§7: Algorithmen und Komplexität

"Def:" Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen. Algorithmen bestehen aus endlich vielen, wohldefinierten Einzelschritten. Damit können sie zur Ausführung in ein Computerprogramm implementiert, aber auch in menschlicher Sprache formuliert werden. Bei der Problemlösung wird eine bestimmte Eingabe in eine bestimmte Ausgabe überführt. [Eingabe = Input, Ausgabe = Output]

Beispiele:

1) Die Addition zweier natürlicher Zahlen.

```
  1 4 7 8 2 5 3
    8 6 5 1 9 2
  1 1 1   1
-------------------
  2 3 4 3 4 4 5
```

2) Die Multiplikation zweier natürlicher Zahlen

```
    4 1 2 8
      3 1 5
-------------
  2 0 6 4 0
  4 1 2 8 0
1 2 3 8 4 0 0
```

```
      2 0 6 4 0
      4 1 2 8 0   +
1 2 3 8 4 0 0
    1 1 1 1
-----------------
1 3 0 0 3 2 0
```

3) Die Division mit Rest zweier natürlicher Zahlen.

$$8250 : 28 = 294$$

56

2650

252

130

112

18

Also $8250 = \underbrace{294}_{\text{Quotient}} \cdot 28 + \underbrace{18}_{\text{Rest}}$.

4) Der ggT zweier natürlicher Zahlen.

$ggT(519, 42) = ?$

$519 = 12 \cdot 42 + 15$

$42 = 2 \cdot 15 + 12$

$15 = 1 \cdot 12 + 3$

$12 = 4 \cdot 3 + 0.$

$ggT(519, 42) = 3.$

5) Ist eine gegebene natürliche Zahl n prim?

Algorithmus: Für alle natürliche Zahlen $m \leq \sqrt{n}$ untersuchen, ob $m | n$.

6) Hat ein Polynom $f(x) \in \mathbb{Z}[x]$ eine rationale Nullstelle?

Sei $f(x) = a_n x^n + \dots + a_1 x + a_0$ mit $a_i \in \mathbb{Z}$ $\forall i \leq n$, und $a_n \neq 0$.

Bekannt (s. §5, S. 1-2): Wenn $\alpha = \frac{r}{s}$ mit $r \in \mathbb{Z}$, $s \in \mathbb{N}^*$, $ggT(r, s) = 1$,

dann gilt: $f(\frac{r}{s}) = 0 \Rightarrow (s|a_n \text{ und } r|a_0)$.

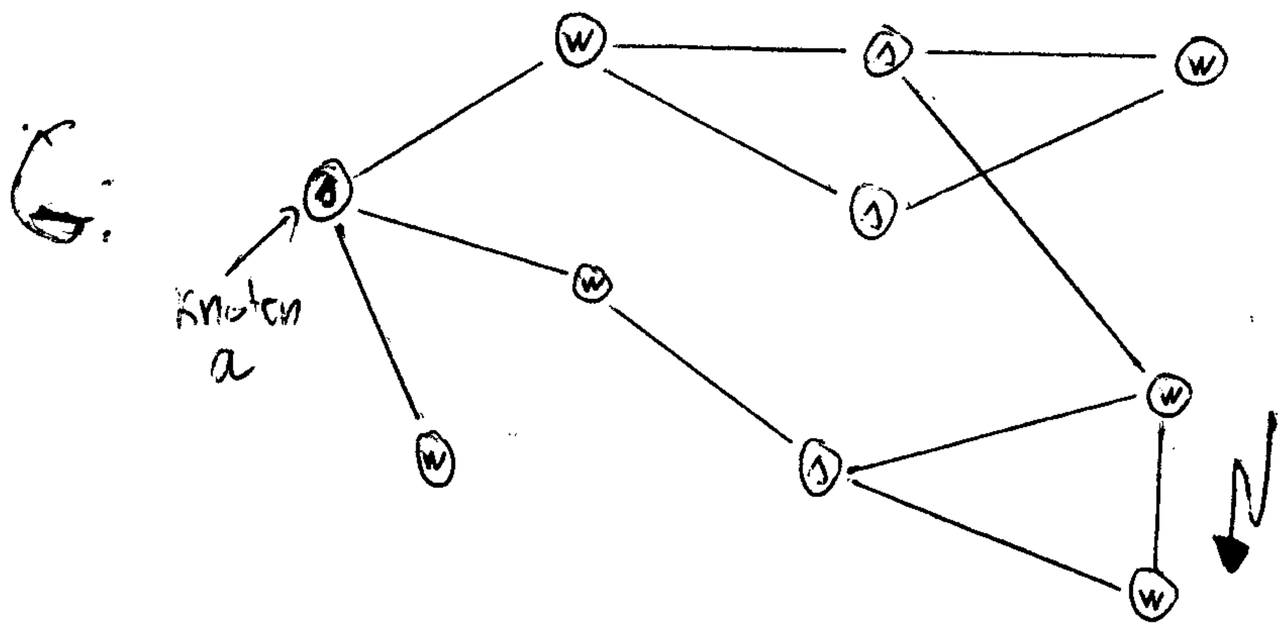
Algorithmus: Wir betrachten alle rationalen Zahlen der Form $\alpha = r/s$ mit $r \in \mathbb{Z}, s \in \mathbb{N}^*, ggT(r,s) = 1, s|a_n$ und $r|a_0$. [Es gibt nur endlich viele solche Zahlen $\alpha = r/s$].

