Comparison of Sorting Algorithms

Research Report

I. Introduction

There are two main sorting method types we have used, Iterative Sort and Recursive Sort. Iterative sort uses loops in the algorithm to sort the data set. Iterative sort has three different sorting methods: Bubble sort, Selection sort and Insertion sort. Bubble sort is the simplest sorting algorithm. This works by taking two adjacent values, comparing them, and sorting them accordingly. This is done through the whole list repeatedly until the whole list is sorted. The Selection sort method takes the lowest value item in the list and swaps it with the first object in the list. Then it moves to the second object on the list and finds the next lowest value in the list and swaps it. It goes down the list until its completed sorted. The third Iterative sort, Insertion sort, works in a similar matter. Here it takes the next object and then compares it to the first object. Then it takes the next object in the list and compares it to the previously sorted objects. This continues until it reaches the end of the list.

Recursive sort relies on the divide and conquer method to sort data sets. It breaks down large data sets into smaller and smaller easier parts and then sorts the small parts repeatedly until the data set is sorted. The two recursive methods used in this research are the Quick sort and Merge sort. Quick sort works by selecting a pivot point and the smaller elements are put before the pivot and the larger elements after it. As the data set is broken down, this methodology of partitioning is applied to each portion of the data set until completely sorted. Merge sort is first broken down to single set elements recursively, and then rebuilt by comparing two sets at a time. It takes two single sets, compares and joins them. Then it takes two two-element sets and compares them. This is done until the data set is merged and sorted.

Recursive sort methods are complex and are harder to implement compared to iterative sorts. Recursive methods also have the potential to cause recursive overhead and stack overflow, cause errors and program crashes. However, Iterative sort methods are much slower, especially with larger data sets. The research tests for each method, will show and compare the

average values, best and worst values for various data sets. This will show how recursive sorts outperforms iterative sorts, for all data set sizes.

II. Argument

The main reasoning for the hypothesis is that iterative and recursive sorts have different methodologies to sorting. Since iterative methods use nested loops in the algorithm, it has a higher temporal complexity. In fact, the average performance for Iterative sort is O(N2). With this relationship, larger datasets result in longer processing time, since it is quadratic. Even with a best-case scenario, where the data set is nearly sorted, iterative sort has an O(N) temporal complexity. The linear relationship therefore increasing process time with the same ratio. The recursive method of divide and conquer, speeds up the processing time. The average performance for the two recursive sort is 0(N log N). With this logarithmic relationship, an increase in the data set will not have a large impact on processing speed. Therefore, as the data set gets larger, recursive sort will outperform due to the temporal complexity.

Recursive methods do have recursion overhead and a potential to crash, but this isn't necessarily a huge disadvantage. It's only a concern when there is linear recursion, which keeps the recursion continuous. The divide and conquer method, breaks the data set by half each time as it goes down, creating a new stack frame. However, it uses a depth-first method, which is to go down one branch of data and going back up. Once it reaches the bottom of a set, usually a one set element, it comes up the recursion method, deleting the previously created stack. This method creates only a small amount of recursion overhead since the spatial complexity is a logarithmic recursion or O(N log N). Comparing this to iterative sort of O(N), recursive methods have a better spatial complexity, therefore neglecting recursive overhead.

Therefore, using recursive sorts is a better choice when dealing with data sets of any size. Even though these methods are more complex than iterative sorting methods, recursive methods have better spatial and temporal complexity, making them process data sets faster.

III. Test Process

A test program was created to help with testing this hypothesis. The program asks the user to choose from a list of the five different sorting methods to be tested. It then takes the following testing parameters from the user, data set size and how many iterations of the test. Whichever method is chosen, the program will create an array of randomized 6-digit numbers with the set size given by the user. The sorting algorithm will sort the data in ascending matter, and the time taken to process this is outputted on the console. This test is repeated, rebuilding a new array and sorting it, as many times as the user requested. The program will also produce a CSV file for processing the information gathered.

