# Rekursion

Rätsel: Sei $a_0 \in \mathbb{N}_0$ beliebig. Betrachte die Folge

$$a_{n+1} := \begin{cases} a_n/2, & \text{falls } a_n \text{ gerade} \\ 3a_n+1, & \text{sonst} \end{cases}$$

Die Folge bricht ab, falls $a_n = 1$.
Erreicht die Folge für alle $a_0$ für ein $n \in \mathbb{N}$ $a_n = 1$?

```
public int ulam(int start){
    if (start == 1)
        return 0;
    if (start % 2 == 0)
        return 1 + ulam(start/2);
    else
        return 1 + ulam(3*start+1);
}
```

# Ackermann-Funktion

- steigt stark an
- verallgemeinert Summe, Produkt, Potenz, ...

$$3+3+3 = 3*3$$
$$3*3*3 = 3^3$$
$$3^{3^3} = \overline{\text{Tower}}(3)$$
$$\vdots$$

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

Beispiel:

$$\begin{aligned} a(1,3) &= a(0, a(1,2)) \\ &= a(0, a(0, a(1,1))) \\ &= a(0, a(0, a(0, \underbrace{a(1,0)}_{a(0,1)=2}))) \\ &= a(0, a(0, a(0,2))) \end{aligned}$$

$= a(0, a(0,3)) = a(0,4) = 5$

---

Für alle $n \in \mathbb{N}$:

$a(0,n) > n$

$a(1,n) > n+1$

$a(2,n) > 2n$

$a(3,n) > 2^n$

$a(4,n) > 2^{2^{2^{\cdots}}}$ ($n$-mal)

$\vdots$

$a(5,4) > 10^{10^{10^{\cdots}}}$ ....

Satz: Für alle $m,n \in \mathbb{N}$ terminiert der Aufruf $a(m,n)$ nach endlich vielen Schritten.

Bew: durch Doppelinduktion

Induktionsanfang $m=0$:

$a(0,n) := n+1$

terminiert also mit nur einem Aufruf.

Induktionsschritt für $m$:
Die Behauptung sei richtig für alle $k \in \{0, \dots, m-1\}$ und alle $n \in \mathbb{N}$.

Betrachte nun $a(m, n)$, $n \in \mathbb{N}$ beliebig.

Induktionsanfang für $n$:

$a(m, 0) := \underbrace{a(m-1, 1)}$

terminiert nach Induktionsvoraussetzung für $m$

Induktionsvoraussetzung für $n$:
$a(m, l)$ terminiere für alle $l < n$.

Betrachte nun

$$a(m, n) = a(m-1, \underbrace{a(m, n-1)})$$

terminiert nach IV an $n$

$\underbrace{\qquad\qquad}$

terminiert nach IV an $m$.

