Yang Song (001003647) – Sec2

Jing Dai (001569042) – Sec6

Program Structures & Algorithms

Fall 2021

Team Project

- ## Task (List down the tasks performed in the Assignment)

- Implement MSD radix sort for Chinese
- Compare with Tim sort, Dual-pivot Quicksort, Husky sort, and LSD radix sort

- ## Relationship Conclusion:

According to benchmark result, the average running time of these sorts are:
T(husky sort) < T(LSD sort) < T(Tim sort) < T(Dual pivot quick sort) < T(MSD sort)
Husky sort has the best performance of sorting Chinese, MSD radix sort behaves worse than other sorting algorithms.

- ## Evidence to support the conclusion:

1. Explanation of working

To sort Chinese, we choose to use Collator class of icu4j which is an enhancement of "java.text.Collator". All Chinese sorts java files are in "edu.neu.coe.info6205.sort.chinese". And we output first 2000 sorted Chinese words to the resources directory called "sortedChineseWords-first-2000.txt" to check the efficiency of the MSD sort.

For comparison sort like Tim sort, Dual-pivot Quicksort. We use compare method in Collator to sort Chinese words. The detail implementation is in edu.neu.coe.info6205.sort.CollatorHelper.

```
protected Collator collator;

{
    collator = Collator.getInstance(Locale.CHINA);
}
```

```
public int compare(X[] xs, int i, int j) {
    // CONSIDER invoking the other compare signature
    return collator.compare(xs[i], xs[j]);
}
```

For non-comparison sort like MSD、LSD sort. We use #getCollationKey().toByteArray() method to create a new order for each Chinese character.

```
private static int ChineseCharAt(String s, int d) {
    if (d < s.length()) {
        byte[] bytes = collator.getCollationKey(String.valueOf(s.charAt(d))).toByteArray();
        if (bytes.length < 7) {
            return (bytes[0] & 0xFF) * 255;
        } else {
            return (bytes[0] & 0xFF) * 255 + (bytes[1] & 0xFF);
        }
    } else return -1;
}
```

It turns out that bytes[0] plus bytes[1] will decide the order of the Chinese character, and the max number of each byte value is 255. But we cannot simply multiply bytes[0] and bytes[1], since bytes[0] firstly decides the order and then it's bytes[1] turn. So we must give bytes[0] priority, multiplying 255 is a way to solve the priority problem. Finally we convert the byte number to int number to fit MSD/LSD sort method, and then replace #charAt() method.

```
private static void sort(String[] a, int lo, int hi, int d) {
    if (hi < lo + cutoff) InsertionSortMSD.sort(a, lo, hi, d);
    else {
        int[] count = new int[radix + 2];        // Compute frequency counts.
        for (int i = lo; i < hi; i++) {
            count[ChineseCharAt(a[i], d) + 2]++;
        }
        for (int r = 0; r < radix + 1; r++)      // Transform counts to indices.
            count[r + 1] += count[r];
        for (int i = lo; i < hi; i++)     // Distribute.
            aux[count[ChineseCharAt(a[i], d) + 1]++] = a[i];
        // Copy back.
        if (hi - lo >= 0) {
            System.arraycopy(aux, srcPos: 0, a, lo, length: hi - lo);
        }
        // Recursively sort for each character value.
        // TO BE IMPLEMENTED
        for (int r = 0; r < radix; r++) {
            sort(a, lo: lo + count[r], hi: lo + count[r + 1], d: d + 1);
        }

    }
}
```

For Husky sort, we use the same method as MSD sort: #getCollationKey().toByteArray() method to acquire the sequence of Chinese character and convert into long type. And after that operation, we also use the compare method in Collator to sort with Pinyin order.