Für all diese rationalen Zahlen α berechnen wir $f(\alpha)$.

- Wenn $f(\alpha) = 0$ für ein solches α, dann hat f eine rationale Nullstelle.
- Wenn nicht, dann nicht.

7) Ist ein gegebener ungerichteter Graph $G = (V, E)$ 2-färbbar?

G:



Algorithmus: Wir starten in irgendeinem Knoten a und geben diesem die Farbe s (schwarz). Alle Nachbarknoten von a müssen anders gefärbt werden, erhalten also die Farbe w (weiß). Anschließend färben wir alle Nachbarknoten der Nachbarknoten von a mit Farbe s. Falls einer dieser Knoten aber vorher die Farbe w erhalten hat, dann ist der Graph G nicht 2-färbbar. Wir fahren auf analoge Weise bei den Nachbarknoten der Nachbarknoten der Nachbarknoten von a fort.

Komplexität von Algorithmen

Die Anzahl der "Schritte", die ein Algorithmus benötigt, wird als die Laufzeit des Algorithmus bezeichnet.

Zu "Schritt": Ein Mensch (oder eine Maschine) muss in der Lage sein, einen einzelnen Schritt in Konstanter Zeit auszuführen.

Die Laufzeit hängt dann im Allgemeinen von der Eingabe ab, insbesondere von der Länge der Eingabe.

Def: Die Laufzeit $T: \mathbb{N} \to \mathbb{N}$ wird definiert durch:

$T(n) :=$ maximale Anzahl der Schritte, die der Algorithmus bei Eingaben der Länge n braucht.

Def: (die O-Notation)

Seien $f, g: \mathbb{N} \to [0, \infty[$.

$f(n) \in O(g(n)) :\Leftrightarrow \exists m \in \mathbb{N}\ \exists K > 0$ mit $f(n) \leq K \cdot g(n)\ \forall n \geq m$.

[f wächst nicht schneller als g wenn $n \to \infty$]

Bem: Für $f(n) \in O(g(n))$ schreibt man auch $f(n) = O(g(n))$.

[⚠ $O(g(n)) = f(n)$ sinnlos]

Beispiele:

i) $2n^3 + 5n^2 + 10n + 20 \in O(n^3)$, $\notin O(n^2)$

ii) $1000n^2 \in O(0{,}001 \cdot n^3)$

iii) $n^{1000} \in O(2^n)$

iv) $n^{\ln(n)} = O((\ln(n))^n)$ (s. Üb)

Laufzeit von ein paar Algorithmen

1) Die Addition $a+b$ mit $a, b \in \mathbb{N}$

Sei $n \in \mathbb{N}$ und $a, b \in \mathbb{N}$ höchstens n-stellig ($\Rightarrow a, b < 10^n$)

$a = x_n x_{n-1} \dots x_2 x_1$
$b = y_n y_{n-1} \dots y_2 y_1$
mit $x_i, y_i \in \{0, 1, 2, \dots, 9\}$ $\forall i \leq n$.

Ein Schritt ist hier eine Addition $x+y$ mit $x, y \in \{0, 1, 2, \dots, 9\}$.
Der Grundschulalgorithmus (s. S. 1) liefert $T(n) \leq 2n$.
$T(n)$ ist linear.

2) Die Multiplikation ab mit $a, b \in \mathbb{N}$

Sei $n \in \mathbb{N}$ und $a, b \in \mathbb{N}$ höchstens n-stellig.

