Link of the Codes: <https://github.com/ssimonxia/CS225H.git>

In CS225H/honors-fa23/lecture/03

I have tried different sort methods and tested them with different sample sizes and ranges to see under which conditions STL sort can be beaten by other sorting algorithms.

In the first test case, I sorted a vector with only two values in it, which can represent bits storage. Based on the time data below, when the size of the vector is less than 10, the quick sort is close to the STL sort but cannot beat it. When the size of the vector is greater than 100, the radix sort is better than the STL sort.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-2 | N (Size): 100 Range: 1-2 | N (Size): 1000 Range: 1-2 | N (Size): 10000 Range: 1-2 | N (Size): 100000  Range: 1-2 |
| Radix Sort | 3.61E-07 | 9.61E-07 | 5.82E-06 | 4.11E-05 | 5.90E-04 |
| STL Sort | 1.40E-07 | 1.00E-06 | 7.01E-06 | 8.45E-05 | 9.23E-04 |
| Bucket Sort | 8.35E-05 | 9.90E-05 | 9.35E-05 | 1.55E-04 |  |
| Heap Sort | 7.26E-05 | 7.68E-05 | 7.45E-05 | 8.90E-05 | 2.00E-03 |
| Quick Sort | 2.41E-07 | 3.98E-06 | 2.38E-04 | 2.29E-02 | 1.17E+00 |
| Merge Sort | 1.13E-06 | 5.09E-06 | 3.25E-05 | 3.24E-04 | 4.25E-03 |
| Intro Sort | 2.26E-01 | 2.24E-01 | 2.35E-01 | 3.42E-01 | 4.68E+00 |

In the second test case, I sorted a vector with ten different values. As you can see from the data below, when the size of the vector is small, the quick sort can be better than the STL sort. When the size of the vector is between 100 and 10000, the bucker sort and the radix sort are faster than the STL sort. However, the bucket sort is very unstable, and the time varies a lot. When the size of the vector is really large, the radix sort is the best.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-10 | N (Size): 100 Range: 1-10 | N (Size): 1000 Range: 1-10 | N (Size): 10000 Range: 1-10 | N (Size): 100000  Range: 1-10 |
| Radix Sort | 4.51E-07 | 1.15E-06 | 1.12E-05 | 6.99E-05 | 8.29E-04 |
| STL Sort | 2.00E-07 | 1.57E-06 | 1.75E-05 | 1.44E-04 | 1.56E-03 |
| Bucket Sort | 1.40E-06 | 1.56E-06 | 1.11E-05 | 4.92E-05 |  |
| Heap Sort | 1.34E-06 | 4.19E-06 | 5.12E-05 | 4.40E-04 | 4.55E-03 |
| Quick Sort | 1.81E-07 | 2.49E-06 | 5.38E-05 | 2.39E-03 | 2.34E-01 |
| Merge Sort | 9.42E-07 | 6.42E-06 | 6.48E-05 | 5.20E-04 | 5.41E-03 |
| Intro Sort | 4.16E-04 | 7.34E-06 | 1.68E-04 | 8.21E-03 | 7.35E-01 |

In the third test case, I sorted a vector with 16 different values, which can represent hexadecimal storage. According to the test data, the trend is similar to the last test. For small sizes of the vector, the bucker sort and radix sort are better than the STL sort. For large sizes of the vector, the radix sort is the best.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-16 | N (Size): 100 Range: 1-16 | N (Size): 1000 Range: 1-16 | N (Size): 10000 Range:1-16 | N (Size): 100000  Range: 1-16 |
| Radix Sort | 4.31E-07 | 8.52E-07 | 9.96E-06 | 9.97E-05 | 9.19E-04 |
| STL Sort | 1.41E-07 | 9.32E-07 | 1.64E-05 | 1.91E-04 | 1.63E-03 |
| Bucket Sort | 1.23E-06 | 3.45E-06 | 8.74E-06 | 6.03E-05 |  |
| Heap Sort | 8.76E-05 | 8.79E-05 | 4.19E-05 | 5.11E-04 | 5.01E-03 |
| Quick Sort | 2.31E-07 | 3.35E-06 | 3.38E-05 | 1.58E-03 | 1.47E-01 |
| Merge Sort | 9.42E-07 | 5.09E-06 | 5.54E-05 | 5.29E-04 | 5.68E-03 |
| Intro Sort | 2.26E-01 | 2.25E-01 | 9.08E-05 | 4.83E-03 | 5.47E-01 |

