

# ALGORITHM COMPLEXITIES

Understanding How Things Scale in Everyday Life

CS 101 - Fall 2025

## Welcome to Algorithm Complexities!



### Today's Mission

Learn the **5 most important complexity levels** that describe how things scale in real life!



Tip

#### What we'll discover:

- $O(1)$  - The Magic Trick Level
- $O(\log n)$  - The Smart Detective Level
- $O(n)$  - The One-by-One Level
- $O(n^2)$  - The Handshake Problem Level
- $O(2^n)$  - The Explosion Level

Ready to become complexity detectives? Let's go!

## Complexity is All About How Things Scale

### Note

Complexity = How much **more work** do you need when you have **more stuff** to deal with?

#### Real-Life Examples:

- **Making dinner for friends:** 2 friends vs 20 friends - how much more work?
- **Finding a book:** In a small pile vs a huge library - how much longer?
- **Gift wrapping:** 5 gifts vs 50 gifts - how much more time?
- **Meeting everyone at a party:** 10 people vs 100 people - how many more handshakes?

## The Big Question

**! When you double the amount of “stuff,” what happens to the amount of work?**

- Does it stay the same?
- Double too?
- Get **way** worse?
- Or explode completely?

That's what complexity tells us!

Let's explore each complexity level!

## O(1) - The Magic Trick Level

 **“No Matter How Much, It Takes the Same Time!”**

**O(1) means:** Whether you have 1 thing or 1 million things, the task takes exactly the same amount of time!

### **i Everyday O(1) Examples:**

**Using a key to open your door** - Same one turn always!  
**Turning on a light switch** - Same flip always!  
**Checking the time on your phone** - Always instant!  
**Using your debit card** - Same swipe time always!

## **Why O(1) is Amazing**

### **💡 The Holy Grail of Algorithms!**

**It's like magic** - the amount of work **never changes**  
**Perfect performance** - always fast, always reliable  
**Every programmer dreams of O(1) solutions!**

### **i Real-World O(1) Examples:**

- Your phone's "Recent Calls" list
- Looking up a contact by name
- Checking account balance
- Skipping to specific song

## **O(log n) - The Smart Detective Level**

### **💡 "Cut the Problem in Half, Over and Over!"**

**O(log n) means:** Each step eliminates half of what's left to search. Super efficient even with huge amounts!

### **i Everyday O(log n) Examples:**

**Guessing a number 1-1000** - Cut problem in half each time, found in ~10 questions max!

**Finding word in dictionary** - Open to middle, go left or right, found in seconds!

**20 Questions game** - Each question eliminates half the possibilities

**Phone contact search** - Type "J" → cuts to J names, type "Jo" → even fewer options

## Why $O(\log n)$ is Amazing

### 💡 Incredible Scaling Performance!

**Amazing scaling:** \* 1,000 items → ~10 steps \* 1,000,000 items → ~20 steps

\* 1,000,000,000 items → ~30 steps

**Smart strategy beats brute force**

### ℹ️ Used everywhere:

- Google searches
- GPS route finding
- Phone contact search

## But What's the Catch?



### ⚠️ The catch: You need things organized first!

## **O(n) - The One-by-One Level**

**i** “Check Every Single Thing, One by One”

**O(n) means:** Double the stuff = Double the work. Fair and predictable!

**i Everyday O(n) Examples:**

**Reading every page in a book** - 100 pages = 100 page flips, 200 pages = 200 page flips

**Counting items in shopping cart** - Must touch each item once, 10 items = 10 counts

**Listening to playlist** - 50 songs =  $50 \times$  the time

**Grading test stack** - 30 tests =  $30 \times$  the work

## **Why O(n) is Pretty Good**

**i Predictable and Fair!**

**Predictable and fair** - work scales linearly

**Often the best you can do** when you need to check everything

**Reasonable for most tasks:** \* Finding highest grade \* Adding up expenses \* Reading all text messages

**⚠ When it gets slow:**

Really large amounts of data - but still very manageable for normal use!

## **O(n<sup>2</sup>) - The Handshake Problem Level**

**⚠ “Everyone Must Meet Everyone Else!”**

**O(n<sup>2</sup>) means:** When you double the people, you get **four times** the work! This gets crazy fast.

**i Everyday  $O(n^2)$  Examples:**

**Party introductions** - 4 people = 6 handshakes, 8 people = 28 handshakes, 16 people = 120 handshakes!

**Sports tournament** - Everyone plays everyone, gets expensive fast!

**Group photo arrangements** - Every person next to every other, gets overwhelming quickly!

**Comparing all student tests** - Looking for identical answers, 30 students = 435 comparisons!

## Why $O(n^2)$ Gets Scary

**⚠ Explosive Growth!**

**Explosive growth:** \* 10 things → 100 operations \* 100 things → 10,000 operations \* 1,000 things → 1,000,000 operations!

