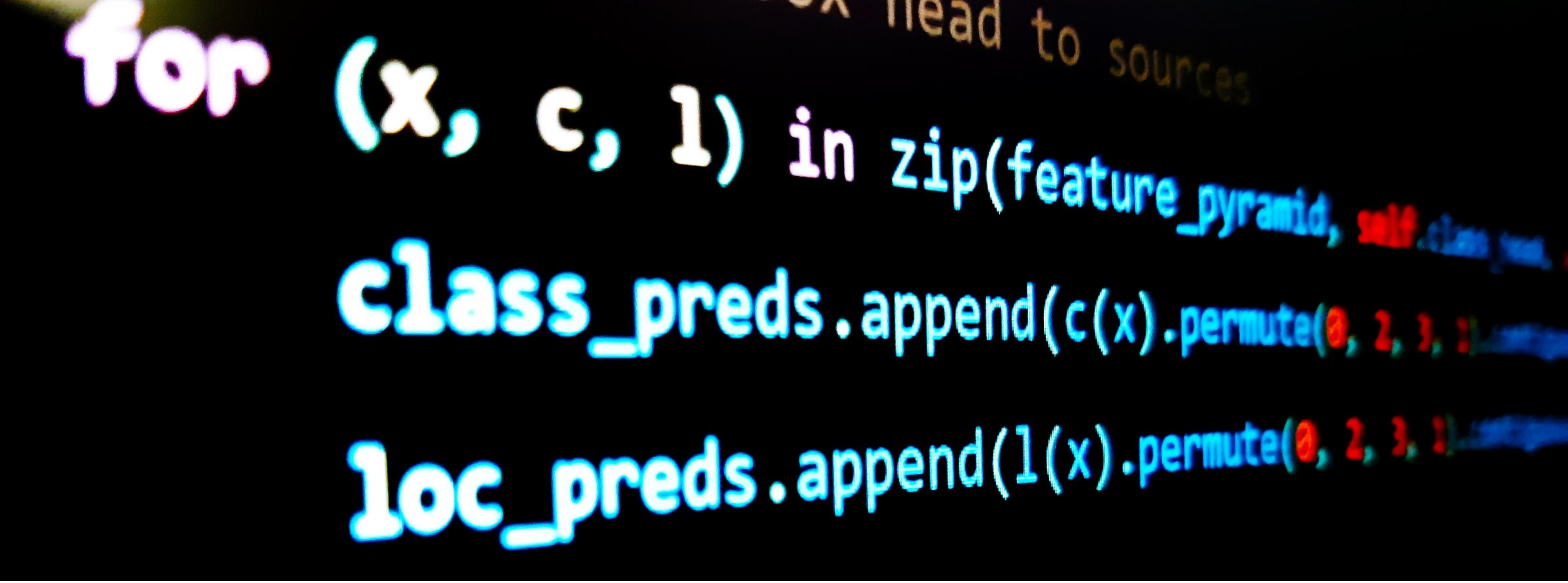
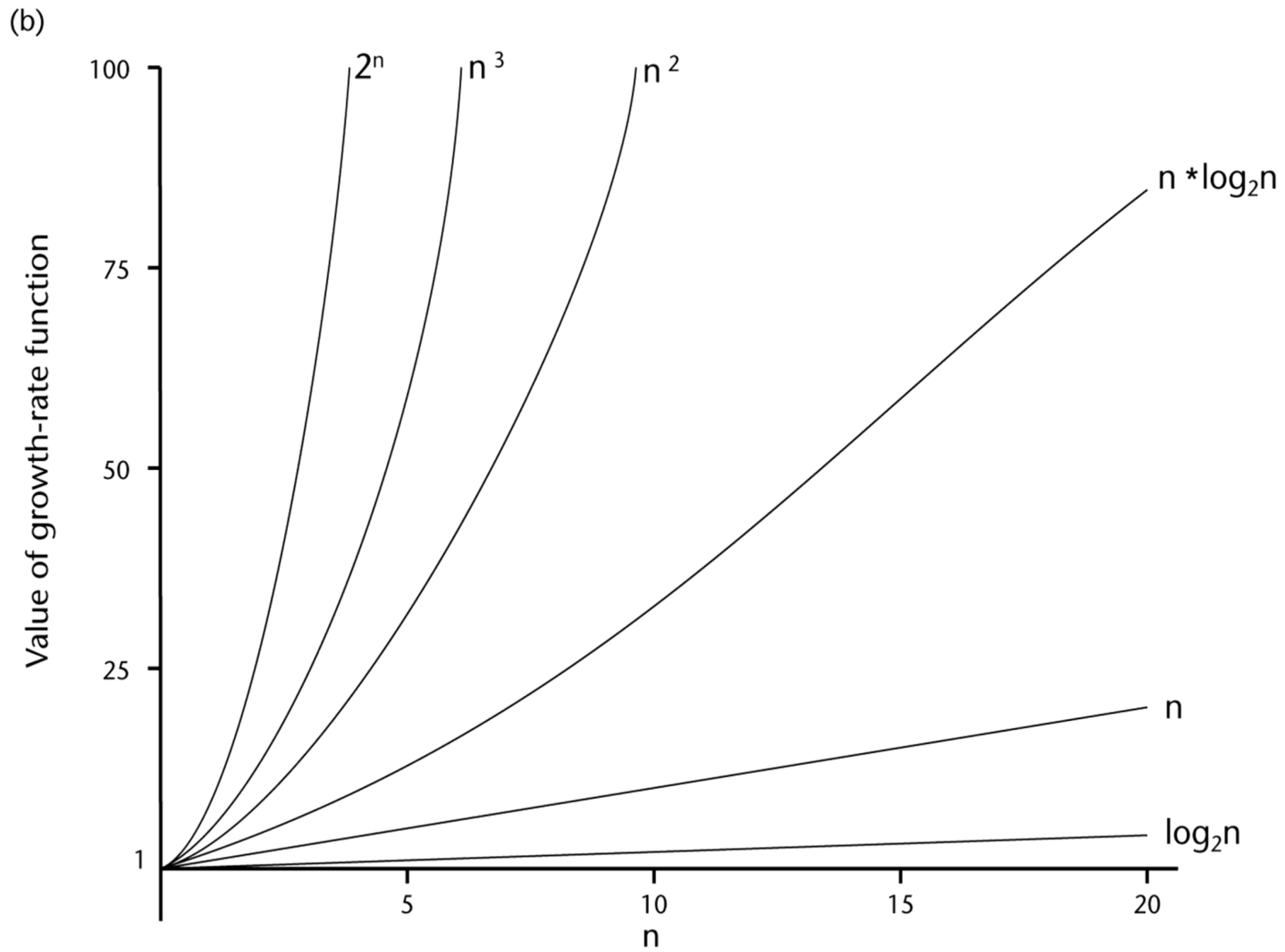


