Algorithms Assignment Document

Owen Strength - April 8th 2024

Time Complexity Calculations

Algorithm 1: $O(p \times q \times r)$

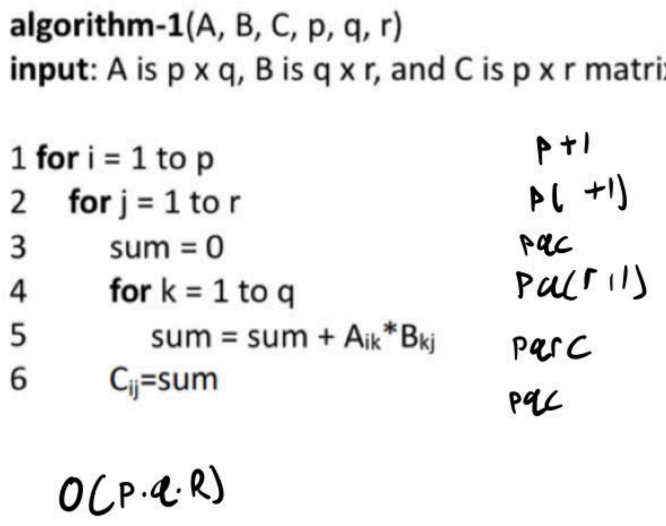
Algorithm 2: $O(\frac{p}{t} \times \frac{q}{t} \times \frac{r}{t} \times t^3) = O(p \times q \times r)$

Algorithm 3: $O(log_2 max(p, q, r) \times p \times q \times r)$

Algorithm 4: $O(2n^3 - n^2)$

Algorithm 5: $O(n^{log_2 7})$

Algorithm 1:

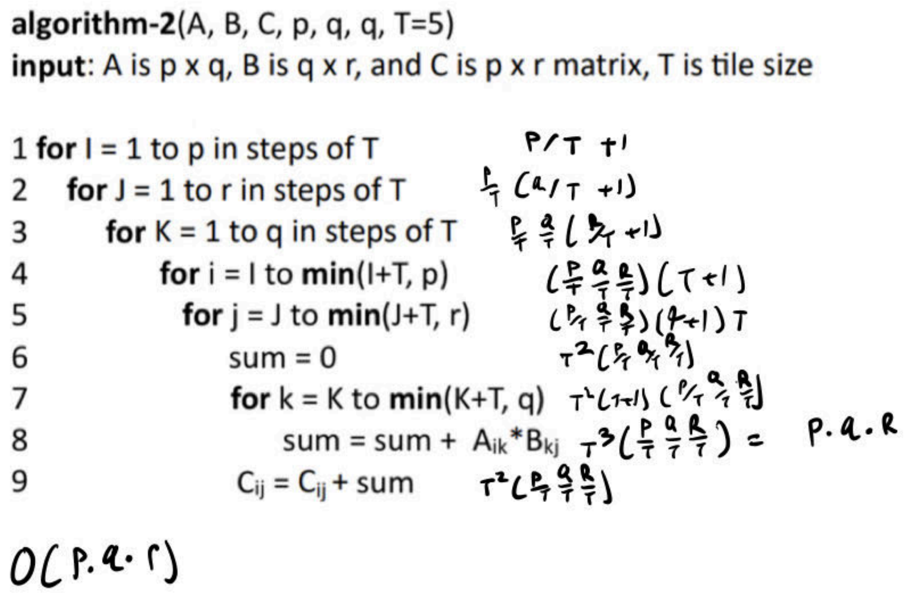


```
algorithm-1(A, B, C, p, q, r)
input: A is p x q, B is q x r, and C is p x r matrix

for i = 1 to p                          p+1
    for j = 1 to r                      p( +1)
        sum = 0                         pqc
        for k = 1 to q                  pq(r+1)
            sum = sum + A_ik*B_kj       pqrc
        C_ij=sum                        pqc
```

$O(p \cdot q \cdot R)$

Algorithm 2:



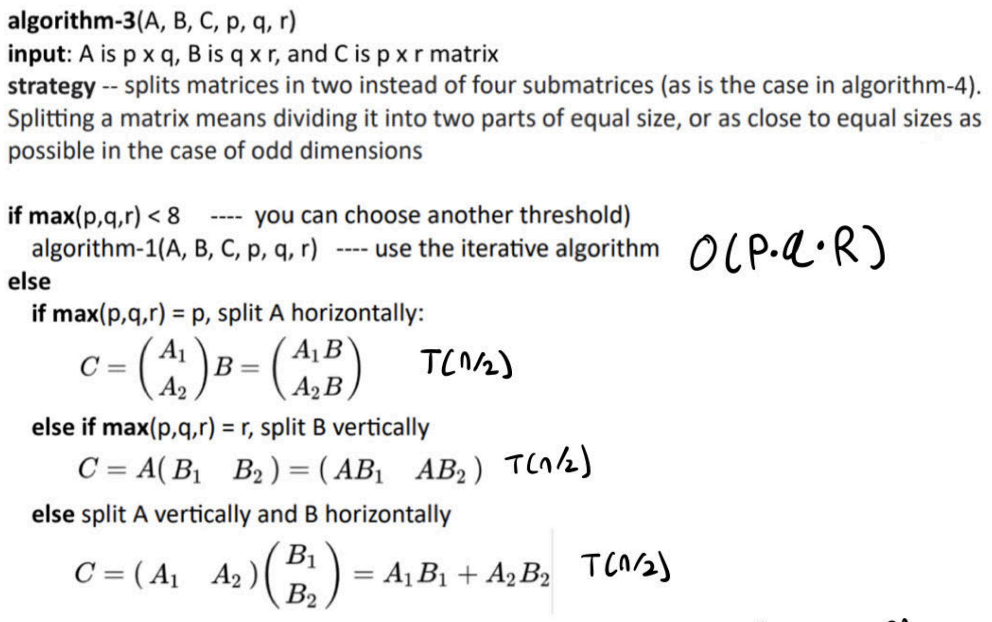
algorithm-2(A, B, C, p, q, q, T=5)

