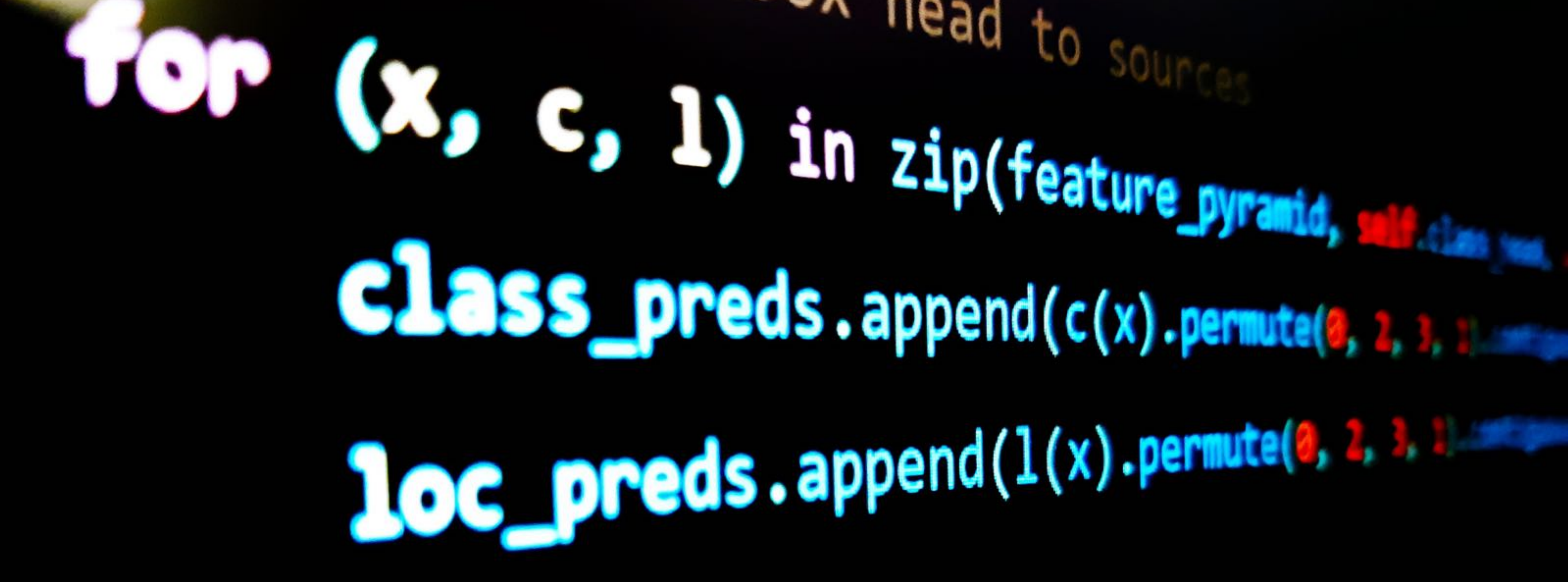


ALGORITHM ANALYSIS

CCDSALG T2 AY 2020-2021

ALGORITHMS

ALGORITHMS

"an algorithm is any well-defined computational **procedure** that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**."

(Cormen, Leiserson, Rivest, & Stein, 2002)

ALGORITHMS

"an algorithm is any well-defined computational **procedure** that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**", to solve a given **problem**

ALGORITHMS

Example

Input: a sequence of n numbers $\{a_1, a_2, \dots, a_n\}$

Output: a permutation (reordering) $\{a'_1, a'_2, \dots, a'_n\}$ such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$

Solution: ???

ALGORITHMS

Example

Input: a sequence of n numbers $\{a_1, a_2, \ldots, a_n\}$

Output: a permutation (reordering) $\{a'_1, a'_2, \ldots, a'_n\}$ such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$

Solution: Sorting algorithms

ALGORITHMS

Image credits

ALGORITHMS

- There can be **more than one algorithm** to solve a given problem.
- An algorithm can be **implemented using different programming languages** on different platforms

DESIGN ISSUES

Correctness

Does the algorithm solve the computational problem?

Efficiency

How fast can the algorithm run?