□

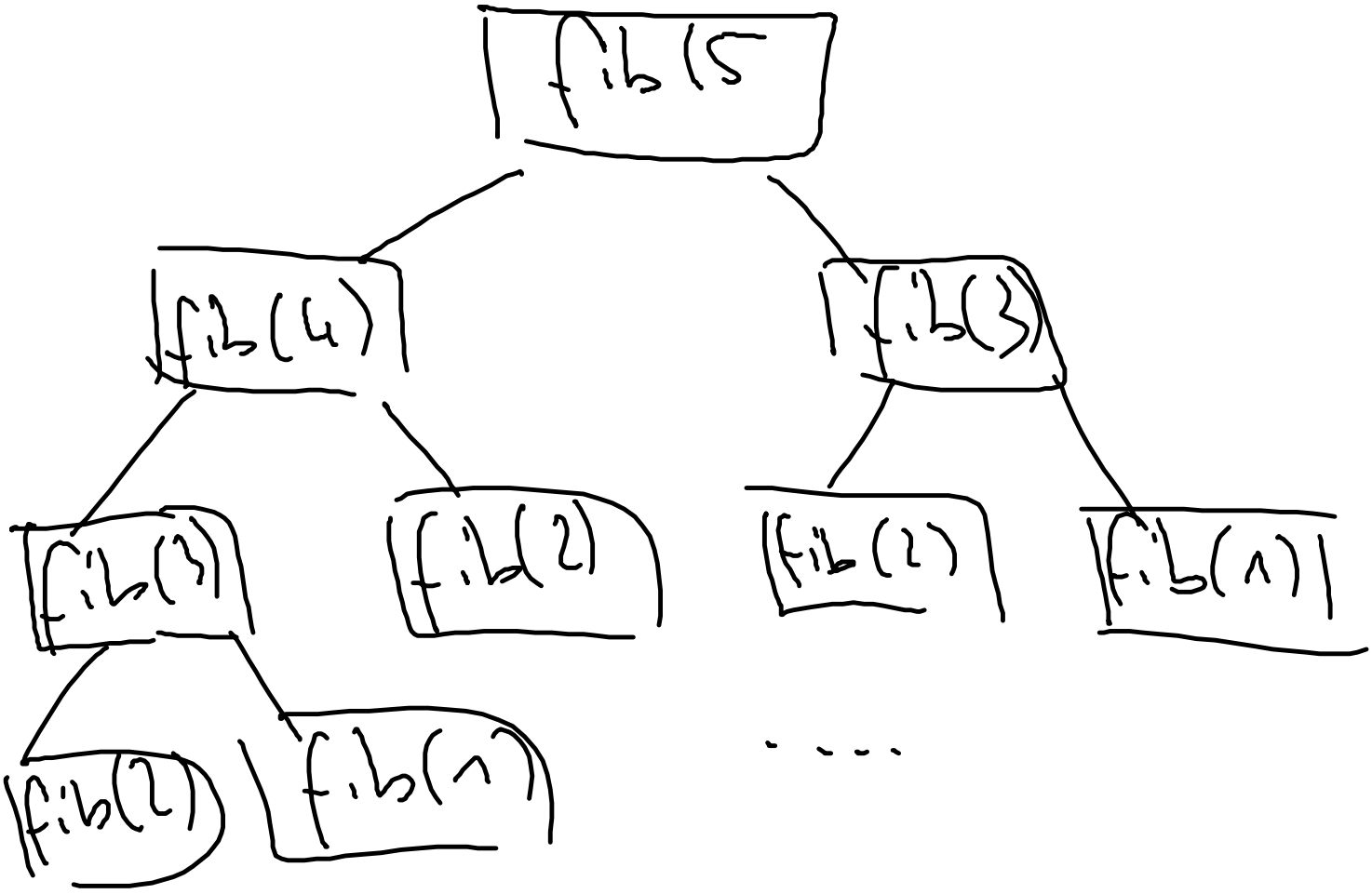
# Fibonacci-Zahlen

Die Folge der Fibonacci-Zahlen ist definiert durch

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \text{für } n \geq 2.$$

## rekursiv

```
public int fib(int n){
    if (n==0)
        return 0;
    if (n==1)
        return 1;
    return fib(n-1) + fib(n-2);
}
```




Satz: $T_{fib}(n)$ wächst mindestens exponentiell.

Bew.: Wir zeigen $T_{fib}(n) \geq 2^{\lfloor n/2 \rfloor}$ für $n \geq 2$.

IA: $n=0$: $T_{fib}(0) = 1 = 2^{\lfloor 0 \rfloor}$ ✓

$n=1$: $T_{fib}(1) = 1 = 2^{\lfloor 1/2 \rfloor}$ ✓

IS: $n \mapsto n+2$

$$T_{fib}(n+1) = 1 + \underbrace{T_{fib}(n)}_{\geq T_{fib}(n-1)} + T_{fib}(n-1)$$

$$\geq 2 \cdot T_{fib}(n-1)$$

$$\overset{IV}{\geq} 2 \cdot 2^{\lfloor (n-1)/2 \rfloor}$$

$$= 2^{\lfloor (n+1)/2 \rfloor}$$

□

iterativ

```
public int fib(int n) {
    if (n==0)
        return 0;
    if (n==1)
        return 1;
```

```
    int currentFib = 1, previousFib = 0;
    for (int i = 1; i < n; ++i) {
        currentFib = currentFib + previousFib;
        previousFib = currentFib - previousFib;
    }
    return currentFib;
}
```

explizit

$$f_n = \frac{\varphi^n - \psi^n}{\sqrt{5}}, \text{ wobei}$$

$\varphi, \psi$ lösen die Gleichung $x^2 - x - 1 = 0$

$$x_{1/2} = +\frac{1}{2} \pm \sqrt{\frac{1}{4} + 1}$$

$$= \frac{1}{2} \pm \sqrt{\frac{5}{4}} = \frac{1 \pm \sqrt{5}}{2}$$

Bew:

IA: $n=0: f_0 = 0 = \frac{\varphi^0 - \psi^0}{\sqrt{5}} = 0$ ✓

$n=1: f_1 = 1 = \frac{\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}}{\sqrt{5}} = 1$ ✓

$n \mapsto n+1$

$$f_{n+1} = f_n + f_{n-1}$$

$$\text{IV} = \frac{\varphi^n - \psi^n + \varphi^{n-1} - \psi^{n-1}}{\sqrt{5}}$$

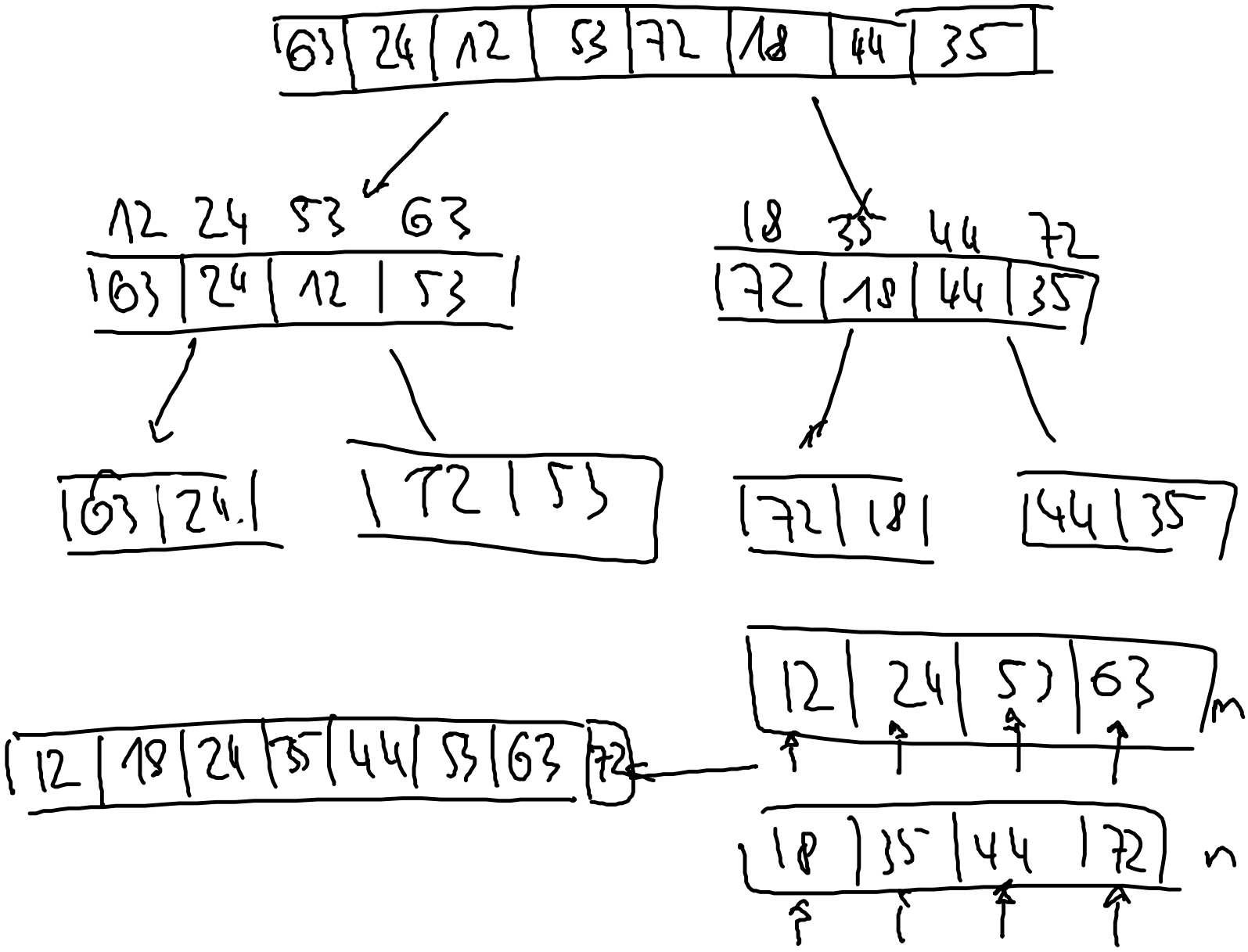
$$= \frac{\varphi^n\left(1+\frac{1}{\varphi}\right) + \psi^n\left(1+\frac{1}{\psi}\right)}{\sqrt{5}}$$