input: A is p x q, B is q x r, and C is p x r matrix, T is tile size

```
for I = 1 to p in steps of T                        $P/T +1$
  for J = 1 to r in steps of T                      $\frac{P}{T}(Q/T+1)$
    for K = 1 to q in steps of T                    $\frac{P}{T}\frac{Q}{T}(R/T+1)$
       for i = I to min(I+T, p)                     $(\frac{P}{T}\frac{Q}{T}\frac{R}{T})(T+1)$
         for j = J to min(J+T, r)                   $(P/T\,\frac{Q}{T}\frac{R}{T})(T+1)T$
             sum = 0                                $T^2(P/T\,Q/T\,R/T)$
             for k = K to min(K+T, q)               $T^2(T+1)(P/T\,\frac{Q}{T}\frac{R}{T})$
                 sum = sum + $A_{ik}$*$B_{kj}$      $T^3(\frac{P}{T}\frac{Q}{T}\frac{R}{T}) = P.Q.R$
             $C_{ij}$ = $C_{ij}$ + sum              $T^2(\frac{P}{T}\frac{Q}{T}\frac{R}{T})$
```

$O(P.Q.r)$

Algorithm 3:



algorithm-3(A, B, C, p, q, r)
input: A is p x q, B is q x r, and C is p x r matrix
strategy -- splits matrices in two instead of four submatrices (as is the case in algorithm-4). Splitting a matrix means dividing it into two parts of equal size, or as close to equal sizes as possible in the case of odd dimensions

```
if max(p,q,r) < 8    ---- you can choose another threshold)
  algorithm-1(A, B, C, p, q, r)  ---- use the iterative algorithm   $O(P\cdot Q\cdot R)$
else
  if max(p,q,r) = p, split A horizontally:
    $C = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} B = \begin{pmatrix} A_1 B \\ A_2 B \end{pmatrix}$   $T(n/2)$
  else if max(p,q,r) = r, split B vertically
    $C = A(\, B_1 \quad B_2 \,) = (\, AB_1 \quad AB_2 \,)$   $T(n/2)$
  else split A vertically and B horizontally
    $C = (\, A_1 \quad A_2 \,)\begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = A_1 B_1 + A_2 B_2$   $T(n/2)$
```

$T(P,Q,R) = T(P/2) + T(Q/2) + T(r/2) + O(P\cdot Q\cdot R)$

$T(n) = 3T(n/2) + O(n^3)$ Masters Method $\Theta(n^3)$

$3[3T(n/2) + O(\frac{n^3}{2^3})] + O(n^3)$

$= 3^2 T(\frac{n}{2^2}) + 3O(n^3) + O(n^3)$

$= 3^3 T(\frac{n}{2^3}) + 3O(n^3) + 3(On^3) + O(n^3)$

$= O(n^3) \cdot \frac{3\log_2 n - 1}{n}$

$= O(\log_2 n \cdot n^3)$

$= O(\log_2(\max(P,Q,R)) \cdot P.Q.R)$

Algorithm 4:

algorithm-4(A, B, C)
input: A is 2^n x 2^n , B is 2^n x 2^n , and C is 2^n x 2^n matrix
strategy – block partitioning - works for all square matrices whose dimensions are powers of two, i.e., the shapes are $2^n \times 2^n$ for some $n >= 0$.

```
if n =0
    C = [a_11*b_11]        O(1)
else
    partition A and B into 4 equal size blocks, each one of which is a 2^{n/2} × 2^{n/2} matrix
```

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

where

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

$$8T(n/2) + O(n^2)$$

$$8[8T(\tfrac{n}{2^2}) + \tfrac{n^2}{2^2}] + n^2$$

$$8^2 T(\tfrac{n}{2^2}) + 2n^2 + n^2$$

$$8^3 T(\tfrac{n}{2^3}) + 4n^2 + 2n^2 + n^2$$

$$8^k = (2^3)^k = (2^k)^3$$

$$(2^k)^3 + n^2[2^0 + 2^1 + 2^2 \dots 2^{k-1}]$$

$$= 2n^3 - n^2$$

Algorithm 5:

```
algorithm-5(A, B, C)
input: A is $2^n$ x $2^n$ , B is $2^n$ x $2^n$ , and C is $2^n$ x $2^n$ matrix

if n < 3
    algorithm-1(A, B, C)   ---- use the iterative algorithm
else
   partition A and B into 4 equal size blocks, each one of which is a $2^{n/2} \times 2^{n/2}$ matrix
```

