NORTHEASTERN UNIVERSITY

COLLEGE OF ENGINEERING

INFO 6205 – Program Structures and Algorithms

Document Control

Document Details	
Author	Anurag Parla
Author	Hiral Nagda
Author	Pranoti Dhabu
Title	MSD Radix sort for a natural language using unicode characters

Version and Distribution History			
Version #	Date	Description of Change	Author
1.0	12/04/2021	Initial Draft	Pranoti Dhabu
1.1	12/05/2021	Added Unicode chart	Pranoti Dhabu

Document Approvals

Name	Title	Signature	Date
Prof. Robin Hillyard	Associate Professor		

Contents

Table of Contents

Solution Overview

This solution implements sorting of strings of any natural language using Unicode characters and benchmarks the performance of various algorithms for a comparative study.

Summary

The main purpose of this solution is to implement Unicode encoding of strings and perform sorting using MSD radix sort, LSD radix sort, dual pivot quick sort, husky sort and timsort. Followed by benchmarking on a natural language with a data set up to 4M records.

MSD Radix sort

MSD Radix sort is a key-indexed counting sorting algorithm majorly used for sorting strings. The algorithm essentially creates a partition for each character of the sort-key while traversing from left to right and this method of sorting is applied to each partition recursively.

Background

Strings are used as the sort key in majority of applications and hence new algorithms were being researched and developed for effective and fast sorting to lower the time complexity. LSD and MSD Radix sort algorithms were developed looking at these constraints. Key-indexed sorting methodology has three main components – computing frequency counts, converting counts to indices and distributing the data.

As part of this project, we have modified algorithms proven to be efficient to sort natural language words based on Unicode value. We consider MSD radix sort, LSD sort, dual pivot quick sort, timsort and husky sort algorithm to compare their time complexity.

Experiment

In our experiment we have used primarily used Devanagari script (for Hindi language) as the natural language for sorting. Before benchmarking for Hindi language, we first established confidence by sorting English words and then proceeded towards native languages like Hindi, Gujarati (Sanskrit script) and simplified Chinese using collators for respective languages.

Further we benchmarked the five sorting algorithms for data set varying from size 1000 to 4M by doubling method.

We have re-used code from the class repository as well as Husky sort repository and made the necessary modifications for passing collator object as well Unicode conversion.

Assumptions

Following are the assumptions for our experiment:

1. The data set (corpus) is a partially random list of words. The corpus contains approx. 40k unique

words which were increased to 4M.

2. All experiments are performed on M1 MacBook Air machine. Any machine dependent factors determining running time will be common across all runs. (It was ensured the runs were performed on a fully charged and plugged-in machine with no applications open).

Challenges

Throughout the progress of our project, we came across below challenges:

1. We found difficulties in getting a clean corpus of Hindi words. While we tried cleaning up the data by writing small python scripts, not all corpus was cleaned. Hence, we decided to repeat the corpus of 40k words and increased it to 4M by duplicating words. Thus, the data set is partially random array of repeating sequences.
2. During benchmarking, since we are also timing pre-processing i.e., Unicode conversion the initial run for sorting 1k words took approx. 850ms for MSD Radix sort. After further analysis, it was observed the logic we used for array creation also included reading the file contents from disk which accounted for large run time. We decided to cache the same and the run time was reduced considerably to 9ms.

Running our experiment

We implemented benchmark java files for all the 5 sorting algorithms and ran the benchmark for data set varying from 1k to 4M.

Benchmarking numbers obtained are an average taken over 10 runs.

Conclusions from the experiment

We can derive the following conclusions from our experiments.

