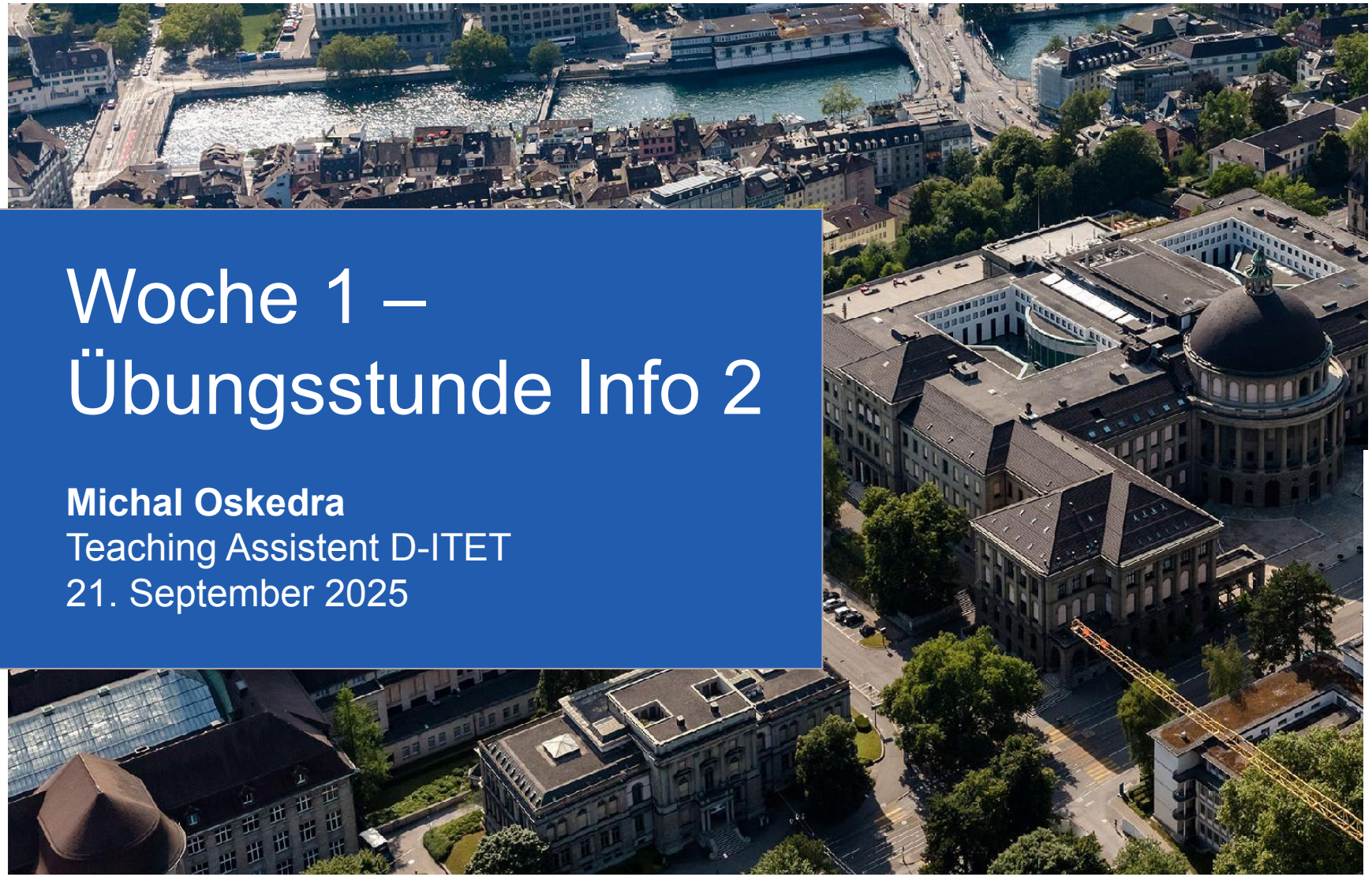
ETH zürich



Woche 1 – Übungsstunde Info 2

Michal Oskedra
Teaching Assistent D-ITET
21. September 2025

Info2 - ML J 34.1 14-16
WhatsApp group

moskedra-web.ethz.ch

00 - Organisatorisches

Wer bin ich?

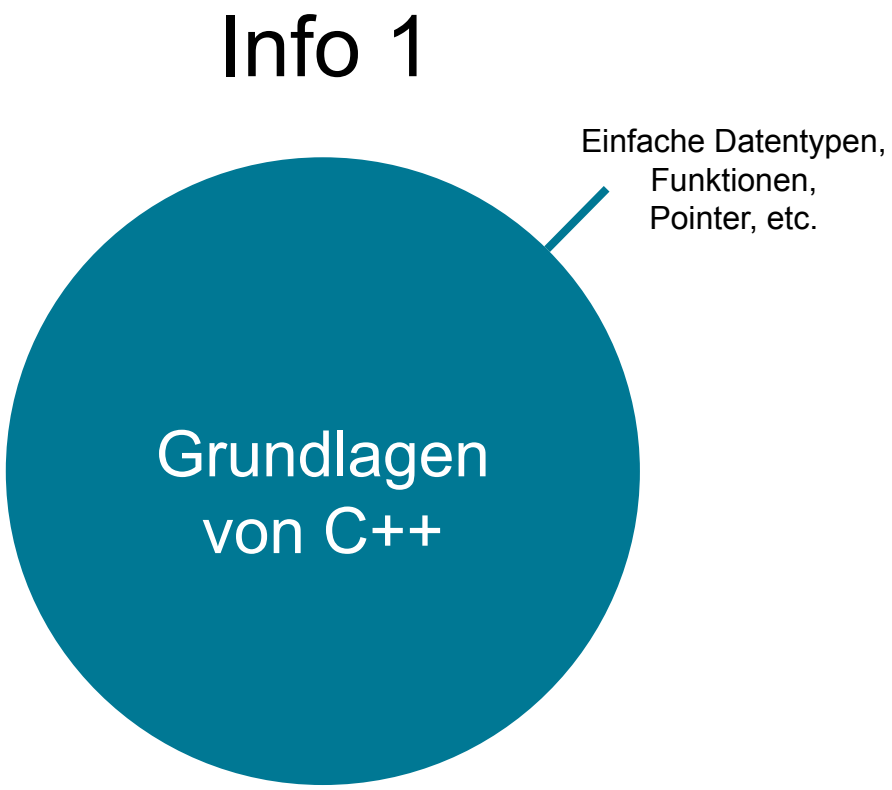
- Michal (einfach Michi), aus Polen (Krakau)
- 7.Sem D-ITET
- Klettern, Reisen, Tüfteln liebe Projekte zu machen!

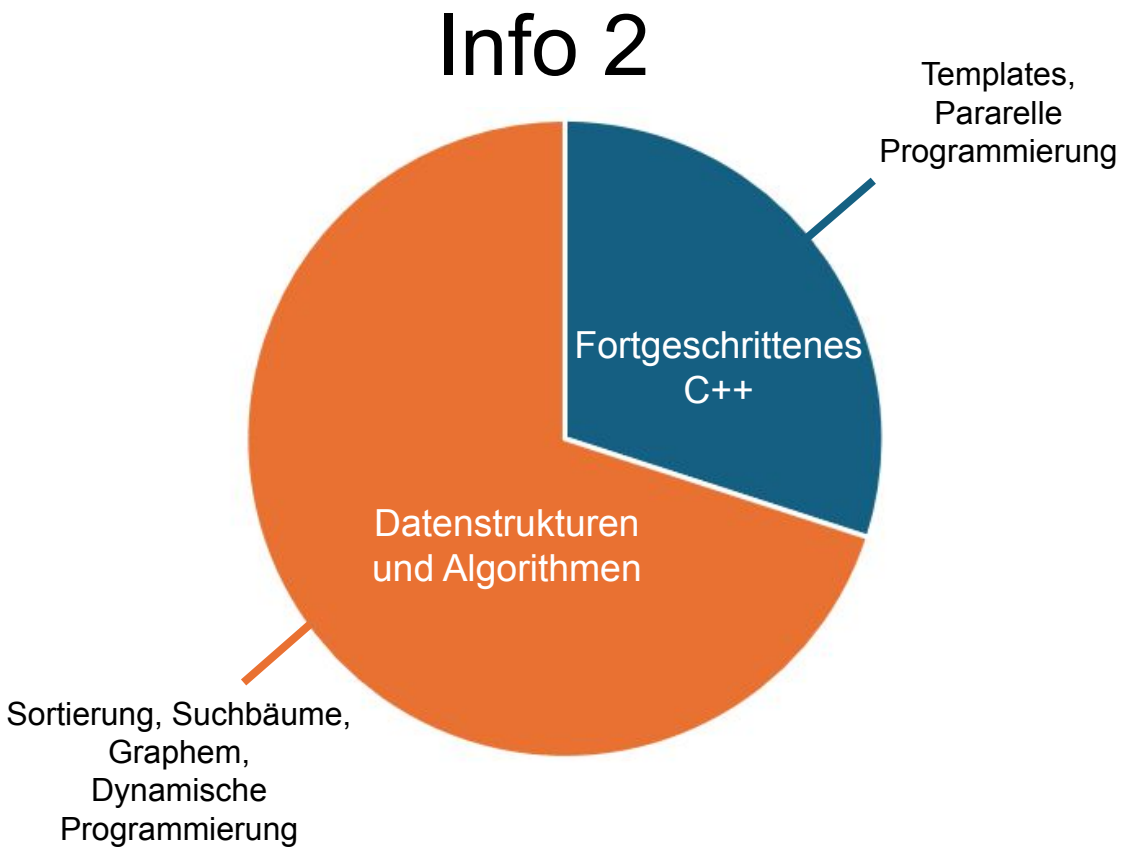






Was ist Info2?





