Wenzhuo Wang (1562172)

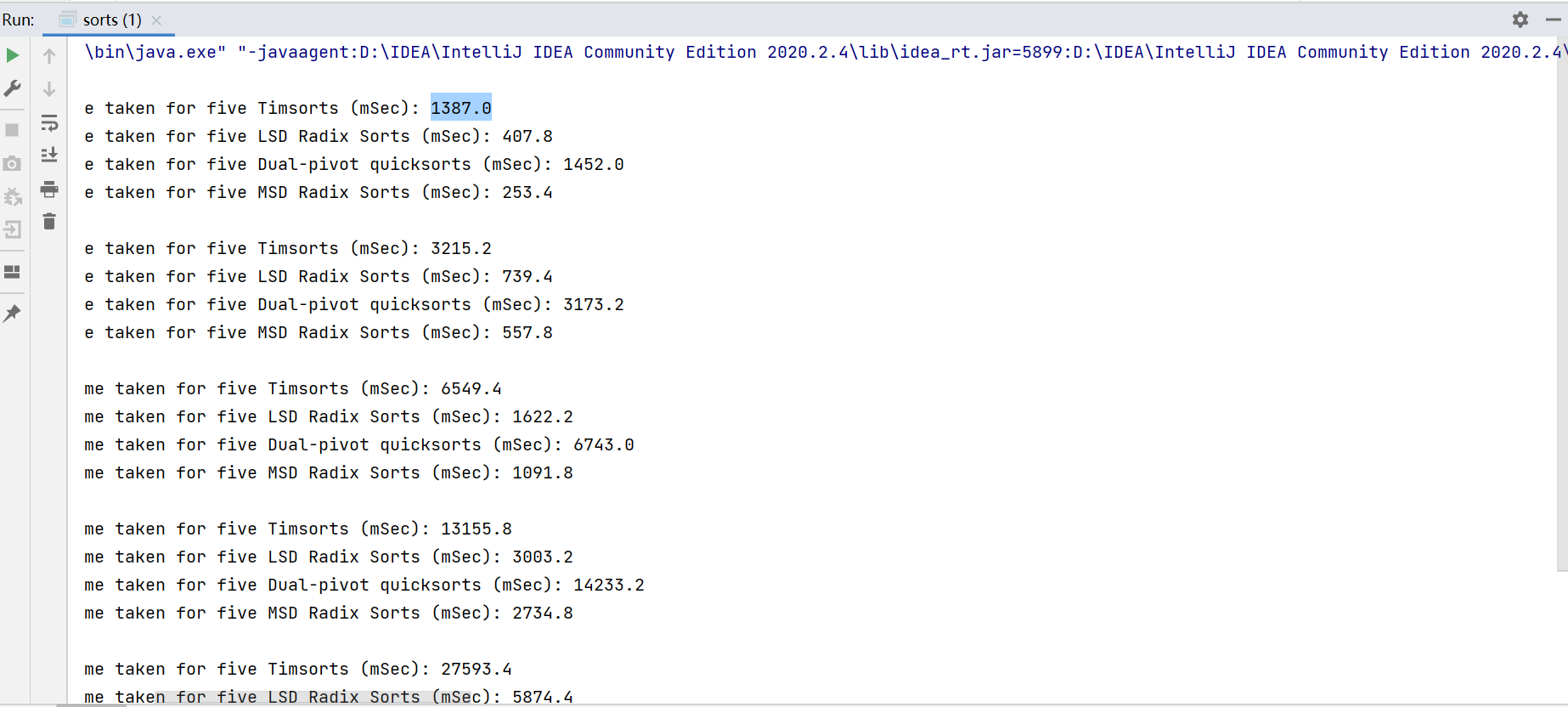
**Program Structures & Algorithms**

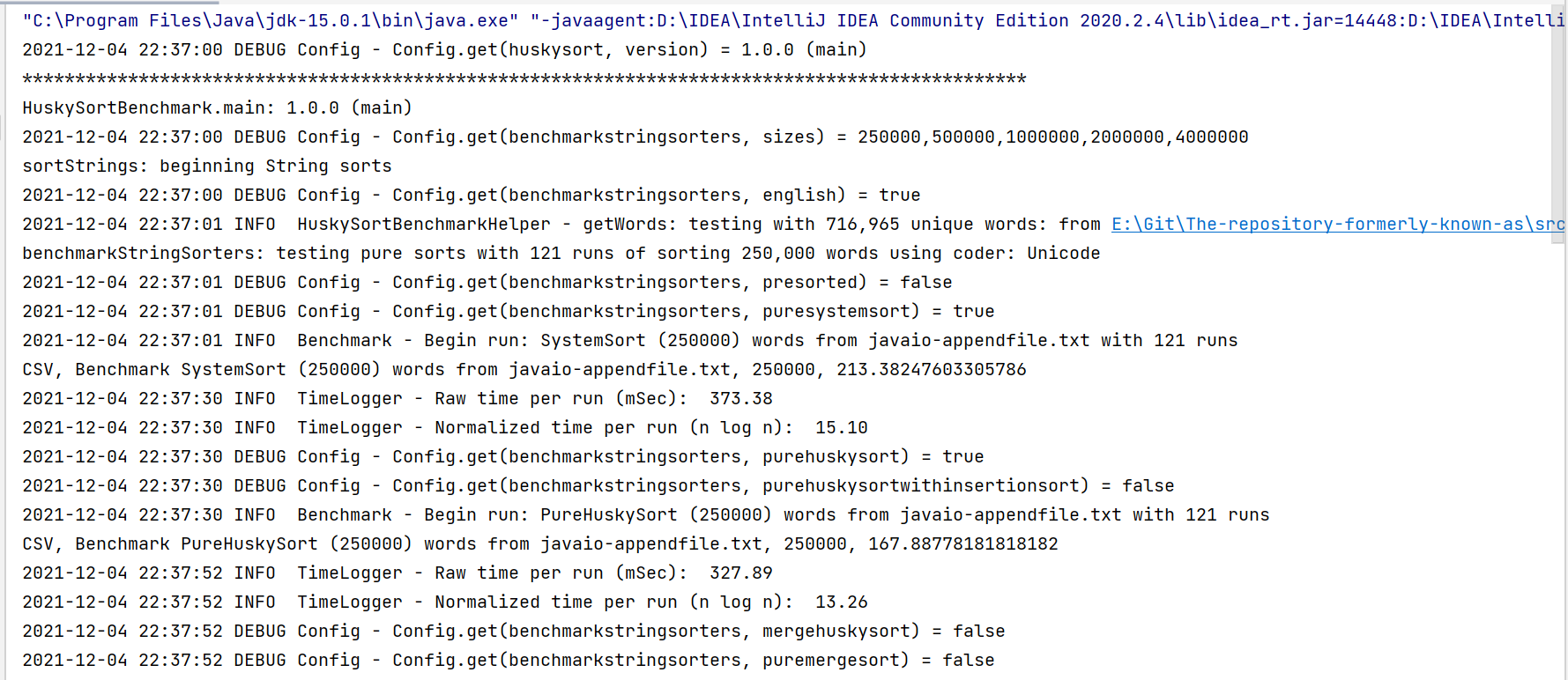
**Fall 2021**

**Final Project report**

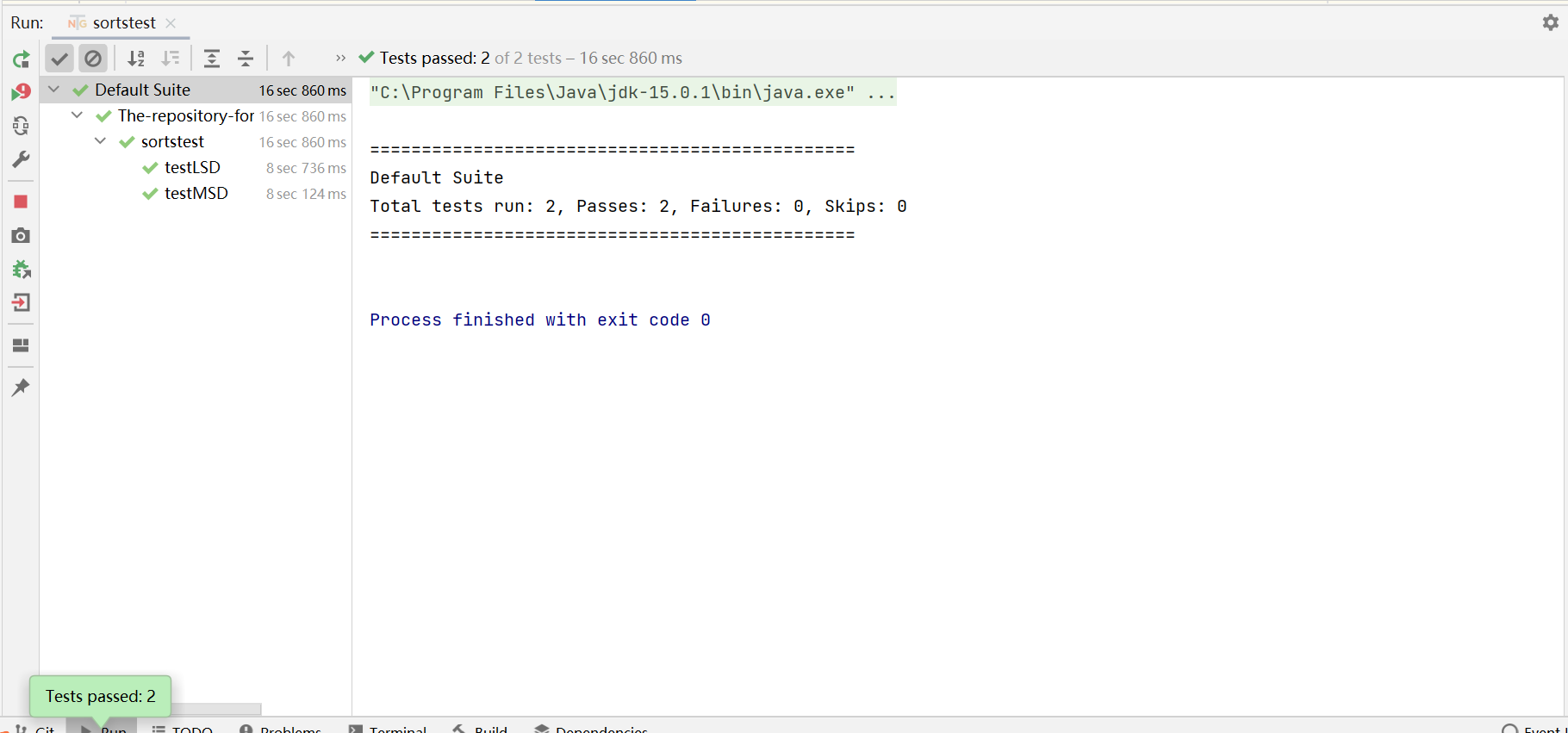
* **Task**
* Use MSD radix sort method to sort Unicode Chinese names in pinyin. Complete the literature review on MSD radix sort and compare it with timport, dual pivot quicksort, huskysport, and LSD radix sort.
* **Relationship Conclusion:**
* When sorting Chinese strings in pinyin form, MSD radixsort is one of the fastest sorting methods. The LSD radixsort method is slightly slower than MSD radixsort (100 to 300 milliseconds slower when sorting strings with a length of 1m). Because these strings vary in length, MSD radix sort is slightly faster than LSD radix sort.
* The time-consuming of dual pivot quicksort method is close to that of timport method, and the time-consuming of both methods is more than four times that of radix sort method.
