

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Big-O Made Simple: How Work Grows When Data Grows

Lecture 4 · Friday 9 October 2026 · Canning, Broder & Lafore, Chapter 2 (Big O Notation)

Dr. Innocent Nyalala | MTech Semester 1 | IIT Madras Zanzibar

Today's Plan

IIT Madras Zanzibar

1. (0–5 min) Recap: counting steps
2. (5–18 min) Four examples, four growth shapes: with tables and charts
3. (18–27 min) Reading Big-O notation, symbol by symbol
4. (27–39 min) Two simple rules, finding Big-O from code, best and worst case
5. (39–48 min) Exercise, solution and quick quiz
6. (48–50 min) Summary

Learning Outcomes

IIT Madras Zanzibar

By the end of this session, you will be able to:

Count	the steps of a simple algorithm for small inputs, using a table
Recognise	four growth shapes: constant, logarithmic, linear, quadratic
Read	Big-O notation aloud and say what it means in plain words
Find	the Big-O of a simple loop or a loop inside a loop

Recap: Yesterday We Counted Steps

IIT Madras Zanzibar

```
total = 0
for m in marks:
    total = total + m
```

n = number of marks	additions
3	3
100	100
1,000,000	1,000,000

1 n

the letter we use for 'how many items'

2 Count steps, not seconds

the same answer on every computer

3 The real question

if the data grows, how much more work?

? **Think:** if the class doubles from 15 to 30 students, what happens to the number of additions?