Wieso ist Info 2 wichtig?



- Ineffiziente Algorithmen können lange gehen...
- Als ElektrotechnikerInnen haben wir oft wenig Ressourcen zur Verfügung
- Kenntnis von Algorithmen wird oft in Bewerbungsgesprächen und Code Jams gefragt!

Ineffiziente Algorithmen können lange gehen...

Fibonacci Folge:

$n_0 = 0, n_1 = 1$

$n_k = n_{k-1} + n_{k-2}$

Also: $n_0 = 0, n_1 = 1, n_2 = 1, n_3 = 2, n_4 = 3, n_5 = 5, n_6 = 8, n_7 = 13$, etc...

Was ist n_{100}?

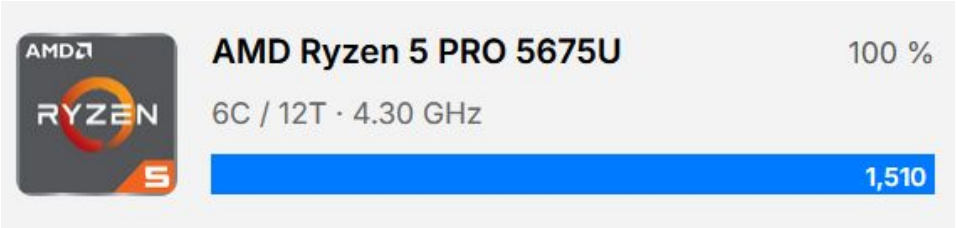
Ineffiziente Algorithmen können lange gehen...

```
#include <iostream>

unsigned long long fib(unsigned int n) {
    if (n < 2) return n; // base cases: 0 and 1
    return fib(n - 1) + fib(n - 2); // recursive case
}

int main() {
    unsigned int n = 100;
    std::cout << fib(n) << '\n';
    return 0;
}
```

Sandbox Task:
https://expert.ethz.ch/ide2/QPS72tPT6Hy5XoswY



442 Milliarden Operationen / Sekunde

Wie lange braucht es, um n_{100} zu berechnen?

a) 5 ns
b) 5 s
c) 5 min
d) 5 h
e) 5 Tage
f) Keins davon?

Ineffiziente Algorithmen können lange gehen...

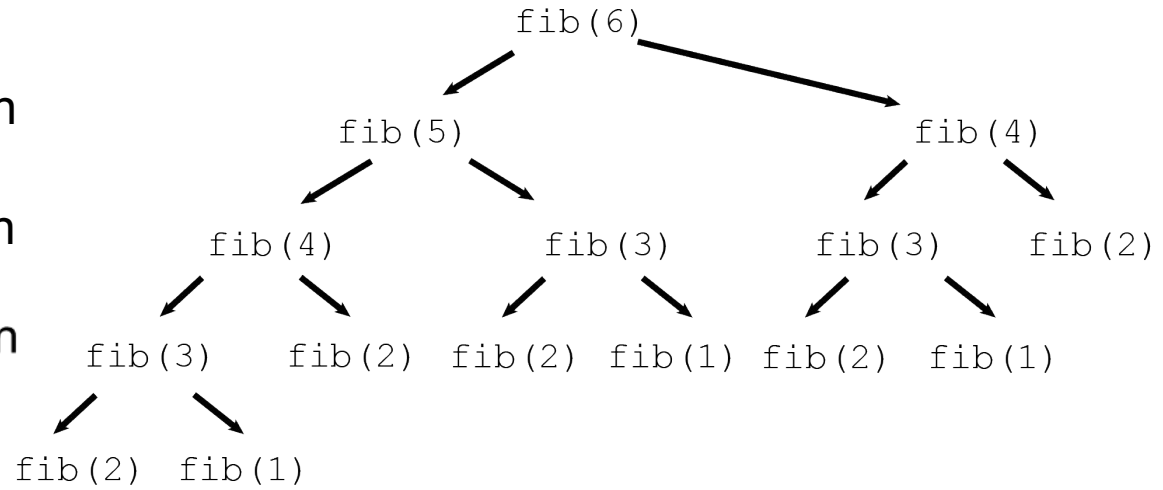
2 Operationen

4 Operationen

8 Operationen

…

2^n Operationen





Alter Der Erde:
4’600’000’000 Jahre

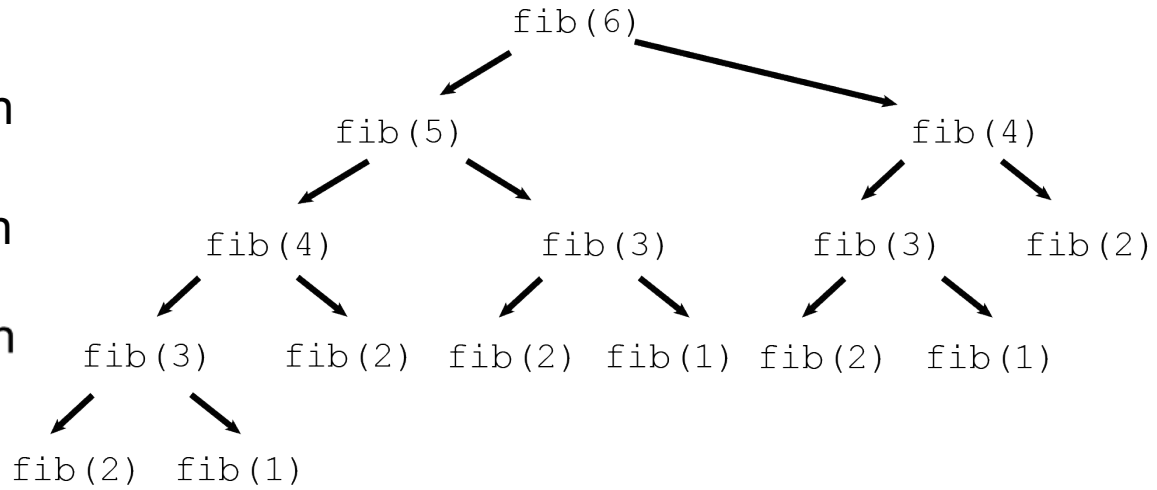
Ineffiziente Algorithmen können lange gehen...

2 Operationen

4 Operationen

8 Operationen

…



2^n Operationen



Alter Der Erde:
4’600’000’000 Jahre

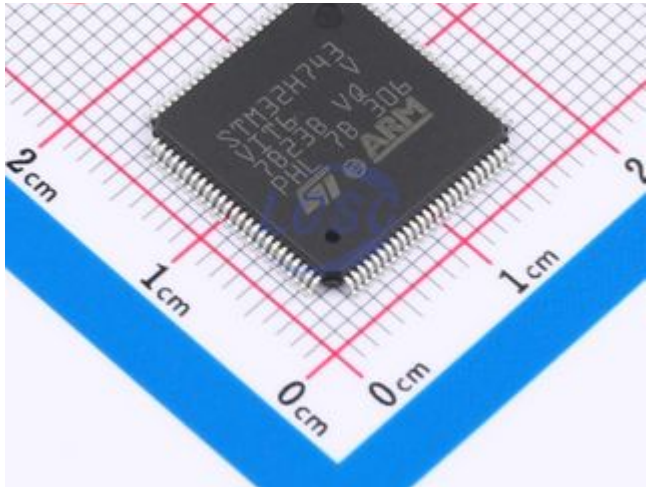
$$2^{100} = 1'267'650'600'228'229'401'496'703'205'376 \text{ Operationen}$$

$$\frac{1'267'650'600'228'229'401'496'703'205'376}{442'000'000'000\ \frac{Operationen}{s}} \approx 2'867'987'783'321'785'976.2368850800362\ s$$

$$\frac{2'867'987'783'321'785'976.2368850800362\ s}{60\frac{s}{min} * 60\frac{min}{h} * 24\frac{h}{Tag} * 365.25 * \frac{Tag}{Jahr}} = 90'905'937'416\ Jahre$$

