DAA

Unit 2
Analysis of an algorithm and Complexity Theory

Contents

- Space Complexity and Time Complexity
- Big Oh Notation / Omega Notation / Theta Notation
- P Class Problem & NP Class Problem
- NP Complete & NP Hard Class
- NP Complex Class
- Sat & 3 Sat Problem

SPACE COMPLEXITY

what is space complexity ?

- Definition
- Amount of memory required by an algorithm to execute and produce output.
- Components of Space Complexity

- Fixed Part: Independent of input (constants, program code, simple variables).
- Variable Part: Depends on input size (dynamic memory, recursion stack, arrays, etc.).

- Formula
- Space Complexity = Fixed Part + Variable Part

- Examples
- Iterative algorithms ↠ less memory (O(1))
- Recursive algorithms ↠ extra stack space (O(n)

TIME COMPLEXITY

what is time complexity ?

- Definition

- Time taken by an algorithm to execute as a function of input size.

- Why Important?

- Measures efficiency of an algorithm.
- Helps compare multiple algorithms for same problem.

- Types of Time Complexity Cases
- Best Case ↠ Minimum time taken.
- Average Case ↠ Expected time for random input.
- Worst Case ↠ Maximum time taken.

EXAMPLES

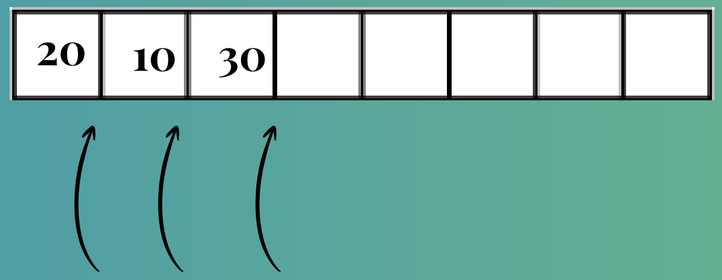
TIME COMPLEXITY

- **Linear Search O(n)**
- **Binary Search O(log n)**
- **Bubble Sort O(n^2)**

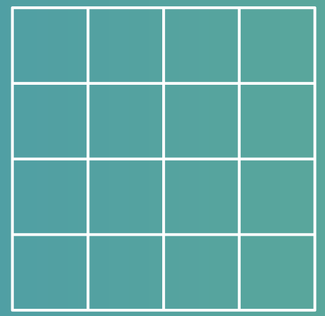
BIG O NOTATION

1. 

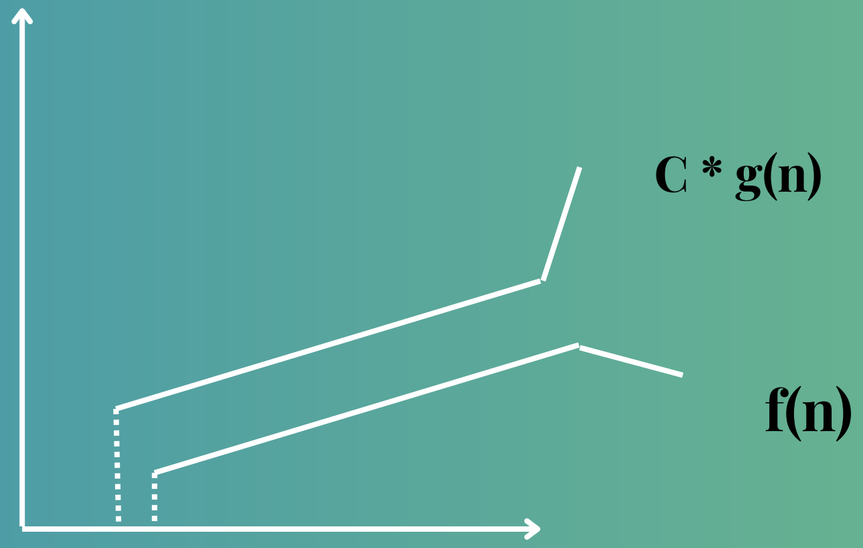
O(1) → Constant time

2. 

O(n) → Linear

3. 

