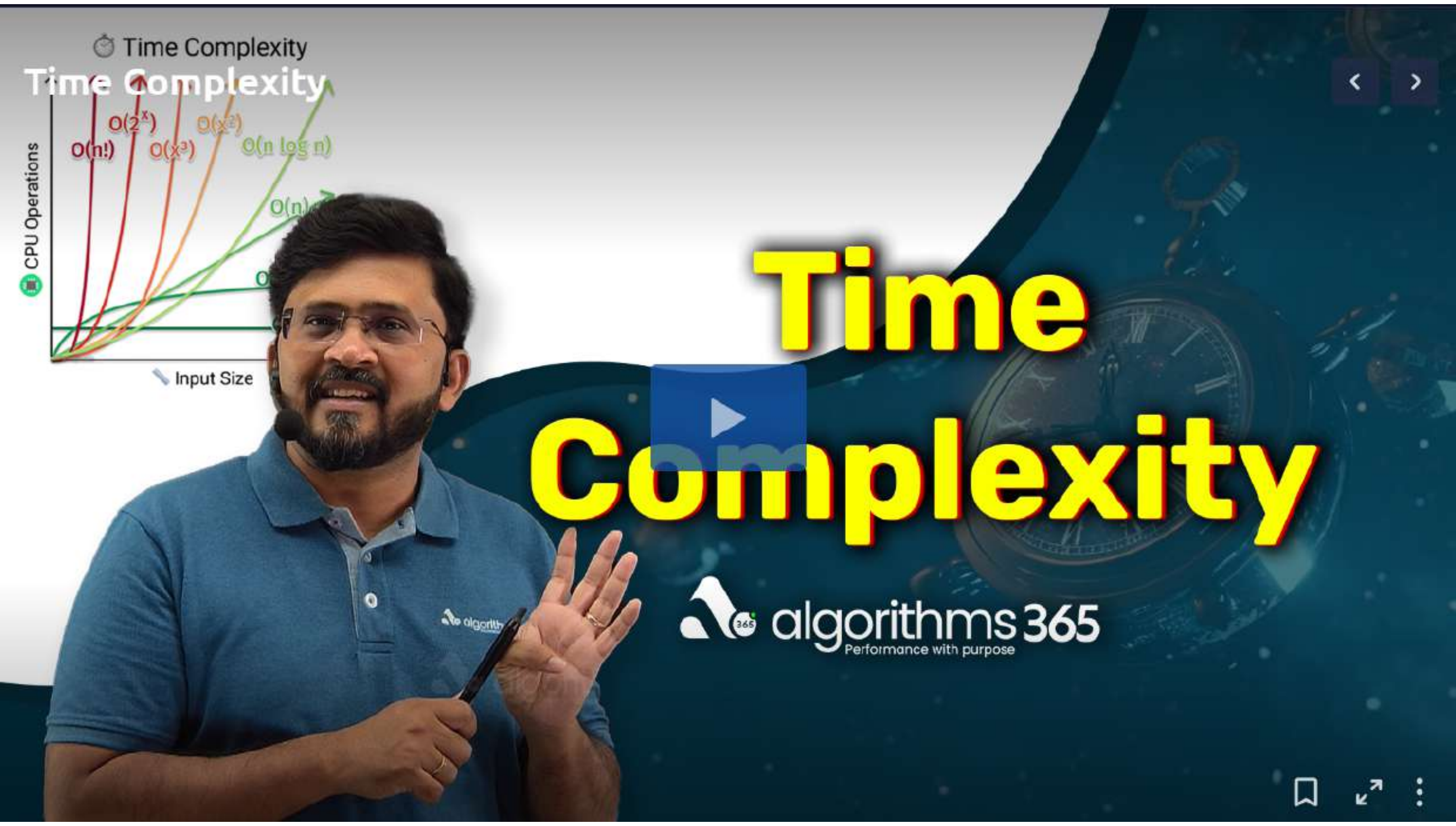
Time Complexity
Time Complexity
O(n!)
O(2^x)
O(x³)
O(x²)
O(n log n)
O(n)
CPU Operations
Input Size
Time
Complexity
algorithms365
Performance with purpose

16 - Tues - 2025
09

Day 27

Time Complexity.

* Time Complexity is a way to describe how long an algorithm. (a step-by-step method to solve a problem) takes to run, depending on the size of the input it's given.

1. Identify the Basic Operation's

Determine the most significant operations in the algorithm. these are operations that most affect the algorithm's runtime, such as comparisions, assignments, and Arithmetic operations

2. Count the Basic operation.

Count how many times these basic operations are executed as a function of the input size n.
This involves analysis loops, recursive calls, and any other constructs that affect the runtime

3 Express the Count as a function

Express the total no, of operations as a mathematical function of n, where n is the size of input

4. Simplify the function

Simplify the function by keeping only the dominant term, which is the term that grows the fastest as n increases. Discard constants and less significant terms because they have a negligible effect on the growth rate for large n

5 Use Big O Notation

Express the simplified function using Big O notation to describe the time complexity. Big O notation

Provides an upper bound on the growth rate, focusing on the dominant term

Constant $O(1)$

Index	Value
0	1
1	2
2	3
3	4
4	5
5	6

numbers[5]

here time is Constant

Logarithmic $O(\log n)$ → data half/Compare

1	2	3	4	5	6	7	8	9	10

Binary search key = 9 → Sorted

Element end → worst case Senario

1	2	3	4	5	6	7	8	9	10

X X ✓

>9

n

n/2 n/4 100 → 10

Linear $O(n)$

23	48	523	65	150	72	93

Search = 150

Quadratic $O(n^2)$

n

#	ID
1	123
2	765
3	999
4	500

n

#	ID
1	459
2	227
3	234
4	587

find the common element in these two arrays

$$n \times n = n^2$$

Cubic $O(n^3)$

#	Id
1	123
2	765
3	999
4	500

#	ID
1	643
2	344
3	123
4	456

#	ID
1	52
2	765
3	45
4	67

$$n \times n \times n = n^3$$

Exponential $O(2^n)$ → takes more time

1	2	3	4	5	6	7	8	9	10

All possible subset

Factorial $O(n!)$ → takes more time than Exponential

Generates all possible seating arrangements for a list of friends' IDs

1	2	3	4	5	6	7	8	9	10

1 2 3
2 3 1
1 3 2

Comparision of time Complexity