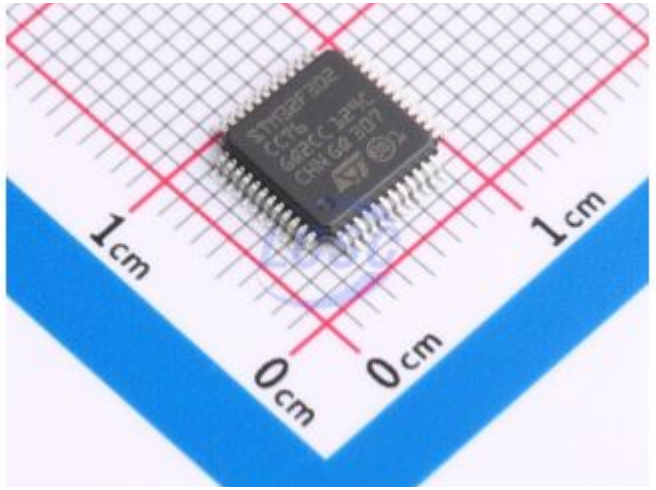
Als ElektrotechnikerInnen haben wir oft wenig Ressourcen zur Verfügung

STM32H743VI Microcontoroller



864 Kbytes of user SRAM

STM32F302CCT6 Microcontroller



32 Kbytes of user SRAM

Fast Fourier Transform

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_n} = E_k + e^{-\frac{2\pi i}{N} k} O_k \qquad \text{for } k = 0, \ldots, \frac{N}{2} - 1.$$

```
67
68    uint16_t raw_data[8192]; //4 bytes * 8192 = 32'768 bytes
69    arm_rfft_q15(&rfft_inst, fft_buf_q15, fft_buf_q15);
70
```

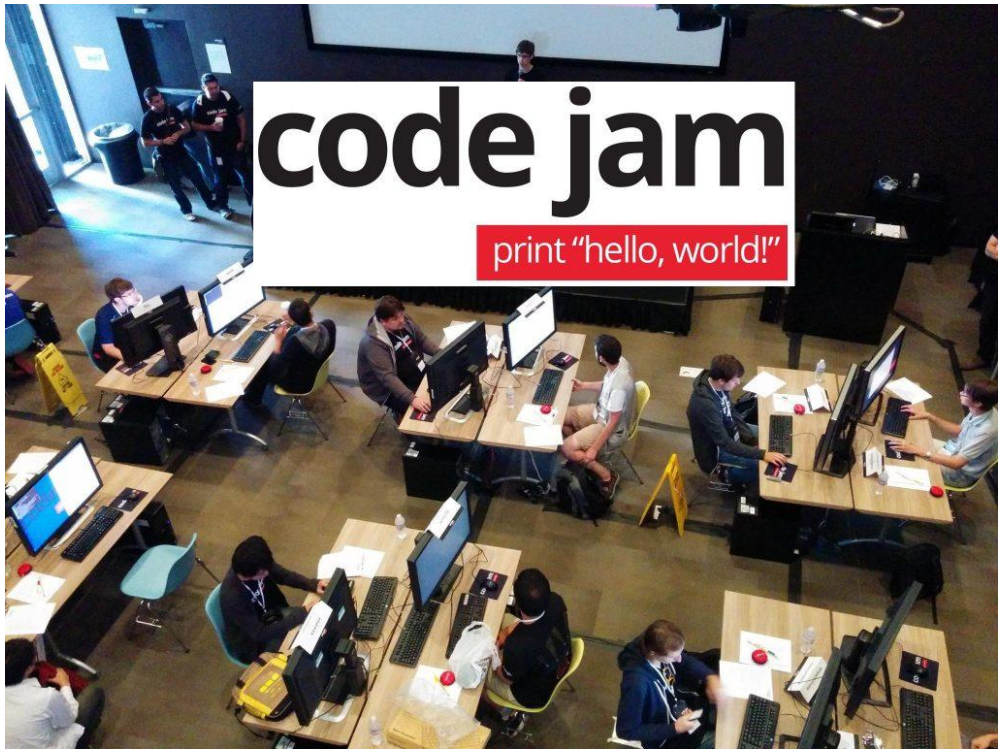
Kenntnis von Algorithmen wird oft in Bewerbungsgesprächen und Code Jams gefragt!

IGotAnOffer

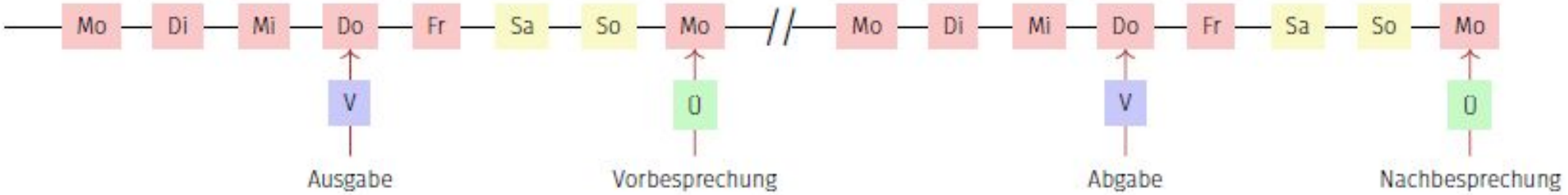
Google coding interview question types

#1	**Arrays & Strings** *E.g. Implement Trie for prefix matching.*	42.9%
#2	**Graphs & Trees** *E.g. Find the lowest common ancestor of two nodes.*	33.7%
#3	**Dynamic Programming** *E.g. Find the minimum number of coins needed for change.*	11.2%
#4	**Recursion** *E.g. Solve the "Generate All Possible Balanced Parentheses" problem recursively.*	6.1%
#5	**Math** *E.g. Evaluate reverse polish notation and return result.*	6.1%

Source: Glassdoor.com data for Google software engineer interview questions. Analysis by IGotAnOffer.



Struktur der Vorlesung



- Übungsausgabe zur Vorlesung (online).
- Vorbesprechung in der folgenden Übung.
- Bearbeitung der Übung bis zur Vorlesung der übernächsten Woche (2 Wochen Bearbeitungszeit). Ausnahme: für die erste Übung haben Sie nur eine Woche Zeit.
- Nachbesprechung der Übung nach der Abgabe. Feedback zu den Abgaben innerhalb einer Woche nach Nachbesprechung.

Sorting Runtimes - Student attempt

```cpp
//Feedback bitte!

std::vector<Value> my_sort(std::vector<Value> a){
  unsigned n = a.size();
  for (unsigned i = 0; i<n ; ++i)
    for (unsigned j = i+1; j<n; ++j)
      if (a[j] < a[i])
        swap(a[i],a[j]);
  return a;
}
```

Nur wer fragt, kriegt Feedback!

Im CodeExpert ein Kommentar hinzufügen
//Feedback bitte!

Fragen?

Immer!

- Per email: moskedra@ethz.ch
- Per WhatsApp: +41 78 854 67 63, WhatsappGruppe
- Website: moskedra-web.ethz.ch

Info2 - ML J 34.1 14-16

WhatsApp group

moskedra-web.ethz.ch

Prüfung

- +/- 90 Punkte
- ca. 30 Multiple-Choice Theorie-Aufgaben à 1-2 Punkte
- 3 Programmier-Aufgaben à 15 Punkte

- Beides Theorie und Programmieren notwendig, um zu bestehen!

Keine Bonusaufgaben!!!

Aufgabe 1: Verschiedenes (30P)

Die Antworten zu dieser Aufgabe müssen innerhalb der totalen Prüfungszeit nach Moodle übertragen werden.

The answers for this task need to be transferred to Moodle within the total examination time.

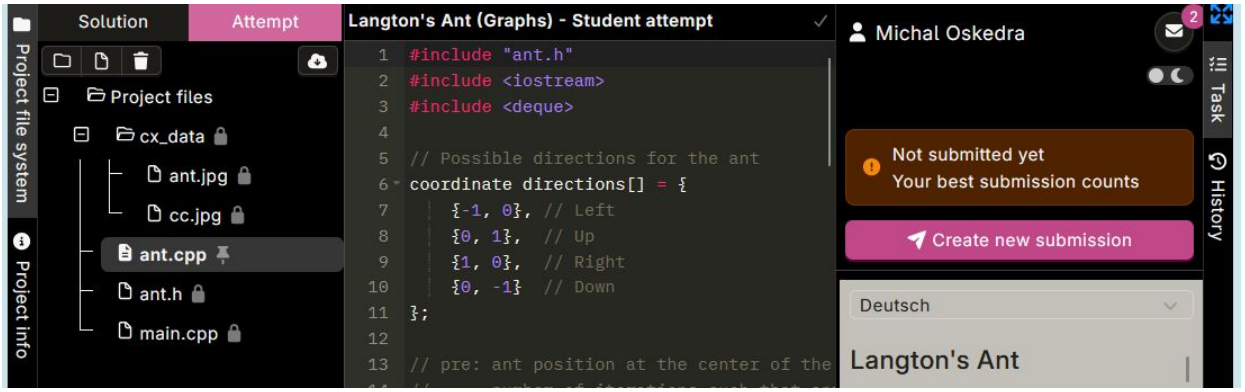
/7P **(1.a)** Kreuzen Sie jeweils an, ob die Aussage wahr oder falsch ist.

For each statement, mark if it is true or false.