2. **Output**

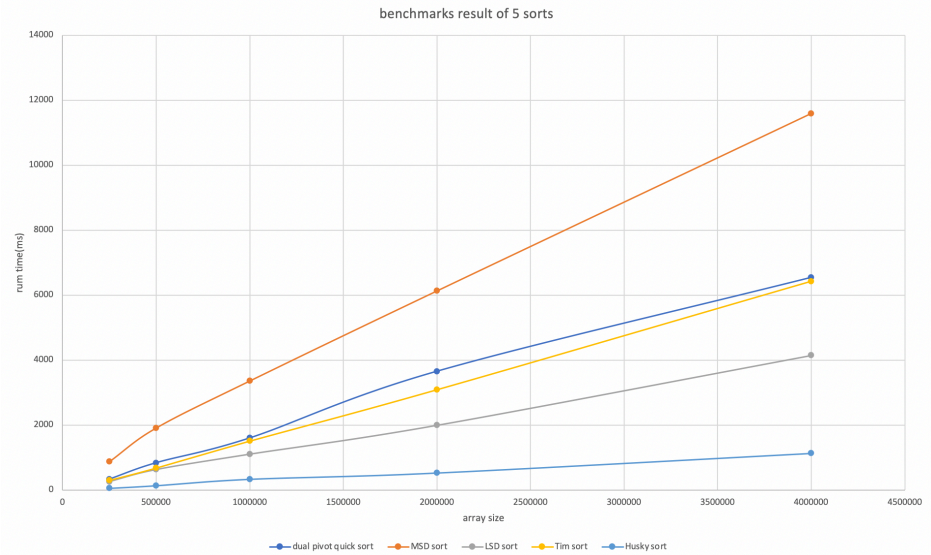
We use benchmarks method in assignment2 to test these sorts. And the array size is from 250k – 4M. (edu.neu.coe.info6205.util.BenchmarkResult)

