Diskrete Mathematik I

Alexander May

Fakultät für Mathematik
Ruhr-Universität Bochum

Wintersemester 08/09

Organisatorisches

- Vorlesung: **Di 10-12** in HNC 30 , **Mi 12-14** in HZO 50 (4+2 SWS, 9 CP)
- Übung: **Di 8-10** in HZO 60 und **Mi 8-10** in NA 02/99
- Klausur: Ende Februar

- Zusammensetzung des Auditoriums?

Übungsbetrieb

- Assistentin: **Maike Ritzenhofen**
- Korrektoren: **M. Mansour Al-Sawadi, A. Meurer**
- Übungsaufgaben werden korrigiert.
  Abgabe: Mo 14:00, 2 Kästen NA 02
- Aufgaben 1+2 und 3+4 separat in je einen Kasten.
- Gruppenabgaben bis 4 Personen
- Bonussystem:
  1 Notenstufe für 50%, 2 Notenstufe für 75%
  Gilt nur, falls man die Klausur besteht!
- Musterlösungen zu Übungsaufgaben
- Präsenzaufgaben ohne Musterlösungen!

Themen

Thematische Gebiete

- Kombinatorik: Abzählprobleme, Ziehen von Elementen
- Graphen: Traversierung, Matching, Planarität, Färbung
- Algebra: Modulare und Polynomarithmetik
- Komplexität: Algorithmik, Laufzeitanalyse
- Wahrscheinlichkeit: Diskrete Verteilungen

Was bedeutet diskret?

- Intuitiv: Alles, was man mit Computern exakt darstellen kann.
- Gegenteil von analog
- Probleminstanzen sind aus Menge mit endlicher Kardinalität

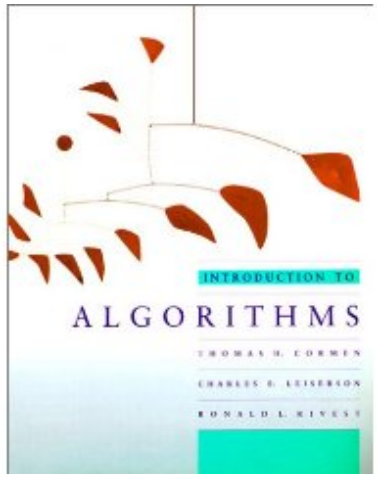
Literatur

Vorlesung richtet sich nach

- A. Steger, "Diskrete Strukturen, Band 1: Kombinatorik, Graphentheorie, Algebra", Springer Verlag
- T. Schickinger, A. Steger, "Band 2: Wahrscheinlichkeitstheorie"

Zusätzliche Literatur:

- Cormen, Leiserson, Rivest, Stein, “Introduction to Algorithmus”, MIT Press
- T. Ihringer, "Diskrete Mathematik", Teubner Verlag
- Aigner, "Diskrete Mathematik", Vieweg Studium, 2006



Notationen für Mengen

- $\mathbb{N}$: natürliche Zahlen ohne Null
- $\mathbb{N}_0$: natürliche Zahlen mit Null
- $\mathbb{Z}$: ganze Zahlen
- $\mathbb{Z}_n$: $\{0, 1, \ldots, n-1\}$
- $[n]$: $\{1, 2, \ldots, n\}$
- $\mathbb{Q}$: rationale Zahlen
- $\mathbb{R}$: reelle Zahlen

Operationen auf Mengen

- Vereinigung $A \cup B := \{x \mid x \in A \text{ oder } x \in B\}$
- Schnittmenge $A \cap B := \{x \mid x \in A \text{ und } x \in B\}$
- Differenz $A \backslash B := \{x \mid x \in A \text{ und } x \notin B\}$
- Symmetrische Differenz $A \triangle B := (A \backslash B) \cup (B \backslash A)$
- Kartesisches Produkt $A \times B := \{(a, b) \mid a \in A \text{ und } b \in B\}$
- Potenzmenge $\mathcal{P}(M) := \{N \mid N \subseteq M\}$

Bsp: M={rot,blau}, $\mathcal{P}(M)$={ $\emptyset$,{rot},{blau}, {rot,blau}}

Relation

Definition Relation

Eine Relation zwischen A und B ist eine Teilmenge $R \subseteq A \times B$.
Falls A=B, spricht man von einer Relation auf A.

Eigenschaften von Relationen auf einer Menge

Reflexiv:	$\forall a \in A : (a, a) \in R$
Symmetrisch:	$\forall a, b \in A : (a, b) \in R \Rightarrow (b, a) \in R$
Antisymetrisch:	$\forall a, b \in A : (a, b) \in R \wedge (b, a) \in R \Rightarrow a = b$
Transitiv:	$\forall a, b, c \in A : (a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R$

Bsp.:

- $R_1 := \{(a, b) \in \mathbb{N}^2 \mid a \text{ teilt } b\}$: r,a,t (partielle Ordnung)
- $R_2 := \{(a, b) \in \mathbb{Z}^2 \mid a = b \bmod 3\}$: r,s,t (Äquivalenzrelation)
- $R_3 := \{(a, b) \in \mathbb{Z}^2 \mid a \text{ teilt } b\}$: r,t (Quasiordnung)
- $R_4 := \{(a, b) \in [8]^2 \mid a = b \bmod 3, a \leq b\}$: r,a,t

Abbildungen/Funktionen

Definition Abbildung/Funktion

Eine Abbildung/Funktion ist eine Relation $R \subseteq A \times B$, falls für alle $a \in A$ gilt

$$|\{b \in B \mid (a, b) \in R\}| = 1.$$

Wir schreiben $f : A \to B, a \mapsto f(a)$. Die Menge der *Urbilder* eines Elements $b \in B$ bezeichnen wir mit $f^{-1}(b) := \{a \in A \mid f(a) = b\}$.

Wir definieren für $A' \subseteq A, B' \subseteq B$ eine Erweiterung auf Mengen:

$$\begin{aligned} f(A') &= \bigcup_{a \in A'} \{f(a)\} \\ f(B') &= \bigcup_{b \in B'} f^{-1}(b) \end{aligned}$$

Eigenschaften von Funktionen, Isomorphie

Definition Eigenschaften von Funktionen

Sei f eine Funktion. Wir bezeichnen f als

1. *injektiv* gdw für alle $b \in B : |f^1(b)| \leq 1$.
2. *surjektiv* gdw für alle $b \in B : |f^1(b)| \geq 1$.
3. *bijektiv* gdw f injektiv und f surjektiv ist.

Isomorphismus

Seien $R_1 \subseteq A_1^2$, $R_2 \subseteq A_2^2$ Relationen. R_1 und R_2 heißen *isomorph* gdw eine bijektive Funktion $f : A_1 \rightarrow A_2$ existiert, so dass für alle $(a, b) \in A_1^2$:

$$(a, b) \in R_1 \Leftrightarrow (f(a), f(b)) \in R_2.$$

Indirekter Beweis/ Widerspruchsbeweis

Satz

Sei $n \in \mathbb{N}$. Dann gilt

$$n^2 \text{ gerade } \Rightarrow n \text{ gerade.}$$

Beweis:

- Kontraposition: $(A \Rightarrow B) \Leftrightarrow (\neg B \Rightarrow \neg A)$
- D.h. es genügt zu zeigen: n ungerade $\Rightarrow n^2$ ungerade
- Wir schreiben n in der Form $n = 2k + 1, k \in \mathbb{N}_0$
- Daher gilt $n^2 = 4k^2 + 4k + 1 = 2(k^2 + 2k) + 1$.
- D.h. n^2 ist ungerade.

Induktionsbeweis

Satz Primfaktorzerlegung

Jede natürliche Zahl $n \geq 2$ lässt sich als Produkt von Primzahlen darstellen.

Beweis: Induktion über n

- (IV) Induktionsverankerung: n=2 prim.
- (IA) Induktionsannahme: Jede Zahl $\leq n$ lässt sich als Produkt von Primzahlen darstellen.
- (IS) Induktionsschrift n$\rightarrow$ n+1: Fallunterscheidung
  - Fall 1: $n+1$ prim, d. h. $n+1$ ist Produkt von Primzahlen.
  - Fall 2: $n+1$ zusammengesetzt, d. h. $n+1 = a \cdot b$ mit $1 < a, b \leq n$. Wende Induktionsannahme auf a und b an.

Widerspruchsbeweis

Satz Anzahl Primzahlen

Es gibt unendlich viele Primzahlen.

Annahme: $\exists$ endlich viele Primzahlen $p_1 ..., p_n$ (n beliebig, aber fest).

- Setze $m = 1 + \prod_{i=1}^{n} p_i$.
- Es gilt $m = 1 \mod p_i$ für $i = 1, \ldots, n$.
- D.h. p_i teilt m nicht (wegen $p_i \geq 2$).
- Insbesondere ist $m \neq p_i$, $i = 1, \ldots, n$
- Daraus folgt, dass m prim.
- Damit existieren mindestens $n + 1$ Primzahlen.
  (Widerspruch: Nach Annahme existieren genau n Primzahlen.)

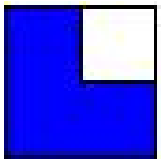
Induktionsbeweis

Satz Kacheln eines Schachbretts

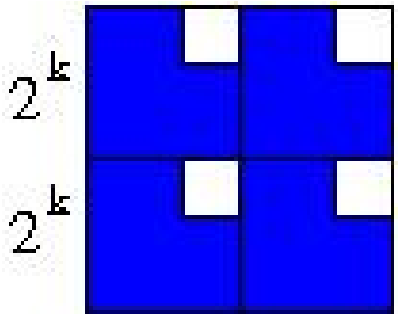
Jedes Schachbrett mit Seitenlänge 2^k lässt sich durch 3-Felder große, L-förmige Teile so kacheln, dass die rechte obere Ecke frei bleibt.

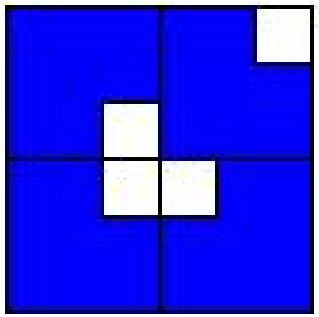
Beweis: Induktion über k

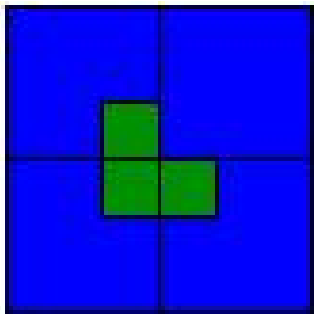
- IV (k=1)



- IA: Satz sei korrekt bis k.
- IS (k$\rightarrow$ k+1):







Landau-Notation $\mathcal{O}$

Definition Landau Notation $\mathcal{O}$

Seien $f(n), g(n)$ Funktionen. Wir schreiben $f(n) = \mathcal{O}(g(n))$ gdw

$$\exists c > 0, n_0 \in \mathbb{N}\, \forall n \geq n_0 : |f(n)| \leq c \cdot |g(n)|.$$

Alternativ: $f(n) = \mathcal{O}(g(n)) \Leftrightarrow \lim_{n\to\infty} \sup \frac{|f(n)|}{|g(n)|} < \infty$

Beispiele:

- $3n^2 + n + 2 = \mathcal{O}(n^2)$
- $3n^2 + n + 2 = \mathcal{O}(n^3 \log n)$
- $\sum_{i=1}^{n} i = \mathcal{O}(n^2)$
- $\sum_{i=1}^{d} a_i n^i = \mathcal{O}(n^d)$
- $\sum_{i=1}^{n} \frac{1}{i} = \mathcal{O}(\log n)$
- $\log_2 n = \mathcal{O}(\log_e n)$

Landau-Notation Ω

Definition Landau Notation Ω

Seien $f(n), g(n)$ Funktionen. Wir schreiben $f(n) = \Omega(g(n))$ gdw

$$\exists c > 0, n_0 \in \mathbb{N}\ \forall n \geq n_0 : |f(n)| \geq c \cdot |g(n)|.$$

Alternativ: $f(n) = \mathcal{O}(g(n)) \Leftrightarrow \lim_{n\to\infty} \inf \frac{|f(n)|}{|g(n)|} > 0$

Beispiele:

- $3n^2 + n + 2 = \Omega(n^2)$
- $3n^2 + n + 2 = \Omega(n \log n)$
- $\sum_{i=1}^{n} i = \Omega(n^2)$
- $\sum_{i=1}^{d} a_i n^i = \Omega(n^d)$
- $\sum_{i=1}^{n} \frac{1}{i} = \Omega(\log n)$
- $\log_2 n = \Omega(\log_e n)$

Landau-Notation Θ

Definition Landau Notation Θ

Seien $f(n), g(n)$ Funktionen. Wir schreiben $f(n) = \Theta(g(n))$ gdw

$$f(n) = \mathcal{O}(g(n)) \text{ und } f(n) = \Omega(g(n)).$$

Bsp:

- $\sum_{i=1}^{d} a_i n^i = \Theta(n^d)$
- $\log_2 n = \Theta(\log_e n)$

Landau-Notation o, ω

Definition Landau Notation o

Seien $f(n), g(n)$ Funktionen. Wir schreiben $f(n) = o(g(n))$ gdw

$$\forall c > 0 \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : |f(n)| \leq c \cdot |g(n)|.$$

Alternativ: $f(n) = o(g(n)) \Leftrightarrow \lim_{n \to \infty} \frac{|f(n)|}{|g(n)|} = 0.$

Bsp:

- $n = o(n^2), \; 10n^2 / \log\log n = 0(n^2)$

Definition Landau Notation ω

Seien $f(n), g(n)$ Funktionen. Wir schreiben $f(n) = \omega(g(n))$ gdw

$$\forall c > 0 \; \exists n_0 \in \mathbb{N} \; \forall n \geq n_0 : |f(n)| \geq c \cdot |g(n)|.$$

Alternativ: $f(n) = \omega(g(n)) \Leftrightarrow \lim_{n \to \infty} \frac{|f(n)|}{|g(n)|} \to \infty.$

Bsp:

- $n^2 = \omega(n), \; 10n^2 \log\log n = \omega(n^2)$

Ziehen von Elementen

Kombinatorik: Bestimmung der Anzahl Anordnungsmöglichkeiten einer (endlichen) Menge von Objekten.

Bsp: Ziehen 2 Elemente aus einer 3-elementigen Menge {1,2,3}.

	geordnet	ungeordnet
mit Zurücklegen	(1,1), (1,2), (1,3), (2,1),(2,2),(2,3), (3,1),(3,2),(3,3)	{1,1},{1,2}, {1,3},{2,2}, {2,3}, {3,3}
ohne Zurücklegen	(1,2),(1,3), (2,1),(2,3), (3,1),(3,2)	{1,2}, {1,3}, {2,3}

Frage: Wieviele Möglichkeiten bestehen für das Ziehen von k Elementen aus einer n-elementigen Menge?

mit Zurücklegen, geordnet

$$(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)$$

- Anzahl Möglichkeiten für 1. Element: n
- Anzahl Möglichkeiten für 2. Element: n

$\vdots$

- Anzahl Möglichkeiten für k. Element: n

Gesamt: n^k Möglichkeiten

Bsp: Für jede EC-Karte gibt es 10^4 mögliche PINs.

ohne Zurücklegen, geordnet

$$(1,2),(1,3),(2,1),(2,3),(3,1),(3,2)$$

- Anzahl Möglichkeiten für 1. Element: n
- Anzahl Möglichkeiten für 2. Element: n-1

$$\vdots$$

- Anzahl Möglichkeiten für k. Element: n-(k-1)

Gesamt: $n \cdot (n-1) \cdot \ldots (n-k+1) = \prod_{i=0}^{k-1} n - i =: n^{\underline{k}}$
Sprechweise: k-te untere Faktorielle von n

Bsp: Anzahl vierstelliger Dezimalzahlen mit verschiedenen Ziffern:

$$10^{\underline{4}} = 5040.$$

Geordnetes Ziehen aller Elemente

Beim geordneten Ziehen aller Elemente betrachten wir den Spezialfall

$$n^{\underline{n}} = \prod_{i=0}^{n-1} n - i = \prod_{i=1}^{n} i := n!$$

Wir definieren weiterhin

- $n^{\underline{0}} = \prod_{i=0}^{-1} n - i := 1$
- $0^0 = 0! := 1$

Bsp: Anzahl Worte der Länge 3 über $\{a, b, c\}$ mit verschiedenen Buchstaben ist $3! = 6$:

- abc,acb,bac,bca,cab,cba

ohne Zurücklegen, ungeordnet

ungeordnet:	{1,2},	{1,3},	{2,3}
geordnet:	(1,2),(2,1),	(1,3),(3,1),	(2,3),(3,2)

- Anzahl geordneter Teilmengen: $n^{\underline{k}}$
- Fassen k-Tupel mit gleichen Elementen zusammen.
- Wieviele k-Tupel mit gleichen Elementen gibt es?
  Anordnung von k-Teilmengen: $k!$

Gesamt: $\frac{n^{\underline{k}}}{k!} = \frac{n\cdot(n-1)\cdot\ldots\cdot(n-(k-1))}{k!} = \frac{n!}{k!(n-k)!} =: \binom{n}{k}$

Bsp:

- Anzahl der Strings $s \in \{0,1\}^5$ mit genau 3 Nullen: $\binom{5}{3}$.
- Anzahl des Auftauchens von a^2b^2 in $(a+b)^4$: $\binom{4}{2}$.

mit Zurücklegen, ungeordnet

$$\{1,1\},\{1,2\},\{1,3\},\{2,2\},\{2,3\},\{3,3\}$$

Multimenge:

- Einzelne Elemente dürfen mit Vielfachheiten vorkommen.
- $M = \{1,1,2,3,3,3\}$ ist Multimenge über $G = \{1,2,3,4,5\}$.
  - Vielfachheit von 1 in M ist 2.
  - Kardinalität von M: Anzahl Elemente mit Vielfachheit, d.h. $|M| = 6$.

Kodierung einer Multimenge:

- Definiere Ordnung auf Grundmenge G, z. B. 1,2,3,4,5.
- Für jedes Element e der Multimenge:
  - Falls e mit Vielfachheit v(e) auftaucht, notiere v(e) Sterne $*$.
  - Trenne einzelne Elemente mit einem Trennstrich $|$.
- Bsp: Kodierung von M über G:

$$**|*|***||.$$

- Kodierungen entsprechen eindeutig den Multimengen, d. h. die Kodierungsabbildung ist ein Isomorphismus.

mit Zurücklegen, ungeordnet

Frage: Wieviele verschiedene Kodierungen gibt es?

- Kodierung besitzt $n + k - 1$ Zeichen
  - k Sterne $*$: Wir ziehen k Elemente.
  - $n - 1$ Trennstriche $|$: Wir müssen n verschiedene Elemente trennen.
- Es müssen k Sterne an beliebigen Stellen der Kodierung platziert werden:
  - Jede Kombination von k Sternen und n-1 Trennstrichen entspricht einer Multimenge.
- Ziehen k-elementige Menge aus $(n + k - 1)$-elementiger Menge.

Gesamt:

$$\binom{n+k-1}{k}$$

Bsp: 25 Eissorten, wir kaufen 3 Kugeln: $\binom{27}{3}$ Möglichkeiten

Zusammenfassung: Ziehe k aus n Elementen

	geordnet	ungeordnet
mit Zurücklegen	n^k	$\binom{n+k-1}{k}$
ohne Zurücklegen	$n^{\underline{k}}$	$\binom{n}{k}$

Kombinatorische Beweisprinzipien

Satz Binomischer Lehrsatz

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

Beweis

- Multipliziere $(a+b)^n$ aus: $(a+b)\cdot(a+b)\cdot\ldots\cdot(a+b)$.
- Aus jedem der n Faktoren wird entweder a oder b verwendet.
- Alle Summanden sind von der Form $a^k b^{n-k}$, $k = 0, \ldots, n$.
- Multiplikation ist kommutativ, d. h. $b \cdot a \cdot b^2 \cdot a = a^2 b^3$.
- Für $a^k b^{n-k}$ muss genau k-mal ein a verwendet werden.
- Ziehen k Positionen für a. Das Ziehen erfolgt
  - ohne Zurücklegen (jede Position darf nur einmal gezogen werden)
  - ungeordnet (aufgrund der Kommutativität der Multiplikation)
- D.h. der Summand $a^k b^{n-k}$ taucht $\binom{n}{k}$-mal als Summand auf.

Summenregel

Lemma Summenregel für disjunkte Vereinigung

Seien $S_1, \ldots, S_n$ disjunkte Mengen. Dann gilt für deren Vereinigung $S = \biguplus_{i=1}^{n} S_i$

$$|S| = \sum_{i=1}^{n} |S_i|.$$

Bsp: Sei $S = \{S_i \subseteq [10] \mid |S_i| = 5 \text{ und } |S_i \cap \{1,2\}| = 1\} \subseteq \mathcal{P}([10])$. Bestimme $|S|$.

- Wir definieren $S = S_1 \cup S_2$ für folgende Mengen $S_1, S_2 \subseteq \mathcal{P}(S)$.
  - S_1: Enthält Mengen $S_{1i} \subseteq [10]$ mit Element 1 und 4 weiteren Elemente aus {3,. . .,10}.
  - S_2: Enthält Mengen $S_{2j} \subseteq [10]$ mit Element 2 und 4 weiteren Elemente aus {3,. . .,10}.
- Die Mengen S_1, S_2 sind disjunkt, da die Elemente S_{1i} und S_{2j} paarweise verschieden sind.
- Die Mengen S_1 und S_2 enthalten jeweils $\binom{8}{4}$ Elemente.
- Mit der Summenregel folgt $|S| = |S_1| + |S_2| = 2 \cdot \binom{8}{4}$.

Produktregel

Lemma Produktregel für das kartesische Produkt

Seien $S_1, \ldots, S_n$ Mengen. Dann gilt für das kartesische Produkt $S = S_1 \times \ldots \times S_n$, dass

$$|S| = \prod_{i=1}^{n} |S_i|.$$

Bsp: Bestimmen die Anzahl fünfstelliger Zahlen, deren i-te Ziffer durch $i + 1$ teilbar ist.

- 1. Ziffer: $S_1 = \{0, 2, 4, 6, 8\}$ mit $|S_1| = 5$.
- 2. Ziffer: $S_2 = \{0, 3, 6, 9\}$ mit $|S_2| = 4$.
- 3. Ziffer: $S_3 = \{0, 4, 8\}$ mit $|S_3| = 3$.
- 4. Ziffer: $S_4 = \{0, 5\}$ mit $|S_2| = 2$.
- 5. Ziffer: $S_5 = \{0, 6\}$ mit $|S_1| = 2$.

- Für $S = S_1 \times \ldots \times S_5$ erhalten wir mit Produktregel $|S| = 5! \cdot 2 = 240$.

Produktregel

Bsp aus der Laufzeitanalyse:

Algorithmus Verschachtelte Schleifen

1. x $\leftarrow$ 0;
2. for i=1 to 5 do
3. for j=1 to 10 do
4. x $\leftarrow$ x+1;

- $S_1 = \{1, 2, 3, 4, 5\}$, $S_2 = \{1, 2, \ldots, 10\}$
- Schleife in Schritt 4 durchläuft alle (i, j) mit $i \in S_1$, $j \in S_2$.
- Produktregel liefert für $S = S_1 \times S_2$ eine Anzahl von $|S| = |S_1| \cdot |S_2| = 5 \cdot 10$ Schleifendurchläufen.

Gleichheitsregel

Lemma Gleichheitsregel für Bijektionen

Seien S, T Mengen und $f : S \to T$ eine Bijektion. Dann gilt

$$|S| = |T|.$$

Bsp:

Algorithmus Wieder verschachtelte Schleifen

1. x $\leftarrow$ 0;
2. for i=1 to 5 do
3. for j=1 to 10 do
4. x $\leftarrow$ x+1;

- Sei $S = [5] \times [10]$. Die Variablen (i, j) nehmen alle Werte aus S an.
- Die Abbildung $f : [5] \times [10] \to [50], (i, j) \mapsto x$ in Schritt 4 ist bijektiv.
- x nimmt in Schritt 4 Werte aus [50] an.
- Gleichheitsregel: Anzahl Schleifendurchläufe ist $|S| = |T| = 50$.

Weiteres Beispiel zur Gleichheitsregel

Bereits bekanntes Beispiel: Ziehen mit Zurücklegen, ungeordnet

- Es existiert eine bijektive Abbildung f zwischen
  - Multimengen der Kardinalität k über einer Grundmenge mit n Elementen und
  - Kodierungen in Form von k Sternen $*$ und $n-1$ Trennstrichen $|$.
- Anstatt die Anzahl der möglichen Multimengen zu zählen, haben wir die Anzahl der Kodierungen gezählt.

Doppeltes Abzählen

Lemma Doppeltes Abzählen

Seien S, T Mengen und $R \subseteq S \times T$ eine Relation. Dann gilt

$$\sum_{s \in S} |\{t \in T \mid (s,t) \in R\}| = \sum_{t \in T} |\{s \in S \mid (s,t) \in R\}|.$$

Relation in Form einer Matrix: Dann gilt Zeilensumme=Spaltensumme.

Algorithmus Verschachtelte Schleife, die Dritte

1. x $\leftarrow$ 0;
2. for i=1 to 5 do
3. j=1 to i do
4. x $\leftarrow$ x+1;

- Definiere $R = \{(i,j) \in [5]^2 \mid j \leq i\}$.
- Zeilensumme: $\sum_{i \in [5]} |\{j \in [5] \mid j \leq i\}| = 1+2+3+4+5$.
- Spaltensumme: $\sum_{j \in [5]} |\{i \in [5] \mid i \geq j\}| = 5+4+3+2+1$.

Schubfachprinzip (Pigeonhole principle)

Satz Schubfachprinzip

Sei $f : X \to Y$ eine Abbildung mit $|X| > |Y|$.
Dann gibt es ein $y \in Y$ mit

$$|f^{-1}(y)| \geq 2.$$

Anders gesagt: Verteilt man n Elemente auf m, $m < n$, Fächer, so gibt es stets ein Fach, das mehr als ein Element enthält.

Bsp:
Unter 367 Leuten gibt es stets zwei Personen, die am gleichen Tag Geburtstag haben.

Schubfachprinzip

Satz

In jeder Menge P von Personen gibt es stets zwei Personen, die die gleiche Anzahl von anderen Personen in P kennen.

Beweis

- Sei $P = \{p_1, p_2, \ldots, p_n\}$.
- Wir setzen voraus, dass die Relation „kennen“ symmetrisch ist, d.h. für alle p_i, p_j gilt: p_i kennt $p_j \Leftrightarrow p_j$ kennt p_i.
- Betrachte $f : P \to \mathbb{Z}_n$ mit $f(p_i) = j \Leftrightarrow p_i$ kennt j Personen.
- Problem: $|P| = |\mathbb{Z}_n| = n$, d. h. f könnte Bijektion sein.
- Annahme: f bijektiv.
  - $\exists p_j, p_i$ mit $f(p_i) = 0$ und $f(p_j) = n - 1$.
  - Widerspruch: p_j kennt jeden, insbesondere p_i,aber p_i kennt keinen.
- f ist nicht surjektiv und wegen $|\bigcup_{p \in P}\{f(p)\}| < n$ nicht injektiv (Anwendung des Schubfachprinzips).
- Es gibt $i, j \in [n]$, $i \neq j$ mit $f(p_i) = f(p_j)$.

Verallgemeinertes Schubfachprinzip

Satz Verallgemeinertes Schubfachprinzip

Sei $f : X \to Y$. Dann gibt es ein $y \in Y$ mit $|f^{-1}(y)| \geq \left\lceil \frac{|X|}{|Y|} \right\rceil$.

Anders gesagt: Verteilt man n Elemente auf m Fächer, so gibt es ein Fach, dass mindestens $\lceil \frac{n}{m} \rceil$ Elemente enthält.

Beweis

- Annahme: Für alle $y \in Y$ gilt: $|f^{-1}(y)| \leq \left\lceil \frac{|X|}{|Y|} \right\rceil - 1$
- Damit folgt $\sum_{y \in Y} |f^{-1}(y)| \leq |Y| \cdot \left(\left\lceil \frac{|X|}{|Y|} \right\rceil - 1 \right) < |Y| \cdot \frac{|X|}{|Y|} = |X|$.
- Widerspruch durch das Prinzip des Doppelten Abzählens:

$$\begin{aligned} \text{Spaltensumme} &= \sum_{y \in Y} |f^{-1}(y)| \\ &< |X| = \sum_{x \in X} 1 = \sum_{x \in X} |\{f(x)\}| = \text{ Zeilensumme} \end{aligned}$$

Anwendung des Verallgemeinerten Schubfachprinzips

Szenario:

- Teams $t_1, \ldots, t_n$ spielen ein Turnier jeder gegen jeden.
- Gewinner erhalten 2 Punkte, bei Remis erhalten beide 1 Punkt.
- **Frage:** Wieviele Punkte hat der Gewinner mindestens?

- Definiere Punkteverteilungs-Funktion $f : [2]\, x\, [n]^2 \rightarrow [n]$ mit
  $f(1,i,j) = f(2,i,j) = i \quad \Leftrightarrow \quad i$ schlägt j
  $f(1,i,j) = f(2,i,j) = j \quad \Leftrightarrow \quad j$ schlägt i
  $f(1,i,j) = i$ und $f(2,i,j) = j \quad \Leftrightarrow$ Remis
- Urbildraum: $|X| = 2 \cdot \binom{n}{2}$, Bildraum: $|Y| = n$
- Verallg. Schubfachprinzip: Der Sieger besitzt mindestens

$$\left\lceil \frac{|X|}{|Y|} \right\rceil = \frac{2n(n-1)}{2n} = n - 1 \text{ Punkte.}$$

Verallg. Schubfachprinzip – Ramsey-Theorie

Satz Ramsey

In jeder Gruppe P von 6 Personen gibt es entweder 3 Personen, die sich alle kennen oder 3 Personen, die sich alle nicht kennen.

Beweis

- Definieren $f : \{p_2, \ldots, p_6\} \to \{0, 1\}$ mit $f(p_i) = 1 \Leftrightarrow p_1$ kennt p_i.
- Das Verallg. Schubfachprinzip liefert:

Fall 1: p_1 kennt mindestens $\lceil 5/2 \rceil = 3$ Personen oder
Fall 2: p_1 kennt mindestens $\lceil 5/2 \rceil = 3$ Personen nicht.

- Betrachte nur Fall 1 (Fall 2 analog): OBdA p_1 kennt p_2, p_3, p_4.
- Weitere Fallunterscheidung:

Fall 1a: p_2, p_3, p_4 kennen sich nicht untereinander nicht.
Dann gibt es 3 Personen die sich nicht kennen.
Fall 1b: Von p_2, p_3, p_4 kennen sich mindestens 2 Personen, oBdA seien dies p_2, p_3. Dann kennen sich die 3 Personen p_1, p_2, p_3.

Prinzip der Inklusion- Exklusion

Ziel: Zählen von Elementen in nicht-disjunkten Mengen.

- **2 Mengen** A_1, A_2: Zählen zunächst die Elemente in A_1.
- Addieren dazu die Anzahl der Elemente in A_2.
- Zählen damit den Schnitt von A_1 und A_2 doppelt.

$$|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$$

- **3 Mengen:** Zählen die Elemente in A_1, A_2 und A_3 einzeln.
- Subtrahieren Anzahl der Elemente in $A_1 \cap A_2$, $A_1 \cap A_3$ und $A_2 \cap A_3$.
- Damit wurden die Elemente in $A_1 \cap A_2 \cap A_3$ dreimal gezählt und dreimal abgezogen.

$$\begin{aligned} |A_1 \cup A_2 \cup A_3| \;=\; & |A_1| + |A_2| + |A_3| \\ & - (|A_1 \cap A_2| + |A_1 \cap A_3| + |A_2 \cap A_3|) \\ & + |A_1 \cap A_2 \cap A_3| \end{aligned}$$

Prinzip der Inklusion-Exklusion

Satz Inklusion-Exklusion

Seien $A_1, \ldots, A_n$ endliche Mengen. Dann gilt

$$|\bigcup_{i=1}^{n} A_i| = \sum_{r=1}^{n} (-1)^{r-1} \sum_{1 \le i_1 < \ldots < i_r \le n} |\bigcap_{j=1}^{r} A_{i_j}|.$$

Beweis

- Idee: Zeigen, dass jedes Element a genau einmal gezählt wird.
- Sei a in k Mengen A_i enthalten.
- a kommt im Schnitt der A_{i_j} vor, gdw $a \in A_{i_j}$ für alle i_j.
- Damit kommt a in genau $\binom{k}{r}$ Schnittmengen vor.
- Insgesamt zählen wir damit jedes a mit der Häufigkeit

$$\begin{aligned} \sum_{r=1}^{n} (-1)^{r-1} \binom{k}{r} &= -\sum_{r=1}^{n} \binom{k}{r} (-1)^r = 1 - \sum_{r=0}^{n} \binom{k}{r} (-1)^r \\ &= 1 - \sum_{r=0}^{k} \binom{k}{r} (-1)^r 1^{k-r} = 1 - (1-1)^k = 1. \end{aligned}$$

Anwendung Inklusion-Exklusion

Bsp: Sei $A = \{x \in [100] \mid (2|x) \text{ oder } (3|x) \text{ oder } (5|x)\}$. Bestimme $|A|$.

- Definieren $A_k := \{n \in [100] \mid k \text{ teilt } n\}$. Damit gilt $A = A_2 \cup A_3 \cup A_5$.
- Für die Kardinalität von A_k erhalten wir $|A_k| := \left\lfloor \frac{100}{k} \right\rfloor$.
- Außerdem gilt $A_i \cap A_j = A_{\text{kgV}\{i,j\}}$
- Damit erhalten wir insgesamt

$$\begin{aligned} |A| &= |A_2 \cup A_3 \cup A_5| \\ &= |A_2| + |A_3| + A_5| - (|A_6| + |A_{10}| + |A_{15}|) + |A_{30}| \\ &= 50 + 33 + 20 - (16 + 10 + 6) + 3 = 74 \end{aligned}$$

Permutationen

Definition Permutation, Fixpunkt, Symmetrische Gruppe

Sei $\pi : A \to A$ eine Funktion.

1. Wir bezeichnen π als *Permutation* gwd π bijektiv ist.
2. Für eine Permutation π bezeichnen wir alle $a \in A$ mit $\pi(a) = a$ als *Fixpunkte*. π heißt *fixpunktfrei*, falls π keine Fixpunkte enthält.
3. Die Menge der Permutationen auf $A = [n]$ bezeichen wir als *symmetrische Gruppe* $\mathcal{G}_n$.

- Es gilt $|\mathcal{G}_n| = n!$.
- Schreibweise für ein $\pi \in \mathcal{G}_5$: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 4 & 1 \end{pmatrix}$
- Das Element 4 ist der einzige Fixpunkt der Permutation π.
- Bei fester Anordnung von $a_1, \ldots, a_n \in A$ würde die zweite Zeile

$$(\pi(a_1)\pi(a_2)\ldots\pi(a_n))$$

genügen. Vorsicht: Verwechslungsgefahr mit Zyklenschreibweise.

Fixpunktfreie Permutationen

Definition Derangementzahl

Wir bezeichnen mit D_n die Anzahl fixpunktfreier Permutationen in $\mathcal{G}_n$. D_n heißt auch *Derangementzahl*. Mit ζ_n bezeichnen wir die Anzahl der Permutationen in $\mathcal{G}_n$ mit mindestens einem Fixpunkt.

- Offenbar gilt: $D_n = |\mathcal{G}_n| - \zeta_n$.
- Um D_n zu bestimmen, genügt es ζ_n zu bestimmen.
- A_i bezeichne die Menge der Bijektionen $\{\pi \in \mathcal{G}_n \mid \pi(i) = i\}$.
- Es gilt $\zeta_n = \bigcup_{i=1}^{n} A_i$.

Derangementzahl D_n

Satz Derangementzahl D_n

Für die Derangementzahl D_n gilt

$$D_n = n! \cdot \sum_{r=0}^{n} (-1)^r \frac{1}{r!}$$

- Anwendung des Inklusion-Exklusion Prinzips liefert $\zeta_n = |\bigcup_{i=1}^{n} A_i| = \sum_{r=1}^{n} (-1)^r \sum_{1 \le i_1 < \ldots < i_r \le n} |\bigcap_{j=1}^{r} A_{i_j}|$.
- Schnittmengen A_{i_j} beinhalten π mit $\pi(i_j) = i_j$ für alle $j = 1, \ldots, r$.
- Alle anderen $n - r$ Elemente dürfen von π beliebig abgebildet werden. Dafür gibt es $(n - r)!$ Möglichkeiten, die Anzahl der Permutation auf $[n] \setminus \{i_1, \ldots, i_r\}$.
- Wir erhalten $\zeta_n = |\bigcup_{i=1}^{n} A_i| = \sum_{r=1}^{n} (-1)^{r-1} \binom{n}{r} \cdot (n - r)! = \sum_{r=1}^{n} (-1)^{r-1} \frac{n!}{r!}$.
- Damit gilt $D_n = n! - \zeta_n = n!(1 + \sum_{r=1}^{n} (-1)^r \frac{n!}{r!}) = n! \cdot \sum_{r=0}^{n} (-1)^r \frac{1}{r!}$.

Zyklen von Permutationen

Definition Zyklus einer Permutation

Sei $\pi \in \mathcal{G}_n$. Wir bezeichnen $(i_1 \dots i_t)$ mit $i_j \in [n]$ als *Zyklus der Länge t* in π falls

$$\pi(i_j) = i_{j+1} \text{ für } 1 \leq j < t \text{ und } \pi(i_t) = i_1.$$

Bsp:

- $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 5 & 7 & 1 & 6 & 4 \end{pmatrix}$
- (1235) ist Zyklus der Länge 4 in π.
- Man beachte, dass $(1235) = (2351) = (3512) = (5123)$.
- Hingegen gilt $(1234) \neq (2135)$.
- (47) und (6) sind Zyklen der Längen 2 und 1 in π.
- Wir können π schreiben als $\pi = (1235)(47)(6)$.

Stirlingzahl erster Art

Definition Stirlingzahl 1. Art

Die *Stirlingzahl erster Art* $s_{n,k}$ bezeichne die Anzahl von Permutationen $\pi \in \mathcal{G}_n$ mit genau k Zyklen.

Es gilt

- $s_{n,k} = 0$ für $n < k$
- $s_{n,0} = 0$
- $s_{0,0} := 1$ nach Definition

Satz Summe der Stirlingzahlen für festes n

$\sum_{k=1}^{n} s_{n,k} = n!$

Beweis

- Jede Permutation besitzt mindestens 1 und höchstens n Zykel.
- Die Anzahl aller Permutation $\pi \in \mathcal{G}_n$ ist $n!$.
- Da $s_{n,k}$ und $s_{n,\ell}$ für $k \neq \ell$ disjunkt sind, folgt der Satz durch Anwendung der Summenregel.

Berechnung der ersten Stirlingzahl

Satz Berechnung $s_{n,k}$

Sei $s_{n,k} = \{\pi \in \mathcal{G}_n \mid \pi \text{ besitzt } k \text{ Zyklen.}\}$. Für alle $k, n \in \mathbb{N}$ gilt

$$s_{n,k} = s_{n-1,k-1} + (n-1)s_{n-1,k}.$$

Beweis

- Fallunterscheidung für $\pi \in \mathcal{G}_n$

Fall 1: n ist in π in einem Zyklus (n) der Länge 1. Dann gibt es für die restlichen $k-1$ Zyklen genau $s_{n-1,k-1}$ Möglichkeiten.

Fall 2: n ist in π in einem Zyklus der Länge mindestens 2. Wir können π darstellen, indem wir n in einen Zyklus von $\pi' \in \mathcal{G}_{n-1}$ mit k Zyklen einfügen. Es gibt $s_{n-1,k}$ Möglichkeiten für π' und $n-1$ Möglichkeiten zum Einfügen von n in einen der Zyklen.

- Da beide Fälle disjunkt sind, folgt der Satz aus der Summenregel.

Beispielkonstruktion: Rekusives Splitten von $s_{4,2}$

Bsp: $s_{4,2}$

- Fall 1: π enthält einen Zyklus (n) und einen Zyklus über [3].
  - $(123)(4), (132)(4)$.
- Fall 2: Betrachten 2 Zykel über [3] und fügen 4 ein.
  - $(12)(3) : (412)(3), (142)(3), (12)(43)$
  - $(13)(2) : (413)(2), (143)(2), (13)(42)$
  - $(1)(23) : (41)(23), (1)(423), (1)(243)$
- Insgesamt gilt: $s_{4,2} = s_{3,1} + 3 \cdot s_{3,2} = 2 + 3 \cdot 3 = 11$.

Stirling-Dreieck erster Art

Rekursionsformel: $s_{n,k} = s_{n-1,k-1} + (n-1) \cdot s_{n-1,k}$

$n = 0$							1							
$n = 1$						0		1						
$n = 2$					0		1		1					
$n = 3$				0		2		3		1				
$n = 4$			0		6		11		6		1			
$n = 5$	0			24		50		35		10		1		

Berechnung von Teilmengen

Satz Anzahl der Teilmengen

$$2^n = \sum_{k=0}^{n} \binom{n}{k}$$

Beweis

- Korollar aus Binomischem Lehrsatz $(1+1)^n = \sum_{k=0}^{n} \binom{n}{k} 1^k 1^{n-k}$.
- Oder kombinatorisch: Sei M Menge mit $|M| = n$.
- Die Kardinalität der Potenzmengen $\mathcal{P}(M)$ ist $|\mathcal{P}(M)| = 2^n$.
- In $\mathcal{P}(M)$ sind alle k-elementigen Teilmengen von M enthalten.
- Sei S_k die Menge der k-elementigen Teilmengen von M.
- Es gilt $|S_k| = \binom{n}{k}$ für $k = 0, \ldots, n$.
- Ferner ist $\mathcal{P}(M) = \biguplus_{k=0}^{n} S_k$ und damit nach Summenregel

$$2^n = |\mathcal{P}(M)| = \sum_{k=0}^{n} |S_k| = \sum_{k=0}^{n} \binom{n}{k}.$$

Rekursive Berechnung von Binomialkoeffizienten

Satz Rekursion Binomialkoeffizienten

Für alle $n, k \in \mathbb{N}$ mit $n > k$ gilt

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

Beweis

- Wir partitionieren die k-elementigen Teilmengen S_k von $[n]$:

Fall 1: S_k enthält n. Damit enthält S_k noch $k-1$ Elemente aus $[n-1]$. Dafür gibt es $\binom{n-1}{k-1}$ Möglichkeiten.

Fall 2: S_k enthält n nicht. Damit enthält S_k insgesamt k Elemente aus $[n-1]$, wofür es $\binom{n-1}{k}$ Möglichkeiten gibt.