* Huskysport cannot sort Chinese in pinyin form. The experimental results are obtained by huskysport's time-consuming sorting of Pinyin strings (English letters + numbers) and the time-consuming simulation of changing Chinese character strings into pinyin strings. Huskysport is very close to MSD radix sort.
* As the number of strings increases, the time-consuming of several sorting methods is linear. The ratio parameter of nlgn remains stable. Huskysport and MSD radix sort parameters are the smallest, LSD radix sort parameters are the second, and timsport is the largest.
* **Evidence to support the conclusion:**

1. **Output**





1. **Unit test**



* Two unit tests are used to verify that the strings can be sorted correctly with the radix sort method according to the pinyin form

1. **Graphical Representation**

* The original data are saved in file project.xlsx

**Literature Review**

*Introduction*

The high efficiency advantage of Radix sorting method is based on its occupation of a large amount of memory resources. In order to reduce the space complexity of cardinality sorting, [1] proposed adding adaptive function and in place sorting capability to MSD radixsort to reduce memory consumption. This method not only reduces the spatial complexity, but also makes the algorithm more friendly to cache than quicksort algorithm, which is in line with the development trend of modern computer processors to expand cache at all levels. Therefore, the author believes that the author of MSD in place radioport ([1] named it ARL-adaptive left Radix) is a more universal algorithm than quicksort. ARL uses a different loop arrangement of keys, which is called permutation loop.