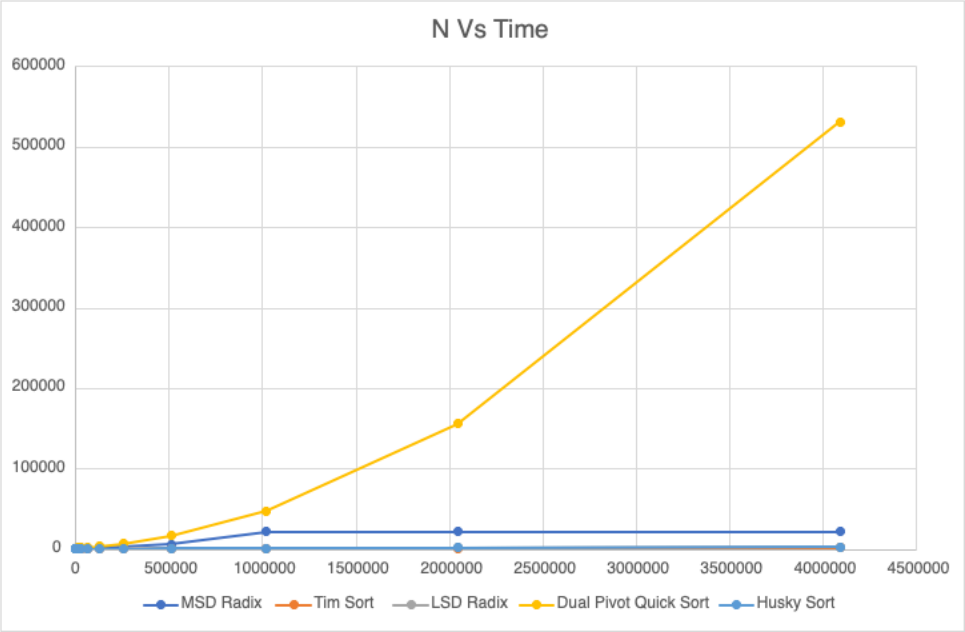
1. Time taken to sort the array of strings is directly proportional to the size of array and doubles as the input array size doubles.
2. Apart from dual pivot quick sort, all algorithms show a linearithmic growth graph.
3. We noticed log-log plot for MSD radix flattens after array size increases beyond 1M.
4. Log log plot indicates timsort, husky sort and LSD radix sort are converging as array size increases. Per the implementation of husky sort, as number of comparisons are less it is more efficient.

So, what these experiments mean,

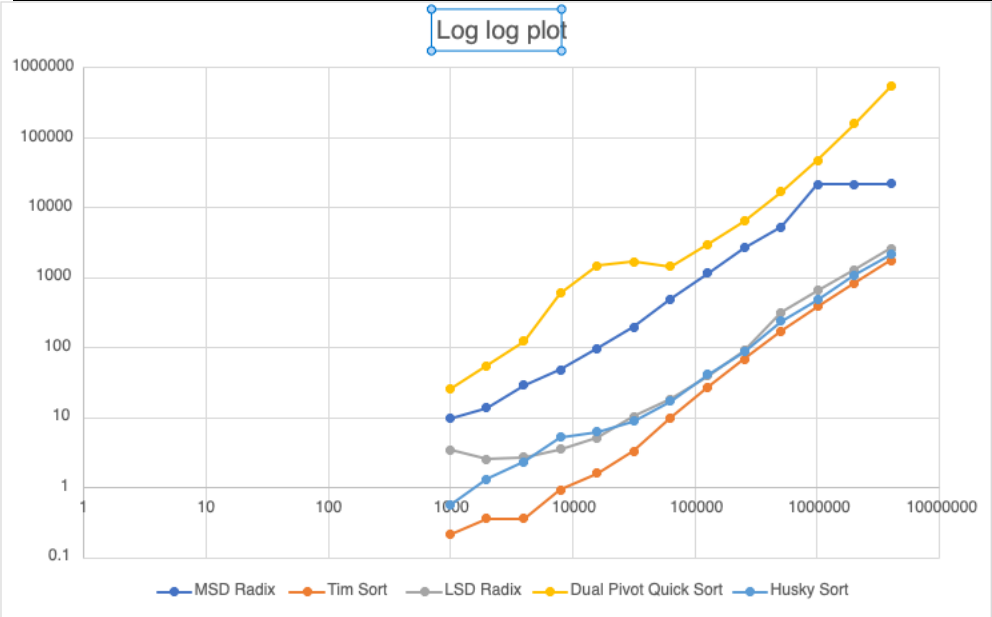
1. Husky sort has proven to be the most efficient algorithm for string sorting closely followed by timsort, LSD radix sort and MSD radix sort when input size is considerably large (over 1M).
2. For small size input, timsort is more efficient sorting algorithm.

