Fundamentals: Wachstum von Funktionen, Komplexität, Zahlen und Matrizen

Prof. Dr. Josef F. Bürgler

Studiengang Informatik
Hochschule Luzern, Informatik

I.BA_DMATH

Thema, Resultate, Ziele, Vorgehen

Thema: Wachstum von Funktionen, Komplexität von Algorithmen, Zahlen und Matrizen

Resultate:

Ziele:

- Die Studierenden können die Anzahl Rechenoperationen und den Speicherbedarf von einfachen Algorithmen durch die Grösse des Problems ausdrücken.
- Sie können das Wachstum von Funktionen mit Hilfe der Big-O Notation ausdrücken.
- Sie können die Begriffe Teiler oder Vielfaches einer Zahl, Primzahl, Primfaktorzerlegung anwenden.
- Sie können abschätzen wie viele Primzahlen z.B. kleiner 10^{300} sind.
- Sie verstehen den Begriff Kongruenz und modulare Arithmetik und können damit umgehen.
- Sie können den (erweiterten) Euklidischen Algorithmus anwenden um den grössten gemeinsamen Teiler zweier ganzer Zahlen zu bestimmen.
- Sie können mit Matrizen (auch binären) umgehen (Summe, Differenz, Produkt)

Vorgehen: Die einzelnen Themen werden der Riehe nach behandelt und gleich an Beispielen gezeigt.

Inhalt

Übersicht

Algorithmen

Viele Aufgaben lassen sich mit Hilfe eines Algoritmus lösen: beispielsweise das Finden des grössten Elementes in einer Liste!

Definition (Algorithmus)

Ein Algorithmus ist eine endliche Menge von präzisen Instruktionen mit deren Hilfe eine Berechnung ausgeführt oder ein Problem gelöst wird.

Maximales Element in einer endlichen Folge:

```
float max(int n, float a[1..n]) {
  float max = a[1];
  for ( int i=2; i <= n; i++ )
    if ( max < a[i] ) then max = a[i];
  return max;
}
```

Suchalgorithmen

Donald Knuth in der Buchreihe "*The Art of Computer Programming*": **A lot of computer power is used for sorting and searching**!
Auf rund 400 Seiten beschreibt er 15 verschiedene Suchalgorithmen, wie

1. Bubble sort
2. Quick sort
3. Insertion sort
4. Binary insertion sort
5. Selection sort
6. Shaker sort
7. Merge sort
8. Tournament sort

um nur einige zu nennen!

Algorithmen (Fort.)

Algorithmen haben folgende Eigenschaften:

- einen genau spezifizierten **Input** und daraus berechneten **Output**.
- die **Instruktionen sind präzise**, **korrekt** für jeden möglichen Input und **in endlicher Zeit durchfürbar**.

Im Programmieren lernen sie Algorithmen für die (lineare) Suche, für das Sortieren (Bubble Sort, Heap Sort), etc.
Eine besondere Art sind die **Greedy Algorithmen** (gefräsige Algorithmen). Sie werden oft in Optimierungsproblemen eingesetzt: dann wählen sie in jedem Schritt die zu diesem Zeitpunkt optimale Lösung. Oft ist dies auch die global beste Lösung!

Übersicht

sehr komplizierte Fkt. $(x^2+1)\ln x + (2^x + x^4) = f(x)$

z.B. Anzahl Operationen bei einem Problem der Grösse x ($x = 10, 10^3, 10^6, \ldots$)

Definition

Seien f und g Funktion von $\mathbb{Z}$ oder $(\mathbb{R})$ nach $\mathbb{R}$. Dann sagt man "f(x) *ist* $\mathcal{O}(g(x))$", falls es Konstanten C und k gibt, so dass gilt:

$$|f(x)| \leqslant C|g(x)|, \quad \forall x > k.$$

einfachere Fkt. wie $x, x^2, x^3, \ldots$ $x^n, 2^x, e^x, \ln(x), x\ln(x), \ldots$

Lies: "f(x) *ist gross-O von* g(x)"

f(x) ist von der Grössenordnung g(x).

Oft schreibt man auch kurz: $f(x) = \mathcal{O}(g(x))$.

f(x) verhält sich für grosse x ungefähr so wie g(x).

Example

Zeige: $f(x) = x^2 + 2x + 1$ ist $\mathcal{O}(x^2)$.

Wir betrachten **nur** reelle Zahlen x mit $x > 1$. Für diese Zahlen gilt auch $x^2 > x$ und $x^2 > 1$ und weiterhin (da f in diesem Bereich nur positive Werte annehmen kann):

$$|f(x)| = |x^2 + 2x + 1| = x^2 + 2\underbrace{x}_{<x^2} + \underbrace{1}_{<x^2} \leq x^2 + 2x^2 + x^2 = 4x^2$$

Die Zahl x^2+2x+1 ist (weil $x>1$) immer positiv.

$$\leq C|x^2|, \quad \forall x > k.$$

$$|x| = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}$$

$(|-3| = -(-3) = 3)$

Example

Zeige: $f(x) = x^2 + 2x + 1$ ist $\mathcal{O}(x^2)$.

Wir betrachten **nur** reelle Zahlen x mit $x > 1$. Für diese Zahlen gilt auch $x^2 > x$ und $x^2 > 1$ und weiterhin (da f in diesem Bereich nur positive Werte annehmen kann):

$$|x^2 + 2x + 1| \;=\; x^2 + 2\underbrace{x}_{<x^2} + \underbrace{1}_{<x^2} \;\leqslant\; x^2 + 2x^2 + x^2 \;=\; 4x^2$$

Insgesamt haben wir also gezeigt: Für alle $x > \underbrace{1}_{=k}$ gilt

$$\underbrace{|x^2 + 2x + 1|}_{=|f(x)|} \;\leqslant\; \underbrace{4}_{=C}\;\underbrace{|x^2|}_{=|g(x)|}$$

also $f(x) = x^2 + 2x + 1$ ist $\mathcal{O}(x^2)$ mit den Zeugen $k = 1$ und $C = 4$.

Example

Zeige: $f(x) = 7x^2$ ist $\mathcal{O}(x^3)$.

Lösung: Falls $x > 7$ ist, so gilt sicher auch $x^3 = x \cdot x \cdot x > 7 \cdot x \cdot x = 7x^2$ also

$$|7x^2| \;=\; 7x^2 \leqslant 1 \cdot x^3$$

Example

Zeige: $f(x) = 7x^2$ ist $\mathcal{O}(x^3)$.

Lösung: Falls $x > 7$ ist, so gilt sicher auch $x^3 = x \cdot x \cdot x > 7 \cdot x \cdot x = 7x^2$ also

$$|7x^2| \;=\; 7x^2 \leqslant 1 \cdot x^3$$

Insgesamt haben wir also gezeigt: Für alle $x > \underbrace{7}_{=k}$ gilt

$$\underbrace{|7x^2|}_{=|f(x)|} \leqslant \underbrace{1}_{=C} \underbrace{|x^3|}_{=|g(x)|}$$

also $f(x) = 7x^2$ ist $\mathcal{O}(x^3)$ mit den Zeugen $k = 7$ und $C = 1$.

