2. Calculate
T(n)
and the order of complexity of each algorithm in the
big-Oh
or
Theta

notation using any of the methods we discussed in class. (a) Show your calculations and

derivations step by step in detail, then (b) state the polynomial (or recurrence relation for

the recursive algorithms) T(n) and (c) determine the complexity order of the algorithm.

Incomplete answers to this part will get a zero, not a partial grade

Algorithm 1:

There are 3 nested loops, each iterates through a dimension of the matrices a,b, and c. The outer loop goes from 1 to p, middle loop from 1 to r, and the inner loop from 1 to q. The innermost loop has constant time ops.

The time complexity is O(p* r * q)
The recurrence relation is T(n) = O(p * r * q)

Algorithm 2:

Loops are divided into blocks the size of t. The outer loops iterate from 1 to p and from 1 to r in steps of T, and the inner loop is iterated from 1 to q in steps of T. Constant time operations inside the innermost loop.

Time complexity is O((p * r * q) / (T^2))
Recurrence relation is T(n) = O((p * r * q) / (T^2))

Algorithm 3:

If (p,q,r) < 8, iterative algorithm is used

Time complexity is T(n)= O(p * r * q)

If matrices are split until reaching base case, time complexity would be O(n^log2(7)) where n is the dimension of the matrices.

Algorithm 4:

Uses divide and conquer strategy with block partitioning. Partitions into 4 equal sixe blocks until base cased is reached

.

The recurrence relation is T(n) = 8*T(n/2) + O(n^2) for matrix multiplication and O(1) for other operations

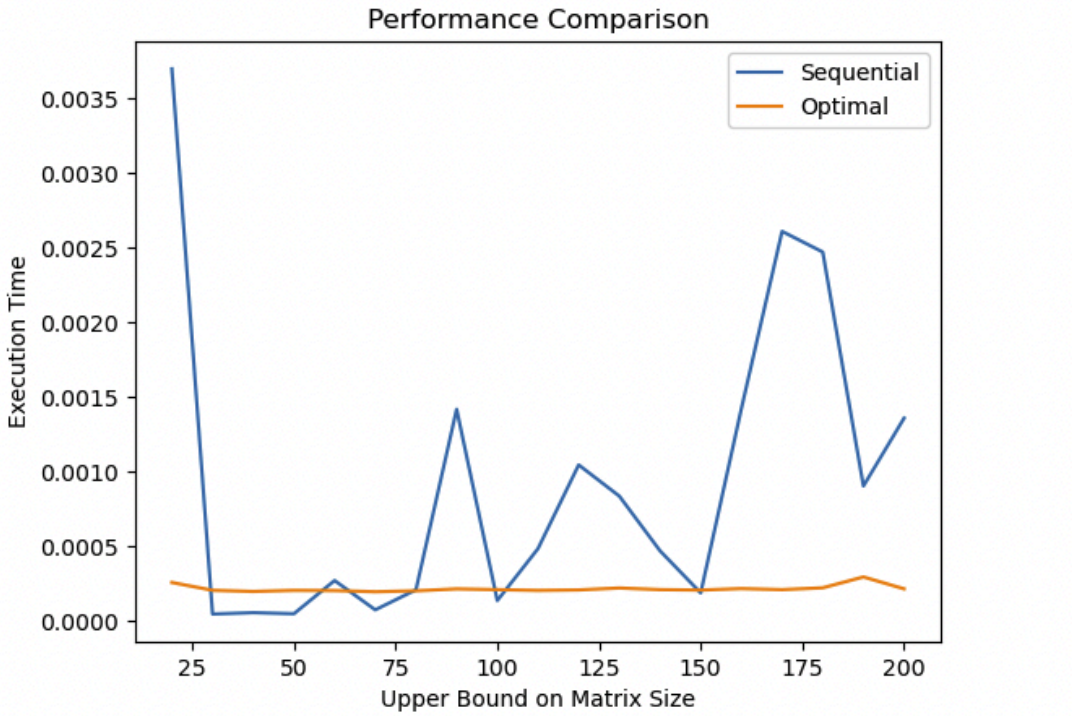
The complexity order is O(n^3) using the master theorem.

