605.620: Algorithms for Bioinformatics

Thomas Muamar

Project 1: Lab Strassen

Due Date: September 28, 2021

Dated Turned In: September 27, 2021

Project 1:

Justification for data structures/reasons for choices:

This program is made in three different classes. The Main class is called Read File which reads the input file by requiring the user to input to argument parameters. The second class contains the actual code for the Strassen multiplication and the last class uses the ordinary multiplication of matrices to produce the product. In main class the output method will print out the two arrays that are being multiplied plus the number of multiplications it takes to get to the product and the time in nanoseconds. The reason I choose to put these in the main method is to show the difference between the two algorithm and what happens to each of the algorithms as the order increases. So, the main method starts by reading the input file and parses out the order and matrix pairs to produce the A and B matrixes into arrays. The next part will write these matrixes in the output and also write the product of the matrix multiplication for both the Strassen and ordinary multiplication. In the end the reason I separated these three into different classes is because it made it a lot easier to debug the different classes by adding a test method without reading it from the text file.

The Strassen multiplication algorithm was produced using are book introduction to algorithms 3rd edition. Following the four steps into producing the algorithms it helped provide an easier way into producing the algorithm. The start of this Strassen algorithm begins by creating 8 new s/2 x s/2 sub matrices for both the A and B values. This step takes 8*O(n^2) and in order to do this I need to create the extra method of splitting the matrix into these 8 new sub matrices. Then there is the next step of creating the 10 matrices by doing the subtractions and additions. These S matrices are produced in 10*O(n^2) time. Then there are the P matrices which is 7 recursive calls of multiplying the values in step 1 with the 10 matrices in step 2. This process takes 7*T(n/2). Then to create the C sub-matrices I end up taking 8O(n^2) because it is another additional 8 subtractions and additions. Then I combine the results of these C sub-matrices to produce the product which is 4 of these sub matrices that get combined. The combination is done through the combine method.

So, the total time for this Strassen algorithm would take:

T(n) = 8*O(n^2) +10*O(n^2)+7*T(n/2)+ 8* O(n^2) +4*O(n^2)

T(n) = 7*T(n/2)+30*(n^2)

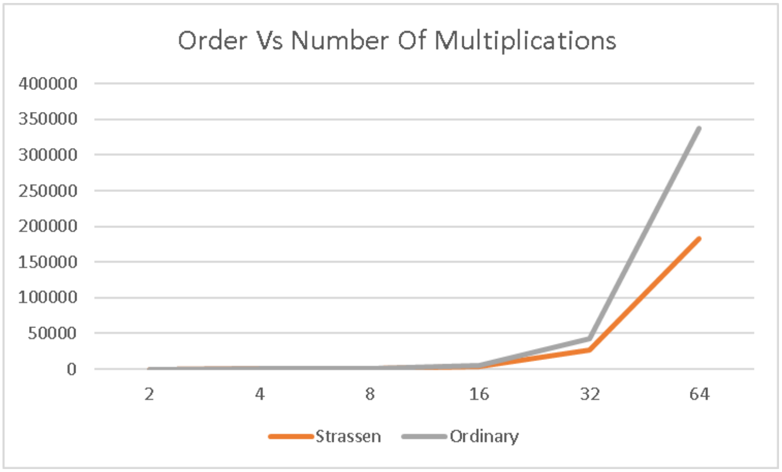
Using the master's theorem here we end up getting the time complexity of O(N^log7) which is about o(n^2.81). This ends up being a very close comparison to the ordinary multiplication algorithm.

The ordinary multiplication goes through 3 for loops which is from lines 9 to 14. These triply nested for loops each run in n iterations and then the last line 16 takes constant time so the rune time that this algorithm would take would be about Ɵ(n^3) time. The space complexity of both the ordinary and Strassen multiplication would be O(n^2).