- D.h. die $\binom{n}{k}$ vielen k-elementigen S_k lassen sich in $\binom{n-1}{k-1}$ und $\binom{n-1}{k}$ Teilmengen partitionieren.
- Aus der Summenregel folgt $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

Pascal´sches Dreieck

Rekursionsformel: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

$n = 0$						1						
$n = 1$					1		1					
$n = 2$				1		2		1				
$n = 3$			1		3		3		1			
$n = 4$		1		4		6		4		1		
$n = 5$	1		5		10		10		5		1	

Vandermonde´sche Identität

Satz Vandermonde Identität

Für alle $k, m, n \in \mathbb{N}_0$ gilt:

$$\binom{n+m}{k} = \sum_{t=0}^{k} \binom{n}{t}\binom{m}{k-t}.$$

Beweis

- Sei $M = \{1, 2, \ldots, n+m\}$.
- Die Anzahl der k-elementigen Teilmengen von M ist $\binom{n+m}{k}$.
- Partitionieren M in $M_1 = \{1, \ldots, n\}$ und $M_2 = \{n+1, \ldots, n+m\}$.
- Die k-elementigen Teilmengen von M lassen sich darstellen als Vereinigung von t-elementigen Teilmengen von M_1 und $(k-t)$-elementigen Teilmengen von M_2 für $t = 0, \ldots, k$.
- Anzahl der t-elementigen Teilmengen von M_1: $\binom{n}{t}$.
- Anzahl der $(k-t)$-elementigen Teilmengen von M_2: $\binom{m}{k-t}$.
- Summenregel: Aus $M = M_1 \uplus M_2$ folgt die Vandermonde Identität.

k-Partition, Stirlingzahl zweiter Art

Definition k-Partition, Stirlingzahl zweiter Art

Sei $A = \{a_1, \ldots, a_n\}$. Eine *k-Partition* von A ist eine Zerlegung von A in k paarweise disjunkte $A_1, \ldots, A_k \subseteq A$ mit $A = \bigcup_{i=1}^{k} A_i$.
Wir bezeichnen mit $S_{n,k}$ die Anzahl von k-Partitionen einer n-elementigen Menge. $S_{n,k}$ heisst auch die *Sterlingzahl zweiter Art*.

Bsp: $A = \{1, 2, 3\}$ und $k = 2$

- $\{1\} \cup \{2, 3\}$, $\{1, 2\} \cup \{3\}$, $\{1, 3\} \cup \{2\}$
- D.h. $S_{3,2} = 3$.

Spezialfälle:

- $S_{n,k} = 0$ für $k > n$
- $S_{n,n} = S_{n,1} = 1$, $S_{n,0} = 0$
- $S_{0,0} := 1$

Rekursive Berechnung von $S_{k,n}$

Satz Rekursive Berechnung der Stirlingzahl 2. Art

Für alle $k, n \in \mathbb{N}$ mit $n \geq k$ gilt

$$S_{n,k} = S_{n-1,k-1} + k \cdot S_{n-1,k}.$$

Beweis:

- Sei $A = \{a_1, \ldots, a_n\}$.
- Wir teilen die k-Partitionen $A_1, \ldots, A_k$ in zwei Klassen auf.

Fall 1: $A_i = \{a_n\}$ für ein $i \in [k]$. Dann befinden sich $a_1, \ldots, a_{n-1}$ in einer $(k-1)$-Partition. Dafür gibt es $S_{n-1,k-1}$ Möglichkeiten.

Fall 2: $a_n \in A_i$ und $|A_i| > 1$. Die Mengen $A_1, \ldots, A_{i-1}, A_i \setminus a_n, A_{i+1}, \ldots, A_k$ bilden eine k-Partition für $\{a_1, \ldots, a_{n-1}\}$. Dafür gibt es $S_{n-1,k}$ Möglichkeiten. Zum Einsortieren von a_n in eine der Teilmengen gibt es k Möglichkeiten. Insgesamt also $k \cdot S_{n-1,k}$ Möglichkeiten.

- Die Summenregel liefert $S_{n,k} = S_{n-1,k-1} + k \cdot S_{n-1,k}$.

Beispiel: Rekursives Berechnen von $S_{4,2}$

Bsp:

- Anzahl der 2-Partitionen von $A = [4]$

Fall 1: $\{4\}$ ist eine Teilmenge. Die andere Teilmengen ist $\{1, 2, 3\}$.

Fall 2: $\{4\}$ ist in einer der drei 2-Partitionen von $\{1, 2, 3\}$:

- $\{1\} \cup \{2, 3\} : \{1, 4\} \cup \{2, 3\}, \{1\} \cup \{2, 3, 4\}$
- $\{1, 2\} \cup \{3\} : \{1, 2, 4\} \cup \{3\}, \{1, 2\} \cup \{3, 4\}$
- $\{1, 3\} \cup \{2\} : \{1, 3, 4\} \cup \{2\}, \{1, 3\} \cup \{2, 4\}$

- D.h. $S_{4,2} = S_{3,1} + 2 \cdot S_{3,2} = 1 + 2 \cdot 3 = 7$.

Stirlingdreieck zweiter Art

Rekursionsformel: $S_{n,k} = S_{n-1,k-1} + k \cdot S_{n-1,k}$

$n = 0$					1					
$n = 1$				0		1				
$n = 2$			0		1		1			
$n = 3$		0		1		3		1		
$n = 4$	0		1		7		6		1	
$n = 5$	0	1		15		25		10		1

Bellzahlen

Definition Bellzahlen

Sei A eine Menge mit n Elementen. Mit B_n bezeichnen wir die Anzahl aller Partitionen von A.

Bsp: $A = \{1, 2, 3\}$

- $\{1, 2, 3\}, \{1\} \cup \{2, 3\}, \{1, 2\} \cup \{3\}, \{1, 3\} \cup \{2\}, \{1\} \cup \{2\} \cup \{3\}$.
- D.h. $B_3 = 5$.

Korollar Bellzahlen mittels Stirlingzahlen zweiter Art

$$B_n = \sum_{k=0}^{n} S_{n,k}.$$

- D.h. B_n ist die n-te Zeilensumme im Stirlingdreieck 2. Art.

Geordnete Zahlpartitionen

Definition Geordnete Zahlpartitionen

Sei $n \in \mathbb{N}$. Sei $Z_{n,k}$ die Anzahl der Möglichkeiten, n als Summe k positiver natürlicher Zahlen zu schreiben. Wir nennen $Z_{n,k}$ auch die Anzahl der *geordneten k-Zahlpartitionen von n*.

Bsp: 3-Zahlpartitionen von 5

- $1+1+3, 1+3+1, 3+1+1, 1+2+2, 2+1+2, 2+2+1$
- D.h. $Z_{5,3} = 6$.

Spezialfälle:

- $Z_{n,n} = Z_{n,1} = 1$
- $Z_{n,k} = 0$ für $k > n$

Berechnung von $Z_{n,k}$

Satz Anzahl geordneter k-Zahlpartitionen von n

Seien $n, k \in N$. Dann gilt

$$Z_{n,k} = \binom{n-1}{k-1}.$$

Beweis:

- Schreiben jede Zahl n als Summe von n Einsen, z.B.
$$5 = 1 + 1 + 1 + 1 + 1.$$
- Wählen $k-1$ der $n-1$ Pluszeichen als Trennzeichen aus, z.B.
$$5 = 1 \oplus 1 + 1 \oplus 1 + 1 = 1 + 2 + 2.$$
  - Ziehen ohne Zurücklegen: Kein doppeltes Pluszeichen.
  - ungeordnet: Reihenfolge der Pluszeichen ist ohne Belang.
- D.h. die Anzahl der geordneten k-Partitionen ist $\binom{n-1}{k-1}$.

Beispiel: Zählen von Lösungen

Bsp: Sei $X = \{(x_1, \ldots, x_k) \in \mathbb{N}_0^k \mid x_1 + \ldots + x_k = n\}$. Bestimme $|X|$.

- Problem: Summanden können Null sein.
- Wir addieren zu jedem der k Summanden eine Eins

$$x_1' + \ldots x_k' = n + k \text{ mit } x_i' \geq 1 \text{ für } i = 1, \ldots, k.$$

- Jede Summe der x_i, die sich zu n aufaddieren entspricht eineindeutig einer geordneten k-Zahlpartition von $n + k$. (Isomorphismus)
- Mit Gleichheitsregel ergibt sich $|X| = \binom{n+k-1}{k-1}$.

Ungeordnete Zahlpartitionen

Definition Ungeordnete Zahlpartitionen

Sei $n \in \mathbb{N}$. Sei $P_{n,k}$ die Anzahl der Möglichkeiten n als Summe k positiver Zahlen zu schreiben, wobei die Reihenfolge der Summanden keine Rolle spielt. Wir nennen $P_{n,k}$ die Anzahl *ungeordneter k-Zahlpartitionen von n*.

Bsp: $P_{7,3}$

- $1+1+5$, $1+2+4$, $1+3+3$, $2+2+3$
- D.h. $P_{7,3} = 4$.

Spezialfälle:

- $P_{n,n} = P_{n,1} = 1$
- $P_{n,k} = 0$ für $k > n$
- $P_{0,0} := 1$

Rekursive Berechnung ungeordneter Zahlpartitionen

Satz Anzahl ungeordneter Zahlpartitionen

Für alle $k, n \in \mathbb{N}$ mit $k < n$ gilt $P_{n+k,k} = \sum_{j=1}^{k} P_{n,j}$.

Beweis:

- Wir zerlegen $n + k$ in i Einsen-Summanden und $k - i$ Summanden größer als 1, d.h.

  $$n + k = 1 + \ldots + 1 + n_{i+1} + \ldots + n_k \text{ mit } n_j \geq 2 \text{ für } j = 1, \ldots, k.$$

- Wir subtrahieren Eins von jedem der Summanden

  $$n = n'_{i+1} + \ldots n'_k \text{ mit } n_j \geq 1 \text{ für } j = i + 1, \ldots, k.$$

- D.h. die n'_j bilden eine ungeordnete $(k - i)$-Zahlpartition von n.
- Andererseits liefert jede $(k - i)$-Zahlpartition von n eineindeutig eine k-Zahlpartition von n mit genau i Einsen (Isomorphismus).
- Mit Gleichheitsregel: $P_{n+k,k}$ mit genau i Einsen ist $P_{n,k-i}$.
- Mit Summenregel: $P_{n+k,k} = \sum_{i=0}^{k-1} P_{n,k-i} = \sum_{j=1}^{k} P_{n,j}$.

Verteilen von Bällen auf Urnen

Szenario:
Wir verteilen n Bälle auf m Urnen, d.h.

$$f : B \to U \text{ mit } B = \{b_1, \ldots, b_n\} \text{ und } U = \{u_1, \ldots, u_m\}.$$

Dabei unterscheiden wir alle Kombinationen der folgenden Fälle

1. Die Bälle sind unterscheidbar oder nicht unterscheidbar.
2. Die Urnen sind unterscheidbar oder nicht unterscheidbar.

Für alle vier Kombinationen untersuchen wir die Fälle

1. f beliebig, d.h. wir verteilen die Bälle beliebig.
2. f injektiv, d.h. jede Urne enthält höchstens einen Ball.
3. f surjektiv, d.h. jede Urne enthält mindestens einen Ball.
4. f bijektiv, d.h. jede Urne enthält genau einen Ball.

Bälle und Urnen sind unterscheidbar

***f* beliebig:**

- m Möglichkeiten für jeden der n Bälle, d.h. insgesamt m^n.

***f* injektiv:** Für $|B| = n \leq m = |U|$ gilt:

- m Möglichkeiten für den ersten Ball, $m - 1$ für den zweiten, usw.
- Insgesamt $m^{\underline{n}}$ Möglichkeiten.

***f* bijektiv:** Für $|B| = n = m = |U|$ gilt:

- m Möglichkeiten für den ersten Ball, $m - 1$ für den zweiten, usw.
- Insgesamt $m!$ Möglichkeiten.

Bälle und Urnen sind unterscheidbar

***f* surjektiv:** Für $|B| = n \geq m = |U|$ gilt:

- Definieren die Urbildmengen $T_u := \{f^{-1}(u) \mid u \in U\}$.
- Die T_u bilden eine m-Partition der Menge B.
- Damit gibt es $S_{n,m}$ Möglichkeiten für die Urbildmenge B.
- Für jede m-Partition $B_1, \ldots, B_m$ von B landen jeweils die Bälle aus einer Menge B_i gemeinsam in einer Urne.
- $m!$ Möglichkeiten, eine m-Partition auf m Urnen zu verteilen.
- D.h. wir erhalten insgesamt $S_{n,m} \cdot m!$ Möglichkeiten.

Urnen unterscheidbar, Bälle nicht

Idee: Wir zählen nur die Anzahl der Bälle pro Urne.

***f* beliebig:**

- Kodieren die Anzahl der Bälle mit insgesamt n Sternen.
- Unterscheiden die Urnen mit $m-1$ Trennstrichen.
- z.B. **|***| bedeutet 2 Bälle in u_1, 3 in u_2 und 0 in u_3.
- Anzahl Kodierungen mit n Sternen und $m-1$ Trennstrichen

$$\binom{n+m-1}{n}.$$

Urnen unterscheidbar, Bälle nicht

***f* injektiv:** Für $|B| = n \leq m = |U|$ gilt:

- Wählen n aus den m Urnen aus, die genau einen Ball enthalten.
- Insgesamt $\binom{m}{n}$ Möglichkeiten.

***f* surjektiv:** Für $|B| = n \geq m = |U|$ gilt:

- Die Urne u_i enthalte x_i Bälle für $i = 1, \ldots, m$.
- Die x_i bilden eine geordnete m-Zahlpartition von n, da
$$x_1 + x_2 + \ldots + x_m = n.$$
- D.h. wir erhalten $\binom{n-1}{m-1}$ Möglichkeiten.

***f* bijektiv:** Für $|B| = n = m = |U|$ gilt:

- Jede Urne enthält einen Ball: Genau eine Möglichkeit.

Bälle und Urnen nicht unterscheidbar

Idee: Anzahl Bälle in Urnen entscheidend, Reihenfolge belanglos.

***f* beliebig:**

- Angenommen, es werden genau k der n Urnen belegt.
- Sei $B_1, \ldots, B_k$ eine k-Partition von Bällen, so dass jeweils alle Bälle aus B_i gemeinsam in einer Urne landen.
- $|B_1| + \ldots + |B_k| = n$, d.h. die $|B_i|$ bilden eine Zahlpartition von n.
  - Dies ist eine ungeordnete Zahlpartition (Urnen ununterscheidbar).
- D.h. wir erhalten für ein festes k genau $P_{n,k}$ Möglichkeiten.
- Insgesamt erhalten wir damit $\sum_{k=1}^{m} P_{n,k}$ Möglichkeiten.

***f* surjektiv:** Für $|B| = n \geq m = |U|$ gilt:

- Da m Urnen belegt sind: $P_{n,m}$ Möglichkeiten.

Bälle und Urnen nicht unterscheidbar

***f* injektiv:** Für $|B| = n \leq m = |U|$ gilt:

- Jede der belegten Urnen enthält einen Ball: 1 Möglichkeit.

***f* bijektiv:** Für $|B| = n = m = |U|$ gilt:

- Alle Urnen enthalten genau einen Ball: 1 Möglichkeit.

Bälle unterscheidbar, Urnen nicht

Idee: Entspricht Partitionierung der Bälle.

***f* beliebig:**

- Seien genau k Urnen belegt.
- Sei $B_1, \ldots, B_k$ eine k-Partition von B, so dass alle Bälle in B_i gemeinsam in einer Urne landen.
- Die Anzahl der k-Partitionen von B beträgt $S_{n,k}$ für festes k.
- Insgesamt $\sum_{k=1}^{m} S_{n,k}$ Möglichkeiten.

***f* surjektiv:** Für $|B| = n \geq m = |U|$ gilt:

- Entspricht einer m-Partition von B, d.h $S_{n,m}$ Möglichkeiten.

Bälle unterscheidbar, Urnen nicht

***f* injektiv:** Für $|B| = n \leq m = |U|$ gilt:

- In jeder Urne höchstens ein Ball: 1 Möglichkeit.

***f* bijektiv:** Für $|B| = n = m = |U|$ gilt:

- In jeder Urne genau ein Ball: 1 Möglichkeit.

Zusammenfassung

$\lvert B\rvert = n$, $\lvert U\rvert = m$	beliebig	injektiv $n \leq m$	surjektiv $n \geq m$	bijektiv $n = m$
Bälle und Urnen unterscheidbar	m^n	$m^{\underline{n}}$	$S_{n,m} \cdot m!$	$m!$
Bälle gleich, Urnen unterscheidbar	$\binom{n+m-1}{n}$	$\binom{m}{n}$	$\binom{n-1}{m-1}$	1
Bälle unterscheidbar, Urnen nicht	$\sum_{k=1}^{m} S_{n,k}$	1	$S_{n,m}$	1
Bälle und Urnen ununterscheidbar	$\sum_{k=1}^{m} P_{n,k}$	1	$P_{n,m}$	1

Überblick über Graphentheorie

Ungerichtete Graphen:

- Baum, Spannbaum, Prüfercode
- Traversierung von Graphen: Breiten- und Tiefensuche
- Besuchen aller Kanten, Besuchen aller Knoten
- Färben von Knoten und Kanten, Matching

Gerichtete Graphen:

- Kreisfreiheit, topologische Sortierung
- Wurzelbaum

Ungerichtete Graphen

Definition Ungerichteter Graph

Ein *ungerichteter Graph* ist ein Tupel $G = (V, E)$ mit

- der Knotenmenge $V = \{v_1, \ldots, v_n\}$ und
- der Kantenmenge $E = \{e_1, \ldots, e_m\} \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$.

D.h. insbesondere sind **nicht** erlaubt:

- Schlingen bzw. Selbstkanten $\{v, v\} \in E$
- Mehrere Kanten zwischen denselben Knoten, d.h. E ist keine Multimenge.

Motivation für Graphprobleme

Szenario:

- Wir veranstalten ein Turnier mit fünf Teilnehmern a, b, c, d, e.
- Jeder soll gegen jeden Spielen, d.h. $\binom{5}{2} = 10$ Partien.
- Keiner soll in aufeinanderfolgenden Spielen antreten.

Modellierung als Graph:

- Wir stellen alle Paarungen als Knoten in einem Graph dar.
- Wir verbinden zwei Knoten, falls sie disjunkte Mannschaften enthalten.

Graphproblem:

- Bestimme einen Pfad, der alle Knoten genau einmal besucht.
- Der Pfad bestimmt die Reihenfolge der Partien.

Isomorphie von Graphen

Definition Isomorphe Graphen

Seien $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ ungerichtete Graphen. G_1 ist *isomorph* zu G_2, falls es eine Bijektion $f : V_1 \to V_2$ gibt mit

$$\{u, v\} \in E_1 \Leftrightarrow \{f(u), f(v)\} \in E_2.$$

Bsp: Anzahl der isomorphen Graphen mit n Knoten

- $n = 1$: Nur ein Graphen möglich.
- $n = 2$: Zwei Graphen, mit einer und keiner Kante.
- $n = 3$: Vier Graphen, mit Kantenzahl $0, 1, 2, 3$.

Anzahl der Isomorphieklassen für Graphen bis $n = 9$

n	1	2	3	4	5	6	7	8	9
Anzahl Graphen	1	2	4	11	34	156	1044	12344	308168

Spezielle Graphen

vollständiger Graph K_n:

- K_n enthält n vollständig miteinander verbundene Knoten.

Pfad P_n:

- P_n enthält n Knoten $v_1, \ldots, v_n$.
- $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$

Kreisgraph C_n:

- C_n enthält n Knoten $v_1, \ldots, v_n$.
- $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$

Spezielle Graphen

Gittergraph $M_{n,m}$:

- $M_{n,m}$ enthält n Zeilen mit jeweils m Knoten.
- Die Knoten jeder Zeile sind als Pfad verbunden.
- Die Knoten jeder Spalte sind ebenfalls als Pfad verbunden.

d-dimensionaler Hyperwürfel Q_n:

- $V = \{0,1\}^d$
- $E = \{\{u,v\} \in V^2 \mid u, v \text{ unterscheiden sich in einer Stelle.}\}$

Nachbarschaft, Grad, regulär, Inzidenz

Definition Eigenschaften von Graphen

Sei $G = (V, E)$ ein ungerichteter Graph.

1. Die *Nachbarschaftschaft* $\Gamma(u)$ eines Knoten $u \in V$ ist $\Gamma(u) := \{v \in V \mid \{u, v\} \in E\}$.
2. Der *Grad* $\deg(u)$ eines Knotens u ist $\deg(u) := |\Gamma(u)|$.
3. G heißt *k-regulär* gdw jeder Knoten in G Grad k besitzt.
4. Falls $e = \{u, v\} \in E$, dann heißen u and v *adjazent*. Man nennt die Endpunkte u und v von e die zu e *inzidenten* Knoten.

Handschlaglemma

Satz Handschlaglemma

Für jeden Graphen $G = (V, E)$ gilt $\sum_{v \in V} \deg(v) = 2 \cdot |E|$.

Beweis:

- Zählen für jeden Knoten die Anzahl seiner Nachbarn

  $$\sum_{v \in V} \deg(v) = \sum_{v \in V} |\Gamma(v)| = \sum_{u \in V} |\{v \in V \mid \{u, v\} \in E\}|.$$

- Zählen dabei jede Kante $e = \{u, v\}$ genau zweimal: Einmal bei den Nachbarn von v und einmal bei den Nachbarn von u.

Handschlaglemma Teil 2

Lemma Knoten mit geradem Grad

Sei $G = (V, E)$ ein ungerichteter Graph. Die Anzahl der Knoten in G mit ungeradem Grad ist gerade.

Beweis:

- Wir partitionieren V in Knoten mit geradem/ungeradem Grad: $V_g := \{v \in V \mid \deg(v) \text{ gerade }\}$, $V_u := \{v \in V \mid \deg(v) \text{ ungerade}\}$.
- Es gilt $2|E| = \sum_{v \in V} \deg(v) = \sum_{v \in V_g} \deg(v) + \sum_{v \in V_u} \deg(v)$.
- $\sum_{v \in V_g} \deg(v)$ ist gerade, da jeder Summand gerade ist.
- Daher ist $\sum_{v \in V_u} \deg(v)$ ebenfalls gerade, sonst könnte die Summe nicht gerade sein.
- $\sum_{v \in V_u} \deg(v)$ enthält nur ungerade Summanden.
- D.h. $|V_u|$ muss gerade sein.

Weg, Pfad, Kreis

Definition Weg, Pfad, Kreis

Sei G ein ungerichteter Graph.

1. Eine Knotenfolge $(v_0, \ldots, v_k) \in V^{k+1}$ mit $\{v_i, v_{i+1}\} \in E$ für $0 \leq i < k$ heißt *Weg der Länge k* mit Anfangsknoten v_0 und Endknoten v_k.
2. Ein knotendisjunkter Weg heißt *Pfad*.
3. Ein *u-v Weg/Pfad* ist ein Weg/Pfad mit Anfangsknoten u und Endknoten v.
4. Ein Kreis der Länge k, $k > 2$ ist eine knotendisjunkte Folge $(v_0, \ldots, v_{k-1})$ mit $\{v_i, v_{i+1 \bmod k}\} \in E$ für $0 \leq i < k$.

Schwacher/induzierter Teilgraph

Definition Schwacher und induzierter Teilgraph

Sei $G = (V, E)$ ein ungerichteter Graph und $V' \subseteq V$.

1. Sei $E_1 \subseteq \{\{u, v\} \in E \mid u, v \in V'\}$. Dann heißt $G_1 = (V', E_1)$ *schwacher Teilgraph* von G.
2. Sei $E_2 = \{\{u, v\} \in E \mid u, v \in V'\}$. Dann heißt $G_2 = (V', E_2)$ der von V' *induzierte Teilgraph*. Wir schreiben auch $G_2 = G[V']$.

Zusammenhang

Definition Zusammenhängend, Zusammenhangskomponente

Sei $G = (V, E)$ ein ungerichteter Graph.

1. G ist zusammenhängend gdw für alle $u, v \in V$ ein u-v Pfad in G existiert.
2. Sei G nicht zusammenhängend und $V' \subseteq V$ maximal mit der Eigenschaft, dass für alle $u, v \in V'$ ein u-v Pfad in $G[V']$ existiert. Dann heißt $G[V']$ *Zusammenhangskomponente* (ZHK).

Beispiel:

- Wir betrachten die Relation $R = \{(u, v) \in V^2 \mid \{u, v\} \in E\}$.
- Sei $R^* = \{(u, v) \in V^2 \mid \text{Es existiert ein } u\text{-}v \text{ Pfad in G}\}$.
- R^* heißt *reflexive, transitive Hülle* von R.
- R^* ist Äquivalenzrelation, d.h. reflexiv, symmetrisch und transitiv.
- Die Äquivalenzklassen definieren die ZHKs von G.

Anzahl Zusammenhangskomponenten

Satz Anzahl ZHKs

Sei $G = (V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $E = \{e_1, \ldots, e_m\}$. Dann besitzt G mindestens $n - m$ Zusammenhangskomponenten.

Beweis:

- Induktion über die Kantenanzahl m.
- **IV für $m = 0$:** Dann besitzt G die ZHKs $G[v_1], \ldots, G[v_n]$.
- **IS ($m - 1 \rightarrow m$):** Sei $e_m = \{u, v\}$ und $E' = E \setminus \{e_m\}$.
- Nach **IA** besitzt $G' = (V, E')$ mindestens $n - (m - 1)$ ZHKs.

Fall 1: u, v liegen in derselben ZHK von G'.
- Dann besitzt G mindestens $n - m + 1$ ZHKs.

Fall 2: u, v liegen in verschiedenen ZHKs von G'.
- Dann besitzt G mindestens $n - m$ Zusammenhangskomponenten.

Für zusammenhängende Graphen

Korollar Kanten in zusammenhängendem Graph

Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph. Dann gilt $|E| \geq |V| - 1$.

Beweis:

- In jedem Graph ist die Anzahl der ZHKs mindestens $|V| - |E|$.
- Da G zusammenhängend ist, besitzt G genau eine ZHK. Damit
$$\#(\text{ZHK}(G)) = 1 \geq |V| - |E|.$$
- Es folgt $|E| \geq |V| - 1$.

Bäume und Wälder

Definition Baum und Wald

Sei G ein ungerichteter Graph.

1. G heißt *kreisfrei* gdw kein schwacher Teilgraph von G ein Kreis ist.
2. G heißt *Baum* gdw G kreisfrei und zusammenhängend ist.
3. G heißt *Wald* gdw die ZHKs von G Bäume sind.
4. Jeder Knoten v eines Baums mit $\deg(v) = 1$ heißt *Blatt*.

Eigenschaften von Bäumen

Satz Eigenschaft von Bäumen

Sei $G = (V, E)$ ein ungerichteter Graph. Folgende Aussagen sind äquivalent:

1. G ist ein Baum.
2. Für alle $u, v \in V$ existiert genau ein u-v Pfad in G.
3. G ist zusammenhängend, aber $G' = (V, E \setminus \{e\})$ ist für alle $e \in E$ nicht zusammenhängend.
4. G ist zusammenhängend und besitzt genau $n - 1$ Kanten.
5. G ist kreisfrei und besitzt genau $n - 1$ Kanten.
6. G ist kreisfrei und für alle nicht adjazenten $u, v \in V$ gilt
$$G' = (V, E \cup \{u, v\}) \text{ enthält einen Kreis.}$$

Beweis: Durch Ringschluss $1 \Rightarrow 2 \Rightarrow \ldots \Rightarrow 6 \Rightarrow 1$.

Ringschluss 1 $\Rightarrow$ 2

Lemma Ringschluss 1 $\Rightarrow$ 2

Sei G ein Baum. Dann existiert für alle $u, v \in V$ genau ein u-v Pfad.

Beweis:

- Da G zusammenhängend ist, existiert ein u-v Pfad
$$p = (v_0 = u, \ldots, v_k = v) \text{ in } G.$$
- **Annahme:** Es existiert ein zweiter u-v Pfad
$$p' = (v'_0 = u, \ldots, v'_t = v) \text{ in } G.$$
- Sei r minimal mit $v_{r+1} \neq v'_{r+1}$, d.h. die Pfade gabeln sich bei v_r.
- Sei v_ℓ, $\ell > r$ der erste Knoten im Pfad $(v_{r+1}, \ldots, v_k)$, der auch im Pfad $(v'_{r+1}, \ldots, v'_t)$ auftritt.
- D.h. beide Pfade werden bei v_ℓ wieder zusammengeführt.
- Man beachte, dass ein solcher Knoten v_ℓ existieren muss, da p und p' denselben Endknoten besitzen.
- Damit ist $(v_r, v_{r+1}, \ldots, v_\ell, \ldots, v'_{r+2}, v'_{r+1})$ ein Kreis in G. (Widerspruch zur Kreisfreiheit eines Baums G).

Ringschluss 2 $\Rightarrow$ 3

Lemma Ringschluss 2 $\Rightarrow$ 3

Falls für alle $u, v \in V$ genau ein u-v Pfad existiert, dann ist G zusammenhängend, aber $G' = (V, E \setminus \{e\})$ ist für alle $e \in E$ nicht zusammenhängend.

Beweis:

- G ist zusammenhängend nach Definition.
- Sei $e = \{u, v\}$ beliebig.
- Nach Voraussetzung ist (u, v) der einzige u-v Pfad.
- D.h. $G' = (V, E \setminus \{e\})$ besitzt keinen u-v Pfad.
- Damit ist G' nicht zusammenhängend.

Ringschluss 3 $\Rightarrow$ 4

Lemma Ringschluss 3 $\Rightarrow$ 4

Sei G zusammenhängend, aber $G' = (V, E \setminus \{e\})$ für alle $e \in E$ nicht zusammenhängend. Dann ist G zusammenhängend und besitzt genau $n - 1$ Kanten.

Beweis:

- Durch Entfernen jeder beliebigen Kante aus E erhöht sich die Anzahl der ZHKs um Eins.
- G ist zusammenhängend, besitzt also eine ZHK.
- $G_1 = (V, E \setminus \{e_1\})$ besitzt 2 ZHKs.
- $G_2 = (V, E \setminus \{e_1, e_2\})$ besitzt 3 ZHKs, usw.
- Damit besitzt $G_{n-1} = (V, \emptyset)$ genau n ZHKs.
- G_{n-1} entsteht durch Entfernen aller $m = n - 1$ Kanten.

Ringschluss 4 $\Rightarrow$ 5

Lemma Ringschluss 4 $\Rightarrow$ 5

G ist zusammenhängend mit genau $n-1$ Kanten gdw G kreisfrei mit genau $n-1$ Kanten ist.

Beweis: Richtung 4 $\Rightarrow$ 5

- **Annahme:** G besitzt einen Kreis $K = (v_0, \ldots, v_{k-1})$.
- Da G zusammenhängend ist, müssen die $n-k$ Knoten außerhalb des Kreises von den k Kreisknoten aus erreichbar sein.
- Einbinden der Nicht-Kreisknoten erfordert eine Kante pro Knoten.
- Damit besitzt G mindestens $k + (n-k)$ Kanten.
  (Widerspruch: G besitzt genau $n-1$ Kanten.)

Rückrichtung 5 $\Rightarrow$ 4

Beweis: Richtung 5 $\Rightarrow$ 4

- Fügen $n-1$ Kanten aus E sukzessive zu $G_0 = (V, \emptyset)$ hinzu.
- G_0 besitzt n Zusammenhangskomponenten.
- Müssen zeigen, dass jede Kante $e = \{u, v\}$ zwei verschiedene ZHKs verbindet.
- **Annahme:** u, v sind bereits durch einen u-v Pfad p verbunden.
- Dann schließt p zusammen mit e einen Kreis in G. (Widerspruch zur Kreisfreiheit von G)
- Damit verringert jede Kante die Anzahl der ZHKs um Eins.
- Da wir genau $n-1$ Kanten einfügen, besitzt G genau eine ZHK.
- D.h. G ist zusammenhängend.

Ringschluss 5 $\Rightarrow$ 6

Lemma Ringschluss 5 $\Rightarrow$ 6

Sei G kreisfrei mit genau $n-1$ Kanten. Dann ist G kreisfrei und für alle nicht adjazenten $u, v \in V$ enthält $G' = (V, E \cup \{u, v\})$ einen Kreis.

Beweis:

- Wegen 5 $\Rightarrow$ 4 folgt, dass G zusammenhängend ist.
- D.h. für alle nicht adjazenten $u, v \in V$ existiert ein u-v Pfad p in G.
- Damit bildet p zusammen mit der Kante $\{u, v\}$ einen Kreis.

Ringschluss 6 $\Rightarrow$ 1

Lemma Ringschluss 6 $\Rightarrow$ 1

Sei G kreisfrei und für alle nicht adjazenten $u, v \in V$ enthalte $G' = (V, E \cup \{u, v\})$ einen Kreis. Dann ist G ein Baum.

Beweis:

- Fallunterscheidung für alle disjunkten $u, v \in V$:

  Fall 1: $\{u, v\} \in E$. Damit existiert ein u-v Pfad in G.

  Fall 2: $\{u, v\} \notin E$. Damit ist $\{u, v\}$ in einem Kreis K von G' enthalten. Entfernt man $\{u, v\}$ aus G', so erhält man G. Es existiert noch immer ein u-v Pfad in G, nämlich der Rest des Kreises K.

- Damit existiert für alle u, v ein u-v Pfad in G.
- D.h. G ist zusammenhängend und nach Voraussetzung kreisfrei, d.h. G ist ein Baum.

Anzahl von Kanten in Wäldern

Satz Anzahl Kanten im Wald

Sei $G = (V, E)$ ein Wald mit k Bäumen. Dann besitzt G genau $|V| - k$ Kanten.

Beweis:

- Seien $G[V_1], \ldots, G[V_k]$ die Bäume in G.
- Jeder Baum $G[V_i]$, $i \in [k]$ besitzt $|V_i| - 1$ Kanten, d.h.

$$m = \sum_{i=1}^{k} (|V_i| - 1) = \left(\sum_{i=1}^{k} |V_i|\right) - k = |V| - k.$$

Spannbäume

Definition Spannbaum

Sei $G = (V, E)$ ein zusammenhängender, ungerichteter Graph. Ein schwacher Teilgraph $T = (V, E_T)$ von G heißt *Spannbaum* gdw T ein Baum ist.

- Jeder zusammenhängende Graph G besitzt einen Spannbaum.
- Spannbäume von G sind im Allgemeinen nicht eindeutig.
- Konstruktion eines Spannbaumes durch Entfernen von Kreisen.

Berechnung von Spannbäumen

Algorithmus SPANNBAUM

EINGABE: $G = (V, E)$ zusammenhängend

1. Solange $G = (V, E)$ einen Kreis K enthält.
   1. Wähle eine beliebige Kreiskante e aus K. Setze $E \leftarrow E \setminus \{e\}$.

AUSGABE: Spannbaum $T = (V, E)$ von G

Satz Korrektheit von SPANNBAUM

Sei $G = (V, E)$ zusammenhängend. Dann berechnet SPANNBAUM einen Spannbaum T von G in $|E| - |V| + 1$ Schleifendurchläufen.

Korrektheit und Laufzeit von SPANNBAUM

Beweis:

- **Korrektheit:** Wir zeigen, dass G stets zusammenhängend bleibt.
- Sei $K = (v_0, \ldots, v_{k-1})$ ein Kreis in G.
- Wir entfernen eine Kreiskante $e = \{v_i, v_{i+1 \bmod k}\}$, $i \in \mathbb{Z}_k$.
- Seien u, v Knoten mit einem u-v Pfad, der e verwendet.
- Ersetze e durch den Restkreis $(v_i, v_{i-1}, \ldots, v_0, \ldots, v_{i+1})$.
- Bei Terminierung ist G zusammenhängend und kreisfrei.
- SPANNBAUM muss terminieren, da die Kantenzahl sukzessive verringert wird.

- **Laufzeit:** T ist ein Spannbaum und besitzt daher $|V| - 1$ Kanten.
- Jeder Schleifendurchlauf verringert die Kantenanzahl um Eins.
- D.h. die Anzahl Schleifendurchläufe ist $|E| - (|V| - 1)$.

Markierte und isomorphe Spannbäume

Bsp:

- Der vollständige Graph K_2 ist selbst ein Spannbaum.
- Der K_3 besteht aus einem Kreis der Länge 3.
- Durch Entfernen einer Kanten entsteht ein Spannbaum.
- Alle 3 Spannbäume sind isomorph, sie unterscheiden sich lediglich durch die Knotenmarkierungen.
- Der K_4 besitzt 2 nicht-isomorphe Spannbäume, den C_4 und einen Sterngraphen mit einem Knoten vom Grad 3.

# Bäume \ $\lvert V\rvert$	2	3	4	5	6	7	8
markiert	1	3	16	125	1296	16807	262144
nicht-isomorph	1	1	2	3	6	11	23

Satz von Cayley

Satz von Cayley

Für $n \geq 2$ gibt es genau n^{n-2} markierte Bäume.

Beweisidee:

- Sei $V = [n]$.
- Sei T_n die Menge der markierten Bäume mit n Knoten.
- Wir definieren eine bijektive Kodierung $f\colon T_n \to [n]^{n-2}$.
- Diese Kodierung bezeichnen wir als den *Prüfercode* eines Baums.
- Anwenden von f entspricht dem Kodieren im Prüfercode.
- Anwenden von f^{-1} entspricht dem Dekodieren im Prüfercode.
- Da f bijektiv ist, folgt $|T_n| = n^{n-2}$.

Berechnung von f

Algorithmus KODIERUNG

EINGABE: Baum $T = ([n], E) \in T_n$

1. Setze $i \leftarrow 1$
2. Solange $|V| > 2$
   1. Sei $v \in V$ das Blatt mit kleinster Markierung.
   2. $t_i \leftarrow$ Nachbar von v in T.
   3. $V \leftarrow V \setminus \{v\}$; $E \leftarrow E \setminus \{v, t_i\}$; $i \leftarrow i + 1$.

AUSGABE: Prüfercode $(t_1, \ldots, t_{n-2})$

Laufzeit:

- In jedem Schleifendurchlauf wird ein Knoten entfernt.
- D.h. KODIERUNG terminiert nach $|V| - 2$ Durchläufen.

Korrektheit von KODIERUNG

Korrektheit:

- Zeigen: In jedem Baum T mit $n > 2$ Knoten existiert ein Blatt.
- T besitzt genau $|E| = n - 1$ Kanten.
- Annahme: Alle Knoten besitzen Grad mindestens 2.
- Mit Handschlaglemma gilt

$$2 \cdot |E| = 2(n-1) = \sum_{v \in V} \deg(v) \geq 2n.$$

- Widerspruch, d.h. es muss einen Knoten mit Grad 1 geben.

- Zeigen nun: Durch Entfernen eines Blattes w bleibt T ein Baum.
- Seien $u, v \neq w$ Knoten mit u-v Pfad $p = (v_0 = u, \ldots, v_k = v)$.
- w kann nicht in p enthalten sein, da alle inneren Knoten $v_1, \ldots, v_{k-1}$ des Pfades mindestens Grad 2 besitzen.

Prüfercode und Grad der Knoten

Lemma Knoten im Prüfercode

Sei $T = (V, E)$ ein Baum. Jedes $v \in V$ kommt im Prüfercode von T genau $(\deg(v) - 1)$-mal vor.

Beweis:

- Blätter von T kommen im Prüfercode nicht vor.
- Sei v ein innerer Knoten von T, d.h. $\deg(v) \geq 2$.

Fall 1: v ist bei Terminierung von KODIERUNG entfernt.

- Dann war v im Laufe des Algorithmus ein Blatt.
- Dazu wurden zunächst $\deg(v) - 1$ Nachbarn von v entfernt.
- Pro Entfernen eines Nachbarn taucht v einmal im Prüferkode auf.

Fall 2: v ist bei Terminierung nicht entfernt.

- Analog zu Fall 1: $\deg(v) - 1$ Nachbarn wurden bereits entfernt.

Dekodieren des Prüfercodes

Algorithmus DEKODIERUNG

EINGABE: $t_1, \ldots, t_{n-2} \in [n]$

1. $V \leftarrow [n]$
2. for $i \leftarrow 1$ to n
   1. $\deg(i) \leftarrow 1$
   2. for $j \leftarrow 1$ to $n-2$
      1. if $(i = t_j)$ then $\deg(i) \leftarrow \deg(i) + 1$
3. for $i \leftarrow 1$ to $n-2$
   1. Sei v kleinster Knoten mit $\deg(v) = 1$.
   2. $E \leftarrow E \cup \{v, t_i\}$
   3. $\deg(v) \leftarrow \deg(v) - 1$; $\deg(t_i) \leftarrow \deg(t_i) - 1$
4. Seien u, v die verbliebenen Knoten mit Grad 1. $E \leftarrow E \cup \{u, v\}$.

AUSGABE: Baum $T = (V, E)$

Korrektheit der DEKODIERUNG

Laufzeit:

- Terminierung nach $\mathcal{O}(n^2)$ Schleifendurchläufen.

Korrektheit:

- Schritt 1: Rekonstruktion der Knotenanzahl $|V| = n$.
- Schritt 2: Rekonstruktion aller Knotengrade aus T. Korrektheit folgt aus zuvor gezeigtem Lemma.
- Schritt 3: Erfolgt analog zur Kodierung:
  - Bestimmung des Blattes v mit kleinster Markierung.
  - Hinzufügen anstatt Entfernen der Kante $\{v, t_i\}$.
  - Anpassen der verbleibenden Nachbarzahl von v, t_i.
- D.h. DEKODIERUNG fügt in der i-ten Iteration von Schritt 3 diejenige Kante $\{v, t_i\}$ in T ein, die im i-ten Schritt von KODIERUNG entfernt wurde.

Speicherung von Graphen

Definition Adjazenzmatrix und Adjazenzliste

Sei $G = ([n], E)$ ein ungerichteter Graph.

1. Die *Adjazenzmatrix* $A = (a_{u,v}) \in \{0,1\}^{n \times n}$ von G ist definiert als
$$a_{u,v} = \begin{cases} 1 & \text{falls } \{u,v\} \in E \\ 0 & \text{sonst} \end{cases}.$$
2. Die *Adjazenzliste* $A[1 \ldots n]$ von G ist ein Array von n verketteten Listen. Die i-te verkettete Liste $A[i]$ beinhaltet alle Nachbarn von i in aufsteigend sortierter Reihenfolge.

Vergleich der Darstellungen: Sei $\min_{u,v} := \min\{\deg(u), \deg(v)\}$.

	Speicherbedarf	$\{u,v\} \in E$?	Bestimme $\Gamma(v)$
Adjazenzmatrix	$\Theta(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$
Adjazenzliste	$\Theta(n+m)$	$\mathcal{O}(\min_{u,v})$	$\Theta(\deg(v))$

Datenstruktur Warteschlange

Datenstruktur Warteschlange

Sei U eine Menge. Eine *Warteschlange bzw. Queue* Q ist eine Datenstruktur auf U mit den folgenden Operationen:

1. Generieren einer Queue: $Q \leftarrow$ new Queue.
2. Einfügen von $u \in U$ in Q: Q.Enqueue(u).
3. Prüfung auf Leerheit von Q: Q.Isempty().
4. Entfernen des *zuerst* eingefügten Elements in Q: Q.Dequeue().

Bemerkungen:

- Queue ist eine FIFO-Datenstruktur, d.h. first in, first out.
- Wir nehmen an, dass alle Operationen Laufzeit $\mathcal{O}(1)$ benötigen.

Datenstruktur Stack

Datenstruktur Stack

Sei U eine Menge. Ein *Stapel bzw. Stack* S ist eine Datenstruktur auf U mit den folgenden Operationen:

1. Generieren eines Stacks S: $S \leftarrow$ new Stack.
2. Einfügen von u in S: S.Push(u).
3. Test auf Leerheit von S: S.Isempty()
4. Entfernen des *zuletzt* eingefügten Elements in S: S.Pop()

Bemerkungen:

- Stack ist eine LIFO-Datenstruktur, d.h. last in, first out.
- Wir nehmen an, dass alle Operationen Laufzeit $\mathcal{O}(1)$ benötigen.