Big-O Made Simple

IIT Madras Zanzibar

→ **Counting Steps**

Four Growth Shapes

Reading Big-O Notation

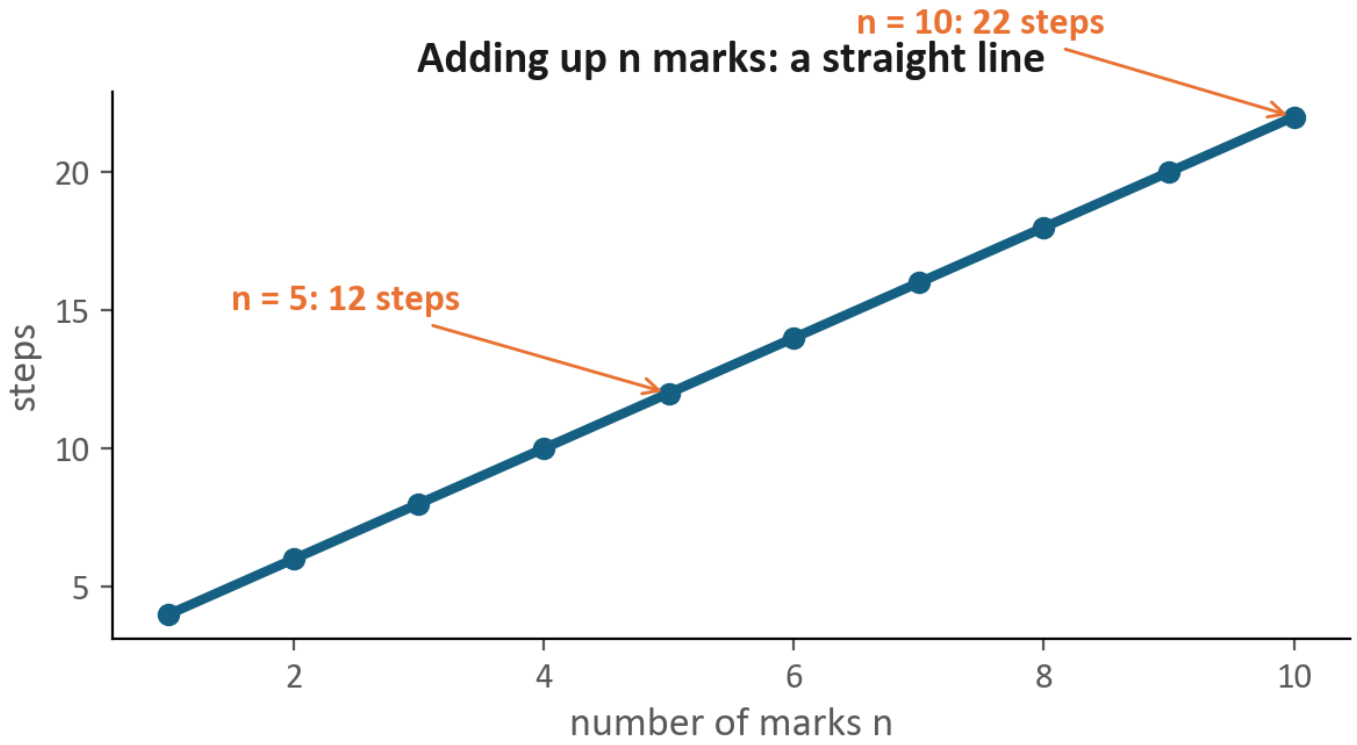
Big-O From Code

In-Class Exercise

Example 1: Adding Marks. Double the Data, Double the Work

IIT Madras Zanzibar

n	steps (2n + 2)
1	4
5	12
10	22
100	202
1,000	2,002

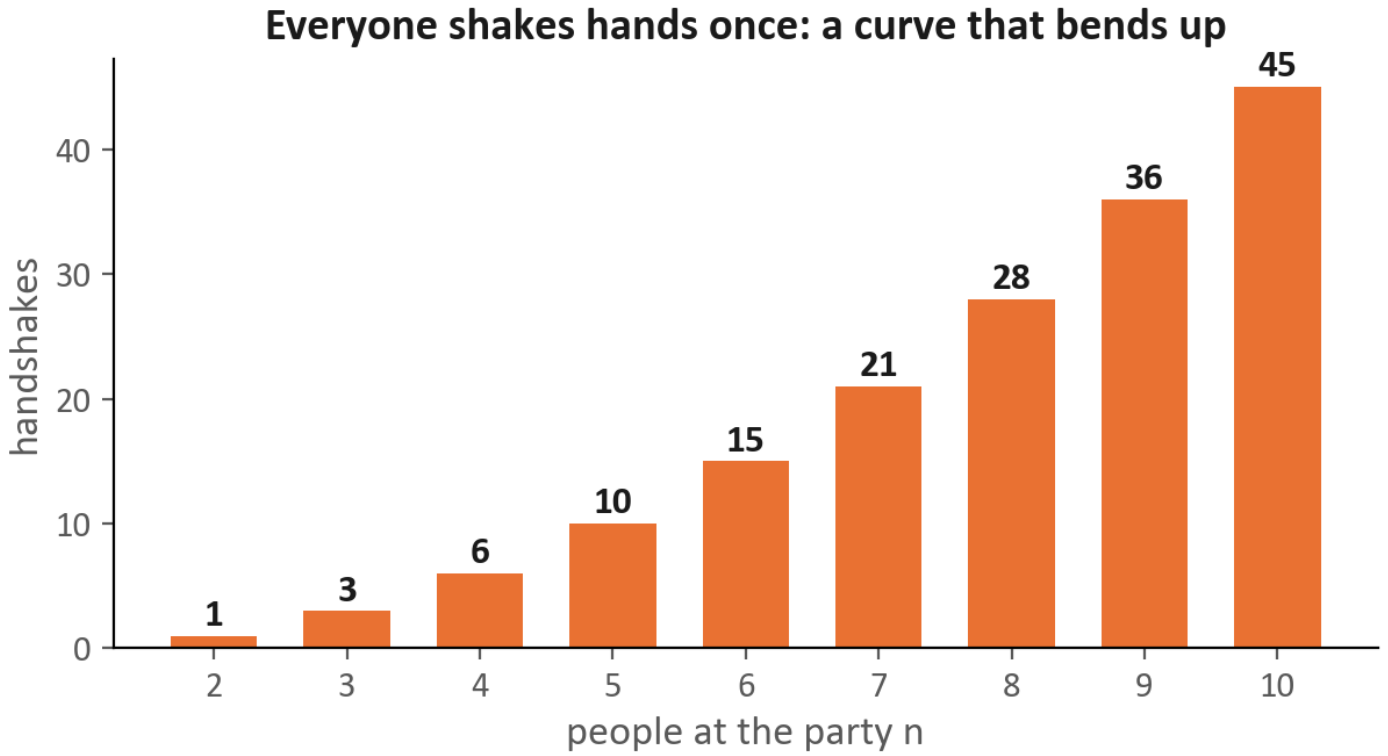


Linear: a straight line. Double the data → double the work.

Example 2: Handshakes. Double the People, 4× the Work

IIT Madras Zanzibar

people n	handshakes
2	1
5	10
10	45
20	190
100	4,950



Quadratic: a curve bending up. Double the data → about FOUR times the work.