Observations

The following observations were noted by our benchmarking runs.



Input Size	MSD Radix	Tim Sort	LSD Radix	Dual Pivot Quick Sort	Husky Sort
1000	9.6	0.21	3.46	25	0.55
2000	13.8	0.36	2.53	54.4	1.32
4000	28.9	0.36	2.68	121.8	2.31
8000	47.8	0.92	3.48	590	5.23
16000	96.1	1.58	5.11	1456.2	6.25
32000	196	3.3	10.2	1662.6	8.81
64000	493.2	9.86	18.22	1427.1	17.12
128000	1139.9	27.15	38.83	2919.4	40.62
256000	2639.8	67.5	90	6387.2	87.12
512000	5266.1	169.74	314.2	16549.2	238.11
1024000	21085.1	380.49	645.98	47029.8	480.24
2048000	21306.8	832.76	1288.6	155186	1078.07
4096000	21581.3	1739.78	2572.97	531048	2144.79

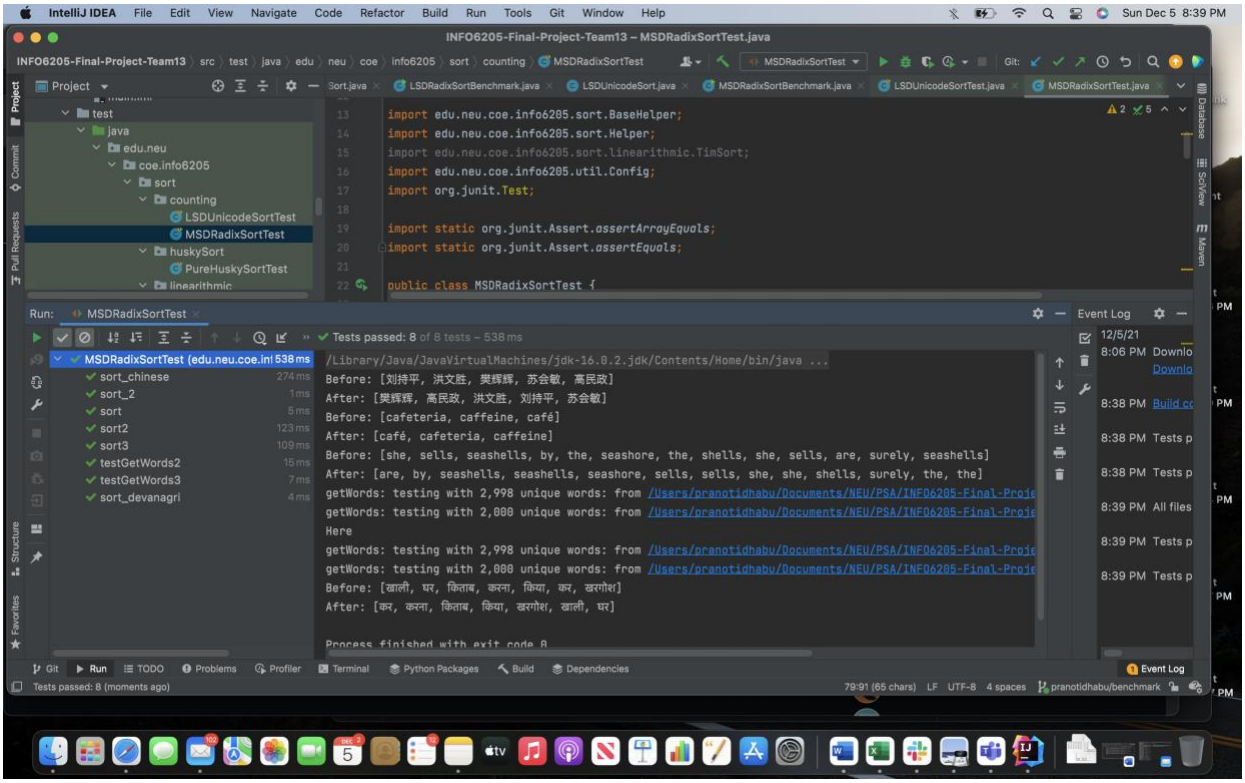

Log log plot
1000000
100000
10000
1000
100
10
1
0.1
1
10
100
1000
10000
100000
1000000
10000000
MSD Radix
Tim Sort
LSD Radix
Dual Pivot Quick Sort
Husky Sort