(1.a.1)

Es gilt $\Theta(f) = \Omega(f) \cap O(f)$ /
It holds that $\Theta(f) = \Omega(f) \cap O(f)$

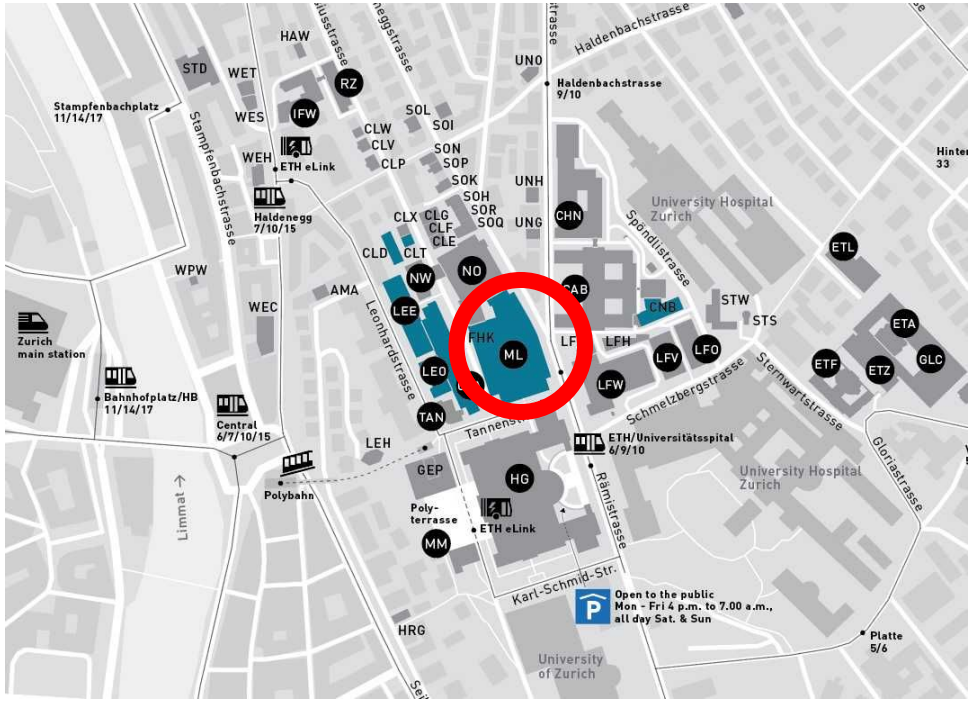
○ Wahr / *True*
○ Falsch / *False*

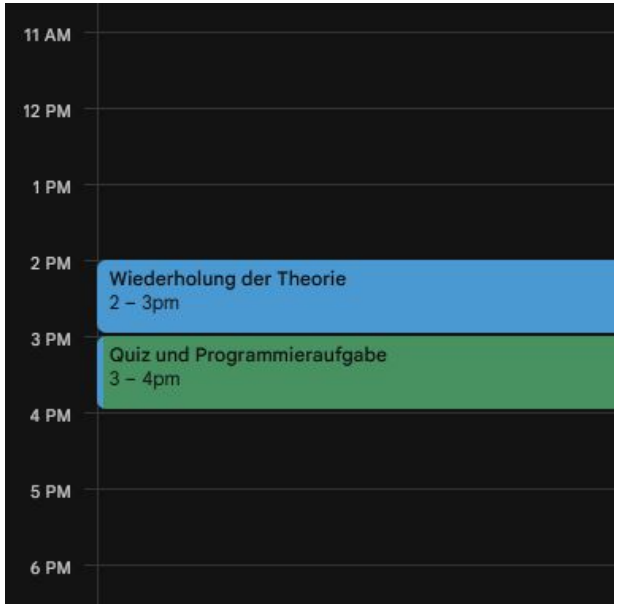


Struktur dieser Übungsstunde

- ML J 34.1 /
- Jeden Montag: 14-16

- 1. Stunde Theorie Wiederholung
- 2. Stunde Kahoot Quiz + Programmieraufgabe

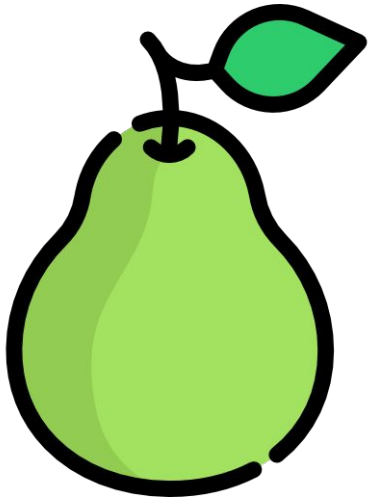




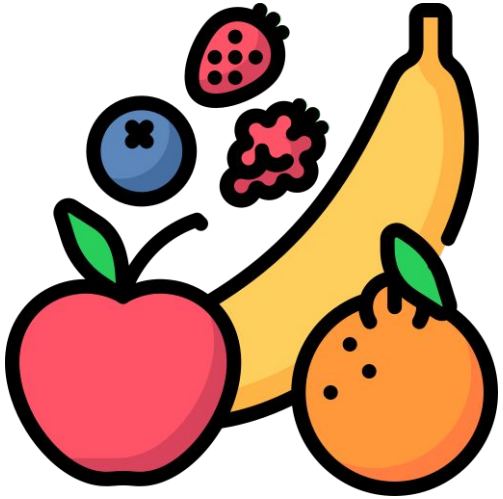
01 – Wiederholung der Theorie

01.1 **Definitionen Effizienz**
01.2 Asymptotische Laufzeiten

Definitionen Effizienz



Ist ein



Ist ein

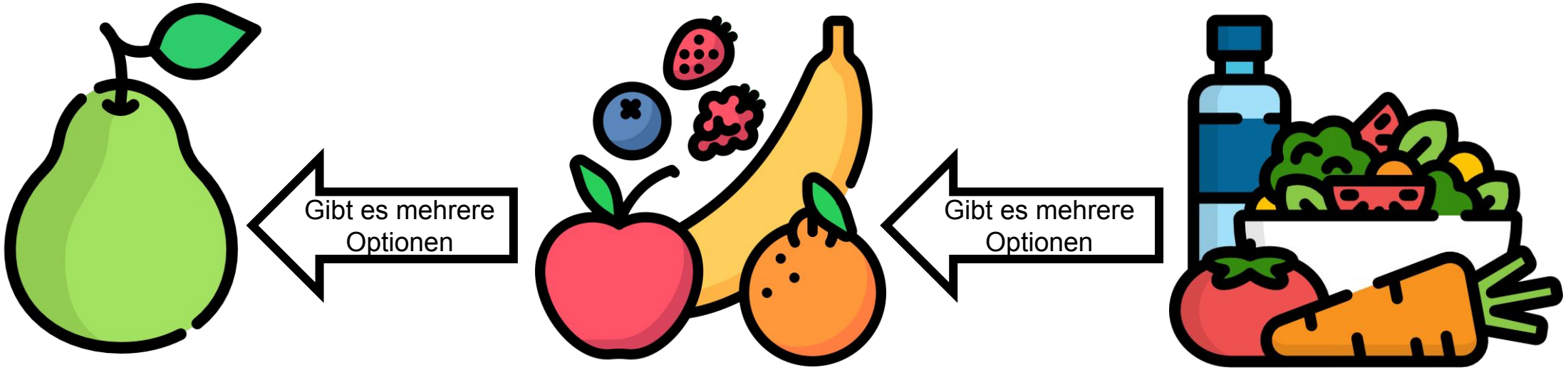


Birne

Frucht

Lebensmittel

Definitionen Effizienz

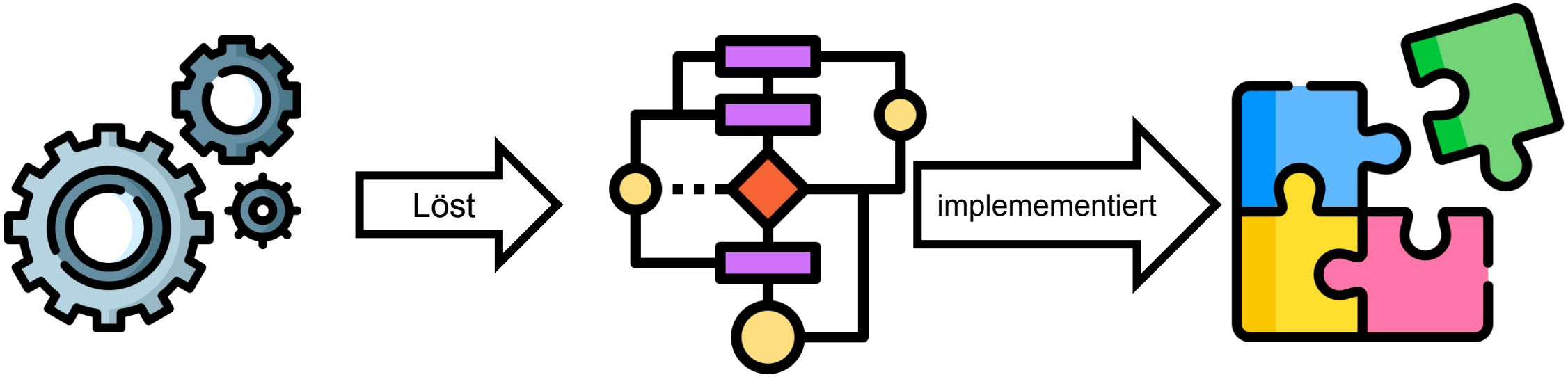


Birne

Frucht

Lebensmittel

Definitionen Effizienz



Programm

Implementierung eines Algorithmus

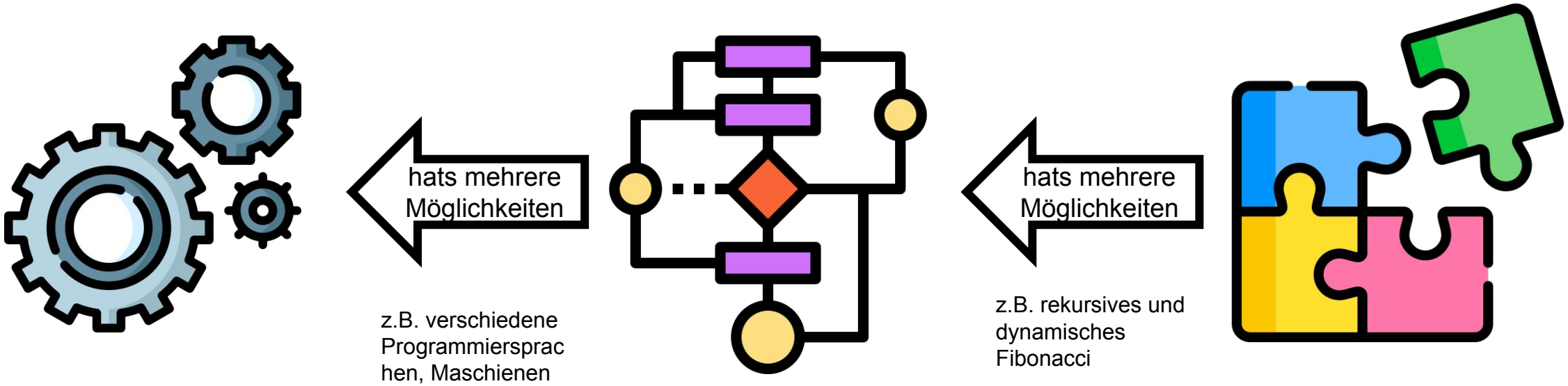
Algorithmus

Eine Berechnungsvorschrift, das eine Ausgabe mit einer Eingabe erzielt

Problem

Eine Abstraktion einer Aufgabe, die wir lösen wollen

Definitionen Effizienz



z.B. verschiedene Programmiersprachen, Maschienen

z.B. rekursives und dynamisches Fibonacci

Programm

Implementierung eines Algorithmus

Algorithmus

Eine Berechnungsvorschrift, das eine Ausgabe mit einer Eingabe erzielt

Problem

Eine Abstraktion einer Aufgabe, die wir lösen wollen

01 – Wiederholung der Theorie

01.1 Definitionen Effizienz
01.2 **Komplexität**

Definition

Komplexität:

Kosten über für ein Algorithmus *A*, ein Problem *P zu* lösen.

Kann sich sowohl auf Rechenaufwand wie auf Speicherplatz beziehen.

Random Access Model

- Instruktionen werden eine nach den anderen ausgeführt
- Elementaroperationen haben gleiche Laufzeit von 1.
- Wir zählen Elemntaroperationen und betrachten Asymptomatisches Verhalten

Elementäroperation	Beispiel