ASYMPTOTIC BOUNDS

CCDSALG T2 AY 2020-2021

RATE OF GROWTH



RATE OF GROWTH

$$T(n) = 60n^2 + 5n + 1$$

n	$T(n)$	$60n^2$
10	6,051	6,000
100	60,501	60,000
1,000	60,005,001	60,000,000
10,000	6,000,050,001	6,000,000,000

- $T(n)$ grows like $60n^2$
- If $T(n)$ is measured in seconds, then in minutes: $n^2 + \frac{5n}{60} + \frac{1}{60}$

RATE OF GROWTH

Observations:

- The **dominant term** (term with the fastest growth rate) in the function determines the behavior of the algorithm.
  - $3n^3 + 2n^2 + 1$ $\qquad$ $3n^3 > 2n^2 > 1$
  - $2^n + 3n^3 + 5$ $\qquad$ $2^n > 3n^3 > 5$

- Any **exponential function** of n dominates any polynomial function of n
  - $2^n + 2n^2 + 1$ $\qquad$ $2^n > n^2$

RATE OF GROWTH

Observations:

- A **polynomial degree** k dominates a polynomial of degree m iff $k > m$
  - $n^{k>7} > \cdots > n^7 > n^6 > \cdots > n^3 > n^2$

- Any **polynomial function** of n dominates any logarithmic function of n
  - $n^{k>3} > n^3 > n \log_2 n > \log_2 n$

RATE OF GROWTH

Observations:

- Any **logarithmic function** of n dominates a constant term.
  - $\log_2 n > c$

RATE OF GROWTH

Observations:

- The **order of growth** is a function of the dominant term of the running time.
- The **dominant term** contributes the most significant increase in $g(n)$ as n increases.
- The **coefficient of the dominant term** is ignored.
  - $3n^3 + 2n^2 + 1$ $\qquad n^3$

RATE OF GROWTH

What is the growth rate corresponding to the following running time?