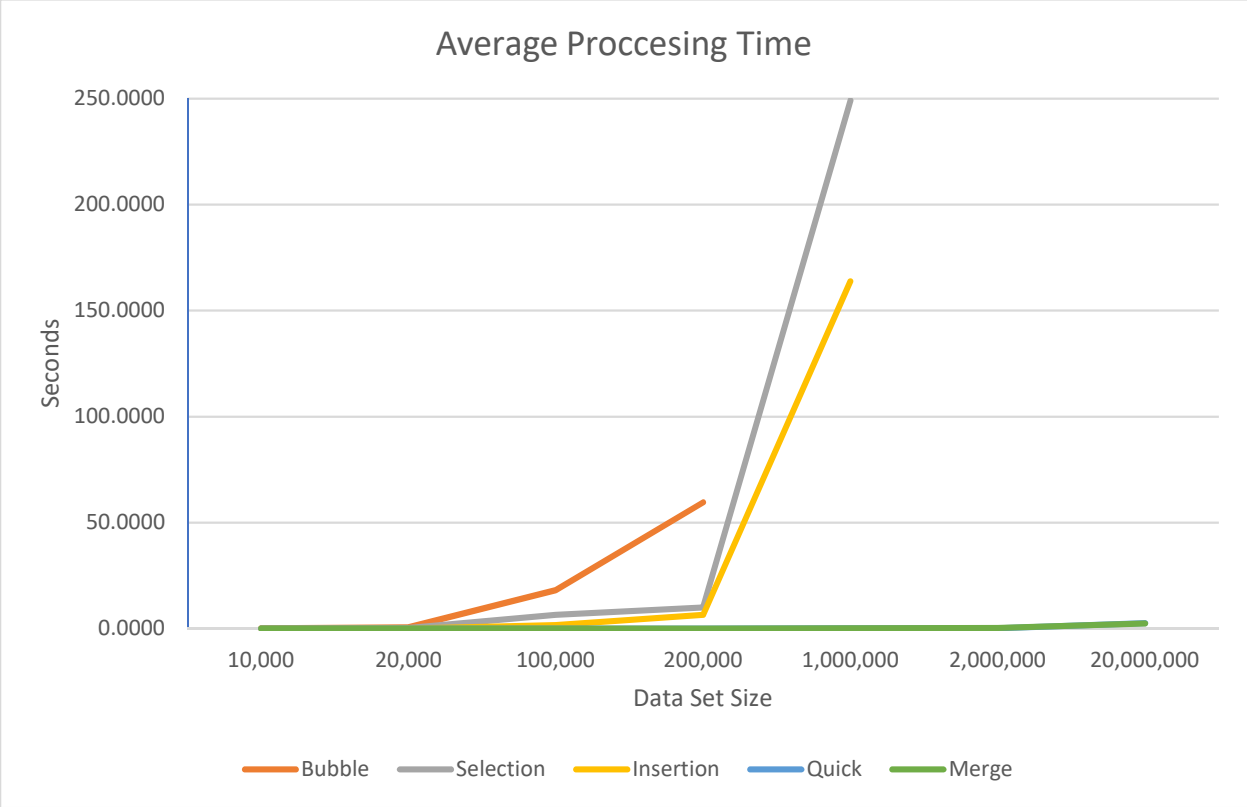
For our test, each method will run 7 data sets, ranging from 10,000 to 20,000,000 elements. Each method will test the sort time for each data set 5 times. The processing time is then averaged and compared between methods and data size. For the purpose of this research, the tests were conducted in the same environment and system settings, running the CPU at minimum of 10%. This was to rule out any other variables that may cause delay in processing time. In addition, any sorting method that took more than 5 minutes, the program was terminated manually, as this was considered too long and unnecessary. If a sorting method took more than 5 minutes to process a data set, any data sets larger than this was neglected as its given that it will also take longer than 5 minutes. The system configuration for the testing environment is below.

System Configuration		
Make:		**Lenovo**
Model:		**Provemce-5R1**
Processor:		**Intel Core i5 7300HQ**
RAM:		**DDR4 16GB**
Op. Sys.:		**Windows 10, 64 bit**
IDE:		**IntellJ IDEA 2019.1.13**
Language:		**Java 11**

IV. Results

	Averages (Seconds)						
	Data Set Size						
Sort Method	**10,000**	**20,000**	**100,000**	**200,000**	**1,000,000**	**2,000,000**	**20,000,000**
Bubble	0.1627	0.6281	18.1173	59.5893	-	-	-
Selection	0.0659	0.2339	6.4499	10.0217	249.1945	-	-
Insertion	0.0241	0.0706	1.6033	6.4711	163.8395	-	-
Quick	0.0010	0.0015	0.0085	0.0168	0.0873	0.1740	2.5515
Merge	0.0017	0.0018	0.0104	0.0207	0.1133	0.2298	2.4031

The table above shows the average processing time for each method and corresponding data size. This table shows how the two recursive methods outperformed the iterative methods even at lower data set sizes. As the data set grew, the processing time for iterative sorts increase exponentially. Quick sort and Merge sort in the other hand only increased slightly. The Bubble sort didn't even pass the 1 million data set time. Selection and Insertion performed better than the Bubble sort but also failed after the 1 million data set.



The graph above summarizes the average time for the sorting methods. As seen from the table and graph, Bubble sort was the slowest and took longer than 5 minutes on only a data set of 200,000 elements. Insertion and Selection sort performed better, but you can see the giant leap it took at 1 million elements, taking about 163 seconds and 250 seconds. It took more than 5 minutes for the 2 million data set. Merge and Quick sort, as mentioned, only had a slight increase, taking only about 2.5 seconds to sort a 20 million data set.

V. Conclusion

As seen from the results, on average the recursive method's processing time was faster, and only increased slightly with increase of data sets. As expected, the processing time for the iterative sort methods were a quadratic relationship, have a temporal complexity of O(N2). This can be seen from data results, especially around the 200,000-length data set, where iterative sort jumps up. The recursion sorts keep a steady and logarithmic growth, having a slight bump in processing time when tested with 20 million elements. The tests also did not produce any error or crashes during the recursive method testing, since as predicted, the recursive overhead is neglectable, even at huge set of 20 million.

Therefore, the recursive methods Quick sort and Merge sort, are far superior than the iterative sort. The complexity of the algorithm of the recursive method is also neglectable, as once a person gets the understanding of how the method works it is easier to implement. Iterative sorts are better for learning purposes and to introduce sort methods, but in real world applications it is not recommended. For future tests, it will be interesting to test the methods in different environments, and data types, such as strings. However, in theory and practice it will show that recursive sort outperforms iterative sorting methods.