Bessere Abschätzung:
Es gilt auch
$f(x) = \mathcal{O}(x^2)$
Man kann zeigen (Hausaufgabe), dass die Zeugen $C = 7$ und $k = 1$)

Wachstum von Funktionen (Fort.)

$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$

Theorem

Für das Polynom $f(x) = \sum_{k=0}^{n} a_k x^k$ *mit reellen Koeffizienten* a_k*, wobei* $k = 0, 1, \ldots, n$ *gilt:* $f(x)$ *ist* $\mathcal{O}(x^n)$*. d.h.* **die höchste Potenz von** x **gibt den Ton an.**

Beweis.

Wir verwenden die Dreiecksungleichung $|a + b| \leqslant |a| + |b|$ und erhalten für $x > 1$ nacheinander:

$$|f(x)| = \left|\sum_{k=0}^{n} a_k x^k\right| \leqslant \sum_{k=0}^{n} |a_k| x^k = x^n \sum_{k=0}^{n} |a_k| / x^{n-k} \leqslant x^n \sum_{k=0}^{n} |a_k|$$

$x^{n-k} > 1$

Mit den *Zeugen* $C = \sum_{k=0}^{n} |a_k|$ und $k = 1$ hat man also, wie behauptet: $|f(x)| \leqslant Cx^n$, $\forall x > k$. □

$x > 1$

$|a \cdot b| = |a| \cdot |b|$ insbesondere $|a_k x^k| = |a_k| |x^k| = |a_k| x^k$

Examples

Nutzen Sie das Theorem, um die folgenden Funktionen $\mathcal{O}$-abzuschätzen. Geben Sie jeweils die Zeugen k und C an. Finden Sie für das zweite Beispiel auch eine bessere Abschätzung?

$$f(x) = x^5 - 3x^3 + 2x^2 - 13$$
$$f(x) = x^3 - 3x^4 + 3$$
$$f(x) = 12x^3 + 3x^2 + 3x + 12$$

$C = |1| + |-3| + |2| + |-13| = 19, \; k = 1$

$1 + |-3| + 3 = 7 = C$

$k = 1$

$C = 12 + 3 + 3 + 12$

$C = 30, \; k = 1$

Exponential- und Logarithmusfunktionen

Sei $a > 0$ und $a \neq 1$, r, s beliebig und $u, v > 0$. Dann gelten die folgenden Rechenregeln:

$a^r \cdot a^s$	$=$	a^{r+s}	$\log_a(u \cdot v)$	$=$	$\log_a(u) + \log_a(v)$
$\frac{a^r}{a^s}$	$=$	a^{r-s}	$\log_a\left(\frac{u}{v}\right)$	$=$	$\log_a(u) - \log_a(v)$
$(a^r)^s$	$=$	$(a^s)^r = a^{r \cdot s}$	$\log_a(u^w)$	$=$	$w \cdot \log_a(u)$

Examples

Finde eine Abschätzung für die Fakultät:

$$(n+1)! = (n+1)\,n!$$

$$n! = \begin{cases} 1, & \text{falls } n = 0, \\ n(n-1)\cdots 2 \cdot 1, & \text{falls } n > 0. \end{cases}$$

sowie für deren Logarithmus!

Lösung: Man findet nach einigen Abschätzungen: $n!$ ist $\mathcal{O}(n^n)$ und daraus folgend $\log(n!)$ ist $\mathcal{O}(n \log n)$.

e^x

$$\log(n!) = \log(n(n-1)(n-2)\cdots 3\cdot 2\cdot 1) = \log n + \underbrace{\log(n-1)}_{<\log n} + \underbrace{\log(n-2)}_{<\log n} + \ldots + \underbrace{\log 2}_{<\log n} + \underbrace{\log 1}_{<\log n}$$

$$\leq n \log n, \quad \forall n > 1$$

$g(x)$ $\quad (C=1, k=1)$

$\ln x$

$\ln(n-1)$ $\quad \ln n$

$n-1$ $\quad n$

$\log n! = \mathcal{O}(n \log n)$

$\log a \leq \log b$

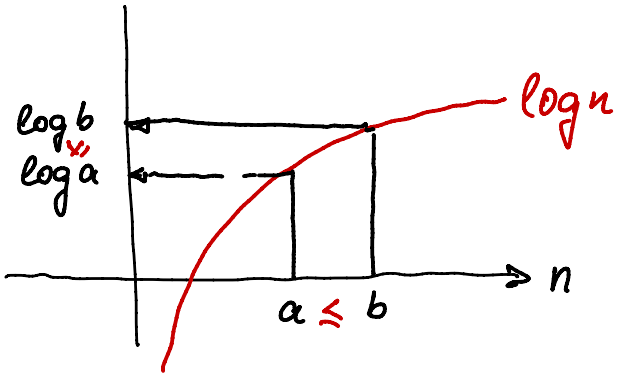
$\downarrow$

$a \leq b$

$$\log n! \leq n \log n = \log n^n \qquad \mid \log^{-1}$$

$$n! \leq n^n, \quad \forall n \geq 1$$

Somit $n! = O(n^n)$ mit den Zeugen $C = 1$ u. $k = 1$.



Example

Gebe eine *gross-O*-Abschätzung für $f(n) = 3n \log(n!) + (n^2 + 3) \log n$.

Begründen Sie jeden Umformungsschritt.

$n > 1$

$$
\begin{aligned}
|f(n)| &= |3n \log(n!) + (n^2 + 3) \log n| \\
&= 3n \log(n!) + (n^2 + 3) \log n \\
&= 3n \log(n!) + n^2 \log n + 3 \log n \\
&\leqslant 3n(n \cdot \log n) + n^2 \log n + 3 \log n \quad \text{falls } n > 1 \\
&= 3n^2 \log n + n^2 \log n + 3 \log n \\
&= 4n^2 \log n + 3 \log n \\
&\leqslant 4n^2 \log n + 3n^2 \log n \quad \text{falls } n > 1 \\
&= 7n^2 \log n \\
&= C
\end{aligned}
$$

$f(n) = O(n^2 \log n)$ mit den Zeugen $C = 7$ und $k = 1$.

Übersicht

Komplexität von Algorithmen

Gegeben: ein bestimmter Algorithmus zur Lösung eines Problem(typs) (z.B. Sortieren einer Liste der Länge n, Auffinden des grössten Elementes in einer Liste der Länge n, ...)

Komplexität von Algorithmen

Gegeben: ein bestimmter Algorithmus zur Lösung eines Problem(typs) (z.B. Sortieren einer Liste der Länge $\mathfrak{n}$, Auffinden des grössten Elementes in einer Liste der Länge $\mathfrak{n}$, ...)