O(n^2) → Quadratic



O(1) < O(n) < O(n^2)

- **Calculation & Observation**

- **f(n)=2n+2, g(n)=n^2**
1. **For n=1: f(1)=4, g(1)=? ↠ compare**
2. **For n=2: f(2)=6, g(2)=? ↠ compare**
3. **For n=3: f(3)=8, g(3)=9 ↠ here f(n)<g(n)f(n) < g(n)f(n)<g(n).**

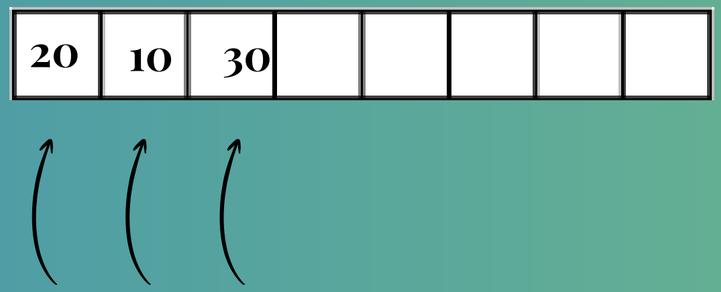
- **Conclusion**

From n>2, we get f(n)≤g(n).
Hence, f(n)=O(g(n)

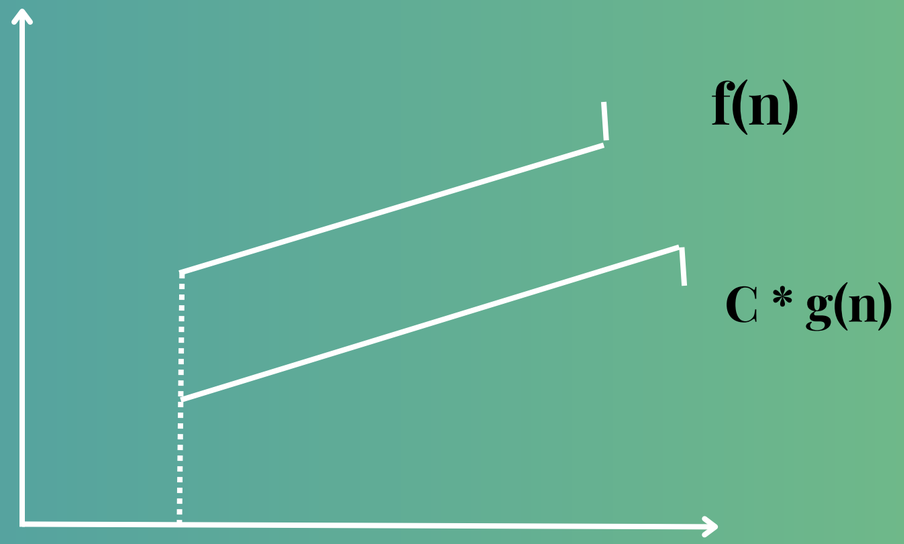
OMEGA NOTATION

1. 

Omega(1) → Constant time

2. 

Omega(n) → Linear



- **Calculation & Comparison**
- f(n) = 2n^2 + 5 , g(n)=7n

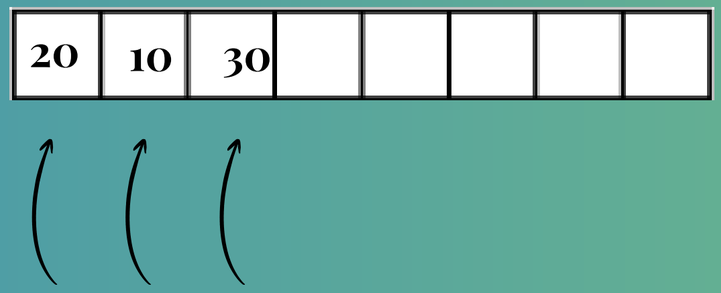
- For n=1: f(1)=7, g(1)=7 ⇾ f(n)=g(n)
- For n=2: f(2)=13, g(2)=14 ⇾ f(n)<g(n)
- For n=3: f(3)=2, g(3)=21 ⇾ f(n)>g(n)