1. $3n + 5n - 2$
2. $6n^2 + 7n + 3$
3. $9n^3 + 6n^2 + n + 2$

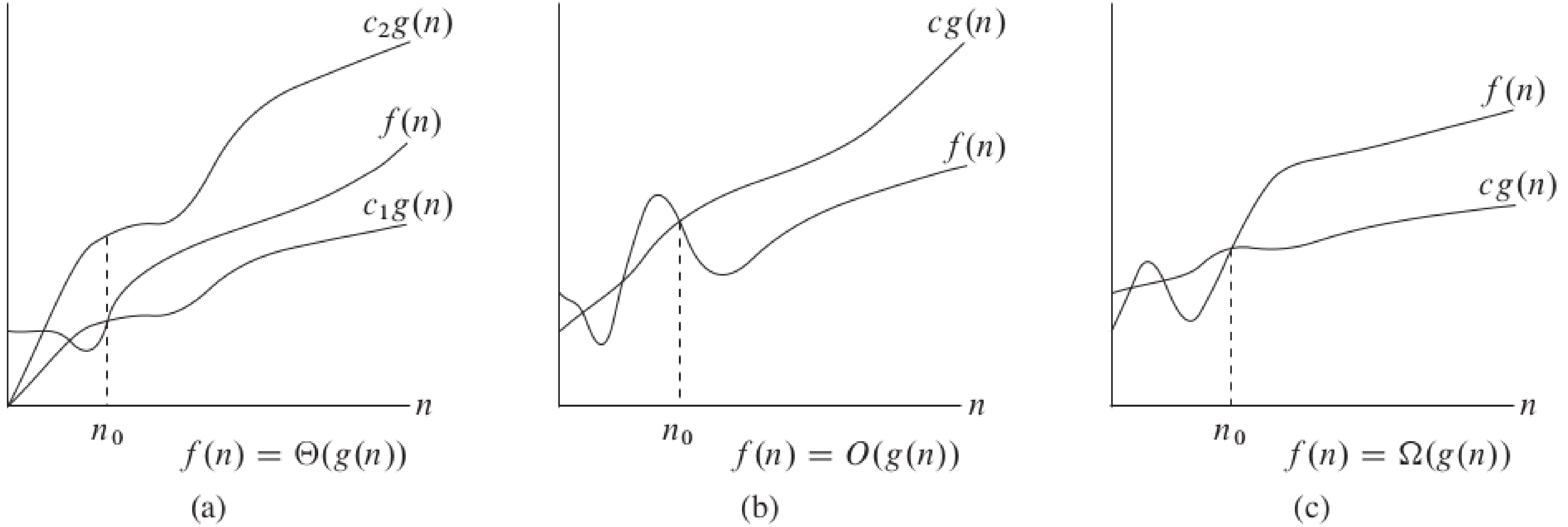
ASYMPTOTIC BOUNDS

- It is hard to get the exact running-time of an algorithm. Why?
- **Asymptotic bounds** are then used instead to describe the complexity of the algorithms
- **Asymptotic bounds** describes only the growth rates of the algorithm as the input size approaches infinity and ignoring most of the small inputs and constant factors.
- Among these bounds are: **Big-Oh, Big-Omega, and Big-Theta**

ASYMPTOTIC BOUNDS

- The **worst-case complexity** of the algorithm is the function defined by the **maximum** number of steps taken in any instance of size n.
- The **best-case complexity** of the algorithm is the function defined by the **minimum** number of steps taken in any instance of size n.
- The **average-case complexity** of the algorithm is the function defined by the **average** number of steps over all instances of size n.