PERFORMANCE

Time

What parts of the algorithm affects the runtime?

Space

How does the choice of data structure affect the runtime?

ANALYZING ALGORITHMS

Study Behavior

What happens if the input size is increased?

Predict Performance

Predict the execution time and memory consumption of an algorithm

ANALYZING ALGORITHMS

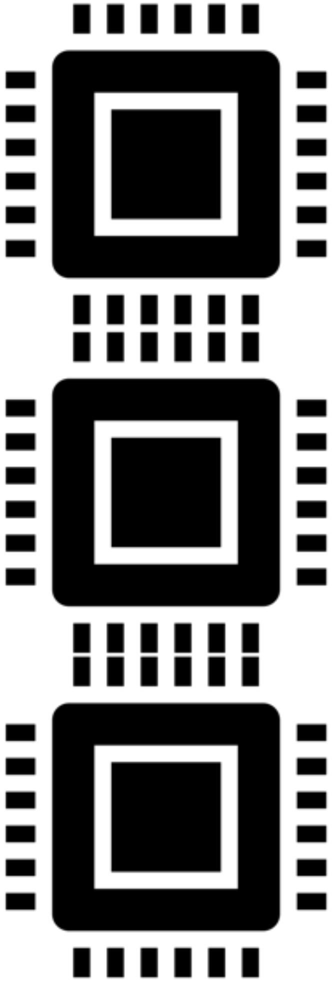
Compare Algorithms

Is there a more efficient way of solving the problem?

Design Better Solutions

Given an existing algorithm, can a modified optimal algorithm be defined?

ANALYZING ALGORITHMS



Assumptions

- Instructions are executed **ONE AT A TIME**, no concurrent operations.
- Instructions are implemented following instructions commonly found in real computers (load, store, copy, add, subtract, multiply, divide, remainder, floor, ceiling, return, etc.).

ANALYSIS METHODS

A priori Analysis

Obtains a function bounding the time complexity through from mathematical facts.

A posteriori Analysis

Study the exact time and space required for execution using actual experiments

ANALYSIS METHODS

It is impossible to know the **exact amount of time** to execute any command unless the following are known:

- Machine for execution
- Machine instruction set
- Time required by each machine instruction
- Translation of the compiler from source to machine language

ANALYSIS METHODS

A posteriori Analysis Example

	Input Size	
	$n = 10$	$n = 100$
Algorithm 1	1 sec	10 secs
Algorithm 2	3 secs	15 secs

Which of the two algorithms is more efficient?

ANALYSIS METHODS

A posteriori Analysis Example

	Input Size		
	$n = 10$	$n = 100$	$n = 1000$
Algorithm 1	1 sec	10 secs	100 secs
Algorithm 2	3 secs	15 secs	30 secs

Which of the two algorithms is more efficient?

ANALYSIS METHODS

A priori Analysis Example

Let A[i] be the ith number on the list $(a_1, a_2, \ldots, a_n)$

```
[1] max, min = A[1]
[2] for i = 2 to n
[3]    if A[i] > max then
[4]            max = A[i]
[5]    if A[i] < min then
[6]            min = A[i]
[7] return max + min
```

```
[1] 2
[2] n
[3] n - 1
[4] n - 1
[5] n - 1
[6] n - 1
[7] 1

Total: 5n-1 = O(n)
```

Assumptions:

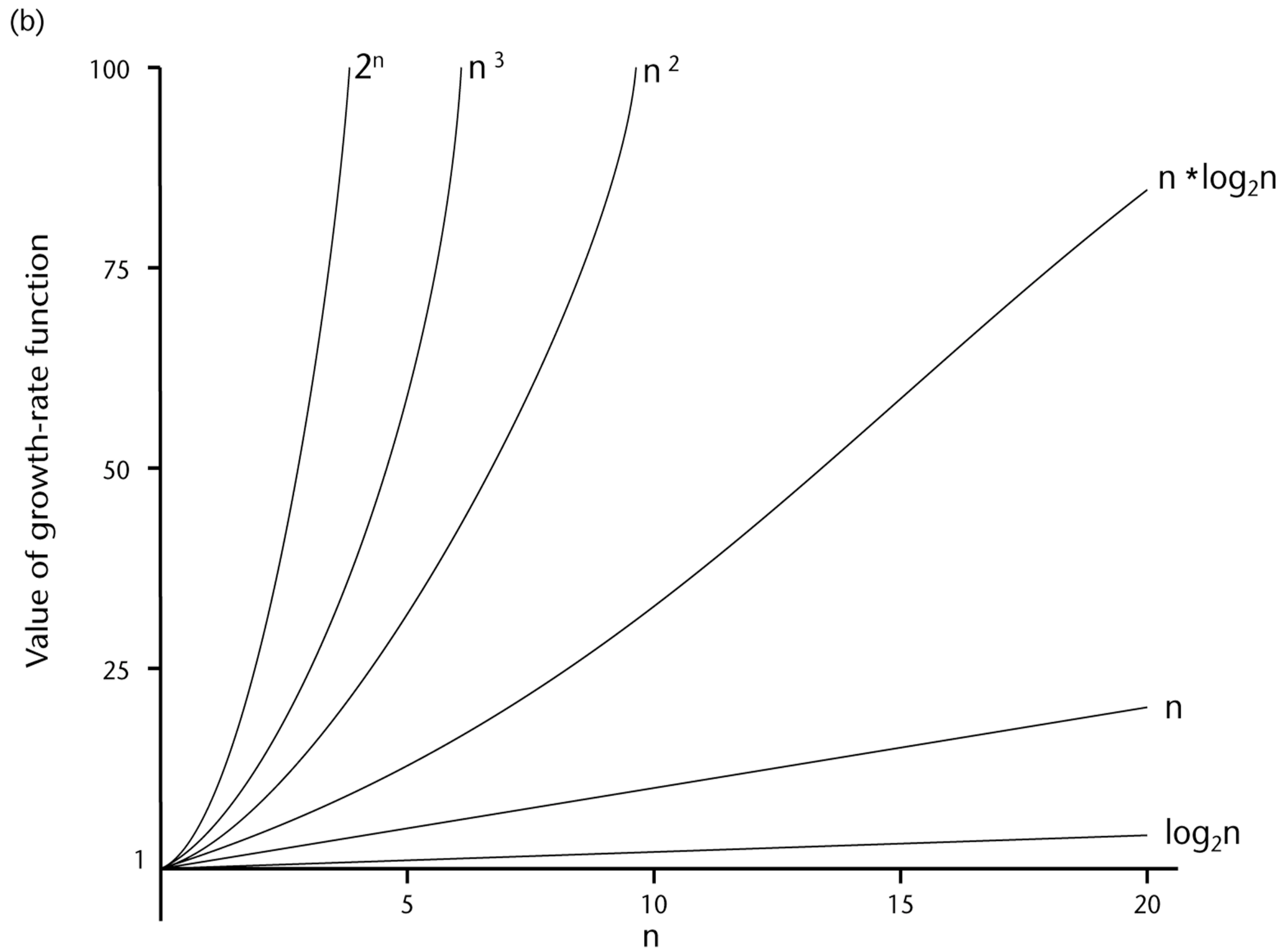
- Instructions are executed sequentially
- Each instruction takes 1-time unit

ANALYSIS METHODS

- Time taken by an algorithm grows with the input size
  - **Input size** – Number of items in the input
  - **Running time** – Number of steps executed