Example 3: Halving. Double the Data, ONE More Step

IIT Madras Zanzibar

Halving 16 until we reach 1



4 halvings, so $\log_2 16 = 4$

n	halvings = $\log_2 n$
8	3
16	4
1,024	10
1,000,000	about 20

log n means: how many times can you halve n before you reach 1?

Logarithmic: double the data → just ONE more step. Very slow growth: excellent.

Example 4: Look at the First Item. The Size Doesn't Matter

IIT Madras Zanzibar

5 marks:



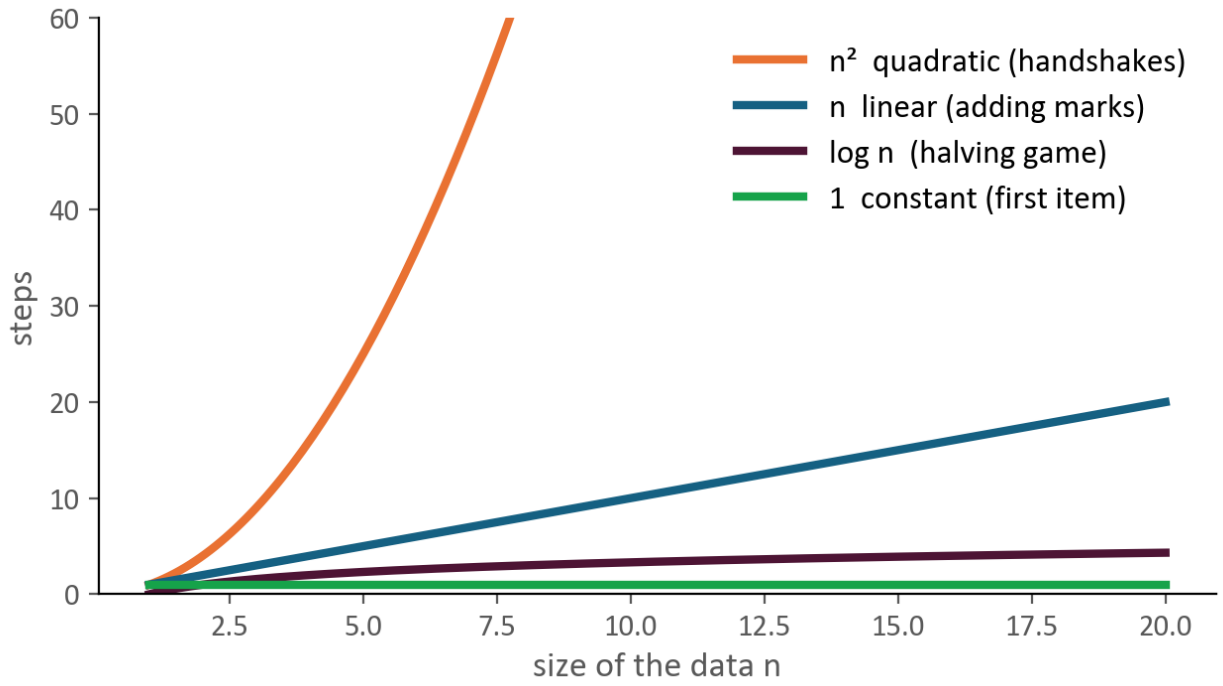
12 marks:



marks[0] reads the orange box: 1 step, whatever the size

Constant, O(1): the work does not grow at all. Reading one item by its index is constant.

