CS 3310 Design and Analysis of Algorithms
Project #1

Abstract

This project asks to implement, test, and analyze the three different matrix multiplication algorithms discussed in class. The matrix size starts 2 by 2 until the maximum size my device can handle due to running out of memory. The input size will be limited in the power of 2 as 2, 4, 8, 16, 32 up to 2^k where k is the largest size of the Integer. The reason for the power of 2 is to ease the programming component of the assignment.

Part 1: Design & Theoretical Analysis

a. Complete the following table for theoretical worst-case complexity of each algorithm. Also need to describe how the worst-case input of each algorithm should be.

Algorithm	theoretical worst-case complexity	describe the worst-case input
Classical matrix multiplication	O(N^3)	O(N^3)
Divide-and-conquer matrix multiplication	O(N^3)	O(N^3)
Strassen's matrix multiplication	O(N^2.8)	O(N^2.8)

b. Design the program by providing pseudocode or flowchart for each sorting algorithm.

Algorithm> Project1 > Documentation > DesignDocs

c. Design the program correctness testing cases. Design at least 10 testing cases to test your program, and give the expected output of the program for each case. We prepare for correctness testing of each of the three programs later generated in Part 2.

The test data execution sample run in main

Part 2: Implementation

a. Code each program based on the design (pseudocode or flow chart) given in Part 1(b).

b. Test you program using the designed testing input data given in the table in Part 1(c), Make sure each program generates the correct answer by marking a "√" if it is correct for each testing case for each program column in the table. Repeat the process of debugging if necessary.

c. For each program, capture a screen shot of the execution (Compile&Run) using one testing case to show how this program works properly

The sample run of the test methods' results are available at the bottom of the main file.

Part 3: Comparative Analysis

a. Run each program with the designed randomly generated input data given in Part 1(d). Generate a table for all the experimental results as follows.

Unit is millisecond, second = 1000ms, minutes = 60000ms

Input Size n	Average time (Classical matrix multiplication)	Average Time (Divide-and-conquer matrix multiplication)	Average Time (Strassen's matrix multiplication)
2	Less than 0 millisecond	Less than 0 millisecond	Less than 0 millisecond
4	Less than 0 millisecond	0.0204	Less than 0 millisecond
8	Less than 0 millisecond	0.0408	0.0612
16	Less than 0 millisecond	0.4081	0.4694
32	0.0204	2.1633	2.3878
64	0.2041	14.5918	13.7551
128	1.8367	117.8367	97.1633
256	15.0204	943.5714	687.102
512	117.5918	7458.5714	4712.6735
1024	2134.4286	59749.7959	33125.102
2048	15951.1429	477756.184	232192.898

b. Plot a graph of each algorithm and summarize the performance of each algorithm based on its own graph.