Algorithm 5:

Uses block partitioning and dividend conquer. Partitions A and B into 4 equal sized blocks recursively until base case. Performs 7 recursive matrix multiplications, then combines them

The recurrence relation is T(n) = 7 * T(n/2) + O(n^2)

	Matrix Chain ID	Upper Bound on Matrix Size	Sequential Time	Optimal Time
0	2	20	0.003694	0.000260
1	3	30	0.000049	0.000208
2	4	40	0.000058	0.000200
3	5	50	0.000051	0.000207
4	6	60	0.000273	0.000206
5	7	70	0.000077	0.000198
6	8	80	0.000211	0.000204
7	9	90	0.001418	0.000217
8	10	100	0.000139	0.000212
9	11	110	0.000486	0.000207
10	12	120	0.001046	0.000210
11	13	130	0.000837	0.000223
12	14	140	0.000472	0.000212
13	15	150	0.000190	0.000209
14	16	160	0.001423	0.000219
15	17	170	0.002607	0.000212
16	18	180	0.002469	0.000224
17	19	190	0.000905	0.000297
18	20	200	0.001359	0.000217



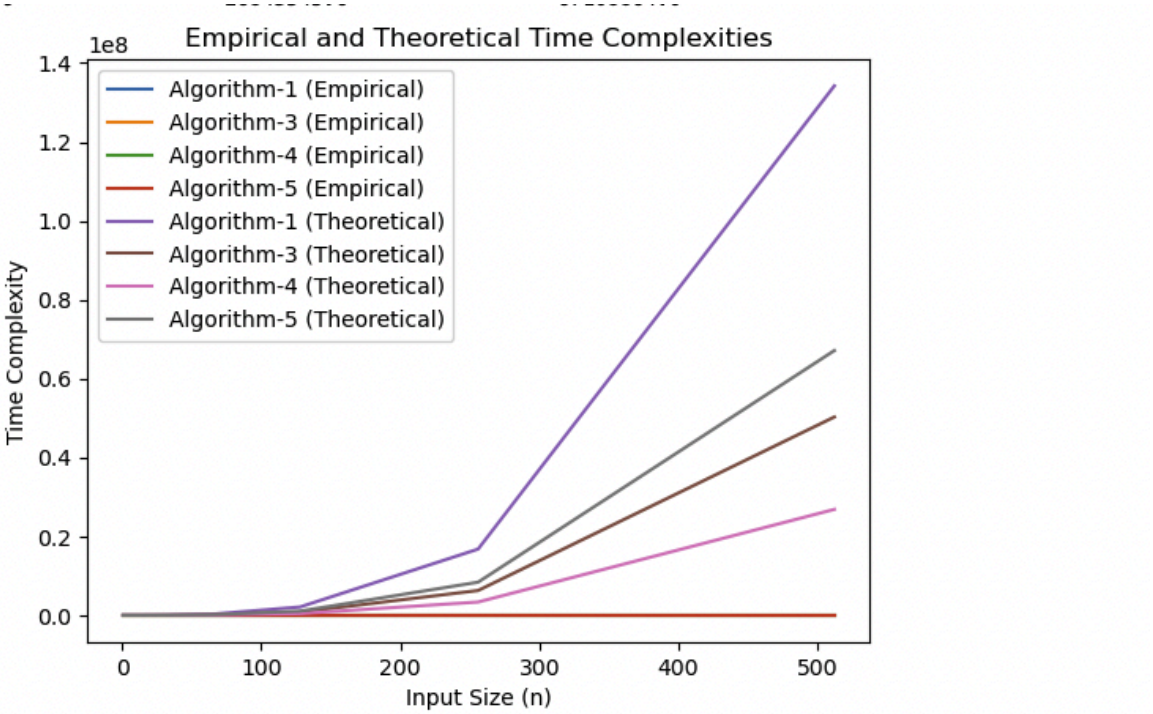
Matrix chain multiplication graph: This graph shows that the optimal solution outperforms the sequential solution significantly. Even though the sequential solution is faster at specific data points, the optimal solution is a more consistent performer that could be over twice as fast in certain scenarios.