The Four Shapes Side by Side



n	1	log n	n	n^2
10	1	3	10	100
1,000	1	10	1,000	1,000,000
1,000,000	1	20	1,000,000	10^{12}

At 10 million steps a second: n^2 for a million items takes over a day.

The lesson: the SHAPE of growth matters far more than the speed of your computer.

Big-O Made Simple

IIT Madras Zanzibar

Counting Steps

Four Growth Shapes

→ **Reading Big-O Notation**

Big-O From Code

In-Class Exercise

Reading Big-O Notation, Symbol by Symbol

IIT Madras Zanzibar

O 'order of' = 'grows like'

( ) holds the growth shape

n the size of the data

Write	Say	Name	Everyday example	Data doubles → work
O(1)	big O of one	constant	read the first mark	stays the same
O(log n)	big O of log n	logarithmic	guessing game, halving	one more step
O(n)	big O of n	linear	adding all the marks	doubles
O(n^2)	big O of n squared	quadratic	everyone shakes hands	about four times

Read O(n) as: "the work grows like n."

Rule 1: Keep Only the Biggest Part

IIT Madras Zanzibar

n	n^2 (big part)	n (small part)	total	big part's share
10	100	10	110	91%
100	10,000	100	10,100	99%
1,000	1,000,000	1,000	1,001,000	99.9%

$$n^2 + n \rightarrow \text{keep the biggest part} \rightarrow O(n^2)$$

? **Think:** which part is biggest in $5n + 3$? In $n^2 + 100n$?

Rule 2: Ignore the Number in Front

$2n + 2$	rule 1: drop + 2, rule 2: drop the 2	$O(n)$
$n^2/2 - n/2$	rule 1: keep $n^2/2$, rule 2: drop the 1/2	$O(n^2)$
5	no n at all: the work never grows	$O(1)$

Why we may ignore the 2: 2n and n both DOUBLE when the data doubles. Same shape.

? **Think:** simplify $7n + 20$ and $3n^2 + 5n + 1$ using the two rules.

Big-O Made Simple

IIT Madras Zanzibar

Counting Steps

Four Growth Shapes

Reading Big-O Notation

→ **Big-O From Code**

In-Class Exercise

Finding Big-O From Code: Three Patterns

IIT Madras Zanzibar

One loop

```
for m in marks:
    total += m
```

O(n)

Loop inside a loop

```
for a in people:
    for b in people:
        shake(a, b)
```

O(n²)

Halving

```
while n > 1:
    n = n // 2
```

O(log n)

No loop

```
first = marks[0]
```

O(1)

Ask three questions: is there a loop? A loop inside a loop? Does something get halved?

? **Think:** two loops ONE AFTER the other (not inside): n + n steps. Which Big-O is that?

Try It: Count the Steps, Then Name the Shape

IIT Madras Zanzibar

Loop A: one loop

```
count = 0
for i in range(n):
    count = count + 1
```

Loop B: a loop inside a loop

```
count = 0
for i in range(n):
    for j in range(n):
        count = count + 1
```

n	Loop A: steps counted	Loop B: steps counted
10	10	100
100	?	?
1,000	?	?

? **Think:** fill in the blanks with your neighbour. When n grows 10 times, how many times more work does each loop do?