$$= \frac{\varphi^{n+1} - \psi^{n+1}}{\sqrt{5}}$$

□

## Rekursive Sortierverfahren

### Mergesort




Beobachtung: Der Aufwand beim Mergen (Zusammenfügen) zweier sortierter Arrays mit Länge $m, n \in \mathbb{N}$ ist

$$C(m,n) = m+n-1.$$

## Analyse von Mergesort

O.B.d.A. sei $n$ eine Potenz der 2.

$$C(2) = 1$$

$$C(n) = 2 \cdot C(n/2) + \underbrace{C(n,n)}_{2n-1}$$

---

ins Unreine

Nebenüberlegung: $n = 2^q$

$$C(n) = 2C(n/2) + 2(n/2) - 1$$
$$= 2C(n/2) + n - 1$$
$$= 4C(n/4) + 2n - 3$$
$$\dots$$
$$= 8C(n/8) + 3n - 7$$
$$\dots$$
$$= 2^{q-1} C(2) + (q-1)n - (2^{q-1} - 1)$$

**Lemma**: Es gilt für $n = 2^q$, dass

$$C(n) = (q-1)2^q + 1.$$

**IA**: $q = 1 \Rightarrow n = 2$,

$$C(2) = 1 = (q-1)2^q + 1. \quad \checkmark$$

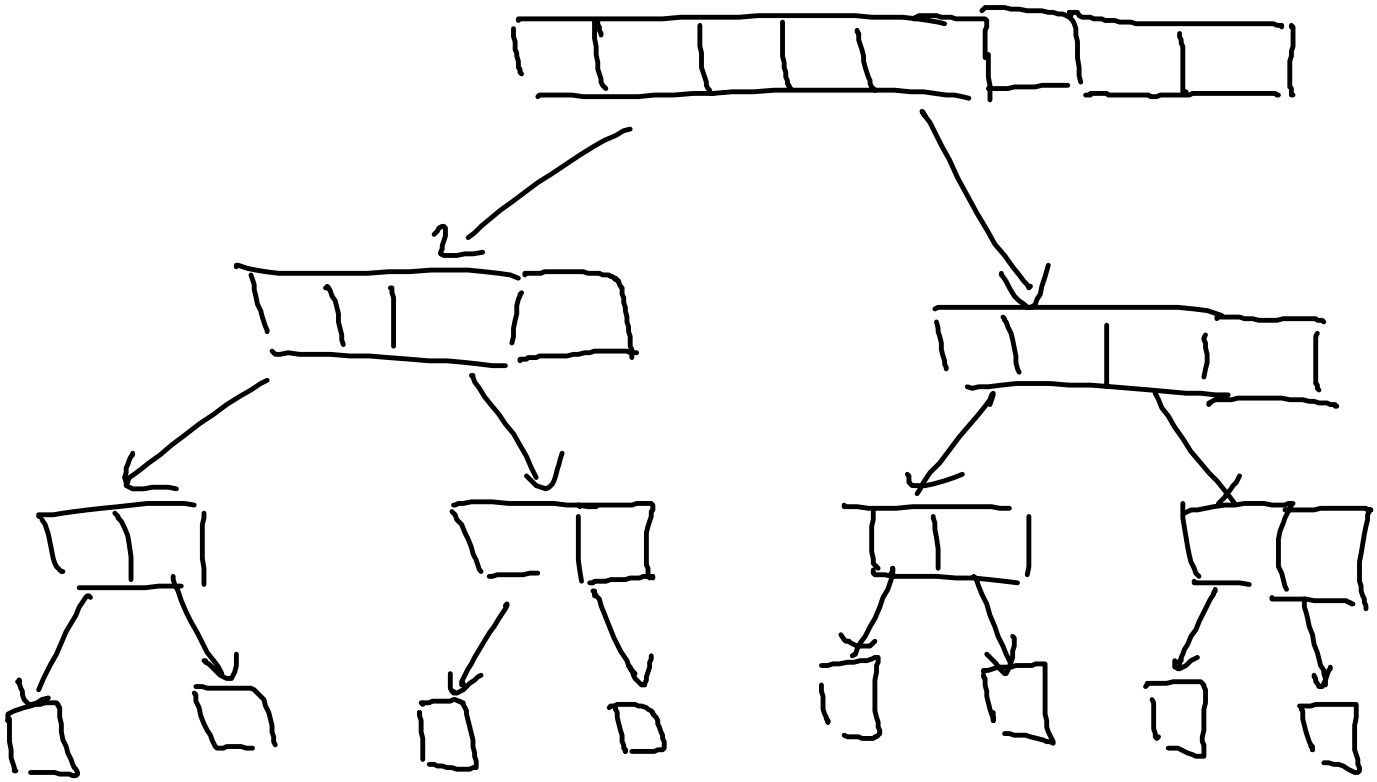
**IS**: $q \mapsto q+1$

$$C(2^{q+1}) = 2\underbrace{C(2^q)}_{\text{IV}} + 2 \cdot 2^q - 1$$

$$= 2 \cdot \left((q-1)2^q + 1\right) + 2^{q+1} - 1$$

$$= q \cdot 2^{q+1} + 1 \qquad \square$$

Was passiert, falls $n$ keine Zweierpotenz ist?