In [2], MSL (map shuffle loop) algorithm is proposed. MSL algorithm is an improvement of ARL algorithm. Instead of seeking each key to enter the destination group during traversal, searches for the next element to insert group by group. This strengthens the adaptability of the algorithm, so that it may use different number sizes for different calls.

*Components*

In ARL, arrays are sorted by permutation cycles. When subsequences are short, insert sort is used for internal sorting. Since we use permutation cycles to exchange elements instead of copying back and forth between arrays, ARL algorithm is unstable [1]. Due to the attribute of in place sort, ARL can reduce the memory consumption by half compared with MSD radixsort.

In addition to reducing memory consumption, MSD in place radixsort is also cache friendly. The processor cache of modern computers is linked to memory after hierarchical distribution, and the process of data search is also level by level search. MSD in place radixsort is suitable for this processor caching mode due to the trend of serialization. (according to [1], if the array is completely random, the data will be much larger than the cache capacity, and the algorithm will spend more time on the communication between caches at all levels).

Quicksort is a cache friendly algorithm, but when the number of sorting elements is large, quicksort takes a long time to sort so that the data set is small enough to be stored in the cache. MSD radixsort algorithm needs to call two sequential reads and one random write in each traversal. In addition, it also needs to build two sequences with length N, which makes the traditional MSD radixsort algorithm cache unfriendly.

The ARL algorithm divides the data into multiple separate sub data sets, which makes some data sets can be stored in the cache, and even the fastest L1 cache and L2 cache.

In [2], MSL algorithm improves permutation cycles (loops) in ARL algorithm to make MSL algorithm run faster. The specific method is to provide shift value and the mask value for each key. Shift value is used for right shifting keys. Mask value is used to extract the bucket number. According to [2], MSL selects the first element in the original group as the root of the replacement cycle (exchange cycle). Each permutation cycle starts and ends with the current root key, which is the first element in the original group that is not in its correct position. At every step of the replacement cycle, exchange occurs. The current key is exchanged with the component in the correct position of the current key. Initially, the current key is the same as the current root key.

Conditionally test whether the current key destination address is in the source group. Once the permutation cycle begins, the array element at the initial address of the root key is invalid. Therefore, when the above condition is true, the elements at the initial address of the current key and the root key cannot be exchanged.

At this point, a replacement cycle is completed, and the algorithm attempts to find a new original group to restart a new replacement cycle. If the search for a new original group fails, the permutation cycle ends.

When the replacement loop is restarted, the pointer to the next array element to be used is advanced group by group, not element by element. Sorting an array with N keys and K groups, ARL uses a loop of N steps to locate the next element to permute, while MSL uses a loop of K steps。 K the worst case is n and the best case is 1. The number of groups is usually much smaller than the input size, so the time saved by MSL is O (n).

*Definition and Use*

From the introduction of ARL and MSL, we can see that the new cardinality algorithm starts to study the performance optimization of various aspects, including reducing the spatial complexity, improving the adaptability of the algorithm to the cache, improving the detail efficiency of the algorithm, etc., rather than trying to reduce the time complexity of the algorithm by another level.

**REFERENCES**

A. Maus, ARL – a faster in-place, cache friendly sorting algorithm. In: N. Stol et al. (eds), Norsk Informatik Koferranse NIK’2002, Kongsberg, 26 November 2002(2002) 85–95. Available at www.nik.no/2002/ (accessed 18 July 2005).

Al-Badarneh, Amer & El-Aker, Fouad. (2004). Efficient Adaptive In‐Place Radix Sorting. Informatica, Lith. Acad. Sci.. 15. 295-302. 10.15388/Informatica.2004.061.