Jose Sandoval

CS331 – Design and Analysis of Algorithms

February 19, 2018

**Project 01 – Sorting and Matrix Multiplication Algorithms**

**Time Analysis**

The purpose of this project is to analyze the time it takes for different sorting and matrix multiplication algorithms to accomplish the same task in order to evaluate their efficiency with different sized data sets.

*Sorting Algorithms*

For the sorting algorithms three different algorithms were used: The Exchange sort, the Merge sort, and the Quicksort. The first algorithm is a polynomial algorithm with a time complexity of O(n2) that takes each element and compares it with the next elements after it looking for a lower or higher number to be swapped with. The next two algorithms use the divide and conquer technique to perform the sorting faster making them have a O(nlogn) and O(n2) respectively. Even though the worst case for the Quicksort is O(n2), on average it is O(nlogn). All three sorting algorithms were tested with randomized arrays of different sizes and its execution times (in seconds) are presented in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Sorting Algorithms | | | |
| n | **Exchange Sort** | **Merge Sort** | **Quicksort** |
| 10,000 | 0.759 | 0.003 | 0.009 |
| 20,000 | 2.146 | 0.005 | 0.018 |
| 50,000 | 5.468 | 0.015 | 0.056 |
| 100,000 | 29.275 | 0.019 | 0.081 |
| 200,000 | 89.238 | 0.058 | 0.039 |
| 500,000 | 711.927 | 0.279 | 0.151 |
| 1,000,000 | Timed out! | 0.545 | 0.526 |
| 2,000,000 | Timed out! | 0.846 | 0.809 |
| 10,000,000 | Timed out! | 2.139 | 1.925 |
| 20,000,000 | Timed out! | 4.088 | 3.351 |
| 50,000,000 | Timed out! | 8.491 | 6.747 |
| 100,000,000 | Timed out! | 25.524 | 21.884 |

As it can be seen, the Exchange sort even in its average case executes in exponential time as the size of the array to sort increases. From the table it can also be appreciated how the Merge sort is faster at lower array sizes but quickly falls behind the Quicksort at the size of the array increases, though the difference is minimal at the teste sizes.

*Matrix Multiplication Algorithms*

For the matrix multiplication algorithms, the classical method and the Strassen method were used for analysis. The algorithms were tested using a square matrix that doubled in size. As it can be appreciated from the following table and chart, the Strassen method is much faster than the classical method due to the fact that the Strassen method divides and conquers the Matrix to find the result matrix.

|  |  |  |
| --- | --- | --- |
| **Matrix Multiplication Algorithms** | | |
| **n** | **Classical** | **Strassen** |
| 2 | 0 | 0 |
| 4 | 0 | 0 |
| 8 | 0 | 0 |
| 16 | 0 | 0 |
| 32 | 0.001 | 0 |
| 64 | 0.002 | 0 |
| 128 | 0.006 | 0 |
| 256 | 0.0334 | 0.002 |
| 512 | 0.37 | 0.006 |
| 1024 | 4.407 | 1.206 |
| 2048 | 172.979 | 18.392 |

In conclusion, the algorithms that used the Divide and Conquer technique are much faster than their counterparts that work on an exponential time.