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},\ B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

where

$$\begin{aligned}
M_1 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \times B_{11}; \\
M_3 &= A_{11} \times (B_{12} - B_{22}); \\
M_4 &= A_{22} \times (B_{21} - B_{11}); \\
M_5 &= (A_{11} + A_{12}) \times B_{22}; \\
M_6 &= (A_{21} - A_{11}) \times (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \times (B_{21} + B_{22})
\end{aligned}$$

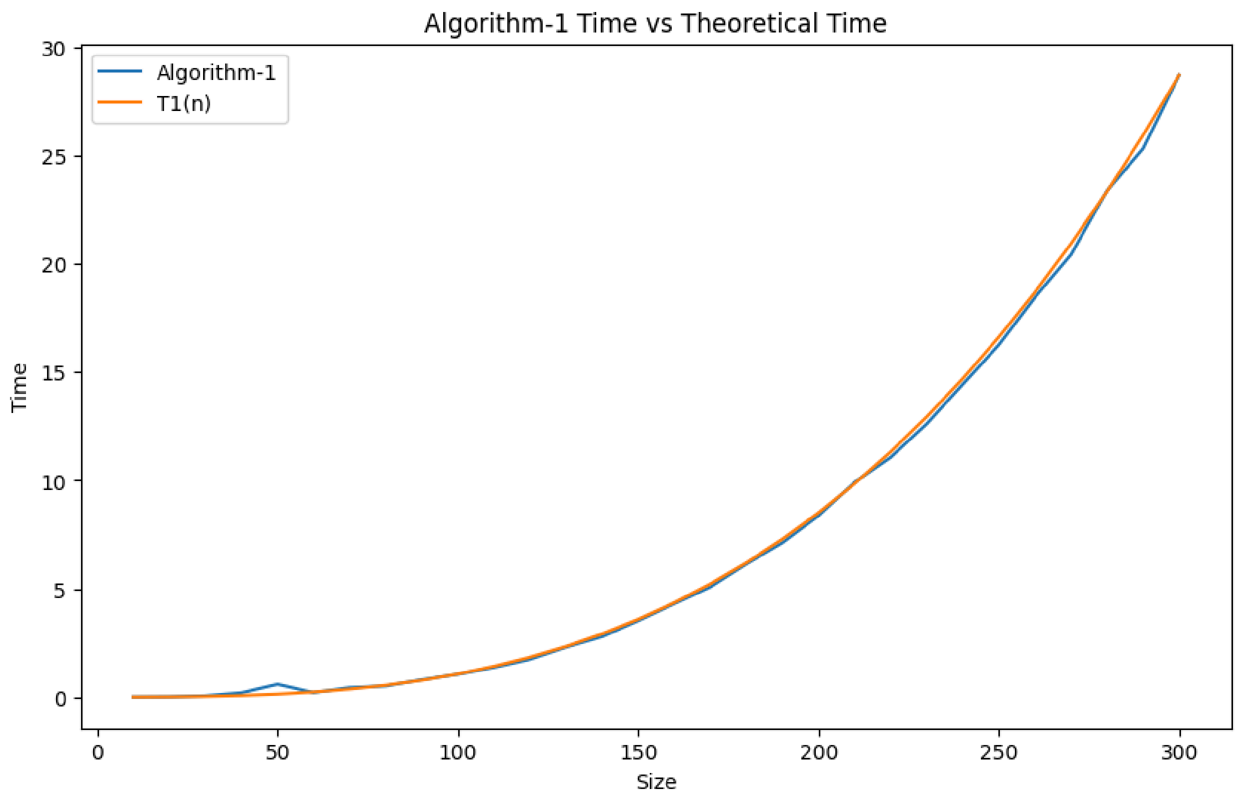
$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

