Lernraum I: Abstrakte Datentypen, Datenstrukturen und Asymptotische Notation

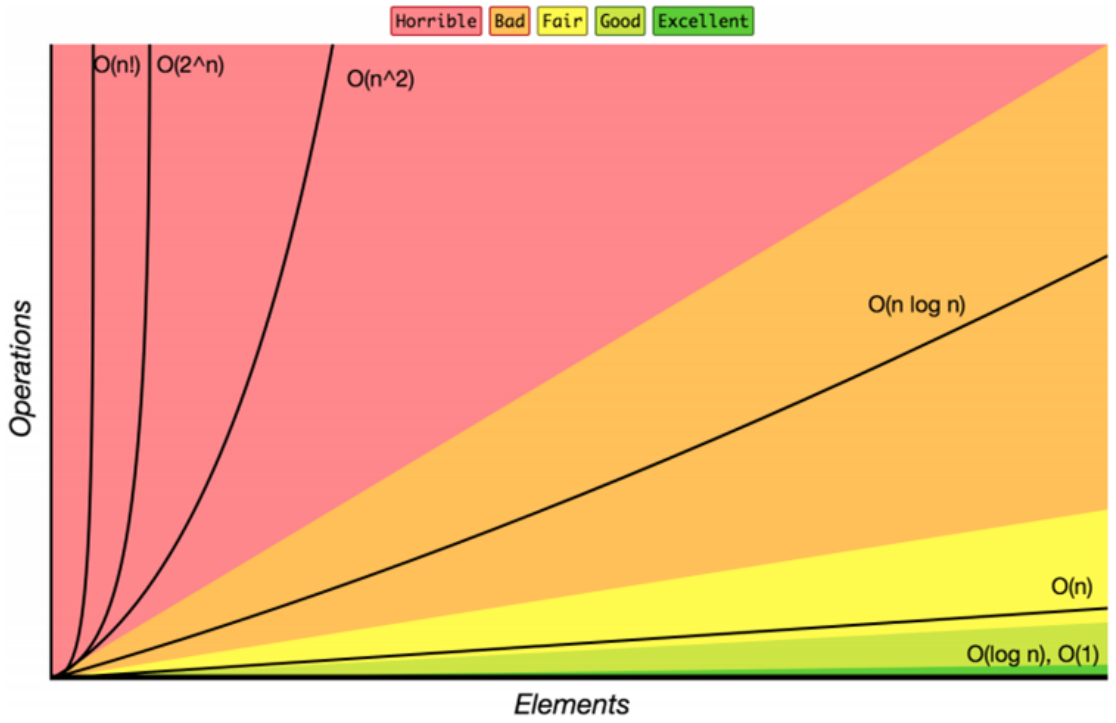
- Abstrakte Datentypen
  - Liste, Stapel, Warteschlange, Tabelle (Dictionary)
- Datenstrukturen
  - Array, Verkettete Liste
- Implementierungen der Abstrakten Datentypen mit den Datenstrukturen …
  - Array
  - Verkettete Listen
- Analyse von Laufzeit und Speicherbedarf der Implementierungen
  - O-Notation
- Laufzeitanalyse von Algorithmen durch Asymptotische Notation
  - O-Notation, Ω-Notation, Θ-Notation
  - Fallanalyse: Worst Case, Best Case Analyse
- Laufzeitanalyse von Rekursiven Algorithmen durch Asymptotische Notation

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Technology Arts Sciences TH Köln

Klassifikation der Wachstumsordnung

Wachstumsordnung: Eine Kategorie der Algorithmeneffizienz, die auf dem Verhältnis des Algorithmus zur Eingabegröße n basiert.

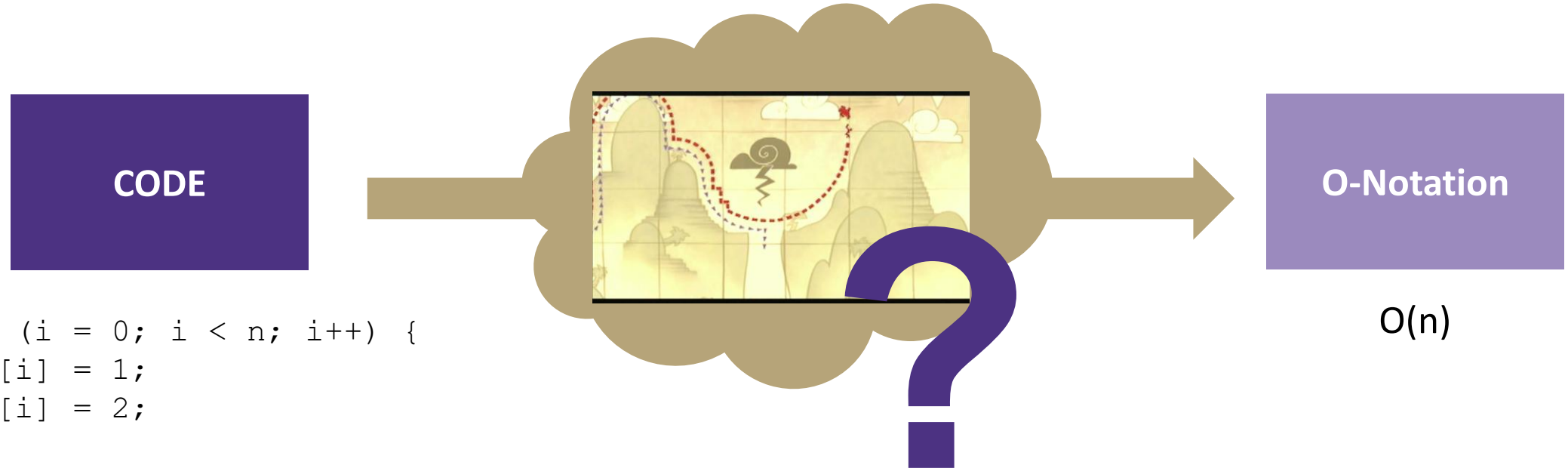
Description	O-Notation	Runtime if you double n	Example Algorithm
constant	O(1)	unchanged	Accessing an index of an array
logarithmic	$O(\log_2 n)$	increases slightly	Binary search
linear	O(n)	doubles	Looping over an array
log-linear	$O(n \log_2 n)$	slightly more than doubles	Merge sort algorithm
quadratic	$O(n^2)$	quadruples	Nested loops!
...	...	...	...
exponential	$O(2^n)$	multiplies drastically	Fibonacci with recursion



Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Quelle: Data Structures and Algorithms (CSE 373), University of Washington, Spring 2022
https://www.bigocheatsheet.com/

Code in O-Notation umwandeln




```
for (i = 0; i < n; i++) {
  a[i] = 1;
  b[i] = 2;
}
```

Bisher: “O(1) bedeutet keine Schleifen, O(n) ist eine Schleife, O(n^2) ist verschachtelte Schleife”

- Das ist weiterhin nützlich!
- In diesem Modul gehen wir mehr ins Detail: Damit wir auch komplizierteren Code in die O-Notation umwandeln können

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik

Analyse von Algorithmen


```
for (i = 0; i < n; i++) {
  a[i] = 1;
  b[i] = 2;
}
```

f(n) = 2n

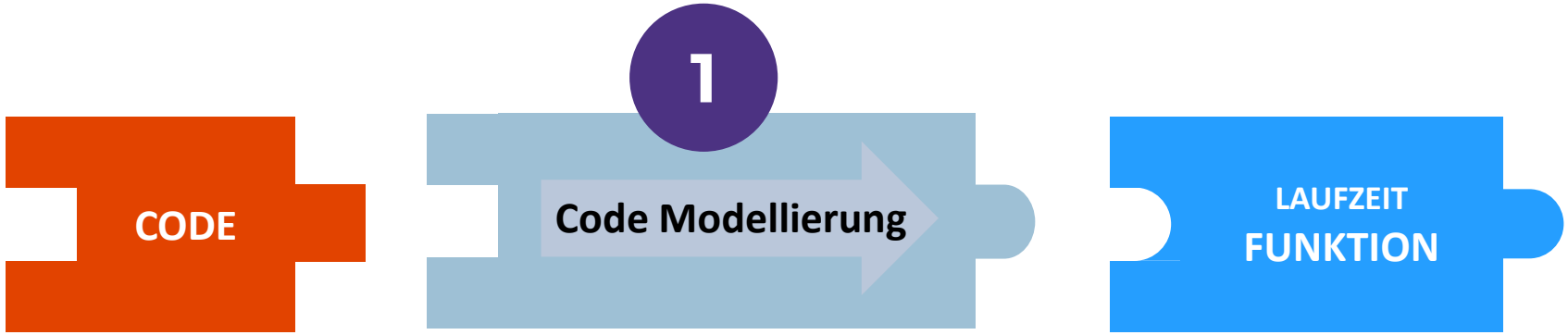
O(n)

Analyse von Algorithmen: Der Gesamtprozess der Zuordnung von Code zu einer Wachstumsordnung, bestehend aus:

Code Modellierung: Code → Laufzeitfunktion, die die Laufzeit des Codes beschreibt

Asymptotische Notation: Funktion → Wachstumsordnung, die das asymptotische Verhalten beschreibt

Code Modellierung



Code Modellierung - der Prozess der mathematischen Darstellung, wie viele Operationen ein Stück Code im Verhältnis zur Eingabegröße **n** ausführt.

