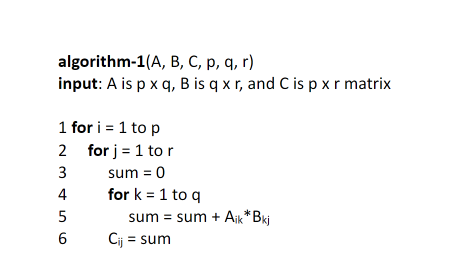
**Algorithm-1**



Nested Loops:

The outer loop runs p times (line 1)

The middle loop runs r times (line 2)

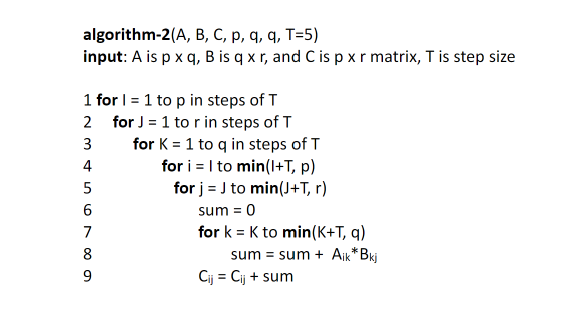
The inner loop runs q times (line 4)

The total number of iterations is p x r x q

The time complexity of the algorithm is O(p x q x r)

O(n3)

**Algorithm-2**



Outer Loops:

Outermost loop runs from 0 to p with a step of T, p/T iterations

Seconds outer loop runs from 0 to r with a step of T, r/T iterations

Third outer loop runs from 0 to q with a step of T, q/T iterations

Inner Loops:

Fourth loop runs from I to min(I+T, r), T iterations

Fifth loop runs from J to min(J+T, r), T iterations

Innermost run from K to min(K+T, q), T iterations

Total Iterations

T(n) = (p/T x r/T x q/T) \* T3

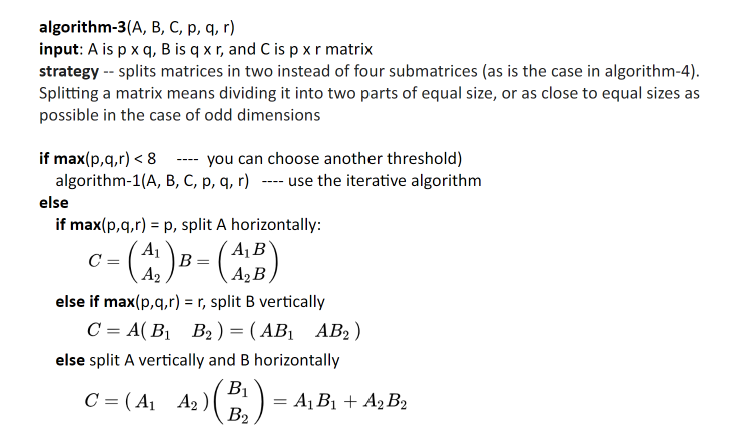
T(n) = (p x r x q)/T3 \* T3

T(n) = p x r x q

O(p x q x r)

O(n3)

**Algorithm-3**



Base Case:

O(1) if n < 8 with constant time for small matrices

Recursive Case:

Recurrence Relation:

For square matrices, p = q = r = n

Master Method

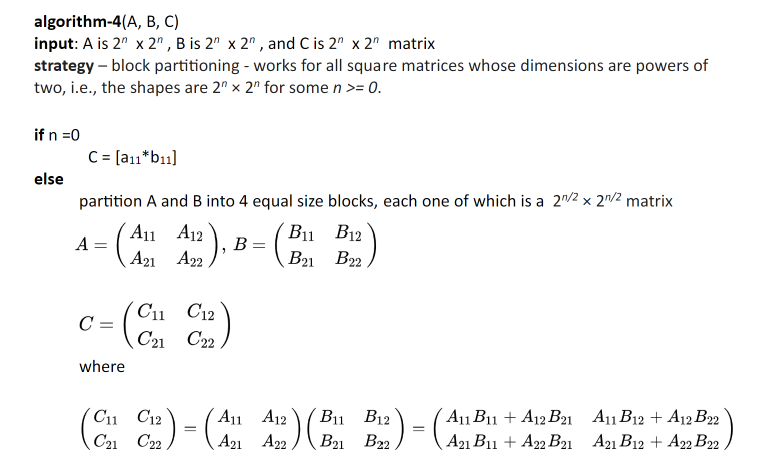
T(n) = 4T(n/2) + O(n2)

a = 4, b = 2, f(n) = O(n2)

log24 = 2

O(n2logn)