	Size	Algorithm-1 (Empirical)	Algorithm-3 (Empirical)	\
0	1	2.384186e-07	2.384186e-07	
1	2	4.291534e-07	4.291534e-07	
2	4	0.000000e+00	0.000000e+00	
3	8	1.192093e-07	1.192093e-07	
4	16	0.000000e+00	0.000000e+00	
5	32	0.000000e+00	0.000000e+00	
6	64	1.907349e-07	1.907349e-07	
7	128	0.000000e+00	0.000000e+00	
8	256	0.000000e+00	0.000000e+00	
9	512	5.245209e-07	5.245209e-07	

	Algorithm-4 (Empirical)	Algorithm-5 (Empirical)	\
0	2.384186e-07	2.384186e-07	
1	4.291534e-07	4.291534e-07	
2	0.000000e+00	0.000000e+00	
3	1.192093e-07	1.192093e-07	
4	0.000000e+00	0.000000e+00	
5	0.000000e+00	0.000000e+00	
6	1.907349e-07	1.907349e-07	
7	0.000000e+00	0.000000e+00	
8	0.000000e+00	0.000000e+00	
9	5.245209e-07	5.245209e-07	

	Algorithm-1 (Theoretical)	Algorithm-3 (Theoretical)	\
0	1	1.250000e-01	
1	8	2.000000e+00	
2	64	2.000000e+01	
3	512	1.760000e+02	
4	4096	1.472000e+03	
5	32768	1.203200e+04	
6	262144	9.728000e+04	
7	2097152	7.823360e+05	
8	16777216	6.275072e+06	
9	134217728	5.026611e+07	

	Algorithm-4 (Theoretical)	Algorithm-5 (Theoretical)
0	0.2	0.5
1	1.6	4.0
2	12.8	32.0
3	102.4	256.0
4	819.2	2048.0
5	6553.6	16384.0
6	52428.8	131072.0
7	419430.4	1048576.0
8	3355443.2	8388608.0
9	26843545.6	67108864.0

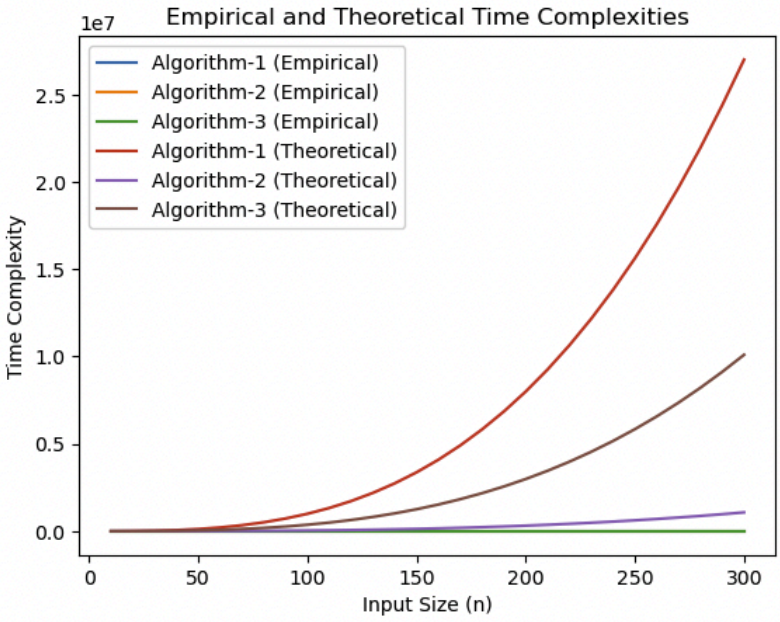


Empirical vs. Theoretical time complexities: Experiment 2: For this experiment, the empirical time does not line up with the theoretical complexity for the smaller input sizes. This could be due to the presence of hidden constants. However, as the size of the input increases, the empirical time seems to be closer to the theoretical complexity figure, indicating correct behavior.