$a = x_n x_{n-1} \dots x_2 x_1$
$b = y_n y_{n-1} \dots y_2 y_1$
mit $x_i, y_i \in \{0, 1, 2, \dots, 9\}$ $\forall i \leq n$.

Ein Schritt ist hier eine Addition oder Multiplikation von zwei Ziffern $x, y \in \{0, 1, 2, \dots, 9\}$. Die Laufzeit $T(n)$ des Grundschulalgorithmus (s. S. 1) ist $T(n) = O(n^2)$. $T(n)$ ist quadratisch.

Satz 7.1: Für die Multiplikation zweier natürlicher Zahlen gibt es Algorithmen mit Laufzeit

a) $O(n^{\log_2 3}) \approx O(n^{1,59})$ [Karatsuba, 1962]
b) $O(n \log n (\log \log n))$ [Schönhage, Strassen, 1971] $\log := \log_2$
c) $O(n \log n)$ [Harvey, van der Hoeven, 2019].

□

3) Die Division $a:b$ mit $a, b \in \mathbb{N}^*$

Wie für die Multiplikation hat der Grundschulalgorithmus Laufzeit $T(n) = O(n^2)$.

4) Der euklidische Algorithmus für $ggT(a,b)$; $a,b \in \mathbb{N}^*$

$$a = q_1 b + r_1$$
$$b = q_2 r_1 + r_2$$
$$r_1 = q_3 r_2 + r_3$$
$$\vdots$$
$$r_{k-3} = q_{k-1} r_{k-2} + r_{k-1}$$
$$r_{k-2} = q_k r_{k-1} + r_k$$
$$\| \; 0$$

Wir brauchen zunächst eine Abschätzung für die Anzahl k der Divisionen mit Rest:

$$k = k(a,b) \leq ?$$

Bem: Als sehr grobe Abschätzung gilt $k \leq b$ denn $b > r_1 > r_2 > \dots > r_k = 0$.

Für den Spezialfall zweier aufeinander folgenden Fibonacci-Zahlen $(a,b) = (f_k, f_{k-1})$ haben wir

↑ s. Skript, §4, S. 3-4

$$f_k = 1 \cdot f_{k-1} + f_{k-2}$$
$$f_{k-1} = 1 \cdot f_{k-2} + f_{k-3}$$
$$f_{k-2} = 1 \cdot f_{k-3} + f_{k-4}$$
$$\vdots$$
$$f_4 = 1 \cdot f_3 + f_2$$
$$f_3 = 2 \cdot f_2 + 0.$$

Die Fibonacci-Folge (f_n):

$f_0 := 0$, $f_1 := 1$, $f_n := f_{n-1} + f_{n-2}$ $\forall n \geq 2$.

$(f_n) = (0, 1, 1, 2, 3, 5, 8, 13, 21, \dots)$

$$f_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right) \quad \forall n \in \mathbb{N}$$

Für $(a,b) = (f_k, f_{k-1})$ (den schlimmsten Fall) braucht es also $k-2$ Divisionen mit Rest.

Lemma 7.2: Seien $a, b \in \mathbb{N}$ mit $a > b \geq 1$.
Benötigt der euklidische Algorithmus zur Berechnung von $ggT(a,b)$ genau k Divisionen mit Rest, dann gilt

$$b \geq f_{k+1} \text{ und } a \geq f_{k+2}.$$

Beweis: Induktion über k.

$k=1$: $b \geq 1 = f_2$ und $a > b \Rightarrow a \geq 2 = f_3$.

$k \to k+1$:

$$\begin{aligned} a &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{k-2} &= q_k r_{k-1} + r_k \\ r_{k-1} &= q_{k+1} r_k + r_{k+1}. \\ &\quad\| \\ &\quad 0 \end{aligned}$$

Genau $k+1$ Divisionen für $ggT(a,b)$
$\Rightarrow$ genau k Divisionen für $ggT(b, r_1)$.
Nach I.V für $b > r_1 \geq 1$ gilt
$r_1 \geq f_{k+1}$ und $b \geq f_{k+2}$.
$a > b \Rightarrow q_1 \geq 1 \Rightarrow a = q_1 b + r_1 \geq b + r_1 \geq f_{k+2} + f_{k+1}$
$= f_{k+3}$.
Also $b \geq f_{k+2}$ und $a \geq f_{k+3}$. □

Korollar 7.3: Seien $a, b \in \mathbb{N}$ mit $a > b \geq 1$ und a höchstens n-stellig.

a) Ist k die genaue Anzahl der Divisionen mit Rest im euklidischen Algorithmus zur Berechnung von $ggT(a,b)$, dann gilt $k < 6n$.
b) Die Laufzeit des euklidischen Algorithmus zur Berechnung von $ggT(a,b)$ ist $T(n) = O(n^3)$.