Laufzeit- oder Speicherplatz-Komplexität: es muss die Anzahl Operationen oder der benötigte Speicherplatz $\mathfrak{f} = \mathfrak{f}(\mathfrak{n})$ als Funktion der Grösse $\mathfrak{n}$ des zu lösenden Problems dargestellt werden:

$$\mathfrak{f}(\mathfrak{n}) = \text{ Anzahl notwendiger (Rechenoperationen) zur Lösung eines Problems der Grösse } \mathfrak{n}$$

Komplexität von Algorithmen

Gegeben: ein bestimmter Algorithmus zur Lösung eines Problem(typs) (z.B. Sortieren einer Liste der Länge $\mathfrak{n}$, Auffinden des grössten Elementes in einer Liste der Länge $\mathfrak{n}$, ...)

Laufzeit- oder Speicherplatz-Komplexität: es muss die Anzahl Operationen oder der benötigte Speicherplatz $\mathfrak{f} = \mathfrak{f}(\mathfrak{n})$ als Funktion der Grösse $\mathfrak{n}$ des zu lösenden Problems dargestellt werden:

$$\mathfrak{f}(\mathfrak{n}) = \text{ Anzahl notwendiger (Rechenoperationen) zur Lösung eines Problems der Grösse } \mathfrak{n}$$

Um verschiedene Algorithmen leichter vergleichen zu können, nutzt man meist $\mathcal{O}$-Abschätzungen der Funktion $\mathfrak{f}$!

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.
- Durchschnittliche (average-case) Komplexität ergibt sich aus der durchschnittlichen Anzahl Operationen.

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.
- Durchschnittliche (average-case) Komplexität ergibt sich aus der durchschnittlichen Anzahl Operationen.
- Es gibt **konstante** ($\mathcal{O}(1)$), **logarithmische** ($\mathcal{O}(\log n)$), **lineare** ($\mathcal{O}(n)$), $\mathbf{n \log n}$ ($\mathcal{O}(n \log n)$), **polynomiale** ($\mathcal{O}(n^b)$), **exponentielle** ($\mathcal{O}(b^n)$, $b > 1$) und **faktorielle** ($\mathcal{O}(n!)$) Komplexität.

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.
- Durchschnittliche (average-case) Komplexität ergibt sich aus der durchschnittlichen Anzahl Operationen.
- Es gibt **konstante** ($\mathcal{O}(1)$), **logarithmische** ($\mathcal{O}(\log n)$), **lineare** ($\mathcal{O}(n)$), $\mathbf{n \log n}$ ($\mathcal{O}(n \log n)$), **polynomiale** ($\mathcal{O}(n^b)$), **exponentielle** ($\mathcal{O}(b^n)$, $b > 1$) und **faktorielle** ($\mathcal{O}(n!)$) Komplexität.
- Probleme die mit polynomialer worst-case Komplexität gelöst werden können heissen *durchführbar*, sonst *nicht durchführbar*.

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.
- Durchschnittliche (average-case) Komplexität ergibt sich aus der durchschnittlichen Anzahl Operationen.
- Es gibt **konstante** ($\mathcal{O}(1)$), **logarithmische** ($\mathcal{O}(\log n)$), **lineare** ($\mathcal{O}(n)$), $\mathbf{n \log n}$ ($\mathcal{O}(n \log n)$), **polynomiale** ($\mathcal{O}(n^b)$), **exponentielle** ($\mathcal{O}(b^n)$, $b > 1$) und **faktorielle** ($\mathcal{O}(n!)$) Komplexität.
- Probleme die mit polynomialer worst-case Komplexität gelöst werden können heissen *durchführbar*, sonst *nicht durchführbar*.
- Probleme für die kein Algorithmus existiert heissen *nicht berechenbar* (*halting problem*); im Gegensatz zu obigen, die *berechenbar* sind.

Komplexität von Algorithmen

- Schlimmstmögliche (worst-case) Komplexität der *linearen Suche* ist schlechter als die der *binären Suche*.
- Durchschnittliche (average-case) Komplexität ergibt sich aus der durchschnittlichen Anzahl Operationen.
- Es gibt **konstante** ($\mathcal{O}(1)$), **logarithmische** ($\mathcal{O}(\log n)$), **lineare** ($\mathcal{O}(n)$), $\mathbf{n \log n}$ ($\mathcal{O}(n \log n)$), **polynomiale** ($\mathcal{O}(n^b)$), **exponentielle** ($\mathcal{O}(b^n)$, $b > 1$) und **faktorielle** ($\mathcal{O}(n!)$) Komplexität.
- Probleme die mit polynomialer worst-case Komplexität gelöst werden können heissen *durchführbar*, sonst *nicht durchführbar*.
- Probleme für die kein Algorithmus existiert heissen *nicht berechenbar* (*halting problem*); im Gegensatz zu obigen, die *berechenbar* sind.
- Mehr zum Studium der Komplexität von Algorithmen zum Lösen von Problemen (**P**-, **NP**- und **NP**-vollständige Probleme) im Kapitel Graphentheorie.

Übersicht

Definition

Falls $a, b \in \mathbb{Z}$ mit $a \neq 0$ dann sagt man: a *teilt* b, falls $\exists c\,(b = ac)$ in der Universalmenge $\mathbb{Z}$. Dann ist a ein *Faktor* von b und b ein *Vielfaches* von a. Man schreibt dann $a \mid b$ und anderenfalls $a \nmid b$.

a teilt b

$2 \mid 4, 2 \mid 6, 3 \nmid 5, 3 \mid 6$

Theorem

Falls $a, b, c \in \mathbb{Z}$, *dann gilt:*

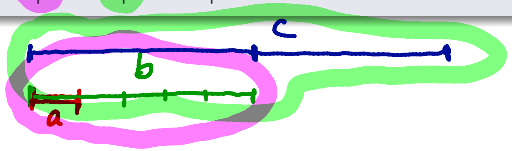
(a) $a|b \wedge a|c \rightarrow a|(b+c)$,

(b) $a|b \rightarrow \forall c(a|bc)$,

(c) $a|b \wedge b|c \rightarrow a|c$.

$3 \mid 6 \wedge 3 \mid 9 \longrightarrow 3 \mid (6+9)$ (15)

$3 \mid 6 \longrightarrow 3 \mid 12, 3 \mid 18, 3 \mid 24, \ldots$



Definition

Falls $a, b \in \mathbb{Z}$ mit $a \neq 0$ dann sagt man: a *teilt* b, falls $\exists c\,(b = ac)$ in der Universalmenge $\mathbb{Z}$. Dann ist a ein *Faktor* von b und b ein *Vielfaches* von a. Man schreibt dann $a \mid b$ und anderenfalls $a \nmid b$.

Theorem

Falls $a, b, c \in \mathbb{Z}$, *dann gilt:*

(a) $a|b \wedge a|c \rightarrow a|(b+c)$,

(b) $a|b \rightarrow \forall c(a|bc)$,

(c) $a|b \wedge b|c \rightarrow a|c$.

Suchen sie für jede der oben aufgestellten Behauptungen ein Beispiel!

Primzahlen

Definition

Eine positive Zahl $n \in \mathbb{Z}$ grösser als 1 heisst *Primzahl*, wenn sie lediglich die Faktoren 1 und n hat. Andernfalls heisst die Zahl *zusammengesetzt:* in diesem Fall gilt: $\exists a\,(a|n \wedge (1 < a < n))$.