- **Conclusion**
- For n>2 $f(n) \geq c \cdot g(n)$
- Hence, $f(n)=\Omega(g(n))$

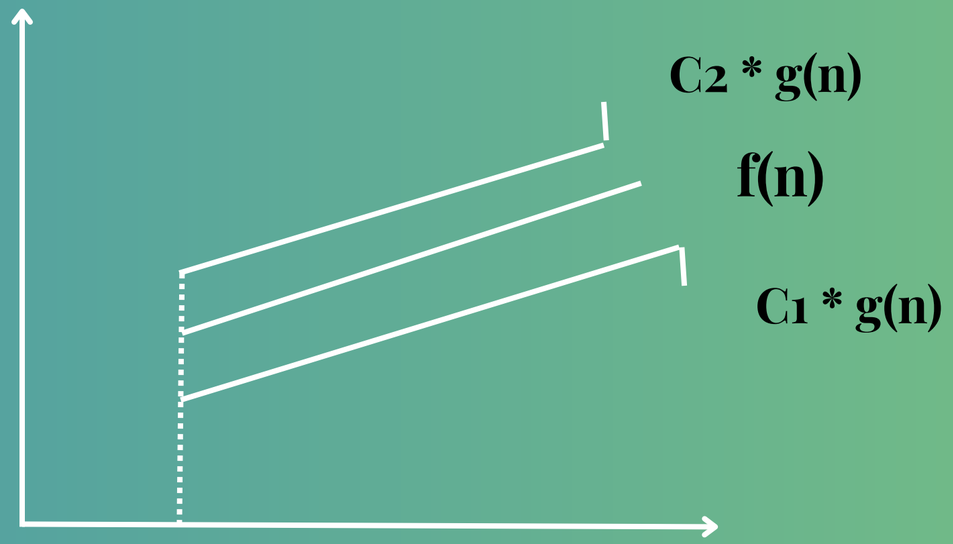
THETA NOTATION

1. 

Theta(1) → Constant time

2. 

Theta(n) → Linear



- **Calculation & Comparison**

- f(n)=2n+8, g(n)=5
- f(n)≥5n for n≥2n
- Also f(n)≤7nf(n) for n≥2n

- **Conclusion**

- Constants: c1=5, c2=7, n0=2
- Hence, f(n)=Θ(g(n))f(n)

P-CLASS PROBLEM

1. Definition

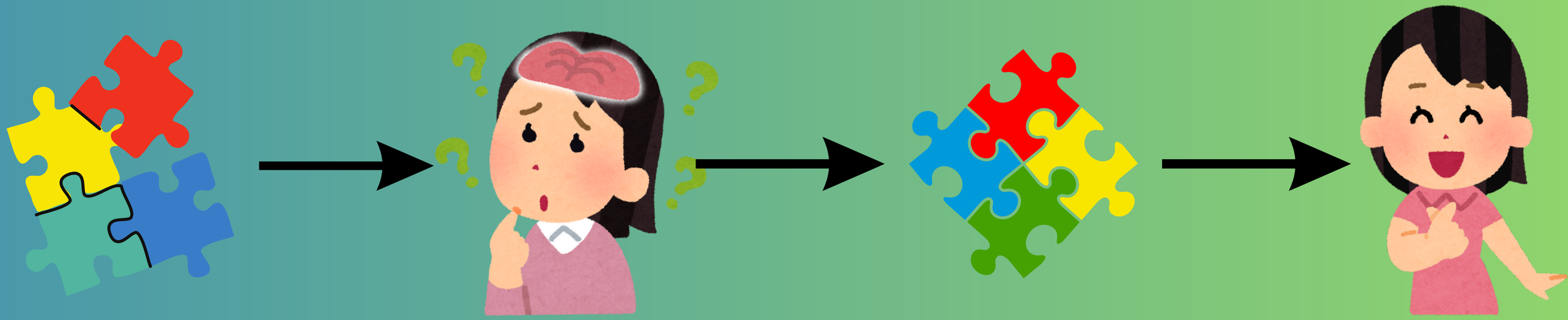
- P class = Set of decision problems that can be solved in polynomial time (e.g., O(n2),O(n3)O(n^2), O(n^3)O(n2),O(n3)).