Best Case and Worst Case: Plan for the Unlucky Day

IIT Madras Zanzibar

Asha	Baraka	Juma	Neema	Zawadi

Best: target is first, 1 step

Average: about n/2 steps

Worst: target is last, n steps

1 Best case

The luckiest input: the target is first. 1 step.

2 Average case

A typical input: the target sits mid-list. About n/2 steps, still O(n).

3 Worst case

The unluckiest input: the target is last or missing. n steps.

4 Our default

Big-O means the WORST case unless we say otherwise.

How we measure: fix the size n, then count the steps for the luckiest and the unluckiest input of that size.

? **Think:** why would a bank care more about the worst case than the best case?

Quick Quiz: Name the Big-O

IIT Madras Zanzibar

Snippet 1

```
print(marks[0])
print(marks[1])
```

Big-O?

Snippet 2

```
for m in marks:
    print(m)
for m in marks:
    print(m)
```

Big-O?

Snippet 3

```
for m in marks:
    if m == target:
        return True
```

Best case? Worst case?

? **Think:** write down all three answers, then check with your neighbour. Snippet 3 has two answers.

Quick Quiz: Answers

IIT Madras Zanzibar

Snippet 1

```
print(marks[0])
print(marks[1])
```

O(1)

Snippet 2

```
for m in marks:
    print(m)
for m in marks:
    print(m)
```

O(n)

Snippet 3

```
for m in marks:
    if m == target:
        return True
```

Best: O(1)
Worst: O(n)

! **Why:** snippet 1 is two single steps. Snippet 2 is n + n = 2n steps, so drop the 2. Snippet 3 stops at once if the target is first but checks all n marks if it is last or missing.

In-Class Exercise (7 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: write your answers, then compare with your neighbor

1. **Handshakes.** At a party, every pair of people shakes hands once. How many handshakes with 6 people? With 12 people? About how many times more is that?

2. **Simplify.** Write each as a Big-O. Rule 1: keep the biggest part. Rule 2: drop the number in front.
   (a) $4n + 9$ (b) $2n^2 + 3n + 1$ (c) 50

3. **Name the Big-O** of each piece of code:
   (a) one loop over all n marks (b) a loop inside a loop, both over all n marks (c) a loop that halves n until it reaches 1 (d) reading one mark, marks[3]

4. **Predict the time.** A program takes 2 seconds for 1,000 items. How long will it take for 2,000 items if it is O(n)? If it is $O(n^2)$? If it is O(1)?

Solution

IIT Madras Zanzibar

1 **1: 15 and 66, about 4 times more**

6 × 5 / 2 = 15; 12 × 11 / 2 = 66. Double the people → about four times the handshakes: quadratic.

2 **2: O(n), O(n²), O(1)**

keep the biggest part, drop the number in front. A plain number never grows.

3 **3: O(n), O(n²), O(log n), O(1)**

one loop; loop inside a loop; halving; a single step.

4 **4: 4 s, 8 s, 2 s**

linear doubles; quadratic is about four times; constant stays the same.

Formal definition: the precise mathematical definition of Big-O comes in Week 5, once these shapes feel natural.

Summary

IIT Madras Zanzibar

Count steps

how the work grows with n, the size of the data

Four shapes

O(1), O(log n), O(n), O(n^2): constant, halving, linear, handshakes

Two rules

keep the biggest part; ignore the number in front

From code

one loop O(n); loop in loop O(n^2); halving O(log n); worst case

Next

Mon 12 Oct: Week 2 begins: the array, our first data structure

Thu 15 Oct, 2:00 pm: Lab 1: build and use your own array

Practise: redo today's exercise; read Canning, Broder & Lafore Ch 2 to 'Big O Notation'

References and Further Reading

1. Canning, J., Broder, A. & Lafore, R. (2022). Data Structures & Algorithms in Python. Addison-Wesley. Chapter 2, Big O Notation.
2. Big-O Cheat Sheet: bigocheatsheet.com
3. VisuAlgo (watch searching and sorting step by step): visualgo.net