**Sorting Algorithms Comparison**

2019-12172 이재원

April 02, 2024

# **1. Introduction**

In this report various sorting algorithms (Insertion sort, Merge sort, Quick sort (randomized pivot), Quick sort (last element pivot), Heap sort, Radix sort, Stooge sort) are compared by their time complexity in **Table 1**.

The time complexity is shown experimentally by calculating the average sorting time in **c++** code changing the seeds of randomly generated data from 0~4.

|  |  |  |
| --- | --- | --- |
| Sorting Algorithm | Average Case Complexity | Worst Case Complexity |
| Insertion Sort |  |  |
| Merge Sort |  |  |
| Quick Sort (Random Pivot) |  |  |
| Quick Sort (Last Pivot) |  |  |
| Heap Sort |  |  |
| Radix Sort |  |  |
| Stooge Sort |  |  |

Table 1. Time complexity of sorting algorithms. The ‘k’ in Radix Sort represents the number of digits in the largest number in the list.

# **2. Methods**

The sorting algorithm for the **c++** code was compile with g++ using optimization level *-Ofast*.

The **c++** code generates **json** file, and it is converted into **pandas DataFrame** in **python**. After using ***pandas.DataFrame.grouby().mean(),*** the data frame are as shown as **Figure 1**.

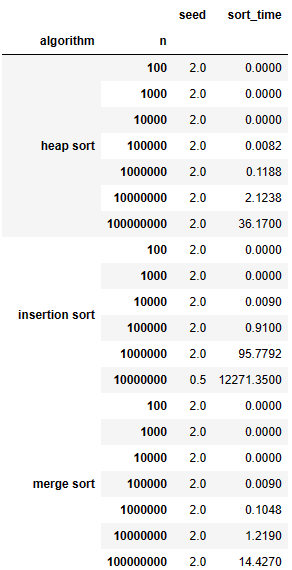


Figure 1. A part of result after using Pandas Dataframe grouby() and mean().

Then using ***pandas.DataFrame.unstack()***, the average sorting time is shown as **Table 2.**

This DataFrame is plotted by ***pandas.DataFrame.plot()*** in section **3. Results.**

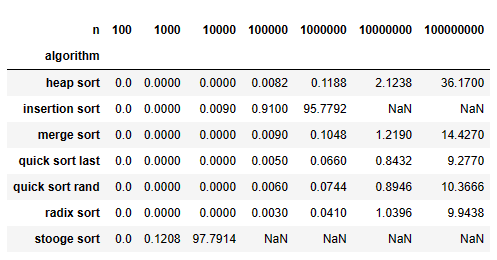


Table 2. The result of average sorting algorithms. The NaN means that the sorting time was not measured because it was over 180 seconds.

# **3. Results**

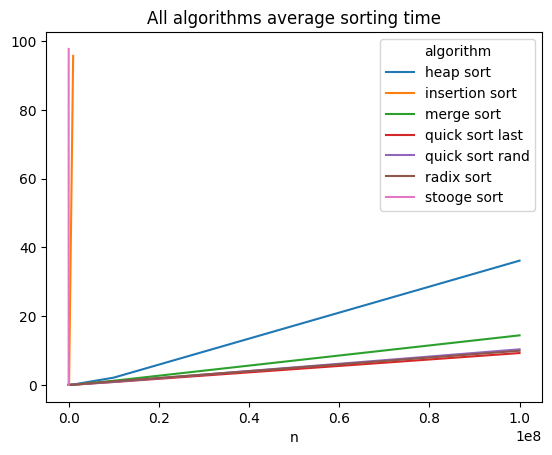


Figure 2. Average sorting time plot.

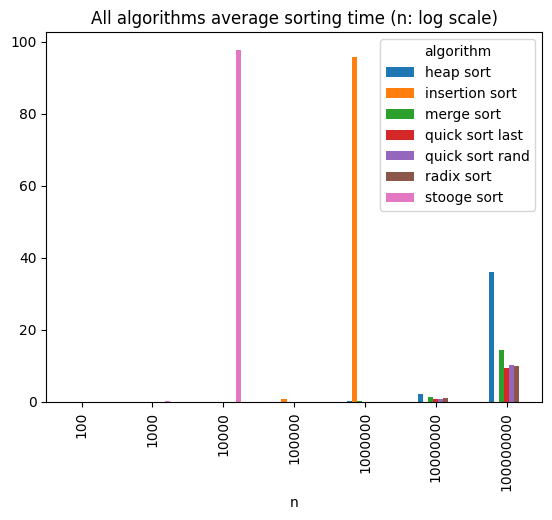


Figure 3. Average sorting time in bar plot, where axis x is log scale.

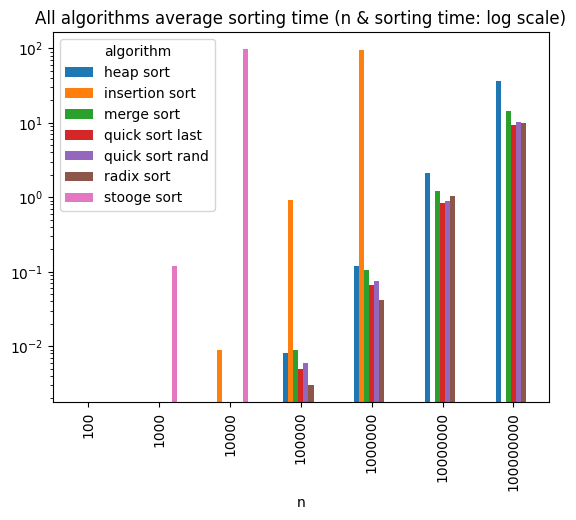


Figure 4. Average sorting time in bar plot, where axis x and y are both in log scale.

# **4. Discussion**

In Figure 2, *insertion sort* and *stooge sort* have high time complexity compared with other sorting algorithms. In Figure 4, it can be seen that *stooge sort* has higher time complexity than *insertion sort*. This makes sense because *stooge sort* has time complexity of , and *insertion sort* has time complexity of .

For other algorithms other than *insertion sort* and *stooge sort, heap sort* was the slowest sorting algorithm. However*, merge sort, quick sort, heap sort* all have same average time complexity. In Figure 2, it can be concluded that *heap sort* has a high constant factor compared to other algorithms.

*Radix sort*, which has time complexity of looks similar to algorithms in all results. This is because the ‘k’ in *Radix Sort* represents the number of digits in the largest number in the list. The data used in this experiment has range from 0 to *MAXINT\_32*, which is 2147483647. Therefore, k is 10, and at is not high enough compared to k = 10. And this makes it look the time complexity of and similar for .

# **5. Conclusion**

In this report, various sorting algorithm times were measured and compared the average time. The results shows the time complexity of each algorithms.