- Polynomial time means the algorithm's running time grows as a polynomial function of the input size.

NP - CLASS PROBLEM

1. Definition

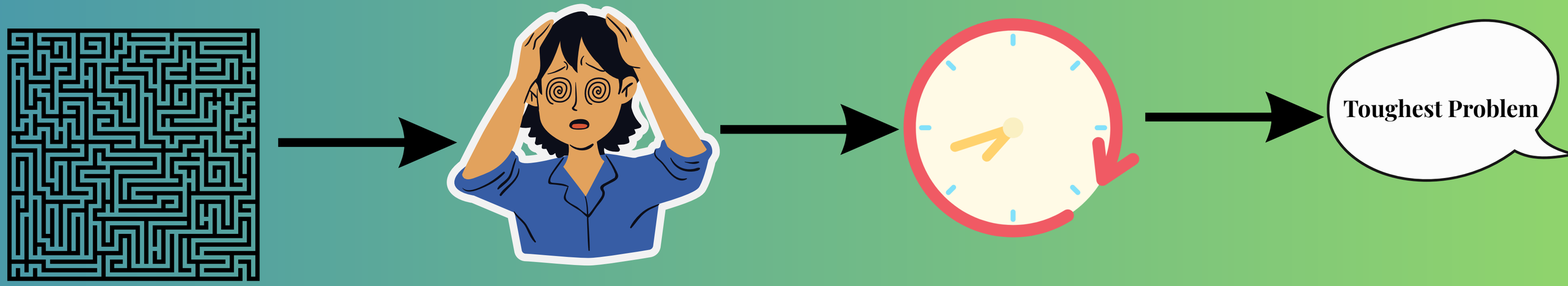
- NP class = Set of decision problems where a given solution can be verified in polynomial time.
- Even if finding the solution may be hard, checking it is easy.



NP - HARD PROBLEM

1. Definition

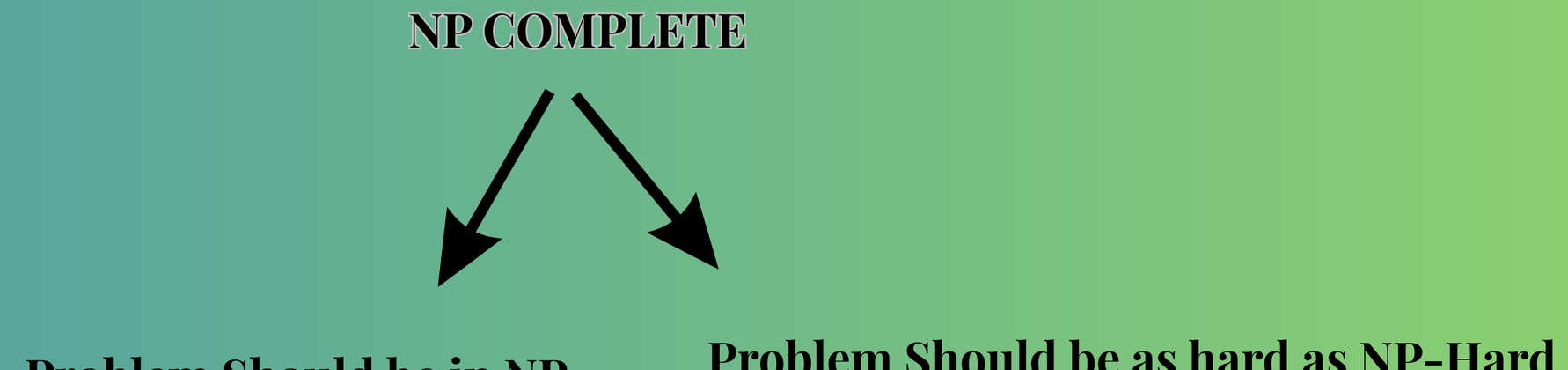
- NP-Hard = Problems that are at least as hard as NP problems.
- They don’t have to be in NP (solution may not be verifiable in polynomial time).