Primzahlen lassen sich z.B. mit dem **Sieb von Eratosthenes** finden. Will man beispielsweise die Primzahlen kleiner $n = 100$ finden muss man lediglich alle Primzahlen $1 < p \leqslant \sqrt{n} = 10$ testen (siehe `http://www.faust.fr.bw.schule.de/mhb/eratosib.htm`).

Theorem

Falls n eine zusammengesetzte Zahl ist, dann hat n einen Primzahlteiler kleiner gleich $\sqrt{n}$.

Primzahlen (Sieb von Eratosthenes)

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	~~25~~	~~26~~	27	~~28~~	29	~~30~~
31	~~32~~	33	~~34~~	~~35~~	~~36~~	37	~~38~~	39	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~
51	~~52~~	53	~~54~~	~~55~~	~~56~~	57	~~58~~	59	~~60~~
61	~~62~~	63	~~64~~	~~65~~	~~66~~	67	~~68~~	69	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	77	~~78~~	79	~~80~~
81	~~82~~	83	~~84~~	~~85~~	~~86~~	87	~~88~~	89	~~90~~
91	~~92~~	93	~~94~~	~~95~~	~~96~~	97	~~98~~	99	~~100~~

Example

Zahlen lassen sich (eindeutig) faktorisieren:

$$744776409 = 3 \cdot 13^3 \cdot 17^3 \cdot 23$$

Fundamentalsatz der Arithmetik

Example

Zahlen lassen sich (eindeutig) faktorisieren:

$$744776409 \quad = \quad 3 \cdot 13^3 \cdot 17^3 \cdot 23$$

Theorem (Fundamentalsatz der Arithmetik)

Jede natürliche Zahl $\mathfrak{n} \geqslant 2$ kann (eindeutig) als Produkt von Primzahlen geschrieben werden.

Fundamentalsatz der Arithmetik

Example

Zahlen lassen sich (eindeutig) faktorisieren:

$$744776409 = 3 \cdot 13^3 \cdot 17^3 \cdot 23$$

Theorem (Fundamentalsatz der Arithmetik)

Jede natürliche Zahl $\mathfrak{n} \geqslant 2$ kann (eindeutig) als Produkt von Primzahlen geschrieben werden.

Examples

Primfaktorzerlegungen von 100, 641, 999 und 1024:

Es gibt unendlich viele Primzahlen

Theorem

Es gibt unendlich viele Primzahlen.

Es gibt unendlich viele Primzahlen

Theorem

Es gibt unendlich viele Primzahlen.

Beweis.

1. Angenommen es gibt nur endlich viele Primzahlen p_1, p_2 $\ldots$, p_n. Betrachte dann die Zahl $Q = p_1 p_2 \cdots p_n + 1$.

Es gibt unendlich viele Primzahlen

Theorem

Es gibt unendlich viele Primzahlen.

Beweis.

1. Angenommen es gibt nur endlich viele Primzahlen p_1, p_2 ..., p_n. Betrachte dann die Zahl $Q = p_1 p_2 \cdots p_n + 1$.
2. Nach dem Fundamentalsatz der Arithmetik ist Q entweder eine Primzahl oder kann als Produkt von zwei oder mehr Primzahlen geschrieben werden.

Es gibt unendlich viele Primzahlen

Theorem

Es gibt unendlich viele Primzahlen.

Beweis.

1. Angenommen es gibt nur endlich viele Primzahlen p_1, p_2 $\ldots$, p_n. Betrachte dann die Zahl $Q = p_1 p_2 \cdots p_n + 1$.
2. Nach dem Fundamentalsatz der Arithmetik ist Q entweder eine Primzahl oder kann als Produkt von zwei oder mehr Primzahlen geschrieben werden.
3. Keine der Zahlen p_j teilt aber Q, da wegen $Q - p_1 p_2 \cdots p_n = 1$ sonst p_j auch 1 teilen müsste; was ein Widerspruch ist.

□

Es gibt unendlich viele Primzahlen

Theorem

Es gibt unendlich viele Primzahlen.

Beweis.

1. Angenommen es gibt nur endlich viele Primzahlen p_1, p_2 $\ldots$, p_n. Betrachte dann die Zahl $Q = p_1 p_2 \cdots p_n + 1$.
2. Nach dem Fundamentalsatz der Arithmetik ist Q entweder eine Primzahl oder kann als Produkt von zwei oder mehr Primzahlen geschrieben werden.
3. Keine der Zahlen p_j teilt aber Q, da wegen $Q - p_1 p_2 \cdots p_n = 1$ sonst p_j auch 1 teilen müsste; was ein Widerspruch ist.
4. Also ist die obige Aufzählung nicht vollständig, d.h. es gibt unendlich viele Primzahlen.

□

Mersenne Primes

Definition (Mersenne primes)

Primzahlen der Form $\mathrm{M_n} = 2^{\mathrm{p}} - 1$ wobei p eine Primzahl ist, heissen **Mersenne Primes**.

Mersenne Primes

Definition (Mersenne primes)

Primzahlen der Form $M_n = 2^p - 1$ wobei p eine Primzahl ist, heissen **Mersenne Primes**.

Example (Primzahlsuche im Internet)

Nicht alle Zahlen der Form $2^n - 1$ sind Primzahlen, wie man mit $2^{11} - 1 = 2047 = 23 \cdot 89$ einfach nachprüft!
Die bis heute (7. Januar 2016) grösste gefundene Mersenne Primzahl ist die 48. Mersenne-Primzahl (siehe `http://www.mersenne.org`, Great Internet Mersenne Prime Search, kurz GIMPS)

$$2^{74,207,281} - 1: \quad \text{eine Zahl mit } 22'338'618 \text{ Ziffern (7. Jan. 2016)}$$

Beachte: $2^x = 10^{x \log_{10}(2)} \approx 10^{0.3x}$ (Beweis: via Rechnen mit Logarithmen).

Gibt es genug Primzahlen?

Theorem (Primzahlsatz)

Die Anzahl Primzahlen kleiner gleich x kann für sehr grosse x abgeschätzt werden durch

$$\pi(x) \approx \frac{x}{\ln x}$$

Gibt es genug Primzahlen?

Theorem (Primzahlsatz)

Die Anzahl Primzahlen kleiner gleich x kann für sehr grosse x abgeschätzt werden durch

$$\pi(x) \approx \frac{x}{\ln x}$$

Example

Beim RSA public-key Verfahren erzeugt jeder Teilnehmer einen privaten und einen öffentlichen Schlüssel. Dazu werden Primzahlen verwendet. Je mehr Primzahlen, umso grösser ist die Auswahl und umso weniger kann der Schlüssel erraten werden.
Wie viele 200-stellige Zahlen müssen wir im Schnitt wählen um darunter mindestens eine Primzahl zu finden?

