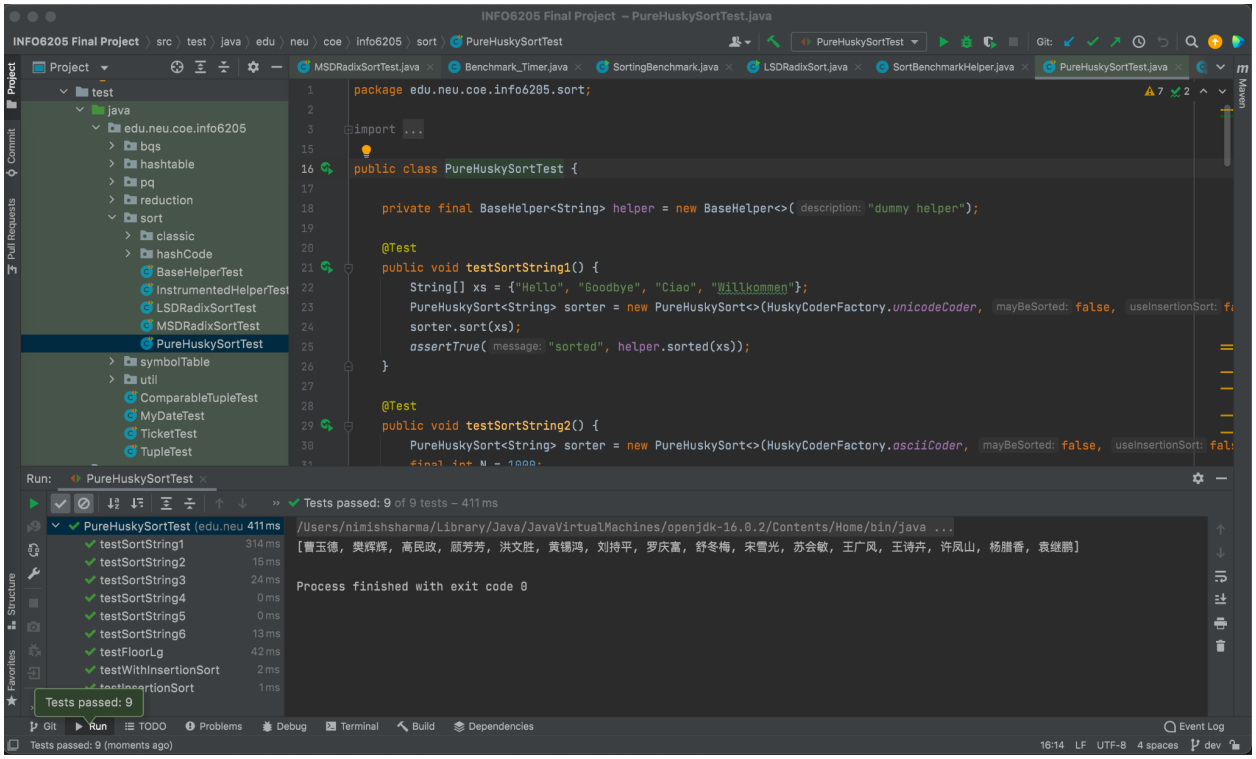


Test Cases:

- MSD & LSD Radix Sort Tests



- Pure Husky Sort Tests



- Dual Pivot Quicksort Tests