Beweis:

Zu a): a höchstens n-stellig $\Rightarrow a < 10^n$
Nach Lemma 7.2 erhält man $a \geq f_{k+2}$ $\Big\} \Rightarrow f_{k+2} \overset{(1)}{<} 10^n$.

Wir wollen eine Abschätzung nach unten von f_{k+2}:

$$f_m = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^m - \left(\frac{1-\sqrt{5}}{2}\right)^m\right) \geq \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^m\right) - 1 \geq \frac{1}{\sqrt{5}} 1{,}6^m - 1 \quad \forall m \in \mathbb{N}.$$

Daraus folgt $f_{6n} \geq \frac{1}{\sqrt{5}} 1{,}6^{6n} - 1 = \frac{1}{\sqrt{5}}(1{,}6^6)^n - 1 > \frac{1}{\sqrt{5}} 16^n - 1 \overset{(2)}{\underset{n \geq 2}{>}} 10^n$.

$(1)+(2) \Rightarrow f_{k+2} < 10^n < f_{6n} \Rightarrow f_{k+2} < f_{6n} \Rightarrow k+2 < 6n \Rightarrow k < 6n$.

↖ Die Fibonacci-Folge ist monoton steigend.

Zu b): Für jede einzelne Division mit Rest braucht es höchstens cn^2 Schritte ($c > 0$ Konstante). Insgesamt haben wir

$$T(n) \leq K \cdot (cn^2) < (6n)cn^2 = (6c)n^3 \Rightarrow T(n) = O(n^3).$$

5) Das Problem PRIM

PRIM ist das Problem: Entscheide für ein gegebenes $a \in \mathbb{N}^*$ ob a prim ist oder nicht.

Sei $a \in \mathbb{N}^*$, a höchstens n-stellig, ($\Rightarrow a < 10^n$).

Der naive Algorithmus (überprüfe ob $m \mid a$ für alle $m \leq \sqrt{a}$) hat Laufzeit $O(n^2 \cdot 10^{n/2}) \leq O(4^n)$.

Satz 7.4 (Agrawal, Kayal, Saxena, 2004)

Es gibt einen Algorithmus für PRIM mit Laufzeit $O(n^{11})$.

6) Das K-Färbbarkeitsproblem (K-Col) mit $K \geq 2$ ($K \in \mathbb{N}$)

Sei $G = (V, E)$ ein ungerichteter Graph mit $|V| = n$.
Ist G K-färbbar? [D.h. gibt es eine Abbildung $f: V \to \{1, 2, \dots, K\}$ mit $f(u) \neq f(v)$ $\forall \{u, v\} \in E$.]

Fall 1: $K = 2$.
Der gegebene Algorithmus zu 2-Col (s. S. 3) hat Laufzeit $O(n^4)$

Fall 2: $K \geq 3$
Sei $G = (V, E)$ mit $n = |V|$.
G hat insgesamt K^n zulässige oder unzulässige "Färbungen".
Für jede "Färbung" kann man in $O(n^2)$-Zeit überprüfen, ob die "Färbung" eine richtige Färbung ist oder nicht. Daher ein Algorithmus mit Laufzeit $O(n^2 \cdot K^n) \subseteq O(2K^n)$.

Ref:
a) Sei A ein Algorithmus, n die Länge der Eingabe.
i) A hat polynomiale Laufzeit wenn $T(n) = O(n^k)$ für ein $k \in \mathbb{N}$.*
ii) A hat exponentielle Laufzeit wenn $T(n) = O(K^n)$ für ein $K \geq 2$).
b) Probleme, für die ein polynomiales (bzw exponentielles) Algorithmus existiert, heißen polynomial (bzw exponentiell) lösbar.

Beispiele: i) PRIM und 2-Col sind polynomial lösbar.
ii) 3-Col ist exponentiell lösbar.

Bekannt: 3-Col ist polynomial lösbar $\Leftrightarrow$ $\underbrace{P = NP}$

eines der 7
Millenium-Probleme.

Literatur: Uwe Schöning
Theoretische Informatik - kurz gefasst
5. Auflage
Spektrum
Akademischer Verlag