Die grösste 200-stellige Zahl ist $10^{200}-1$, die kleinste 10^{199}.
Nach Euler gibt es ungefähr

$$n = \frac{10^{200}}{\ln 10^{200}} - \frac{10^{199}}{\ln 10^{199}} = \frac{1}{\ln 10}\left(\frac{10^{200}}{200} - \frac{10^{199}}{199}\right)$$

200 stellige Primzahlen. Es gibt $9 \cdot 10^{199}$ 200-stellige Zahlen.
Der Anteil von Primzahlen ist somit

$$\frac{n}{9\cdot 10^{199}} = \frac{1}{9\cdot \ln 10}\left(\frac{10}{200} - \frac{1}{199}\right) = \frac{1989}{18\cdot 19900\cdot \ln(10)} = 2{,}41\cdot 10^{-3}$$

Falls wir nur ungerade Zahlen betrachten, ist im Schnitt jede

$$\frac{1}{2{,}41\cdot 10^{-3}\,\frac{1}{2}} \approx 207.\ \text{Zahl eine Primzahl!}$$

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$
- Man bezeichnet: $q = a$ **div** d und $r = a$ **mod** d!

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$
- Man bezeichnet: $q = a$ **div** d und $r = a$ **mod** d!
- Der *grösste gemeinsame Teiler* (ggT oder gcd) von zwei Zahlen a und b (nicht beide gleich Null) ist die grösste ganze Zahl d für die $d|a$ und $d|b$ gilt.

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$
- Man bezeichnet: $q = a$ **div** d und $r = a$ **mod** d!
- Der *grösste gemeinsame Teiler* (ggT oder gcd) von zwei Zahlen a und b (nicht beide gleich Null) ist die grösste ganze Zahl d für die $d|a$ und $d|b$ gilt.
- Zwei Zahlen heissen teilerfremd (relativ prim), falls $\mathrm{ggT}(a, b) = 1$.

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$
- Man bezeichnet: $q = a$ **div** d und $r = a$ **mod** d!
- Der *grösste gemeinsame Teiler* (ggT oder gcd) von zwei Zahlen a und b (nicht beide gleich Null) ist die grösste ganze Zahl d für die $d|a$ und $d|b$ gilt.
- Zwei Zahlen heissen teilerfremd (relativ prim), falls $\mathrm{ggT}(a, b) = 1$.
- Analog definiert man paarweise teilerfremde Zahlen!

ggT und kgV

- Der Divisionsalgorithmus: Dividiert man $a = 7$ (Dividend) durch $d = 2$ (Divisor) erhält man den $q = 3$ (Quotient) und $r = 1$ (Rest): Es gilt also $a = dq + r$ $(0 \leqslant r < d)$! Beispiel: $7 = 2 \cdot 3 + 1$
- Man bezeichnet: $q = a$ **div** d und $r = a$ **mod** d!
- Der *grösste gemeinsame Teiler* (ggT oder gcd) von zwei Zahlen a und b (nicht beide gleich Null) ist die grösste ganze Zahl d für die $d|a$ und $d|b$ gilt.
- Zwei Zahlen heissen teilerfremd (relativ prim), falls $\mathrm{ggT}(a, b) = 1$.
- Analog definiert man paarweise teilerfremde Zahlen!
- Das *kleinste gemeinsame Vielfache* (kgV oder lcm) zweier Zahlen ist die kleinste positive Zahl die teilbar ist durch a und b. Es gilt: $ab = \mathrm{ggT}(a, b) \cdot \mathrm{kgV}(a, b)$.

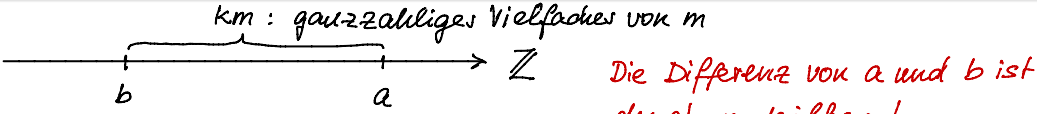
$$\Rightarrow \mathrm{kgV}(a,b) = \frac{a \cdot b}{\mathrm{ggT}(a,b)}$$

Definition (Kongruenz)

Sei $m \in \mathbb{N}$. Dann nennt man zwei ganze Zahlen a und b **kongruent modulo** m, falls $m|(a-b)$ d.h. a und b liegen ein Vielfaches von m auseinander. Man schreibt dann $a \equiv b \bmod m$ und sagt: "a *ist kongruent zu* b *modulo* m".

Definition (Kongruenz)

Sei $m \in \mathbb{N}$. Dann nennt man zwei ganze Zahlen a und b **kongruent modulo** m, falls $m|(a-b)$ d.h. a und b liegen ein Vielfaches von m auseinander. Man schreibt dann $a \equiv b \bmod m$ und sagt: "a *ist kongruent zu* b *modulo* m".



Example

$$13 \equiv 1 \bmod 4 \text{ denn } 4|(13-1)$$
$$13 \equiv 1 \bmod 3 \text{ denn } 3|(13-1)$$
$$13 \not\equiv 1 \bmod 5 \text{ denn } 5 \not|(13-1)$$

Wenn wir das Modul $\mathfrak{m}$ fixieren, so können wir uns mit dem folgenden Schema einen guten Überblick über die Kongruenzen machen, indem wir die ganzen Zahlen zeilenweise aufschreiben. In den Spalten stehen dann jeweils kongruente Zahlen!
Wir wählen als Beispiel $\mathfrak{m} = 9$

0 3
-18 + -6 = -24
= 3 mod 9
3

			.....			-21	-20	-19
-18	-17	-16	-15	-14	-13	-12	-11	-10
-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	.....					

alle diese Zahlen sind kongruent zu 1 modulo 9.

ditto: kongruent zu 7 modulo 9.

- Man sieht sofort: $a \equiv b \bmod m \leftrightarrow a \bmod m = b \bmod m$.

- Man sieht sofort: $a \equiv b \bmod m \leftrightarrow a \bmod m = b \bmod m$.
- Ebenso gilt: $a \equiv b \bmod m \leftrightarrow \exists k \in \mathbb{Z}\,(a = b + km)$. $\rightsquigarrow$ $a - b = km$

Vielfaches von m

- Man sieht sofort: $a \equiv b \bmod m \leftrightarrow a \bmod m = b \bmod m$.
- Ebenso gilt: $a \equiv b \bmod m \leftrightarrow \exists k \in \mathbb{Z}\,(a = b + km)$.
- Die Menge aller Zahlen kongruent zu a modulo m heisst **Kongruenzklasse von a modulo m** und wird oft mit $[a]_m$ bezeichnet. Kongruenzklasse von 3 modulo 5 ist $[3]_5 = \{\ldots, -2, 3, 8, \ldots\}$! $[2]_5 = \{\ldots, -8, -3, 2, 7, 12, 17, \ldots\}$

- Man sieht sofort: $a \equiv b \bmod m \leftrightarrow a \bmod m = b \bmod m$.
- Ebenso gilt: $a \equiv b \bmod m \leftrightarrow \exists k \in \mathbb{Z}\,(a = b + km)$.
- Die Menge aller Zahlen kongruent zu a modulo m heisst **Kongruenzklasse von a modulo m** und wird oft mit $[a]_m$ bezeichnet. Kongruenzklasse von 3 modulo 5 ist $[3]_5 = \{\ldots, -2, 3, 8, \ldots\}$!
- Falls $a \equiv b \bmod m$ und $c \equiv d \bmod m$, dann gilt erstens $a + c \equiv b + d \bmod m$ und zweitens $ac \equiv bd \bmod m$. Beim modularen Rechnen darf man irgend wann modulo m rechnen.

$$6^{13} \bmod 5 = \begin{cases} 13\,060\,694\,016 \bmod 5 = 1 & \text{blöd!} \\ (\underbrace{6 \bmod 5}_{1})^{13} \bmod 5 = 1 \bmod 5 = 1 \end{cases}$$

Übersicht

Die Studierenden sollten sich selber zu folgenden Themen schlau machen:

- Umwandlung von Binär-, Oktal-, Dezimal- und Hexadezimalzahlen.
- Versuchen Sie die Aussage des Theorems auf der folgenden Seite zu verstehen und bestimmen Sie mit seiner Hilfe den ggT der Zahlen 12345 und 54321.