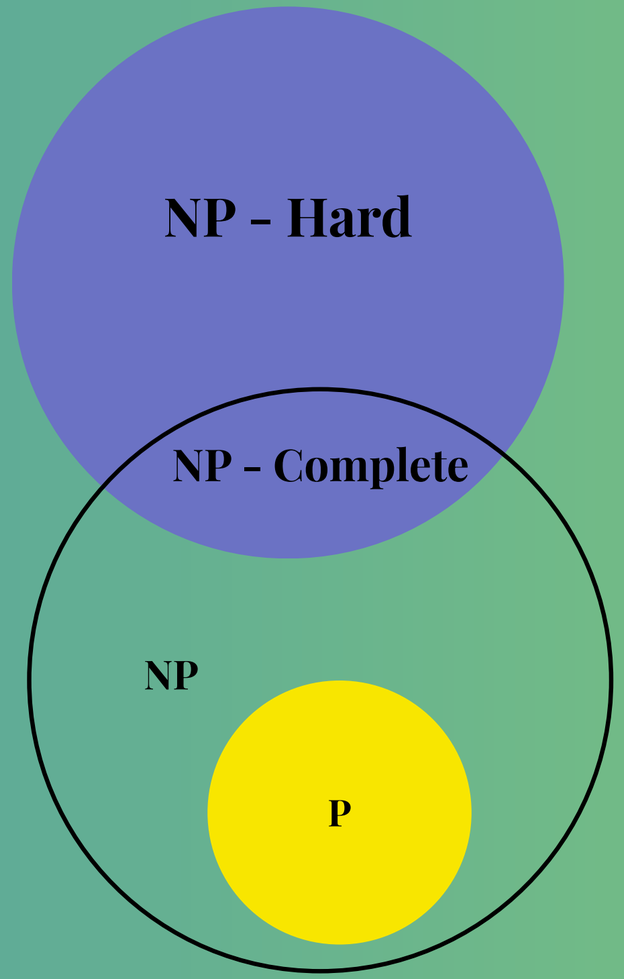


NP - COMPLETE PROBLEM

1. **Definition**

- **NP-Complete = Problems that are both in NP and NP-Hard.**
- **They can be verified in polynomial time, and every NP problem can be reduced to them.**




NP - Hard
NP - Complete
NP
P

SAT PROBLEM

1. Definition

- SAT (Boolean Satisfiability Problem): Given a Boolean formula, decide if there exists an assignment of variables (true/false) that makes the formula evaluate to true.
- First problem proven to be NP-Complete (Cook's theorem).

2. Examples & Importance

- Example: Formula (x∨y)∧(¬x∨z)(x ∨ y) ∧ (¬x ∨ z)(x∨y)∧(¬x∨z) ↠ satisfiable if x=false,y=true,z=truex = false, y = true, z = truex=false,y=true,z=true.
- SAT is a cornerstone of complexity theory; many real-world problems (circuit design, AI planning) can be reduced to SAT.

3 SAT PROBLEM

1. Definition

- 3-SAT = A special case of the SAT problem where each clause has exactly 3 literals (variables or their negations).
- It asks: Is there an assignment of variables that makes the whole formula true?

2. Examples & Importance

- Example: $(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w)(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w)(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w)$
- 3-SAT is one of the most famous NP-Complete problems; widely used to prove hardness of other problems.

Q4) a) What is SAT AND 3-SAT problem? Prove that 3-SAT problem is NP complete. **[8]**

- **STEP 1: 3-SAT ∈ NP**
- **STEP 2: SAT ≤P 3-SAT (POLYNOMIAL REDUCTION)**
- **STEP 3: THEREFORE, 3-SAT IS NP-COMPLETE**

c) Is $6n^3 = \Theta(n^2)$? Justify your answer. [3]

1. Definition & Comparison

- f(n)=6n3, g(n)=n^2
- Ratio: f(n)/g(n)=6n→∞

2. Conclusion

- 6n^3 grows faster than n2.
- Hence, **6n3≠O(n2) (but it is Θ(n3))**

Thank you