→ mit der Bekannten Analyse gilt

$$\mathcal{O}(n' \log_2 n'), \text{ wobei}$$

$$n' = \min\{2^r : r \in \mathbb{N}, 2^r \geq n\}$$

$$n \geq n'/2$$

$$\log_2 2 + \log_2 n$$

$$\mathcal{O}(n' \log_2 n') \leq \mathcal{O}(2n \log_2(2n))$$

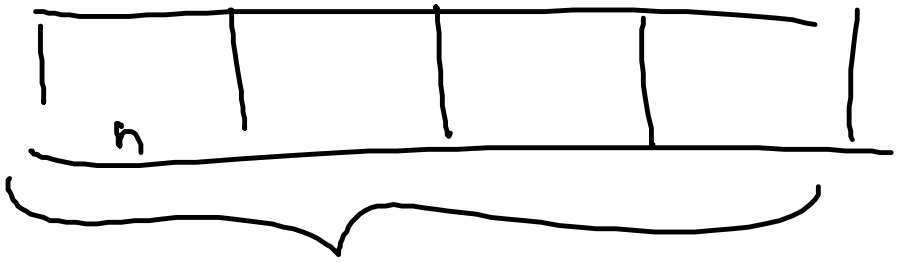
$$= \mathcal{O}(n \log_2 n)$$

# Aufteilungs-Beschleunigungssätze
(Master-Theorem für Divide & Conquer)

Gegeben ein Problem der Größe $a \cdot n$, $a \in \mathbb{N}$.

Idee: Zerlege Problem in $b$ Teilprobleme der Größe $n$

Benötige $c \cdot n$ Aufwand zum Zusammenfügen der Teillösungen





Zusammensetzen $c \cdot n$

**Satz** Seien $a, b, c \in \mathbb{N}$ und gelte

$$f(1) \leq \frac{c}{a}$$

$$f(a \cdot n) \leq b \cdot f(n) + c \cdot n \quad \text{für } n > 1.$$

[Mergesort $a = b = 2$, $c = 2$]

Dann gilt:

$$f(n) \in \begin{cases} \mathcal{O}(n), & \text{falls } a > b \\ \mathcal{O}(n \log n), & \text{falls } a = b \\ \mathcal{O}(n^{\log_a b}), & \text{falls } a < b \end{cases}$$