**The danger zone** - where apps become unusably slow

**i Common culprits:** \* Comparing every item to every other \* Nested loops in programming \* Poor algorithm choices

**❗ When to worry:** Anything over ~1,000 items gets really slow!

## $O(2)$ - The Explosion Level

**❗ “Every Choice Doubles Your Problems!”**

**$O(2)$  means:** Add just one more thing, and you **double** all the work! This explodes instantly.

**i Everyday O(2 ) Examples:**

**Family tree exploration** - 2 parents → 4 grandparents → 8 great-grandparents  
→ 16 great-great-grandparents

**Password cracking** - Each digit doubles possibilities, 10-digit PIN = 1+ billion combos!

**Gift wrapping combinations** - Each gift: wrapped or not, 20 gifts = 1+ million combinations!

## Why O(2 ) is Terrifying

**! Grows Impossibly Fast!**

**Grows impossibly fast:** \* 10 things → 1,024 operations \* 20 things → 1,048,576 operations

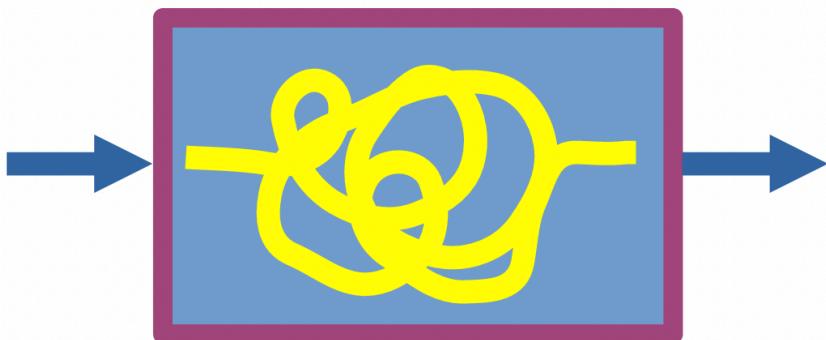
\* 30 things → 1,073,741,824 operations!

**Usually unusable** for anything but *tiny* problems

**i Real-world impact:** \* Why cryptography works (good!) \* Why some problems are “impossible” (bad!)

**⚠ Bottom line:** Avoid at all costs unless you have < 20 items!

## The Complexity Race!



## 💡 How They Compare With 1,000 Items

Let's see what happens when we have 1,000 things to process:

### The Complexity Race Table

Complexity	Name	Steps Needed	Real-World Feeling
$O(1)$	Magic Trick	1 step	Instant!
$O(\log n)$	Smart Detective	~10 steps	Super fast!
$O(n)$	One-by-One	1,000 steps	Takes a moment
$O(n^2)$	Handshake Problem	1,000,000 steps	Ugh, so slow...
$O(2^n)$	Explosion	$2^n$ steps	Heat death of universe

### The Big Takeaway

❗ Small differences in complexity = HUGE differences in real-world performance!

This is why choosing the right approach matters so much in programming!

Ready for your challenge? Let's become complexity detectives!

### Your Turn: Complexity Detectives!

💡 Now You're Ready for the Challenge!

You've learned the 5 complexity levels. Time to become **complexity detectives** and find examples from your own life!

## The “Build a Better Algorithm” Challenge



Tip

### Your Mission:

1. **Brainstorm** real-life situations that match each complexity level
2. **Work in teams** to find creative examples
3. **Think about** when you'd choose one approach over another
4. **Share** your discoveries with the class!

## Remember the Levels!



### The 5 Complexity Levels:

- **$O(1)$**  - Magic Trick (always same time)
- **$O(\log n)$**  - Smart Detective (cut in half)
- **$O(n)$**  - One-by-One (check everything)
- **$O(n^2)$**  - Handshake Problem (everyone meets)
- **$O(2^n)$**  - Explosion (choices double work)

## Questions for Detective Work!



### Questions to Ask Yourself:

- What happens when I double the input?
- Do I compare everything to everything?
- Can I organize data for faster searching?
- Am I exploring all combinations?

**Pro Tip:** Look for efficient patterns in your daily life!

Let's see what amazing complexity examples you can find!