ASYMPTOTIC BOUNDS



ASYMPTOTIC BOUNDS

ASYMPTOTIC BOUNDS

BIG-OH

- The **Big-Oh** of a function $f(n)$ is $O(g(n))$, iff there exist positive numbers c and n_0 such that:

$$0 \leq f(n) \leq c\, g(n) \text{ where } n \geq n_0$$

- Describes an **asymptotically loose upper-bound** of the algorithm
- Represents the **worst-case running time** of the algorithm

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

- $f(n) = 2n^2 + 3$
- The Big-Oh of a function $f(n)$ is $O(g(n))$, iff there exist positive numbers c and n_0 such that:

$$0 \leq f(n) \leq c\, g(n) \text{ where } n \geq n_0$$

- Suppose $g(n) = n^2$, then

$$0 \leq 2n^2 + 3 \leq c\, n^2 \text{ where } n \geq n_0$$

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

$$0 \leq 2n^2 + 3 \leq c\, n^2 \text{ where } n \geq n_0$$

- Suppose $n_0 = 1$ and $n \geq n_0$ then:
  1. $n \geq 1$
  2. $n^2 \geq n$ Multiply Eq. 1 by n
  3. $n^2 \geq n \geq 1$ Combine Eq. 1 and Eq. 2
  4. $2n^2 \geq 2n \geq 2$ Multiply Eq. 3 by 2
  5. $3n^2 \geq 3n \geq 3$ Multiply Eq. 3 by 3

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

$$0 \leq 2n^2 + 3 \leq c\, n^2 \text{ where } n \geq n_0$$

- Since $2n^2 \geq 2n^2$ and $3n^2 \geq 3n \geq 3$ (Eq. 5), then we can form the inequality:

$$0 \leq 2n^2 + 3 \leq 2n^2 + 3n^2$$
$$0 \leq 2n^2 + 3 \leq 5n^2$$

- The condition is satisfied for $c = 5, n_0 = 1, n \geq n_0$

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

- $f(n) = 5n^3 + 3n^2 + 4$
- The Big-Oh of a function $f(n)$ is $O(g(n))$, iff there exist positive numbers c and n_0 such that:

$$0 \leq f(n) \leq c\, g(n) \text{ where } n \geq n_0$$

- Suppose $g(n) = n^3$, then

$$0 \leq 5n^3 + 3n^2 + 4 \leq c\, n^3 \text{ where } n \geq n_0$$

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

$$0 \leq 5n^3 + 3n^2 + 4 \leq c\, n^3 \text{ where } n \geq n_0$$

- Suppose $n_0 = 1$ and $n \geq n_0$ then:

1. $n \geq 1$	
2. $n^2 \geq n$	Multiply Eq. 1 by n
3. $n^3 \geq n^2$	Multiply Eq. 2 by n
4. $n^3 \geq n^2 \geq n \geq 1$	Combine Eq. 1, Eq. 2, Eq. 3
5. $5n^3 \geq 5n^2 \geq 5n \geq 5$	Multiply Eq. 4 by 5
6. $3n^3 \geq 3n^2 \geq 3n \geq 3$	Multiply Eq. 4 by 3
7. $4n^3 \geq 4n^2 \geq 4n \geq 4$	Multiply Eq. 4 by 4

ASYMPTOTIC BOUNDS

BIG-OH EXAMPLE

$$0 \leq 5n^3 + 3n^2 + 4 \leq c\,n^3 \text{ where } n \geq n_0$$

- Since $5n^3 \geq 5n^3$ (Eq. 5), $3n^3 \geq 3n^2$ (Eq. 6), $4n^3 \geq 4n^2 \geq 4n \geq 4$ (Eq. 7), then we can form the inequality:

$$0 \leq 5n^3 + 3n^2 + 4 \leq 5n^3 + 3n^3 + 4n^3$$
$$0 \leq 5n^3 + 3n^2 + 4 \leq 12n^3$$

- The condition is satisfied for $c = 12, n_0 = 1, n \geq n_0$

Asymptotic Bounds: Guidelines

1) Assume that $n_0 = 1$ and $n \geq n_0$
2) Multiply both sides of equation by some power of n until you get the dominant term
3) Combine all previous equations into one
4) Multiply the equation to the coefficients needed. Separate the equations per coefficient.
5) Combine the equations to form $f(n)$ and $\mathrm{g}(n)$

ASYMPTOTIC BOUNDS

Properties of Growth-Rate Functions:

- We can ignore low-order terms in an algorithm's growth-rate function
- If an algorithm is $O(n^3 + 4n^2 + 3n)$, it is also $O(n^3)$.