Breitensuche BFS (Breadth First Search)

Algorithmus BREITENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. If $(v = s)$ then $d[v] \leftarrow 0$ else $d[v] \leftarrow \infty$;
   2. pred$[v] \leftarrow$ nil;
2. $Q \leftarrow$ new Queue; Q.Enqueue(s);
3. While (Q.Isempty() $\neq$ TRUE)
   1. $v \leftarrow Q$.Dequeue(Q);
   2. Für alle $u \in \Gamma(v)$
      1. If $d[u] = \infty$ then $d[u] \leftarrow d[v] + 1$; pred$[u] \leftarrow$ v; Q.Enqueue(u);

AUSGABE: Arrays $d[v]$, pred$[v]$ für alle $v \in V$

Kürzeste u-v Pfade

Satz Kürzeste u-v Pfade

BREITENSUCHE berechnet bei Eingabe $G = (V, E)$, v für jeden Knoten $u \in V$ einen kürzesten u-v Pfad $p = (u, \text{pred}[u], \text{pred}[\text{pred}[u]], \ldots, v)$ in Zeit $\mathcal{O}(|V| + |E|)$.

Beweis: Korrektheit

- p ist ein Pfad, denn kein Knoten wird zweimal in Q eingefügt.
- Wegen $d[\text{pred}[u]] = d[u] - 1, \ldots, d[v] = 0$ besitzt p Länge $d[u]$.
- **Annahme:** $\exists$ Pfad $p' = (u = v_0, \ldots, v = v_k)$ mit Länge $k < d[u]$.
- Für jede Kante $\{u', v'\} \in E$ gilt $d[u'] \leq d[v'] + 1$. Damit folgt

$$d[u] \leq d[v_1] + 1 \leq \ldots \leq d[v_k] + k = k.$$

(Widerspruch zu $k < d[u]$.)

Laufzeit Breitensuche

Beweis: Laufzeit

- Schritt 1: $\mathcal{O}(|V|)$, Schritt 2: $\mathcal{O}(1)$.
- Schritt 3: Sei s in der ZHK $V' \subseteq V$.
- Für alle $v \in V'$ werden alle Nachbarn $u \in \Gamma(v)$ besucht, d.h.

$$\sum_{v \in V'} |\Gamma(v)| \leq \sum_{v \in V} \deg(v) = 2|E|.$$

  Damit ist die Laufzeit von Schritt 3 $\mathcal{O}(|E|)$.

Spannbaum mit kürzesten u-v Pfaden

Satz Spannbaum mit kürzesten u-v Pfaden

BREITENSUCHE berechnet bei Eingabe eines zusammenhängenden $G = (V, E)$ und $v \in V$ in Zeit $\mathcal{O}(|V| + |E|)$ einen Spannbaum $T = (V, E')$, $E' = \{\{u, \text{pred}(u)\} \mid u \in V \setminus \{v\}\}$ mit kürzesten u-v Pfaden.

Beweis:

- Kürzeste u-v Pfade in G sind von der Form $(u, \text{pred}[u], \ldots, v)$.
- Wegen $E' = \{\{u, \text{pred}(u)\} \mid u \in V \setminus \{v\}\}$ gilt $|E'| = n - 1$.
- Zeigen nun, dass G zusammenhängend ist. Damit folgt zusammen mit $|E'| = n - 1$, dass T ein Baum ist.
- Für alle $u, w \in V$ sind $p_u = (u, \text{pred}[u], \ldots, v)$ und $p_w = (w, \text{pred}[w], \ldots, v)$ ein u-v bzw. ein w-v Pfad.

Fall 1: p_u, p_w sind bis auf v knotendisjunkt. Dann ist $p = (u, \text{pred}[u], \ldots, v_k, \ldots, \text{pred}[w], w)$ ein u-w Pfad.

Fall 2: p_u, p_w sind nicht knotendisjunkt. Dann ist p aus Fall 1 ein u-w Weg. Dieser kann zu einem u-w Pfad verkürzt werden.

Zusammenhangskomponenten

Algorithmus VOLLSTÄNDIGE BREITENSUCHE

EINGABE: $G = (V, E)$

1. Setze $i \leftarrow 1$.
2. While $V \neq \emptyset$
   1. Starte BREITENSUCHE in beliebigem Startknoten $s \in V$.
   2. $V_i \leftarrow \{v \in V \mid d[v] < \infty\}$.
   3. $V \leftarrow V \setminus V_i$; $G \leftarrow G[V]$; $i \leftarrow i + 1$;

AUSGABE: ZHKs $G[V_1], \ldots, G[V_k]$

Laufzeit für ZHKs

Satz Berechnung von ZHKs

VOLLSTÄNDIGE BREITENSUCHE berechnet bei Eingabe $G = (V, E)$ die ZHKs von G in Zeit $\mathcal{O}(|V| + |E|)$.

Beweis:

- **Laufzeit:** Jeder Knoten wird genau einmal in Q eingefügt. Laufzeit: $\mathcal{O}(|V|)$.
- Jede Kante $\{u, v\}$ wird einmal bei den Nachbarn von u und einmal bei den Nachbarn von v betrachtet. Laufzeit: $\mathcal{O}(|E|)$.
- **Korrektheit:** Folgt aus den Sätzen zuvor.

Tiefensuche DFS (Depth First Search)

Algorithmus TIEFENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. pred[v] $\leftarrow$ nil;
2. $S \leftarrow$ new Stack; S.Push(s);
3. While (S.Isempty() $\neq$ TRUE)
   1. $v \leftarrow S$.Pop();
   2. If ($\exists u \in \Gamma(v) \setminus \{s\}$ mit $\mathrm{pred}[u] =$nil) then
      1. S.Push(v); S.Push(u);
      2. $\mathrm{pred}[u] \leftarrow v$;

AUSGABE: $\mathrm{pred}[v]$ für alle $v \in V$

Rekursive Version von DFS

Algorithmus REKURSIVE TIEFENSUCHE

EINGABE: $G = (V, E)$ als Adjazenzliste, Startknoten $s \in V$

1. Für alle $v \in V$
   1. pred[v] $\leftarrow$ nil;
2. DFS-rekursiv(s);

Funktion DFS-rekursiv(v)

1. While ($\exists u \in \Gamma(v) \setminus \{s\}$ mit $\mathrm{pred}[u] =$nil)
   1. $\mathrm{pred}[u] \leftarrow v$;
   2. DFS-rekursiv(u);

AUSGABE: $\mathrm{pred}[v]$ für alle $v \in V$

Anmerkung:

- Die rekursiven Aufrufe simulieren den Stack S.

Berechnung eines Spannbaums

Satz Berechnung eines Spannbaums

TIEFENSUCHE berechnet bei Eingabe eines zusammenhängenden $G = (V, E)$ und $v \in V$ in Zeit $\mathcal{O}(|E|)$ einen Spannbaum $T = (V, E')$ mit $E' = \{\{u, \text{pred}[u]\} \mid u \in V \setminus \{v\}\}$.

Beweis:

- **Korrektheit:** Es gilt $|E'| = n - 1$.
- Falls T zusammenhängend ist, so ist T ein Baum.
- Für jeden Knoten $u \in V \setminus \{v\}$ ist $(u, \text{pred}[u], \text{pred}[\text{pred}[u]], \ldots, v)$ ein u-v Pfad.
- Konstruktion von u-w Pfaden für alle $u, w \in V$ analog zu BFS.
- **Laufzeit:** Analog zur Analyse bei BFS.

Wurzelbäume

Definition Wurzelbaum

Sei $T = (V, E)$ ein Baum und $v \in V$. Wir definieren den in v gewurzelten Baum T_v wie folgt.

1. v heißt *Wurzel* des Baums.
2. Alle zu v adjazenten Knoten u heißen *Kinder* des *Vater-/bzw. Elternknotens* v.
3. Kinder mit gleichem Elternknoten heißen *Geschwister*.
4. Sei u ein Kind von v. Dann ist u Wurzel eines Teil-Baums, dessen Kinder die Elemente der Menge $\Gamma(u) \setminus \{v\}$ sind.
5. Sei u ein Knoten mit u-v Pfad der Länge k. Dann gilt $\mathrm{Tiefe}[u] = k$.
6. Die *Höhe* des Baums T_v ist $h(T_v) = \max_{u \in V}\{\mathrm{Tiefe}[u]\}$.

Binärbäume

Definition Binärbaum

Sei $T_v = (V, E)$ ein in v gewurzelter Baum.

- T_v heißt *Binärbaum*, falls jeder Knoten höchstens zwei Kinder besitzt.
- Ein Binärbaum heißt *vollständig*, falls jedes Nicht-Blatt genau zwei Kinder besitzt und alle Blätter gleiche Tiefe haben.

Bemerkungen:

- Vollständige Binärbäume können als Array realisiert werden.
- Dabei erhalten Knoten in Tiefe t Indizes $2^t, \ldots, 2^{t+1} - 1$.
- Die Kinder eines Nicht-Blatts i besitzen Indizes $2i$ und $2i + 1$.
- Der Vaterknoten eines Knotens i ist $\lfloor \frac{i}{2} \rfloor$.
- Vollständige Bäume mit n Knoten besitzen Höhe $\Theta(\log n)$.

Binäre Suchbäume

Definition Binärer Suchbaum

Sei $T_v = (V, E)$ ein Binärbaum mit $V \subseteq \mathbb{Z}$. T_v heißt *Suchbaum*, falls für alle $u \in V$ gilt

1. Für alle u_l im linken Teilbaum von u gilt $u_l \leq u$.
2. Für alle u_r im rechten Teilbaum von u gilt $u_r \geq u$.

Suche in Binärbäumen

Algorithmus SUCHE-ELEMENT

EINGABE: Suchbaum $T_v = (V, E)$, u

1. While ($w \neq u$ und w ist kein Blatt)
   1. If ($u \leq w$) then $w \leftarrow$ linkes Blatt von w.
   2. Else $w \leftarrow$ rechtes Kind von w.

AUSGABE: $\begin{cases} u \text{ gefunden} & , \text{falls } w = u \\ u \text{ nicht gefunden} & , \text{sonst} \end{cases}$

Laufzeit:

- Die Laufzeit ist $\mathcal{O}(h(T_v))$.
- Verschiedene Strategien, um $h(T_v) = \mathcal{O}(\log |V|)$ zu erzwingen: höhen-/gewichtsbalancierte Bäume, Rot-Schwarz-Bäume, etc.

Hamiltonscher Kreis

Definition Hamiltonscher Kreis

Sei $G = (V, E)$ ein zusammenhängender Graph.

1. Ein *Hamiltonscher Kreis* in G ist ein Kreis, der alle Knoten enthält.
2. Graphen mit Hamiltonkreis bezeichnet man als *hamiltonsch*.

Bsp:

- 5 Personen setzen sich an einen runden Tisch.
- Frage: Gibt es zwei Konstellationen, bei denen jeder andere Nachbarn besitzt?
- Modellierung als Graphproblem: u, v sind Nachbarn, falls $\{u, v\}$ Kante eines Hamiltonkreises ist.
- D.h. wir suchen zwei Konstellationen mit kantendisjunkten Hamiltonkreisen.
- Der K_5 besitzt zwei kantendisjunkte Hamiltonkreise.

Naiver Algorithmus für Hamiltonkreis

Algorithmus HAMILTON

EINGABE: $G = ([n], E)$ in Adjazenzmatrixdarstellung

1. Für alle Permutationen $\pi : [n] \to [n]$.
   1. Falls $(\pi(1), \pi(2), \ldots, \pi(n))$ ein Kreis in G ist, AUSGABE "G hamiltonsch", EXIT.
2. AUSGABE "G nicht hamiltonsch".

- **Korrektheit:** HAMILTON testet alle möglichen Hamiltonkreise.
- **Laufzeit:** Schleife 1 durchläuft $n!$ viele Iterationen.
- Stirling-Formel liefert $n! = \Theta\left(\sqrt{n}\left(\frac{n}{e}\right)^n\right)$.
- D.h. die Laufzeit ist exponentiell in n.
- In Diskrete Mathematik 2: Vermutlich gibt es für allgemeine Graphen keinen Algorithmus mit Laufzeit polynomiell in n.

Hamiltonkreis in dichten Graphen

Satz Hinreichendes Kriterium für hamiltonsch

Sei $G = (V, E)$ mit $\deg(u) + \deg(v) \geq |V|$ für alle $u, v \in V$. Dann ist G hamiltonsch.

Beweis:

- **Annahme:** $G = (V, E)$ besitze keinen Hamiltonkreis.
- Sei E maximal, so dass G nicht hamiltonsch ist.
- D.h. für alle $\{u, v\} \notin E$ ist $G' = (V, E \cup \{u, v\})$ hamiltonsch.
- Sei $K = (u = v_{\pi(1)}, v_{\pi(2)}, \ldots, v = v_{\pi(n)})$ ein Hamiltonkreis in G'.
- Wir definieren $S = \Gamma(u)$ und $T = \{v_{\pi(i)} \in K \mid \{v, v_{\pi(i-1)}\} \in E\}$.
- Es gilt $u \notin S$ und $u \notin T$. Damit ist $|S \cup T| \leq n - 1$.
- Andererseits gilt $|S| + |T| = \deg(u) + \deg(v) \geq n$.
- Schubfachprinzip: Es gibt ein $j \in \{2, \ldots, n-1\}$ mit $v_{\pi(j)} \in S \cap T$.
- Damit ist sowohl $\{u, v_{\pi(j)}\} \in E$ als auch $\{v, v_{\pi(j-1)}\} \in E$.
- D.h. $(u, v_{\pi(j)}, v_{\pi(j+1)}, \ldots, v, v_{\pi(j-1)}, \ldots, v_{\pi(2)})$ ist Hamiltonkreis. (Widerspruch zur Annahme, dass G nicht hamiltonsch ist.)

Algorithmus für dichte Graphen

Algorithmus DENSE-HAMILTON

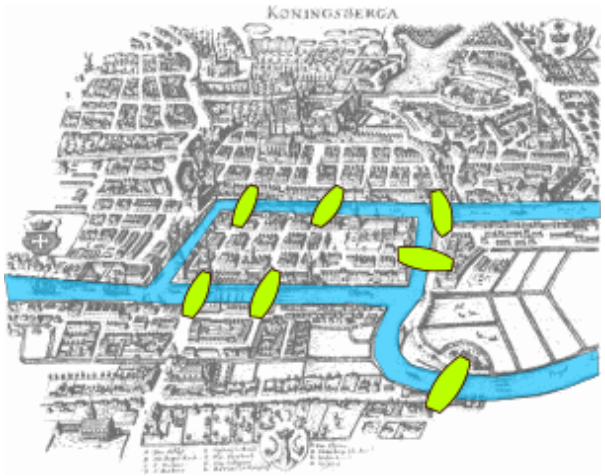
EINGABE: $G = ([n], E)$ mit $deg(u) + \deg(v) \geq n$ für alle $u, v \in V$.

1. Setze $K = (v_0, v_1, \ldots, v_{n-1}) \leftarrow (1, 2, \ldots, n)$
2. While (K ist kein Kreis in G)
   1. Wende auf K die Konstruktion aus dem Beweis zuvor an.

AUSGABE: Hamiltonkreis K

- **Korrektheit:** Fügen in jeder Iteration zwei Kreiskanten hinzu.
- **Laufzeit:** $\mathcal{O}(n)$ Iterationen.

Königsberger Brückenproblem (Euler 1736)



Frage: Gibt es einen Rundweg durch Königsberg, der alle Brücken genau einmal verwendet?

Definition Eulertour

Sei $G = (V, E)$ zusammenhängend.

1. Eine *Eulertour* in G ist ein Weg, der alle Kanten genau einmal besucht und bei dem Anfangs- und Endknoten übereinstimmen.
2. Falls G eine Eulertour enthält, heißt G *eulersch*.

Kriterium für eulersche Graphen

Satz Eulerscher Graph

Sei $G = (V, E)$ zusammenhängend. G ist eulersch gdw $\deg(v)$ gerade ist für alle $v \in V$.

Beweis: $\Rightarrow$

- Sei G eulersch mit Eulertour $p = (v_0, v_1, \ldots, v_k, v_0)$.
- Jeder Knoten $v \neq v_0$ komme t-mal in p vor.
- Dann ist $\deg(v) = 2t$ gerade.
- Der Knoten v_0 komme $t + 2$-mal, $t \geq 0$, in p vor.
- Dann ist $\deg(v_0) = 2t + 2$ ebenfalls gerade.

$\deg(v)$ gerade $\Rightarrow$ G eulersch

Algorithmus EULERTOUR

EINGABE: $G = (V, E)$ zusammenhängend mit geraden Knotengraden

1. Wähle Weg $W_0 \leftarrow (v)$ für ein beliebiges $v \in V$. Setze $i \leftarrow 0$.
2. While (nicht alle Kanten von G besucht)
   1. $i \leftarrow i + 1$
   2. Wähle Knoten v_i in $W_{i-1} = (v_0, \ldots, v_k = v_0)$, der zu nicht besuchter Kante $\{v_i, u\}$ inzident ist.
   3. Konstruiere Weg $W_i' = (v_i, u, \ldots, v_i)$.
   4. $W_i \leftarrow (v_0, \ldots, v_i, u, \ldots, v_i, \ldots, v_0)$, d.h. verschmelze W_{i-1} und W_i'.

AUSGABE: Eulertour W_i

Korrektheit von EULERTOUR

Korrektheit:

- Bei Terminierung sind alle Kanten besucht.
- EULERTOUR terminiert, da G zusammenhängend ist.
- Die Anzahl besuchter Kanten pro Knoten ist für jedes W_i gerade:
  - Jeder der inneren Knoten $v_1, \ldots, v_{k-1}$ wird über eine Kante besucht und wieder verlassen.
  - Da jedes der W_i ein nicht knotendisjunkter Kreis ist, wird auch der Startknoten v_0 verlassen und wieder besucht.
- D.h. jeder Knoten $v \in V$ besitzt geraden Restgrad $\deg(v)$.
- W_i' kann in Schritt 3.3 konstruiert werden, da alle besuchten Knoten auch wieder verlassen werden können.
- Ein solcher Weg muss schließlich wieder in v_i enden.

Planare Graphen

Definition Planarer Graph

Sei $G = (V, E)$ ein ungerichteter Graph. G heißt *planar*, falls er in den $\mathbb{R}^2$ einbettbar ist. D.h. falls seine Kanten so dargestellt werden können, dass sie sich paarweise nicht schneiden.
Eine planare Darstellung von G bezeichnen wir als *ebenes Diagramm*.

Anmerkung:

- Der Einfachheit halber verstehen wir hier Kanten als Kurven.
- Die Verwendung von Strecken als Kanten liefert dieselbe Klasse planarer Graphen (Fary's Theorem).

Bsp:

- Der K_4 ist ein planarer Graph.
- Ein bipartiter Graph $G = K_{n,n'}$ besteht aus zwei Knotenmengen V_1, V_2 mit $|V_1| = n, |V_2| = n'$ und $E = \{\{u, v\} \mid u \in V_1, v \in V_2\}$.
- Die bipartiten Graphen $K_{n,2}$ sind planar.

Anzahl Flächen planarer Graphen

Satz Eulersche Polyederformel

Sei $G = (V, E)$ zusammenhängend und planar. Sei f die Anzahl der Flächen eines ebenen Diagramms von G. Dann gilt $f = |E| - |V| + 2$.

Beweis: per Induktion über $|E|$

- **IV** für $|E| = n - 1$, da G für $|E| < n - 1$ nicht zusammenhängend.
  - Da G zusammenhängend mit $|E| = n - 1$, ist G ein Baum.
  - Damit gilt für ein ebenes Diagramm $f = 1 = (n - 1) - n + 2$.
- **IS** $|E| - 1 \rightarrow |E|$.
  - $G = (V, E)$ muss einen Kreis K enthalten. Sei e eine Kreiskante.
  - $G' = (V, E \setminus \{e\})$ besitzt $|E| - 1$ Kanten und daher $f' = |E| - |V| + 1$ Flächen. (Anwendung der IA)
  - In G werden zwei Flächen von G' durch e getrennt.
  - Damit besitzt G genau $f' + 1 = |E| - |V| + 2$ Flächen.

Nicht zusammenhängende Graphen

Korollar Allgemeine Eulersche Polyederformel

Sei $G = (V, E)$ planar mit k ZHKs. Dann besitzt ein ebenes Diagramm von G genau $f = |E| - |V| + k + 1$ Flächen.

Beweis:

- Sei $G[V_1] = (V_1, E_1), \ldots, G[V_k] = (V_k, E_k)$ die ZHKs.
- Für jedes ebene Diagramm von $G[V_i]$ gilt $f_i = |E_i| - |V_i| + 2$.
- Dabei wird die Außenfläche für jede ZHK gezählt. D.h.

$$\begin{aligned} f &= \left(\sum_{i=1}^{k} |E_i| - |V_i| + 2 \right) - (k-1) \\ &= \left(\sum_{i=1}^{k} |E_i| - \sum_{i=1}^{k} |V_i| \right) + 2k - k + 1 = |E| - |V| + k + 1. \end{aligned}$$

Kriterium für nicht-planare Graphen

Satz Anzahl Kanten in planaren Graphen

Für jeden planaren Graphen $G = (V, E)$ gilt $|E| \leq 3(|V| - 2)$.

Beweis:

- Wir bezeichnen mit $R = \{r_1, \ldots, r_f\}$ die Flächen von G.
- Definieren Relation $A = \{(e, r) \in E \times R \mid e \text{ ist in Fläche } r\}$.
- Dabei ist eine Kante e auch in r, falls sie r begrenzt.
- Prinzip des doppelten Abzählens:
  - Zeilensumme: Jedes e begrenzt höchstens zwei Flächen, d.h. $|A| \leq 2|E|$.
  - Spaltensumme: Jede Fläche wird von mindestens 3 Kanten begrenzt, d.h. $|A| \geq 3f$.
- Daraus folgt $3f = 3(|E| - |V| + 2) \leq 2|E|$ bzw. $|E| \leq 3(|V| - 2)$.

Für planare G ohne Kreise der Länge 3 folgt sogar die Ungleichung

$$4(|E| - |V| + 2) \leq 2|E| \Leftrightarrow |E| \leq 2(|V| - 2).$$

Nicht-planare Graphen

Korollar K_5 und $K_{3,3}$

Die Graphen K_n, $n \geq 5$ und $K_{3,3}$ sind nicht planar.

Beweis:

- Der K_n besitzt $\binom{n}{2}$ Kanten.
- Es gilt $\binom{n}{2} \geq 3(n-2)$ für $n \geq 5$.

- Zeigen: Bipartite Graphen besitzen keine Kreise der Länge 3.
- **Annahme:** $K_{n,n'}$ besitzt Kreis $K = (v_1, v_2, v_3)$ der Länge 3.
- v_1 und v_3 sind in derselben Knotenpartition von $K_{n,n'}$.
- Daher kann keine Kante $\{v_3, v_1\}$ existieren (Widerspruch).
- Damit gilt für den $K_{3,3}$ die stärke Schranke $|E| \leq 2(|V| - 2)$.
- Für den $K_{3,3}$ folgt aber $|E| = 9 > 2 \cdot 4 = 2(|V| - 2)$.

Satz von Kuratowski

Definition Unterteilung eines Graphen

Sei $G = (V, E)$ und $e = \{u, v\} \in E$.

1. Das Einfügen eines neuen Knoten w in die Kante e führt zum Graphen $G' = (V \cup \{w\}, E \setminus e \cup \{\{u, w\}, \{w, v\}\})$.
2. Der Graph $H = (V', E')$ heißt *Unterteilung von G*, falls H aus G durch Einfügen von Knoten konstruiert werden kann.

Satz von Kuratowski

Ein Graph $G = (V, E)$ ist planar gdw G keine Unterteilung von K_5 oder $K_{3,3}$ als schwachen Teilgraphen enthält.

Grad eines Knoten in planaren Graphen

Satz Knoten kleinen Grads

Sei $G = (V, E)$ planar. Dann existiert $v \in V$ mit $\deg(v) \leq 5$.

Beweis:

- **Annahme:** $\deg(v) \geq 6$ für alle $v \in V$.
- Dann folgt $2|E| = \sum_{v \in V} \deg(v) \geq 6|V|$ bzw. $|E| \geq 3|V|$.
  (Widerspruch: $|E| < 3(|V| - 2)$)

Färben von Knoten eines Graphen

Definition *k*-Färbbarkeit und chromatische Zahl

Sei $G = (V, E)$ ein Graph. G heißt *k-färbbar*, falls es eine Abbildung $c : V \to [k]$ gibt mit $c(u) \neq c(v)$ für alle $\{u, v\} \in E$. Die *chromatische Zahl* von G ist definiert als $\chi(G) = \min\{k \in \mathbb{N} \mid G \text{ ist } k\text{-färbbar.}\}$

Bsp:

- Jeder Graph mit n Knoten ist n-färbbar.
- Die chromatische Zahl des K_n ist n.
- Jeder Kreis der Form C_{2n} ist 2-färbbar.
- Jeder Kreis der Form C_{2n+1} ist 3-färbbar.
- Bipartite Graphen $K_{n,n'}$ sind 2-färbbar.
- Jeder Baum ist 2-färbbar.

Anwendungen:

- Verteilen von Senderfrequenzen
- Färben von Landkarten

Bipartite Graphen

Satz 2-Färbbarkeit und Kreise

Sei $G = (V, E)$ ein Graph. G ist 2-färbbar gwd G keine Kreise ungerader Länge enthält.

Beweis:

- "$\Rightarrow$": **Annahme:** G enthält Kreis $K = (v_1, \ldots, v_{2k+1})$.
- Sei $c : V \to [2]$ eine 2-Färbung von G.
- Dann gilt $c(v_1) = c(v_3) = \ldots = c(v_{2k+1})$, aber $\{v_{2k+1}, v_1\} \in E$. (Widerspruch: c ist eine Färbung.)

- "$\Leftarrow$": Starte BREITENSUCHE in beliebigem $s \in V$.
- Definiere Färbung $c : V \to [2]$ mit $c(v) = (d[v] \bmod 2) + 1$.
- c ist eine 2-Färbung für den durch die Breitensuche definierten Baum mit den Kanten $\{v, \mathrm{pred}[v]\}$ für alle $v \in V$, $v \neq s$.
- Alle weiteren Kanten in G schließen Kreise gerader Länge.
- D.h. alle benachbarten Knoten erhalten unterschiedliche Farben.

5-Färbbarkeit

Satz von Heawood

Sei $G = (V, E)$ planar. Dann ist G ein 5-färbbarer Graph.

Beweis: per Induktion über $|V|$

- **IV**: Satz ist für $|V| \leq 5$ sicherlich korrekt.
- **IS** $|V| - 1 \rightarrow |V|$: Betrachten ebenes Diagramm von $G = (V, E)$.
- G enthält einen Knoten v mit $\deg(v) \leq 5$.
- Nach IS existiert für $G[V \setminus \{v\}]$ eine 5-Färbung c.

Fall 1: G besitzt höchstens 4 Nachbarn. Dann kann v mit einer Farbe gefärbt werden, die für keinen der Nachbarn verwendet wird.

Fall 2: G besitzt 5 Nachbarn $v_1, \ldots, v_5$ mit $\{c(v_1), \ldots, c(v_5)\} \neq [5]$. Dann kann v mit einer Restfarbe gefärbt werden.

Fall 3: G besitzt 5 Nachbarn $v_1, \ldots, v_5$ mit Farben $c(v_i) = i$ für $i \in [5]$. Idee: Färbe einen der Nachbarn von v um. (s. nächste Folie)

Fall 3: Alle Nachbarn unterschiedlich gefärbt.

- Definieren $V_i = \{v \in V \mid c(v) = i\}$.
- Fall 3(a): v_1 und v_3 sind in verschiedenen ZHKs von $G[V_1 \cup V_3]$.
- Tausche Farben 1 und 3 in der ZHK von $G[V_1 \cup V_3]$ in der v_1 liegt.
- Danach besitzt kein Nachbar von v Farbe 1. Färbe $c(v) = 1$.
- Fall 3(b): v_1 und v_3 liegen in derselben ZHK von $G[V_1 \cup V_3]$.
- Dann existiert ein v_1-v_3 Pfad ausschließlich mit Farben 1 und 3.
- Damit existiert kein v_2-v_4 Pfad mit den Farben 2 und 4, da dieser wegen der Planarität Knoten des v_1-v_3 Pfades kreuzen muss.
- Vertausche analog zu Fall 3(a) die Farben 2 und 4 in der ZHK von $G[V_2 \cup V_4]$, in der v_2 liegt. Färbe v mit Farbe 2.

Vierfarbensatz

Satz Vierfarbensatz von Appel, Haken (1977)

Sei G planar. Dann ist G ein 4-färbbarer Graph.

Bemerkungen:

- Beweis der Korrektheit durch massiven Computereinsatz.
- Beweis liefert $\mathcal{O}(|V^2|)$-Algorithmus für planare Graphen.
- Für allgemeine Graphen $G = (V, E)$ gibt es vermutlich keinen effizienten Algorithmus, der für gegebenes k entscheidet, ob $\chi(G) \leq k$.

Nicht-optimale Färbung

Algorithmus GREEDY-FÄRBUNG

EINGABE: $G = ([n], E)$

1. $c[1] \leftarrow 1$; $C \leftarrow 1$
2. For $i \leftarrow 2$ to n
   - $c[i] = \min_{k \in \mathbb{N}}\{k \neq c[v]$ für alle bereits gefärbten $v \in \Gamma(i)\}$.
   - If $(c[i] > C)$ $C \leftarrow c[i]$;

AUSGABE: Färbung $c : V \rightarrow [C]$

- **Korrektheit:** c ist Färbung, da alle Nachbarknoten verschiedene Farben erhalten.
- **Laufzeit:** $\mathcal{O}(|V|)$ Iterationen.
- **Güte der Lösung:** Sei $\Delta(G) = \max_v\{\deg(v_i)\}$.
- Es gilt $\chi(G) \leq C \leq \Delta(G) + 1$.
- Für K_n oder C_{2n+1} gilt $\chi(G) = \Delta(G) + 1$.
- Für alle andere Graphen gibt es einen effizienten Algorithmus, der eine Färbung mit höchstens $\Delta(G)$ Farben liefert.

Kantenfärbung von Graphen

Definition Kantenfärbung

Sei $G = (V, E)$ ein Graph. Eine *Kantenfärbung* von G ist eine Abbildung $c : E \to [k]$ mit $c(e) \neq c(e')$ für $e, e' \in E$ mit $e \cap e' \neq \emptyset$. Der *chromatische Index* ist $\chi'(G) = \min_{k \in \mathbb{N}} \{G \text{ besitzt } k\text{-Kantenfärbung.}\}$.

- Es ist leicht zu sehen, dass $\chi'(G) \geq \Delta(G)$.
- Der sogenannte Satz von Vinzing zeigt, dass $\chi'(G) \leq \Delta(G) + 1$.
- Das Problem zu entscheiden ob $\chi'(G) = \Delta(G)$ ist, gehört zu den algorithmisch schweren Problemen.
- Es gibt allerdings einen Algorithmus, der in Zeit $\mathcal{O}(|V| \cdot |E|)$ eine k-Kantenfärbung berechnet mit $\Delta(G) \leq \chi'(G) \leq k \leq \Delta(G) + 1$.

Das Heiratsproblem

Szenario:

- Gegeben: n Frauen und $m > n$ Männer.
- Bekanntschaftsbeziehungen zwischen allen Männern und Frauen.

Fragestellung:

- Wann gibt es für jede der Frauen einen Heiratspartner?
- Modellierung als bipartiter Graph $K_{n,m}$.
- Gesucht ist knotendisjunkte Kantenmenge E der Größe n.

Definition Matching

Sei $G = (V, E)$ ein Graph. Ein *Matching* $M \subseteq E$ ist eine knotendisjunkte Kantenmenge, d.h. dass für alle verschiedenen $e, e' \in M$ gilt $e \cap e' = \emptyset$. Die Größe von M ist $|M|$.

1. M *überdeckt* ein $v \in V$ gdw $v \cap e \neq \emptyset$ für ein $e \in M$.
2. M ist *perfekt*, falls M alle Knoten überdeckt, d.h. falls $|M| = \frac{|V|}{2}$.

Heiratssatz

Satz von Hall

Sei $G = (A \dot\cup B, E)$ bipartit mit Knotenpartitionen $A \dot\cup B$. G enthält ein Matching M der Größe $|M| = |A|$ gdw $|X| \leq |\bigcup_{x \in X} \Gamma(x)|$ für alle $X \subseteq A$.

- Sei $\Gamma(X) = \bigcup_{x \in X} \Gamma(x)$ für alle $X \subseteq A$
- "$\Rightarrow$": Sei M ein Matching der Größe $|M| = |A|$.
- Wir betrachten den vom Matching definierten schwachen Teilgraph $G' = (A \dot\cup B, M)$.
- In G' besitzt jedes $X \subseteq A$ genau $|X|$ Nachbarn.
- Damit besitzt in G jedes $X \subseteq A$ mindestens $|X|$ Nachbarn.

Rückrichtung des Heiratssatzes

- "$\Leftarrow$": **Annahme:** G besitzt maximales Matching M mit $|M| < |A|$.
- Dann existiert ein nicht-überdecktes $a_1 \in A$.
- Wegen $|\Gamma(a_1)| \geq 1$ besitzt a_1 mindestens einen Nachbarn.
- Falls $b_1 \in \Gamma(a_1)$ nicht-überdeckt ist, füge $\{a_1, b_1\}$ zu M hinzu. (Widerspruch: M ist ein maximales Matching.)
- Sonst wähle überdecktes b_1 und starte folgenden Algorithmus AUGMENTIERENDER-PFAD.

Algorithmus AUGMENTIERENDER-PFAD

Algorithmus AUGMENTIERENDER-PFAD

EINGABE: $G = (A \dot{\cup} B, E)$, M, a_1, b_1

1. $k \leftarrow 1$
2. While (b_k wird von M überdeckt)
   1. $a_{k+1} \leftarrow$ Nachbar von b_k im Matching M
   2. If ($\exists$ nicht-überdecktes $v \in \Gamma(\{a_1, \ldots, a_{k+1}\}) \setminus \{b_1, \ldots, b_k\}$)
      1. $b_{k+1} \leftarrow v$
      2. $b_{k+1} \leftarrow$ beliebiges $v \in \Gamma(\{a_1, \ldots, a_{k+1}\}) \setminus \{b_1, \ldots, b_k\}$
   3. $k \leftarrow k + 1$.

AUSGABE: augmentierender Pfad $p_a = (a_1, b_1, \ldots, a_k, b_k)$

Korrektheit von AUGMENTIERENDER-PFAD

Korrektheit:

- Zeigen zunächst, dass b_{k+1} in Schritt 2.2 stets existiert. Es gilt
$$|\Gamma(\{a_1, \ldots, a_{k+1}\}) \setminus \{b_1, \ldots, b_k\}| \geq (k+1) - k = 1.$$
- Für alle $\{a_i, b_i\}$, $i \in [k]$ gilt, dass $\{a_i, b_i\} \in E \setminus M$.
- Damit sind diese k Kanten nicht im Matching enthalten.
- Die $k-1$ Kanten $\{b_i, a_{i+1}\}$ sind dagegen alle in M.
- D.h. durch Entfernen aller $\{b_i, a_{i+1}\}$ aus M und Hinzunahme aller $\{a_i, b_i\}$ zu M wird das Matching um Eins vergrößert.

  (Widerspruch: M ist nach Annahme maximal.)

Konstruktion eines maximalen Matchings

Algorithmus Maximales-Matching

EINGABE: $G = (A \cup B, E)$ mit $|X| \leq |\Gamma(X)|$ für alle $X \subseteq A$

1. $M \leftarrow \emptyset$
2. While (es gibt ein nicht-überdecktes $a_1 \in A$)
   1. $b_1 \leftarrow$ Nachbar von a_1
   2. $p = (a_1, b_1, \ldots, a_k, b_k) \leftarrow$ AUGMENTIERENDER-PFAD(G, M, a_1, b_1)
   3. For $i \leftarrow 1$ to k
      1. $M \leftarrow M \cup \{a_i, b_i\}$; If $(i > k)$ $M \leftarrow M \setminus \{b_i, a_{i+1}\}$;

AUSGABE: Matching M mit $|M| = |A|$

Korrektheit und Laufzeit:

- M wird in jeder Iteration um ein Element vergrößert.
- Nach $|A|$ Iterationen gilt $|M| = |A|$.

k-reguläre bipartite Graphen

Satz Perfekte Matchings für bipartite Graphen

Sei $G = (A \dot\cup B, E)$ ein k-regulärer bipartiter Graph. Dann gilt:

1. G besitzt ein perfektes Matching.
2. Der chromatische Index von G beträgt $\chi'(G) = k$.

Beweis: von (1)

- Falls $|X| \leq |\Gamma(X)|$ für alle $X \subseteq A$, verwende Satz von Hall.
- Liefert Matching für alle $a \in A$. Falls $|A| = |B|$, dann ist M perfekt.
- In jedem k-regulären bipartiten $G = (A \dot\cup B, E)$ gilt $|A| = |B|$, denn
$$|E| = \sum_{a \in A} \deg(a) = |A| \cdot k = \sum_{b \in B} \deg(b) = |B| \cdot k.$$
- **Annahme:** Es existiert ein $X \subseteq A$ mit $|\Gamma(X)| < |X|$.
- Wir betrachten die Multimenge $M = \bigcup_{x \in X} \Gamma(x)$.
- Da G ein k-regulärer Graph ist, gilt $|M| = k \cdot |X|$.
- Schubfachprinzip: $\exists b \in \Gamma(X)$ mit $\deg(b) \geq \left\lceil \frac{|M|}{|\Gamma(X)|} \right\rceil > \frac{k \cdot |X|}{|X|} = k$.
(Widerspruch: G ist k-regulär, d.h. $deg(b) = k$ für alle $b \in B$.)

$\chi'(G) = k$ in k-regulären bipartiten Graphen

Beweis: von (2)

- Wir beweisen $\chi'(G) = k$ per Induktion über k.
- **IV** für $k = 1$: G besitzt nach (1) ein perfektes Matching M.
- Daher sind alle $a \in A$ mit genau einem $b \in B$ verbunden.
- D.h. $M = E$ und alle Kanten können mit Farbe 1 gefärbt werden.
- **IS** $k - 1 \rightarrow k$: G besitzt nach (1) ein perfektes Matching M.
- $G' = (A \dot{\cup} B, E \setminus M)$ ist $(k-1)$-regulär.
- D.h. G' besitzt nach IA eine $(k-1)$-Kantenfärbung.
- Wir färben alle Kanten des Matchings M mit der Farbe k.

Gerichtete Graphen

Definition Gerichteter Graph bzw. Digraph

Ein *gerichteter Graph bzw. Digraph* ist ein Tupel $D = (V, E)$ mit

- der Knotenmenge $V = \{v_1, \ldots, v_n\}$ und
- der Kantenmenge $E = \{(u, v) \subseteq V \times V \mid u \neq v\}$.

Anmerkung: Wir definieren analog zu ungerichteten Graphen

- Weg, Pfad, u-v Pfad, Kreis, kürzester Pfad
- Nachbarschaft $\Gamma(v) = \{u \in V \mid (v, u) \in E\}$.

Definition DAG

Sei $D = (V, E)$ ein Digraph. D heißt *DAG* (Directed Acyclic Graph), falls D kreisfrei ist.

DAGs und die topologische Sortierung

Algorithmus TOPOLOGISCHE SORTIERUNG (DFS-Variante)

EINGABE: DAG $D = (V, E)$ in Adjazenzlistendarstellung

1. For alle $v \in V$: $\text{pred}[v] \leftarrow$ nil; $f[v] \leftarrow 0$;
2. $i \leftarrow n$; $S \leftarrow$ new Stack;
3. While (es gibt unbesuchte Knoten $s \in S$)
   1. $s \leftarrow$ minimaler unbesuchter Knoten; S.Push(s);
   2. While (S.Isempty() $\neq$ TRUE)
      1. $v \leftarrow S$.Pop();
      2. If (es gibt ein $u \in \Gamma$ mit $\text{pred}[u] =$nil) then
         S.Push(v); S.Push(u);
         $\text{pred}[u] \leftarrow v$;
      3. else if ($f[v] = 0$) then
         $f[v] \leftarrow i$; $i \leftarrow i - 1$;

AUSGABE: $\text{pred}[v], f[v]$ für alle $v \in V$

Finish-Zahl $f[v]$

Satz Finish-Zahl

Sei $D = (V, E)$ ein DAG und sei $f[v]$ für alle $v \in V$ definiert durch Anwendung von TOPOLOGISCHE SORTIERUNG auf D. Für $u, v \in V$ gilt: Falls $f[u] > f[v]$, dann folgt $(u, v) \notin E$.

Beweis:

- **Annahme:** $(u, v) \in E$
- Es gibt keinen v-u Pfad in G, denn dieser würde mit (u, v) zusammen einen Kreis schließen.
- Jedem $w \in V$ wird ein Wert $f[w] > 0$ zugewiesen, sobald alle Nachbarn von w besucht sind und w vom Stack entfernt wird. (Fortsetzung auf nächster Folie.)

Finish-Zahl $f[v]$

- **Fall 1:** Knoten v wird vor Knoten u besucht.
- Damit wird v vom Stack entfernt, bevor u auf den Stack kommt.
- Da der Zähler i für die Finishzahl dekrementiert wird, gilt $f[v] > f[u]$. (Widerspruch: $f[u] > f[v]$ nach Voraussetzung)
- **Fall 2:** Knoten u wird vor Knoten v besucht.
- Wegen $\{u, v\} \in E$ liegt u im Stack unter v.
- Sobald v vom Stack entfernt wird, erhält v ein $f[v] > 0$.
- Danach erst wird u eine Finish-Zahl $f[u]$ zugewiesen.
- Daraus folgt wiederum $f[u] < f[v]$. (Widerspruch)

Anwendung der Topologischen Sortierung

Szenario:

- **Gegeben:** n Aufgaben und Abhängigkeiten $(u, v) \in [n]^2$ mit der Bedeutung, dass u vor v ausgeführt werden muss.
- **Gesucht:** Gültige Reihenfolge, alle Aufgaben abzuarbeiten.

- Modellierung als Digraph D mit gerichteten Kanten (u, v).
- Falls D kein DAG ist, dann existiert ein Kreis in D. Dann sind die Abhängigkeiten widersprüchlich und es existiert keine Lösung.
- Sei also D ein DAG. Dann kann mit TOPOLOGISCHE SORTIERUNG für jeden Knoten $v \in V$ eine Finish-Zahl $f[v]$ berechnet werden.
- Ordne die Knoten nach steigender Finish-Zahl $f[v]$.
- Für $d[u] > d[v]$ gilt $\{u, v\} \notin E$.
- D.h. Aufgabe u kann nach Aufgabe v ausgeführt werden.
- Damit können alle Aufgaben in der Reihenfolge steigender Finish-Zahl durchgeführt werden.

Relationen und Graphen, starker Zusammenhang

Anmerkung:

- Sei $D = (V, E)$. Dann ist $A \subseteq V \times V$ eine Relation auf V.
- Sei andererseits $R \subseteq S \times S$ eine Relation auf S.
- Dann definiert $D = (S, R)$ einen gerichteten Graphen.
- D.h. Gerichtete Graphen sind Darstellungen von Relationen.

Definition starker und schwacher Zusammenhang

Sei $D = (V, E)$ ein gerichteter Graph. Der ungerichtete $G = (V, E')$ mit $E' = \{\{u, v\} \in V^2 \mid (u, v) \in E \text{ oder } (v, u) \in E\}$ heißt *zugrundeliegender Graph* von D.

1. D ist *stark zusammenhängend*, falls für alle $u, v \in V$ ein u-v Pfad in D existiert.
2. D ist *schwach zusammenhängend*, falls der zugrundeliegende Graph G von D zusammenhängend ist.