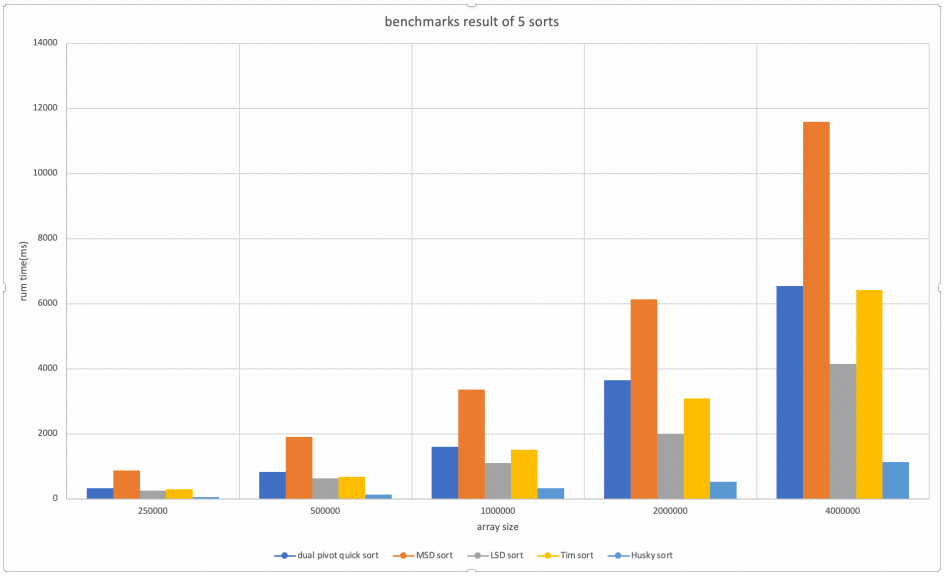
```
getWords: testing with 1,000,000 unique words: from /Users/songyang/IdeaProjects/INFO6205-Final-Project/target/classes/shuffledChinese.txt
----------array size is 250000----------
2021-11-28 18:19:25 INFO  Benchmark - Begin run: Chinese dual pivot quick sort with 5 runs
when n is 5, run time is 328.1868106
2021-11-28 18:19:27 INFO  Benchmark - Begin run: MSD Chinese sort with 5 runs
when n is 5, run time is 867.2820422000001
2021-11-28 18:19:34 INFO  Benchmark - Begin run: LSD Chinese sort with 5 runs
when n is 5, run time is 256.1526946
2021-11-28 18:19:35 INFO  Benchmark - Begin run: Chinese Tim sort with 5 runs
when n is 5, run time is 295.2662282
2021-11-28 18:19:38 INFO  Benchmark - Begin run: Chinese husky sort with 5 runs
when n is 5, run time is 52.6795702
getWords: testing with 1,000,000 unique words: from /Users/songyang/IdeaProjects/INFO6205-Final-Project/target/classes/shuffledChinese.txt
----------array size is 500000----------
2021-11-28 18:19:39 INFO  Benchmark - Begin run: Chinese dual pivot quick sort with 5 runs
when n is 5, run time is 832.1055478000001
2021-11-28 18:19:45 INFO  Benchmark - Begin run: MSD Chinese sort with 5 runs
when n is 5, run time is 1904.4873892
2021-11-28 18:19:59 INFO  Benchmark - Begin run: LSD Chinese sort with 5 runs
when n is 5, run time is 625.3304449999999
2021-11-28 18:20:03 INFO  Benchmark - Begin run: Chinese Tim sort with 5 runs
when n is 5, run time is 675.6229574
2021-11-28 18:20:08 INFO  Benchmark - Begin run: Chinese husky sort with 5 runs
when n is 5, run time is 133.2183184
getWords: testing with 1,000,000 unique words: from /Users/songyang/IdeaProjects/INFO6205-Final-Project/target/classes/shuffledChinese.txt
----------array size is 1000000----------
2021-11-28 18:20:09 INFO  Benchmark - Begin run: Chinese dual pivot quick sort with 5 runs
when n is 5, run time is 1596.007458
2021-11-28 18:20:20 INFO  Benchmark - Begin run: MSD Chinese sort with 5 runs
when n is 5, run time is 3352.8166905999997
2021-11-28 18:20:44 INFO  Benchmark - Begin run: LSD Chinese sort with 5 runs
when n is 5, run time is 1099.4700626
2021-11-28 18:20:52 INFO  Benchmark - Begin run: Chinese Tim sort with 5 runs
when n is 5, run time is 1504.2389005999999
2021-11-28 18:21:02 INFO  Benchmark - Begin run: Chinese husky sort with 5 runs
when n is 5, run time is 329.7977596
getWords: testing with 1,000,000 unique words: from /Users/songyang/IdeaProjects/INFO6205-Final-Project/target/classes/shuffledChinese.txt
----------array size is 2000000----------
2021-11-28 18:21:05 INFO  Benchmark - Begin run: Chinese dual pivot quick sort with 5 runs
when n is 5, run time is 3650.2534934000005
2021-11-28 18:21:31 INFO  Benchmark - Begin run: MSD Chinese sort with 5 runs
when n is 5, run time is 6125.1808704000005
2021-11-28 18:22:13 INFO  Benchmark - Begin run: LSD Chinese sort with 5 runs
when n is 5, run time is 1988.8807256
2021-11-28 18:22:28 INFO  Benchmark - Begin run: Chinese Tim sort with 5 runs
when n is 5, run time is 3085.868225
2021-11-28 18:22:49 INFO  Benchmark - Begin run: Chinese husky sort with 5 runs
when n is 5, run time is 523.042688
getWords: testing with 1,000,000 unique words: from /Users/songyang/IdeaProjects/INFO6205-Final-Project/target/classes/shuffledChinese.txt
----------array size is 4000000----------
2021-11-28 18:22:53 INFO  Benchmark - Begin run: Chinese dual pivot quick sort with 5 runs
when n is 5, run time is 6541.859975400001
2021-11-28 18:23:40 INFO  Benchmark - Begin run: MSD Chinese sort with 5 runs
when n is 5, run time is 11589.5494942
2021-11-28 18:25:02 INFO  Benchmark - Begin run: LSD Chinese sort with 5 runs
when n is 5, run time is 4142.1860142
2021-11-28 18:25:31 INFO  Benchmark - Begin run: Chinese Tim sort with 5 runs
when n is 5, run time is 6423.5621642
2021-11-28 18:26:16 INFO  Benchmark - Begin run: Chinese husky sort with 5 runs
when n is 5, run time is 1123.6951291999999
```