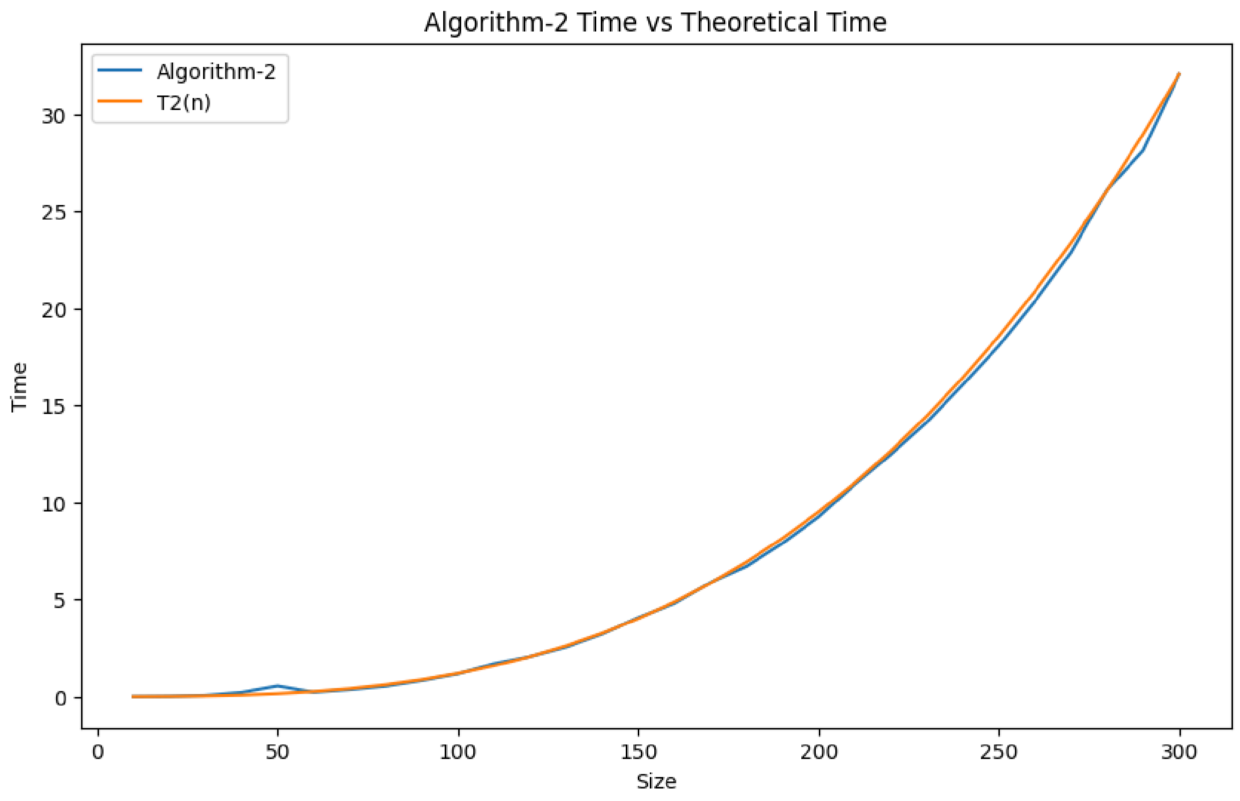
$T(n) = 7T(n/2) + n^2$ Using Masters Method