Starker Zusammenhang algorithmisch

Zwei Algorithmen zum Testen von starkem Zusammenhang:

- Führe TIEFENSUCHE für alle Startknoten $s \in V$ durch.
- Falls stets alle Knoten erreicht werden, dann ist G stark zusammenhängend.
- Laufzeit dieses Algorithmus: $\mathcal{O}(|V| \cdot (|V| + |E|))$.

- Verbesserter Algorithmus: Verwendung von Ruckwärtsgraph $\bar{D}$.
- Digraph $\bar{D} = (V, \bar{E})$ besitzt $\bar{E} = \{(u, v) \in V^2 \mid (v, u) \in E\}$.
- Man kann zeigen, dass eine Anwendung von Tiefensuche auf D und auf $\bar{D}$ genügt. (Beweis ist nicht-trivial)
- Führt zu einem Algorithmus mit Laufzeit $\mathcal{O}(|V| + |E|)$.

Berechnung der transitiven Hülle

Algorithmus SIMPLE-TRANSITIV

EINGABE: Relation $R \subseteq S \times S$ in Adjazenzmatrix-Darstellung

1. While ($\exists x, y, z$ mit $(x, y), (y, z) \in R$ und $(x, z) \notin R$)
   1. $R \leftarrow R \cup \{(x, z)\}$

AUSGABE: Transitive Hülle $R^+ = R$

- **Korrektheit:** Solange transitive Beziehungen fehlen, werden diese zur Relation R hinzugefügt.
- **Laufzeit:** Die Menge R^+ enthält höchstens $|S|^2$ Tupel aus $S \times S$.
- In jeder Iteration wird ein Tupel (x, z) hinzugefügt.
- D.h. die Schleife durchläuft maximal $\mathcal{O}(|S|^2)$ Iterationen.
- Pro Iteration: Prüfe für alle $x, y, z \in S$, ob für die Einträge in der Adjazenzmatrix $(x, y) = (y, z) = 1$ und $(x, z) = 0$ gilt.
- Die Anzahl aller möglichen $x, y, z \in S$ ist $\mathcal{O}(|S|^3)$.
- Damit ist die Gesamtlaufzeit von SIMPLE-TRANSITIV $\mathcal{O}(|S|^5)$.

Mit Dynamischer Programmierung

- Sei $D = (V, E)$. Die transitive Hülle $D^+ = (V, E^+)$ besitzt die Kantenmenge $E^+ = \{(u, v) \in V^2 \mid \exists\ u\text{-}v \text{ Pfad in } D.\}$.

Definition Wege mit eingeschränkten inneren Knoten

Sei $D = ([n], E)$ Wir definieren für $k, u, v \in [n]$

$$W_k[u, v] = \begin{cases} 1 & \exists\ u\text{-}v \text{ Pfad in } D \text{ mit inneren Knoten aus } [k] \\ 0 & \text{sonst} \end{cases}.$$

- Für $k = 0$ gilt $\{(u, v) \in V^2 \mid W_0[u, v] = 1\} = E$.
- Für $k = n$ gilt $\{(u, v) \in V^2 \mid W_n[u, v] = 1\} = E^+$.
- Berechnen $W_k[u, v]$ rekursiv aus W_{k-1}. $W_k[u, v] = 1$, falls
  - Es gibt einen u-v Pfad mit inneren Knoten aus $[k-1]$.
  - Es gibt einen u-k Pfad und k-v Pfad mit inneren Knoten aus $[k-1]$.
- D.h. $W_k[u, v] = \max\{W_{k-1}[u, v], W_{k-1}[u, k] \cdot W_{k-1}[k, v]\}$.

Algorithmus von Warschall

Algorithmus WARSCHALL

EINGABE: $D = ([n], E)$ in Adjazenzmatrix-Darstellung

1. Für alle $u, v \in V$
   1. If $(u, v) \in E$ then $W[u, v] \leftarrow 1$ else $W[u, v] \leftarrow 0$;
2. For $k \leftarrow 1$ to n
   1. Für alle $u, v \in V$
      1. $W[u, v] \leftarrow \max\{W[u, v], W[u, k] \cdot W[k, v]\}$

AUSGABE: $D^+ = (V, E^+)$ mit $E^+ = \{(u, v) \mid W[u, v] = 1\}$

Transitive Hülle in kubischer Laufzeit

Satz Transitive Hülle

Sei $D = (V, E)$. Dann berechnet WARSCHALL die transitive Hülle $D^+ = (V, E^+)$ in Zeit $\mathcal{O}(|V|^3)$ und mit Speicherbedarf $\mathcal{O}(|V|^2)$.

Korrektheit:

- Wissen $W_k[u, v] = \max\{W_{k-1}[u, v], W_{k-1}[u, k] \cdot W_{k-1}[k, v]\}$.
- WARSCHALL verwendet nur ein zweidimensionales Array für W.
- D.h. die benötigten Zwischenwerte $W_{k-1}[u, k]$ bzw. $W_{k-1}[k, v]$ werden durch $W_k[u, k]$ bzw. $W_k[k, v]$ überschrieben.
- Zeigen: $W_{k-1}[u, k] = 1$ gdw $W_k[u, k] = 1$. (analog für $W_{k-1}[k, v]$)
  - "$\Rightarrow$": Jeder u-k Pfad mit inneren Knoten aus $[k-1]$ ist auch ein u-k Pfad mit inneren Knoten aus $[k]$.
  - "$\Leftarrow$": Sei $p = (u, v_1, \ldots, v_\ell, k)$ ein u-k Pfad. Aufgrund der Pfadeigenschaft sind die inneren Knoten $v_1, \ldots, v_\ell$ aus $[k-1]$.

Laufzeit und Speicherplatz

- Laufzeit Schritt 1: $\mathcal{O}(|V|^2)$, Schritt 2: $\mathcal{O}(|V|^3)$.
- Speicherbedarf für Array W: $\mathcal{O}(|V|^2)$.

Definition Teiler, ggT, kgV, Primzahlen

Seien $a, b \in \mathbb{Z}$.

1. *a teilt b*, geschrieben $a|b$, falls ein $k \in \mathbb{Z}$ existiert mit $ak = b$.
2. Falls a die Zahl b nicht teilt, schreiben wir $a \nmid b$.
3. Der *größte gemeinsame Teiler* von a und b ist
$$\mathrm{ggT}(a, b) = \max_{k \in \mathbb{N}} \{k | a \text{ und } k | b\}.$$
4. Das *kleinste gemeinsame Vielfache* von a und b ist
$$\mathrm{kgV}(a, b) = \min_{k \in \mathbb{N}} \{a | k \text{ und } b | k\}.$$
5. $a > 1$ heißt *Primzahl*, falls die einzigen Teiler $k \in \mathbb{N}$ von a die Zahlen 1 und a sind.
6. a, b heißen *teilerfremd*, falls $\mathrm{ggT}(a, b) = 1$.

Modulare Arithmetik

Definition Modulare Arithmetik

Seien $a, b \in \mathbb{Z}$ und $m \in \mathbb{N}$. Wir nennen a und b *kongruent modulo* m, falls $m|a-b$. Wir schreiben $a \equiv b \bmod m$ oder auch $a = b \bmod m$.

Satz Äquivalenzklassen der modularen Arithmetik

Sei $m \in \mathbb{N}$. $R = \{(a,b) \in \mathbb{Z}^2 \mid a = b \bmod m\}$ ist eine Äquivalenzrelation auf $\mathbb{Z}$.

Beweis:

- **Reflexivität:** $a = a \bmod m$, denn $m|(a-a)$.
- **Symmetrie:** $a = b \bmod m$, d.h. $a - b = km$ für ein $k \in \mathbb{Z}$.
- Damit gilt $b - a = (-k)m$ bzw. $b = a \bmod m$.
- **Transitivität:** Seien $a = b \bmod m$ und $b = c \bmod m$.
- Damit gilt $a - b = k_1 m$ und $b - c = k_2 m$. Es folgt $a - c = (a-b) + (b-c) = (k_1 + k_2)m$ und daher $a = c \bmod m$.

Form der Äquivalenzklassen

Anmerkung:

- Es gilt $a = a \pm m = a \pm 2m = \ldots = a + km \bmod m$ für alle $k \in \mathbb{Z}$.
- Wir schreiben auch $\{x \in \mathbb{Z} \mid x = a + mk, k \in \mathbb{Z}\} = a + m\mathbb{Z}$.
- Es gibt m verschiedene Äquivalenzklassen modulo m:

$$0 + m\mathbb{Z}, 1 + m\mathbb{Z}, \ldots, (m-1) + m\mathbb{Z}.$$

- Sei $a = \lfloor \frac{a}{m} \rfloor m + r$ mit $r \in \mathbb{Z}_m$. Es gilt $r = a \bmod m$.
- Wir repräsentieren $a + m\mathbb{Z}$ durch eindeutiges $r \in \mathbb{Z}_m$.

Anwendung: modularer Arithmetik

- Pseudozufallszahlen mittels Linearem Kongruenzgenerator.
- Beginne mit zufälligem Startwert $x_0 \in \mathbb{Z}_m$.
- Berechne iterativ für festes $a, b \in \mathbb{Z}_m$: $x_i = ax_{i-1} + b \bmod m$.
- $x_1, x_2, x_3, \ldots$ definieren eine Pseudozufallsfolge.

Teilbarkeit durch 3

Satz Teilbarkeit durch 3

Sei $n \in \mathbb{N}$. Dann gilt $3|n$ gwd 3 die Quersumme von n teilt.

Beweis:

- Sei $n_k \ldots n_1 n_0$ die Dezimaldarstellung von n, d.h. $n = \sum_{i=0}^{k} n_i 10^i$.
- Modulo 3 gilt
$$n = \sum_{i=0}^{k} n_i 10^i = \sum_{i=0}^{k} n_i (10 \bmod 3)^i = \sum_{i=0}^{k} n_i \bmod 3.$$
- D.h. $n = 0 \bmod 3$ genau dann wenn $\sum_{i=0}^{k} n_i = 0 \bmod 3$.
- Damit gilt $3|n$ genau dann wenn 3 die Quersumme $\sum_{i=0}^{k} n_i$ teilt.

Teilbarkeit von Linearkombinationen

Lemma Linearkombination

Seien $a, b, d \in \mathbb{N}$. Falls d sowohl a als auch b teilt, dann teilt d ebenfalls $ax + by$ für alle $x, y \in \mathbb{Z}$.

Beweis:

- Es gilt nach Voraussetzung $a = dk_a$ und $b = dk_b$ für $k_a, k_b \in \mathbb{Z}$.
- Daraus folgt $ax + by = dk_a x + dk_b y = d(k_a x + k_b y)$.
- Damit teilt d die ganzzahlige Linearkombination $ax + by$.

Lemma von Bézout

Lemma von Bézout

Seien $a, b \in \mathbb{Z}$. Dann gilt $\mathrm{ggT}(a, b) = \min\{ax + by \in \mathbb{N} \mid x, y \in \mathbb{Z}\}$.

Beweis:

- Sei $S = \{ax + by \in \mathbb{Z} \mid x, y \in \mathbb{Z}\}$ und $s \in S \cap \mathbb{N}$ minimal.
- Wir zeigen zunächst, dass $\mathrm{ggT}(a, b) \leq s$.
- Lemma Linearkombination: $\mathrm{ggT}(a, b) | s$ und damit $\mathrm{ggT}(a, b) \leq s$.
- Bleibt zu zeigen, dass ebenfalls $\mathrm{ggT}(a, b) \geq s$ gilt.
- Sei $q = \lfloor \frac{a}{s} \rfloor$. Dann gilt

$$a \bmod s = a - qs = a - q(ax + by) = a(1 - qx) + b(-qy).$$

- D.h. $a \bmod s \in S$ und $a \bmod s < s$.
- Aufgrund der Minimalität von s muss $a \bmod s = 0$ gelten.
- Damit folgt $s|a$. Analog kann $s|b$ gezeigt werden.
- D.h. s ist ein gemeinsamer Teiler von a, b und $s \leq \mathrm{ggT}(a, b)$.

Korollar für den größten gemeinsamen Teiler

Korollar ggT-Korollar

Seien $a, b \in \mathbb{Z}$. Falls d sowohl a als auch b teilt, dann teilt d ebenfalls $\mathrm{ggT}(a, b)$.

Beweis:

- Lemma Linearkombination: d teilt $ax + by$ für alle $x, y \in \mathbb{Z}$.
- Lemma von Bézout: $\mathrm{ggT}(a, b) = \min\{ax + by \in \mathbb{N} \mid x, y \in \mathbb{Z}\}$.
- Damit teilt d ebenfalls $\mathrm{ggT}(a, b)$.

Teilerfremde Zahlen

Satz zur Teilerfremdheit

Seien $a, b, p \in \mathbb{Z}$. Falls $\mathrm{ggT}(a, p) = 1$ und $\mathrm{ggT}(b, p) = 1$, dann gilt $\mathrm{ggT}(ab, p) = 1$.

- Nach Lemma von Bézout existieren Zahlen $x, y, x', y' \in \mathbb{Z}$ mit
$$ax + py = \mathrm{ggT}(a, p) = 1 \text{ und } bx' + py' = \mathrm{ggT}(b, p) = 1.$$
- Multiplikation der beiden Gleichungen liefert
$$axbx' + axpy' + pybx' + pypy' = ab(xx') + p(axy' + ybx' + ypy') = 1.$$
- Daraus folgt $\mathrm{ggT}(ab, p) = 1$.

ISBN-Code

Anwendung: ISBN-Code

- Format: Ländercode-Verlagsnummer-laufendeNr-Prüfziffer
- ISBN ist 10-stellig mit Ziffern $z_1, \ldots, z_{10}$.
- Prüfziffer $z_{10} = \sum_{i=1}^{9} iz_i \bmod 11$, d.h. $\sum_{i=1}^{10} iz_i = 0 \bmod 11$.

Satz Fehlererkennung des ISBN-Codes

Der ISBN-Code erkennt einen Fehler und einen Zahlendreher.

Beweis:

- **Fehler:** Sei $z'_j = z_j + e_j$ fehlerhaft mit Fehlerterm $e_j \in \mathbb{Z}_{10}$.
- Prüfung liefert $(\sum_{i=1}^{10} iz_i) + je_j \bmod 11$. Benötigen $je_j \neq 0 \bmod 11$.
- Da $\mathrm{ggT}(j, 11) = 1$ und $\mathrm{ggT}(e_j, 11) = 1$ gilt $\mathrm{ggT}(je_j, 11) = 1$.
- D.h. $je_j \neq 0 \bmod 11$.
- **Dreher** $z_j \leftrightarrow z_{j+1}$: Erhalten $(\sum_{i=1}^{10} iz_i) + z_j - z_{j+1} \bmod 11$.
- Für den Fehlerterm gilt $z_j - z_{j+1} \neq 0 \bmod 11$, sofern $z_j \neq z_{j+1}$.

Der Teilersatz

Satz Teilersatz

Seien $a, b \in \mathbb{Z}$ und p prim. Falls $p|ab$, dann gilt $p|a$ oder $p|b$.

Beweis:

- **Annahme:** p teilt weder a noch b.
- Dann gilt $\mathrm{ggT}(a, p) = 1$ und $\mathrm{ggT}(b, p) = 1$, da p prim ist.
- Daraus folgt $\mathrm{ggT}(ab, p) = 1$. (Widerspruch: Nach Voraussetzung gilt $p|ab$ und damit $\mathrm{ggT}(ab, p) = p$.)

Fundamentalsatz der Arithmetik

Satz Fundamentalsatz der Arithmetik

Jedes natürliche Zahl $n > 1$ lässt sich eindeutig als Produkt von Primzahlen darstellen, d.h. $n = \prod_{i=1}^{k} p_i^{e_i}$ für prime p_i.

Beweis:

- Existenz der Darstellung $\prod_{i=1}^{k} p_i^{e_i}$ per Induktion über n (Folie 12).
- Eindeutigkeit: Sei $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k = q_1 \cdot q_2 \cdot \ldots q_r$.
- Teilersatz: $p_1 | \prod_{i=1}^{r} q_i$ und damit teilt p_1 ein q_i.
- Da q_i prim ist, gilt $p_1 = q_i$. Wir teilen beide Darstellungen durch p_1.
- Iterierung des Arguments liefert für jedes p_j ein passendes q_i.
- Damit sind beide Darstellungen identisch.

Satz zur Berechnung des ggTs

Satz ggT-Satz

Sei $a \in \mathbb{N}_0$, $b \in \mathbb{N}$. Dann gilt $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \bmod b)$

- Wir zeigen zunächst $\mathrm{ggT}(a, b) | \mathrm{ggT}(b, a \bmod b)$.
- Sei $d = \mathrm{ggT}(a, b)$, d.h. d teilt sowohl a als auch b.
- Damit teilt d jede ganzzahlige Linearkombination von a und b.
- Es gilt $a \bmod b = a - qb$ mit $q = \lfloor \frac{a}{b} \rfloor$. Damit teilt d auch $a \bmod b$.
- ggT-Korollar: d teilt $\mathrm{ggT}(b, a \bmod b)$.

- Zeigen nun, dass $\mathrm{ggT}(b, a \bmod b) | \mathrm{ggT}(a, b)$.
- Sei $d = \mathrm{ggT}(b, a \bmod b)$, d.h. $d|b$ und $d|a \bmod b$.
- Schreibe $a = qb + (a \bmod b)$. Damit teilt d auch a.
- ggT-Korollar: d teilt $\mathrm{ggT}(a, b)$.

Euklidischer Algorithmus (300 v. Chr.)

Algorithmus EUKLID

EINGABE: $a, b \in \mathbb{N}$

1. If $(b = 0)$ then return a;
2. Else return EUKLID$(b, a \bmod b)$

AUSGABE: $\mathrm{ggT}(a, b)$

Korrektheit:

- Schritt 1: $\mathrm{ggT}(a, 0) = a$. Schritt 2: folgt aus ggT-Satz.

Laufzeit:

- Für die Laufzeitanalyse benötigen wir die Fibonaccizahlen

$$F_0 := 0,\ F_1 := 1,\ F_i := F_{i-1} + F_{i-2} \text{ für } i \geq 2.$$

Laufzeit von EUKLID

Satz Laufzeit von EUKLID

Seien $a, b \in \mathbb{N}$ mit $a > b$. Sei k die Anzahl der Rekursionen von EUKLID(a, b). Dann gilt $a \geq F_{k+2}$ und $b \geq F_{k+1}$.

Beweis: per Induktion über k

- **IV** für $k = 1$: $b \geq 1 = F_2$ und $a > b$. D.h. $a \geq 2 = F_3$.
- **IS** $k - 1 \to k$: EUKLID(a, b) ruft EUKLID$(b, a \bmod b)$ auf.
- EUKLID$(b, a \bmod b)$ benötigt $k - 1$ Rekursionen, d.h. nach IA gilt
$$b \geq F_{k+1} \text{ und } (a \bmod b) \geq F_k.$$
- $b + (a \bmod b) = b + (a - \lfloor \frac{a}{b} \rfloor b) \leq a$, da $\lfloor \frac{a}{b} \rfloor \geq 1$ wegen $a > b$.
- Damit erhalten wir $a \geq b + (a \bmod b) \geq F_{k+1} + F_k = F_{k+2}$.

Logarithmische Laufzeit von EUKLID

Satz Laufzeit von EUKLID

Seien $a, b \in \mathbb{N}$ mit $a \geq F_{k+1} > b$. Dann benötigt EUKLID(a, b) Laufzeit $\mathcal{O}(\log^3 a)$.

Beweis:

- Wegen $b < F_{k+1}$ benötigt EUKLID weniger als k Rekursionen.
- Am Ende von Dima I: Herleitung expliziter Formel für F_k.
- Es gilt $F_k = \Omega(\phi^k)$, wobei $\phi = \frac{1+\sqrt{5}}{2}$ der goldene Schnitt ist.
- Daraus folgt $\phi^k = \mathcal{O}(F_k)$ bzw. $k = \mathcal{O}(\log F_k) = \mathcal{O}(\log a)$.
- D.h. EUKLID(a, b) benötigt weniger als $k = \mathcal{O}(\log a)$ Aufrufe.
- Multiplikation und Division mit Operanden der Bitlänge $\mathcal{O}(\log a)$ benötigen Zeit $\mathcal{O}(\log^2 a)$. D.h. die Gesamtlaufzeit ist $\mathcal{O}(\log^3 a)$.

Anmerkungen:

- Die Operanden in EUKLID werden sukzessive kleiner.
- Eine exakte Analyse liefert eine Schranke von $\mathcal{O}(\log^2 a)$.

Erweiterter Euklidischer Algorithmus

Algorithmus ERWEITERTER EUKLIDISCHER ALG. (EEA)

EINGABE: $a, b \in \mathbb{N}$

1. If $(b = 0)$ then return $(a, 1, 0)$;
2. $(d, x, y) \leftarrow \text{EEA}(b, a \bmod b)$;
3. $(d, x, y) \leftarrow (d, y, x - \lfloor \frac{a}{b} \rfloor y)$;

AUSGABE: $d = \text{ggT}(a, b) = xa + yb$

Korrektheit:

- Schritt 1: Für $b = 0$ gilt $d = a = \text{ggT}(a, 0) = 1a + 0b$.
- Schritt 2: $d = \text{ggT}(a, b) = \text{ggT}(b, a \bmod b)$ folgt aus dem ggT-Satz.
- Schritt 3: Für die Koeffizienten gilt
$$d = \text{ggT}(b, a \bmod b) = xb + y(a - \lfloor \tfrac{a}{b} \rfloor b) = ya + (x - \lfloor \tfrac{a}{b} \rfloor y)b.$$
- D.h. $(x, y) \leftarrow (y, x - \lfloor \frac{a}{b} \rfloor y)$ erhält die Invariante $d = xa + yb$.

Laufzeit:

- $\mathcal{O}(\log^2(\max\{a, b\}))$ analog zu EUKLID(a, b)

Beispiel $\mathrm{ggT}(15, 11)$

Tabellarische Schreibweise:

a	b	$\lfloor \frac{a}{b} \rfloor$	x	y
15	11	1	3	-4
11	4	2	-1	3
4	3	1	1	-1
3	1	3	0	1
1	0	-	1	0

Schreibweise als Gleichungen:

$$
\begin{aligned}
15 - 1 \cdot 11 &= 4 \\
11 - 2 \cdot 4 &= 3 \\
4 - 1 \cdot 3 &= 1 \\
3 - 3 \cdot 1 &= 0
\end{aligned}
\qquad
\begin{aligned}
1 &= 11 + 3(15 - 1 \cdot 11) = 3 \cdot 15 - 4 \cdot 11 \\
1 &= 4 - 1(11 - 2 \cdot 4) = -11 + 3 \cdot 4 \\
1 &= 4 - 1 \cdot 3
\end{aligned}
$$

Abelsche Gruppen

Definition Abelsche Gruppe

Eine abelsche Gruppe ist ein Tupel $(G, \circ)$ bestehend aus einer Menge G und einer Verknüpfung $\circ$ mit den folgenden Eigenschaften

1. Abgeschlossenheit: $\circ : G \times G \to G, (a, b) \mapsto a \circ b$
2. Kommutativität: $a \circ b = b \circ a$ für alle $a, b \in G$
3. Assoziativität: $(a \circ b) \circ c = a \circ (b \circ c)$ für alle $a, b, c \in G$
4. Neutrales Element: $\exists! e \in G : a \circ e = a$ für alle $a \in G$
5. Inverses Element: Schreibweisen
   - multiplikativ: $\forall a \in G \; \exists! a^{-1} \in G$ mit $a \circ a^{-1} = e$
   - additiv: $\forall a \in G \; \exists! (-a) \in G$ mit $a \circ (-a) = e$

Wir schreiben auch abkürzend G statt $(G, \circ)$, wenn $\circ$ bekannt ist.

Falls nicht anders bezeichnet, so schreiben wir G multiplikativ mit der Verknüpfung $\circ = \cdot$.

Beispiele für Gruppen

Bsp: Unendliche Gruppen

- $(\mathbb{Z}, +)$ ist eine additive Gruppe
  1. Addition ganzer Zahlen liefert eine ganze Zahl.
  2. $a + b = b + a$ für alle $a, b \in \mathbb{Z}$
  3. $(a + b) + c = a + (b + c)$ für alle $a, b, c \in \mathbb{Z}$
  4. Neutrales Element ist $0 \in \mathbb{Z}$.
  5. Inverses von $a \in Z$ ist $(-a) \in Z$, denn $a + (-a) = 0$.
- $(\mathbb{Q} \setminus \{0\}, \cdot)$ ist eine multiplikative Gruppe
  1. Produkt von Null verschiedener rationaler Zahlen ist nicht Null.
  2. Kommutativität: $\frac{a}{b} \cdot \frac{c}{d} = \frac{ac}{bd} = \frac{c}{d} \cdot \frac{a}{b}$ für $\frac{a}{b}, \frac{c}{d} \in \mathbb{Q}$.
  3. Assoziativität gilt analog in $\mathbb{Q}$.
  4. Neutrales Element ist $1 \in \mathbb{Q} \setminus \{0\}$.
  5. Inverses von $\frac{a}{b} \in \mathbb{Q} \setminus \{0\}$ ist $\frac{b}{a} \in \mathbb{Q} \setminus \{0\}$.
- $(\mathbb{Z}, \cdot)$ ist keine Gruppe, denn 2 besitzt kein Inverses.
- $(\mathbb{N}, +)$ ist keine Gruppe, denn 1 besitzt kein Inverses.

Wir betrachten nun Gruppen G mit endlicher Kardinalität $|G|$.

Die additive Gruppe $(\mathbb{Z}_m, +)$

Satz $(\mathbb{Z}_m, +)$ ist eine Gruppe

Sei $m \in \mathbb{N}$. Dann ist $\mathbb{Z}_m$ zusammen mit der Addition modulo m

$$+ : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m,\ (a, b) \mapsto a + b \bmod m$$

eine additive Gruppe.

Beweis:

1. Sei $a + b = qm + r$ mit $r \in \mathbb{Z}_m$. Dann gilt $a + b = r$ in $\mathbb{Z}_m$.
2. Folgt aus Kommutativität der Addition in $\mathbb{Z}$.
3. Folgt aus Assoziativität der Addition in $\mathbb{Z}$.
4. Neutrales Element ist $0 \in \mathbb{Z}_m$, denn $a + 0 = a \bmod m$.
5. Inverses von a ist

$$(-a) = \begin{cases} m - a & \text{für } a \neq 0 \\ 0 & \text{sonst.} \end{cases}$$

Definition $\mathbb{Z}_n^*$

Definition Eulersche ϕ-Funktion

Sei $n \in \mathbb{N}$. Wir definieren $\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \mid \mathrm{ggT}(a, n) = 1\}$. Die *Eulersche phi-Funktion* bezeichnet die Kardinalität von $\mathbb{Z}_n^*$, d.h. $\phi(n) = |\mathbb{Z}_n^*|$.

Bsp:

- Sei p prim. Dann gilt $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ und $\phi(p) = p - 1$.
- Sei $n = pq$ mit p, q prim. Dann gilt
$$\mathbb{Z}_n^* = \mathbb{Z}_n \setminus \{0, p, 2p, \ldots, (q-1)p, q, 2q, (p-1)q\}.$$
- Damit folgt $\phi(n) = |\mathbb{Z}_n^*| = n - 1 - (q-1) - (p-1) = (p-1)(q-1)$.

Die multiplikative Gruppe $\mathbb{Z}_n^*$

Satz $(\mathbb{Z}_n^*, \cdot)$ ist eine Gruppe

Sei $n \in \mathbb{N}$. Dann ist $\mathbb{Z}_n^*$ zusammen mit der Multiplikation modulo n

$$\cdot : \mathbb{Z}_n^* \times \mathbb{Z}_n^* \to \mathbb{Z}_n^*,\ (a, b) \mapsto a \cdot b \bmod n$$

eine multiplikative Gruppe.

Beweis:

- Abschluss: Seien $a, b \in \mathbb{Z}_n^*$, d.h. $\mathrm{ggT}(a, n) = \mathrm{ggT}(b, n) = 1$.
- Aus Satz zur Teilerfremdheit folgt damit $\mathrm{ggT}(ab, n) = 1$.
- $\mathrm{ggT}(ab, n) = \mathrm{ggT}(ab \bmod n, n) = 1$, d.h. $(ab \bmod n) \in \mathbb{Z}_n^*$.
- Kommutativität und Assoziativität folgen aus der Kommutativität und Assoziativität der Multiplikation über $\mathbb{Z}$.
- $1 \in \mathbb{Z}_n^*$ ist neutrales Element, denn $1 \cdot a = a \bmod n$ für alle $a \in \mathbb{Z}_n^*$.

Berechnung von Inversen in $\mathbb{Z}_n^*$

Fortsetzung Beweis: Inverse in $\mathbb{Z}_n^*$

- Lemma von Bézout: $\exists x, y \in \mathbb{Z}$ mit $1 = \mathrm{ggT}(a, n) = ax + ny$.
- OBdA gilt $x \in \mathbb{Z}_n$, denn $1 = a(x + kn) + n(y - ak)$ für alle $k \in \mathbb{Z}$.
- Es gilt $ax = 1 - ny = 1 \bmod n$, d.h. $x = a^{-1} \in \mathbb{Z}_n$.
- Ferner gilt $x \in \mathbb{Z}_n^*$, da $xa + ny = 1 = \mathrm{ggT}(x, n)$ nach Lemma von Bézout.

Korollar Berechnung von Inversen

Sei $a \in \mathbb{Z}_n^*$. Dann kann $a^{-1} \in \mathbb{Z}_n^*$ in Zeit $\mathcal{O}(\log^2 n)$ berechnet werden.

- Berechne mit EEA Zahlen $x, y \in \mathbb{Z}$ mit $\mathrm{ggT}(a, n) = ax + ny$.
- Setze $a^{-1} = x \bmod n$.

Eine nicht-abelsche Gruppe

Satz Die symmetrische Gruppe

Sei $\mathcal{G}_n$ die Menge aller Permutation auf $[n]$. Dann ist $(\mathcal{G}_n, \circ)$ mit $\pi \circ \pi' = \pi(\pi')$ als Hintereinanderausführung von Permutationen eine nicht-abelsche Gruppe.

Beweis:

1. Abgeschlossenheit: $\pi(\pi')$ ist eine Permutation.
2. Nicht-kommutativ, d.h. nicht-abelsch:
$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \neq \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$$
3. Assoziativität:
$\pi \circ (\pi' \circ \pi'') = \pi \circ (\pi'(\pi'')) = \pi(\pi'(\pi'')) = \pi(\pi') \circ \pi'' = (\pi \circ \pi') \circ \pi''$.
4. Neutrales Element ist die Identität $\pi_{id} = \begin{pmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \end{pmatrix}$
5. $\pi = \begin{pmatrix} 1 & 2 & \dots & n \\ \pi(1) & \pi(2) & \dots & \pi(n) \end{pmatrix}, \pi^{-1} = \begin{pmatrix} \pi(1) & \pi(2) & \dots & \pi(n) \\ 1 & 2 & \dots & n \end{pmatrix}$.

Lösbarkeit von linearen Gleichungen

Satz Eindeutige Lösung

Sei $n \in \mathbb{N}$ und $a \in \mathbb{Z}$ mit $\mathrm{ggT}(a, n) = 1$. Dann besitzt die Gleichung $az = b$ für jedes $b \in \mathbb{Z}_n$ genau eine Lösung $z \in \mathbb{Z}_n$.

Beweis:

- **Existenz:** EEA liefert $x, y \in \mathbb{Z}$ mit $ax + ny = 1$.
- Dann gilt $a^{-1} = x \bmod n$ und damit $z = a^{-1}b \bmod n$.
- **Eindeutigkeit:** Seien z und z' Lösungen.
- Dann gilt $az - b = az' - b \bmod n$ bzw. $a(z - z') = 0 \bmod n$.
- Wegen $\mathrm{ggT}(a, n) = 1$ folgt $n|(z - z')$ und damit $z = z' \bmod n$.

Ordnung einer Gruppe

Definition Ordnung einer Gruppe

Sei G eine (multiplikative) endliche Gruppe mit neutralem Element 1.

1. Die *Ordnung* von G ist $\mathrm{ord}(G) := |G|$.
2. Die *Ordnung* eines Elements $a \in G$ ist
$$\mathrm{ord}_G(a) := \min\{i \in \mathbb{N} \mid a^i = 1\}.$$
3. $H \subseteq G$ heißt *Untergruppe* von G, falls H eine Gruppe ist.
4. Wir bezeichnen mit $\langle a \rangle := \{a, a^2, a^3, \ldots, a^{\mathrm{ord}_G(a)}\}$ die *von a erzeugte Untergruppe*.
5. Die von einem Element a erzeugten Gruppen heißen *zyklisch*. Das Element a heißt *Generator* oder auch *primitives Element*.

Beispiel für Ordnungen und Untergruppen

Beispiel: $G = \mathbb{Z}_7^*$

- Die Ordnung von $\mathbb{Z}_7^*$ ist $\mathrm{ord}(\mathbb{Z}_7^*) = |\{1, 2, 3, 4, 5, 6\}| = 6$.
- $\mathrm{ord}_G(4) = 3$, denn $4^1 = 4, 4^2 = 2$ und $4^3 = 1$.
- $H = \{1, 2, 4\}$ ist eine Untergruppe.
- In H gilt $2 \cdot 4 = 1$, d.h. $2^{-1} = 4$ und $4^{-1} = 2$.
- $H = \langle 4 \rangle$ ist eine zyklische Untergruppe der Ordnung 3 mit Generatoren 2 und 4.
- $\mathbb{Z}_7^* = \langle 3 \rangle$ ist ebenfalls zyklisch.
- $\langle 6 \rangle = \{1, 6\}$ ist eine Untergruppe der Ordnung 2.
- $\langle 1 \rangle = \{1\}$ ist eine Untergupppe der Ordnung 1.

Satz von Euler

Satz von Euler

Sei G eine (multiplikative) Gruppe mit neutralem Element 1. Dann gilt für alle $a \in G$ die Identität $a^{|G|} = 1$.

Beweis:

- Sei $G = \{g_1, \ldots, g_n\}$, d.h. $n = \mathrm{ord}(G) = |G|$.
- Wir betrachten die Abbildung $f : G \to G$, $g \mapsto ag$.
- f ist bijektiv mit Umkehrabbildung $f^{-1} : G \to G$, $g \mapsto a^{-1}g$.
- Da f bijektiv ist, gilt $G = f(G)$ bzw. $\{g_1, \ldots, g_n\} = \{ag_1, \ldots, ag_n\}$.
- Daraus folgt $\prod_{i=1}^{n} g_i = \prod_{i=1}^{n} ag_i = a^n \prod_{i=1}^{n} g_i$.
- Damit gilt $a^n = a^{|G|} = 1$.

Elementordnung

Satz Elementordnung

Sei G eine endliche (muliplikative) Gruppe. Dann gilt für alle $a \in G$ die Ungleichung $\mathrm{ord}_G(a) \leq \mathrm{ord}(G)$.

Beweis:

- **Annahme:** Sei a ein Element mit $k = \mathrm{ord}_G(a) \geq |G| + 1$.
- Wir betrachten die Abbildung $f : [k] \to G$, $i \mapsto a^i$.
- Nach Schubfachprinzip muss es $i < j$ mit $i, j \in [k]$ geben mit $f(i) = f(j)$, d.h. $a^i = a^j$.
- Multiplikation mit $(a^i)^{-1}$ liefert $a^{j-i} = 1$ mit $0 < j - i < k$. (Widersprich: $\mathrm{ord}_G(a) = k$ nach Voraussetzung)

Elementordnung teilt Gruppenordnung

Satz Elementordnung teilt Gruppenordnung

Sei G eine endliche (multiplikative) Gruppe. Dann gilt für alle $a \in G$, dass $\mathrm{ord}_G(a) | \mathrm{ord}(G)$.

Beweis:

- **Annahme:** $ord_G(a)$ teilt $\mathrm{ord}(G)$ nicht
- Division mit Rest liefert $\mathrm{ord}(G) = q \cdot \mathrm{ord}_G(a) + r$ mit
$$0 < r < \mathrm{ord}_G(a).$$
- Es gilt $a^r = a^{\mathrm{ord}(G) - q \cdot \mathrm{ord}_G(a)} = (a^{\mathrm{ord}(G)})(a^{\mathrm{ord}_G(a)})^{-q} = 1 \cdot (1^q)^{-1} = 1.$
(Widerspruch zur Minimalität von $\mathrm{ord}_G(a)$)

Nebenklassen von Untergruppen

Definition Nebenklasse

Sei $(G, \cdot)$ eine abelsche Gruppe und $H \subseteq G$ eine Untergruppe von G. Für jedes $b \in G$ heißt $b \cdot H = \{b \cdot h \mid h \in H\}$ *Nebenklasse* von H.

Bsp:

- Betrachte $G = (\mathbb{Z}_8, +)$. $H = \{0, 4\}$ ist eine Untergruppe von G.
- Nebenklassen sind

$$1 + H = \{1, 5\}, 2 + H = \{2, 6\}, 3 + H = \{3, 7\}, 4 + H = H.$$

- Betrachte $G = (\mathbb{Z}_7^*, \cdot)$ mit Untergruppe $H = \{1, 2, 4\}$.
- Nebenklassen sind

$$2H = \{2, 4, 1\} = H = 4H, 3H = \{3, 6, 5\} = 6H = 5H.$$

Eigenschaften von Nebenklassen

Satz Eigenschaften von Nebenklassen

Sei $(G, \cdot)$ eine abelsche Gruppe und $H \subseteq G$ eine Untergruppe.

1. $hH = H$ für alle $h \in H$.
2. Für $a, b \in G$ gilt entweder $aH = bH$ oder $aH \cap bH = \emptyset$.
3. $|aH| = |H|$ für alle $a \in G$.
4. Die Nebenklassen aH für $a \in G$ partitionieren G.

Beweis:

- **ad 1**: Die Abgeschlossenheit von H liefert $hH \subseteq H$.
- Bleibt zu zeigen, dass $H \subseteq hH$. Sei $g \in H$ beliebig.
- Dann gilt $g = 1g = hh^{-1}g = h(h^{-1}g) \in hH$.
- **ad 2**: Sei $a, b \in G$ mit $aH \cap bH \neq \emptyset$.
- Dann gibt es $h_1, h_2 \in H$ mit $ah_1 = bh_2$.
- Damit gilt $aH = (bh_1^{-1}h_2)H = b(h_1^{-1}h_2H) = bH$.

Eigenschaften von Nebenklassen

Beweis Fortsetzung:

- **ad 3**: Sei $a \in G$. Betrachten die Abbildung $f : H \to aH, h \mapsto ah$.
- f ist eine Bijektion in G, d.h. $|H| = |aH|$ nach Gleichheitsregel.
- **ad 4**: $1 \in H$, da H eine Gruppe ist.
- Daher gilt $a \in aH$ für alle $a \in G$. D.h. $G \subseteq \bigcup_{a \in G} aH$.
- Aufgrund der Abgeschlossenheit von G gilt auch $\bigcup_{a \in G} aH \subseteq G$.
- Aufgrund von 2. bilden die Nebenklassen aH eine Partition von H.

Index einer Untergruppe

Definition Index einer Untergruppe

Sei G eine abelsche Gruppe und $H \subseteq G$ eine Untergruppe. Wir bezeichnen mit G/H die Menge der Nebenklassen von G. Die Kardinalität von G/H heißt *Index von H* in G, d.h. $\mathrm{ind}_G(H) = |G/H|$. Alternativ verwendet man für den Index auch die Notation $[G : H]$.

Bsp:

- Sei $G = (\mathbb{Z}_8, +)$ und $H = \{0, 4\}$.
- Dann gilt $G/H = \{H, 1 + H, 2 + H, 3 + H\}$, d.h. $\mathrm{ind}_G(H) = 4$.
- Sei $G = (\mathbb{Z}_7^*, \cdot)$ und $H = \{1, 2, 4\}$.
- Dann gilt $G/H = \{H, 3H\}$, d.h. $\mathrm{ind}_G(H) = 2$.

Untergruppenordnung teilt Gruppenordnung

Satz von Lagrange

Sei G eine abelsche Gruppe und $H \subseteq G$ eine Untergruppe. Dann gilt $|G| = |H| \cdot \mathrm{ind}_G(H)$, d.h. insbesondere $\mathrm{ord}_G(H)$ teilt $\mathrm{ord}(G)$.

Beweis:

- Alle Nebenklassen von H besitzen dieselbe Kardinalität $|H|$.
- Es gibt $\mathrm{ind}_G(H)$ viele verschiedene Nebenklassen von H in G.
- Alle Nebenklassen bilden eine Partition von G.

Die Faktorgruppe G/H

Satz Faktorgruppe G/H

Sei G eine abelsche Gruppe mit Untergruppe $H \subseteq G$. Dann ist G/H zusammen mit der Multiplikation $\cdot : G/H \times G/H \to G/H$, $(aH, bH) \mapsto abH$ eine Gruppe, die sogenannte *Faktorgruppe*.

Beweis:

- Wir zeigen zunächst die Repräsentanten-Unabhängigkeit der Multiplikation, d.h. die Multiplikation ist wohldefiniert.
- Seien $aH = a'H$, $bH = b'H$.
- Es gilt $a \in aH = a'H$ und $b \in bH = bH'$.
- Damit gibt es $h_1, h_2 \in H$, so dass $a = a'h_1$ und $b = b'h_2$.
- Es folgt $abH = (a'h_1b'h_2)H = a'b'(h_1h_2H) = a'b'H$.
- Damit ist die Multiplikation unabhängig von den Repräsentanten.
- **Neutrales** Element von G/H ist H, denn $aH \cdot H = aH$.
- **Inverses** Element von aH in G/H ist $a^{-1}H$, wobei a^{-1} das inverse Element von a in G ist. Es gilt $a^{-1}H \cdot aH = a^{-1}aH = H$.

Beispiele für Faktorgruppen

Bsp:

- $G = (\mathbb{Z}, +)$ mit Untergruppe $H = 3\mathbb{Z}$.
- Nebenklassen sind $H = 3\mathbb{Z}, 1 + H = 1 + 3\mathbb{Z}$ und $2 + H = 2 + 3\mathbb{Z}$.
- Repräsentanten-Unabhängigkeit: $H = 6 + H$ und $2 + H = 5 + H$.
- Dann gilt $H + 2 + H = 2 + H = (6 - 6 + 5 - 3) + H$
  $= 6 + 5 + (-6 - 3 + H) = 6 + 5 + H$.
- Neutrales Element ist H, denn $H + a + H = a + H$.
- Inverses Element zu $a + H \neq H$ ist $(-a) + H$, denn
  $a + H + (-a) + H = 0 + H = H$.
- **Man beachte:** Unsere Gruppe $(\mathbb{Z}_m, +)$ ist lediglich eine alternative Notation für die Faktorgruppe $(\mathbb{Z}/m\mathbb{Z}, +)$.

- $G = \mathbb{Z}_7^*$ mit Untergruppe $H = \{1, 2, 4\} = 2H = 4H$
- Nebenklassen sind H und $H_2 = \{3, 5, 6\} = 3H = 5H = 6H$.
- H ist neutrales Element, denn $H \cdot H = H$ und $H \cdot H_2 = H_2$.
- $(H_2)^{-1} = (3^{-1}H) = 5H = H_2$, denn $H_2 \cdot H_2 = H$.

Isomorphismus

Definition Gruppen-Isomorphismus

Seien $(G, +)$ und $(G', \cdot)$ Gruppen. Die Abbildung $f : G \to G'$ heißt *Gruppen-Isomorphismus*, falls gilt

1. f ist bijektiv
2. $f(u + v) = f(u) \cdot f(v)$ für alle $u, v \in G$, die sogenannte *Homomorphismus*-Eigenschaft von f.