Rechenoperation (+,-,*,/,%)	10 + 1;
Logische Operationen (>, <, ...)	0 > 1;
Sprünge	If ([Bedinung]) , else
Zuweisung	int a = 10;
Maschinenantworten	cout >> '' Hello World'';
Pointer Operationen	*int a = &b;

Wie viele Elementäroperationen?

```cpp
#include <iostream>
#include <cstdlib>   // for rand() and srand()
#include <ctime>     // for time()

n = 10;

int main() {
    // Seed the random number generator with the current time
    srand(static_cast<unsigned int>(time(0))); // ignore that command

    int secretNumber = rand() % 10 + 1; // random number between 1 and 10
    int guess = 0;

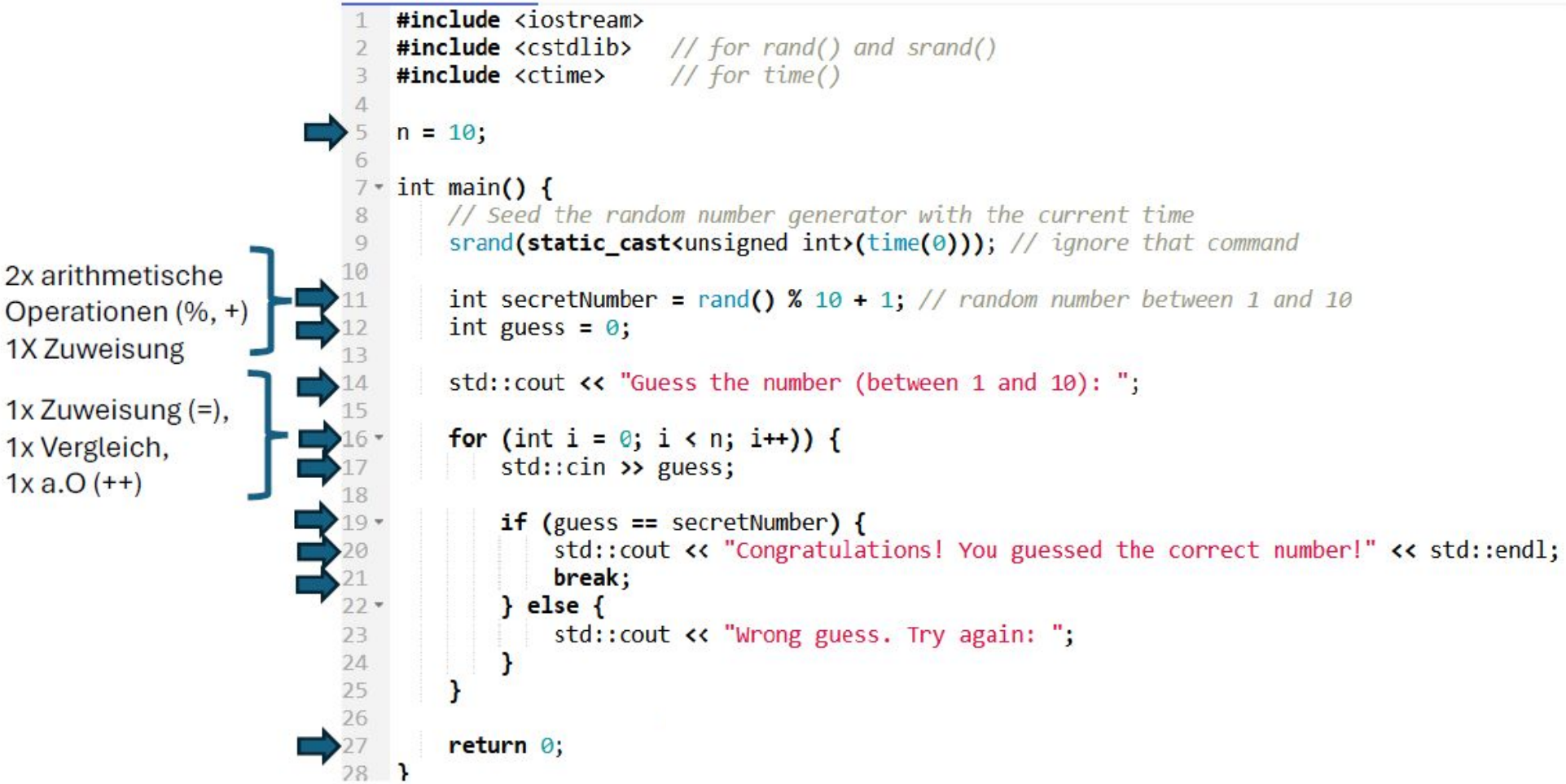
    std::cout << "Guess the number (between 1 and 10): ";

    for (int i = 0; i < n; i++)) {
        std::cin >> guess;

        if (guess == secretNumber) {
            std::cout << "Congratulations! You guessed the correct number!" << std::endl;
            break;
        } else {
            std::cout << "Wrong guess. Try again: ";
        }
    }

    return 0;
}
```

Wie viele Elementäroperationen?




```cpp
#include <iostream>
#include <cstdlib>   // for rand() and srand()
#include <ctime>     // for time()

n = 10;

int main() {
    // Seed the random number generator with the current time
    srand(static_cast<unsigned int>(time(0))); // ignore that command

    int secretNumber = rand() % 10 + 1; // random number between 1 and 10
    int guess = 0;

    std::cout << "Guess the number (between 1 and 10): ";

    for (int i = 0; i < n; i++)) {
        std::cin >> guess;

        if (guess == secretNumber) {
            std::cout << "Congratulations! You guessed the correct number!" << std::endl;
            break;
        } else {
            std::cout << "Wrong guess. Try again: ";
        }
    }

    return 0;
}
```

