

# Komplexität von Algorithmen

## Sonderübung

Nadine Schärmann & Thomas Klütz

PROG

21. Juni 2021

1. Komplexität - Begriffe

2. Beispiele

# Begriffe

- Komplexität von Algorithmen meint deren Effizienz
- nicht mit effektiv zu verwechseln
- Effektivität = löst der Algorithmus das Problem?
- Effizienz = wie einfach/schnell wird das Problem gelöst?
- Unterteilung in Laufzeit- und Speicherkomplexität

# Laufzeitkomplexität

Laufzeitkomplexität wird üblicherweise mit  $T(n)$  angegeben. Es beschreibt quasi wie schnell der Algorithmus ein Problem zeitlich lösen kann, in Abhängigkeit von der Größe der Eingabedaten  $n$ .

- worst case  $T_W$
- average case  $T_A$
- best case  $T_B$

# Speicherkomplexität

Speicherkomplexität,  $S(n)$  beschreibt wie viel Speicher (im RAM) benötigt wird, um das Problem zu lösen.

Laufzeiteffizienz kostet Speicher

Speichereffizienz kostet Laufzeit

# Definition

Komplexität ist in der O-Notation üblich:

- $O(f(n)) = \{g(n) \mid \exists c > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 : 0 \leq g(n) \leq c \cdot f(n)\}$
- $\Omega(f(n)) = \{g(n) \mid \exists c > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 : 0 \leq c \cdot f(n) \leq g(n)\}$
- $\Theta(f(n)) = \{g(n) \mid \exists c_1, c_2 > 0 \exists n_0 \in \mathbb{N} \forall n > n_0 : c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)\}$

Welche Funktionen (Polynome) sind in ... ?

- $O(x^n)$
- $\Omega(x^n)$
- $\Theta(x^n)$

# Lineare Suche

Bestimme die Position vom Wert  $x$  in einem Vektor.

Von der ersten Position Vektor durchgehen, bis  $x$  gefunden ist.

```
position = 0
DO i = 1,SIZE(v)
    IF(v(i)==x)THEN
        position = i
        EXIT
    END IF
END DO
```

# Fibonacci

Programmiert die Fibonacci-Folge iterativ und rekursiv. Wie verhalten sich die Laufzeit und der Speicherbedarf jeweils?

Wie lässt sich die Folge direkt berechnen?



# Ackermann-Funktion

Implementiert die Ackermann-Funktion und testet sie für kleine Eingaben.

$$a(m, n) = \begin{cases} n + 1, & m = 0, n \geq 0 \\ a(m - 1, 1), & n = 0, m \geq 1 \\ a(m - 1, a(m, n - 1)), & m, n \geq 1 \end{cases}$$

Einmal rekursiv, wie in der Definition und einmal iterativ mit Stacks. Dafür wird wieder das `stackmod.mod` verwendet. Hier muss der Stack allerdings genau entgegen gesetzt aufgebaut werden (erst die Operanden, dann die Operation "a"). Es bietet sich an einen INTEGER Stack zu verwenden.