Wir nennen G und G' *isomorph*, falls ein solcher Gruppen-Isomorphismus existiert. Notation: $G \cong G'$.

Zyklische Gruppen und $(\mathbb{Z}_m, +)$

Satz Isomorphie zu $\mathbb{Z}_m$

Sei G eine zyklische Gruppe mit Ordnung m. Dann gilt $G \cong (\mathbb{Z}_m, +)$.

Beweis:

- Da G zyklisch ist, gibt es ein $a \in G$ mit $G = \{a^i \mid i \in \mathbb{Z}_m\}$.
- Wir betrachten die Abbildung $f : \mathbb{Z}_m \to G, i \mapsto a^i$.
- Wegen $G = \{a^i \mid i \in \mathbb{Z}_m\}$ ist f surjektiv.
- Da $|G| = |\mathbb{Z}_m|$, ist f ebenfalls bijektiv.
- f ist ein Homomorphismus, da für alle $i, j \in \mathbb{Z}_m$ gilt
$$f(i + j) = a^{i+j} = a^i \cdot a^j = f(i) \cdot f(j).$$
- Damit ist f ein Gruppen-Isomorphismus und $G \cong (\mathbb{Z}_m, +)$.

Diskreter Logarithmus Problem

DLP: Diskreter Logarithmus Problem

Gegeben: G mit Generator $a \in G$, Element $b \in G$
Gesucht: $i \in \mathbb{Z}_{|G|}$ mit $a^i = b$

Lösung des DLP:

- Betrachten den Gruppen-Isomorphismus $f : i \mapsto a^i$.
- Für die Umkehrfunktion $f^{-1} : a^i \mapsto i$ gilt $f(b) = i$.
- D.h. f^{-1} löst das Diskrete Logarithmus Problem.
- Da DLP in einigen Gruppen als schweres Problem gilt, kann f nicht immer in beiden Richtungen effizient berechenbar sein.

Komplexität modularer Arithmetik

Satz Modulare Addition

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a + b \bmod m$ in Zeit $\mathcal{O}(n)$ berechnet werden.

Beweis: Schulmethode

- Schreibe alle Operanden in Binärform $a = a_{n-1} \dots a_0 = \sum_{i=0}^{n} a_i 2^i$.
- Addiere a, b bitweise mit Übertrag, beginnend bei a_0, b_0.
- Falls $a + b > m$, subtrahiere bitweise m vom Ergebnis.
- **Laufzeitkomplexität** dieser Methode: $\mathcal{O}(n) = \mathcal{O}(\log m)$.
- D.h. Addition/Subtraktion besitzen Laufzeit linear in der Größe der Operanden.

Komplexität modularer Multiplikation

Satz Modulare Multiplikation

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a \cdot b \bmod m$ in Zeit $\mathcal{O}(n^2)$ berechnet werden.

Beweis:

Algorithmus MULTIPLIKATION SCHULMETHODE

EINGABE: $a = a_{n-1} \ldots a_{n-1}, b, m$

1. $c \leftarrow 0$; $h \leftarrow b$;
2. For $i \leftarrow 0$ to $n-1$
   1. If $(a_i = 1)$ then $c \leftarrow c + h \bmod m$
   2. $h \leftarrow 2h \bmod m$

AUSGABE: $a + b \bmod m$

- **Korrektheit:** $a \cdot b = \left(\sum_{i=0}^{n} a_i 2^i\right) \cdot b = \sum_{i=0}^{n} a_i \cdot \left(b \cdot 2^i\right) \bmod m$.
- In Schritt 2.1 wird $h = b \cdot 2^i \bmod m$ addiert.
- **Laufzeit:** $n \cdot \mathcal{O}(n) = \mathcal{O}(n^2) = \mathcal{O}(\log^2 m)$

Komplexität modularer Division

Korollar Modulare Division

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a \in \mathbb{Z}_m$, $b \in \mathbb{Z}_m^*$. Dann kann $\frac{a}{b} \bmod m$ in Zeit $\mathcal{O}(n^2)$ berechnet werden.

Beweis:

- Berechne $b^{-1} \bmod m$ mit EEA in Zeit $\mathcal{O}(\log^2 m)$.
- Berechne $a \cdot b^{-1} \bmod m$ in Zeit $\mathcal{O}(\log^2 m)$.

Die Karatsuba Methode

Satz von Karatsuba

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a \cdot b \bmod m$ in Zeit $\mathcal{O}(n^{\log_2 3})$ berechnet werden.

Beweis:

- Vereinfachende Annahme: Bitlänge ist Zweierpotenz $n = 2^k$.
- Teilen Operanden in der Mitte, d.h. $a = A_1 2^{\frac{n}{2}} + A_0$ mit $A_1 = \sum_{i=\frac{n}{2}}^{n-1} a_i 2^i$ und $A_0 = \sum_{i=0}^{\frac{n}{2}-1} a_i 2^i$. Dann gilt

$$\begin{aligned} a \cdot b &= \left(A_1 2^{\frac{n}{2}} + A_0\right)\left(B_1 2^{\frac{n}{2}} + B_0\right) \\ &= A_1 B_1 2^n + (A_1 B_0 + A_0 B_1) 2^{\frac{n}{2}} + A_0 B_0 \\ &= A_1 B_1 2^n + ((A_0 + A_1)(B_0 + B_1) - A_0 B_0 - A_1 B_1) 2^{\frac{n}{2}} + A_0 B_0 \end{aligned}$$

- D.h. die Multiplikation von n-Bit Zahlen a, b kann zurückgeführt werden auf 3 Multiplikationen von $\frac{n}{2}$-Bit Zahlen.
- Rekursiv: $\frac{n}{2}$-Bit mittels 3 Multiplikationen mit $\frac{n}{4}$-Bit Zahlen, usw.

Laufzeit Karatsuba

Beweis: Laufzeit der Karatsuba-Methode

- Rekursiver Aufruf erfordert Aufwand von 6 Additionen und 2 Shifts.
- Sei $T(n)$ die Laufzeit zum Multiplizieren zweier n-Bit Zahlen.
- Dann gilt $T(n) = 3T(\frac{n}{2}) + c \cdot n$ für festes $c > 0$.

$$
\begin{aligned}
T(n) &= 3\left(3T\left(\frac{n}{4}\right) + c\frac{n}{2}\right) + cn = 3^2 T\left(\frac{n}{4}\right) + cn\left(1 + \frac{3}{2}\right) \\
&= 3^2\left(3T\left(\frac{n}{8}\right) + c\frac{n}{4}\right) + cn\left(1 + \frac{3}{2}\right) \\
&= 3^3 T\left(\frac{n}{8}\right) + cn\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2\right) \\
&= \vdots \\
&= 3^i T\left(\frac{n}{2^i}\right) + cn\sum_{j=0}^{i-1}\left(\frac{3}{2}\right)^j
\end{aligned}
$$

- Abbruch der Rekursion für $T(1)$, d.h. $n = 2^i$ bzw. $i = \log_2 n$.

Laufzeit Karatsuba

Beweis: Fortsetzung Laufzeit Karatsuba-Methode

- Wir erhalten für $i = \log_2 n$ und $T(1) = \mathcal{O}(1)$

$$\begin{aligned} T(n) &= 3^{\log_2 n} \cdot T(1) + cn \sum_{j=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^j \\ &= n^{\log_2 3} \cdot \mathcal{O}(1) + cn \cdot \frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\frac{3}{2} - 1} \\ &= \mathcal{O}(n^{\log_2 3}) + \mathcal{O}\left(n \cdot \frac{3^{\log_2 n}}{2^{\log_2 n}}\right) \\ &= \mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.59}) \end{aligned}$$

Anmerkung:

- Wir lernen bald ein Verfahren kennen mit Komplexität $\mathcal{O}(n \log n \log\log n) = \mathcal{O}(n^{1+\epsilon})$ für jedes $\epsilon > 0$.

Modulare Exponentiation

Satz Modulare Exponentiation

Sei $m \in \mathbb{N}$ mit Bitlänge n und $a, b \in \mathbb{Z}_m$. Dann kann $a^b \bmod m$ in Zeit $\mathcal{O}(n^3)$ berechnet werden.

Beweis:

Algorithmus SQUARE & MULTIPLY

EINGABE: a, $b = b_{n-1} \dots b_0$, m

1. $c \leftarrow 1$
2. For $i \leftarrow 0$ to $n-1$
   1. If $(b_i = 1)$ then $c \leftarrow c \cdot a \bmod m$
   2. $a \leftarrow a^2 \bmod m$

AUSGABE: $c = a^b \bmod m$

- **Korrektheit:** $a^b = a^{\sum_{i=0}^{n-1} b_i 2^i} = \prod_{i=0}^{n-1} a_i^{b_i 2^i} = \prod_{i=0}^{n-1} \left(a_i^{2^i}\right)^{b_i}$
- **Laufzeit:** $n \cdot \mathcal{O}(n^2) = \mathcal{O}(n^3) = \mathcal{O}(\log^3 m)$

Kleiner Satz von Fermat

Satz von Fermat

Sei p prim. Dann gilt

1. $a^{p-1} = 1 \bmod p$ für alle $a \in \mathbb{Z}_p^*$.
2. $a^p = a \bmod p$ für alle $a \in \mathbb{Z}_p$.

Beweis:

- ad 1: Nach Satz von Euler gilt $a^{|G|} = a^{|\mathbb{Z}_p^*|} = a^{p-1} = 1$.
- ad 2: Für alle $a \in \mathbb{Z}_p^*$ folgt damit $a^p = a \bmod p$.
- Weiterhin gilt für $a = 0$ die Identität $0^p = 0 \bmod p$.

Primzahltest

- Wir wollen testen, ob eine gegebene Zahl n eine Primzahl ist.
- Effizienter Algorithmus zum Faktorisieren ist unbekannt.
- Kontraposition des Kleinen Satzes von Fermat liefert:
  Falls $a^{n-1} \neq 1 \bmod n$ für ein $a \in \mathbb{Z}_n^*$, dann ist n nicht prim.
- Leider muss ein solcher Zeuge $a \in \mathbb{Z}_n^*$ nicht immer existieren.

Definition Carmichael-Zahl

Sei $n \in \mathbb{N}$ zusammengesetzt. Wir bezeichnen n als *Carmichael-Zahl* falls $a^{n-1} = a \bmod n$ für alle $a \in \mathbb{Z}_n^*$.

Bsp:

- Die kleinsten Carmichael-Zahlen sind 561, 1105, 1729, 2465.
- Es gibt unendlich viele Carmichael-Zahlen (Beweis 1994).

Definition Fermatsche Pseudoprimzahl-Basis

Sei $n \in \mathbb{N}$ zusammengesetzt. Wir bezeichnen $a \in \mathbb{Z}_N^*$ als *Pseudoprimzahl-Basis* für n, falls $a^{n-1} = 1 \bmod n$.

Dichte der Pseudoprimzahl-Basen

Satz Pseudoprimzahlen-Basen bilden Untergruppe

Sei $n \in \mathbb{N}$ zusammengesetzt und keine Carmichael-Zahl. Dann ist höchstens die Hälfte aller $a \in \mathbb{Z}_N^*$ eine Pseudoprimzahl-Basis für n.

Beweis:

- Pseudoprimzahl-Basen $P = \{a \in \mathbb{Z}_n^* \mid a^{n-1} = 1 \bmod n\}$.
- Wir zeigen nun, dass P eine echte Untergruppe von $\mathbb{Z}_n^*$ ist.
- *Abgeschlossenheit:* Seien $a, b \in P$. Dann ist $(ab)^{n-1} = a^{n-1}b^{n-1} = 1 \bmod n$, d.h. $ab \in P$.
- *Neutrales Element:* $1 \in P$, denn $1^{n-1} = 1 \bmod n$.
- *Inverses Element* zu $a \in P$ ist $a^{n-2} \bmod n$, denn
  - $a \cdot a^{n-2} = a^{n-1} = 1 \bmod n$.
  - $a^{n-2} \in P$, da $(a^{n-2})^{n-1} = (a^{n-1})^{n-2} = 1 \bmod n$.
- Da n keine Carmichael-Zahl ist, gilt $P \neq \mathbb{Z}_n^*$.
- Nach Satz von Lagrange gilt $|\mathbb{Z}_n^*| = |P| \cdot \mathrm{ind}_{\mathbb{Z}_n^*}(P)$.
- D.h. $\mathrm{ind}_{\mathbb{Z}_n^*}(P) \geq 2$ und damit $|P| \leq \frac{1}{2}|\mathbb{Z}_n^*|$.

Primzahltest für Nicht-Carmichael-Zahlen

Satz Fermattest

Sei $n \in \mathbb{N}$ keine Carmichael-Zahl und $k \in \mathbb{N}$. Dann kann in Zeit $\mathcal{O}(k \log^3 n)$ mit Fehlerwahrscheinlichkeit $\approx 2^{-k}$ entschieden werden, ob n prim ist.

Beweis:

Algorithmus FERMATTEST

EINGABE: $n, k \in \mathbb{N}$

1. For $i \leftarrow 1$ to k
   1. Wähle $a \in_R \mathbb{Z}_n \setminus \{0\}$.
   2. Falls $\mathrm{ggT}(a, n) > 1$, Ausgabe "n zusammengesetzt".
   3. Falls $a^{n-1} \neq 1 \mod n$, Ausgabe "n zusammengesetzt".
2. Ausgabe "n prim".

- **Laufzeit:** $\mathcal{O}(k \log^3 n) = \mathcal{O}(\log^3 n)$ für konstantes k.
- In der Praxis genügt die Wahl $k = 80$.

Korrektheit von FERMATTEST

Beweis: Fehlerwahrscheinlichkeit

- Falls n prim ist, dann ist die Ausgabe stets korrekt.
- Schritt 2.2: Falls $a \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$, dann ist Teiler von n gefunden.
- Schritt 2.3: Falls $a^{n-1} \neq 1 \bmod n$, dann ist a ein Zeuge für die Zusammengesetztheit von n.
- Für zusammengesetzte n ist fehlerhafte Ausgabe "prim" möglich.
- Pro Iteration:

$$\text{Ws}(\text{Ausgabe "n zusammengesetzt"} \mid \text{n zusammengesetzt}) \geq \tfrac{1}{2}.$$

- Nach k Iterationen:

$$\begin{aligned} \epsilon &:= \text{Ws}(\text{Ausgabe "n prim"} \mid \text{n zusammengesetzt}) \\ &= (1 - \text{Ws}(\text{Ausgabe "n zusammengesetzt"} \mid \text{n zusammengesetzt}))^k \\ &\leq 2^{-k} \end{aligned}$$

- Fehlerwahrscheinlichkeit

$$\text{Ws}(\text{n zusammengesetzt} \mid \text{Ausgabe "n prim"}) \approx \epsilon.$$

Spezieller Chinesischer Restsatz

Satz Spezieller Chinesischer Restsatz

Seien $m, n \in \mathbb{N}$ teilerfremd und $a, b \in \mathbb{Z}$. Dann existiert genau eine Lösung $x \in \mathbb{Z}_{mn}$ des Gleichungssystems

$$\left| \begin{array}{rcl} x & = & a \bmod m \\ x & = & b \bmod n \end{array} \right| .$$

Beweis:

- **Existenz:** EEA liefert $r, s \in \mathbb{Z}$ mir $mr + ns = \mathrm{ggT}(m, n) = 1$.
- D.h. $mr = 1 \bmod n$ und $ns = 1 \bmod m$.
- Wir definieren $x = ans + bmr \bmod mn$.
- Damit gilt $x = a \bmod m$ und $x = b \bmod n$.

- **Eindeutigkeit:** Seien x, x' Lösungen des Gleichungssystems.
- Dann gilt $x = a = x' \bmod m$ und $x = b = x' \bmod n$.
- Damit wird die Differenz $x - x'$ sowohl von m als auch n geteilt.
- Da $\mathrm{ggT}(m, n) = 1$, folgt $mn \mid x - x'$ und damit $x = x' \bmod mn$.

Chinesischer Restsatz

Satz Chinese Remainder Theorem (CRT)

Seien $m_1, \ldots, m_n \in \mathbb{N}$ paarweise teilerfremd und $a_1, \ldots, a_n \in \mathbb{Z}$. Dann existiert genau eine Lösung $x \in \mathbb{Z}_{m_1 \cdot \ldots \cdot m_n}$ des Gleichungssystems

$$\left| \begin{array}{rcl} x & = & a_1 \bmod m_1 \\ x & = & a_2 \bmod m_2 \\ & \vdots & \\ x & = & a_n \bmod m_n \end{array} \right| .$$

Beweis: Induktion über n

- **IV** für $n = 2$ liefert der Spezielle Chinesische Restsatz.
- **IS** für $n - 1 \to n$. Nach IA existiert für die ersten $n - 1$ Gleichungen eine eindeutige Lösung $y \in \mathbb{Z}_{m_1 \cdot \ldots \cdot m_n}$.
- Spezieller CRT-Satz liefert eindeutiges $x \in \mathbb{Z}_{m_1 \cdot \ldots \cdot m_n}$ mit

$$\left| \begin{array}{rcl} x & = & y \bmod m_1 \cdot \ldots \cdot m_{n-1} \\ x & = & a_n \bmod m_n \end{array} \right| .$$

Additiver CRT-Isomorphismus

Satz Additiver CRT-Isomorphismus

Sei $N = m_1 \cdot \ldots \cdot m_n$ für paarweise teilerfremde $m_1, \ldots, m_n$. Dann gilt

$$\mathbb{Z}_N \cong \mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n}.$$

Beweis:

- Wir definieren $f : \mathbb{Z}_N \to \mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n}$ vermöge
$$x \bmod N \mapsto (x \bmod m_1, \ldots, x \bmod m_n).$$
- CRT-Satz liefert zu jedem $f(x)$ das eindeutige x, d.h. f ist injektiv.
- Da $|\mathbb{Z}_N| = N = m_1 \cdot \ldots \cdot m_n = |\mathbb{Z}_{m_1}| \cdot \ldots \cdot |\mathbb{Z}_{m_n}|$, ist f bijektiv.
- Ferner ist f ein Homomorphismus, denn

$$\begin{aligned} f(x+y) &= ((x+y \bmod N) \bmod m_1, \ldots, (x+y \bmod N) \bmod m_n) \\ &= (x+y \bmod m_1, \ldots, x+y \bmod m_n) = f(x) + f(y). \end{aligned}$$

- Damit ist f ein Isomorphismus.
- Man beachte: Sowohl f als auch f^{-1} sind effizient berechenbar.

Anwendung des CRT-Isomorphismus

Korollar Anwendung des CRT-Isomorphismus

Sei $N = m_1 \cdot \ldots \cdot m_n$ für paarweise teilerfremde $m_1, \ldots, m_n$. Dann gilt für alle $x, a \in \mathbb{Z}$, dass $x = a \bmod N$ gdw

$$\left| \begin{array}{rcl} x & = & a \bmod m_1 \\ x & = & a \bmod m_2 \\ & \vdots & \\ x & = & a \bmod m_n \end{array} \right| .$$

Anwendung:

- Seien die m_i, $i \in [n]$ Primzahlen oder Primzahlpotenzen.
- Für viele Probleme sind effiziente Algorithmen in $\mathbb{Z}_N$ nicht bekannt, wohl aber in $\mathbb{Z}_{m_i}$. Vorgehensweise zum Lösen in $\mathbb{Z}_N$:
  - Berechne mittels Isomorphismus f Darstellung in $\mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n}$.
  - Löse die algorithmischen Probleme für alle $\mathbb{Z}_{m_i}$, $i \in [n]$ separat.
  - Berechne mittels Isomorphismus f^{-1} die Lösungen in $\mathbb{Z}_N$.

Multiplikativer CRT-Isomorphismus

Satz Multiplikativer CRT-Isomorphismus

Sei $N = m_1 \cdot \ldots \cdot m_n$ für paarweise teilerfremde $m_1, \ldots, m_n$. Dann gilt

$$\mathbb{Z}_N^* \cong \mathbb{Z}_{m_1}^* \times \ldots \times \mathbb{Z}_{m_n}^*.$$

Beweis:

- Definieren denselben Isomorphismus f wie zuvor.
- Zeigen zunächst, dass für $a \in \mathbb{Z}_N^*$ gilt $f(a) \in \mathbb{Z}_{m_1}^* \times \ldots \times \mathbb{Z}_{m_n}^*$.
- Es gilt $\mathrm{ggT}(a, N) = 1$, d.h. es gibt $x, y \in \mathbb{Z}$ mit $ax + Ny = 1$.
- Damit ist $ax + m_1 \cdot \ldots \cdot m_n y = 1$ bzw. $\mathrm{ggT}(a, m_i) = 1$ für $i \in [n]$.
- Sei umgekehrt $\mathrm{ggT}(a, m_i) = 1$ für alle $i \in [n]$.
- Aus dem Satz zur Teilerfremdheit folgt damit $\mathrm{ggT}(a, N) = 1$.
- f ist nach CRT-Satz injektiv und für teilerfremde m_i gilt
  $|\mathbb{Z}_N^*| = \phi(N) = \phi(m_1 \cdot \ldots \cdot m_n) = \phi(m_1) \cdot \ldots \phi(m_n) = |\mathbb{Z}_{m_1}^*| \cdot \ldots \cdot |\mathbb{Z}_{m_n}^*|$.
  (Übungsaufgabe)
- Dass f ein Homomorphismus ist, folgt analog zum vorigen Beweis.

Anzahl Nullstellen modularer Gleichungen

Satz Nullstellen einer Quadratischen Gleichung

Sei $N \in \mathbb{N}$ ungerade mit Primfaktorzerlegung $N = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$. Dann existieren 2^k Lösungen der Gleichung $x^2 = 1 \bmod N$ in $\mathbb{Z}_N^*$.

Beweis:

- CRT-Isomorphismus: $x^2 = 1 \bmod N$ gdw $x^2 = 1 \bmod p_i^{e_i}$, $i \in [n]$.
- Für jede Gleichung $x^2 = 1 \bmod p_i^{e_i}$ sind $x = \pm 1$ Lösungen.
- $1 \neq (-1) \bmod p_i^{e_i}$, da $p_i^{e_i} > 2$. D.h. die Lösungen sind verschieden.
- Damit sind alle möglichen Vektoren $v \in \{-1, 1\}^k$ Lösungen in $\mathbb{Z}^*_{p_1^{e_1}} \times \ldots \times \mathbb{Z}^*_{p_k^{e_k}}$.
- CRT-Satz: Diese 2^k Lösungen führen zu 2^k Lösungen in $\mathbb{Z}_N$.

Spezialfall RSA-Modul $N = pq$

Korollar Anzahl Lösungen für RSA-Modul

Sei $N = pq$ mit p, q prim. Dann besitzt die Gleichung $x^2 = 1 \mod N$ vier Lösungen in $\mathbb{Z}_N^*$.

Bsp:

- $x^2 = 1 \mod 15$ besitzt die Lösungen

$$\left| \begin{array}{l} x_1 = 1 \mod 3 \\ x_1 = 1 \mod 5 \end{array} \right|, \left| \begin{array}{l} x_2 = 1 \mod 3 \\ x_2 = 4 \mod 5 \end{array} \right|, \left| \begin{array}{l} x_3 = 2 \mod 3 \\ x_3 = 1 \mod 5 \end{array} \right|, \left| \begin{array}{l} x_4 = 2 \mod 3 \\ x_4 = 4 \mod 5 \end{array} \right|.$$

- In $\mathbb{Z}_N^*$ entspricht dies $x_1 = 1, x_2 = 4, x_3 = 11, x_4 = 14$.

Faktorisieren mit nicht-trivialen Wurzeln

Satz Faktorisieren mit nicht-trivialen Wurzeln

Sei $N = pq$ ungerade mit p, q prim. Sei $x \in \mathbb{Z}_N^*$ eine Lösung von $x^2 = 1 \bmod N$ mit $x \neq \pm 1 \bmod N$. Dann kann die Faktorisierung von N in Zeit $\mathcal{O}(\log^2 N)$ berechnet werden.

Beweis:

- $x^2 = 1 \bmod N$ besitzt die vier Wurzel
$$1 = (1,1),\ -1 = (-1,-1),\ (1,-1) \text{ und } (-1,1).$$
- OBdA sei $x = (1,-1)$, d.h. $x = 1 \bmod p$ und $x = (-1) \bmod q$.
- D.h. $x - 1 = (0,-2)$ und $x - 1 = 0 \bmod p$, $x - 1 = (-2) \bmod q$.
- Damit gilt p teilt $x - 1$ und q teilt $x - 1$ nicht wegen $q > 2$.
- Daraus folgt $\mathrm{ggT}(N, x-1) = p$.
- p kann in Zeit $\mathcal{O}(\log^2 N)$ mittels EUKLID berechnet werden.
- Analog kann man $\mathrm{ggT}(N, x+1) = q$ zeigen.

RSA Parameter

RSA Parameter
$$\begin{cases} \text{öffentlich:} & N = pq \text{ mit } p, q \text{ prim und } e \in \mathbb{Z}^*_{\phi(N)} \\ \text{geheim:} & d \in \mathbb{Z}^*_{\phi(N)} \text{ mit } ed = 1 \bmod \phi(N). \end{cases}$$

Satz RSA Parameter Generierung

RSA-Parameter (N, e, d) können in Zeit $\mathcal{O}(\log^4 N)$ generiert werden.

Algorithmus RSA Schlüsselgenerierung

EINGABE: 1^k, wobei k ein Sicherheitsparameter ist.

1. $p, q \leftarrow$ Wähle mittels Primzahltest zwei zufällige k-Bit Primzahlen.
2. $N \leftarrow pq$
3. $\phi(N) \leftarrow (p-1)(q-1)$
4. $e \leftarrow$ Wähle zufälliges $e' \in \mathbb{Z}_{\phi(N)}$ bis $\mathrm{ggT}(e', \phi(N)) = 1$.
5. $d \leftarrow e^{-1} \bmod \phi(N)$.

AUSGABE: (N, e, d)

Laufzeit RSA Parameter Generierung

Erwartete Laufzeit:

- Schritt 1: $\mathcal{O}(k^3 \cdot k)$, da eine k-Bit Zahl mit Ws $\approx \frac{1}{k}$ prim ist. (Folgt aus dem Gaußschen Primzahlsatz)
- D.h. wir müssen ca. k Zahlen wählen, bis eine Zahl prim ist.
- Schritt 2-5: $\mathcal{O}(k^2)$, d.h. gesamt $\mathcal{O}(k^4) = \mathcal{O}(\log^4 N)$.
- Man beachte: Laufzeit ist polynomiell im Sicherheitsparameter 1^k.

Definition RSA Ver- und Entschlüsselung

Seien (N, e, d) RSA Parameter.

1. Verschlüsselung: $E : \mathbb{Z}_N \to \mathbb{Z}_N$ mit $m \mapsto m^e \bmod N$
2. Entschlüsselung: $D : \mathbb{Z}_N \to \mathbb{Z}_N$ mit $c \mapsto c^d \bmod N$

Laufzeit:

- E und D können in Zeit $\mathcal{O}(\log^3 N)$ berechnet werden.

Korrektheit RSA Ver- und Entschlüsselung

Satz Korrektheit RSA

Seien (N, e, d) RSA-Parameter. Dann gilt $D(E(m)) = m$ für alle $m \in \mathbb{Z}_N$.

Beweis:

- Nach CRT-Isomorphismus gilt $D(E(m)) = m^{ed} = m \bmod N$ gdw
$$m^{ed} = m \bmod p \text{ und } m^{ed} = m \bmod q.$$
- Wir zeigen die Korrektheit modulo p. (Beweis analog modulo q)
- Wir schreiben $ed = 1 \bmod \phi(N)$ als $ed = 1 + k\phi(N)$ für ein $k \in \mathbb{N}$.
- D.h. $m^{ed} = m^{1+k\phi(N)} = m \cdot (m^{p-1})^{k(q-1)} = m \bmod p$ für $m \in \mathbb{Z}_p^*$.
- Für $m = 0 \notin \mathbb{Z}_p^*$ gilt ebenfalls $0^{ed} = 0 \bmod p$.

Faktorisieren und Berechnen von d

Definition Polynomialzeit-Äquivalenz

Wir nennen zwei Probleme *Polynomialzeit-äquivalent*, falls ein Polynomialzeit-Algorithmus für das eine Problem einen Polynomialzeit-Algorithmus für das andere Problem impliziert.

Satz Faktorisieren und RSA

Die folgenden drei Probleme sind Polynomialzeit-äquivalent.

1. Faktorisieren von N
2. Berechnen von $\phi(N)$
3. Berechnen von d aus (N, e)

Beweis:

- $1 \Rightarrow 2$: Berechne $\phi(N) = (p-1)(q-1)$.
- $2 \Rightarrow 3$: Berechne $d = e^{-1} \bmod \phi(N)$.
- $3 \Rightarrow 1$: Müssen zeigen, dass (N, e, d) das Tupel (p, q) liefert.

Ringschluss: Faktorisieren mit Hilfe von d

Beweisidee: $3 \Rightarrow 1$

- Sei $ed - 1 = k(p-1)(q-1) = 2^r t$ für $r \geq 2$ und t ungerade.
- Satz von Euler: Für beliebiges $a \in \mathbb{Z}_N^*$ gilt $a^{2^r t} = 1 \bmod N$.
- Wir ziehen nun sukzessive Quadratwurzel von $a^{2^r t} = 1 \bmod N$.
- **Fall 1:** Die Quadratwurzeln $a^{2^{r-1} t}, \ldots, a^t \bmod N$ sind alle 1.
  - In diesem Fall muss ein anderes $a \in \mathbb{Z}_N^*$ gewählt werden.
- **Fall 2:** $a^{2^\ell t} = (-1) \bmod N$ für ein $\ell \in \mathbb{Z}_r$.
  - In diesem Fall wird ebenfalls ein anderes $a \in \mathbb{Z}_N^*$ gewählt.
- **Fall 3:** $a^{2^\ell t} \neq (\pm 1) \bmod N$ und $a^{2^{\ell+1} t} = 1 \bmod N$ für ein $\ell \in \mathbb{Z}_r$.
  - D.h. es wurde eine nicht-triviale Quadratwurzel der Eins gefunden.
  - Damit liefert $\mathrm{ggT}(a^{2^\ell t} \pm 1, N)$ die Faktoren p, q.
- Man kann zeigen, dass Fall 3 für $a \in_R \mathbb{Z}_N^*$ mit Ws mind. $\frac{1}{2}$ eintritt.
- D.h. im Erwartungswert wählt man zwei $a \in_R \mathbb{Z}_N^*$.
- Die erwartete Laufzeit zum Faktorisieren ist damit $\mathcal{O}(\log^3 N)$.

Sicherheit von RSA

RSA Problem:

Gegeben: $N, e, c = m^e \bmod N$
Gesucht: $m \in \mathbb{Z}_N$

Anmerkungen:

- D.h. $m = c^{\frac{1}{e}}$ ist die e-te Wurzel von c in $\mathbb{Z}_N$.
- Das RSA-Problem ist nicht für alle m, e schwer.
- Z.B. kann für $m^e < N$, die Wurzel $c^{\frac{1}{e}}$ effizient berechnet werden.
- Das RSA-Problem ist nicht NP-schwer.
  (unter geeigneten Komplexitätsannahmen)
- RSA-Problem besitzt auf Quantenrechnern Laufzeitkomplexität
$$\mathcal{O}(\log^2 N \cdot \log\log N \cdot \log\log\log N).$$
- **Offenes Problem:** Sind das RSA Problem und das Faktorisierungsproblem Polynomialzeit-äquivalent?

Polynome

Definition Polynom

Sei $R \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}\}$. Wir bezeichnen die Elemente der Menge

$$R[x] = \{\sum_{i=1}^{n} a_i x^i \mid n \geq (-1), a_i \in R \text{ und } a_n \neq 0\}$$

als *Polynome über R*. Sei $p(x) = \sum_{i=1}^{n} a_i x^i \in R[x]$.

1. $a_0, \ldots, a_n$ heißen Koeffizienten von $p(x)$.
2. a_n heißt *führender Koeffizient*. $p(x)$ heißt *monisch* falls $a_n = 1$.
3. $p(x)$ besitzt *Grad* $grad(p) = n$. Das Nullpolynom $p'(x) = \sum_{i=0}^{-1} a_i x_i$ besitzt Grad -1.
4. *Evaluation* von $p(x)$ ist die Abbildung $\text{eval} : R \to R$ mit $x_0 \mapsto p(x_0)$.

Abkürzende Notation: p statt $p(x)$.

Anmerkungen

- Wir nehmen vereinfachend an, dass die Operationen $(+, -, \cdot, :)$ auf Elementen aus R in Zeit $\mathcal{O}(1)$ ausgeführt werden können.

Komplexität von eval, Addition und Subtraktion

Evaluation:

- Naives Auswerten von $p(x)$ mit Grad n liefert $\sum_{i=0}^{n+1} = \mathcal{O}(n^2)$ Multiplikationen und n Additionen, d.h. Komplexität $\mathcal{O}(n^2)$.
- Das sogenannte **Horner-Schema** wertet $p(x)$ wie folgt aus

$$p(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1)x + a_0.$$

- Dies erfordert n Multiplikationen und n Additionen.
- eval besitzt damit lineare Komplexität $\mathcal{O}(n)$.

Addition/Subtraktion:

- Sei $a(x), b(x) \in R[x]$ mit $n = \mathrm{grad}(a) > \mathrm{grad}(b) = m$.
- Wir setzen $b_{m+1} = \ldots = b_n = 0$.
- Berechne $(a \pm b)(x) = \sum_{i=0}^{n} c_i x^i$ mit $c_i = a_i \pm b_i$ für $i \in [n+1]$.
- Damit besitzt die Addition/Subtraktion lineare Komplexität $\mathcal{O}(n)$.

Multiplikation mit Schulmethode

Multiplikation:

- Seien $a(x), b(x) \in R[x]$ mit Grad n bzw. m und $n \geq m$.
- Wir setzen $a_{n+1} = \ldots a_{n+m} = 0$ und $b_{m+1} = \ldots = b_{n+m} = 0$.
- Dann gilt

$$\begin{aligned}(ab)(x) &= (a_n b_m)x^{n+m} + (a_{n-1}b_m + a_n b_{m-1})x^{n+m-1} + \ldots \\ &\quad + (a_0 b_1 + a_1 b_0)x + a_0 b_0.\end{aligned}$$

- D.h. $(ab)(x) = \sum_{i=0}^{n+m} c_i x^i$ mit $c_i = \sum_{j=0}^{i} a_j b_{i-j}$.
- $(ab)(x)$ besitzt $n + m + 1 = \mathcal{O}(n)$ Koeffizienten.
- Pro Koeffizient benötigt man $\mathcal{O}(n)$ Additionen und Multiplikationen.
- Damit erhalten wir insgesamt eine Komplexität von $\mathcal{O}(n^2)$.

Division mit Schulmethode

Satz Division mit Rest von Polynomen

Seien $a(x), b(x) \in \mathbb{Q}[x]$ mit $b(x) \neq 0$. Dann gibt es eindeutige $q(x), r(x) \in \mathbb{Q}[x]$ mit $a(x) = q(x) \cdot b(x) + r(x)$ und $\text{grad}(r) < \text{grad}(b)$.

Beweis:

- Sei $\text{grad}(a) = n$ und $\text{grad}(b) = m$.
- Für $n < m$ setze $q(x) = 0$ und $r(x) = a(x)$. Sei also $n \geq m$.
- **Existenz:** Beweis per Induktion über n.
- **IA:** Für $n = 0$ gilt $m = 0$. Setze $q(x) = \frac{a_0}{b_0}$ und $r(x) = 0$.
- **IS** $n-1 \to n$: Sei $\text{grad}(a) = n$.
- Dann besitzt $a'(x) = a(x) - \frac{a_n}{b_m} x^{n-m} b(x)$ Grad höchstens $n-1$.
- IV: $\exists q'(x), r'(x)$ mit $a'(x) = q'(x)b(x) + r'(x)$, $\text{grad}(r') < \text{grad}(b)$.
- D.h. $a(x) = a'(x) + \frac{a_n}{b_m} \cdot b(x) = (q'(x) + \frac{a_n}{b_m} x^{n-m})b(x) + r'(x)$ mit $\text{grad}(r') < \text{grad}(b)$.

Eindeutigkeit von Quotient und Rest

Beweis: Eindeutigkeit

- Sei $a = qb + r = q'b + r'$ mit $(q, r) \neq (q', r')$ und $\text{grad}(r), \text{grad}(r') < b$.
- D.h. $b(q - q') = r' - r$.
- Wegen $\text{grad}(r' - r) \leq \max\{\text{grad}(r), \text{grad}(r')\} < b$ folgt $q = q'$.
- Damit ist $0 = r' - r$ und daher $r = r'$.

Anmerkung:

- Beweis liefert Komplexität $\mathcal{O}(n^2)$ der Division mit Rest.

Definition Point-value Form

Sei $a \in R[x]$ mit $\text{grad}(a) \leq n$. Seien $x_0, \ldots, x_n \in R$ paarweise verschieden und $y_i = a(x_i)$ für $i = 0, \ldots, n$. Dann bezeichnen wir die Menge $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ als *Point-value Form* von a.

Koeffizienten aus Point-value Form

Satz Berechnung der Koeffizienten

Sei $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ eine Point-value Form. Dann existiert ein eindeutiges $a \in R[x]$ mit $\mathrm{grad}(a) \leq n$ und $a(x_i) = y_i$ für $i = 0, \ldots, n$.

Beweis:

- Wir definieren die Koeffizienten $a_0, \ldots, a_n$ von a als Lösung von

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

- Die Matrix heißt Vandermonde-Matrix V.
- V ist invertierbar für paarweise verschiedene x_i.
- Damit existiert genau eine Lösung

$$(a_0, \ldots, a_n)^t = V^{-1}(y_0, \ldots, y_n)^t.$$

Lagrange Interpolation

Beweis: alternativ mittels Lagrange Interpolation

- Wir setzen $p_i(x) = \frac{\prod_{j \neq i} x - x_j}{\prod_{j \neq i} x_i - x_j}$ und $a(x) = \sum_{i=0}^{n} y_i p_i(x)$.
- Jedes Polynom p_i besitzt Grad n und damit $\text{grad}(a) \leq n$.
- Es gilt $p_i(x_k) = \begin{cases} 1 & \text{für } i = k \\ 0 & \text{sonst} \end{cases}$.
- Daraus folgt $a(x_i) = y_i$ für $i = 0, \ldots, n$.

Arithmetik in Point-value Form

Addition/Subtraktion:

- Sei $a(x) = \{(x_0, y_0), \ldots, (x_n, y_n)\}$, $b(x) = \{(x_0, y'_0), \ldots, (x_n, y'_n)\}$.
- Dann gilt $(a \pm b)(x) = \{(x_0, y_0 \pm y'_0), \ldots, (x_n, y_n \pm y'_n)\}$.
- D.h. Addition/Subtraktion besitzt lineare Komplexität $\mathcal{O}(n)$.

Multiplikation:

- Seien $\mathrm{grad}(a), \mathrm{grad}(b) \leq n$ und
  $a(x) = \{(x_0, y_0), \ldots, (x_{2n}, y_{2n})\}$, $b(x) = \{(x_0, y'_0), \ldots, (x_{2n}, y'_{2n})\}$.
- Dann gilt $(ab)(x) = \{(x_0, y_0 y'_0), \ldots, (x_{2n}, y_{2n} y'_{2n})\}$.
- D.h. die Multiplikation besitzt ebenfalls lineare Komplexität $\mathcal{O}(n)$.

n-te Einheitswurzeln

Problem:

- Konvertierung von Koeffizientenform in Point-value Form erfordert das Evaluieren an n Stellen, d.h. $\mathcal{O}(n^2)$ mit Horner-Schema.
- Konvertierung von Point-value Form in Koeffizientenform erfordert Lagrange Interpolation, d.h. wiederum Komplexität $\mathcal{O}(n^2)$.

Lösung:

- Werte Polynome an geeigneten Stellen aus, den Einheitswurzeln.

Definition n-te Einheitswurzel

Sei $\omega \in \mathbb{C}$. Wir nennen ω eine *n-te Einheitswurzel* falls $\omega^n = 1$

Anmerkungen:

- Die Werte $e^{2\pi i \frac{k}{n}}$, $k = 0, \ldots, n-1$ sind genau die n-ten Einheitswurzeln. Es gilt $e^{2\pi ik} = (e^{2\pi i})^k = 1$.
- Erinnerung: $e^{i\alpha} = \cos(\alpha) + i \sin(\alpha)$. Die n-ten Einheitswurzeln teilen den Einheitskreis in n Segmente.
- Wir bezeichnen $\omega_n = e^{\frac{2\pi i}{n}}$ als primitive n-te Einheitswurzel.

Einheitswurzeln bilden Gruppe

Satz Gruppe der Einheitswurzeln

Sei $\omega_n = e^{\frac{2\pi i}{n}}$. Dann ist $(\langle \omega_n \rangle, \cdot)$ eine zyklische Gruppe der Ordnung n.

Beweis:

- Die Ordnung von ω_n ist n. Es gilt $\omega_n^n = e^{2\pi i} = 1$.
- Die Gruppe ist abgeschlossen, denn $\omega_n^i \cdot \omega_n^j = \omega_n^{i+j \bmod n}$.
- Das neutrale Element der Gruppe ist 1.
- Das zu ω_n^i inverse Element ist ω_n^{n-i}.

Lemma Eliminationslemma

Für alle $n, k \in \mathbb{N}_0$ und $d \in \mathbb{N}$ gilt $\omega_{dn}^{dk} = \omega_n^k$.

Beweis:

- Es gilt $\omega_{dn}^{dk} = e^{2\pi i \frac{dk}{dn}} = e^{2\pi i \frac{k}{n}} = \omega_n^k$.

Halbierungslemma

Korollar zum Eliminationslemma

Sei $n \in \mathbb{N}$ gerade. Dann gilt $\omega_n^{\frac{n}{2}} = (-1)$.

Beweis:

- Es gilt $\omega_n^{\frac{n}{2}} = \omega_2 = e^{\pi i} = \cos(\pi) + i \sin(\pi) = (-1)$.

Lemma Halbierungslemma

Sei $n \in \mathbb{N}$ gerade. Dann sind die Quadrate der n-ten Einheitswurzeln genau die $\frac{n}{2}$-ten Einheitswurzeln.

Beweis:

- Eliminationslemma liefert $(\omega_n^k)^2 = \omega_n^{2k} = \omega_{\frac{n}{2}}^k$ für $k \in [n]$.
- Je zwei n-te Einheitswurzeln besitzen dasselbe Quadrat, denn

$$(\omega_n^k)^2 = (-\omega_n^k)^2 = (\omega_n^{\frac{n}{2}} \cdot \omega_n^k)^2 = (\omega_n^{k+\frac{n}{2}})^2 \text{ für alle } k \in [n].$$

Die Diskrete Fourier-Transformation (DFT)

Definition Diskrete Fourier-Transformation

Sei $a(x) = \sum_{i=1}^{n} a_i x^i \in R[x]$ vom Grad $n-1$. Sei $y_k = a(\omega_n^k)$ für $k = 0, \ldots, n-1$.

1. Der Vektor $a = (a_0, \ldots, a_{n-1})$ heißt *Koeffizientenvektor* von $a(x)$.
2. $(y_0, \ldots, y_{n-1}) \in \mathbb{C}^n$ heißt *diskrete Fouriertransfomierte* von a.

Wir schreiben $y = \mathrm{DFT}_n(a, \omega)$.