Ordinary Method Vs. Strassen Method:

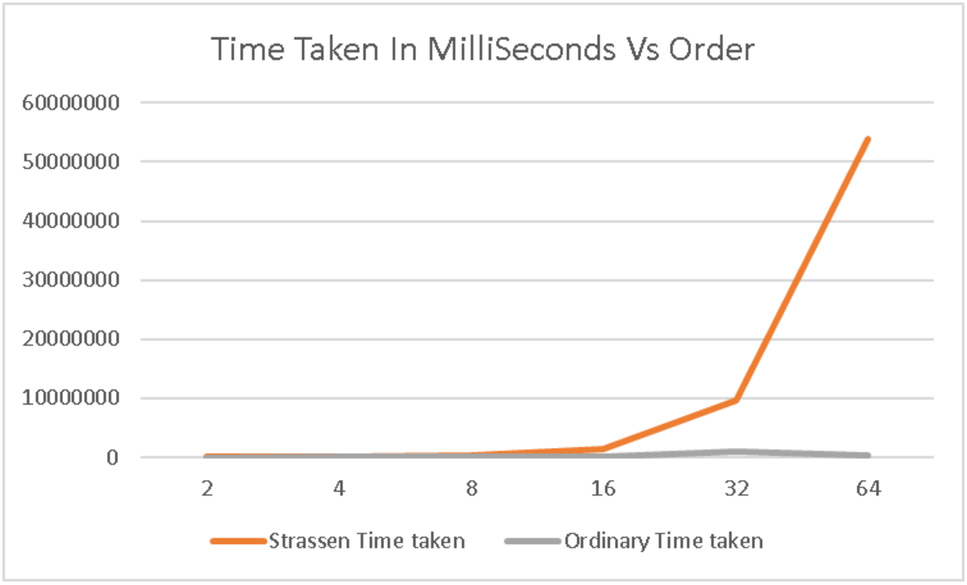
In comparing the time complexity between the two it seems that Strassen multiplication would be just a little bet better than ordinary multiplication. One thing that makes Strassen multiplication better is when there is a larger amount of data. An order of greater than 1000 seems to be where it starts being better than ordinary multiplication. One of the inputs I attempted that froze my eclipse was trying to do larger orders and ordinary method just took way longer than the usually amounts.

Order	Strassen	Ordinary
2	7	8
4	70	80
8	525	656
16	3724	5264
32	26131	42128
64	182994	337040



This chart shows that ordinary multiplication ends up becoming worse in the number of multipications that it takes when the order of the matrixies become larger. When looking at Strassen at the different values of order it seems to have a less number of multiplications but just out performs the ordinary as it ends up getting to that bigger order of values.

Order	Strassen Time taken	Ordinary Time taken
2	11000	2200
4	94300	3500
8	269500	21100
16	1398600	126100
32	9664300	998200
64	53806600	342900



When taking the time taken in milliseconds of these algorithms from start to end it seems like Strassen algorithm takes a lot longer than the ordinary algorithm even though the total number of multiplications in the Strassen algorithm is much less. The reason I think Strassen seems to take a longer time could be due to the methods that I created to split and combine the matrixes. With ordinary multiplication it did not require a huge number of methods it just required nested for loops to produce the product.

<u>What I Learned:</u>

I learned how the Strassen algorithm works all together, and from my understanding is the time efficiency of the Strassen ends up becoming better if it is a large order. Overall, got a better understanding on how to code Strassen algorithm and ordinary multiplication algorithm of matrices.

<u>What I might do differently next time/if:</u>

Something I might do differently next time is finding a way to have the program calculate the average time for a huge amount of runs instead of doing it myself. Another thing that I would do if I had more time is to try and figure out a way to produce a more efficient Strassen algorithm. The parts that I would need to make efficient would probably finding a way to splitting the matrices in a better way.

Maybe something that would make a better comparison between the two algorithms would be making a recursive ordinary multiplication that ended up splitting matrices like the Strassen algorithm.

<u>Usefulness In Bioinformatics:</u>

One of the usefulness of matrices that come to mind is PAM matrices. PAM stands for the percent accepted mutations and the reason for the use of matrices are useful here is because we can get the relative frequency of each type of mutation. One example that I found was to “multiply PAM1 by itself many times to arrive at some particular evolutionary distance measure in PAMs.” (Martti). The algorithm that I would most likely choose would depend on how big the data would end up becoming.

So if matrices were order is 1000+ I would most likely go with strassen's algorithm but ordinary multiplication seems to work perfectly fine when it comes to lower order input.

References:

1. https://bioinformaticshome.com/bioinformatics_tutorials/sequence_alignment/substitution_matrices_page2.html