**[Algorithm Analysis](https://classroom.google.com/u/1/c/NjE3MDMwNDQzNTRa" \t "_blank)**

**IZMIR**

**19.04.2020**

**1 – Overview**

Observing and comparing the theoretical algorithm complexity and real system results of 3 different sorting algorithms, and examining the asymptotic increases of these 3 algorithms in different data sizes

**2- Algorithms and Complexity**

**2.1 Heap Sort:** Heap sort is a comparison based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the maximum element and place the maximum element at the end. We repeat the same process for remaining element. (Unknown, Geeksforgeeks, 2016)

|  |
| --- |
| **Compexity** |
| **Best Case** | O(n lgn) |
| **Avarage Case** | O(n lgn) |
| **Worst Case** | O(n lgn) |

**2.2 DualPivotQuick Sort:** The idea of dual pivot quick sort is to take two pivots, one in the left end of the array and the second, in the right end of the array. The left pivot must be less than or equal to the right pivot, so we swap them if necessary. (Unknown, Geegforgeeks, 2015)

|  |
| --- |
| **Compexity** |
| **Best Case** | O(n lgn) |
| **Avarage Case** | O(n lgn) |
| **Worst Case** | O(n2) |

**2.3 Shell Sort:** Shell sort is an algorithm that first sorts the elements far apart from each other and successively reduces the interval between the elements to be sorted. It is a generalized version of insertion sort.(Unknown, Programiz, 2017)

|  |
| --- |
| **Compexity** |
| **Best Case** | O(n lgn) |
| **Avarage Case** | O(n lgn) |
| **Worst Case** | O(n2) |

**3- Hypothesis**

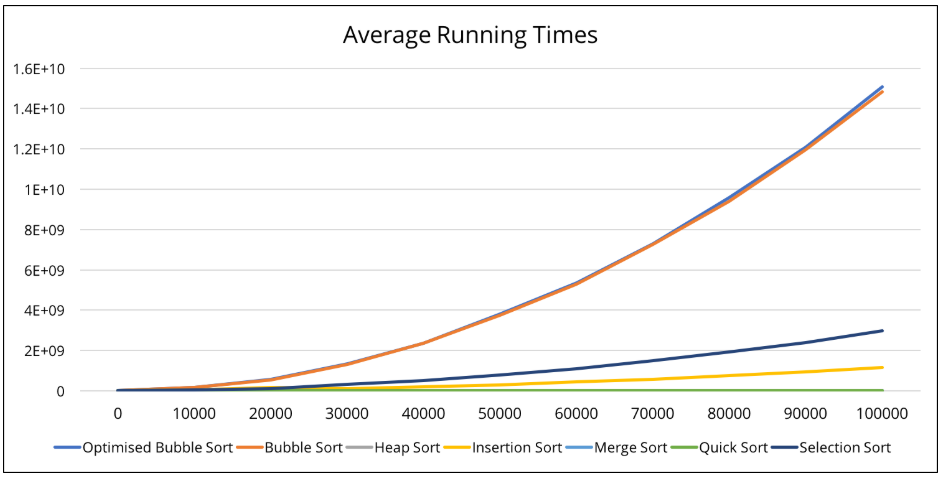
Heap Sort is expected to give better results in big data because The Heap Sort algorithm is expected to be n.lg(n) for all 3 complexity states and also to be more stable for three cases, according to the theoretical calculation of the algorithm complexity.

**4-Experiment and Inference**

As a first impression, if we just look at the time tables, we can think that heap sort is better. It looks like heapsort, which will give much better results in theory. In addition, it can be perceived that there is no difference between dualPivotSort and ShellSort. But according to the test result made in real time system:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **EQUAL**  **INTEGERS** | | | **RANDOM INTEGERS** | | | **INCREASING INTEGERS** | | | **DECREASING**  **INTEGERS** | | |
|  | 1,000 | 10,000 | 100,000 | 1,000 | 10,000 | 100,000 | 1,000 | 10,000 | 100,000 | 1,000 | 10,000 | 100,000 |
| heapSort | 0ms | 1ms | 4ms | 0ms | 2ms | 14 ms | 0ms | 2ms | 6ms | 0ms | 0ms | 7ms |
| dualPivot  QuickSort | 1ms | 2ms | 8ms | 1ms | 1ms | 9ms | 1ms | 65ms | 1606ms | 1ms | 20ms | 1544ms |
| shellSort | 1ms | 2 ms | 6ms | 1ms | 2 ms | 15 ms | 0ms | 1 or 0  ms | 1 or 2 ms | 0ms | 1 or 0 ms | 2 or 3 ms |