Der Euklidische Algorithmus (ggT)

Theorem

Seien $a, b \in \mathbb{Z}$ *mit* $a > b$. *Dann gilt:*

$$
\begin{array}{lcll}
a & = & q_1 \cdot b + r_1 & 0 \leqslant r_1 < b \\
b & = & q_2 \cdot r_1 + r_2 & 0 \leqslant r_2 < r_1 \\
r_1 & = & q_3 \cdot r_2 + r_3 & 0 \leqslant r_3 < r_2 \\
 & & & \vdots \\
r_{t-2} & = & q_t \cdot r_{t-1} + r_t & 0 \leqslant r_t < r_{t-1} \\
r_{t-1} & = & q_{t+1} \cdot r_t &
\end{array}
\qquad
\begin{array}{ll}
\mathrm{ggT}(a, b) & = \mathrm{ggT}(b, r_1) \\
\mathrm{ggT}(b, r_1) & = \mathrm{ggT}(r_1, r_2) \\
\mathrm{ggT}(r_1, r_2) & = \mathrm{ggT}(r_2, r_3) \\
 & \\
\mathrm{ggT}(r_{t-2}, r_{t-1}) & = \mathrm{ggT}(r_{t-1}, r_t) \\
\mathrm{ggT}(r_{t-1}, r_t) & = \mathrm{ggT}(r_t, 0)
\end{array}
$$

Dabei ist r_t *der letzte nicht verschwindende Rest und dieser ist gleich dem* $\mathrm{ggT}(a, b)$*!*

Der Euklidische Algorithmus (Beispiele)

Berechne ggT(67, 24) und ggT(201, 72).

$$
\begin{aligned}
67 &= 2 \cdot 24 + 19 \\
24 &= 1 \cdot 19 + 5 \\
19 &= 3 \cdot 5 + 4 \\
5 &= 1 \cdot 4 + 1 \\
4 &= 4 \cdot 1 + 0
\end{aligned}
\qquad 1 = ggT(67, 24)
$$

$$
\begin{aligned}
201 &= 2 \cdot 72 + 57 \\
72 &= 1 \cdot 57 + 15 \\
57 &= 3 \cdot 15 + 12 \\
15 &= 1 \cdot 12 + 3 \\
12 &= 4 \cdot 3 + 0
\end{aligned}
$$

$ggT(54321, 12345) = ?$

$$54321 = 4 \cdot 12345 + 4941$$
$$12345 = 2 \cdot 4941 + 2463$$
$$4941 = 2 \cdot 2463 + 15$$
$$2463 = 164 \cdot 15 + 3$$
$$15 = 5 \cdot 3 + 0$$

$3 = ggT(54321, 12345)$

Übersicht

Definition (Matrix)

Eine $m \times n$-Matrix ist eine rechteckige Anordnung von Zahlen in m Zeilen (Reihen) und n Spalten. Für eine quadratische Matrix gilt $m = n$.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Man schreibt kurz auch $\mathbf{A} = [a_{ij}]$.

Example

Identifizieren sie die 1. Zeile, die 2. Spalte sowie das $(2,2)$-Element der obigen Matrix!

Definition (Addition zweier Matrizen)

Zwei $m \times n$-Matrizen $\mathbf{A} = \left[a_{ij}\right]$ und $\mathbf{B} = \left[b_{ij}\right]$ werden addiert, indem man entsprechende Glieder der Matrix addiert:

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}$$

Example

Addieren sie die beiden Matrizen $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$ und $\mathbf{B} = \begin{pmatrix} -2 & 1 \\ 2 & -4 \end{pmatrix}$. Beachte: beide Matrizen müssen die selbe Grösse (Dimensionen) haben. Statt eckiger, werden auch runde Klammern verwendet!

Matrix Addition (A, B, m, n)

```
for i = 1 to m          // für alle Zeilen
    for j = 1 to n      // für alle Spalten
        C[i][j] = A[i][j] + B[i][j]
    endfor
endfor
return C
```

Definition (Multiplikation einer Matrix mit einer Zahl)

Eine $m \times n$-Matrizen $\mathbf{A} = [a_{ij}]$ wird mit einer Zahl $\alpha \in \mathbb{R}$ multipliziert, indem man jedes Matrixelement mit dieser Zahl multipliziert:

$$\alpha\mathbf{A} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{bmatrix}$$

Example

Multiplizieren sie die Matrix $\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$ (a) mit 3 und (b) mit -1!

$$3A = \begin{pmatrix} 3\cdot 1 & 3\cdot 2 \\ 3\cdot 3 & 3\cdot 1 \end{pmatrix} = \begin{pmatrix} 3 & 6 \\ 9 & 3 \end{pmatrix}; \quad -1A = \begin{pmatrix} -1 & -2 \\ -3 & -1 \end{pmatrix}$$

Rechnen mit Matrizen (Fort.)

- Durch die Matrixaddition wird zwei $m \times n$-Matrizen wieder eine $m \times n$-Matrix zugeordnet.
- Die **Nullmatrix** (d.h. die Matrix mit lauter Nullen) ist das **Neutralelement** der Matrixaddition: $\mathbf{A} + \mathbf{0} = \mathbf{0} + \mathbf{A} = \mathbf{A}$.
- Das Negative einer Matrix $\mathbf{A}$ ist die Matrix, bei der jedes Matrixelement mit -1 multipliziert wurde: $\mathbf{A} = (a_{ij}) \rightarrow -\mathbf{A} = (-a_{ij})$. Man nennt $-\mathbf{A}$ das **Inverse** von $\mathbf{A}$ bezüglich Addition.
- Man kann leicht nachrechnen, dass (bezüglich Addition von Matrizen) das **Assoziativgesetz**, $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$, gilt.
- Ebenso leicht kann man zeigen, dass (bezüglich Addition von Matrizen) das **Kommutativgesetz**, $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$, gilt.
- Damit bildet die Menge der $m \times n$-Matrizen zusammen mit der Matrixaddition eine kommutative Gruppe.

