Strassen's Matrix Multiplication in Java

Analysis versus Naive Algorithm

TCSS 343 - Extra Credit Assignment

February 26, 2016

Brandon Chambers

INSTRUCTIONS

1.) **Run program from the file MainSM.java**
2.) **You will be prompted in your ide console to enter the order of matrix (for a 64x64, type in '64' and hit enter).**
3.) **All data will be displayed in console.**
4.) **All data will also be output to 'matrixMult.txt' in the project root folder (more readable for large matrices).**

OVERVIEW

My program is split into three classes: "MainSM.java", "Naive.java", and "Strassen.java". The main driver class is MainSM.java, run the code from there. Within main(), I have provided the convenience of user input at the console to input the desired matrix order; so if you want a 64x64 matrix, simply run, type in '64' at console, and press enter.

At that point, my program will output both randomized matrices, A and B, of the specified order to the console, neatly formatted and labeled.

Directly under A and B matrices will be two separate product matrices, 'C'. First one will be the result with the Naive algorithm followed by various data points directly underneath it. Then at the end we have the resulting matrix from Strassen's algorithm, again followed by various data points, however there are unique data points here that show the difference between the data of the Naive method versus Strassen's method.

For both I have provided the most accurate running time possible with the java nanoTime() method and presented in milliseconds. Also, I have included counter variables in both algorithms for the addition/subtraction and multiplication operations.

My program also outputs all of this data into a text file entitled "matrixMult.txt".

ANALYSIS

The Naive algorithm is slightly faster for matrices of size 2 and 4 in my implementation. However, starting at matrix order 8 and up, Strassen's algorithm for matrix multiplication increasingly gets faster, and vastly so. For instance, on matrix order 128, on my machine, Strassen's is 7,458,370 milliseconds faster than the Naive algorithm (strassen's: 0.08269 ms, naive: 7.54106 ms).

As far as the number of arithmetic operations go, Strassen's greatly increases the number of addition/subtraction operations. Using the same run example of order 128: strassen sub/add=4,842,954 and naive add=2,097,152. That's more than double the amount for Strassen's. However, as we know, multiplications are instead reduced for Strassen's, which is why we get better running times at order > 4: strassen's mult=960,799 and naive mult=2,097,152. Multiplication operations are significantly more expensive, and with a reduction that large, we see great gains. The Naive algorithm has multiplications exactly equal to n^3 where n = matrix order. In this example: 128^3. That is why it is a huge deal that Strassen's algo is able to reduce the multiplications per iteration at the cost of many more add/subtracts (constant time for those).

Naive Algorithm: $\in O(n^3)$

Strassen's Algorithm: $\in$

$$O([7+o(1)]^n) = O(N^{\log_2 7+o(1)}) \approx O(N^{2.8074})$$

We have 7 multiplications and some constant number of add/sub operations per iteration, all raised to n=order of matrix.