It is seen in the table that there is no serious difference between heapSort and shellSort, and in some cases shellSort performs better.  While the asymptotic complexity of heap sort makes it look faster than quicksort, in real systems heap sort is often slower. (Unknown, Interview Cake, 2020) When we analyze it in detail, the reason for this is that we throw constant numbers while analyzing. Although theoretically these fixed numbers do not matter, if this constant coefficient value is too large, it is observed that the algorithm affects the run time. When looking at the table, the performance difference between shell sort and heap sort is for this reason. An example of the incompatibility of theoretical algorithm complexity with real-time system operation is the rubik cube solving algorithm. In Rubik's cube algorithm, it can be observed that this situation shows its validity. Although the algorithm complexity is n2/ (lg n), the constant coefficient at the beginning is at the level of millions, so it can take years for the computer to analyze it. (Demaine, Demaine, Eisenstat, Lubiw, & Winslow, 2011) Although they have the same algorithm complexity as the data sizes increase between Shell Sort and Dual Pivot Sort, the reason they give different results is that the pivot element is not kept at an optimum level. Also, the effect of recursive, which is another problem in sort algorithms, affects this.



(Punchihewa, 2017)

Considering the average working time in the table above, considering that Shell Sort is an insert sort derivative and DP Sort is a quick sort, it is expected that the DP algorithm will give better results. However, the effects of constants, which seemed to have no effect on pivot selection and algorithm analysis, on the actual system have shown the exact opposite of this table.

**4.1 Problems :**

The error I encountered was StackOverFlow error in DualPivotQuick Sort. The reason for this error is that the recursive function is added to the stack structure in the background of Java each time we call the function. I found 2 solutions, the first one was to change from the original source of the java from the cmd terminal with

-XX commands at <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/java.html> site (it didn't work on me). The second solution is to manually set it with Eclipse Run Configuration setting. Setting Xss to 4mb was enough for the test. What I observed after this adjustment was a decrease in performance when I increased the stack capacity. But after making these changes, I chose to write the previous results, not the current results . Results marked in red are values after the Xss change.

**5- Scenario**

We aim to place students at universities according to their central exam grades and

preferences. If there are millions of students in the exam, which sorting algorithm would you

use to do this placement task faster?

**5.1 Solution Strategy :**

Theoretically, although the selection of the heap sort algorithm seems correct, it has been a more stable was shellSort as seen in real system tests. Considering that there are millions of data, although ShellSort's worst case is O (n ^ 2), I would choose it based on its performance in the real system.

**6 – Conclusion**

In sort algorithms, the algorithms whose theoretical complexity appears the same or different can show different results in real system tests or real situations. The algorithm to be used should be selected based on data size and hardware requirements and actual system test results. It is also possible to obtain different results for different variations of these algorithms. According to our first expectation, we can also say that the lower results of the Dual Pivot sorting algorithm are due to the wrong pivot selection.(Yaroslavskiy, 2009)

# References

Demaine, E. D., Demaine, M. L., Eisenstat, S. C., Lubiw, A., & Winslow, A. (2011, 8). *MIT*. Retrieved from MIT Open Courses: https://dspace.mit.edu/handle/1721.1/73771

Punchihewa, H. (2017). *Analysis of Sorting Algorithms.* London: Punchihewa, Hashan.

Unknown. (2015). *Geegforgeeks*. Retrieved from Geegforgeeks: https://www.geeksforgeeks.org/dual-pivot-quicksort/

Unknown. (2016). *Geeksforgeeks*. Retrieved from Geeksforgeeks: https://www.geeksforgeeks.org/heap-sort/

Unknown. (2017). *Programiz*. Retrieved from Programiz: https://www.programiz.com/dsa/shell-sort

Unknown. (2020). *Interview Cake*. Retrieved from Interview Cake: https://www.interviewcake.com/concept/java/heapsort

Yaroslavskiy, V. (2009, 12 11). *Dual-Pivot Quicksort algorithm.* Retrieved from Dual-Pivot Quicksort algorithm: https://codeblab.com/wp-content/uploads/2009/09/DualPivotQuicksort.pdf