Idee der Schnellen Fourier-Transformation (FFT)

Idee: FFT

- Sei $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Wir nehmen vereinfachend $n = 2^k$ an.
- Wir definieren $a_g(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{\frac{n}{2}-1}$ und
$$a_u(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{\frac{n}{2}-1}.$$
- Dann gilt $a(x) = a_g(x^2) + x \cdot a_u(x^2)$.
- D.h. anstatt $a(x)$ an den n-ten Einheitswurzeln auszuwerten, werten wir $a_g(x)$ und $a_u(x)$ an deren Quadraten aus.
- Quadrate der n-ten Einheitswurzeln sind $\frac{n}{2}$-te Einheitswurzeln.
- D.h. wir evaluieren 2 Polynome vom Grad kleiner $\frac{n}{2}$ an $\frac{n}{2}$ Stellen.
- Danach kombinieren wir die Auswertungen von a_g und a_u.
- Sei $y = (y_0, \ldots, y_{n-1}) = \mathrm{DFT}(a, \omega)$. Dann gilt:
$$y_k = a(\omega_n^k) = a_g(\omega_n^{2k}) + \omega_n^k \cdot a_u(\omega_n^{2k}) = a_g(\omega_{\frac{n}{2}}^k) + \omega_n^k \cdot a_u(\omega_{\frac{n}{2}}^k).$$

FFT-Algorithmus

Algorithmus FFT

EINGABE: $(a_0, \ldots, a_{n-1}, n)$ mit $n = 2^k$

1. If $(n = 1)$ return $a = (a_0)$
2. $\omega_n \leftarrow e^{\frac{2\pi i}{n}}; \omega \leftarrow 1;$
3. $(y_0^g, y_1^g, \ldots, y_{\frac{n}{2}-1}^g) \leftarrow \mathrm{FFT}(a_0, a_2, \ldots, a_{n-2}, \frac{n}{2})$
4. $(y_0^u, y_1^u, \ldots, y_{\frac{n}{2}-1}^u) \leftarrow \mathrm{FFT}(a_1, a_3, \ldots, a_{n-1}, \frac{n}{2})$
5. For $k \leftarrow 0$ to $\frac{n}{2} - 1$
   1. $y_k \leftarrow y_k^g + \omega y_k^u$
   2. $y_{k+\frac{n}{2}} \leftarrow y_k^g - \omega y_k^u$
   3. $\omega \leftarrow \omega \cdot \omega_n$

AUSGABE: $(y_0, \ldots, y_n) = \mathrm{DFT}(a, \omega)$

Laufzeit FFT

Satz FFT

Sei $a = (a_0, \ldots, a_{n-1})$ ein Koeffizientenvektor mit $n = 2^k$. Algorithmus FFT berechnet bei Eingabe von (a, n) in Zeit $\mathcal{O}(n \log n)$ die Ausgabe $\mathrm{DFT}(a, n)$.

Laufzeit: Sei $T(n)$ die Gesamtlaufzeit.

- Rekursion in Schritt 3 und 4: $2T(\frac{n}{2})$.
- Kosten pro Rekursionsschritt in Schritt 2 und 5: $\mathcal{O}(n)$.
- Kosten für $T(1)$ in Schritt 1: $\mathcal{O}(1)$.
- Liefert Rekursionsgleichung $T(n) = 2T(\frac{n}{2}) + \mathcal{O}(n)$, $T(1) = \mathcal{O}(1)$.
- Daraus folgt eine Laufzeit von $T(n) = \mathcal{O}(n \log n)$. (Übungsaufgabe)

Korrektheit FFT

Korrektheit:

- Schritt 1: $y = \mathrm{DFT}(a_0) = a(\omega_1) = a_0\omega_1^0 = a_0$.
- Schritt 2+5.2: ω enthält stets ω_n^k. Die Inititialisierung mit $k = 0$ erfolgt in Schritt 2, das Update erfolgt in Schritt 5.2.
- Schritt 3+4:
  - $(y_0^g, y_1^g, \ldots, y_{\frac{n}{2}-1}^g) = \mathrm{DFT}_{\frac{n}{2}}(a_g, \omega)$, d.h. $y_k^g = a_g(\omega_{\frac{n}{2}}^k) = a_g(\omega_n^{2k})$.
  - $(y_0^u, y_1^u, \ldots, y_{\frac{n}{2}-1}^u) = \mathrm{DFT}_{\frac{n}{2}}(a_u, \omega)$, d.h. $y_k^u = a_u(\omega_{\frac{n}{2}}^k) = a_u(\omega_n^{2k})$.
- Schritt 5.1: Es gilt für $k = 0, \ldots, \frac{n}{2} - 1$

$$y_k = y_k^g + \omega_n^k \cdot y_k^u = a_g(\omega_n^{2k}) + \omega_n^k \cdot a_u(\omega_n^{2k}) = a(\omega_n^k).$$

- Schritt 5.2: Es gilt für $k + \frac{n}{2} = \frac{n}{2}, \ldots, n - 1$

$$\begin{aligned} y_{k+\frac{n}{2}} &= y_k^g - \omega_n^k y_k^u = y_k^g + \omega_n^{k+\frac{n}{2}} y_k^u \\ &= a_g(\omega_n^{2k}) + \omega_n^{k+\frac{n}{2}} a_u(\omega_n^{2k}) \\ &= a_g(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} a_u(\omega_n^{2k+n}) = a(\omega_n^{k+\frac{n}{2}}). \end{aligned}$$

Die inverse Diskrete Fourier Transformation

- Konvertierung von der Point-value Form in Koeffizientenform.
- Dazu stellen wir die DFT als Matrix-Vektor Produkt dar

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

- D.h. die Vandermonde-Matrix besitzt Einträge $v_{ij} = \omega_n^{ij}$ für $i, j \in \mathbb{Z}_n$.
- Man beachte, dass $V = V^T$. D.h. $v_{ij} = v_{ji}$ für alle i, j.
- V ist invertierbar (s. Übung). Damit ist die inverse DFT

$$a = DFT_n^{-1}(y, \omega) = V^{-1}y.$$

Summationslemma

Lemma Summationslemma

Für alle $n, i \in \mathbb{N}$ mit $n \nmid i$ gilt $\sum_{k=0}^{n-1} (\omega_n^i)^k = 0$.

Beweis:

$$\sum_{k=0}^{n-1} (\omega_n^i)^k = \frac{(\omega_n^i)^n - 1}{\omega_n^i - 1} = \frac{(\omega_n^n)^i - 1}{\omega_n^i - 1} = \frac{1^i - 1}{\omega_n^i - 1} = 0.$$

Berechnung der inversen DFT

Lemma Invertieren von V

Sei $V^{-1} = (v'_{i,j})_{1 \le i,j \le n}$. Dann gilt $v'_{i,j} = \frac{1}{n} \cdot \omega_n^{-ij}$.

Beweis:

- Wir zeigen, dass $V \cdot V^{-1} = I_n$, d.h. die $(n \times n)$-Einheitsmatrix.
- Wir betrachten dazu den (i,j)-Eintrag von $V \cdot V^{-1}$ für $i, j \in \mathbb{Z}_n$

$$\sum_{k=0}^{n-1} \omega_n^{ik} \cdot \frac{1}{n} \cdot \omega_n^{-kj} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{k(i-j)}.$$

- **Fall 1:** $i = j$. Dann ist $\omega_n^{k(i-j)} = 1$ und die Summation liefert 1.
- **Fall 2:** $i \neq j$. Es gilt $|i - j| < n$ und damit $n \nmid (i - j)$.
- Summationslemma liefert in diesem Fall $\sum_{k=0}^{n-1} (\omega_n^{(i-j)})^k = 0$.

Korollar Berechnung der inversen DFT

Für die inverse DFT gilt $\mathrm{DFT}_n^{-1}(y, \omega) = \frac{1}{n} \cdot \mathrm{DFT}_n(y, \omega^{-1})$.
D.h. DFT_n und DFT_n^{-1} lassen sich beide in Zeit $\mathcal{O}(n \log n)$ berechnen.

Multiplizieren von Polynomen

Satz FFT Polynom-Multiplikation

Seien $a, b \in R[x]$ vom Grad kleiner n. Dann lässt sich $ab \in R[x]$ in Zeit $\mathcal{O}(n \log n)$ berechnen.

Beweis:

- Seien $a = (a_0, \ldots, a_{n-1})$, $b = (b_0, \ldots, b_{n-1})$ Koeffizientenformen.
- Berechne in Zeit $\mathcal{O}(n \log n)$ die Point-value Formen
  - $(a(\omega_{2n}^0), \ldots, a(\omega_{2n}^{2n-1})) = \mathrm{DFT}_{2n}((a_0, \ldots, a_{n-1}, 0, \ldots, 0), \omega)$ und
  - $(b(\omega_{2n}^0), \ldots, b(\omega_{2n}^{2n-1})) = \mathrm{DFT}_{2n}((b_0, \ldots, b_{n-1}, 0, \ldots, 0), \omega)$
- Multipliziere in Zeit $\mathcal{O}(n)$ punktweise

$$(a(\omega_{2n}^0) \cdot b(\omega_{2n}^0), \ldots, a(\omega_{2n}^{2n-1}) \cdot b(\omega_{2n}^{2n-1})) =: (c(\omega_{2n}^0), \ldots, c(\omega_{2n}^{2n-1})).$$

- Berechne in Zeit $\mathcal{O}(n \log n)$ die inverse DFT als

$$\tfrac{1}{2n} \cdot \mathrm{DFT}_{2n}(c(\omega_{2n}^0), \ldots, c(\omega_{2n}^{2n-1}), \omega^{-1}) := (c_0, \ldots, c_{2n-1}).$$

- $(c_0, \ldots, c_{2n-1})$ ist die Koeffizientenform von ab.

Körper und Ring

Definition Körper und Ring

Sei K eine Menge. Das Tupel $(K, +, \cdot)$ ist ein *Körper* falls

1. $(K, +)$ ist eine additive abelsche Gruppe mit neutralem Element 0.
2. $(K \setminus \{0\}, \cdot)$ ist eine multiplikative abelsche Gruppe.
3. Distributivität: $a \cdot (b + c) = a \cdot b + a \cdot c$ für alle $a, b, c \in K$.

Wir bezeichnen $(K, +, \cdot)$ als *Ring*, falls unter Punkt 2 *nicht* die Existenz von multiplikativen Inversen gefordert wird.

Notation: Wir schreiben abkürzend K statt $(K, +, \cdot)$.

Beispiele:

- $\mathbb{Q}$, $\mathbb{R}$ und $\mathbb{C}$ sind Körper.
- $\mathbb{Z}$ ist kein Körper, aber ein Ring.
- $\mathbb{Q}[x], \mathbb{R}[x], \mathbb{C}[x], \mathbb{Z}[x]$ sind ebenfalls Ringe.

Endlicher Körper

Satz Primkörper

Sei p prim. Dann ist $(\mathbb{Z}_p, +, \cdot)$ ein Körper mit p Elementen. Wir bezeichnen diesen Körper als *Primkörper* $\mathbb{F}_p$.

Beweis:

1. $(\mathbb{Z}_p, +)$ ist eine additive abelsche Gruppe.
2. $(\mathbb{Z}_p \setminus \{0\}, \cdot) = (\mathbb{Z}_p^*, \cdot)$ ist eine multiplikative Gruppe.
3. Die Distributivität folgt aus der Distributivität über $\mathbb{Z}$.

Nullteilerfreiheit in Körpern

Lemma Multiplikation mit Null

Sei K ein Körper. Dann gilt für alle $a \in K$, dass $a \cdot 0 = 0 \cdot a = 0$.

Beweis:

- Es gilt $a \cdot 0 = a \cdot (0 + 0) = a \cdot 0 + a \cdot 0$.
- Subtraktion von $a \cdot 0$ auf beiden Seiten liefert $a \cdot 0 = 0$.
- $0 \cdot a = 0$ folgt aus der Kommutativität.

Satz Nullteilerfreiheit

Sei K ein Körper. Dann gilt für alle $a, b \in K$, dass $ab = 0$ gdw $a = 0$ oder $b = 0$.

Beweis:

- $\Leftarrow$: Folgt aus obigem Lemma.
- $\Rightarrow$: Sei $ab = 0$ und $a \neq 0$. Dann gilt

$$b = 1 \cdot b = a^{-1} \cdot a \cdot b = a^{-1} \cdot 0 = 0.$$

Polynome über Körpern

Definition Polynome über Körpern

Sei K ein Körper. Dann bezeichnen wir ein Polynom $a(x) \in K[x]$ mit Koeffizienten aus K als *Polynom über K*.

Anmerkungen:

- Arithmetik $+, -, \cdot, \mathrm{eval}$ ist für $a(x) \in K[x]$ kanonisch definiert.
- Erinnerung: Für $a, b \in \mathbb{Q}[x]$ mit $b \neq 0$ gibt es $q, r \in \mathbb{Q}[x]$ mit
$$a = q \cdot b + r \quad \text{und } \mathrm{grad}(r) < \mathrm{grad}(b).$$
- Gilt analog, falls $\mathbb{Q}$ durch einen anderen Körper K ersetzt wird.

Fundamentalsatz der Algebra

Satz Fundamentalsatz der Algebra

Sei K ein Körper. Sei $p(x) \in K[x]$ mit $\text{grad}(p) = n \geq 0$. Dann besitzt $p(x)$ höchstens n Nullstellen.

Beweis: per Induktion über n.

- **IV** für $n = 0$: Es gilt $p(x) \neq 0$, d.h. $p(x)$ besitzt keine Nullstelle.
- **IS** für $n - 1 \rightarrow n$.
- **Fall 1:** $p(x)$ besitzt keine Nullstelle und damit höchstens n.
- **Fall 2:** Sei $p(x)$ vom Grad $n \geq 1$ mit Nullstelle x_0.
- Euklidische Division liefert q, r mit
$$p = q \cdot (x - x_0) + r \text{ mit } \text{grad}(r) < 1.$$
- Wir werten $p(x)$ an der Nullstelle x_0 aus
$$0 = p(x_0) = q(x_0) \cdot (x_0 - x_0) + r(x_0) = r(x_0).$$
- Damit folgt $r(x) = 0$ und $p(x) = q(x) \cdot (x - x_0)$ mit $\text{grad}(q) = n - 1$.
- Nach IA besitzt q höchstens $n - 1$ Nullstellen.
- Damit besitzt p höchstens n Nullstellen.

Ordnungen und Teilbarkeit

Ziel: Wir wollen zeigen, dass $(K^*, \cdot)$ für jeden Körper K zyklisch ist.

Lemma Teilbarkeit der Ordnung

Sei G eine abelsche (multiplikative) Gruppe. Dann gilt für alle $a \in G$ und $k \in N$, dass $a^k = 1$ gdw $\mathrm{ord}(a) | k$.

Beweis:

- $\Leftarrow$: Sei $k = \mathrm{ord}(a)q$ für ein $q \in \mathbb{N}$. Dann ist $a^k = (a^{\mathrm{ord}(a)})^q = 1$.
- $\Rightarrow$: Falls $a^k = 1$, dann gilt $k \geq \mathrm{ord}(a)$.
- Euklidische Division: q, r mit $k = q \cdot \mathrm{ord}(a) + r$ und $r < \mathrm{ord}(a)$.
- Es gilt $1 = a^k = a^{q \cdot \mathrm{ord}(a) + r} = 1^q \cdot a^r = a^r$ mit $r < \mathrm{ord}(a)$. (Widerspruch zur Minimalität von $\mathrm{ord}(a)$)

Multiplikativität der Ordnung

Lemma Multiplikativität der Ordnung

Sei G eine abelsche Gruppe. Seien a, b in G mit $\mathrm{ggT}(\mathrm{ord}(a), \mathrm{ord}(b)) = 1$. Dann gilt $\mathrm{ord}(ab) = \mathrm{ord}(a) \cdot \mathrm{ord}(b)$.

Beweis:

- Es gilt $\mathrm{ord}(ab) | \mathrm{ord}(a) \cdot \mathrm{ord}(b)$ wegen
$$(ab)^{\mathrm{ord}(a)\cdot\mathrm{ord}(b)} = (a^{\mathrm{ord}(a)})^{\mathrm{ord}(b)} \cdot (b^{\mathrm{ord}(b)})^{\mathrm{ord}(a)} = 1.$$
- Ann: $\mathrm{ord}(ab) \cdot k = \mathrm{ord}(a) \cdot \mathrm{ord}(b)$ mit $k > 1$.
- OBdA $k' = \mathrm{ggT}(\mathrm{ord}(a), k) > 1$ mit $k' | k$. Dann gilt
$$1 = (ab)^{\frac{\mathrm{ord}(a)\mathrm{ord}(b)}{k'}} = a^{\frac{ord(a)\mathrm{ord}(b)}{k'}} \cdot (b^{\mathrm{ord}(b)})^{\frac{\mathrm{ord}(a)}{k}'} = a^{\frac{ord(a)\mathrm{ord}(b)}{k'}}.$$
- D.h. $\mathrm{ord}(a) | \frac{ord(a)\mathrm{ord}(b)}{k'}$ und $\mathrm{ggT}(\mathrm{ord}(a), \mathrm{ord}(b)) = 1$.
- Damit folgt $\mathrm{ord}(a) | \frac{ord(a)}{k'}$. (Widerspruch wegen $k' > 1$)

Elementordnung teilt maximale Ordnung

Lemma Ordnung teilt maximale Ordnung

Sei G eine endliche abelsche Gruppe. Sei $a \in G$ mit maximaler Ordnung. Dann gilt für alle $b \in G$, dass $\mathrm{ord}(b) | ord(a)$.

Beweis:

- Annahme: $\mathrm{ord}(b) \nmid \mathrm{ord}(a)$
- D.h. es existiert eine Primzahlpotenz p^i mit
$$p^{i+1} \mid \mathrm{ord}(b) \text{ und } p^i \mid \mathrm{ord}(a) \text{ aber } p^{i+1} \nmid \mathrm{ord}(a).$$
- Wir definieren $a' = a^{p^i}$ und $b' = b^{\frac{\mathrm{ord}(b)}{p^{i+1}}}$.
- Damit gilt $\mathrm{ord}(a') = \frac{\mathrm{ord}(a)}{p^i}$ und $\mathrm{ord}(b') = p^{i+1}$.
- Wegen $p \nmid \mathrm{ord}(a')$ folgt $\mathrm{ggT}(\mathrm{ord}(a'), ord(b')) = 1$.
- Lemma zur Multiplikativität: $\mathrm{ord}(a'b') = \mathrm{ord}(a) \cdot p > \mathrm{ord}(a)$. (Widerspruch zur Maximalität von $\mathrm{ord}(a)$)

K^* ist zyklisch

Satz K^* ist zyklisch

Sei K ein endlicher Körper. Dann ist die multiplikative Gruppe $(K^*, \cdot) = (K \setminus \{0\}, \cdot)$ zyklisch.

- Sei $a \in K^*$ ein Element maximaler Ordnung.
- Die Elementordnung teilt die Gruppenordnung, d.h. $\mathrm{ord}(a) || K^* |$.
- Wir betrachten das Polynom $p(x) = x^{\mathrm{ord}(a)} - 1$.
- Für alle $b \in K^*$ gilt $\mathrm{ord}(b) \mid \mathrm{ord}(a)$.
- Damit ist $p(b) = b^{\mathrm{ord}(a)} - 1 = 0$, d.h. jedes b ist eine Nullstelle.
- D.h. $p(x)$ besitzt mindestens $|K^*|$ viele Nullstellen.
- Fundamentalsatz: Jedes Polynom vom Grad $\mathrm{ord}(a)$ besitzt höchstens $\mathrm{ord}(a)$ viele Nullstellen über einem Körper K.
- Es folgt $|K^*| \leq \mathrm{ord}(a)$. Mit $\mathrm{ord}(a) \mid |K^*|$ gilt daher $\mathrm{ord}(a) = |K^*|$.
- D.h. das Element a ist ein Generator für K^*.

Anzahl der Generatoren

Satz Anzahl Generatoren eines Körpers

Sei K ein Körper mit q Elementen. Dann besitzt K^* genau $\phi(q-1)$ viele Generatoren.

Beweis:

- K^* ist zyklisch, d.h. K^* besitzt einen Generator a mit
$$K = \{a, a^2, \ldots, a^{|K^*|}\}.$$
- Wir bestimmen $\mathrm{ord}(a^j)$ für ein $j \in \mathbb{Z}_{|K^*|}$. D.h. wir suchen ein minimales $k = ord(a^j)$ so dass jk ein Vielfaches von $|K^*|$ ist.
- Für $k = \frac{|K^*|}{\mathrm{ggT}(j,|K^*|)}$ gilt $jk = \frac{j}{\mathrm{ggT}(j,|K^*|)} \cdot |K^*| = 0 \bmod |K^*|$.
- k ist minimal mit dieser Eigenschaft, d.h. $\mathrm{ord}(a^j) = \frac{|K^*|}{\mathrm{ggT}(j,K^*)}$.
- D.h. falls $\mathrm{ggT}(j, |K^*|) = 1$, dann gilt $\mathrm{ord}(a^j) = |K^*| = q-1$.
- Es existieren $|\{j \in \mathbb{Z}_{|K^*|} \mid \mathrm{ggT}(j, |K^*|) = 1\}| = \phi(|K^*|) = \phi(q-1)$ viele Elemente mit Ordnung $q-1$.
- D.h. es gibt $\phi(q-1)$ viele Generatoren in K^*.

Konstruktion von Generatoren

Satz Konstruktion von Generatoren

Sei K^* zyklisch. Ein Element $a \in K^*$ ist ein Generator falls $a^k \neq 1$ für alle nicht-trivialen Teiler k von $|K^*|$.

Beweis:

- Da K^* zyklisch ist, besitzt es einen Generator mit Ordnung $|K^*|$.
- Die Ordnung jeden Elements a teilt die maximale Ordnung $|K^*|$.
- D.h. es genügt, für $\mathrm{ord}(a)$ alle möglichen Teiler von $|K^*|$ zu testen.

Beispiel: zyklische Gruppe $\mathbb{Z}_{11}^*$

- $\mathbb{Z}_{11}^*$ besitzt 10 Elemente und $\phi(10) = 4$ Generatoren.
- 2 ist ein Generator, denn $2^2 = 4$ und $2^5 = 10$.
- Da $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$, sind auch die Elemente $2^3 = 8, 2^7 = 7$ und $2^9 = 6$ Generatoren von $\mathbb{Z}_{11}^*$.

Teilbarkeitsbegriff für Polynome

Definition Teilbarkeit von Polynomen

Sei K ein Körper und $f, g, \pi \in K[x]$. Wir definieren

- $g \mid f$ gdw ein $h \in K[x]$ existiert mit $gh = f$.
- $f = g \bmod \pi$ gdw π die Differenz $f - g$ teilt.
- $\mathrm{ggT}(f, g)$ ist ein Polynom maximalen Grads, das sowohl f als auch g teilt. Vorsicht: $\mathrm{ggT}(f, g)$ ist im allgemeinen nicht eindeutig.

Anmerkungen:

- Bei der Reduktion modulo π erhalten wir Äquivalenzklassen.
- Repräsentanten der Äquivalenzklassen sind
$$R = \{f \in K[x] | \mathrm{grad}(f) < \mathrm{grad}(\pi)\}.$$
- Sei K ein Körper mit p Elementen und $\mathrm{grad}(\pi) = n$.
- Dann gilt $|R| = p^n$. Wir schreiben $R = K[x]/(\pi(x))$.

Erweiterter Euklischer Algorithmus für Polynome

Algorithmus ERWEITERTER EUKLIDISCHER ALG. (EEA)

EINGABE: $a(x), b(x) \in K[x]$

1. If $(b = 0)$ then return $(a, 1, 0)$;
2. $(d, r, s) \leftarrow \text{EEA}(b, a \bmod b)$;
3. $(d, r, s) \leftarrow (d, s, r - \lfloor \frac{a}{b} \rfloor s)$;

AUSGABE: $d = \text{ggT}(a, b) = ra + sb$

Korrektheit:

- Analog zu EEA über $\mathbb{Z}$.

Beispiel für EEA

Beispiel: $\mathrm{ggT}(x^3+2x^2-1, x^2+x)$ in $\mathbb{F}_3[x]$

a	b	$\lfloor \frac{a}{b} \rfloor$	r	s
x^3+2x^2-1	x^2+x	$x+1$	1	$-x-1$
x^2+x	$2x-1$	$2x$	0	1
$2x-1$	0	-	1	0

- D.h. $\mathrm{ggT}(a,b) = 2x-1$. Man beachte, dass der ggT eindeutig bis auf Multiplikation mit Elementen aus $\mathbb{F}_3^*$ ist.
- Z.B. ist $2(2x-1) = x+1$ ebenfalls ein Teiler von a, b.
- $x-(-1)$ teilt a, b, da (-1) gemeinsame Nullstelle von a, b ist.
- Der EEA für Polynome liefert die Linearkombination

  $\mathrm{ggT}(a,b) = ra+sb = x^3+2x^2-1-(x+1)(x^2+x) = -x-1 = 2x-1.$

Irreduzible Polynome

Definition Irreduzibilität von Polynomen

Sei K ein Körper und $\pi \in K[x]$. Wir bezeichnen π als *irreduzibel über* K gdw jede Zerlegung $\pi = \pi_1 \cdot \pi_2$ mit $\pi_1, \pi_2 \in K[x]$ impliziert dass $\mathrm{grad}(\pi_1) = 0$ oder $\mathrm{grad}(\pi_2) = 0$.
Andernfalls bezeichen wir π als *reduzibel über* K.

Beispiel: Polynom $f(x) = x^2 + 1$

- f ist irreduzibel über $\mathbb{R}$, denn $f(x) > 0$ für alle $x \in R$.
- D.h. f besitzt keine Nullstelle x_0 in $\mathbb{R}$ und damit können wir keinen Linearfaktor $(x - x_0)$ von $f(x)$ abspalten.
- f ist reduzibel über $\mathbb{C}$, denn $f(x) = (x + i)(x - i)$.
- f ist irreduzibel über $\mathbb{F}_3$, denn weder $0, 1$ noch 2 sind Nullstellen von $f(x)$ in $\mathbb{Z}_3$.
- f ist reduzibel über $\mathbb{F}_2$, denn $f(x) = (x + 1)(x + 1)$.

Polynomringe modulo reduziblen Polynomen

- Sei K ein Körper und $\pi \in K[x]$ reduzibel.
- Dann gilt $\pi = \pi_1 \cdot \pi_2$ mit $0 < \text{grad}(\pi_1), \text{grad}(\pi_2) < \text{grad}(\pi)$.
- Wir definieren den Ring $K[x]/(\pi)$.
- Es gilt $\pi_1, \pi_2 \in K[x]/(\pi)$.
- Ferner gilt $\pi_1 \cdot \pi_2 = \pi = 0$ in $K[x]/(\pi)$.
- D.h. das Produkt von zwei von Null verschiedenen Elementen liefert Null in $K[x]/(\pi)$. Damit sind π_1, π_2 Nullteiler in $K[x]/(\pi)$.
- Es folgt, dass der Ring $K[x]/(\pi)$ kein Körper sein kann.

Galoiskörper

Satz Galoiskörper

Sei K ein Körper mit p Elementen. Sei $\pi \in K[x]$ irreduzibel vom Grad n. Dann ist $K[x]/(\pi)$ ein Körper mit $q = p^n$ Elementen.

Notation: $K[x]/(\pi) = \mathbb{F}_{p^n} = \mathbb{F}_q$ oder auch *Galoiskörper* $GF(q)$.

Beweisskizze:

- **Additiv:** $(K[x]/(\pi), +)$ ist eine abelsche Gruppe.
- Dazu definieren wir $H = \{k \cdot \pi \mid k \in K[x]\}$.
- H ist eine Untergruppe von $K[x]$.
- $K[x]/(\pi)$ ist isomorph zur Faktorgruppe $K[x]/H$.
- **Multiplikativ:** $((K[x] \setminus \{0\})/(\pi), \cdot)$ ist eine abelsche Gruppe.
- Abgeschlossenheit: Seien $f, g \in K[x] \setminus \{0\}$.
- Annahme: $fg = 0 \bmod \pi$. Dann folgt $\pi \mid fg$ und damit $\pi \mid f$ oder $\pi \mid g$. (Widerspruch, da $f, g \neq 0$)
- Berechnen von Inversen liefert der EEA für Polynome.
- **Distributivgesetz:** Folgt aus Distributivität von $K[x]$.

Beispiel: Körper $\mathbb{F}_4$

Beispiel: $\mathbb{F}_{2^2} = \mathbb{F}_4$

- Der Körper $\mathbb{F}_4$ ist eine Körpererweiterung des Körpers $\mathbb{F}_2$.
- Über $\mathbb{F}_2$ ist $\pi = x^2 + x + 1$ irreduzibel, denn $\pi(0) = \pi(1) = 1$.
- D.h. wir können den $\mathbb{F}_4$ definieren als $\mathbb{F}_2[x]/(x^2 + x + 1)$.
- Die Repräsentanten des $\mathbb{F}_4$ sind $\{0, 1, x, x + 1\}$.
- Berechnung des Inversen von x mittels EEA(π, x) liefert
$$1 \cdot (x^2 + x + 1) + (x + 1) \cdot x = 1.$$
- Das Inverse von x in $\mathbb{F}_4$ ist $x + 1$, denn $(x + 1)x = 1 \bmod \pi$.

- Alternativ können wir $x^{-1} = ax + b$ setzen und a, b ermitteln:
$1 = (ax + b)x = ax^2 + bx = a(-x - 1) + bx = (a + b)x + a$ in $\mathbb{F}_4$.
- Koeffizientenvergleich liefert $a = 1$ und $a + b = 0$, d.h. $b = 1$.
- Damit ist $ax + b = x + 1$ das Inverse von x in $\mathbb{F}_4$.

- $\mathbb{F}_4$ besitzt die $\phi(3) = 2$ Generatoren x und $x + 1$.

Existenz und Eindeutigkeit der Galoiskörper $\mathbb{F}_{p^n}$

Satz Existenz von irreduziblen Polynomen

Für jedes prime p und jedes $n \in \mathbb{N}$ existiert ein irreduzibles Polynom $\pi \in \mathbb{F}_p[x]$ vom Grad n.

(ohne Beweis)

Korollar Existenz eines Galoiskörpers $\mathbb{F}_{p^n}$

Für jedes prime p und jedes $n \in \mathbb{N}$ existiert ein Galoiskörper $\mathbb{F}_{p^n} = \mathbb{F}_p[x]/(\pi)$ für ein irreduzibles $\pi \in \mathbb{F}_p[x]$.

Satz Eindeutigkeit der Galoiskörper

Je zwei endliche Körper mit gleicher Anzahl von Elementen sind isomorph.

(ohne Beweis)

Abschnitt: Algorithmendesign und Laufzeitanalyse

Definition Divide-and-Conquer Paradigma

Divide-and-Conquer Algorithmen verwenden die Strategien

1. **Divide:** Teile das Problem *rekursiv* in Subproblem gleicher Struktur auf. Sofern die Größe der Subprobleme hinreichend klein ist, verwende trivialen Algorithmus zum Lösen.
2. **Conquer:** Kombiniere Lösungen der Subprobleme zur Lösung des Ausgangsproblems.

Beispiele:

- Multiplikation von zwei n-Bit Zahlen mit Karatsuba-Methode.
  - Divide: Teile in Multiplikation von drei $\frac{n}{2}$-Bit Zahlen.
  - Conquer: Kombiniere die $\frac{n}{2}$-Bit Ergebnisse zur n-Bit Lösung.
- Multiplikation von zwei Polynomen vom Grad n mittels FFT.
  - Divide: Teile in zwei Polynome a_g, a_u vom Grad $\frac{n}{2}$.
  - Conquer: Kombiniere $a(x) = a_g(x^2) + x \cdot a_u(x^2)$.

Idee der Binären Suche

Problem Suche eines Elements

Gegeben: Sortiertes Array $a[1 \ldots n]$ mit $a[1] < \ldots < a[n]$, Element x

Gesucht: Index $i \in \mathbb{Z}_{n+1}$ mit $i = \begin{cases} j & \text{falls } x = a[j] \text{ für ein } j \in [n] \\ 0 & \text{sonst.} \end{cases}$

Lösung mittels Divide and Conquer

- Wir treffen die vereinfachende Annahme $n = 2^k$.
- **Divide:** Teile $a[1 \ldots n]$ in Teilarrays $a[1 \ldots \frac{n}{2}]$ und $a[\frac{n}{2} + 1 \ldots n]$.
- **Conquer:** Falls $x \leq a[\frac{n}{2}]$, ist x im linken Teilarray.
- Falls $x > a[\frac{n}{2}]$, ist x im rechten Teilarray.
- Das Problem wird rekursiv im korrekten Teilarray gelöst.
- Kombination erfordert die korrekte Verwaltung der Indizes.

Binäre Suche

Algorithmus BINÄRE-SUCHE

EINGABE: Array a aufsteigend sortiert, x, Intervallgrenzen ℓ, r

1. If $(\ell = r)$ then
   1. If $(x = a[r])$ return r;
   2. else return 0;
2. else
   1. $m \leftarrow \ell + \lfloor \frac{r-\ell}{2} \rfloor$;
   2. If $x \leq a[m]$ then BINÄRE-SUCHE(a, x, ℓ, m);
   3. else BINÄRE-SUCHE$(a, x, m+1, r)$;

AUSGABE: Index $i \in \mathbb{Z}_{n+1}$ mit $i = \begin{cases} j & \text{falls } x = a[j] \text{ für ein } j \in [n] \\ 0 & \text{sonst.} \end{cases}$

- Initialer Aufruf ist BINÄRE-SUCHE$(a, x, 1, n)$.
- Korrektheit folgt aus der Überlegung auf der vorigen Folie.

Laufzeit Binäre Suche

Satz Laufzeit BINÄRE-SUCHE

BINÄRE-SUCHE benötigt für Arrays der Länge $n = 2^k$ genau $k + 1 = \log_2 n + 1$ Elementvergleiche.

Beweis: per Induktion über k

- **IV** für $k = 0$, d.h. $n = 1$: Benötigen einen Vergleich in Schritt 1.1.
- **IS** $k - 1 \leftarrow k$: Wir benötigen einen Vergleich $x \leq a[m]$.
- Rekursion liefert Teilarray der Größe $\frac{n}{2} = 2^{k-1}$.
- Nach IA benötigen wir für das Teilarray $(k - 1) + 1$ Vergleiche.
- Damit erhalten wir insgesamt $k + 1$ Elementvergleiche.

Anmerkung für beliebiges n

- Wir erhalten $\lceil \log n \rceil$ durch Auffüllen von a auf die nächste 2er-Potenz.

Sortieren von n Elementen

Problem Sortieren von n Elementen

Gegeben: Array $a[1 \ldots n]$
Gesucht: $a[1 \ldots n]$ mit $a[1] \leq a[2] \leq \ldots \leq a[n]$

Idee: Sortiere $a[i]$ in $a[1 \ldots i-1]$ ein für $i = 2, \ldots, n$.

Algorithmus INSERTION-SORT

EINGABE: $a[1 \ldots n]$

1. For $i \leftarrow 2$ to n do
   1. merk $\leftarrow a[i]$; $j \leftarrow i-1$;
   2. While ($j > 0$ and merk$< a[j]$) do $a[j+1] \leftarrow a[j]$; $j \leftarrow j-1$;
   3. $a[j+1] \leftarrow$ merk;

AUSGABE: $a[1 \ldots n]$ aufsteigend sortiert

Korrektheit und Laufzeit INSERTION-SORT

- **Korrektheit:** Nach der i-ten Iteration ist $a[1 \ldots i]$ sortiert.
- D.h. für $i = n$ ist das Array $a[1 \ldots n]$ aufsteigend sortiert.

- **Laufzeit:** In der i-ten Iteration werden $\leq i - 1$ Vergleiche benötigt.
- Damit ist die Gesamtzahl der Vergleiche höchstens
$$\sum_{i=2}^{n} i - 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \mathcal{O}(n^2).$$
- Man beachte: Im best-case, d.h. für aufsteigend sortierte Arrays, werden nur $n - 1 = \mathcal{O}(n)$ Vergleiche benötigt.

Divide and Conquer Version von Insertion-Sort

- Wir formulieren Insertion-Sort mittels Divide and Conquer.
- **Divide:** Splitte $a[1 \ldots n]$ in $a[1 \ldots n-1]$ und $a[n]$.
- **Conquer:** Sortiere $a[1 \ldots n-1]$ rekursiv. Abbruch für $a[1]$.
- Kombination: Sortiere $a[i]$ in sortiertem $a[1 \ldots i-1]$ ein.
- Sei $T(i)$ die Anzahl der Vergleiche, um $a[i]$ einzusortieren.
- Dann gilt $T(i) \leq i-1$ mit naivem Verfahren.
- Damit erhalten wir wie zuvor

$$T(n) \leq T(n-1) + n - 1 \leq \ldots \leq T(1) + \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}.$$

- **Frage:** Was passiert, wenn man statt des naiven Verfahrens BINÄRE-SUCHE verwendet?
- Dann kann mit nur $\mathcal{O}(\log i)$ Vergleichen der korrekte Index j zum Einsortieren für $a[i]$ gefunden werden.
- Allerdings benötigt das Verschieben des Arrays Zeit $\mathcal{O}(i-j)$.

Das Mergesort Verfahren

Idee des Mergesort Verfahrens

- Mergesort verwendet eine Divide and Conquer Strategie.
- Wir treffen die vereinfachende Annahme $n = 2^k$.
- **Divide:** Teile a in zwei Teilarrays gleicher Größe.
- **Conquer:** Sortiere die Teilarrays rekursiv.
- Ein Abbruch der Rekursion erfolgt für Arraylänge 1.
- Kombination: Verschmelze, d.h. merge, zwei sortierte Teilarrays der Länge $\frac{n}{2}$ zu einem sortierten Array der Länge n.

Verschmelzen von zwei Teilarrays

Algorithmus MERGE

EINGABE: $a[\ell \ldots m]$, $a[m+1 \ldots r]$ jeweils aufsteigend sortiert

1. $p_1 \leftarrow \ell$; $p_2 \leftarrow m+1$; $i \leftarrow 1$;
2. While ($p_1 \leq m$ and $p_2 \leq r$)
   1. If ($a[p_1] < a[p_2]$) then $b[i] \leftarrow a[p_1]$; $p_1 \leftarrow p_1 + 1$;
   2. else $b[i] \leftarrow a[p_2]$; $p_2 \leftarrow p_2 + 1$;
   3. $i \leftarrow i + 1$;
3. If ($p_1 \leq m$) then kopiere $a[p_1 \ldots m]$ nach $b[i \ldots r - \ell + 1]$.
4. else kopiere $a[p_2 \ldots r]$ nach $b[i \ldots r - \ell + 1]$.
5. Kopiere $b[1 \ldots r - \ell + 1]$ nach $a[\ell \ldots r]$.

AUSGABE: $a[l \ldots r]$ aufsteigend sortiert

Korrektheit und Laufzeit von MERGE

- **Korrektheit:** Array b enthalte i Elemente.
- Dann gilt $b[i] \leq a[j]$ für $j = p_1, \ldots, m$ und $j = p_2, \ldots, r$.
- D.h. $b[1 \ldots i]$ ist aufsteigend sortiert und enthält die i kleinsten Elemente beider Teilarrays.
- Damit ist $b[1 \ldots r - \ell + 1] = a[\ell \ldots r]$ aufsteigend sortiert.

- **Laufzeit:** Wir benötigen höchstens $r - \ell$ Vergleiche.
- D.h. die Anzahl der Vergleiche ist linear in der Intervallänge von a.

Mergesort

Algorithmus MERGESORT

EINGABE: $a[1 \ldots n], \ell, r$

1. If $(\ell < r)$
   1. $m \leftarrow \ell + \lfloor \frac{r-\ell}{2} \rfloor$;
   2. MERGESORT(a, ℓ, m);
   3. MERGESORT$(a, m+1, r)$;
   4. MERGE$(a[\ell \ldots m], a[m+1 \ldots r])$;

AUSGABE: $a[1 \ldots n]$ aufsteigend sortiert

- **Korrektheit:** Folgt aus Korrektheit von MERGE.
- **Laufzeit:** Sei $T(n)$ die Anzahl der Vergleiche.
- Dann gilt $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(n)$ mit $T(1) = \mathcal{O}(1)$.
- Lösung der Rekursion liefert $T(n) = \mathcal{O}(n \log n)$.
- Man beachte, dass MERGESORT im best-case ebenfalls $\mathcal{O}(n \log n)$ Vergleiche benötigt.

Das Quicksort-Verfahren

Idee des Quicksort-Verfahrens

- **Divide:** Wähle sogenanntes Pivotelement $p = a[i]$ für ein $i \in [n]$.
- Partitioniere das Array $a[n]$ in zwei nicht-leere Teilarrays $a[1 \ldots j], a[j+1 \ldots n]$ mit Elementen kleiner bzw. größer als p.
- D.h. $a[k] \leq p$ für $k = 1, \ldots, j$ und $a[k] \geq p$ für $k = j+1, \ldots, n$.
- **Conquer:** Sortiere beide Teilarrays rekursiv.
- Kombination: $a[n] = a[1 \ldots j]a[j+1 \ldots n]$ ist bereits sortiert.

Algorithmus QUICKSORT

EINGABE: $a[1 \ldots n], \ell, r$

1. If $(\ell < r)$
   1. $j \leftarrow$ PARTITION(a, ℓ, r);
   2. QUICKSORT(a, ℓ, j);
   3. QUICKSORT$(a, j+1, r)$;

AUSGABE: $a[1 \ldots n]$ aufsteigend sortiert

Die Partitionierung eines Arrays

Algorithmus PARTITION

EINGABE: $a[\ell \ldots r]$

1. $p \leftarrow a[\ell]$; $i \leftarrow \ell - 1$; $j \leftarrow r + 1$;
2. While (TRUE)
   1. Repeat ($i \leftarrow i + 1$) until $a[i] \geq p$;
   2. Repeat ($j \leftarrow j - 1$) until $a[j] \leq p$;
   3. If ($i < j$) vertausche $a[i] \leftrightarrow a[j]$;
   4. else return j;

AUSGABE: j mit $a[\ell], \ldots, a[j] \leq p$ und $p \leq a[j+1], \ldots, a[r]$.

- **Korrektheit:** $a[\ell \ldots i-1]$ enthält stets nur Elemente $\leq p$.
- Teilarray $a[j+1 \ldots r]$ enthält stets nur Elemente $\geq p$.
- Ein Abbruch der While-Schleife erfolgt für $i \geq j$.
- **Laufzeit:** Wir benötigen maximal $r - \ell + 1$ Vergleiche.
- D.h. die Anzahl der Vergleiche ist linear in der Intervallänge von a.

Eigenschaften von Quicksort

Laufzeit:

- Im worst-case: $T(n) = T(1) + T(n-1) + \mathcal{O}(n) = \mathcal{O}(n^2)$.
- Im best-case: $T(n) = 2 \cdot T(\frac{n}{2}) + \mathcal{O}(n) = \mathcal{O}(n \log n)$.
- Im average-case: Man kann zeigen, dass $T(n) = \mathcal{O}(n \log n)$ bei zufälliger Wahl des Pivotelements gilt.

Vorteile:

- Die Konstanten in der $\mathcal{O}$-Notation sind klein.
- QUICKSORT sortiert *in place*, d.h. QUICKSORT benötigt keinen zusätzlichen Speicherplatz.
- MERGESORT: Verwendung von zusätzlichem Array zum Mergen.
- QUICKSORT ist in der Praxis oft der schnellste Algorithmus.

Entscheidungsbäume

Frage: Gibt es einen Sortieralgorithmus mit $o(n \log n)$ Vergleichen?