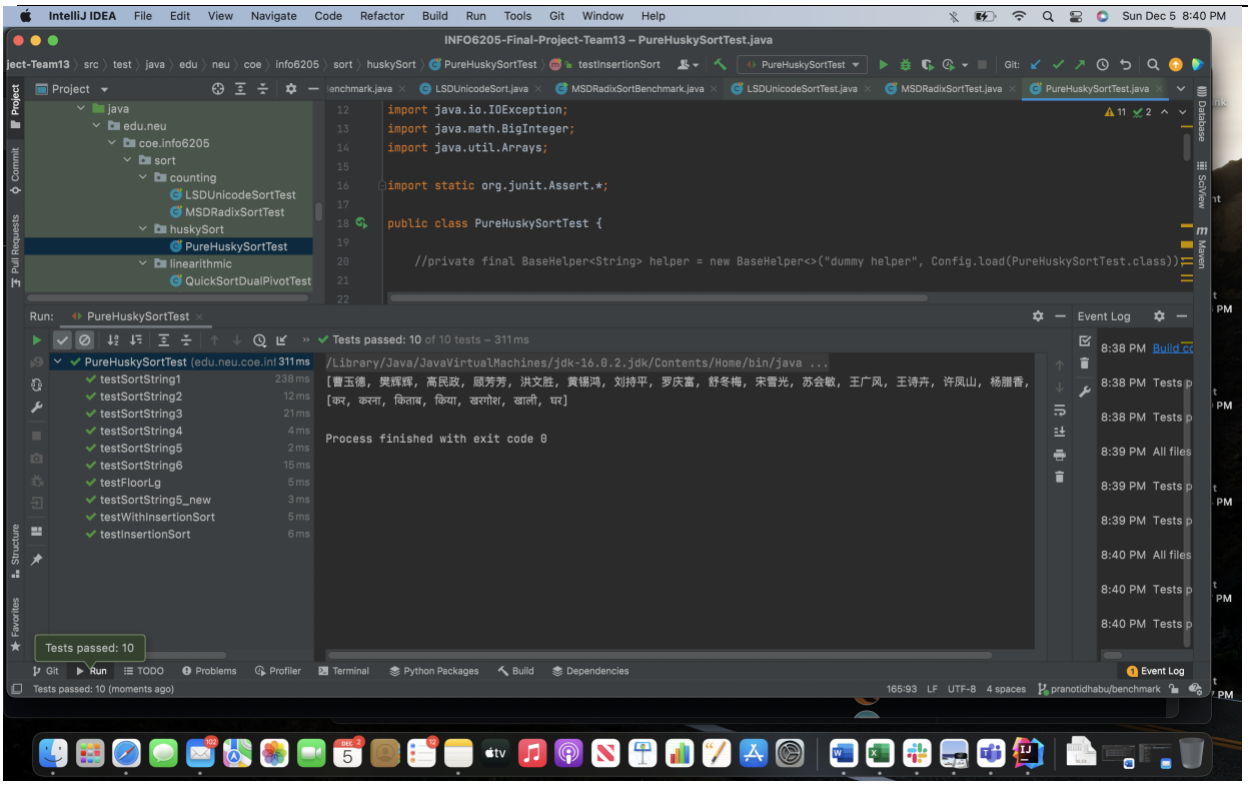
Conclusion

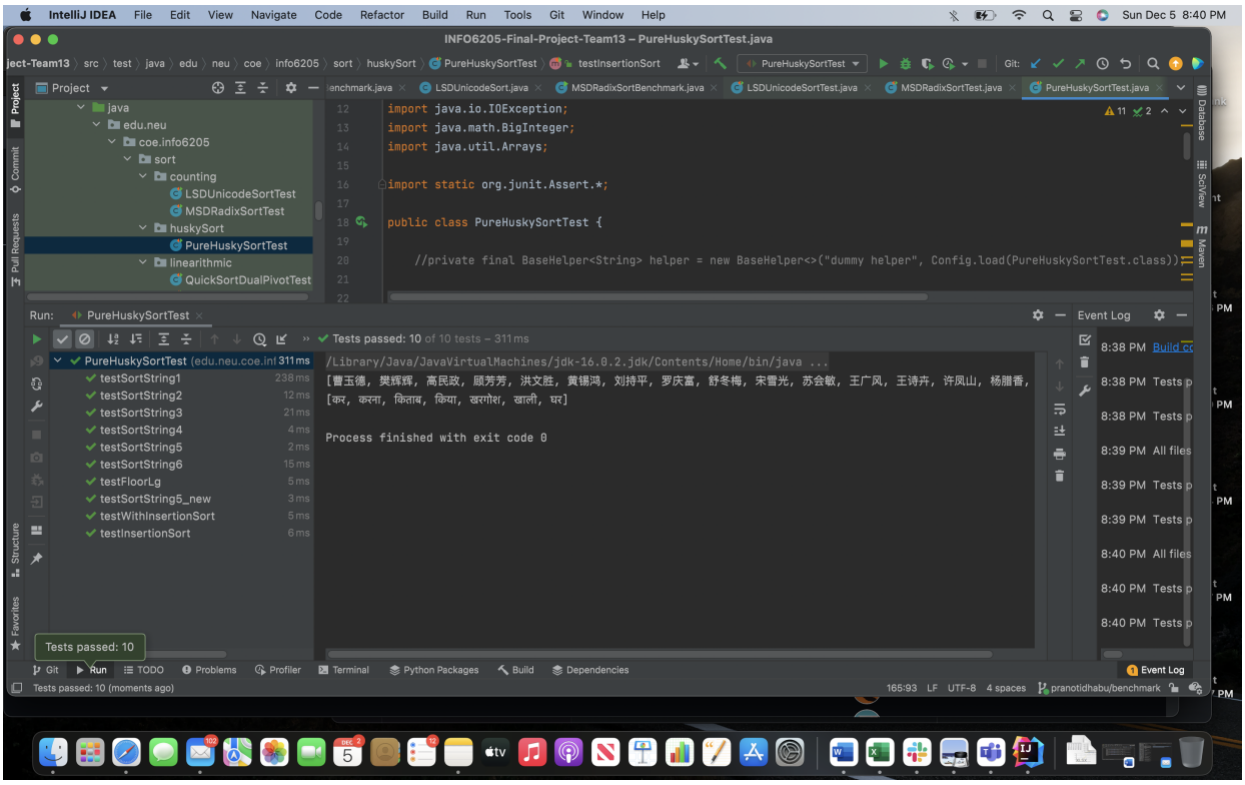
As demonstrated in the previous section, it is evident that dual pivot quick sort is not effective for sorting strings. Husky sort, MSD radix sort, timsort and LSD radix sort are proven to be efficient as per the benchmarking runs and graphs plotted.