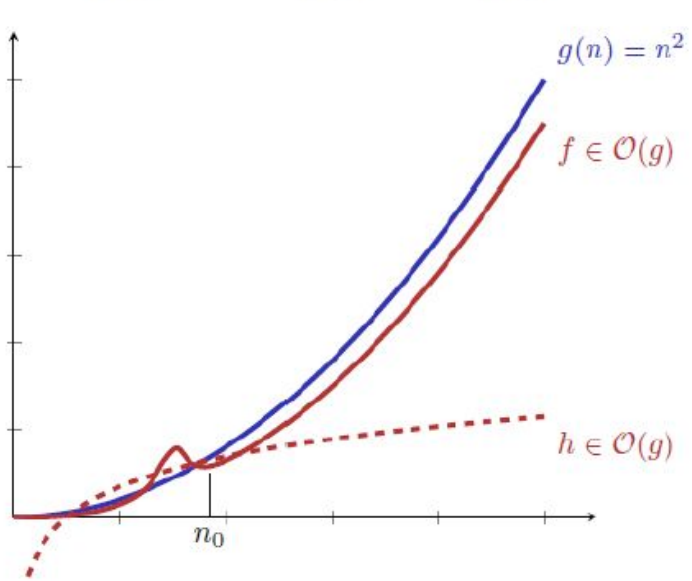
Definition

Asymptotik:

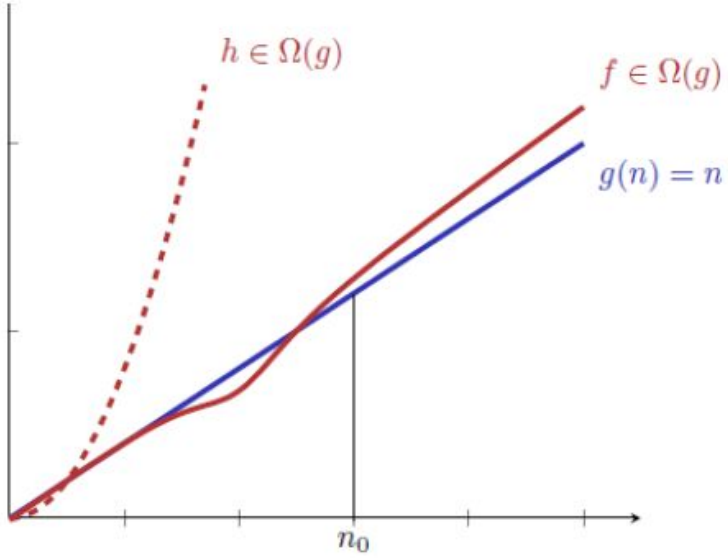
Verhalten der Elementäroperationen bei wachsendem Input

Asymptotische Laufzeiten Definitionen

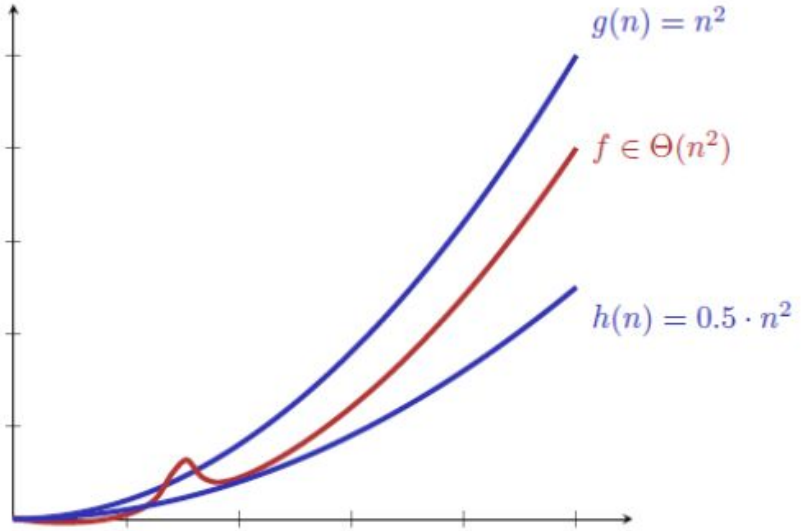
$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq f(n) \leq c \cdot g(n)\}$$
$$\Omega(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} | \forall n \geq n_0 : 0 \leq c \cdot g(n) \leq f(n)\}$$
$$\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$$



Obere Schranke $\mathcal{O}$

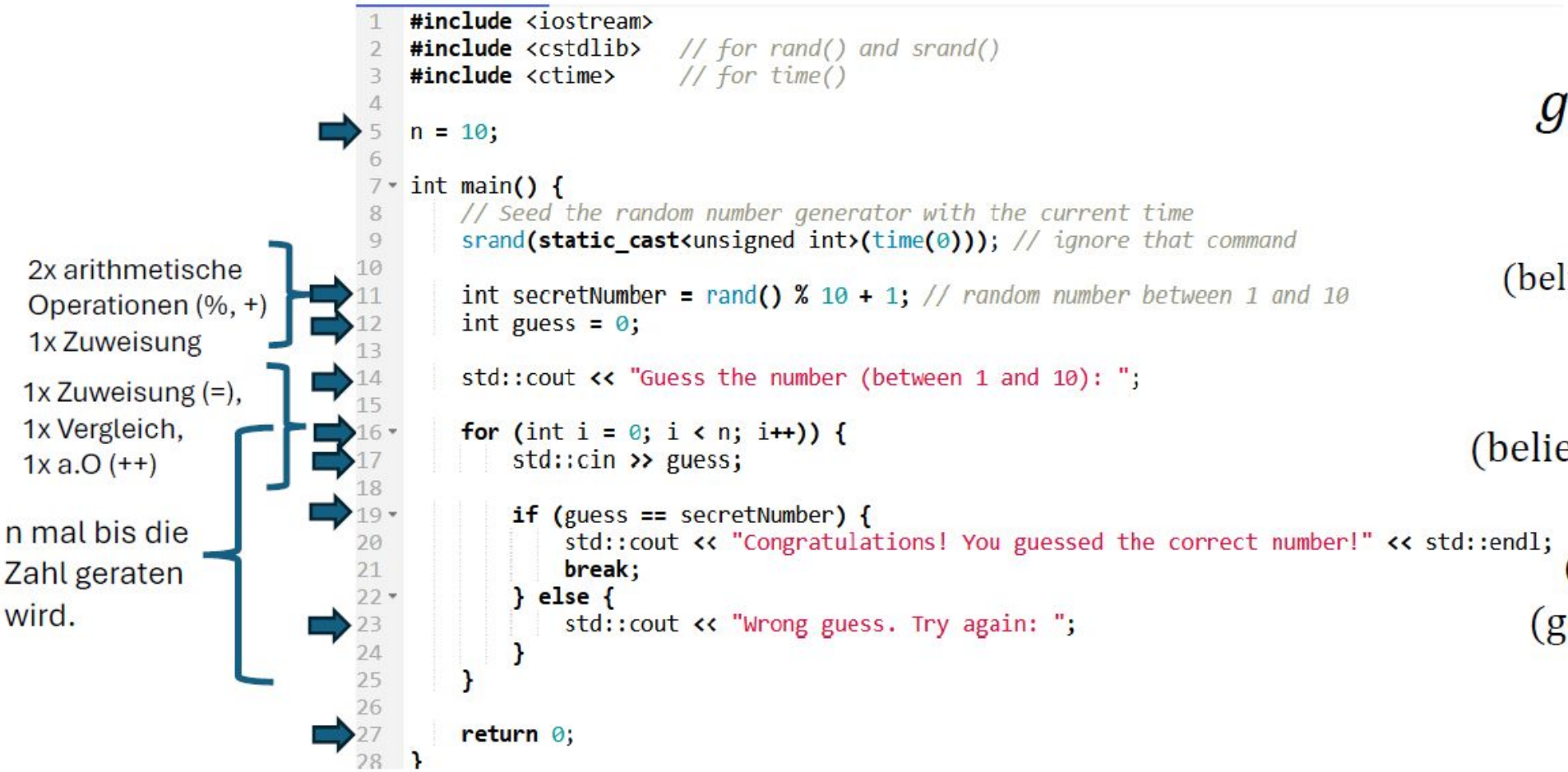


Untere Schranke Ω



Exakte Schranke Θ
→ kleinste mögliche $\mathcal{O}(g)$.
→ grösste mögliche $\Omega(g)$

Asymptotische Laufzeiten Definitionen




```cpp
#include <iostream>
#include <cstdlib>   // for rand() and srand()
#include <ctime>     // for time()

n = 10;

int main() {
    // Seed the random number generator with the current time
    srand(static_cast<unsigned int>(time(0))); // ignore that command

    int secretNumber = rand() % 10 + 1; // random number between 1 and 10
    int guess = 0;

    std::cout << "Guess the number (between 1 and 10): ";

    for (int i = 0; i < n; i++)) {
        std::cin >> guess;

        if (guess == secretNumber) {
            std::cout << "Congratulations! You guessed the correct number!" << std::endl;
            break;
        } else {
            std::cout << "Wrong guess. Try again: ";
        }
    }

    return 0;
}
```

$$g(n) = 6n + 7$$

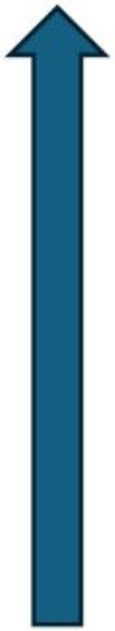
$\mathcal{O}(g(n)) = 7\mathrm{n}$
(beliebige obere Schranke)