Definition Entscheidungsbaum

Sei T ein Binärbaum und $A = \{a_1, \ldots, a_n\}$ eine zu sortierenden Menge. T ist ein *Entscheidungsbaum* für A, falls

1. innere Knoten einen Vergleich von $a_i, a_j \in A$ enthalten. Falls $a_i < a_j$ wird nach links verzweigt, sonst nach rechts.
2. Blätter eine Permutation $a_{\pi(1)}, \ldots, a_{\pi(n)}$ enthalten, die direkt aus den Vergleichen folgt.

Beispiel: Entscheidungsbaum T für a_1, a_2, a_3

- 3 paarweise Vergleiche genügen, um 3 Elemente zu sortieren.
- Damit besitzt T Tiefe 3.
- Ferner besitzt T als Blätter die $3! = 6$ Permutationen von A.

Untere Schranke für die Anzahl Vergleiche

Satz Untere Schranke für die Anzahl Vergleiche

Jeder vergleichsbasierte Sortieralgorithmus benötigt zum Sortieren von n Elementen $\Omega(n \log n)$ Vergleiche im worst-case.

Beweis:

- Sei T ein Entscheidungsbaum für A, $|A| = n$ der Höhe h.
- Damit muss T als Blätter alle Permutationen von A enthalten.
- D.h. T besitzt mindestens $n!$ Blätter. Jeder Binärbaum der Höhe h besitzt andererseits höchstens 2^h Blätter.
- Daher gilt $2^h \geq n!$ bzw. $h \geq \log_2(n!)$.
- Aus der Stirling-Formel folgt $n! > \left(\frac{n}{e}\right)^n$ und damit

$$h \geq n \log_2 \left(\frac{n}{e}\right) = n \log n - n \log e = \Omega(n \log n).$$

- D.h. es gibt einen Pfad in T mit $\Omega(n \log n)$ Vergleichen.

Berechnung von Binomialkoeffizienten

Problem: Berechne aus $n, k \in \mathbb{N}_0$ effizient $\binom{n}{k}$.

- Wir kennen bereits die Rekursion $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.
- Berechnen Binomialkoeffizienten mittels Divide and Conquer.

Algorithmus DIV-BINOM

EINGABE: $n, k \in N_0$

1. If $(k = 0)$ return 1;
2. If $(n < k)$ return 0;
3. return DIV-BINOM$(n-1, k-1)$ + DIV-BINOM$(n-1, k)$;

AUSGABE: $\binom{n}{k}$

- **Korrektheit:** Folgt aus obiger Rekursionsformel.

Laufzeit von DIV-BINOM

Satz Laufzeit von DIV-BINOM

DIV-BINOM benötigt zur Berechnung von $\binom{2n}{n}$ mindestens $\frac{4^n}{2n+1}$ Aufrufe.

Beweis:

- DIV-BINOM bricht die Rekursion mit Werten 0 und 1 ab.
- $\binom{n}{k}$ wird als Summe der Rückgabewerte 0 und 1 berechnet.
- Damit benötigt man mindestens $\binom{n}{k}$ viele Aufrufe.
- Es gilt $\binom{2n}{n} \geq \binom{2n}{i}$ für $i = 0, \ldots, 2n$.
- Daraus folgt

$$(2n+1)\binom{2n}{n} \geq \sum_{i=0}^{2n} \binom{2n}{i} = 2^{2n} = 4^n.$$

- D.h. DIV-BINOM benötigt mindestens $\frac{4^n}{2n+1}$ Aufrufe.

Dynamische Programmierung

Nachteile des Divide and Conquer Ansatzes

- Subprobleme sind nicht unabhängig voneinander.
- Zwischenergebnisse werden nicht gespeichert.
- Damit werden die Subprobleme oft mehrfach gelöst.

Definition Paradigma Dynamische Programmierung

Dynamische Programmierung verwendet die Strategien

1. **Bottom-up** Ansatz: Man beginnt bei trivialen Subproblemen.
2. **Kombination** gespeicherter Lösungen von Subproblemen: Setze Lösungen zu Lösungen größerer Probleme zusammen.

Beispiel aus der Vorlesung

- Warschall-Algorithmus berechnet Erreichbarkeit in (V, E).
  - **Bottom-up**: $W_0[u, v] = 1$ falls $(u, v) \in E$. Sonst $W_0[u, v] = 0$.
  - **Kombination**: $W_k[u, v] = \max\{W_{k-1}[u, v], W_{k-1}[u, k] \cdot W_{k-1}[k, v]\}$.

Binomialkoeffizienten revisited

Algorithmus DYN-BINOM

EINGABE: $n, k \in \mathbb{N}_0$

1. If $(k > n)$ return 0;
2. For $i \leftarrow 0$ to n
   1. $a[i, 0] = 1$; $a[i, i] = 1$;
3. For $i \leftarrow 1$ to n
   1. For $j \leftarrow 1$ to $i - 1$
      1. $a[i, j] \leftarrow a[i - 1, j - 1] + a[i - 1, j]$;
4. return $a[n, k]$;

AUSGABE: $\binom{n}{k}$

- **Korrektheit:** Zeilenweise Berechnung des Pascal'schen Dreiecks.
- Schritt 1 initialisiert die Grenzen des Pascal'schen Dreiecks.
- Schritt 2 berechnet die i-te Zeile des Pascal'schen Dreiecks.
- **Laufzeit:** $\mathcal{O}(nk)$ für $k \leq n$, d.h. $\mathcal{O}(n^2)$.

Motivation von Optimierungsproblemen

Definition Vollständig geklammertes Matrixprodukt

Sei $M = A_1 \cdot \ldots \cdot A_n$ ein Produkt von Matrizen. M heißt *vollständig geklammertes Matrixprodukt*, falls $n = 1$ oder M ein geklammertes Produkt von zwei vollständig geklammerten Matrixprodukten ist.

Beispiel: $n = 4$

- Es existieren die 5 vollständig geklammerten Matrixprodukte $(((A_1A_2)A_3)A_4), ((A_1(A_2A_3))A_4), (A_1((A_2A_3)A_4)),$ $(A_1(A_2(A_3A_4))), ((A_1A_2)(A_3A_4))$.
- Man kann zeigen, dass die Anzahl der vollständig geklammerten Matrixprodukte exponentiell in n ist.

Kosten einer Lösung

Definition Kosten einer Matrixmultiplikation

Sei $A_1 = (a_{i,j})$ eine (n_1, n_2)-Matrix und A_2 eine $(n_2 \times n_3)$-Matrix. Dann definieren wir die Kosten der Berechnung von $A_1 \cdot A_2$ als $n_1 n_2 n_3$.

Die Kosten eines vollständigen geklammerten Matrixprodukts sind definiert als die Summe der Kosten aller Matrixprodukte.

Begründung:

- Seien $a^{(1)}_{i,j}$, $a^{(2)}_{i,j}$ die Einträge von A_1 bzw. A_2
- Sei $m_{i,j} = \sum_{k=1}^{n_2} a^{(1)}_{i,k} \cdot a^{(2)}_{k,j}$ für $i \in [n_1], j \in [n_3]$.
- Die Berechnung aller $m_{i,j}$ erfordert $n_1 \cdot n_2 \cdot n_3$ Multiplikationen.

Beispiel: $A_1 \in \mathbb{Z}^{10\times 5}, A_2 \in \mathbb{Z}^{5\times 10}$ und $A_3 \in \mathbb{Z}^{10\times 10}$

- $((A_1 A_2)A_3)$ liefert Kosten von $500 + 1000 = 1500$.
- $(A_1(A_2 A_3))$ liefert dagegen nur Kosten von $500 + 500 = 1000$.

Optimierungsprobleme

Definition Optimierungsproblem

Sei Π ein Problem mit Lösungsraum L und $c : L \to \mathbb{R}$ eine Kostenfunktion. Wir bezeichnen Π als *Optimierungsproblem*, falls eine Lösung $\ell \in L$ mit optimalen Kosten $c(\ell)$ gesucht wird.

1. Falls $c(\ell)$ maximiert wird, nennen wir Π ein *Maximierungsproblem*.
2. Falls $c(\ell)$ minimiert wird, nennen wir Π ein *Minimierungsproblem*.

Problem Minimierungsproblem Matrizenklammerung Π_M

Gegeben: $(p_i \times p_{i+1})$-Matrizen für $i \in [n]$.
Gesucht: Vollständige Klammerung mit minimalen Kosten.

Aufsplitten einer optimalen Lösung

- Sei ℓ eine optimale vollständige Klammerung für Π_M.
- Wir verwenden die Notation $A_{1\ldots n}$ für $A_1 \cdot \ldots \cdot A_n$.
- In ℓ werden $A_{1\ldots k}$ und $A_{k+1\ldots n}$ für ein $k \in [n-1]$ berechnet.

Lemma Optimalität der Subprobleme

Sei Π_M ein Matrizenklammerungs-Problem mit $M = A_1 \ldots A_n$ und ℓ eine optimale Lösung, die $A_{1\ldots k}$, $A_{k+1\ldots n}$ berechnet. Dann liefert ℓ für diese Subprobleme optimale Lösungen.

Beweis:

- **Annahme:** Sei ℓ' eine bessere Lösung für $A_{1\ldots k}$.
- Seien $c(\ell(A_{i\ldots j}))$ die Kosten der Lösung ℓ für $A_{i\ldots j}$.
- Berechne in ℓ das Produkt $A_{1\ldots k}$ gemäß ℓ' mit $c(\ell') < c(\ell(A_{1\ldots k}))$.
- Dann gilt $c(\ell') + c(\ell(A_{k+1\ldots n})) < c(\ell(A_{1\ldots k})) + c(\ell(A_{k+1\ldots n})) = c(\ell)$. (Widerspruch zur Minimalität der Lösung ℓ für Π_M)
- Beweis liefert analog die Optimalität von ℓ für $A_{k+1\ldots n}$.

Dynamische Programmierung für Π_M

Lösung mittels Dynamischer Programmierung

- Seien $m[i,j]$ die minimalen Kosten zur Berechnung von $A_{i\ldots j}$.
- **Bottom-up:** $m[i,i] = 0$ für $i = 1,\ldots,n$.
- **Kombination:** Berechnung von $A_{i\ldots j}$ für alle $i < j$.
- Bestimme k mit $i \leq k < j$, das folgende Kosten minimiert:
  - Berechne $A_{i\ldots k}$ und $A_{k+1\ldots j}$ mit Kosten $m[i,k] + m[k+1,j]$.
  - Berechne $A_{i\ldots k} \cdot A_{k+1\ldots j}$ mit Kosten $p_i p_{k+1} p_{j+1}$
- D.h. wähle $m[i,j] = \min_{i\leq k<j}\{m[i,k] + m[k+1,j] + p_i p_{k+1} p_{j+1}\}$.
- Die Gesamtkosten für $M = A_1 \cdot \ldots \cdot A_n$ sind $m[1,n]$.

Algorithmus DYN-MATRIXKLAMMERUNG

Algorithmus DYN-MATRIXKLAMMERUNG

EINGABE: $p_1, \ldots, p_{n+1} \in \mathbb{N}$

1. For $i \leftarrow 1$ to n
   1. $m[i,i] \leftarrow 0$;
2. For $\ell \leftarrow 2$ to n //ℓ ist die Länge des Intervalls $[i,j]$
   1. For $i \leftarrow 1$ to $n-(\ell-1)$
      1. $j \leftarrow i+\ell-1$;
      2. $m[i,j] \leftarrow \min_{i \le k < j}\{m[i,k]+m[k+1,j]+p_i p_{k+1} p_{j+1}\}$.
      3. $s[i,j] \leftarrow k$, an dem das Minimum angenommen wird.

AUSGABE: Arrays m und s

- **Korrektheit:** Die Intervalllängen ℓ wachsen stetig.
- Dadurch werden in Schritt 2.1.2 nur bekannte Werte verwendet.
- Bei Terminierung stehen in $m[1,n]$ die minimalen Kosten.
- Übung: s liefert die optimale Klammerung.
- **Laufzeit** für die Schleifen über ℓ, i, k : $\mathcal{O}(n^3)$.

Beispiel für ein Maximierungsproblem

Problem Maximierungsproblem Rucksack Π_R

Gegeben: n Gegenstände mit Gewichten $w_i \in \mathbb{N}$ und Profiten $p_i \in \mathbb{N}$ für $i \in [n]$. Kapazitätsschranke B

Gesucht: $\max_{J \subseteq [n]} \{\sum_{j \in J} p_j \mid \sum_{j \in J} w_j \leq B\}$, bzw. dasjenige J, an dem das Maximum angenommen wird.

Anmerkungen:

- Lösung von Rucksack mit Dynamischer Programmierung möglich.
- Liefert einen Algorithmus mit Laufzeit $\mathcal{O}(n \sum_{i \in [n]} p_i)$.
- Man beachte: Eingabelänge der p_i ist nur $\Theta(\sum_{i \in [n]} \log p_i)$.
- D.h. der Algorithmus ist exponentiell in der Eingabelänge.
- Π_R gehört zu den algorithmisch schweren Problemen (s. DiMa II).

Greedy-Strategie

Definition Paradigma Greedy

Der Greedy-Ansatz verwendet die Strategie

1. **Top-down** Auswahl: Bestimme in jedem Schritt eine lokal optimale Lösung, so dass man eine global optimale Lösung erhält.

Definition Kompatible Veranstaltungen

$S = [n]$ heißt eine Menge von *Veranstaltungen*, falls jedes $i \in [n]$ eine Startzeiten s_i und eine Endzeiten $f_i \geq s_i$ besitzt. Wir nennen $i, j \in S$ *kompatibel*, falls $[s_i, f_i)$ und $[s_j, f_j)$ nicht-überlappend sind. $J \subseteq S$ heißt *kompatibel*, falls je zwei verschiedene Elemente aus J kompatibel sind.

Problem Scheduling

Gegeben: S=[n] Menge von Veranstaltungen
Gesucht: $J \subseteq S$ kompatibel mit maximaler Größe $|J|$

Greedy Lösung für das Scheduling-Problem

Annahme: Die Veranstaltungen seien aufsteigend nach f_i sortiert.

Algorithmus GREEDY-SCHEDULING

EINGABE: $s[1 \ldots n], f[1 \ldots n]$ mit $f[1] \leq \ldots \leq f[n]$

1. $J \leftarrow \{1\}; j \leftarrow 1;$
2. For $i \leftarrow 2$ to n
   1. If $(s_i \geq f_j)$ then $J \leftarrow \{i\}; j \leftarrow i;$

AUSGABE: J

Anmerkung:

- **Korrektheit:** Wir zeigen zunächst, dass J kompatibel ist.
- Invariante: j ist ein Element maximaler Endzeit f_j in J.
- Schritt 2.1: Falls i zu j kompatibel ist, dann auch zu allen anderen Elementen in J, da $f_j \geq f_k$ für alle $k \in J$.
- **Laufzeit:** $\mathcal{O}(n)$.

Optimalität der Greedy-Lösung

Satz Optimalität von GREEDY-SCHEDULING

GREEDY-SCHEDULING liefert ein kompatibles J maximaler Größe.

Beweis: Beweisstruktur

1. Es gibt eine optimale Lösung J mit $1 \in J$.
2. $J \setminus \{1\}$ ist eine optimale Lösung für $S_1 = \{i \in S \mid s_i \geq f_1\}$.

Der Satz folgt per Induktion über die Anzahl der gewählten Elemente.

- **ad 1:** Sei J' eine optimale Lösung mit $1 \notin J'$ und k minimal in J'.
- D.h. $f_k \geq f_1$ bzw. 1 ist kompatibel zu jedem Element in $J' \setminus \{k\}$.
- Damit ist $J = J' \setminus \{k\} \cup \{1\}$ eine optimale Lösung mit $1 \in J$.
- **ad 2:** Annahme: Sei J optimal für S mit $1 \in J$. Sei J' eine bessere Lösung für S_1 als $J \setminus \{1\}$, d.h. $|J'| > |J| - 1$.
- $J' \cup \{1\}$ ist kompatibel, d.h. eine gültige Lösung für S.
- Damit folgt $|J' \cup \{1\}| = |J'| + 1 > |J|$.
  (Widerspruch zur Optimalität von J)

Eigenschaften von Greedy-Problemen

Struktur: Optimale Lösungen mit Greedy-Ansatz erfordern

1. Optimalität der Greedy-Wahl
   - Es existiert ein optimale Lösung, die das lokale Optimimum enthält.
   - Gierige Wahl hängt *nicht* von der Lösung der Subprobleme ab.
   - D.h. die Wahl hängt nur von bisherigen Entscheidungen ab. (Top-down Ansatz)
2. Optimalität der Subprobleme
   - Optimale Lösung beinhaltet optimale Lösungen der Subprobleme.

Anmerkungen:

- Nicht jeder Greedy-Ansatz liefert eine optimale Lösung.
- Nicht erfolgreich beim Scheduling sind z.B. die gierige Wahl von kürzester Dauer und kleinster Überlappung. (Übungsaufgabe)
- Das Scheduling-Problem kann auch mittels Dynamischer Programmierung gelöst werden. (Übungsaufgabe)
- Diese Lösung liefert aber eine schlechtere Laufzeit.

Greedy versus Dynamische Programmierung

Problem Rucksack Π_R

Gegeben: n Gegenstände mit Gewichten und Profiten $w_i, p_i \in \mathbb{N}$, Kapazitätsschranke B

Gesucht: $\alpha \in \{0,1\}^n$ mit Profit $\max_\alpha\{\sum_{i=1}^n \alpha_i p_i \mid \sum_{i=1}^n \alpha_i w_i \leq B\}$.

Problem Rationale Variante Rucksack Π'_R

Gegeben: n Gegenstände mit Gewichten und Profiten $w_i, p_i \in \mathbb{N}$, Kapazitätsschranke B

Gesucht: $\alpha \in [0,1]^n$ mit Profit $\max_\alpha\{\sum_{i=1}^n \alpha_i p_i \mid \sum_{i=1}^n \alpha_i w_i \leq B\}$.

Anmerkung: Beide Probleme besitzen Optimalität der Subprobleme.

- Π_R: Entscheide, ob i in optimaler Lösung L ist, d.h. ob $\alpha_i = 1$.
  - Falls $i \notin L$: Bestimme optimale Lösung des Subproblems ohne i.
  - Falls $i \in L$: Bestimme optimale Lösung ohne i mit Schranke $B - w_i$.
- Π'_R: Sei Bruchteil α_i des Gegenstands i in optimaler Lösung.
  - Bestimme optimale Lösung ohne i mit Schranke $B - \alpha_i w_i$.

(Nicht-)Optimalität der Greedy-Wahl

Algorithmus GREEDY-RATIONALER-RUCKSACK

EINGABE: n, $p_1, \ldots, p_n$, $w_1, \ldots, w_n$, B

1. Sortiere die Elemente absteigend nach $\frac{p_i}{w_i}$.
2. For $j \in [n]$ in Reihenfolge absteigender Quotienten $\frac{p_i}{w_i}$
   1. Nimm von j maximalen Bruchteil α_j, der in den Rucksack passt.

AUSGABE: $\alpha \in [0,1]^n$

- **Korrektheit:** Algorithmus liefert eine optimale Lösung. (Übung)
- **Laufzeit:** $\mathcal{O}(n \log n)$.

Anmerkung:

- Greedy-Strategie funktioniert nicht für Π_R. Gegenbeispiel: $(w_1, p_1) = (1,3)$, $(w_2, p_2) = (2,4)$, $(w_3, p_3) = (4,4)$ und $B = 6$.
- Greedy wählt die Gegenstände 1 und 2. Optimal ist aber 2 und 3.

Greedy-Algorithmus Minimaler Spannbaum (MST)

Definition Minimaler Spannbaum

Sei $G = (V, E)$ ein ungerichteter Graph und $w : E \to \mathbb{N}$. Das *Gewicht* $w(T)$ eines Spannbaums $T = (V, E_T)$ ist $\sum_{e \in E_T} w(e)$. Ein *minimaler Spannbaum* (MST) ist ein Spannbaum minimalen Gewichts.

Algorithmus KRUSKAL

EINGABE: $G = (V, E)$ mit $w : E \to \mathbb{N}$

1. $E_T \leftarrow \emptyset$
2. Sortiere die Kanten aufsteigend nach Gewicht
3. For $e \in E$ in Reihenfolge aufsteigenden Gewichts
   1. If $((V, E_T \cup \{e\})$ ist kreisfrei) then $E_T \leftarrow E_T \cup \{e\}$.

AUSGABE: MST $T = (V, E_T)$

- **Korrektheit:** T ist ein Spannbaum. Minimalität s. nächste Folie.
- **Laufzeit:** $\mathcal{O}(|E| \log |E|)$.

Optimalität von KRUSKAL

Satz MST-Eigenschaft von KRUSKAL

KRUSKAL berechnet einen minimalen Spannbaum von G.

Beweis:

- Seien $e_1, \ldots, e_m \in E$ aufsteigend nach Gewicht sortiert.
- **Zeigen:** Es existiert ein MST, der e_1 enthält.
- Sei T ein MST, der e_1 nicht enthält. Wir fügen e_1 zu T hinzu.
- e_1 schließt einen Kreis in T. Entfernen einer beliebigen Kreiskante e liefert wiederum einen Spannbaum T'.
- Wegen $w(e_1) \leq w(e')$ folgt $w(T') \leq W(T)$.

Beweis der Optimalität

Beweis: Fortsetzung

- Sei nun T ein optimaler Spannbaum mit Kante e_1.
- Entfernen von e_1 liefert zwei ZHKs $G[V_1]$ und $G[V_2]$.
- T induziert Spannbäume T_1, T_2 für $G[V_1]$ und $G[V_2]$.
- Annahme: Sei T' ein MST für $G[V_1]$ mit $w(T') < w(T)$.
- Wir ersetzen in T die Kanten von T_1 durch die Kanten von T'.
- Dies liefert einen Spannbaum von G mit Gewicht kleiner als $w(T)$. (Widerspruch zur Minimalität von T)

Lösen von Rekursionsgleichungen

Ziel: Lösen von Rekursionsgleichungen der Form

1. $T(n) = aT\left(\frac{n}{b}\right) + f(n)$
2. $x_n = a_1 x_{n-1} + \ldots + a_k x_{n-k} + b_k$

Beide Gleichungstypen treten häufig in der Laufzeitanalyse auf.

Das Master-Theorem

Satz Master-Theorem

Sei $T(n) = aT(\frac{n}{b}) + f(n)$ mit $a, b \geq 1$. Dann gilt:

1. **Fall 1:** $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$, $\epsilon > 0$:
$$T(n) = \Theta(n^{\log_b a})$$
2. **Fall 2:** $f(n) = \Theta(n^{\log_b a})$:
$$T(n) = \Theta(n^{\log_b a} \log n)$$
3. **Fall 3:** $f(n) = \Omega(n^{\log_b a + \epsilon})$ und $af(\frac{n}{b}) \leq cf(n)$ für ein $c < 1$:
$$T(n) = \Theta(f(n)).$$

Anmerkung:

- Man kann zeigen, dass $\frac{n}{b}$ sowohl als $\lfloor \frac{n}{b} \rfloor$ als auch als $\lceil \frac{n}{b} \rceil$ interpretiert werden kann, ohne die Analyse zu beeinflussen.

Beispiele bereits bekannter Anwendungen

Mergesort und FFT:

- Rekursionsgleichung $T(n) = 2T(\frac{n}{2}) + \Theta(n)$.
- D.h. wir erhalten die Parameter $a = b = 2$ und $f(n) = \Theta(n)$.
- Damit gilt $n^{\log_b a} = n$, d.h. wir sind in Fall 2: $f(n) = \Theta(n^{\log_b a})$.
- Es folgt mit Master-Theorem $T(n) = \Theta(n \log n)$.

Karatsuba:

- Rekursionsgleichung $T(n) = 3T(\frac{n}{2}) + \Theta(n)$.
- D.h. wir erhalten die Parameter $a = 3$, $b = 2$ und $f(n) = \Theta(n)$.
- Es gilt $n^{\log_b a} = n^{\log_2 3} \approx n^{1.58}$.
- Damit Fall 1: $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ für $0 < \epsilon < \log_2 3 - 1$.
- Master-Theorem liefert $T(n) = \Theta(n^{\log_2 3})$.

Weitere Beispiele

Binäre Suche:

- Rekursionsgleichung $T(n) = T(\frac{n}{2}) + \Theta(1)$.
- Wir erhalten Parameter $a = 1$, $b = 2$ und $f(n) = \Theta(1)$.
- Wegen $n^{\log_b a} = 1$ gilt Fall 2: $f(n) = \Theta(n^0) = \Theta(n^{\log_b a})$.
- Das Master-Theorem liefert $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Beispiel für Fall 3:

- Verwende Rekursionsgleichung $T(n) = 3T(\frac{n}{4}) + n \log n$.
- Es gilt $n^{\log_b a} = n^{\log_4 3} \approx n^{0.79}$.
- D.h. wir sind in Fall 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ für $0 < \epsilon \leq 1 - \log_4 3$.
- Ferner gilt $af(\frac{n}{b}) = 3\frac{n}{4} \log(\frac{n}{4}) \leq \frac{3}{4} n \log n = cf(n)$ für $c = \frac{3}{4} < 1$.
- Das Master-Theorem liefert $T(n) = \Theta(f(n)) = \Theta(n \log n)$.

Nicht-Anwendbarkeit des Master- Theorems

Beispiel:

- Betrachten die Rekursionsgleichung $T(n) = 2T(\frac{n}{2}) + n \log n$.
- Es gilt sicherlich $f(n) = \Omega(n^{\log_b a}) = \Omega(n)$, aber **nicht**
$$f(n) = \Omega(n^{\log_b a + \epsilon}).$$
- Man beachte, dass $\frac{f(n)}{n} = \log n = o(n^{\epsilon})$ für jedes $\epsilon > 0$.
- Damit ist Fall 3 nicht anwendbar.
- Fälle 1 und 2 sind nicht anwendbar, da $f(n) = \omega(n)$.
- D.h. es existieren Lücken zwischen den Fällen 1 und 2 bzw. zwischen den Fällen 2 und 3.

Beweis des Master-Theorems

Wir treffen die vereinfachte Annahme $n = b^i$.

Lemma Iterieren der Rekursionsgleichung

Sei $T(n) = aT(\frac{n}{b}) + f(n)$ und $T(1) = \Theta(1)$. Dann gilt

$$T(n) = \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(\frac{n}{b^i}).$$

Beweis:

$$\begin{aligned}
aT\left(\frac{n}{b}\right) + f(n) &= a\left(aT\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)\right) + f(n) = a^2 T(\frac{n}{b^2}) + af(\frac{n}{b}) + f(n) \\
&= a^3 T\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n) \\
&\vdots \\
&= a^{\log_b n} T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b_i}\right) \\
&= \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)
\end{aligned}$$

Interpretation des Master-Theorems

- Wir betrachten den Rekursionsbaum von $T(n)$
- Die Wurzel besitzt Kosten $f(n)$ und a Kinder mit Kosten $f\left(\frac{n}{b}\right)$.
- Jedes dieser Kinder besitzt a Kinder mit Kosten $f\left(\frac{n}{b^2}\right)$.
- D.h. in Tiefe 2 sind a^2 Kinder mit Kosten insgesamt $a^2 f\left(\frac{n}{b^2}\right)$.
- In Tiefe i sind damit allgemein a^i Kinder mit Kosten $a^i f\left(\frac{n}{b^i}\right)$.
- In den inneren Knoten entstehen Kosten beim Aufsplitten in Subprobleme und beim Zusammensetzen der Lösungen.
- Es existieren $n^{\log_b a}$ Blätter in Tiefe $i = \log_b n$.
- Die Kosten der Blätter sind Kosten zum Lösen der Subprobleme.

Master-Theorem: Gesamtkosten werden je nach Fall dominiert von

1. Kosten in den Blättern: $\Theta(n^{\log_b a})$
2. gleichverteilt über alle Ebenen: $\Theta(n^{\log_b a}) \cdot \log_b n = \Theta(n^{\log_b a} \log n)$
3. Kosten in der Wurzel: $\Theta(f(n))$

Beschränken der Summe

Lemma Summation im Master-Theorem

Sei $g(n) = \sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right)$. Dann gilt

1. **Fall 1** $f(n) = \mathcal{O}(n^{\log_b a - \epsilon}), \epsilon > 0$: $g(n) = \Theta(n^{\log_b a})$.
2. **Fall 2** $f(n) = \Theta(n^{\log_b a})$: $g(n) = \Theta(n^{\log_b a} \log n)$.
3. **Fall 3** $af\left(\frac{n}{b}\right) \leq cf(n)$ für ein $c < 1$: $g(n) = \Theta(f(n))$.

Folgerung:
Mit vorigem Lemma folgern wir für die Fälle des Master-Theorems

1. **Fall 1:** $T(n) = \Theta(n^{\log_b a}) + g(n) = \Theta(n^{\log_b a})$
2. **Fall 2:** $T(n) = \Theta(n^{\log_b a}) + g(n) = \Theta(n^{\log_b a} \log n)$
3. **Fall 3:** $T(n) = \Theta(n^{\log_b a}) + g(n) = \Theta(f(n))$
   wegen $f(n) = \Omega(n^{\log_b a + \epsilon})$.

Beweis des Lemmas

Beweis:

- Wir beweisen hier nur Fall 2 & 3, Fall 1 ist Übungsaufgabe.
- **Fall 2:** Es gilt $g(n) = \Theta\left(\sum_{i=0}^{\log_b n-1} a^i (\frac{n}{b^i})^{\log_b a}\right)$ mit

$$\sum_{i=0}^{\log_b n-1} a^i (\tfrac{n}{b^i})^{\log_b a} = n^{\log_b a} \sum_{i=0}^{\log_b n-1} \left(\frac{a}{b^{\log_b a}}\right)^i = n^{\log_b a} \cdot \log_b n.$$

- **Fall 3:** $g(n) \geq \sum_{i=0}^{0} a^i f\left(\frac{n}{b^i}\right) = \Omega(f(n))$.
- Aus $af\left(\frac{n}{b}\right) \leq cf(n)$ folgt $a^i f\left(\frac{n}{b^i}\right) \leq c^i f(n)$.
- Hieraus folgt wiederum

$$g(n) \leq \sum_{i=0}^{\log_b n-1} c^i f(n) \leq f(n) \sum_{i=0}^{\infty} c^i = f(n) \cdot \tfrac{1}{1-c} = \mathcal{O}(f(n)).$$

Lineare Rekursionsgleichungen

Definition Lineare Rekursionsgleichung

Seien $a_1, \ldots, a_{k+1}, c_0, \ldots, c_{k-1} \in \mathbb{Z}$. Die Gleichung $x_n = a_1 x_{n-1} + \ldots + a_k x_{n-k} + a_{k+1}$, $n \geq k$ mit den Anfangsbedingungen $x_i = c_i$ für $i \in [k]$ heißt *lineare Rekursionsgleichung k-ter Ordnung*. Wir bezeichnen die Rekursionsgleichung als *homogen* falls $a_{k+1} = 0$.

Beispiel:

- Sei $x_n = a x_{n-1}$ für $n \geq 1$ und $x_0 = b_0$.
- Iteration liefert $x_n = a^2 x_{n-2} = \ldots = a^n x_0 = a^n b_0$.
- Anwendung: Verzinse 1000€ mit jährlich 4%.
- Nach n Jahren besitzt man $x_n = (1,04) x_{n-1}$€ mit $x_0 = 1000$€.
- D.h. man besitzt $x_n = (1,04)^n \cdot 1000$€.

Inhomogene Rekursion 1. Ordnung

Satz Rekursion 1. Ordnung

Sei $x_n = ax_{n-1} + b_1$ und $x_0 = b_0$. Dann gilt

$$x_n = \begin{cases} a^n b_0 + b_1 \frac{a^n - 1}{a-1} & \text{falls } a \neq 1 \\ b_0 + b_1 n & \text{sonst.} \end{cases}$$

Beweis:

- Mit Iterationsmethode erhalten wir

$$\begin{aligned} x_n &= ax_{n-1} + b_1 = a(ax_{n-2} + b_1) + b_1 = a^2 x_{n-2} + b_1(1+a) \\ &= a^3 x_{n-3} + b_1(1 + a + a^2) \\ &\vdots \\ &= a^n b_0 + b_1 \sum_{i=0}^{n-1} a^i \end{aligned}$$

- Es gilt $\sum_{i=0}^{n} a^i = \begin{cases} \frac{a^n - 1}{a-1} & \text{für } a \neq 1 \\ n & \text{sonst.} \end{cases}$

Formale Potenzreihen und Erzeugende Funktion

Rekursionsgleichungen beschreiben Folgen $(x_n)_{n\geq 0} = x_0, x_1, x_2, \ldots$

Definition Potenzreihe und Erzeugende Funktion

Sei $(a_n)_{n\geq 0}$ eine Folge. Wir bezeichnen $A(x) = \sum_{n\geq 0} a_n x^n$ als *formale Potenzreihe* bzw. als *Erzeugende Funktion* der Folge.

Den Koeffizienten a_n von x^n in $A(x)$ bezeichnen wir mit $[x^n]A(x)$.
Wir definieren $a_n = 0$ für $n < 0$.

Anmerkung:

- **Ziel:** Wir suchen eine geschlossene Darstellungen für
  - $A(x)$ als Funktion von x und
  - a_n als Funktion von n.
- Mit geschlossenen Darstellungen lässt sich einfach rechnen.

Arithmetik auf Potenzreihen

Linearkombinationen von Potenzreihen:

- Seien $(a_n)_{n\geq 0}, (b_n)_{n\geq 0}$ Folgen und c, d Konstanten.
- Wir definieren $(c_n)_{n\geq 0}$ vermöge

$$c_n = ca_n + db_n \text{ für alle } n \geq 0.$$

- Es gilt für die Erzeugende Funktion

$$\begin{aligned} C(x) &= \sum_{n\geq 0}(ca_n + db_n)x^n \\ &= c\sum_{n\geq 0} a_n x^n + d\sum_{n\geq 0} b_n x^n = cA(x) + dB(x). \end{aligned}$$

Multiplikation von Potenzreihen

Multiplikation von Potenzreihen:

- Seien $A(x), B(x)$ erzeugende Funktionen. Dann gilt

$$\begin{aligned} A(x) \cdot B(x) &= \left(\sum_{n\geq 0} a_n x^n\right) \cdot \left(\sum_{n\geq 0} b_n x^n\right) \\ &= \sum_{n\geq 0} \left(\sum_{k\geq 0}^{n} a_k b_{n-k}\right) x^n. \end{aligned}$$

- Dies entspricht der Folge $(c_n)_{n\geq 0}$ mit $c_n = \sum_{k=0}^{n} a_k b_{n-k}$.
- $(c_n)_{n\geq 0}$ heißt Faltung oder Konvolution von $(a_n)_{n\geq 0}$ und $(b_n)_{n\geq 0}$.

Spezialfälle der Multiplikation:

- Sei $B(x) = x^m$. Erhalten $c_n = a_{n-m}$, d.h. $c_m = a_0, c_{m+1} = a_1, \ldots$
- Liefert eine Verschiebung der Folge $(a_n)_{n\geq 0}$ um m nach rechts.
- Dabei füllen wir die Folge von links mit Nullen auf.
- Mit $B(x) = \sum_{n\geq 0} x^n$ gilt $c_n = \sum_{k=0}^{n} a_k$. (Kumulative Summe)
- Dies liefert die Summe der ersten n Folgenglieder von $(a_n)_{n\geq 0}$.

Die geometrische Reihe

Definition Geometrische Reihe

Die erzeugende Funktion der Einserfolge 1,1,1,... heißt *geometrische Reihe* $G(x) = \sum_{n\geq 0} x^n$.

Anmerkung: Zusammenhang zur endlichen geometrischen Reihe.

- Wir subtrahieren von der Einserfolge die um m nach rechts verschobene Einserfolge

$$\begin{aligned} G(x) - x^m G(x) &= \sum_{n\geq 0} x^n - \sum_{n\geq 0} x^{n+m} \\ &= \sum_{n\geq 0} x^n - \sum_{n\geq m} x^n = \sum_{n=0}^{m-1} x^n. \end{aligned}$$

- Entspricht der Folge $1, 1, \ldots, 1, 0, 0, \ldots$ mit genau m Einsen.
- $\sum_{n=0}^{m-1} x^n$ ist die endliche geometrische Reihe mit m Elementen.

Einfache Operationen auf Potenzreihen

Operationen auf Potenzreihen:

- **Verschieben** nach links: $a_0, a_1, a_2, \ldots \mapsto a_m, a_{m+1}, a_{m+2}, \ldots$
- Realisierung als formale Potenzreihe

$$\frac{A(x)-a_0-a_1x-\ldots-a_{m-1}x^{m-1}}{x^m} = \sum_{n\geq m} a_n x^{n-m} = \sum_{n\geq 0} a_{n+m}x^n.$$

- **Alternieren**: $a_0, a_1, a_2, a_3, \ldots \mapsto a_0, -a_1, a_2, -a_3, \ldots$
- Realisierung als formale Potenzreihe

$$A(-x) = \sum_{n\geq 0} a_n(-x)^n = \sum_{n\geq 0}(-1)^n a_n x^n.$$

- **Index-Koeffizienten:** $a_0, a_1, a_2, a_3, \ldots \mapsto a_1, 2a_2, 3a_3, 4a_4, \ldots$
- Realisierung als formale Potenzreihe

$$\frac{d}{dx}A(x) = \sum_{n\geq 1} n a_n x^{n-1} = \sum_{n\geq 0}(n+1)a_{n+1}x^n.$$

Invertieren von Potenzreihen

- Sei $E(x)$ die Erzeugende Funktion der Reihe $1, 0, 0, 0, \ldots$.
- $E(x)$ ist neutrales Element der Multiplikation von Potenzreihen.

Definition Inverses einer Potenzreihe

Sei $A(x), B(x)$ formale Potenzreihen. $A(x)$ ist *invers* zu $B(x)$ falls $A(x)B(x) = E(x)$.

Satz Existenz von Inversen

Sei K ein Körper. Sei $A(x)$ die Erzeugende Funktion von $(a_n)_{n\geq 0}$ mit $a_n \in K$. Das Inverse von $A(x)$ existiert gdw $a_0 \neq 0$.

Beweis:

- „ $\Rightarrow$ “ : Sei $B(x)$ invers zu $A(x)$.
- Dann gilt $[x_0]A(x)B(x) = a_0 b_0 = 1$, d.h. $a_0 \neq 0$.

Geometrische Reihe

Beweis: Fortsetzung

- „ $\Leftarrow$ “ : Wir zeigen die Existenz von b_n per Induktion über n.
- **IA** für $n = 0$: $b_0 = \frac{1}{a_0}$ existiert wegen $a_0 \neq 0$.
- **IS** $n-1 \to n$: Wir benötigen $\sum_{k=0}^{n} a_k b_{n-k} = 0$.
- Damit gilt $b_n = -\frac{1}{a_0} \cdot \sum_{k=1}^{n} a_k b_{n-k}$.

Anwendung:

- Suchen geschlossene Form der geometrischen Reihe $1, 1, 1, \ldots$
- Dazu bestimmen wir das Inverse $B(x)$ von $G(x) = \sum_{n \geq 0} x^n$.
- Es gilt $b_0 = \frac{1}{g_0} = 1$. Ferner ist $b_n = -\sum_{k=1}^{n} b_{n-k} = -\sum_{k=0}^{n-1} b_k$.
- Dies liefert $b_1 = (-1)$ und $b_2 = b_3 = \ldots = 0$.
- Damit folgt $(1-x)G(x) = 1$.
- Wir erhalten die bekannte geschlossene Form $G(x) = \frac{1}{1-x}$.
- **Warnung:** Wir vernachlässigen hier den sog. Konvergenzradius.

Weitere geschlossene Formen

Geschlossene Formen:

- Endliche geometrische Reihe
$$\sum_{n=0}^{m-1} x^n = G(x) - x^m G(x) = \frac{1}{1-x} - x^m \frac{1}{1-x} = \frac{1-x^m}{1-x}.$$
- Reihe $1, 2, 3, 4, \ldots$
$$B(x) = \sum_{n\geq 1} n x^{n-1} = \frac{d}{dx} \sum_{n\geq 0} x^n = \frac{d}{dx} G(x), \text{ d.h. } B(x) = \frac{1}{(1-x)^2}.$$

Verschiedene Herleitungen der geschlossenen Form von $1, 0, 1, 0, \ldots$

1. Mittels Erzeugende Funktion $B(x) = \sum_{n\geq 0} x^{2n}$.
   - Wir substitutieren $x \mapsto x^2$ in der geometrischen Reihe.
   - Dies liefert $B(x) = \frac{1}{1-x^2}$.
2. Kumulative Summe der Folge $1, -1, 1, -1$ liefert $1, 0, 1, 0, \ldots$
   - $1, -1, 1, -1, \ldots$ besitzt die Erzeugende Funktion $G(-x) = \frac{1}{1+x}$.
   - D.h. $B(x) = G(x)G(-x) = \frac{1}{1+x} \cdot \frac{1}{1-x} = \frac{1}{1-x^2}$
3. Addition von $1, -1, 1, -1, \ldots$ mit $1, 1, 1, 1, \ldots$ liefert $2, 0, 2, 0, \ldots$
   - $1, -1, 1, -1, \ldots$ besitzt die Erzeugende Funktion $G(-x) = \frac{1}{1+x}$.
   - D.h. $B(x) = \frac{1}{2}\left(\frac{1}{1+x} + \frac{1}{1-x}\right) = \frac{1}{1-x^2}$.

Polyas Geldwechsel

Definition Polyas Geldwechselproblem

Gegeben: Betrag n Cent, Münzen $1, 5, 10$ Cent
Gesucht: #(Möglichkeiten), n mit den Münzen zu zahlen

Lösungsansatz:

- Sei a_n die Anzahl Möglichkeiten, n mit 1-Cent Münzen zu zahlen.
- Wir erhalten die Folge $(a_n)_{n\geq 0} = 1, 1, 1, 1, \ldots$
- Erzeugende Funktion von $(a_n)_{n\geq 0}$ ist $A(x) = \sum_{n\geq 0} x^n = \frac{1}{1-x}$.
- Sei b_n die Anzahl Möglichkeiten, n mit 5-Cent Münzen zu zahlen.
- Dann gilt $(b_n)_{n\geq 0} = 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, \ldots$.
- Die Erzeugende Funktion ist $B(x) = \sum_{n\geq 0} x^n = \frac{1}{1-x^5}$.
- Analog definieren wir $C(x) = \frac{1}{1-x^{10}}$ für 10-Cent Münzen.

Divide and Conquer Lösung

Divide and Conquer für Polyas Geldwechselproblem:

- Betrachten zunächst nur Zahlungen mit 1- und 5-Cent Münzen.
- Für $k = 0, \ldots, n$ zahlen wir k mit 1 Cent und $(n-k)$ mit 5 Cent.
- Dies liefert $\sum_{k=0}^{n} a_k b_{n-k}$, d.h. die Faltung von $(a_n)_{n\geq 0}$ und $(b_n)_{n\geq 0}$.
- Die Faltung können wir mittels Produkt $A(x) \cdot B(x)$ berechnen.
- $[x^n]A(x)B(x)$ liefert die Anzahl der Möglichkeiten, den Betrag n mit 1-Cent und 5-Cent Münzen zu zahlen.
- Nehmen wir noch 10-Cent hinzu, so erhalten wir
$$A(x)B(x)C(x) = \frac{1}{1-x} \cdot \frac{1}{1-x^5} \cdot \frac{1}{1-x^{10}}.$$
- $[x^n]A(x)B(x)C(x)$ ist die Lösung von Polyas Geldwechselproblem.

Ziel: Geschlossene Form für $[x^n]A(x)B(x)C(x)$ als Funktion von n.

Beispiel für eine geschlossene Form

Satz Lineare Rekursion

Sei $a_n = a_{n-1} + 1$ für $n \geq 1$, $a_0 = 1$. Dann gilt $a_n = n + 1$ für alle $n \geq 0$.