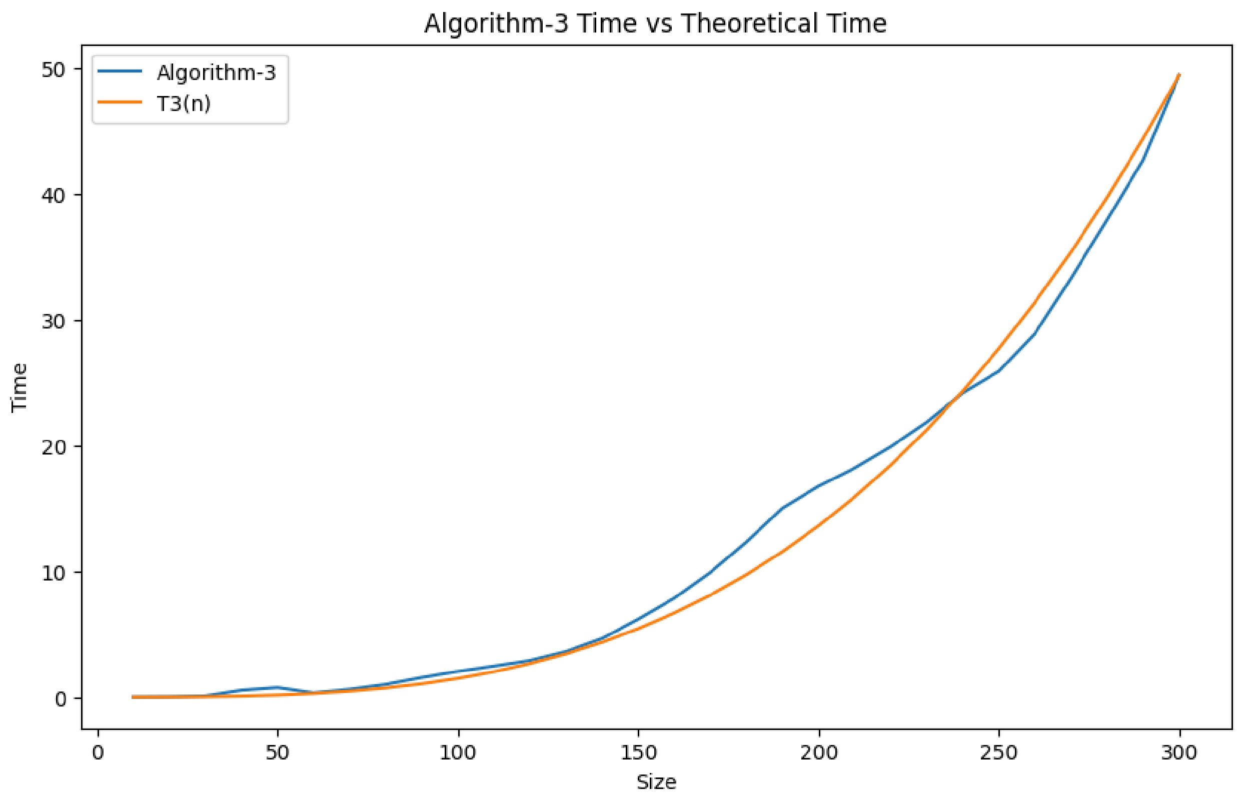
$T(n) = \Theta(n^{\log_2 7})$

Experiment I

The graphs below from Experiment I show that the predicted time and actual time taken are very. There is a small discrepancy on the Algorithm-3 Time vs. Theoretical Time graph, but because the actual time line seems to go between being above and below the theoretical line, we can assume that one average, the theoretical line is correct. Additionally, it is important to note that the time is units of seconds. We can see that Algorithm 1 and Algorithm 2 took about the same amount of time which is expected. We can compare these times to the times gathered from Algorithm 3 to conclude that the time complexity $O(log_2 max(p, q, r) \times p \times q \times r)$ is correct for Algorithm 3.


Algorithm-1 Time vs Theoretical Time
Algorithm-1
T1(n)
Time
Size
0
5
10
15
20
25
30
0
50
100
150
200
250
300

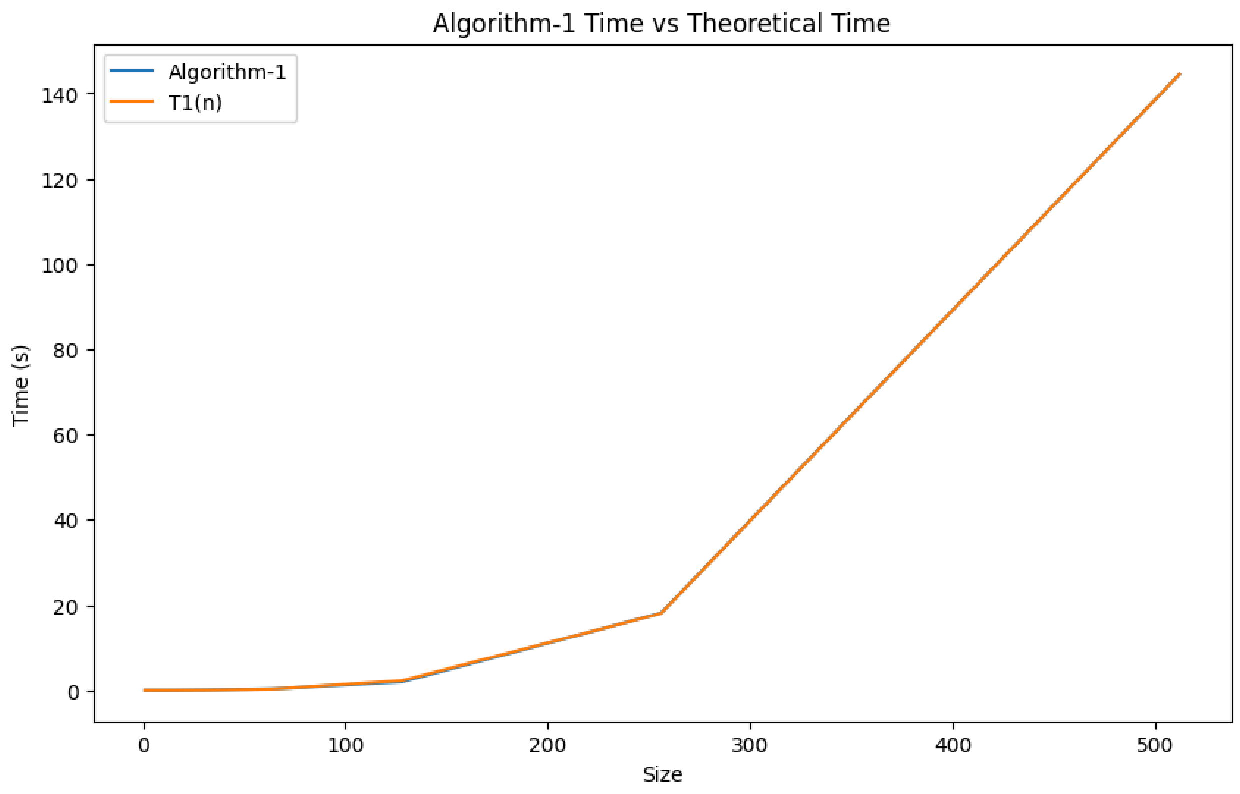

Algorithm-2 Time vs Theoretical Time
Algorithm-2
T2(n)
Time
Size
0
5
10
15
20
25
30
0
50
100
150
200
250
300