$\Omega(g(n)) = 7$
(beliebige untere Schranke

$\Theta(g(n)) = 6n + 7$
(genau die Schranke)

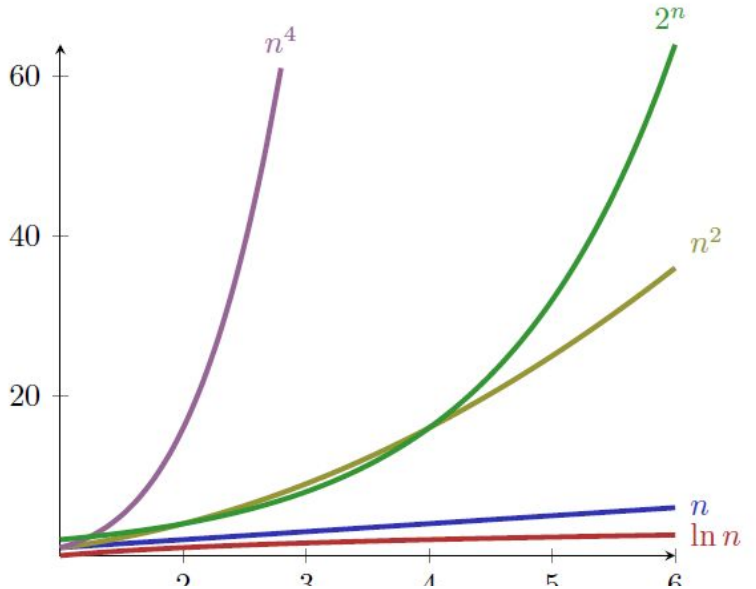
Wachstumsbezeichnungen:

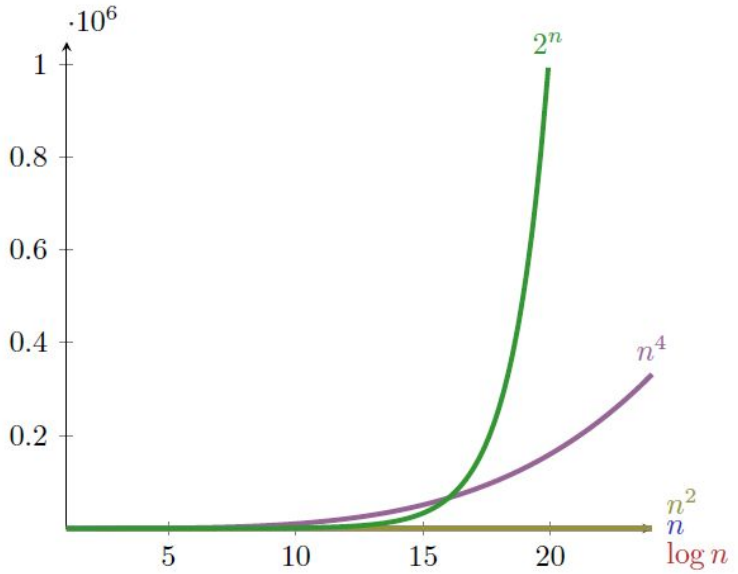
best



worst

$\Theta(1)$	beschränkt
$\Theta(\log\log n)$	doppelt logarithmisch
$\Theta(\log n)$	logarithmisch
$\Theta(\sqrt{n})$	wie die Wurzelfunktion
$\Theta(n)$	linear
$\Theta(n\log n)$	superlinear / loglinear
$\Theta(n^2)$	quadratisch
$\Theta(n^c)$	polynomiell
$\Theta(c^n)$	exponentiell
$\Theta(n!)$	faktoriell





$$f(n) = 6n + 7 \Rightarrow \Theta(f(n)) = n$$
$$f(n) = 5n + 7n * n^{1/2} \Rightarrow \Theta(f(n)) = ?$$
$$f(n) = n * \log(n) + \log(n) \Rightarrow \Theta(f(n)) = ?$$

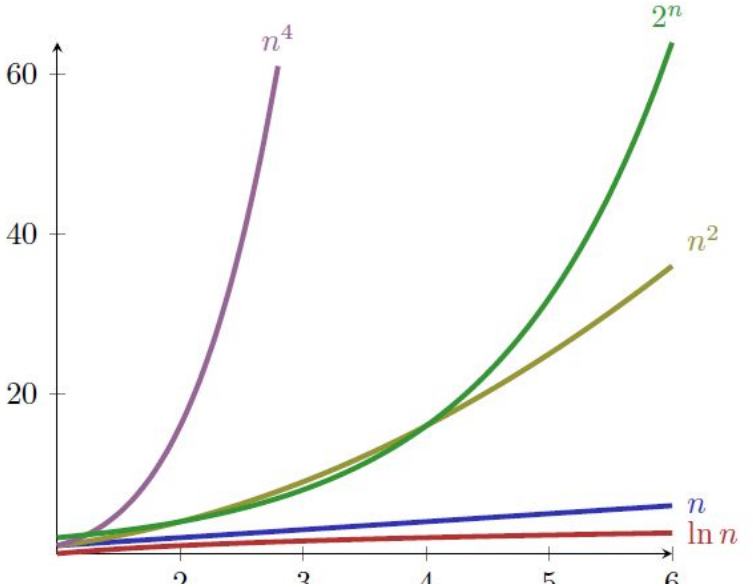
→ grössten Faktor betrachten und Vorfaktoren weglassen

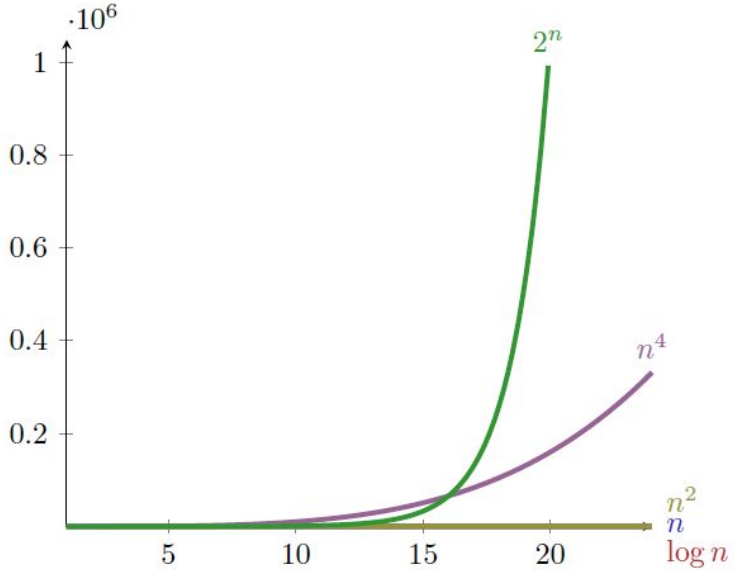
Wachstumsbezeichnungen:

best

worst

$\Theta(1)$	beschränkt
$\Theta(\log\log n)$	doppelt logarithmisch
$\Theta(\log n)$	logarithmisch
$\Theta(\sqrt{n})$	wie die Wurzelfunktion
$\Theta(n)$	linear
$\Theta(n\log n)$	superlinear / loglinear
$\Theta(n^2)$	quadratisch
$\Theta(n^c)$	polynomiell
$\Theta(c^n)$	exponentiell
$\Theta(n!)$	faktoriell





$$f(n) = 6n + 6 \Rightarrow \Theta(f(n)) = n$$
$$f(n) = 5n + 7n * n^{1/2} \Rightarrow \Theta(f(n)) = n^{3/2}$$
$$f(n) = n * \log(n) + \log(n) \Rightarrow \Theta(f(n)) = n * \log(n)$$

→ grössten Faktor betrachten und Vorfaktoren weglassen

Fortgeschrittenes Beispiel:

```cpp
#include <iostream>

n = 10;

int main() {

  for(int i = 0; i < n; i++){
    for(int j = 0; j < i; j++){
      std::cout << "moin";
    }
  }
    return 0;
}
```

Wert i	Wie oft “moin“ ausgerufen?
0	0
1	1
2	2
3	3
...	...
10	10

Also $g(n) = 1 + 2 + 3 + 4 + \ ... \ + n$, wie können wir das vereinfachen?

Fortgeschrittenes Beispiel:

```cpp
#include <iostream>

n = 10;

int main() {

  for(int i = 0; i < n; i++){
    for(int j = 0; j < i; j++){
      std::cout << "moin";
    }
  }
    return 0;
}
```

Wert i	Wie oft "moin" ausgerufen?
0	0
1	1
2	2
3	3
...	...
10	10

Also $g(n) = 1 + 2 + 3 + 4 + \ ... \ + n$, wie können wir das vereinfachen?