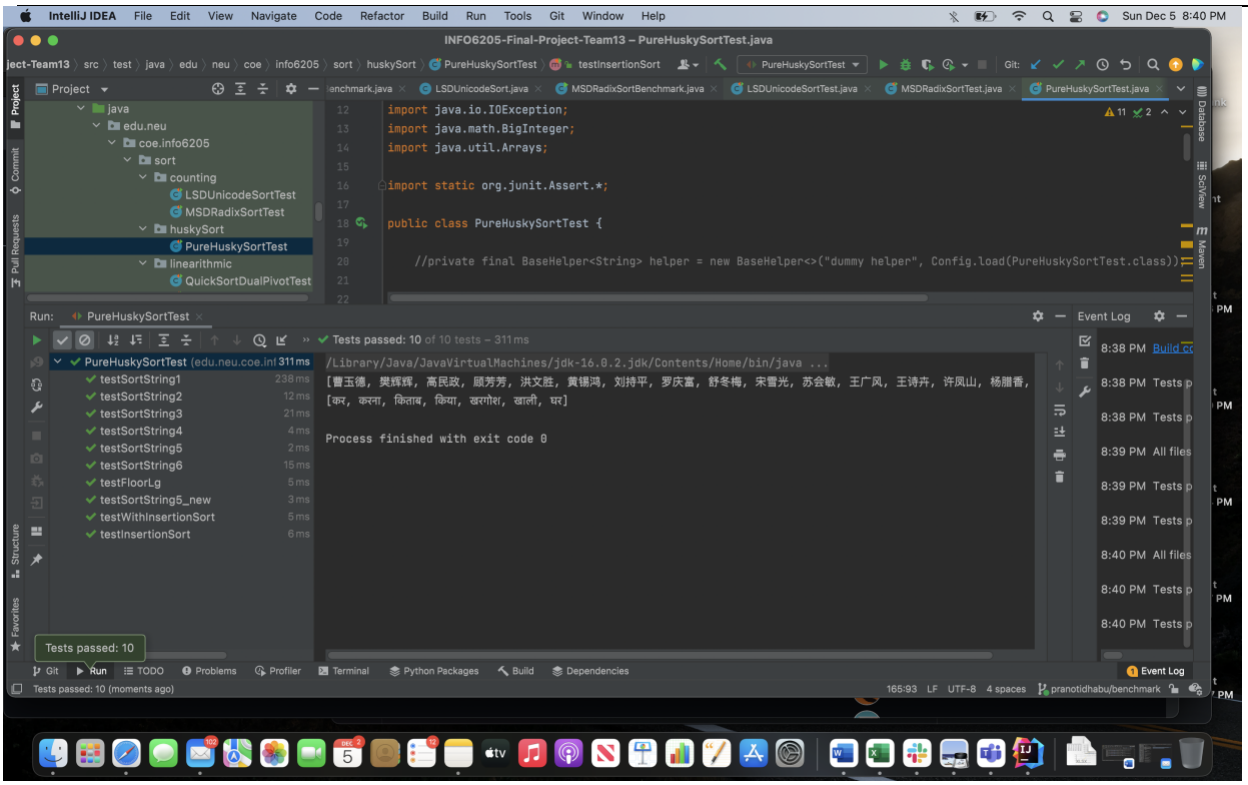
Unit Tests

The following unit tests will demonstrate the validity of our experiments.




IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Sun Dec 5 8:40 PM
INFO6205-Final-Project-Team13 – PureHuskySortTest.java
Project
java
edu.neu
coe.info6205
sort
counting
LSDUnicodeSortTest
MSDRadixSortTest
huskySort
PureHuskySortTest
linearithmic
QuickSortDualPivotTest
import java.io.IOException;
import java.math.BigInteger;
import java.util.Arrays;
import static org.junit.Assert.*;
public class PureHuskySortTest {
//private final BaseHelper<String> helper = new BaseHelper<>("dummy helper", Config.load(PureHuskySortTest.class));
Run: PureHuskySortTest
Tests passed: 10 of 10 tests – 311 ms
PureHuskySortTest (edu.neu.coe.inf 311 ms
testSortString1 238 ms
testSortString2 12 ms
testSortString3 21 ms
testSortString4 4 ms
testSortString5 2 ms
testSortString6 15 ms
testFloorLg 5 ms
testSortString5_new 3 ms
testWithInsertionSort 5 ms
testInsertionSort 6 ms
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java ...
[曹玉德, 樊辉辉, 高民政, 顾芳芳, 洪文胜, 黄锡鸿, 刘持平, 罗庆富, 舒冬梅, 宋雪光, 苏会敏, 王广凤, 王诗卉, 许凤山, 杨腊香,
[कर, करना, किताब, किया, खरगोश, खाली, घर]
Process finished with exit code 0
Event Log
8:38 PM Build
8:38 PM Tests p
8:38 PM Tests p
8:39 PM All files
8:39 PM Tests p
8:39 PM Tests p
8:40 PM All files
8:40 PM Tests p
8:40 PM Tests p
Tests passed: 10
Git Run TODO Problems Profiler Terminal Python Packages Build Dependencies
Event Log
Tests passed: 10 (moments ago)
165:93 LF UTF-8 4 spaces pranotidhabu/benchmark


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Sun Dec 5 8:40 PM
INFO6205-Final-Project-Team13 – PureHuskySortTest.java
Project
java
edu.neu
coe.info6205
sort
counting
LSDUnicodeSortTest
MSDRadixSortTest
huskySort
PureHuskySortTest
linearithmic
QuickSortDualPivotTest
import java.io.IOException;
import java.math.BigInteger;
import java.util.Arrays;
import static org.junit.Assert.*;
public class PureHuskySortTest {
//private final BaseHelper<String> helper = new BaseHelper<>("dummy helper", Config.load(PureHuskySortTest.class));
Run: PureHuskySortTest
Tests passed: 10 of 10 tests – 311 ms
PureHuskySortTest (edu.neu.coe.inf 311 ms
testSortString1 238 ms
testSortString2 12 ms
testSortString3 21 ms
testSortString4 4 ms
testSortString5 2 ms
testSortString6 15 ms
testFloorLg 5 ms
testSortString5_new 3 ms
testWithInsertionSort 5 ms
testInsertionSort 6 ms
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java ...
[曹玉德, 樊辉辉, 高民政, 顾芳芳, 洪文胜, 黄锡鸿, 刘持平, 罗庆富, 舒冬梅, 宋雪光, 苏会敏, 王广凤, 王诗卉, 许凤山, 杨腊香,
[कर, करना, किताब, किया, खरगोश, खाली, घर]
Process finished with exit code 0
Event Log
8:38 PM Build
8:38 PM Tests p
8:38 PM Tests p
8:39 PM All files
8:39 PM Tests p
8:39 PM Tests p
8:40 PM All files
8:40 PM Tests p
8:40 PM Tests p
Tests passed: 10
Git Run TODO Problems Profiler Terminal Python Packages Build Dependencies
Event Log
Tests passed: 10 (moments ago)
165:93 LF UTF-8 4 spaces pranotidhabu/benchmark


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Sun Dec 5 8:40 PM
INFO6205-Final-Project-Team13 – PureHuskySortTest.java
Project
java
edu.neu
coe.info6205
sort
counting
LSDUnicodeSortTest
MSDRadixSortTest
huskySort
PureHuskySortTest
linearithmic
QuickSortDualPivotTest
import java.io.IOException;
import java.math.BigInteger;
import java.util.Arrays;
import static org.junit.Assert.*;
public class PureHuskySortTest {
//private final BaseHelper<String> helper = new BaseHelper<>("dummy helper", Config.load(PureHuskySortTest.class));
Run: PureHuskySortTest
Tests passed: 10 of 10 tests – 311 ms
PureHuskySortTest (edu.neu.coe.inf 311 ms
testSortString1 238 ms
testSortString2 12 ms
testSortString3 21 ms
testSortString4 4 ms
testSortString5 2 ms
testSortString6 15 ms
testFloorLg 5 ms
testSortString5_new 3 ms
testWithInsertionSort 5 ms
testInsertionSort 6 ms
/Library/Java/JavaVirtualMachines/jdk-16.0.2.jdk/Contents/Home/bin/java ...
[曹玉德, 樊辉辉, 高民政, 顾芳芳, 洪文胜, 黄锡鸿, 刘持平, 罗庆富, 舒冬梅, 宋雪光, 苏会敏, 王广凤, 王诗卉, 许凤山, 杨腊香,
[कर, करना, किताब, किया, खरगोश, खाली, घर]
Process finished with exit code 0
Event Log
8:38 PM Build co
8:38 PM Tests p
8:38 PM Tests p
8:39 PM All files
8:39 PM Tests p
8:39 PM Tests p
8:40 PM All files
8:40 PM Tests p
8:40 PM Tests p
Tests passed: 10
Git Run TODO Problems Profiler Terminal Python Packages Build Dependencies
Event Log
Tests passed: 10 (moments ago)
165:93 LF UTF-8 4 spaces pranotidhabu/benchmark