	Size	Algorithm-1 (Empirical)	Algorithm-2 (Empirical
0	10	1.748403e-07	1.748403e-0
1	20	1.986821e-07	1.986821e-0
2	30	3.973643e-07	3.973643e-0
3	40	0.000000e+00	0.000000e+0
4	50	2.384186e-07	2.384186e-0
5	60	1.351039e-07	1.351039e-0
6	70	1.271566e-07	1.271566e-0
7	80	3.337860e-07	3.337860e-0
8	90	4.371007e-07	4.371007e-0
9	100	1.986821e-07	1.986821e-0
10	110	1.033147e-07	1.033147e-0
11	120	1.430511e-07	1.430511e-0
12	130	3.814697e-07	3.814697e-0
13	140	2.384186e-07	2.384186e-0
14	150	3.019969e-07	3.019969e-0
15	160	5.563100e-08	5.563100e-0
16	170	1.589457e-07	1.589457e-0
17	180	1.033147e-07	1.033147e-0
18	190	2.781550e-07	2.781550e-0
19	200	2.463659e-07	2.463659e-0
20	210	2.145767e-07	2.145767e-0
21	220	1.827876e-07	1.827876e-0
22	230	2.861023e-07	2.861023e-0
23	240	7.152557e-08	7.152557e-0
24	250	1.986821e-07	1.986821e-0
25	260	2.622604e-07	2.622604e-0
26	270	2.861023e-07	2.861023e-0
27	280	2.702077e-07	2.702077e-0
28	290	2.304713e-07	2.304713e-0
29	300	2.940496e-07	2.940496e-0

	Algorithm-3 (Empirical)	Algorithm-1 (Theoretical)	\
0	1.748403e-07	1000	
1	1.986821e-07	8000	
2	3.973643e-07	27000	
3	0.000000e+00	64000	
4	2.384186e-07	125000	
5	1.351039e-07	216000	
6	1.271566e-07	343000	
7	3.337860e-07	512000	
8	4.371007e-07	729000	
9	1.986821e-07	1000000	
10	1.033147e-07	1331000	
11	1.430511e-07	1728000	
12	3.814697e-07	2197000	
13	2.384186e-07	2744000	
14	3.019969e-07	3375000	
15	5.563100e-08	4096000	
16	1.589457e-07	4913000	
17	1.033147e-07	5832000	
18	2.781550e-07	6859000	
19	2.463659e-07	8000000	
20	2.145767e-07	9261000	
21	1.827876e-07	10648000	
22	2.861023e-07	12167000	
23	7.152557e-08	13824000	
24	1.986821e-07	15625000	
25	2.622604e-07	17576000	
26	2.861023e-07	19683000	
27	2.702077e-07	21952000	
28	2.304713e-07	24389000	
29	2.940496e-07	27000000	

	Algorithm-2 (Theoretical)	Algorithm-3 (Theoretical)
0	40.0	350.0
1	320.0	2900.0
2	1080.0	9900.0
3	2560.0	23600.0
4	5000.0	46250.0
5	8640.0	80100.0
6	13720.0	127400.0
7	20480.0	190400.0
8	29160.0	271350.0
9	40000.0	372500.0
10	53240.0	496100.0
11	69120.0	644400.0
12	87880.0	819650.0
13	109760.0	1024100.0
14	135000.0	1260000.0
15	163840.0	1529600.0
16	196520.0	1835150.0
17	233280.0	2178900.0
18	274360.0	2563100.0
19	320000.0	2990000.0
20	370440.0	3461850.0
21	425920.0	3980900.0
22	486680.0	4549400.0
23	552960.0	5169600.0
24	625000.0	5843750.0
25	703040.0	6574100.0
26	787320.0	7362900.0
27	878080.0	8212400.0
28	975560.0	9124850.0
29	1080000.0	10102500.0



Empirical vs. Theoretical time complexities: Experiment 1: for this experiment, the empirical time does not consistently match each algorithm's predicted complexity as input size increases. This seems to be occurring because all the empirical time is the same. This could be a result of the same input sizes giving a consistent answer for empirical time. Ensuring different matrix pairs could help reflect the true behavior of the algorithms.

```
Algorithm-1: 90,100,110,120;202,228,254,280;314,356,398,440;426,484,542,600
Algorithm-2: 90,100,110,120;202,228,254,280;314,356,398,440;426,484,542,600
Algorithm-3: 90,100,110,120;202,228,254,280;314,356,398,440;426,484,542,600
Algorithm-4: 90,100,110,120;202,228,254,280;314,356,398,440;426,484,542,600
Algorithm-5: 90,100,110,120;202,228,254,280;314,356,398,440;426,484,542,600
```

Matrix multiplication output