M2: Hands-On: Efficiency and Empirical Analysis

Due Feb 12 at 11:59pm **Points** 5 **Questions** 5 **Time Limit** None
Allowed Attempts Unlimited

Instructions

500	0.042	1.35
1000	0.112	2.67
2000	0.340	3.04
4000	1.298	3.82
8000	5.334	4.11
16000	21.880	4.10
32000	85.763	3.92
64000	345.634	4.03

Of course not all algorithms have polynomial time complexity. Common orders of growth that we will see in this course include *log N*, *N*, *N log N*, *N^2*, *N^3*, *2^N*, and *N!*. For the purposes of this lab, however, we will restrict ourselves to talking about algorithms with time complexity proportional to a *polynomial*. (*f(N) = N^k* for *k* > 0)

The `TimeComplexity` class demonstrates how you can generate data that will allow you to characterize polynomial time complexity.

1. Open `TimeComplexity.java` and compile it.
2. Run this program and observe its output.
3. Interact with this code and make sure that you understand everything that it does and how you might apply it for your own purposes

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	5,753 minutes	5 out of 5

Score for this attempt: **5** out of 5
Submitted Feb 12 at 9:35pm
This attempt took 5,753 minutes.

Create and fill in a table like the one below to record the data generated by the TimeComplexity class. Depending on the speed of your computer, you should be able to generate five rows of data in no more than 10 or 15 minutes.

N	Elapsed Time	Ratio

Question 1

1 / 1 pts

Based on your analysis of the data generated by the TimeComplexity class, which of the following expressions best characterizes this program's time complexity function?

○ $O(N)$

○ $O(N^2)$

Correct! [✓ $O(N^3)$]

○ $O(N^4)$

Question 2

1 / 1 pts

Suppose you attempted to empirically discover the big-O running time of a program, and you were able to generate the following timing data.

N	Time	Ratio
128	0.622	—
256	8.886	14.286
512	141.048	15.873
1024	2249.704	15.950
2048	35958.996	15.984

In the table above, the N column records the size of the input for each run, the Time column records the elapsed time in seconds for each run, and Ratio is the elapsed time for the current run divided by the elapsed time for the previous run (i.e., $Time_{\mathrm{i}}/Time_{\mathrm{i\text{-}1}}$).

Based on the timing data presented in this table, what is the most reasonable conclusion regarding the underlying big-O time complexity of the program being timed?

A. $O(15)$

B. $O(\log_{15} N)$

C. $O(N)$

D. $O(N^4)$

A

B

C

Correct! [✓ D]

Question 3

1 / 1 pts

Suppose you attempted to empirically discover the big-oh running time of a program, and you were able to generate the following data. What big-oh time complexity is suggested by this data?

Run	N	Time	Ratio	lg *Ratio*
0	200	0.12		
1	400	0.13	1.10	0.14
2	800	0.76	5.68	2.51
3	1600	1.88	2.47	1.31
4	3200	7.50	3.99	1.99
5	6400	29.85	3.98	1.99
6	12800	131.37	4.40	2.14
7	25600	520.78	3.96	1.99

A. $O(\log N)$

B. $O(N)$

C. $O(N^2)$

D. $O(N^4)$

A

B

Correct!

[✓ C]

D

Question 4

1 / 1 pts

Suppose you attempted to empirically discover the big-O running time of a program, and you were able to generate the following timing data.

N	Time	Ratio
128	0.513	—
256	3.223	6.283
512	23.444	7.274
1024	186.849	7.970
2048	1493.114	7.991

In the table above, the N column records the size of the input for each run, the Time column records the elapsed time in seconds for each run, and Ratio is the elapsed time for the current run divided by the elapsed time for the previous run (i.e., $Time_i / Time_{i-1}$).

Based on the timing data presented in this table, what is the most reasonable conclusion regarding the underlying big-O time complexity of the program being timed?

A. $O(8)$

B. $O(\log_3 N)$

C. $O(N^3)$

D. $O(N^8)$

A

B

Correct!

[✓ C]

D

Question 5

1 / 1 pts

Suppose you have implemented an algorithm in a method named `foo`, which takes an array of N floating point numbers as data. Suppose also that this algorithm has $O(N^3)$ best, average, and worst case time complexity and that a timing analysis of `foo` showed that approximately 2 seconds were required to process an array of size $N = 256$.

What is the largest array (N) that `foo` could process in less than one hour?

A. 512

B. 1024

C. 2048

D. 4096

A

B

Correct! [✓ C]

D

Quiz Score: **5** out of 5