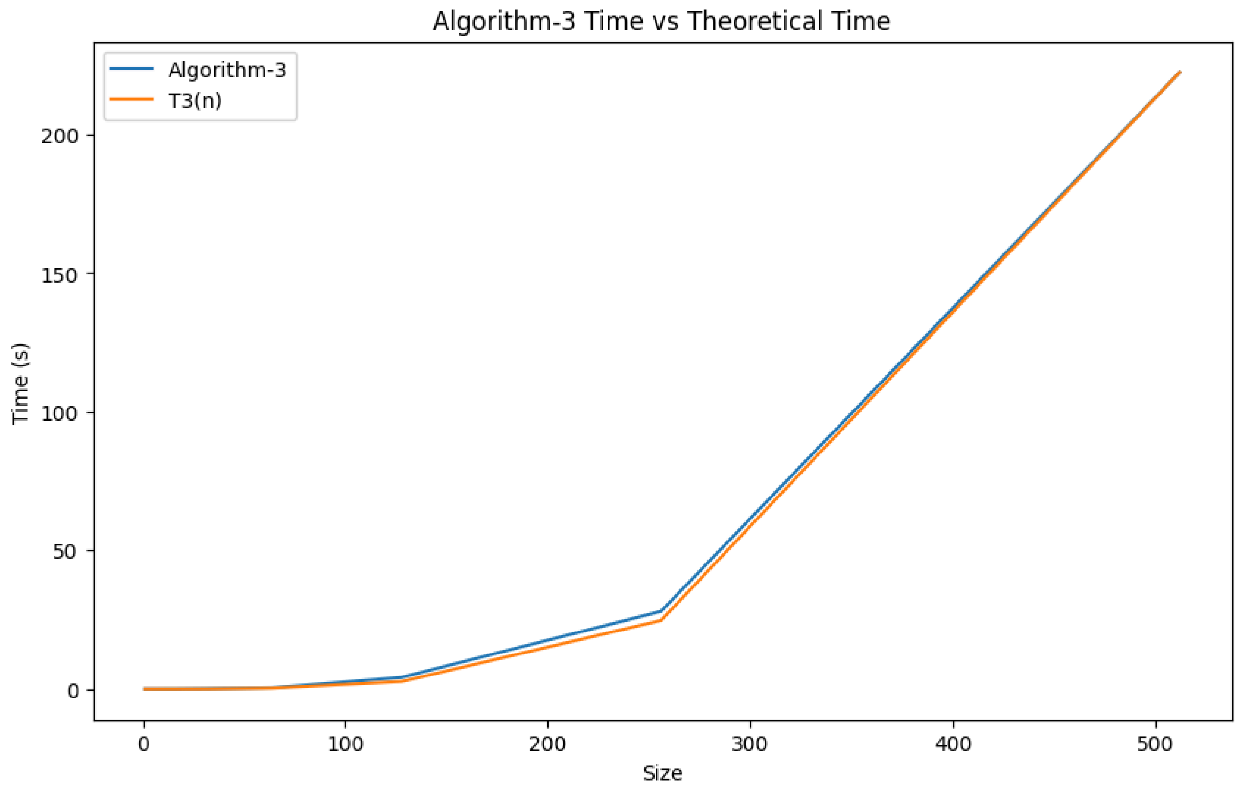


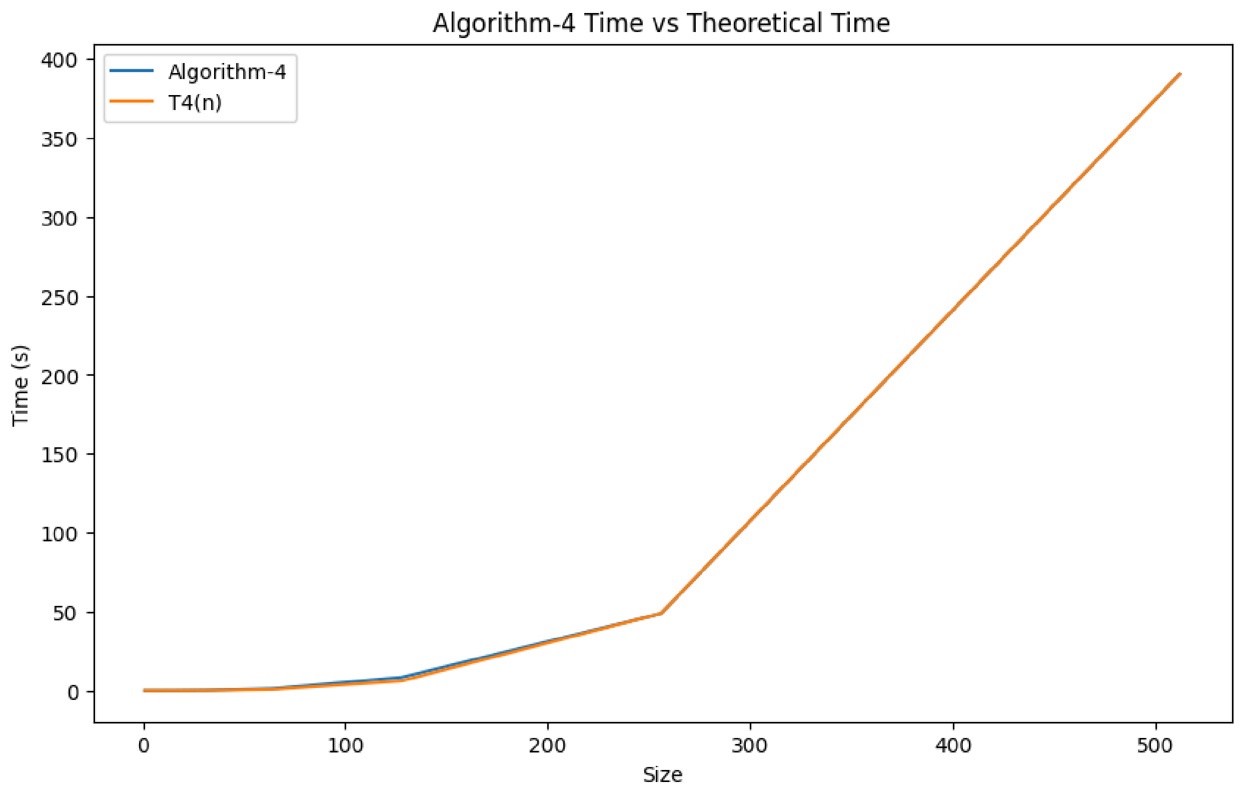
Experiment II

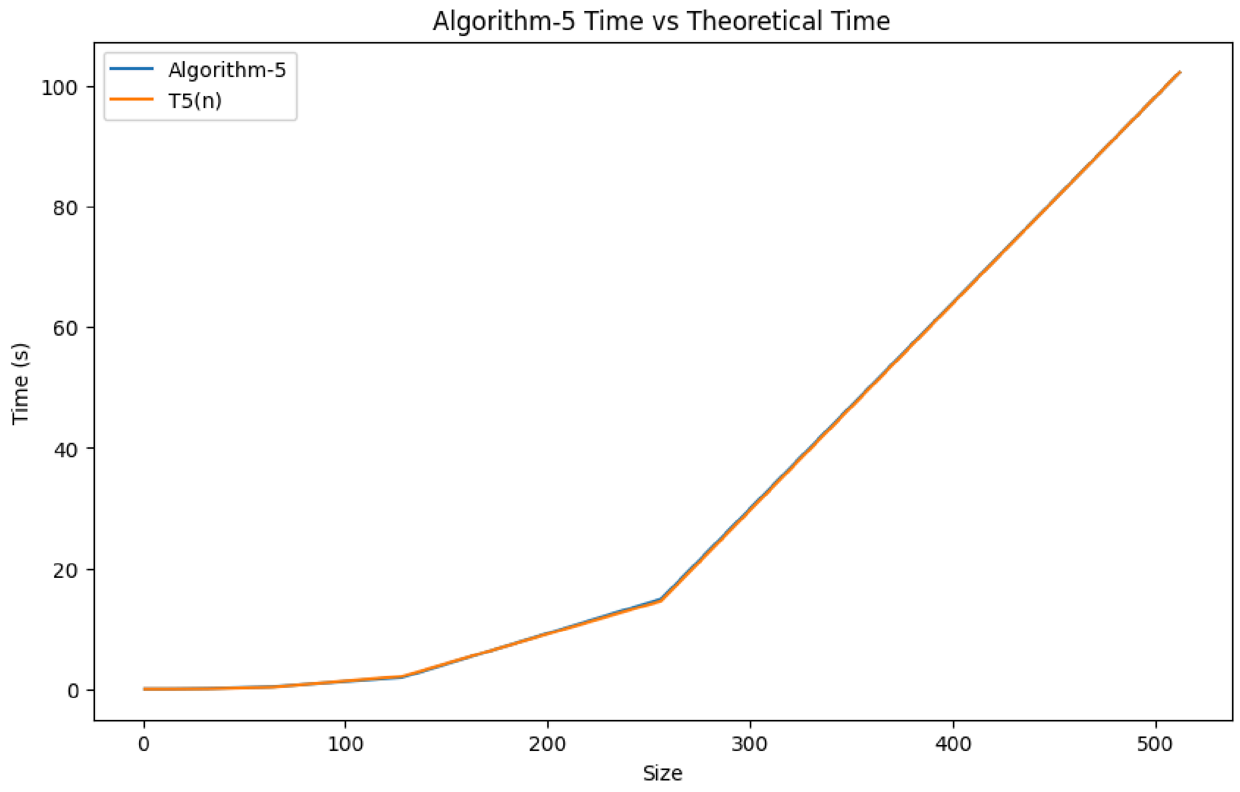
Similar to Experiment I, Experiment II proves the theoretical time complexities to be correct for Algorithm 1, Algorithm 3, Algorithm 4, and Algorithm 5. Additionally, it is important to note that the time is units of seconds. We see in this experiment that the theoretical time complexities are very close to the actual time complexities which confirms our theoretical time complexity calculations. These graphs look similar based on the way they are scaled. Look at the y-axis for better idea of the speed of Algorithm 4 vs Algorithm 5.

We see that the ordering from fastest to slowest algorithm is Algorithm 5, Algorithm 1, Algorithm 3, and then Algorithm 4. We see that Algorithm 4 is significantly slower the other algorithms which is what was predicted in the theoretical time complexity calculations.