ASYMPTOTIC BOUNDS

Properties of Growth-Rate Functions:

- We can ignore a multiplicative constant in the higher-order term of an algorithm's growth rate function.
- If an algorithm is $O(5n^3)$, it is also $O(n^3)$.

ASYMPTOTIC BOUNDS

Properties of Growth-Rate Functions:

- We can combine growth-rate functions:

$$O(f(n)) + O(g(n)) = O(f(n) + g(n))$$

- If an algorithm is $O(n^3) + O(4n^2)$, it is also $O(n^3 + 4n^2)$, thus, $O(n^3)$.
- Similar rules hold for multiplication.

ASYMPTOTIC BOUNDS

BIG-OMEGA

- The **Big-Omega** of a function $f(n)$ is $\Omega(g(n))$, iff there exists a positive number c and n_0 such that:

$$0 \leq c\ g(n) \leq f(n) \text{ where } n\ \geq\ n_0$$

- Describes an **asymptotically loose lower-bound** of the algorithm
- Represents the **best-case running time** of the algorithm

ASYMPTOTIC BOUNDS

BIG-THETA

- The **Big-Theta** of a function $f(n)$ is $\theta(g(n))$, iff there exists a positive number c_1, c_2 and n_0 such that:

$$c_1\, g(n) \leq f(n) \leq c_2\, g(n) \text{where } n > \ n_0$$

- Describes an **asymptotically tight bound** of the algorithm
- Represents the **average-case running time** of the algorithm

ASYMPTOTIC BOUNDS

GROWTH-RATE FUNCTIONS

$\boldsymbol{O(1)}$	**Constant** – time requirement is independent of the problem's size.
$\boldsymbol{O(\log_2 n)}$	**Logarithmic** – time requirement increases slowly as the problem increases.
$\boldsymbol{O(n)}$	**Linear** – time requirement increases directly with the size of the problem.
$\boldsymbol{O(n \log_2 n)}$	**Linear-logarithmic** – time requirement increases more rapidly than linear.

ASYMPTOTIC BOUNDS

GROWTH-RATE FUNCTIONS

$\boldsymbol{O(n^2)}$ **Quadratic** – time requirement increases rapidly with the size of the problem.

$\boldsymbol{O(n^3)}$ **Cubic** – time requirement increases more rapidly with the size of the problem than the time requirement for quadratic.

$\boldsymbol{O(2^n)}$ **Exponential** – time requirement increases too rapidly to be practical.

Extras

Asymptotic Bounds

Rough Guide

class	in English	meaning	key phrases
$f(n) = O(g(n))$	big-oh	$f(n) \leq g(n)$	$f(n)$ is asymptotically no worse than $g(n)$ $f(n)$ grows no faster than $g(n)$
$f(n) = \Theta(g(n))$	big-theta	$f(n) \approx g(n)$	$f(n)$ is asymptotically equivalent to $g(n)$ $f(n)$ grows the same as $g(n)$
$f(n) = \Omega(g(n))$	big-omega	$f(n) \geq g(n)$	$f(n)$ is asymptotically no better than $g(n)$ $f(n)$ grows at least as fast as $g(n)$

Formal Definitions

class	formally	working
$f(n) = O(g(n))$	$\exists c > 0,\ \exists n_0,\ \forall n > n_0, f(n) \leq c \cdot g(n)$	
$f(n) = \Theta(g(n))$	$\exists c_1, c_2 > 0,\ \exists n_0,\ \forall n > n_0, c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$
$f(n) = \Omega(g(n))$	$\exists c > 0,\ \exists n_0,\ \forall n > n_0, c \cdot g(n) \leq f(n)$	$g(n) = O(f(n))$

Using Limits

$$\text{If } \lim_{n \to \infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & \text{then } f(n) = O(g(n)) \\ \text{some finite, non-zero, positive constant} & \text{then } f(n) = \Theta(g(n)) \\ \infty & \text{then } f(n) = \Omega(g(n)) \end{cases}$$

Principle of Mathematical Induction

To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

BASIS STEP: We verify that $P(1)$ is true.

INDUCTIVE STEP: We show that the conditional statement $P(k) \rightarrow P(k + 1)$ is true for **all positive integers *k***.