Appendix – A – References

Titles	Description/Link
Husky sort repo	https://github.com/rchillyard/The-repository-formerly-known-as
Class repo	https://github.com/rchillyard/INFO6205
Hindi corpus	https://wortschatz.uni-leipzig.de/en/download/Hindi
Textbook (for code references)	*Algorithms*, 4th Edition by Robert Sedgewick and Kevin Wayne, Addison-Wesley Professional, 2011, ISBN 9780321573513
Devanagari Unicode	https://unicode.org/charts/PDF/U0900.pdf

Appendix – B – Devanagari unicode chart

	090	091	092	093	094	095	096	097
0	ँ 0900	ऐ 0910	ठ 0920	र 0930	ी 0940	ॐ 0950	ॠ 0960	॰ 0970
1	ं 0901	ऑ 0911	ड 0921	ऱ 0931	ु 0941	॑ 0951	ॡ 0961	ॱ 0971
2	ः 0902	ओ 0912	ढ 0922	ल 0932	ू 0942	॒ 0952	ॢ 0962	ॲ 0972
3	ः 0903	ओ 0913	ण 0923	ळ 0933	ृ 0943	॓ 0953	ॣ 0963	ॳ 0973
4	अ 0904	औ 0914	त 0924	ऴ 0934	ॄ 0944	॔ 0954	। 0964	ॴ 0974
5	आ 0905	क 0915	थ 0925	व 0935	ॅ 0945	ॕ 0955	॥ 0965	ॵ 0975
6	इ 0906	ख 0916	द 0926	श 0936	ॆ 0946	ॖ 0956	० 0966	ॶ 0976
7	ई 0907	ग 0917	ध 0927	ष 0937	े 0947	ॗ 0957	१ 0967	ॷ 0977
8	उ 0908	घ 0918	न 0928	स 0938	ै 0948	क़ 0958	२ 0968	ॸ 0978
9	ऊ 0909	ङ 0919	ऩ 0929	ह 0939	ॉ 0949	ख़ 0959	३ 0969	ॹ 0979
A	ऋ 090A	च 091A	प 092A	ऺ 093A	ॊ 094A	ग़ 095A	४ 096A	ॺ 097A
B	ऌ 090B	छ 091B	फ 092B	ऻ 093B	ो 094B	ज़ 095B	५ 096B	ॻ 097B
C	ऍ 090C	ज 091C	ब 092C	़ 093C	ौ 094C	ड़ 095C	६ 096C	ॼ 097C
D	ऎ 090D	झ 091D	भ 092D	ऽ 093D	् 094D	ढ़ 095D	७ 096D	ॽ 097D
E	ए 090E	ञ 091E	म 092E	ा 093E	ॎ 094E	फ़ 095E	८ 096E	ॾ 097E
F	ऐ 090F	ट 091F	य 092F	ि 093F	ॏ 094F	य़ 095F	९ 096F	ॿ 097F

The Unicode Standard 14.0, Copyright © 1991-2021 Unicode, Inc. All rights reserved.