**Algorithm-4**



Base Case:

If (p = 1), (q = 1), and (r = 1) a single multiplication is performed

O(1)

Recursive Case:

Matrices A and B are divided into 4 submatrices

Recurrence Relation:

For square matrices q = p = r

Master Method

T(n) = 8T(n/4) + O(n2)

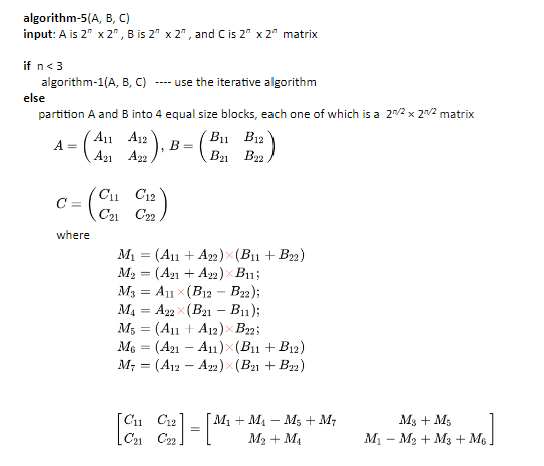
a = 8, b = 4, f(n) = O(n2)

log48 < 2

T(n) = O(n2)

O(n2)

**Algorithm-5**



Base Case:

When n < 3 the algorithm calls Algorithm1

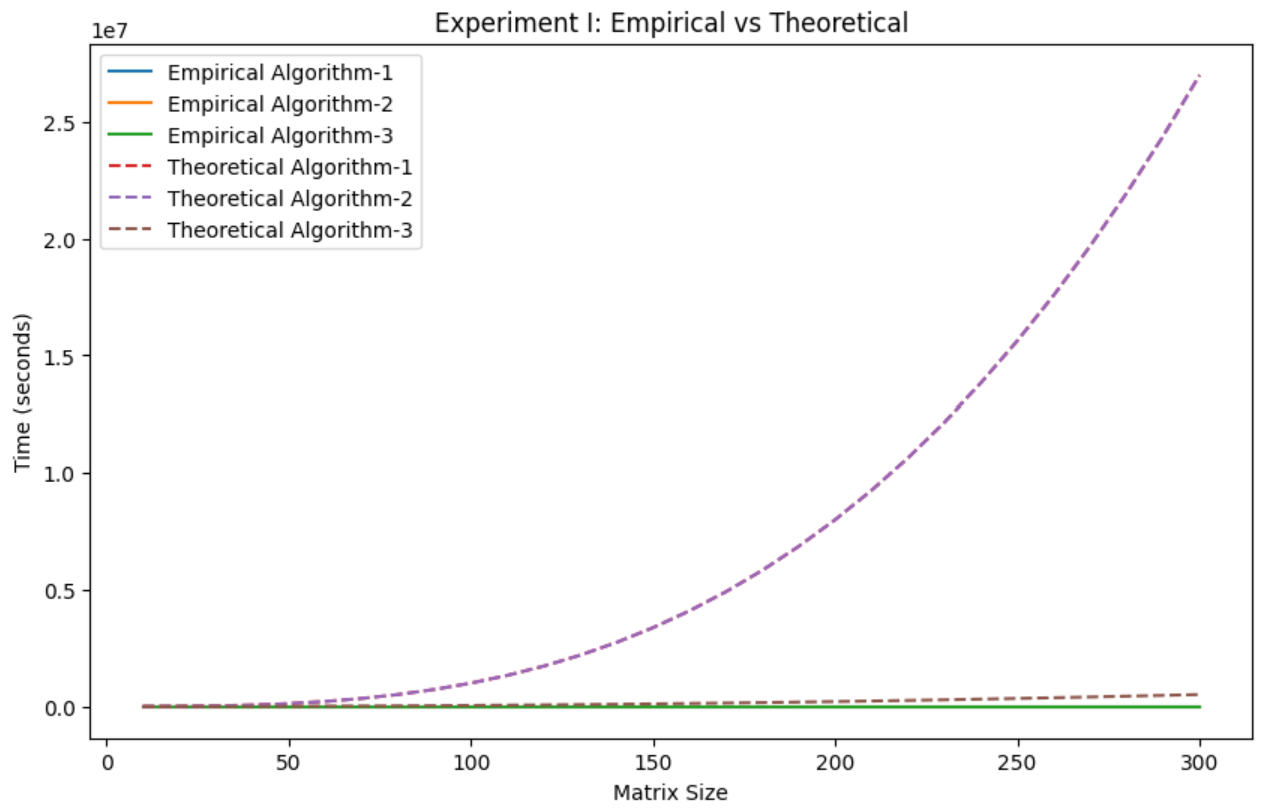
Recursive Case:

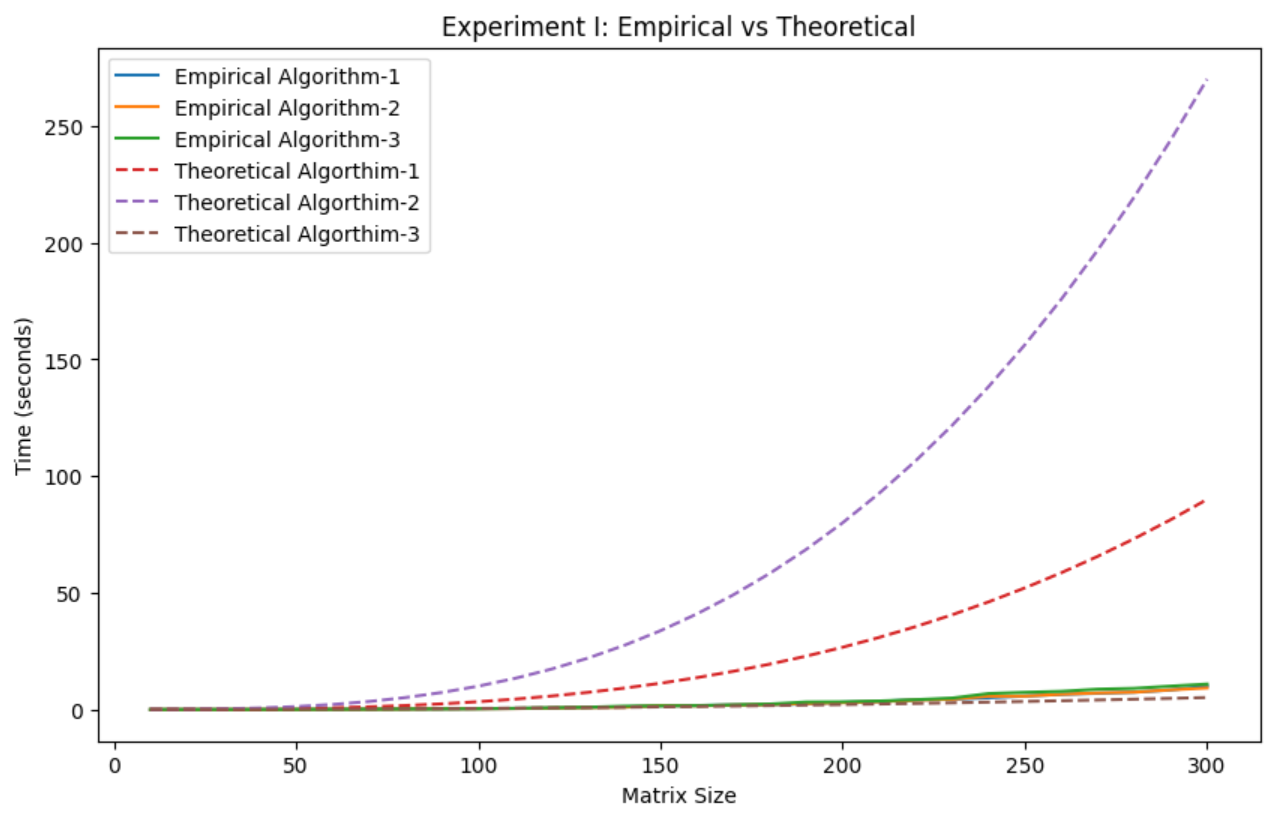
Matrices A and B are divided into 4 submatrices. It makes 8 recursive calls to Algorithm4. Each call has a O(n2)

Recurrence Relation:

The results are combined which take O(n2)