Definition (Multiplikation zweier Matrizen)

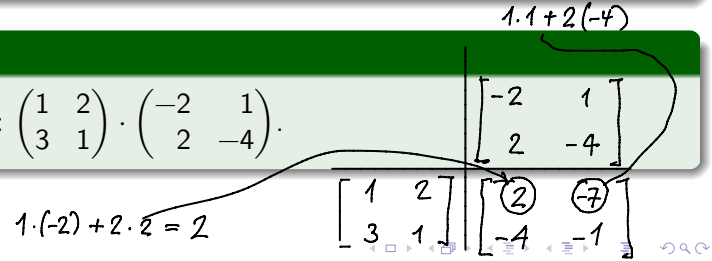
Das Produkt der $m \times k$-Matrix $\mathbf{A} = [a_{ij}]$ und der $k \times n$-Matrix $\mathbf{B} = [b_{ij}]$ ist eine $m \times n$-Matrix, definiert durch

$$\mathbf{AB} = [c_{ij}] \text{ wobei } c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj} = \sum_{l=1}^{k} a_{il}b_{lj}$$

c_{ij} ist das Skalarprodukt des i-ten Zeilenvektors von $\mathbf{A}$ mit dem j-ten Spaltenvektor von $\mathbf{B}$.

Example (Falk'sches Schema)

Berechne mit dem Falk'schen Schema: $\begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \cdot \begin{pmatrix} -2 & 1 \\ 2 & -4 \end{pmatrix}$.



Multiplikation von Matrizen (Falk'sches Schema)

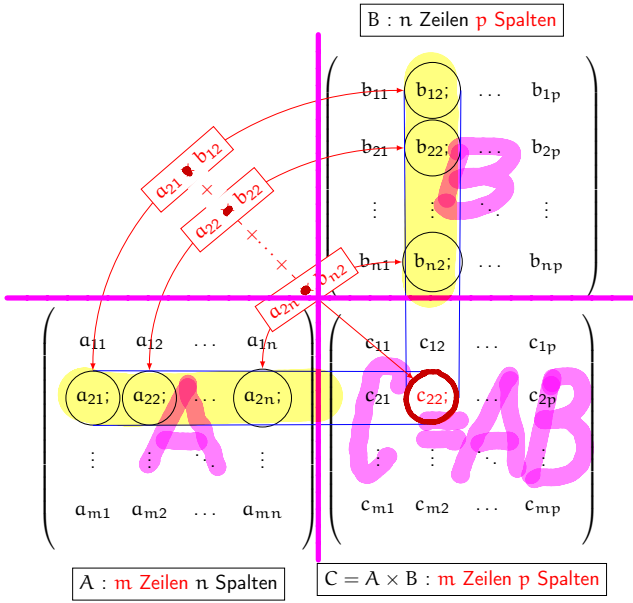
Speziell

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + \ldots + a_{2n}b_{n2} = \sum_{l=1}^{n} a_{2l}b_{l2}$$

Allgemein

$$c_{ij} = \sum_{l=1}^{n} a_{il}b_{lj}$$

wobei $1 \leqslant i \leqslant m$ und $1 \leqslant j \leqslant p$.



Example (Falk'sches Schema)

Zeigen sie, dass die Matrix-Multiplikation i.A. nicht kommutativ ist indem sie $\mathbf{AB}$ und $\mathbf{BA}$ berechnen, falls

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \text{ und } \mathbf{B} = \begin{pmatrix} -2 & 1 \\ 2 & -4 \end{pmatrix}.$$

von vorhin:

$$AB = \begin{bmatrix} 2 & -7 \\ -4 & -1 \end{bmatrix}$$

	$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix}$ A
$\begin{bmatrix} -2 & 1 \\ 2 & -4 \end{bmatrix}$ B	$\begin{bmatrix} 1 & -3 \\ -10 & 0 \end{bmatrix}$

$= BA \neq AB$

Definition (Transponierte Matrix)

Die **transponierte Matrix** $\mathbf{A}^T$ der $m \times n$-Matrix $\mathbf{A} = [a_{ij}]$ ergibt sich durch Vertauschen von Zeilen und Spalten, d.h.

$$\text{Falls } \mathbf{A}^T = [b_{ij}] \text{ dann gilt } b_{ij} = a_{ji} \text{ für } i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m.$$

Example (Transponierte Matrix)

Wie lauten die Transformierten der folgenden Matrizen:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \text{ und } \mathbf{B} = \begin{pmatrix} -2 & 1 & 3 \\ 2 & -4 & -2 \end{pmatrix}.$$

$$A^T = \begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix} \qquad B^T = \begin{bmatrix} -2 & 2 \\ 1 & -4 \\ 3 & -2 \end{bmatrix}$$

Rechnen mit Matrizen (Fort.)

- Eine Matrix $\mathbf{A}$ heisst **symmetrisch**, falls $\mathbf{A}^{\mathsf{T}} = \mathbf{A}$.
- Eine Matrix $\mathbf{A}$ heisst **antisymmetrisch**, falls $\mathbf{A}^{\mathsf{T}} = -\mathbf{A}$.
- Eine symmetrische oder antisymmetrische Matrix ist quadratisch!
- Die n-dimensionale **Einheitsmatrix** $\mathbf{I}_n$ ist eine Matrix bei der alle Elemente auf der Diagonalen Eins und alle anderen Null sind.
- Ist $\mathbf{A}$ eine $n \times n$-Matrix, dann kann man deren k-**te Potenz** rekursiv definieren durch: $\mathbf{A}^0 = \mathbf{I}_n$ und $\mathbf{A}^n = \mathbf{A}\,\mathbf{A}^{n-1}$, $n = 1, 2, \ldots$.
- Matrizen werden in **MatLab** (steht für **Mat**rix **Lab**oratory) zur Darstellung von Bildern verwendet: dabei entspricht das (i, j)-Matrixelement dem Grauwert des entsprechenden Pixels (i, j). Der Nullpunkt befindet sich oben links, die erste Koordinate zeigt nach unten, die zweite nach rechts!

$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ ist anti-symm.

$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & & \vdots \\ 0 & 0 & 1 & & \\ & & & \ddots & 0 \\ 0 & 0 & 0 & & 1 \end{bmatrix}$

Octave (free, open source)

$4^{-1} = \frac{1}{4}$ aber $A^{-1} \neq \frac{1}{A}$ weil A eine Matrix ist

Definition (Inverse Matrix)

Die **inverse Matrix $\mathbf{A}^{-1}$** der (quadratischen) $n \times n$-Matrix $\mathbf{A} = [a_{ij}]$ hat die Eigenschaft

$$\mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}_n.$$

Einheitsmatrix

Example

Bestimmen Sie jeweils die inversen Matrizen (falls sie existieren!) durch direkte Rechnung.