Algorithm-1 Time vs Theoretical Time
Algorithm-1
T1(n)
Time (s)
Size
0
20
40
60
80
100
120
140
0
100
200
300
400
500


Algorithm-3 Time vs Theoretical Time
Algorithm-3
T3(n)
Time (s)
Size
0
50
100
150
200
0
100
200
300
400
500


Algorithm-4 Time vs Theoretical Time
Algorithm-4
T4(n)
Time (s)
Size
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500

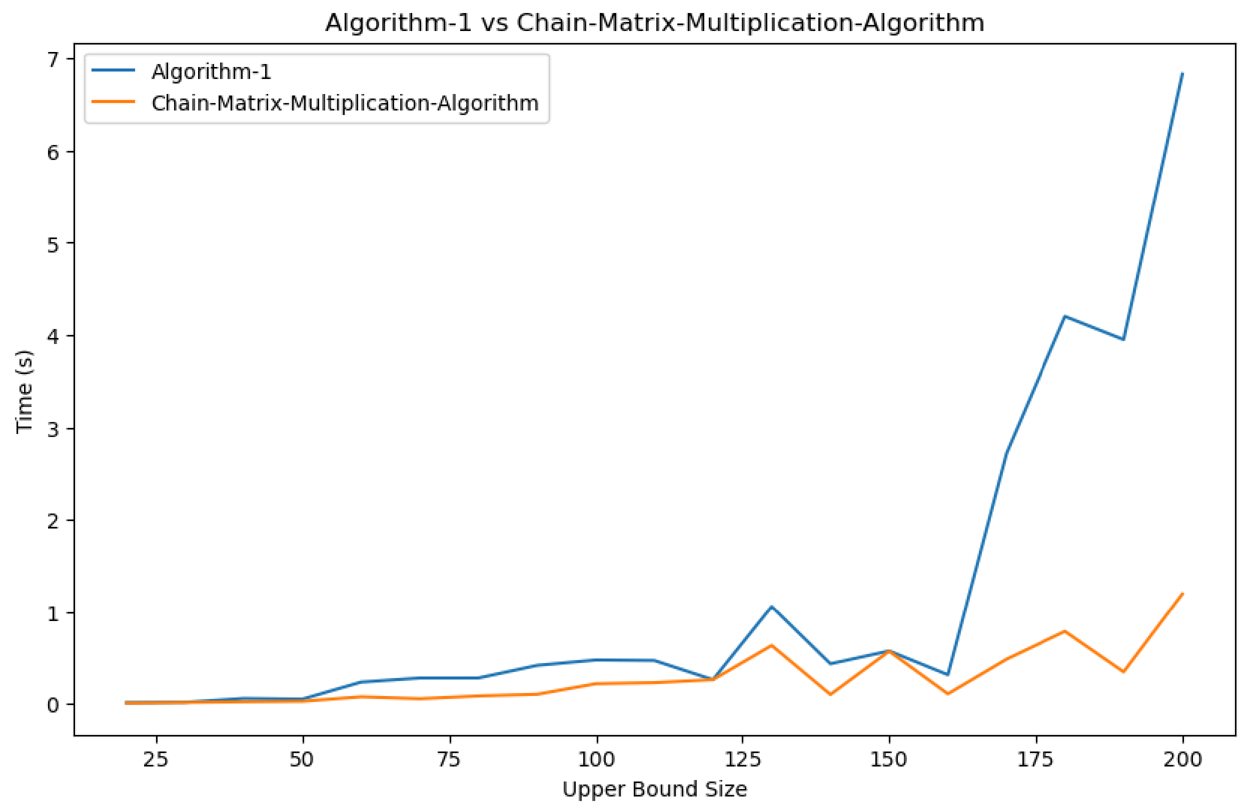

Algorithm-5 Time vs Theoretical Time
Algorithm-5
T5(n)
Time (s)
Size
0
20
40
60
80
100
0
100
200
300
400
500

Maxtrix Chain Experiment

The Matrix Experiment had varying results, the assignment stated that we should plot the time it takes for sequential multiplication and optimal multiplication to run against the upper bound of the matrix. I think plotting against the upper bound is flawed because we are randomly generating the sizes of the matrices. 1 matrix chain could have a larger resultant matrix than the matrix chain with a larger upper bound because these dimensions are generated randomly. We are not guaranteeing that for example matrix chain 14 will have a smaller resultant matrix than matrix chain 15. We should be plotting against the size of the resultant matrix.

This fundamental flaw explains the changes in direction in the graph, but generally as the upper bound increases so does the time taken. An interesting observation is that as the upper bound increases the more significant the difference in time taken to compute the product between the optimal vs sequential ordering. When the upper bound is 200, the optimal ordering is about 6 times faster than the sequential ordering. When the upperbound is around 100, the optimal ordering is about 1.5 - 2 times faster than the sequential ordering

Additionally at upper bound 150, we see that sequential and optimal ordering times are the same, which shows that the matrix can already be in its optimal ordering.


Algorithm-1 vs Chain-Matrix-Multiplication-Algorithm
Algorithm-1
Chain-Matrix-Multiplication-Algorithm
Time (s)
Upper Bound Size