In the fourth case, I sorted a vector with 100 different values. Based on the test and data, when the size is very small, like 10, the quick sort can beat the STL sort. But they are very closed. When the size is from 100 to 10000, bucket sort and radix sort can beat the STL sort. Usually, bucket sort is the best, but it's unstable. Radix sort can beat STL sort all the time. As for the large size of the vector, the radix sort can beat the STL sort.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-100 | N (Size): 100 Range: 1-100 | N (Size): 1000 Range: 1-100 | N (Size): 10000 Range: 1-100 | N (Size): 100000  Range: 1-100 |
| Radix Sort | 5.51E-07 | 1.50E-06 | 1.24E-05 | 1.09E-04 | 1.23E-03 |
| STL Sort | 2.41E-07 | 2.01E-06 | 2.31E-05 | 2.19E-04 | 2.19E-03 |
| Bucket Sort | 1.68E-06 | 2.08E-05 | 3.00E-05 | 5.83E-05 |  |
| Heap Sort | 6.21E-07 | 2.01E-04 | 4.42E-05 | 5.61E-04 | 6.33E-03 |
| Quick Sort | 2.30E-07 | 2.61E-06 | 2.78E-05 | 4.54E-04 | 2.66E-02 |
| Merge Sort | 1.22E-06 | 6.72E-06 | 6.22E-05 | 6.22E-04 | 6.72E-03 |
| Intro Sort | 3.75E-04 | 3.48E-04 | 3.59E-05 | 1.22E-03 | 1.01E-01 |

In addition, I had another three tests with different distinct values in the vector, 1-1000, 1-10000, and 1-100000. In general, for the very small size of the vector, the quick sort can sometimes beat the STL sort, but not always. When the size of the vector is between 100 and 10000, and the number of distinct values in the vector is not a lot, the bucket sort is the best one. However, in general, the radix sort is MVP in the sort competition. When the size of the vector is larger than 100, regardless of the number of distinct values in the vector, the radix sort can always beat the STL sort and most other sort algorithms. Besides, heap sort is very weird. When the size of the vector is very large, it will be really slow compared to other sorting algorithms. This reason may be the cache.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-1000 | N (Size): 100 Range: 1-1000 | N (Size): 1000 Range: 1-1000 | N (Size): 10000 Range: 1-1000 | N (Size): 100000  Range: 1-1000 |
| Radix Sort | 6.31E-07 | 2.21E-06 | 1.75E-05 | 1.50E-04 | 1.65E-03 |
| STL Sort | 1.91E-07 | 2.06E-06 | 2.56E-05 | 2.98E-04 | 2.88E-03 |
| Bucket Sort | 3.15E-06 | 3.82E-06 |  |  |  |
| Heap Sort | 4.91E-07 | 3.33E-06 | 4.49E-05 | 5.83E-04 | 7.40E-03 |
| Quick Sort | 2.51E-07 | 2.63E-06 | 3.24E-05 | 3.51E-04 | 6.48E-03 |
| Merge Sort | 1.13E-06 | 6.82E-06 | 6.30E-05 | 7.10E-04 | 7.30E-03 |
| Intro Sort | 2.59E-06 | 4.73E-06 | 3.54E-05 | 4.38E-04 | 1.45E-02 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-10000 | N (Size): 100 Range: 1-10000 | N (Size): 1000 Range: 1-10000 | N (Size): 10000 Range: 1-10000 | N (Size): 100000  Range: 1-10000 |
| Radix Sort | 7.31E-07 | 1.82E-06 | 1.56E-05 | 1.92E-04 | 2.05E-03 |
| STL Sort | 2.30E-07 | 2.03E-06 | 2.74E-05 | 3.43E-03 | 3.69E-03 |
| Bucket Sort | 9.57E-06 | 1.64E-05 |  |  |  |
| Heap Sort | 3.11E-07 | 3.42E-06 | 4.51E-05 | 5.92E-04 | 7.52E-03 |
| Quick Sort | 2.00E-07 | 2.97E-06 | 3.25E-05 | 4.16E-04 | 4.43E-03 |
| Merge Sort | 9.21E-07 | 6.87E-06 | 6.30E-05 | 7.79E-04 | 8.04E-03 |
| Intro Sort | 4.20E-04 | 6.07E-06 | 3.58E-05 | 4.40E-04 | 6.02E-03 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sort Method | N (Size): 10 Range: 1-100000 | N (Size): 100 Range: 1-100000 | N (Size): 1000 Range: 1-100000 | N (Size): 10000 Range: 1-100000 | N (Size): 100000  Range: 1-100000 |
| Radix Sort | 7.32E-07 | 2.13E-06 | 2.28E-05 | 2.00E-04 | 2.37E-03 |
| STL Sort | 1.90E-07 | 2.12E-06 | 2.60E-05 | 3.28E-04 | 4.04E-03 |
| Bucket Sort | 8.20E-05 | 1.34E-04 |  |  |  |
| Heap Sort | 3.00E-07 | 3.16E-06 | 4.51E-05 | 6.01E-04 | 7.69E-03 |
| Quick Sort | 2.01E-07 | 2.52E-06 | 3.25E-05 | 4.02E-04 | 4.75E-03 |
| Merge Sort | 9.71E-07 | 6.55E-06 | 6.36E-05 | 7.04E-04 | 1.04E-02 |
| Intro Sort | 2.08E-06 | 5.03E-06 | 3.64E-05 | 4.33E-04 | 6.33E-03 |