3. **Graphical Representation**

Average run time with 5 runs, time(ms) results are

array size	dual pivot quick sort	MSD sort	LSD sort	Tim sort	Husky sort
250000	328.1868106	867.2820422	256.1526946	295.2662282	52.67957
500000	832.1055478	1904.487389	625.330445	675.6229574	133.21832
1000000	1596.007458	3352.816691	1099.470063	1504.238901	329.79776
2000000	3650.253493	6125.18087	1988.880726	3085.868225	523.04269
4000000	6541.859975	11589.54949	4142.186014	6423.562164	1123.6951





From the chart above, we can conclude that the best performance is husky sort, LSD sort is better than Tim sort and Dual pivot quick sort which have similar performance, MSD sort behaves worse than other sorts when sorting Chinese words.

4. **Unit tests result**

We output first 2000 sorted Chinese words to the resources directory called “sortedChineseWords-first-2000.txt” to check the efficiency of the MSD sort. (edu.neu.coe.info6205.sort.chinese)

```java
@Test
public void sort2() {
    String[] words = GetWordsUtil.getWords( resource: "/shuffledChinese.txt", GetWordsUtil::lineAsList,
            MSDChineseSortTest.class);
    String[] res = MSDChineseSort.sort(words);
    String[] sorted = Arrays.copyOfRange(res,  from: 0,  to: 2000);
    GetWordsUtil.writeWords(sorted, String.join(File.separator,  ...elements: "src", "main", "resources",
            "sortedChineseWords-first-2000.txt"));
    assertEquals( expected: "阿安", res[0]);
}
```

✔ CollatorHelperTest (edu.neu.coe.info6205.sort)	211 ms
✔ close	0 ms
✔ compare	0 ms
✔ getDescription	0 ms
✔ getSetN	0 ms
✔ instrumented	0 ms
✔ inversions	1 ms
✔ less	0 ms
✔ postProcess	0 ms
✔ random	49 ms
✔ sorted	1 ms
✔ swap	0 ms
✔ swapConditional	4 ms
✔ swapStable	156 ms
✔ swapStableConditional	0 ms
✔ testToString	0 ms
✔ BenchmarkResultTest (edu.neu.coe.info6205.util)	11 ms
✔ extendArrayTest	11 ms
✔ LSDChineseSortTest (edu.neu.coe.info6205.sort.chinese)	250 ms
✔ preProcessTest	0 ms
✔ sort	168 ms
✔ sort1	82 ms
✔ MSDChineseSortTest (edu.neu.coe.info6205.sort.chinese)	4 s 995 ms
✔ preProcessTest	0 ms
✔ sort	162 ms
✔ sort1	81 ms
✔ sort2	4 s 752 ms
✔ PureHuskyChineseSortTest (edu.neu.coe.info6205.sort.chinese)	2 s 610 ms
✔ preProcessTest	0 ms
✔ sort	2 s 414 ms
✔ testSort	1 ms
✔ testSortString	190 ms
✔ testWithInsertionSort	5 ms
✔ QuickSortChineseDualPivotTest (edu.neu.coe.info6205.sort.chinese)	306 ms
✔ sort	4 ms
✔ sort1	18 ms
✔ testSortWithChinesePartition	284 ms
✔ TimChineseSortTest (edu.neu.coe.info6205.sort.chinese)	245 ms
✔ preProcessTest	0 ms
✔ sort	225 ms
✔ sort1	20 ms