Example: Prove that $\mathrm{k} < 2^k$ for all positive integers k

BASIS STEP: Show P(1)

$1 < 2$ ✓

INDUCTIVE STEP:

Assume P(k): $k < 2^k$

Show P(k+1): $k + 1 < 2^{k+1}$

$k \quad < 2^k$

$k + 1 < 2^k + 1$

$2^k + 1 < 2^k + 2^k$, true because $1 < 2^k$

$k + 1 \quad < \quad 2^k + 2^k$

$k + 1 \quad < \quad 2^{k+1}$ ✓

Example: Prove that $2^k < k!$ for every integer k with $k \geq 4$

BASIS STEP: Show P(4)

$16 < 24$ ✓

INDUCTIVE STEP:

Assume P(k): $2^k < k!$

Show P(k+1): $2^{k+1} < (k+1)!$

$2(2^k) < 2(k!)$

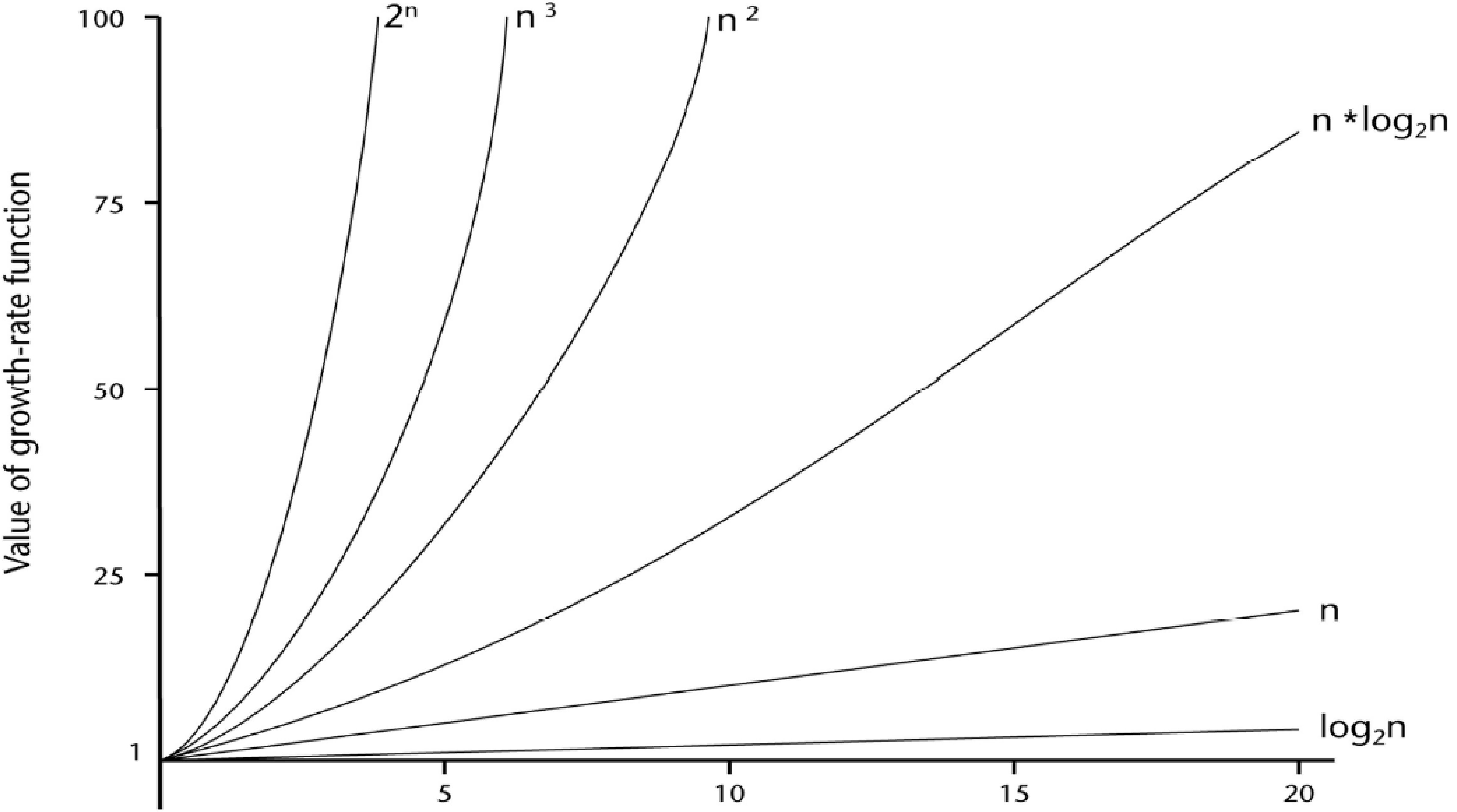
$2^{k+1} < 2(k!)$