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$$

$$1 + 2 + 3 + 4 + \ ... \ + n = 10 * \frac{10+1}{2} = 55$$

$$\Theta\big(g(n)\big) = n^2$$

Weitere nützliche Summenformel:

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$$

$$\sum_{i=0}^{n} (n-i) = \sum_{i=0}^{n} i$$

$$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Logarithmus-Rechenregeln:

$$\log_a x^y = y \log_a x$$

$$\log_a n! = \sum_{i=1}^{n} \log i$$

$$a^{\log_b x} = x^{\log_b a}$$

Weitere Rechenregel:

Theorem 1

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g), \mathcal{O}(f) \subsetneq \mathcal{O}(g)$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C > 0$ (C konstant) $\Rightarrow f \in \Theta(g)$
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g \in \mathcal{O}(f), \mathcal{O}(g) \subsetneq \mathcal{O}(f)$

Beispiel 1

1. $\lim_{n\to\infty} \frac{n}{n^2} = 0 \Rightarrow n \in \mathcal{O}(n^2), \mathcal{O}(\mathrm{n}) \subsetneq \mathcal{O}(n^2)$
2. $\lim_{n\to\infty} \frac{2\mathrm{n}}{\mathrm{n}} = 2 > 0 \Rightarrow 2n \in \Theta(\mathrm{n})$
3. $\lim_{n\to\infty} \frac{n^2}{\mathrm{n}} = \infty \Rightarrow n \in \mathcal{O}(n^2), \mathcal{O}(\mathrm{n}) \subsetneq \mathcal{O}(n^2)$

Theorem 2

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow g \in \Omega(f), \Omega(g) \subsetneq \Omega(f)$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f \in \Omega(g), \Omega(f) \subsetneq \Omega(g)$

Beispiel 2

1. $\lim_{n\to\infty} \frac{\log(n)}{n\log(n)} = 0 \Rightarrow log(n) \in \Omega(n\log(n))$
2. $\lim_{n\to\infty} \frac{\mathrm{n}\log(n)}{\log(n)} = \infty \Rightarrow \mathrm{n}\log(n) \in \Omega(\log(n))$

Theorem 3

1. $f_1 \in \mathcal{O}(g), ; f_2 \in \mathcal{O}(g); \Rightarrow; f_1 + f_2 \in \mathcal{O}(g)$

Beispiel 3

1. $3n^2 \in \mathcal{O}(n^2), ; 5n \in \mathcal{O}(n^2); \Rightarrow; 3n^2 + 5\mathrm{n} \in \mathcal{O}(n^2)$

02 – Quiz

ETH zürich

$f(n)$	$f \in \mathcal{O}(?)$
$3n + 4$	
$2n$	
$n^2 + 100n$	
$n + \sqrt{n}$	

ETH zürich

$f(n)$	$f \in \mathcal{O}(?)$
$3n + 4$	$\mathcal{O}(n)$
$2n$	$\mathcal{O}(n)$
$n^2 + 100n$	$\mathcal{O}(n^2)$
$n + \sqrt{n}$	$\mathcal{O}(n)$

n wächst schneller als 4 (konstanter Faktor)

Faktor 2 weglassen

n^2 wächst schneller als 100n

n wächst Schneller als √n

$1 \in \mathcal{O}(15)$?

$2n + 1 \in \Theta(n)$?

$\sqrt{n} \in \mathcal{O}(n)$?

$\sqrt{n} \in \Omega(n)$?

$n \in \Omega(\sqrt{n})$?

ETH zürich

$1 \in \mathcal{O}(15)$?	✓	Ja, denn $\mathcal{O}(15)$ beinhaltet auch einen konstanten Faktor
$2n + 1 \in \Theta(n)$?	✓	Ja, denn $\Theta(2n + 1)$ kann zu $\Theta(n)$ vereinfacht werden (konstante Faktoren weglassen)
$\sqrt{n} \in \mathcal{O}(n)$?	✓	Ja, denn $\frac{\sqrt{n}}{n} \to 0$, siehe Theorem 1, Punkt 3., Slide 34
$\sqrt{n} \in \Omega(n)$?	✗	Nein, denn $\frac{n}{\sqrt{n}} \to \infty$, siehe, Theorem 2, Punkt 2.
$n \in \Omega(\sqrt{n})$?	✓	Ja, denn $\frac{\sqrt{n}}{n} \to 0$, siehe, Theorem 2, Punkt 1.

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = 1; j<n; ++j)
      op();
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = 1; j<n; ++j)
      op();
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

$$\sum_{i=1}^{n-1}\sum_{j=1}^{n-1} 1 = \sum_{i=1}^{n-1}(n-1) = (n-1)\cdot(n-1) \in \Theta(n^2)$$

```
void run(int n){
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j*j <= n; ++j)
      for(int k = n; k >= 2; --k)
        op();
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

```
void run(int n){
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j*j <= n; ++j)
      for(int k = n; k >= 2; --k)
        op();
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

$$\sum_{i=1}^{n} \sum_{j=1}^{\lfloor \sqrt{n} \rfloor} n - 1 \in \Theta\left(\sum_{i=1}^{n} n^{3/2} \right) = \Theta(\sqrt{n^5})$$

```
int f(int n){
    i=1;
    while (i <= n*n*n){
        i = i*2;
        op();
    }
    return i;
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

```
int f(int n){
    i=1;
    while (i <= n*n*n){
        i = i*2;
        op();
    }
    return i;
}
```

Wie oft wird `op()` in Abhängigkeit von n aufgerufen?

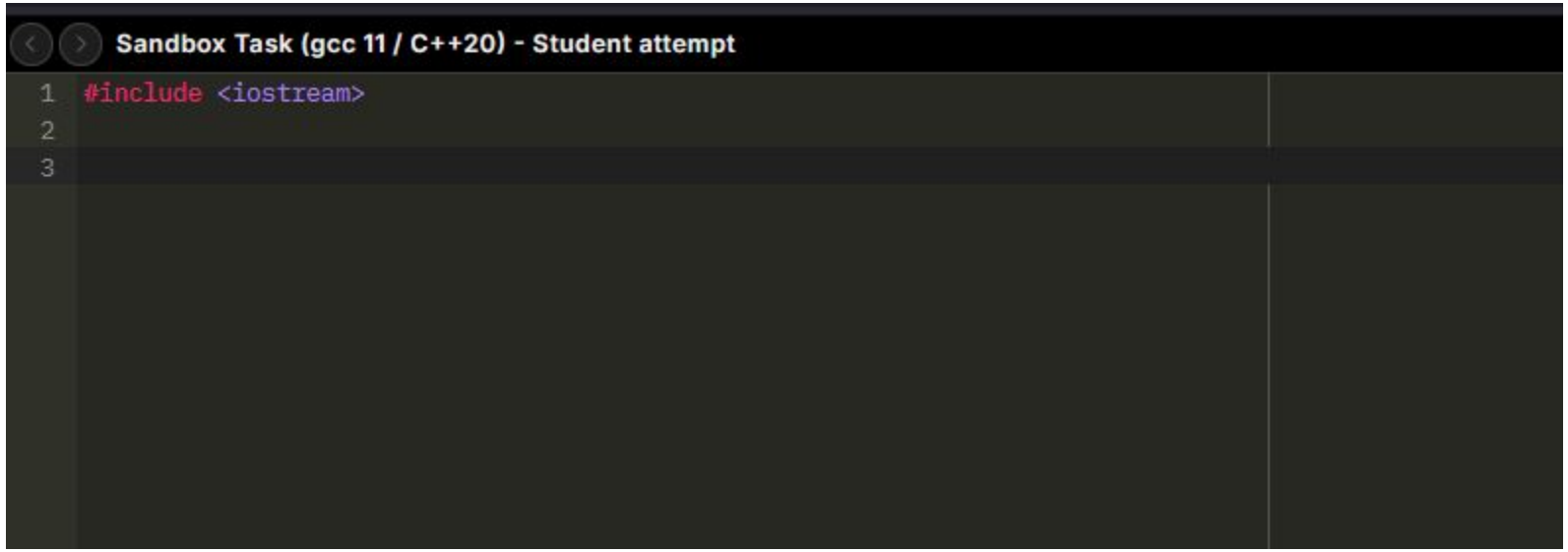
$$|\{k \in \mathbb{N} : 2^k \leq n^3\}| \in \Theta(\log_2 n^3) = \Theta(\log n)$$

03- Programmieraufgabe

ETH zürich

Programmieraufgabe

1. Geh auf CodeExpert → Code Examples → Sandbox Task
2. Versuch die Fibonnaci Folge schneller als in den Slides zu lösen



Code Example – Fibonacci Lösung

```cpp
//improved fibonacci
#include <iostream>

int main() {
    long long unsigned int n;
    std::cin >> n;

    long long int a = 0;
    long long int b = 1;

    for (unsigned int i = 0; i < n; i++) {
        long long int sum = a + b;
        a = b;
        b = sum;
    }

    std::cout << a;
    return 0;
}
```

```
100
3736710778780434371
```

ETH zürich

Michal Oskedra
Teaching Assistent D-ITET
moskedra@ethz.ch