Beweis:

- Wir stellen die Erzeugenden Funktion $A(x)$ geschlossen dar als

$$\begin{aligned} A(x) &= \sum_{n\geq 0} a_n x^n = a_0 + \sum_{n\geq 1} a_n x^n = a_0 + \sum_{n\geq 1} (a_{n-1} + 1) x^n \\ &= 1 + x \sum_{n\geq 1} a_{n-1} x^{n-1} + \sum_{n\geq 1} x^n \\ &= x \sum_{n\geq 0} a_n x^n + \sum_{n\geq 0} x^n = x \cdot A(x) + \frac{1}{1-x}. \end{aligned}$$

- Auflösen nach $A(x)$ liefert $A(x) = \frac{1}{(1-x)^2}$.
- Wir kennen bereits die Reihenentwicklung $\frac{1}{(1-x)^2} = \sum_{n\geq 1} n x^{n-1}$.
- Einsetzen: $\sum_{n\geq 0} a_n x^n = \frac{1}{(1-x)^2} = \sum_{n\geq 1} n x^{n-1} = \sum_{n\geq 0} (n+1) x^n$.
- Koeffizientenvergleich liefert geschlossene Form $a_n = n + 1$.

Strategie für geschlossene Form

Strategie zum Finden einer geschlossenen Form

1. Aufstellen der Erzeugenden Funktion $A(x) = \sum_{n \geq 0} a_n x^n$.
2. Einsetzen von Anfangswerten und Rekursionsgleichung.
3. Darstellung aller a_n durch $A(x)$.
4. Auflösen nach $A(x)$ liefert eine geschlossene Form $f(x)$.
5. Entwicklung von $f(x) = \sum_{n \geq 0} f_n x^n$ als formale Potenzreihe. Wir verwenden hier als Hilfsmittel die Partialbruchzerlegung.
6. Koeffizientenvergleich liefert geschlossene Form $a_n = f_n$.

Ableiten von $G(x)$

Lemma Partialbruchlemma

Für alle $a \in \mathbb{R}$, $k \in \mathbb{N}$ gilt $\frac{1}{(1-ax)^k} = \sum_{n\geq 0} \binom{n+k-1}{k-1} a^n x^n$.

Beweis:

- Ableiten der geometrischen Reihe liefert $\sum_{n\geq 1} nx^{n-1} = \frac{1}{(1-x)^2}$.
- Erneutes Ableiten führt zu $\sum_{n\geq 2} n(n-1)x^{n-2} = \frac{2}{(1-x)^3}$.
- k-maliges Ableiten ergibt $\sum_{n\geq k} n \cdot \ldots \cdot (n-k+1)x^{n-k} = \frac{k!}{(1-x)^{k+1}}$.
- Daraus folgt $\sum_{n\geq 0} \binom{n+k}{k} x^n = \frac{1}{(1-x)^{k+1}}$.
- Wir erhalten $\frac{1}{(1-ax)^k} = \sum_{n\geq 0} \binom{n+k-1}{k-1} a^n x^n$.

Partialbruchzerlegung

Satz Partialbruchzerlegung

Seien $f, g \in \mathbb{R}[x]$ mit $f = (1 - a_1 x)^{k_1} \cdot \ldots \cdot (1 - a_r x)^{k_r}$ und $\operatorname{grad}(g) < \operatorname{grad}(f)$. Dann existieren $g_i(x), i \in [r]$ mit $\operatorname{grad}(g_i) < k_i$ und

$$\frac{g(x)}{f(x)} = \frac{g_1(x)}{(1-a_1 x)^{k_1}} + \ldots + \frac{g_r(x)}{(1-a_r x)^{k_r}}.$$

Beweis:

- Wir suchen g_i der Form $g(x) = \sum_{i=1}^{r} g_i(x) \prod_{j \in [r] \setminus \{i\}} (1 - a_j x)^{k_j}$.
- $\operatorname{grad}(g_i) < k_i$, d.h. jedes g_i besitzt höchstens k_i Koeffizienten.
- Insgesamt gibt es $\sum_{i=1}^{r} k_i = \operatorname{grad}(f)$ viele Koeffizienten der g_i.
- Durch Ausmultiplizieren und Koeffizientenvergleich erhalten wir $\operatorname{grad}(f)$ viele Gleichungen für unsere $\operatorname{grad}(f)$ viele Unbekannte.

Bsp. Partialbruchzerlegung

Beispiel: Partialbruchzerlegung

- $g(x) = x, f(x) = x^2 - 1$. Dies liefert den Ansatz
$$\frac{x}{x^2-1} = \frac{a}{x+1} + \frac{b}{x-1}.$$
- Multiplikation mit $f(x)$ führt zu
$$x = a(x-1) + b(x+1) = (a+b)x - a + b.$$
- Koeffizientenvergleich ergibt
$$\left| \begin{array}{rcl} a+b & = & 1 \\ -a+b & = & 0 \end{array} \right|.$$
- Damit erhalten wir $a = b = \frac{1}{2}$, d.h.
$$\frac{x}{x^2-1} = \frac{1}{2}\left(\frac{1}{x+1} + \frac{1}{x-1}\right) = \frac{1}{2}\left(\frac{1}{1-(-x)} + \frac{1}{1-x}\right) = \frac{1}{2}(G(-x) + G(x)).$$

Reflektiertes Polynom

Definition Reflektiertes Polynom

Sei $f(x) = f_0 + f_1 x + \ldots + f_n x^n \in \mathbb{R}[x]$. Dann heißt
$f^R(x) = f_n + f_{n-1} x + \ldots f_0 x^n$ das *reflektierte Polynom* von f.

- Es gilt $f^R(x) = x^n \cdot f(\frac{1}{x})$. Daraus folgt $f^R(\frac{1}{x}) = x^{-n} \cdot f(x)$.

Lemma Reflexionslemma

Sei $f \in \mathbb{R}[x]$ mit $f^R(x) = (x - a_1) \cdot \ldots \cdot (x - a_n)$. Dann gilt
$f(x) = (1 - a_1 x) \cdot \ldots \cdot (1 - a_n x)$.

Beweis:

- Es gilt
  $f(x) = x^n f^R(\frac{1}{x}) = x(\frac{1}{x} - a_1) \cdot \ldots \cdot x(\frac{1}{x} - a_n) = (1 - a_1 x) \cdot \ldots \cdot (1 - a_n x)$.

Partialbruchzerlegung

Beispiel: Partialbruchzerlegung

- Seien $g(x) = x$ und $f(x) = 1 - x - x^2$.
- $f^R(x) = x^2 - x - 1$ besitzt die beiden Nullstellen $\frac{1}{2} \pm \sqrt{\frac{1}{4} + 1}$, d.h.

$$\phi = \tfrac{1+\sqrt{5}}{2} \text{ und } \bar{\phi} = \tfrac{1-\sqrt{5}}{2}.$$

- Damit gilt $f(x) = (1 - \phi x)(1 - \bar{\phi} x)$.
- Wir erhalten den Partialbruchansatz $\frac{x}{(1-\phi x)(1-\bar{\phi} x)} = \frac{a}{1-\phi x} + \frac{b}{1-\bar{\phi} x}$.
- Multiplikation mit f liefert $x = (1 - \bar{\phi} x)a + (1 - \phi x)b$.
- Koeffizientenvergleich ergibt $\left| \begin{array}{rcl} -\bar{\phi} a - \phi b & = & 1 \\ a + b & = & 0 \end{array} \right|$.
- Dies impliziert $a = \frac{1}{\phi - \bar{\phi}} = \frac{1}{\sqrt{5}} = -b$.

Fibonacci-Rekursion

Satz Formel für Fibonacci-Zahlen

Sei $F_0 = 0, F_1 = 1$ und $F_n = F_{n-1} + F_{n-2}$ für $n \geq 2$. Dann gilt

$$F_n = \frac{1}{\sqrt{5}} \left(\phi^n - \bar{\phi}^n \right).$$

Beweis:

- Einsetzen von $F(x) = \sum_{n \geq 0} F_n x^n$ in die Rekursionsgleichung
  $F(x) = F_0 + F_1 x + \sum_{n \geq 2} F_n x^n = x + \sum_{n \geq 2} (F_{n-1} + F_{n-2}) x^n.$
- Wir stellen die Summen wieder durch $F(x)$ dar.
  $$\begin{aligned} F(x) &= x + x \sum_{n \geq 1} F_n x^n + x^2 \sum_{n \geq 0} F_n x^n \\ &= x + x(F(x) - F_0) + x^2 F(x) = x + F(x)(x + x^2). \end{aligned}$$
- Auflösen nach $F(x)$ ergibt $F(x) = \frac{x}{1 - x - x^2}$.
- Partialbruchzerlegung $F(x) = \frac{1}{1 - \phi x} + \frac{b}{1 - \bar{\phi} x}$ mit $a = (-b) = \frac{1}{\sqrt{5}}$.
- Partialbruchlemma liefert $F(x) = a \sum_{n \geq 0} (\phi x)^n + b \sum_{n \geq 0} (\bar{\phi} x)^n$.
- Durch Koeffizientenvergleich erhalten wir $F_n = \frac{1}{\sqrt{5}} \left(\phi^n - \bar{\phi}^n \right)$.

Wahrscheinlichkeitsraum

Definition Wahrscheinlichkeitsraum

Seien $\omega_1, \omega_2, \ldots$ Elementareignisse mit Wahrscheinlichkeiten $0 \leq \Pr[\omega_1], \Pr[\omega_2], \ldots \leq 1$. Wir bezeichnen $\Omega = \{\omega_1, \omega_2, \ldots\}$ als *Ergebnismenge*. Ω definiert einen diskreten *Wahrscheinlichkeitsraum* falls $\sum_{\omega \in \Omega} \Pr[\omega] = 1$.

Eine Teilmenge $E \subseteq \Omega$ heißt Ereignis mit $\Pr[E] := \sum_{\omega \in E} \Pr[\omega]$.

Beispiel: Fairer Würfel

- $\omega_i = i, i \in [6]$ bezeichnen die Elementarereignis, i zu würfeln.
- Der Ergebnisraum ist $\Omega = \{1, 2, 3, 4, 5, 6\}$.
- Bei einem fairen Würfel gilt $\Pr[i] = \frac{1}{6}$ für alle $i \in [6]$.
- D.h. $\sum_{i \in \Omega} \Pr[i] = 1$. Damit definiert Ω einen Wsraum.
- Sei $E = \{3, 6\}$, d.h. es wird eine durch 3 teilbare Zahl gewürfelt.
- Dann gilt $\Pr[E] = \Pr[3] + \Pr[6] = \frac{1}{3}$.

Beispiel Wsraum

Beispiel: Wsraum

- Wir modellieren 2 Kartenspieler mit je 10 aus 52 Karten.
- Karten $K = \{\text{Karo, Herz, Pik, Kreuz}\} \times \{2, \ldots, 10, B, D, K, A\}$.
- Ergebnismenge $\Omega = \{(X, Y) \subseteq K^2 \mid X \cap Y = \emptyset, |X| = |Y| = 10\}$.
- Elementareignisse $(X, Y) \in \Omega$ entsprechen Kartenverteilungen.
- Es gilt $\Pr(\omega) = \frac{1}{|\Omega|}$ für alle $\omega \in \Omega$. (Übung: Bestimmen Sie $|\Omega|$.)
- Das Ereignis, das Spieler X vier Asse besitzt, ist

  $$E = \{(X, Y) \in \Omega \mid \{(\text{Karo,A}),(\text{Herz,A}), (\text{Pik,A}), (\text{Kreuz,A})\} \subset X\}.$$
- Für bessere Lesbarkeit schreiben wir oft $E =$“Spieler X besitzt 4 Asse.” und analog $\Pr[E] = \Pr[\text{“Spieler X besitzt vier Asse”}]$.

Unendlicher Wsraum

Problem Laufzeit von probabilistischen Las-Vegas Algorithmen

Gegeben: Algorithmus, der in jeder Iteration eine Ausgabe mit Ws von $p \in (0,1)$ liefert.

Gesucht: Ws, dass genau i Iterationen durchgeführt werden.

Modellierung als unendlicher Wsraum

- Sei w_i, $i \in \mathbb{N}$ das Elementarereignis, das genau i Iterationen des Algorithmus durchgeführt werden.
- Die Ergebnismenge $\Omega = \{\omega_1, \omega_2, \ldots\}$ ist unendlich.
- Für ω_i benötigt man zunächst $i-1$ Misserfolge, dann Erfolg.
- D.h. $\Pr[\omega_i] = (1-p)^{i-1}p$. Damit definiert Ω einen Wsraum, denn $\sum_{\omega\in\Omega} \Pr[\omega] = \sum_{i\geq 1}(1-p)^{i-1}p = p\sum_{i\geq 1}(1-p)^i = p\cdot\frac{1}{1-(1-p)} = 1$.

Additionslemma und Gegenereignis

Lemma Additionslemma

Sei Ω ein Wsraum. Für paarweise disjunkte $A_1, \ldots, A_n \subseteq \Omega$ gilt $\Pr[\bigcup_{i=1}^{n} A_i] = \sum_{i=1}^{n} \Pr[A_i]$.

Beweis:

- $\Pr[\bigcup_{i=1}^{n} A_i] = \sum_{a \in A_1} \Pr[a] + \ldots + \sum_{a \in A_n} \Pr[a] = \sum_{i=1}^{n} \Pr[A_i]$.

Lemma Gegenereignis

Sei Ω ein Wsraum. Sei $A \subseteq \Omega$ mit Gegenereignis $\bar{A} = \Omega \setminus A$. Dann gilt $\Pr[\bar{A}] = 1 - \Pr[A]$.

Beweis:

- $\Pr[A] + \Pr[\bar{A}] = \Pr[A \cup \bar{A}] = \Pr[\Omega] = 1$.
- Daraus folgt $\Pr[\bar{A}] = 1 - \Pr[A]$.

Teilereignisse

Lemma Teilereignis

Sei Ω ein Wsraum und $A, B \subseteq \Omega$ mit $A \subseteq B$. Dann gilt $\Pr[A] \leq \Pr[B]$.

Beweis:

- $\Pr[B] = \Pr[A \cup (B \cap \bar{A})] = \Pr[A] + \Pr[B \cap \bar{A}] \geq \Pr[A]$.

Inklusion/Exklusion für nicht-disjunkte Ereignisse

Satz Additionsformel

Sei Ω ein Wsraum mit Ereignissen $A_1, \ldots, A_n$. Dann gilt

$$\Pr[\bigcup_{i=1}^{n} A_i] = \sum_{i=1}^{n} \Pr[A_i] - \sum_{1 \le i_1 < i_2 \le n} \Pr[A_{i_1} \cap A_{i_2}] + \ldots + (-1)^{n-1} \Pr[A_1 \cap \ldots A_n].$$

Beweis:

- Wir zeigen nur $n = 2$, für allg. n folgt der Beweis per Induktion.
- D.h. zu zeigen ist $\Pr[A_1 \cup A_2] = \Pr[A_1] + \Pr[A_2] - \Pr[A_1 \cap A_2]$.
- Sei $B = A_1 \setminus A_2$. Dann sind B und $A_1 \cap A_2$ disjunkt.
- Damit gilt $\Pr[A_1] = \Pr[B \cup (A_1 \cap A_2)] = \Pr[B] + \Pr[A_1 \cap A_2]$.
- Es folgt

$$\begin{aligned} \Pr[A_1 \cup A_2] &= \Pr[B \cup A_2] = \Pr[B] + \Pr[A_2] \\ &= \Pr[A_1] - \Pr[A_1 \cap A_2] + \Pr[A_2]. \end{aligned}$$

Boolesche Ungleichung

Satz Boolesche Ungleichung

Sei Ω ein Wsraum mit Ereignissen $A_1, \ldots, A_n$. Dann gilt

$$\Pr[\bigcup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr[A_i].$$

Beweis:

- Sei $B = \bigcup_{i=1}^{n} A_i$. Dann folgt

$$\Pr[B] = \sum_{\omega \in B} \Pr[\omega] \leq \sum_{i=1}^{n} \sum_{\omega \in A_i} \Pr[\omega] = \sum_{i=1}^{n} \Pr[A_i].$$

Prinzip von Laplace

Definition Prinzip von Laplace

Sei Ω eine Ergebnismenge. Beim *Prinzip von Laplace* setzen wir $\Pr[\omega] = \frac{1}{|\Omega|}$ für alle $\omega \in \Omega$.

Anmerkung:

- Das Prinzip von Laplace liefert eine Gleichverteilung.
- Für alle $E \subseteq \Omega$ gilt $\Pr(E) = \sum_{\omega \in E} \Pr(\omega) = \sum_{\omega \in E} \frac{1}{|E|} = \frac{|E|}{|\Omega|}$.
- D.h. wir erhalten die Faustformel “Günstige durch Mögliche”.

Bedingte Wahrscheinlichkeit

Definition Bedingte Wahrscheinlichkeit

Sei Ω ein Wsraum mit $A, B \subseteq \Omega$ und $\Pr[B] > 0$. Dann definieren wir die *bedingte Wahrscheinlichkeit* $\Pr[A|B] := \frac{\Pr[A \cap B]}{\Pr[B]}$.

Anmerkungen:

- $\Pr[A|B]$ bezeichnet die Wahrscheinlichkeit, dass Ereignis A eintrifft unter der Bedingung dass Ereignis B eintrifft.
- Es folgt $\Pr[A \cap B] = \Pr[A|B] \cdot \Pr[B] = \Pr[B|A] \cdot \Pr[A]$.
- Ferner gilt $\Pr[A|A] = 1$ und $\Pr[A|\bar{A}] = 0$.
- Es gilt $\Pr[\omega|B] = \begin{cases} 0 & \text{für } \omega \notin B \\ \frac{\Pr[\omega]}{\Pr[B]} & \text{für } \omega \in B \end{cases}$.
- D.h. für $\omega \in B$ wird der Wsraum mit dem Faktor $\frac{1}{\Pr[B]}$ skaliert.
- Dies liefert einen Wsraum, denn
$$\sum_{\omega \in \Omega} \Pr[\omega|B] = \sum_{\omega \in \Omega} \frac{\Pr[\omega \cap B]}{\Pr[B]} = \frac{1}{\Pr[B]} \sum_{\omega \in B} \Pr[\omega] = \frac{\Pr[B]}{\Pr[B]} = 1.$$

Beispiel bedingte Wahrscheinlichkeiten

Beispiel:

- Wir betrachten einen Laplace-Würfel mit Wsraum $\Omega = [6]$.
- Sei $A =$"Augenzahl ist durch 3 teilbar".
- Sei $B =$"Augenzahl ist größer als 2".
- $\Pr[A \cap B] = \frac{|\{3,6\}|}{|\Omega|} = \frac{1}{3}$ und $\Pr[B] = \frac{|\{3,4,5,6\}|}{|\Omega|} = \frac{2}{3}$.
- Damit folgt $\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]} = \frac{1}{3} \cdot \frac{3}{2} = \frac{1}{2}$.
- Der Skalierungsfaktor $\frac{1}{\Pr[B]}$ ist $\frac{3}{2}$.
- Alternativ können wir $\Pr[A|B]$ wie folgt bestimmen.
- Falls B eintritt, verändert dies den Wsraum zu $\Omega' = \{3, 4, 5, 6\}$.
- Damit gilt $\Pr[A|B] = \frac{|\{3,6\}|}{|\Omega'|} = \frac{1}{2}$.

- Wir betrachten ein weiteres Beispiel aus der Kryptographie.
- Perfekte Sicherheit wird in der Kryptographie definiert als

  $$\Pr[\text{Klartext ist } p] = \Pr[\text{Klartext ist } p \mid \text{Chiffretext ist } c].$$
- D.h. c liefert keine Information über das zugrundeliegende p.

Das Zweikinderproblem

Definition Zweikinderproblem

Eine Familie besitzt zwei Kinder. Wie groß ist die Wahrscheinlichkeit Pr[“Beide Kinder sind Mädchen.”|“Eines der Kinder ist ein Mädchen”]?

Lösung:

- Sei $A =$“Beide Kinder sind Mädchen.”
- Sei $B =$“Eines der Kinder ist ein Mädchen.”
- Wir betrachten den Wsraum $\Omega = \{MM, JM, MJ, JJ\}$.
- Wir treffen die Laplace Annahme, dass $\Pr[\omega] = \frac{1}{4}$ für alle $\omega \in \Omega$.
- Es gilt $\Pr[A \cap B] = \Pr[A] = \frac{|\{MM\}|}{|\Omega|} = \frac{1}{4}$.
- Wegen $\Pr[B] = \frac{|\{MM,JM,MJ\}|}{|\Omega|} = \frac{3}{4}$ folgt $\Pr[A|B] = \frac{1}{3}$.
- D.h. das zweite Kind ist ein Mädchen mit Ws $\frac{1}{3}$.

- **Vorsicht:** Für $B' =$ “Das ältere Kind ist ein Mädchen” gilt $\Pr[B'] = \frac{1}{2}$ und damit $\Pr[A|B'] = \frac{1}{2}$.
- Analog erhalten wir $\Pr[A|B] = \frac{1}{2}$ mit $\Omega' = \{MM, JM, JJ\}$.

Multiplikationssatz

Satz Multiplikationssatz

Seien $A_1, \ldots, A_n$ Ereignisse mit $\Pr[A_1 \cap \ldots \cap A_n] > 0$. Dann gilt
$\Pr[A_1 \cap \ldots \cap A_n] = \Pr[A_1] \cdot \Pr[A_2|A_1] \cdot \ldots \Pr[A_n|A_1 \cap \ldots \cap A_{n-1}]$.

Beweis:

- Es gilt $0 < \Pr[A_1 \cap \ldots \cap A_n] \leq \Pr[A_1 \cap \ldots \cap A_{n-1}] \leq \ldots \leq \Pr[A_1]$.
- n-malige Anwendung der Definition für bedingte Ws führt zu

$$\Pr[A_1] \cdot \Pr[A_2|A_1] \cdot \Pr[A_3|A_1 \cap A_2] \cdot \ldots \cdot \Pr[A_n|A_1 \cap \ldots \cap A_{n-1}]$$

$$= \frac{\Pr[A_1]}{1} \cdot \frac{\Pr[A_1 \cap A_2]}{\Pr[A_1]} \cdot \frac{\Pr[A_1 \cap A_2 \cap A_3]}{\Pr[A_1 \cap A_2]} \cdot \ldots \cdot \frac{\Pr[A_1 \cap \ldots \cap A_n]}{\Pr[A_1 \cap \ldots \cap A_{n-1}]}.$$

- Kürzen liefert $\Pr[A_1 \cap \ldots \cap A_n]$.

Geburtstagsproblem

Definition Geburtstagsproblem

Gegeben: m Personen
Gesucht: Ws, dass ≥ 2 Personen denselben Geburtstag haben

Lösung:

- Wir werfen nacheinander m Bälle in $n = 365$ Urnen.
- Definieren $\bar{A}$ =“In einer Urne liegen mindestens 2 Bälle.”
- Dann gilt A =“Alle Bälle liegen allein in einer Urne.”
- Sei A_i =“Ball i liegt allein in einer Urne.”
- Dann gilt $\Pr[A] = \Pr[A_1 \cap \ldots \cap A_m]$. Der Multiplikationssatz liefert
$$\Pr[A] = \Pr[A_1] \cdot \Pr[A_2|A_1] \cdot \ldots \Pr[A_n|A_1 \cap \ldots \cap A_{n-1}].$$
- Für $j \in [m]$ gilt $\Pr[A_j|A_1 \cap \ldots \cap A_{j-1}] = \frac{n-(j-1)}{n} = 1 - \frac{j-1}{n}$, da der j-te Ball in einer der $n - (j-1)$ freien Urnen landen muss.

Geburtstagsproblem: Abschätzen der Ws

Lösung: Fortsetzung

- Wir erhalten $\Pr[A] = \prod_{j=1}^{m} \Pr[A_j | A_1 \cap \ldots \cap A_{j-1}] = \prod_{j=1}^{m} \left(1 - \frac{j-1}{n}\right)$.
- Die Abschätzung $1 - x \leq e^{-x}$ liefert

$$\Pr[A] \leq \prod_{j=1}^{m} e^{-\frac{j-1}{n}} = e^{-\frac{1}{n}\sum_{j=0}^{m-1} j} = e^{-\frac{m(m-1)}{2n}}.$$

- Damit erhalten wir $\Pr[\bar{A}] = 1 - \Pr[A] \geq 1 - e^{-\frac{m(m-1)}{2n}}$.
- D.h. wir erhalten eine konstante Ws $\Pr[\bar{A}]$ für $m = \Theta(\sqrt{n})$.

Anwendung: Kryptographische Hashfunktion $H : \{0,1\}^* \to \{0,1\}^n$.

- Wir werten H für verschiedene Urbilder $x_1, \ldots, x_m$ aus.
- Benötigen $m = \Theta(\sqrt{n})$ für eine Kollision $x_i \neq x_j$ mit $H(x_i) = H(x_j)$.

Satz von der totalen Ws

Satz von der totalen Wahrscheinlichkeit

Sei Ω ein Wsraum mit paarweise disjunkten $A_1, \ldots, A_n \in \Omega$ und $B \subseteq A_1 \cup \ldots \cup A_n$. Dann gilt

$$\Pr[B] = \sum_{i=1}^{n} \Pr[B|A_i] \cdot \Pr[A_i].$$

Beweis:

- Aus $B \subseteq A_1 \cup \ldots \cup A_n$ folgt $B = (B \cap A_1) \cup \ldots \cup (B \cap A_n)$.
- A_i sind paarweise disjunkt, d.h. $B \cap A_i$ sind paarweise disjunkt.
- Aus dem Additionssatz folgt

$$\Pr[B] = \Pr[B \cap A_1] + \ldots + \Pr[B \cap A_n] = \Pr[B] \cdot \Pr[B|A_1] + \ldots + \Pr[B] \cdot \Pr[B|A_n].$$

Korollar zum Satz der totalen Wahrscheinlichkeit

Für alle $A, B \in \Omega$ gilt $\Pr[B] = \Pr[B|A] \cdot \Pr[A] + \Pr[B|\bar{A}] \cdot \Pr[\bar{A}]$.

Das Ziegenproblem

Definition Ziegenproblem

In einer Quizshow mit drei verschlossenen Türen stehen hinter 2 Türen eine Ziege, hinter der dritten Tür ein Auto.

1. Ein Kandidat K wählt eine der drei Türen aus.
2. Der Quizmaster öffnet eine Tür mit einer Ziege.
3. Der Kandidat darf nun die Tür wechseln.

Frage: Soll der Kandidat bei seiner Tür bleiben oder wechseln?

Lösung:

- Uns interessiert $\Pr[G]$ für $G =$"Kandidat gewinnt bei Türwechsel."
- Wir betrachten das Ereignis $A =$"K wählt in Schritt 1 das Auto aus."
- Es gilt $\Pr[G|A] = 0]$ und $\Pr[G|\bar{A}] = 1$.
- Der Satz von der totalen Wahrscheinlichkeit liefert
$$\Pr[G] = \Pr[G|A] \cdot \Pr[A] + \Pr[G|\bar{A}] \cdot \Pr[\bar{A}] = 0 \cdot \tfrac{1}{3} + 1 \cdot \tfrac{2}{3} = \tfrac{2}{3}.$$
- D.h. K gewinnt beim Türwechsel mit Ws $\frac{2}{3}$, sonst nur mit Ws $\frac{1}{3}$.

Satz von Bayes

Satz von Bayes

Sei Ω ein Wsraum mit $A_1, \ldots, A_n$ und $B \subseteq A_1 \cup \ldots \cup A_n$, $\Pr[B] > 0$.
Dann gilt $\Pr[A_i|B] = \frac{\Pr[A_i \cap B]}{\Pr[B]} = \frac{\Pr[B|A_i] \cdot \Pr[A_i]}{\sum_{j=1}^{n} \Pr[B|A_j] \cdot \Pr[A_j]}$ für alle $i \in [n]$.

Beweis:

- Folgt direkt aus dem Satz von der totalen Wahrscheinlichkeit.

Korollar zum Satz von Bayes

Sei $\Pr[B] > 0$. Dann gilt $\Pr[A|B] = \frac{\Pr[B|A] \cdot \Pr[A]}{\Pr[B]}$.

Anmerkung:

- Der Satz von Bayes erlaubt ein Vertauschen der Bedingung.
- D.h. anstatt A_i unter der Bedingung B betrachtet man B unter der Bedingung aller A_j.

Anwendung: Fehlerhafter Datenkanal

Anwendung:

- Über einen fehlerhaften Datenkanal werden Bits 0,1 gesendet.
- Sei S_i =“Bit i wird gesendet” mit $\Pr[S_0] = \frac{3}{10}$ und $\Pr[S_1] = \frac{7}{10}$.
- Sei R_i =“Bit i wird empfangen” mit $\Pr[R_1|S_0] = \frac{3}{10}$, $\Pr[R_0|S_1] = \frac{1}{10}$.
- Wir erhalten einen Übertragungsfehler mit Ws
  $\Pr[S_0 \cap R_1] + \Pr[S_1 \cap R_0] = \Pr[R_1|S_0] \cdot \Pr[S_0] + \Pr[R_0|S_1] \cdot \Pr[S_1] = \frac{16}{100}$.
- Wir empfangen eine Eins mit der Ws
  $\Pr[R_1] = \Pr[R_1|S_0] \cdot \Pr[S_0] + \Pr[R_1|S_1] \cdot \Pr[S_1] = \frac{3}{10} \cdot \frac{3}{10} + \frac{9}{10} \cdot \frac{7}{10} = \frac{72}{100}$.
- Falls eine 0 empfangen wird, wurde eine 0 gesendet mit Ws

$$\Pr[S_0|R_0] = \frac{\Pr[R_0|S_0] \cdot \Pr[S_0]}{\Pr[R_0]} = \frac{\frac{7}{10} \cdot \frac{3}{10}}{\frac{28}{100}} = \frac{75}{100}.$$

Würfel sind erinnerungslos

Motivation: Unabhängigkeit

- Wir würfeln zweimal, d.h. $\Omega = [6]^2$ mit $\Pr[\omega] = \frac{1}{36}$ für alle $\omega \in \Omega$.
- Sei $A_i =$"Augenzahl in Wurf i ist gerade."
- A_2 sollte unabhängig von A_1 sein, d.h. $\Pr[A_2|A_1] = \Pr[A_2]$.
- Wir rechnen nach, dass $A_1 \cap A_2 = \{2,4,6\} \times \{2,4,6\}$ und damit

$$\Pr[A_2|A_1] = \frac{\Pr[A_1 \cap A_2]}{\Pr[A_1]} = \frac{\frac{9}{36}}{\frac{1}{2}} = \frac{1}{2} = \Pr[A_2].$$

Definition Unabhängigkeit

Definition Unabhängigkeit

Sei Ω ein Wsraum mit $A, B \in \Omega$. Wir nennen A und B *unabhängig* falls

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B].$$

Anmerkung:

- Für $\Pr[B] > 0$ gilt $\Pr[A] = \frac{\Pr[A \cap B]}{\Pr[B]} = \Pr[A|B]$.
- D.h. Eintreffen von B hat keinen Einfluss auf das Eintreffen von A.

Bsp: Szenario wie auf der Folie zuvor

- Sei $B =$"Summe der Augenzahlen beträgt 7."
- Es gilt $A_1 \cap B = \{(2,5),(4,3),(6,1)\}$ und
$$\Pr[B] = \frac{|\{(1,6),(2,5),(3,4),(4,3),(5,2),(6,1)\}|}{|\Omega|} = \frac{1}{6}.$$
- D.h. $\Pr[A_1 \cap B] = \frac{3}{36} = \frac{1}{2} \cdot \frac{1}{6} = \Pr[A] \cdot \Pr[B]$.
- Damit sind die Ereignisse A_1 und B unabhängig.

Paarweise Unabhängigkeit vs. Unabhängigkeit

Beispiel: Wir betrachten das Szenario von zuvor.

- Wissen bereits, dass A_1, A_2 und A_1, B unabhängig sind.
- Analog folgt, dass A_2 und B unabhängige Ereignisse sind.
- **Frage:** Was ist mit den Ereignisse $A_1 \cap A_2$ und B?
- Für $A_1 \cap A_2$ sind beide Würfe gerade, d.h. die Summe ist gerade.
- Damit gilt $\Pr[(A_1 \cap A_2) \cap B] = 0$, aber $\Pr[A_1 \cap A_2] \cdot \Pr[B] > 0$.
- D.h. die Ereignisse $A_1 \cap A_2$ und B sind abhängig.

Definition Unabhängigkeit von n Ereignissen

Sei Ω ein Wsraum mit $A_1, \ldots, A_n \in \Omega$. Wir nennen $A_1, \ldots, A_n$ unabhängig, falls $\Pr[\bigcap_{i \in I} A_i] = \prod_{i \in I} \Pr[A_i]$ für alle $I \subseteq [n]$.

Eigenschaft von Unabhängigkeit

Lemma Unabhängige Ereignisse

Sei Ω ein Wsraum mit unabhängigen Ereignissen $A_1, \ldots, A_n$. Dann gilt für alle $s \in \{0,1\}^n$ dass $\Pr[A_1^{s_1} \cap \ldots \cap A_n^{s_n}] = \Pr[A_1^{s_1}] \cdot \ldots \cdot \Pr[A_n^{s_n}]$, wobei $A_i^0 = \bar{A}_i$ und $A_i^1 = A_i$.

Beweis: per Induktion über die Anzahl k der Nullen in s.

- **IA** für $k = 0$, d.h. $s = 1^n$. Damit gilt
$$\Pr[A_1 \cap \ldots \cap A_n] = \Pr[A_1] \cdot \ldots \Pr[A_n].$$
- **IS** $k-1 \to k$: s enthalte k Nullen, oBdA $s_1 = 0$. Es gilt
$\Pr[\bar{A}_1 \cap A_2^{s_2} \cap \ldots \cap A_n^{s_n}] = \Pr[A_2^{s_2} \cap \ldots \cap A_n^{s_n}] - \Pr[A_1 \cap A_2^{s_2} \cap \ldots \cap A_n^{s_n}]$.
- Nach IV können wir dies schreiben als
$$\begin{aligned} & \Pr[A_2^{s_2}] \cdot \ldots \cdot \Pr[A_n^{s_n}] - \Pr[A_1] \cdot \Pr[A_2^{s_2}] \cdot \ldots \cdot \Pr[A_n^{s_n}] \\ = \; & (1 - \Pr[A_1]) \cdot \Pr[A_2^{s_2}] \cdot \ldots \cdot \Pr[A_n^{s_n}] \\ = \; & \Pr[\bar{A}_1] \cdot \Pr[A_2^{s_2}] \cdot \ldots \cdot \Pr[A_n^{s_n}]. \end{aligned}$$

Anmerkung: Die Rückrichtung des Lemmas gilt ebenfalls.

Unabhängigkeit der Gegenereignisse

Korollar Unabhängigkeit der Gegenereignisse

Sei Ω ein Wsraum mit unabhängigen Ereignisse A_1, A_2. Dann sind $A_1, \bar{A}_2$ und $\bar{A}_1, A_2$ und $\bar{A}_1, \bar{A}_2$ jeweils unabhängig.

Beweis:

- Aus Unabhängigkeit von A_1, A_2 folgt für $s = (1, 0)$, dass
$$\Pr[A_1 \cap \bar{A}_2] = \Pr[A_1] \cdot \Pr[\bar{A}_2].$$
- Die beiden anderen Fälle folgen für $s = (0, 1)$ bzw. $s = (0, 0)$.

Schnitt, Vereinigung unabhängiger Ereignisse

Lemma Unabhängigkeit abgeschlossen unter $\cap$, $\cup$

Sei Ω ein Wsraum mit unabhängigen Ereignisse A, B, C. Dann sind

1. $A \cap B$, C unabhängig.
2. $A \cup B$, C unabhängig.

Beweis:

- **Schnitt:** Es gilt

  $$\Pr[(A \cap B) \cap C] = \Pr[A \cap B \cap C] = \Pr[A] \cdot \Pr[B] \cdot \Pr[C] = \Pr[A \cap B] \cdot \Pr[C].$$

- **Vereinigung:** Es gilt

$$\begin{aligned} \Pr[(A \cup B) \cap C] &= \Pr[(A \cap C) \cup (B \cap C)] \\ &= \Pr[A \cap C] + \Pr[B \cap C] - \Pr[A \cap B \cap C] \\ &= (\Pr[A] + \Pr[B] - \Pr[A \cap B]) \cdot \Pr[C] \\ &= \Pr[A \cup B] \cdot \Pr[C]. \end{aligned}$$

Rechnen mit unabhängigen Ereignissen

Szenario:

- Routen R_1, R_2 zwischen einem Sender und einem Empfänger.
- R_1 geht durch die beiden Knoten k_1, k_2, R_3 führt durch k_3.
- Jeder Knoten ist unabhängig mit Ws p intakt.
- **Frage:** Wie groß ist die Ws, dass eine der Routen verfügbar ist?

- Sei $R_i =$"Route i ist verfügbar" für $i = 1, 2$.
- Sei $K_i =$"Knoten j ist verfügbar" für $j \in [3]$.
- Dann gilt $\Pr[R_1] = \Pr[K_1 \cap K_2] = \Pr[K_1] \cdot \Pr[K_2] = p^2$.
- Ferner ist $\Pr[R_2] = p$. Wir sind interessiert an

$$\begin{aligned} \Pr[R_1 \cup R_2] &= 1 - \Pr[\overline{R_1 \cup R_2}] = 1 - \Pr[\bar{R}_1 \cap \bar{R}_2] \\ &= 1 - \Pr[\bar{R}_1] \cdot \Pr[\bar{R}_2] = 1 - (1 - p^2)(1 - p) \\ &= p + p^2 + p^3. \end{aligned}$$

Zufallsvariable und Dichtefunktion

Definition Zufallsvariable

Sei Ω ein Wsraum. Eine Abbildung $X : \Omega \to \mathbb{R}$ heißt *Zufallsvariable* mit Wertebereich $W_X := X(\Omega) = \{y \in \mathbb{R} \mid \exists \omega \in \Omega \text{ mit } X(\omega) = y\}$.

Sei $A = X^{-1}(y) = \{\omega \in \Omega \mid X(\omega) = y\}$. Dann definieren wir $\Pr[X = y] := \Pr[A]$ und $\Pr[X \leq y] := \sum_{x \in W_X, x \leq y} \Pr[X = x]$.

Beispiel: dreimaliger Münzwurf

- Wsraum $\Omega = \{K, Z\}^3$ für Kopf und Zahl.
- $Y : \Omega \to \{0, 1, 2, 3\}$ beschreibe die Anzahl von Kopf.
- Es gilt $\Pr[Y = 0] = \Pr[(Z, Z, Z)] = \frac{1}{8}$. Ferner gilt
$$\Pr[Y = 1] = \Pr[(K, Z, Z)] + \Pr[(Z, K, Z)] + \Pr[(Z, Z, K)] = \frac{3}{8}.$$
- Analog folgern wir $\Pr[Y = 2] = \frac{3}{8}$ und $\Pr[Y = 3] = \frac{1}{8}$.
- Man beachte, dass die sogenannte Dichtefunktion f_X von X mit
$$f_X : W_X \to [0, 1] \text{ und } x \mapsto \Pr[X = x]$$
einen Wsraum auf W_X definiert.

Erwartungswert

Definition Erwartungswert

Sei X eine Zufallsvariable. Der *Erwartungswert* von X ist definiert als

$$E[X] := \sum_{x \in W_X} x \cdot \Pr[X = x],$$

sofern $\sum_{x \in W_X} |x| \cdot \Pr[X = x]$ konvergiert.

Beispiel: Sei Y definiert wie auf der Folie zuvor.

- Es gilt $E[Y] = \sum_{i=0}^{3} i \cdot \Pr[Y = i] = 0 \cdot \frac{1}{8} + 1 \cdot \frac{3}{8} + 2 \cdot \frac{3}{8} + 3 \cdot \frac{1}{8} = \frac{3}{2}$.
- D.h. die erwartete Anzahl von Kopf bei 3 Münzwürfen ist $\frac{3}{2}$.

Beispiel Erwartungswert

Szenario: Münzwurfspiel in einem Spielcasino

- Es wird eine Münze geworfen, bis zum erstem Mal Kopf erscheint.
- Sei k die Anzahl der Würfe.
- Falls $k = \begin{cases} \text{ungerade} & \text{Spieler zahlt } k \text{ Euro an die Bank} \\ \text{gerade} & \text{Bank zahlt } k \text{ Euro an den Spieler} \end{cases}$.
- **Frage:** Was ist der erwartete Gewinn der Bank?

- Wir definieren eine Zufallsvariable $G : \mathbb{N} \to \mathbb{Z}$ für den Gewinn

$$G(k) = \begin{cases} k & \text{falls } k \text{ ungerade} \\ -k & \text{falls } k \text{ gerade} \end{cases}.$$

- Es gilt $\Pr[\text{“Anzahl Würfe ist } k\text{”}] = 2^{-k}$ und damit

$$E[G] = \sum_{k=1}^{\infty} (-1)^{k-1} k \cdot \left(\tfrac{1}{2}\right)^k = \tfrac{1}{2} \sum_{k=1}^{\infty} k \cdot \left(-\tfrac{1}{2}\right)^{k-1} = \tfrac{1}{2} \cdot \frac{1}{(1+\frac{1}{2})^2} = \frac{2}{9}.$$

- Man überprüfe analog die Konvergenz von

$$\sum_{k=1}^{\infty} |(-1)^{k-1} k| \cdot \left(\tfrac{1}{2}\right)^k = \sum_{k=1}^{\infty} k \cdot \left(\tfrac{1}{2}\right)^{k-1}.$$

Erwartungswert muss nicht definiert sein

Beispiel:

- Wir betrachten ein modifiziertes Spiel mit dem Gewinn

$$G(k) = \begin{cases} 2^k & \text{falls } k \text{ ungerade} \\ 2^{-k} & \text{falls } k \text{ gerade} \end{cases}.$$

- Dann gilt $\sum_{k=1}^{\infty} |(-1)^{k-1} 2^k| \cdot \left(\frac{1}{2}\right)^k = \sum_{k=1}^{\infty} 1^{k-1}$.
- D.h. $E[G]$ ist nicht definiert, da die Summe nicht konvergiert.

- Casino-Spiel mit einer Münze, die mit Ws p Kopf zeigt.
- Sei $X : \mathbb{N} \to \mathbb{N}$ eine Zufallsvariable für die Anzahl der Münzwürfe.
- Es gilt $\Pr[X = k] = (1-p)^{k-1} p$ und damit

$$E[X] = p \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} = p \cdot \frac{1}{(1-(1-p))^2} = \frac{1}{p}.$$

Linearität des Erwartungswerts

Satz Linearität des Erwartungswerts

Sei X eine Zufallsvariable und $a, b \in \mathbb{R}$. Dann gilt

$$E[aX + b] = a \cdot E[X] + b.$$

Beweis:

$$\begin{aligned} E[aX+b] &= \sum_{x \in W_X} (ax+b) \cdot \Pr[X = x] \\ &= a \sum_{x \in W_X} x \cdot \Pr[X = x] + b \cdot \sum_{x \in W_X} \Pr[X = x] \\ &= a \cdot E[X] + b \end{aligned}$$

Varianz und Standardabweichung

Definition Varianz und Standardabweichung

Sei X eine Zufallsvariable. Dann definieren wir die *Varianz* von X als $\mathrm{Var}[X] := E[(X - E[X])^2]$. Mit $\sqrt{\mathrm{Var}[X]}$ bezeichnen wir die *Standardabweichung* von X.

Viel Erfolg in der Klausur am Do., den 26.02.09!