$2(k!) < (k+1)(k!)$, true because $2 < k+1$

$2^{k+1} < (k+1)(k!)$

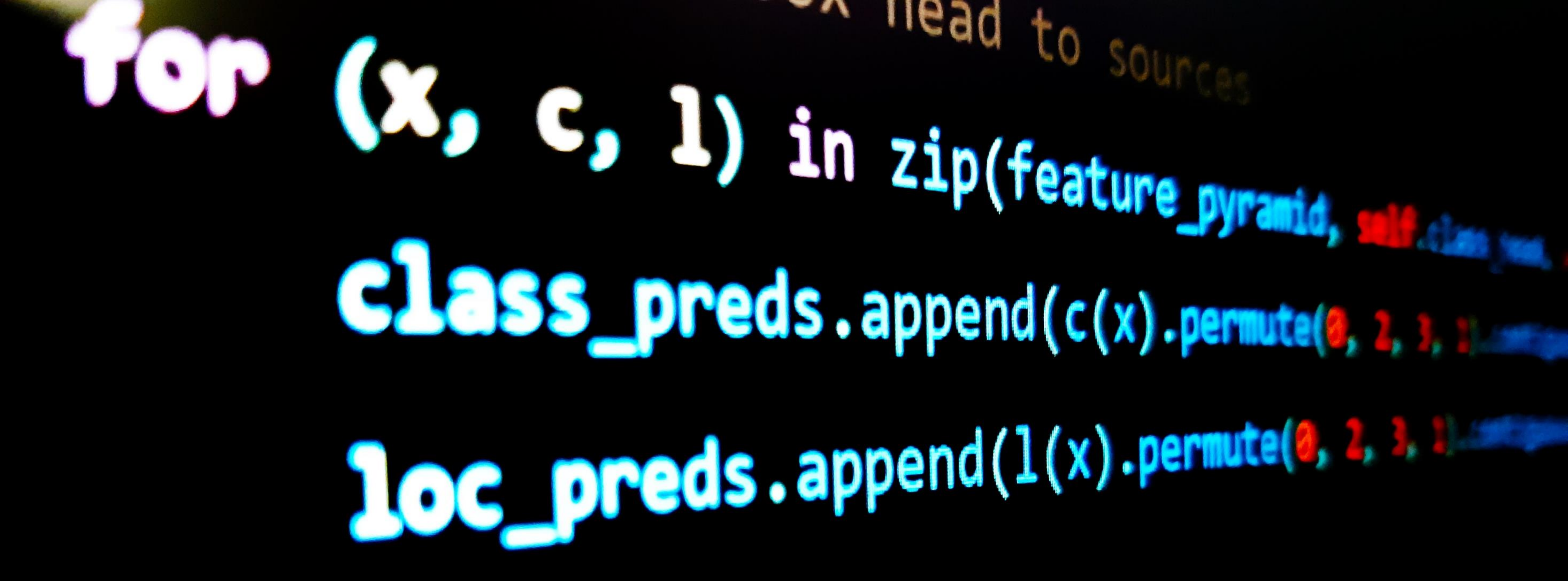
$2^{k+1} < (k+1)!$ ✓

(b)


Value of growth-rate function
100
75
50
25
1
5
10
15
20
2ⁿ
n 3
n 2
n *log2n
n
log2n

Asymptotic Bounds

- Any **exponential** function of n dominates any **polynomial** function of n
- A **polynomial degree k** dominates a **polynomial of degree m** iff **k > m**
- Any **polynomial** function of n dominates any **logarithmic** function of n
- Any **logarithmic** function of n dominates a **constant** term



ASYMPTOTIC BOUNDS

CCDSALG T2 AY 2020-2021