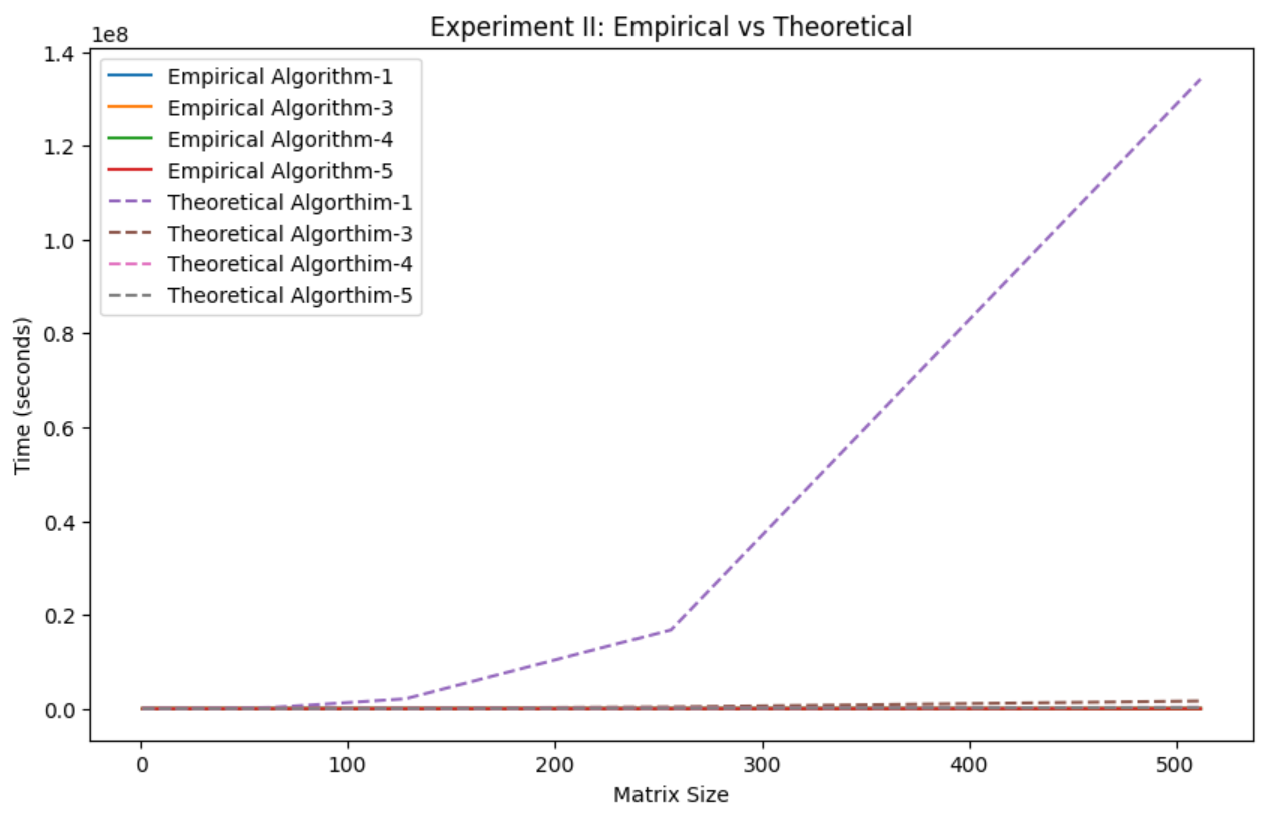
**Experiment I:** The first graph is the non-adjusted graph. The second one I divided the theoretical so I could visualize the comparison.