- Our concern: **Rate of Growth** or **Order of Growth**

RATE OF GROWTH



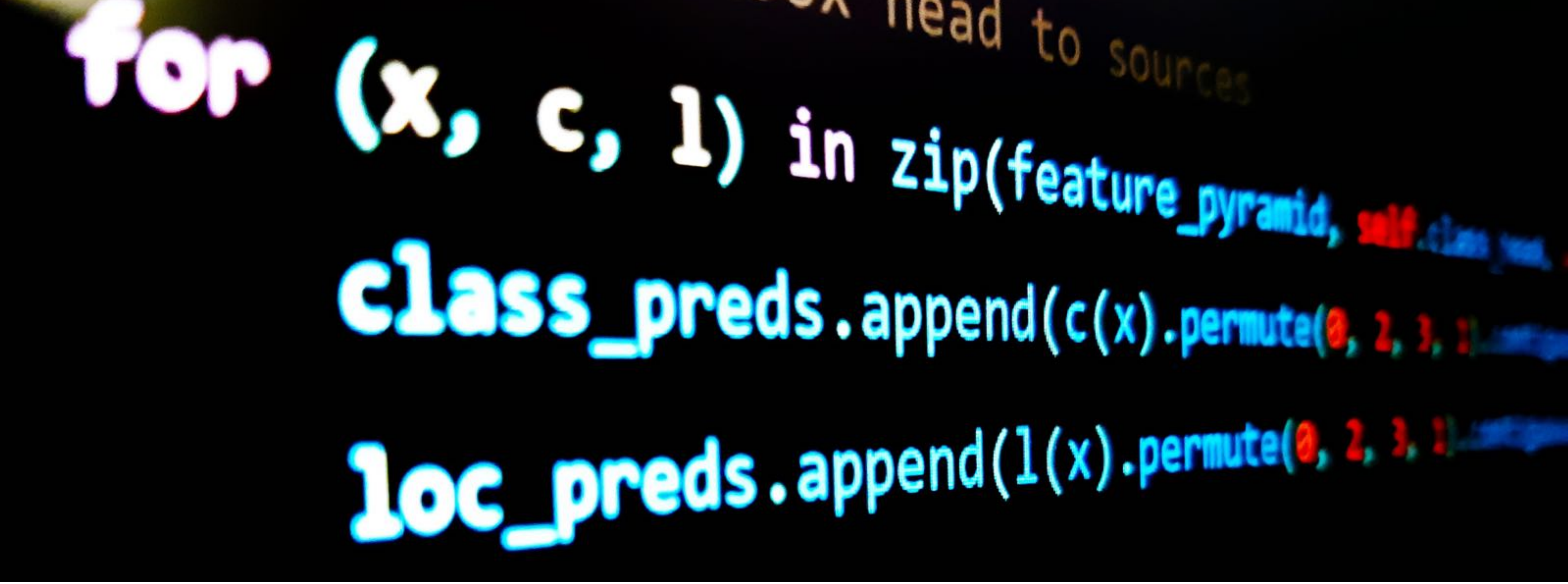
RATE OF GROWTH

If an algorithm takes 1 second to run with the problem size 8, what is the time requirement (approximately) for that algorithm with the problem size 16?

$\boldsymbol{O(1)}$	$T(n) = 1$ second
$\boldsymbol{O(log_2 n)}$	$T(n) = \frac{(1\times \log_2 16)}{\log_2 8} = \frac{4}{3}$ seconds
$\boldsymbol{O(n)}$	$T(n) = \frac{1\times16}{8} = 2$ seconds
$\boldsymbol{O(n \log_2 n)}$	$T(n) = \frac{(1\times16\times \log_2 16)}{8\times \log_2 8} = \frac{8}{3}$ seconds
$\boldsymbol{O(n^2)}$	$T(n) = \frac{(1\times16^2)}{8^2} = 4$ seconds
$\boldsymbol{O(n^3)}$	$T(n) = \frac{(1\times16^3)}{8^3} = 8$ seconds
$\boldsymbol{O(2^n)}$	$T(n) = \frac{(1\times2^{16})}{2^8} = 2^8 = 256$ seconds

RATE OF GROWTH

Function	10	100	$1,000$	$10,000$	$100,000$	$1,000,000$
$\mathbf{1}$	1	1	1	1	1	1
$\log_2 n$	3	6	9	13	16	19
n	10	10^2	10^3	10^4	10^5	10^6
$n \log_2 n$	30	664	$9,965$	10^5	10^6	10^7
n^2	10^2	10^4	10^6	10^8	10^{10}	10^{12}
n^3	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
2^n	10^3	10^{30}	10^{301}	$10^{3,010}$	$10^{30,103}$	$10^{301,030}$



ALGORITHM ANALYSIS

CCDSALG T2 AY 2020-2021