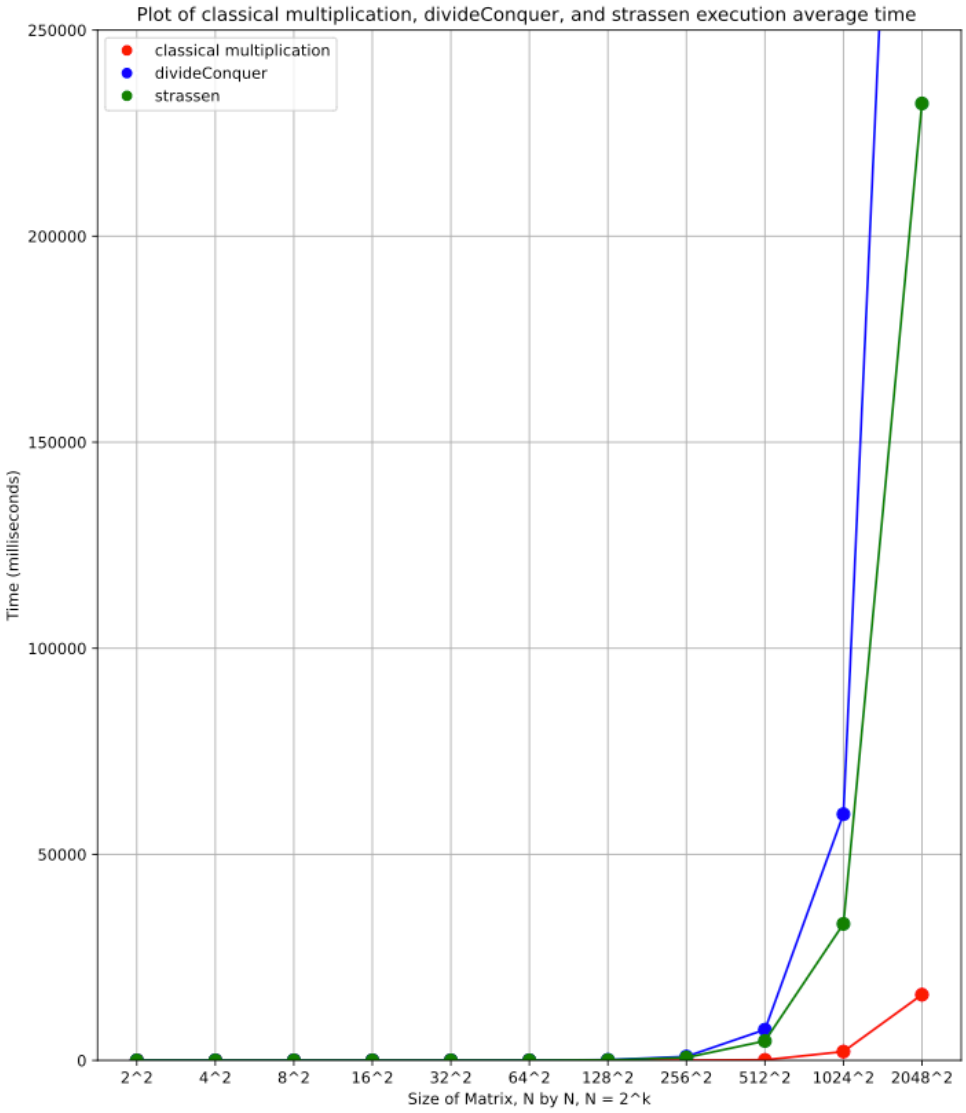
The Classical matrix multiplication is the least run time, compared to the other 2 algorithms. The Divide-and-conquer matrix multiplication and the Strassen's matrix is recursive call and the Strassen's is slightly faster result because the Strassen's has O(N^2.8) and the Divide-and-Conquer is O(N^3). It makes sense, but my question is why the classical matrix multiplication is the least run time among the three algorithms.

Plot all three graphs on the same graph and compare the performance of all three algorithms. Explain the reasons for which algorithm is faster than others.

<Insert - three-graphs-in-one graph here>

c. Please see the attached PDF pages.
   1. Classical matrix multiplication Red

2. Divide-and-conquer matrix multiplication Blue
3. Strassen's matrix multiplication Green
4. All the three plot together.



d. Compare the theoretical results in Part 1(a) and empirical results here. Explain the possible factors that cause the difference.

<Insert - report the findings and explain>

My assumption for the reason of the result is the cache locality. The classical algorithm is three nested loops, and it exhibits good spatial locality. It accesses memory locations in an aligned way with how data is organized in the memory hierarchy. However, the Divide-and-Conquer method and Strassen's method call itself recursively, which causes the complex memory access pattern. As a result, it leads to cache misses.

Another thought is the size of the matrix. Since the experiment is performed on my personal device, it can handle a certain matrix size. My case is 2048 by 2048 was the maximum size. I cannot keep running the program after this size because it takes more than 8 hours, and I need to use my device for my remote work. If I could use a more powerful computer and try a more significant size matrix, it might be a different result.

e. Give a spec of your computing environment, e.g. computer model, OS, hardware/software info, processor model and speed, memory size, ...

- Model Name: Mac mini
- Mac OS Sonoma 14.0 (23A344)
- Chip: Apple M1
- Total Number of Cores: 8 (4 performance and 4 efficiency)
- Memory: 16 GB

f. Conclude your report with the strength and constraints of your work. At least 200 words. Note: It is reflection of this project. Ask yourself if you have a chance to re-do this project again, what you will do differently (e.g. your computing environment, programing language, data structure, data set generation, …) in order to design a better performance evaluation experiment.

<Insert - write a conclusion about strength and constraints of your work here.>

This project is such a fun project, and I experienced and discovered many interesting results. The one thing I can improve is file structure. I struggled with how to separate the main three different algorithms, my test cases, data set, and plots. Since this project requires three different files for each algorithm, I tried to make methods-only files in Java. However, Java requires making an Object even though it only has methods, and I was not familiar with how to set up only methods in Java. It took time to figure out how to construct a utility class.

Also, I did not know how to show my test cases as a separate class. Usually, I use a test unit in Java, and it is very convenient for me to test out my method before I start using it. But the test cases can execute only my environment. I had to figure out how to set up my test cases to execute in any environment to show my functions work properly. I decided to put my test cases in the main as a static method, which I did not prefer to do. However, I followed this way because the instructions showed I needed to put a main and three files for algorithms. For the next project, what I can improve is to think about file structure based on the first project's experience.