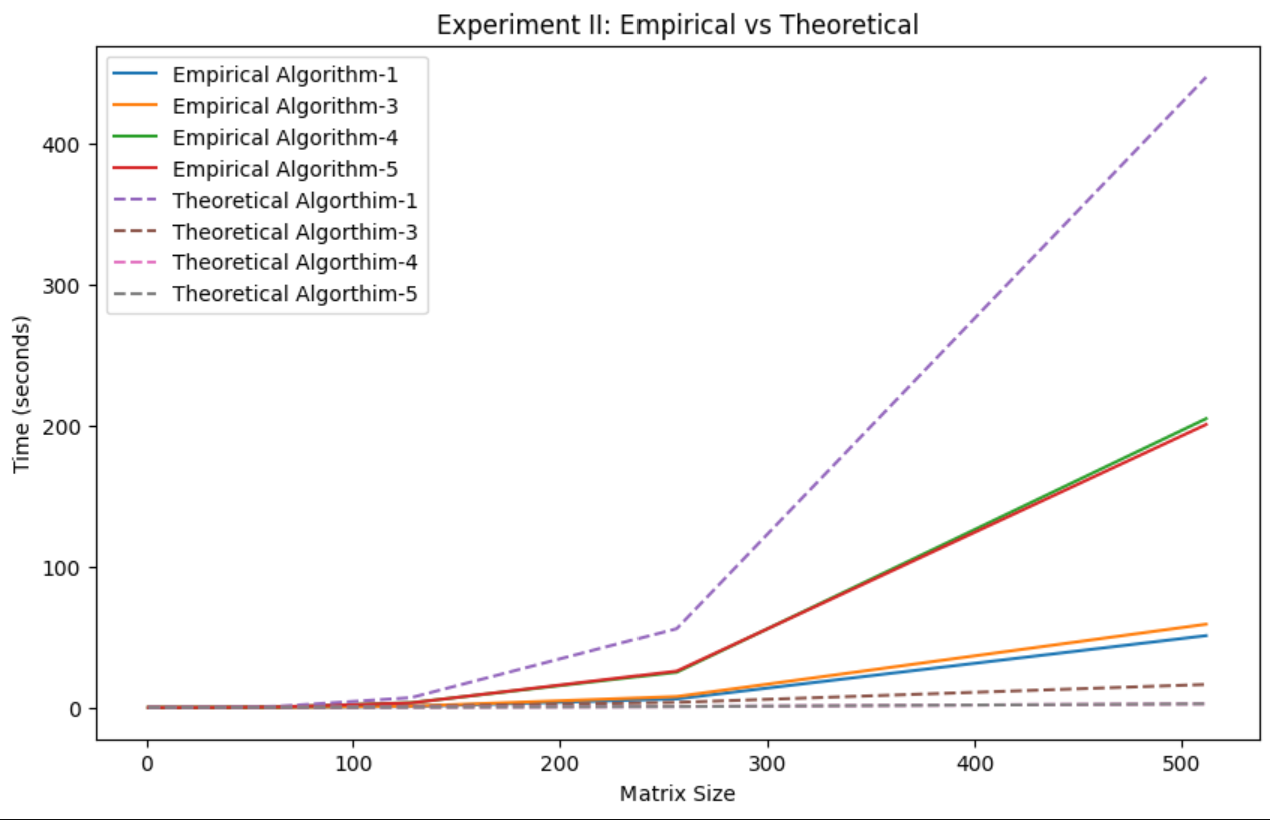
****

****

The graph illustrates the time the different algorithms take as the size of the matrices increase for the Experiment I. Most of the algorithms empirically perform better over time except for Algorithm 3. The actual time outputs execute much faster than it is expected. This can be due to the sizes working better than expected.

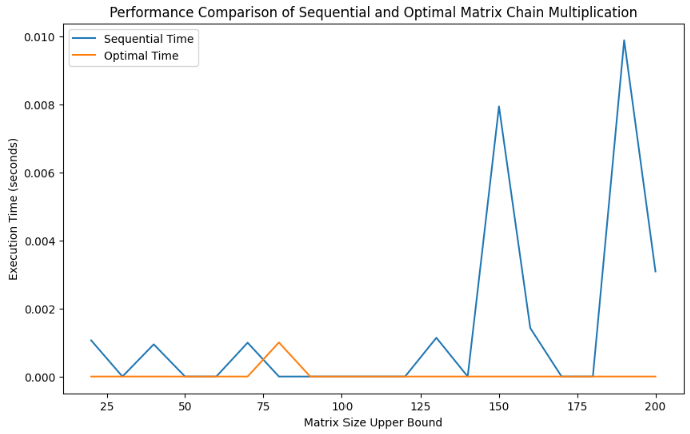
**Experiment II:** The first graph is the non-adjusted graph. The second one I divided the theoretical so I could visualize the comparison.

****

****

The graph illustrates the time the different algorithms take as the size of the matrices increase for the Experiment II. Most of the algorithms empirical perform better over time. Theoretical 3 and 5 grow more slowly over time as the matrix time increases due to the time complexity being smaller. Theoretical 1 grows at a n3 rate so that is why it is much better than the empirical algorithms.

**Matrix-Chain Multiplication­­­**



For the majority of the graph above the optimal time outperforms the sequential time. You can see the orange line which is the orange line below the blue line for the majority of the graph. Towards the end of the graph the optimal time is nearly 4 times better than the sequential time.