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix} \text{ und } \mathbf{B} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}.$$

inv(A)

$$\left[\begin{array}{cc|cc} 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 1 \end{array}\right]$$

$$-\frac{1}{5}\begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix}$$

$$A^{-1} = \frac{1}{1 \cdot 1 - 2 \cdot 3}\begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix} = \frac{1}{-5}\begin{bmatrix} 1 & -2 \\ -3 & 1 \end{bmatrix}; \quad A^{-1}A = \frac{-1}{5}\begin{bmatrix} -5 & 0 \\ 0 & -5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbb{1}_2$$

Rechenregeln für Matrizen I

1a. $\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$

1b. Im Allg.: $\mathbf{AB}\neq\mathbf{BA}$

2a. $(\mathbf{A}+\mathbf{B})+\mathbf{C}=\mathbf{A}+(\mathbf{B}+\mathbf{C})$

2b. $(\mathbf{AB})\mathbf{C}=\mathbf{A}(\mathbf{BC})$

3a. $\mathbf{A}+\mathbf{0}=\mathbf{A}$

3b. $\mathbf{A}\mathrm{I}=\mathrm{I}\mathbf{A}=\mathbf{A}$, ($\mathbf{A}$ quadr.)

4. $\mathbf{AB}=\mathbf{0} \quad \not\Rightarrow \quad \mathbf{A}=\mathbf{0}$ oder $\mathbf{B}=\mathbf{0}$

5. $\mathbf{AB}=\mathbf{AC} \quad \not\Rightarrow \quad \mathbf{B}=\mathbf{C}$

6. $\lambda(\mathbf{A}+\mathbf{B}) = \lambda\mathbf{A}+\lambda\mathbf{B} \quad \lambda\in\mathbb{R}$

7. $\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}$

8. $(\mathbf{A}+\mathbf{B})\mathbf{C} = \mathbf{AC}+\mathbf{BC}$

Rechenregeln für Matrizen II

9. $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
10. $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
11. $(\mathbf{A}^{\mathsf{T}})^{\mathsf{T}} = \mathbf{A}$
12. $(\mathbf{A}+\mathbf{B})^{\mathsf{T}} = \mathbf{A}^{\mathsf{T}} + \mathbf{B}^{\mathsf{T}}$
13. $(\mathbf{AB})^{\mathsf{T}} = \mathbf{B}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}$
14. $(\mathbf{A}^{-1})^{\mathsf{T}} = (\mathbf{A}^{\mathsf{T}})^{-1}$

Für $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ mit $ad - bc \neq 0$ gilt $\mathbf{A}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

Übersicht

Null-Eins Matrizen

Null-Eins (Zero-One) Matrizen werden z.B. in der Graphentheorie verwendet.

Definition (Oder- und Und-verknüpfung)

Die **Oder-** und die **Und-verknüpfung** zweier $m \times n$-Matrizen $\mathbf{A}$ und $\mathbf{B}$ wird definiert durch

$$\mathbf{A} \vee \mathbf{B} = \left[a_{ij} \vee b_{ij}\right] \quad \text{und} \quad \mathbf{A} \wedge \mathbf{B} = \left[a_{ij} \wedge b_{ij}\right]$$

wobei $\vee$ und $\wedge$ für die boolschen Oder- und Und-Operationen stehen.

Example (Oder- und Und-verknüpfung)

Wie lauten die Oder- und Und-Verknüpfungen der folgenden Matrizen:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ und } \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

$A \vee B = \begin{bmatrix} 1\vee 0 & 0\vee 1 & 1\vee 0 \\ 0\vee 1 & 1\vee 1 & 0\vee 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ $\quad A \wedge B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

Definition (Boolsches Produkt)

Das **boolsche Produkt** der $m \times n$-Matrix $\mathbf{A} = [a_{ij}]$ mit der $n \times p$-Matrix $\mathbf{B} = [b_{ij}]$ wird definiert durch

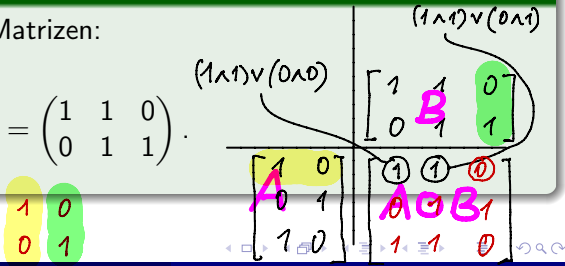
$$\mathbf{A} \odot \mathbf{B} = [c_{ij}] \text{ wobei } c_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee \cdots \vee (a_{in} \wedge b_{nj}).$$

mit $1 \leqslant i \leqslant m$ und $1 \leqslant j \leqslant p$.

Example (Boolsches Produkt)

Wie lauten das boolsches Produkt der folgenden Matrizen:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ und } \mathbf{B} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$



Null-Eins Matrizen (Fort.)

Definition (r-te Boolsches Potenz)

Die r-**te boolsche Potenz** der quadratischen $n \times n$-Matrix $\mathbf{A}$ ist definiert durch:

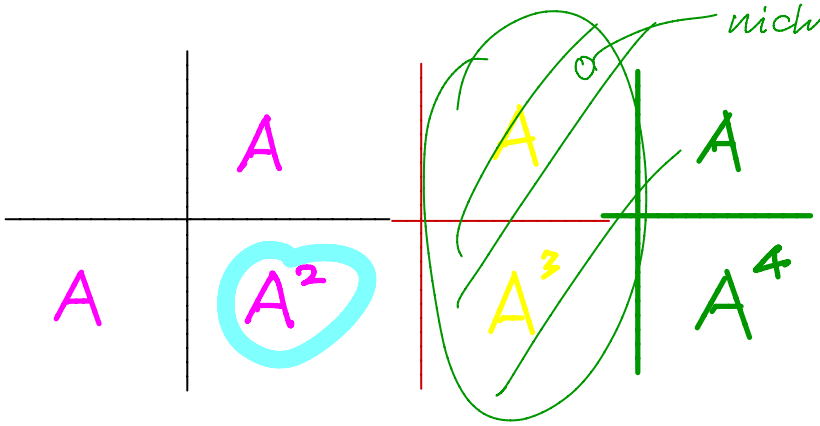
$$\mathbf{A}^{[r]} = \underbrace{\mathbf{A} \odot \mathbf{A} \odot \cdots \odot \mathbf{A}}_{r \text{ Faktoren}}.$$

wobei $\mathbf{A}^{[0]} = \mathbf{1}_n$ verwendet wird.

Example (2-te boolsche Potenz)

Wie lauten die 2-te boolsche Potenz der folgenden Matrix:

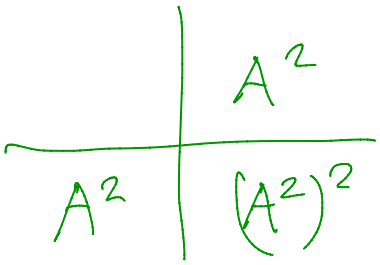
$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$



nicht nötig um A^4 zu berechnen!!

Um $A^4 = (A^2)^2$ zu berechnen:

- A^2
- $(A^2)^2$



Zusammenfassung

Wir haben in diesem Kapitel folgende Themen kennengelernt:

- Komplexität von Algorithmen, Wachstum von Funktionen (big-O)
- Zahlen und Division
- Zahlen und Algorithmen
- Ein bisschen Zahlentheorie
- Matrizen