- Umwandlung von Code in eine Laufzeitfunktion, die seine Laufzeit darstellt

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Data Structures and Algorithms (CSE 373), University of Washington, Spring 2022
CSE 373 20 AU – SCHAFER

Code Modellierung

Beispiel 2

```
public int method2(int n) {
    int sum = 0;                    +1
    int i = 0;                      +1
    while (i < n) {                 +1
        int j = 0;                  +1
        while (j < n) {             +1
            if (j % 2 == 0) {       +2
                // do nothing
            }
            sum = sum + (i * 3) + j;    +4
            j = j + 1;              +2
        }
        i = i + 1;                  +2
    } return sum;                   +1
}
```

$$f(n) = (9n+4)n + 3$$

Innere Schleife läuft n-mal

+9 *n

Äußere Schleife läuft n-mal

9n + 4 *n

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Data Structures and Algorithms (CSE 373), University of Washington, Spring 2022
CSE 373 20 AU – SCHAFER

O-, Ω- und Θ-Notation

Übung

Welche der folgenden Funktionen ist in O(n²)? Ω(n²)? Θ(n²)?

a. $f(n) = 42$

f(n) ∈ O(n²)

b. $f(n) = 5n + 100$

f(n) ∈ O(n²)

c. $f(n) = nlog_2(3n)$

f(n) ∈ O(n²)

d. $f(n) = 4n^2 - 2n + 10$

f(n) ∈ O(n²) f(n) ∈ Ω(n²) f(n) ∈ Θ(n²)

e. $f(n) = 2^n$

f(n) ∈ Ω(n²)

O-Notation

$f(n) \in O(g(n))$ falls positive Konstanten c und n_0 existieren, sodass $0 \leq f(n) \leq c \cdot g(n)$ für alle $n \geq n_0$

Ω-Notation

$f(n) \in \Omega(g(n))$ falls positive Konstanten c und n_0 existieren, sodass $f(n) \geq c \cdot g(n)$ für alle $n \geq n_0$

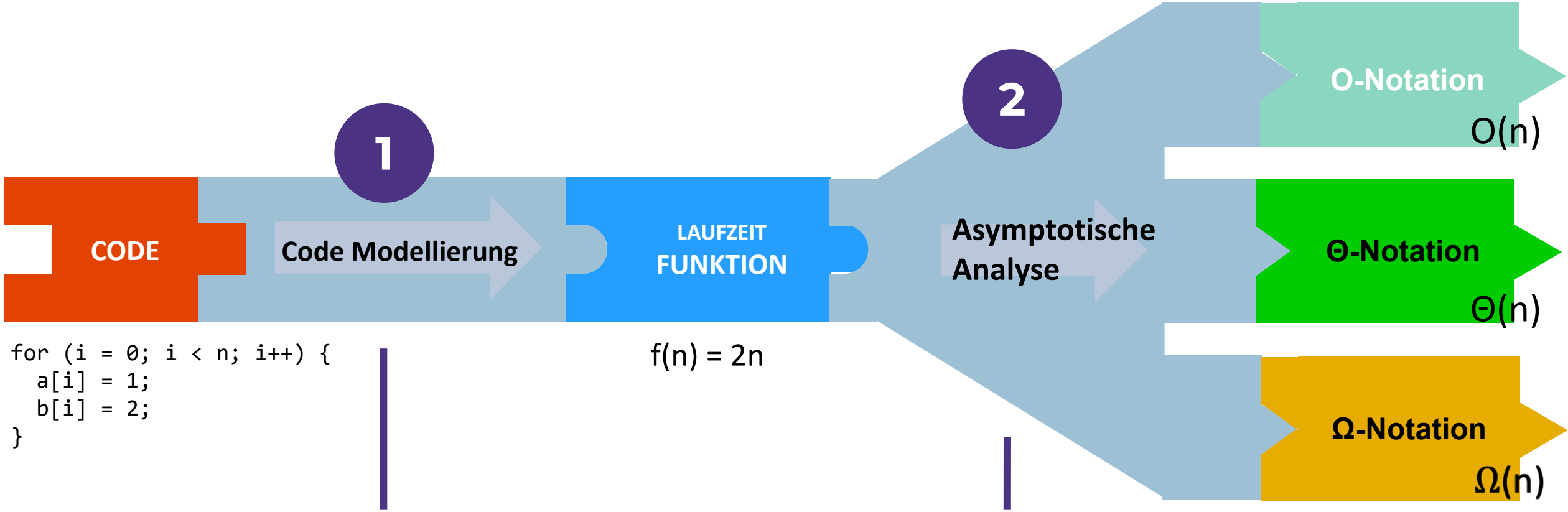
Θ-Notation

$f(n) \in \Theta(g(n))$ falls $f(n) \in O(g(n))$ und $f(n) \in \Omega(g(n))$.

$f(n) \in \Theta(g(n))$ falls positive Konstanten c_1, c_2 und n_0 existieren, sodass $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ für alle $n \geq n_0$

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Data Structures and Algorithms (CSE 373), University of Washington, Spring 2022
CSE 373 20 SP – CHUN & CHAMPION

Analyse von Algorithmen




```
for (i = 0; i < n; i++) {
  a[i] = 1;
  b[i] = 2;
}
```

f(n) = 2n

Schauen wir uns dieses Werkzeug nun genauer an. Wie genau kommen wir zu dieser Funktion?

Wir haben gerade dieses Werkzeug fertiggestellt, um eine Funktion anhand einiger nützlicher Schranken zu charakterisieren!

Prof. Dr. Daniel Gaida
Professor für Cyber-Physische Systeme
Fakultät für Informatik und Ingenieurwissenschaften - Institut für Informatik
Quelle: Data Structures and Algorithms (CSE 373), University of Washington, Spring 2022