Diskrete Mathematik

SS 2002

Bernhard Schmitt

Inhaltsverzeichnis

0 Einleitung

Bei vielen Verfahren aus dem (weiteren) Bereich der Informatik treten mathematische Hintergründe als Grundlage der Verfahren oder bei ihrer (Aufwands-) Analyse auf. Einige Problemstellungen sollen diese Perspektive im Beispiel aufzeigen.

Aufwandsanalyse

Ein klassisches Beispiel ist die Aufwandsanalyse bei Sortier-Algorithmen.

Elementares Beispiel: Bestimme Maximalwert von $x[1], \ldots, x[n]$, $n \geq 1$;

	Anweisung	*Aufwand*
Z1:	$k := n-1;\ m := x[n];$	2
Z2:	while $k > 0$ do begin	n
Z3:	if $x[k] > m$ then	$n-1$
Z4:	$m := x[k];$	A_n
Z5:	$k := k-1;$	n
	end	

Der Algorithmus benötigt $2n-1$ Vergleiche und $A_n + n + 2$ sonstige Oper./Zuweisungen.

Hauptproblem: Wert von A_n? In welchem Sinn?

Minimalwert	$A_n = 0,$	wenn $x[m] = \max_i x[i]$,
Maximalwert	$A_n = n-1,$	wenn $x[1] > x[2] > \ldots x[n]$,
Mittel-Wert	$A_n = ??$	*(unter welchen Annahmen?)*

A_n hängt offensichtlich von der Anordnung der $x[i]$, aber nicht von deren tatsächlichen Werten ab. Also: **oBdA:** $x[i] \in \{1, \ldots, n\}$ (d.h. Gleichheit ausgeschlossen).

Daher sind alle $n!$ Permutationen dieser Menge zu betrachten, z.B., bei $n = 3$ also folgende 6 Fälle :

Perm.:	123	132	213	231	312	321	
$A_n =$	0	1	0	1	1	2	Mittelwert=5/6

Im allgemeinen Fall ist nachweisbar: $A_n \cong \ln n$ (*Rekursionsformel*)

Ähnliche Situation: Quicksort. $\rightarrow$ $\boxed{\textbf{A}\ \text{Kombinatorik}}$

Optimierungsprobleme in Graphen

Das Vorgehen bei mehrstufigen Entscheidungsprozessen kann oft als das Durchlaufen eines Entscheidungs-Baums/-Graphen interpretiert werden. Je nach Fragestellung kann sich das Problem dann auf eine der folgenden abstrakten Fragen reduzieren:

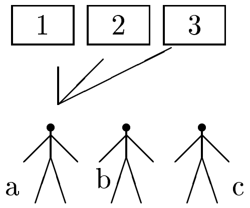
— Finde einen Weg durch alle Knoten des Graphen,
— Finde kürzesten Weg zwischen zwei Knoten des Graphen,
— Finde Teilgraphen mit minimalen Kantengewichten, usw.

Beispiel:

— Zuordnung von Mitarbeitern auf Aufgaben

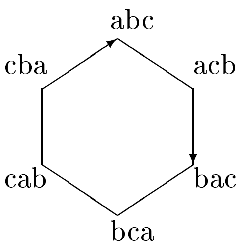
— Zuordnung von Prozessen auf Prozessoren (Parallel-Rechner)

— etc.

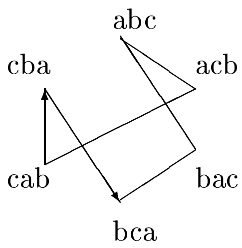


Primitiv-Algorithmus: Betrachte alle Permutationen von $\{a, b, c\}$, berechne jeweils Kosten/Gewinn bei Zuordnung auf die Probleme 1,2,3.

Frage: Erzeugung aller Permutationen ($\rightarrow$§1.4)? Effizienz dabei?



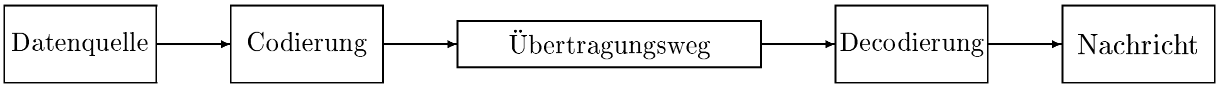
Von *acb* zu *bac* wechseln *alle* Zuordnungen, alle zugehörigen Kosten sind dort neu zu berechnen. Ist Durchlauf möglich nur mit Einzel-Vertauschungen? *Rechts:* Nur Transpositionen (Nachbarvertauschungen, Rest unverändert)



Codierung

Problem 'Codierung' in den Bedeutungen (drei **K**):			Mathem. Hilfsmittel:
1.	**K**orrektur:	Übertragungsfehler erkennen/korrigieren	endliche Vektorräume
2.	**K**omprimierung:	Daten-Umfang verringern	Graphen/Bäume
3.	**K**ryptographie:	Kopie, Fälschung von Daten verhindern	endliche Zahlkörper

Daten-Übertragung, Stationen:



Übertragungsweg konkret: Telefonleitung, Netzwerk, Datenträger (CD, Papier).

Problem 1:	Übertragung, Speicherung störanfällig (Störsignal, Beschädigung)
Problem 2:	Transferzeit, Speicherplatz von Datenmengen reduzieren
Problem 3:	Bei offenen Wegen Mithören, Kopien, Fälschung durch Dritte verhindern

$\boxed{\text{Zu 1}}$ Fehler-erkennende / -korrigierende Codes

Einfachster Fall: Zusätzliche Prüfziffer, der Code wird *redundant*, z.B.,

1a Paritätsbit bei Binärcodes, 7-Bit+Parität:

$$\text{'D'}= \boxed{0}\ \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}$$

das höchste Bit wird so gesetzt, daß die Zahl der Einsen gerade ist. Erkennt Verfälschung eines einzelnen Bits (Korrektur $\rightarrow$ *Protokoll*)

1b Prüfzeichen im täglichen Leben, bei

Nummern für Verkaufsartikel, Bankkonten, Geldscheinen, Bücher (ISBN).

Häufiger menschlicher Fehler: Vertauschung aufeinanderfolgender Zeichen ('Flip').

Generelle Prüfkonstruktion:

Zulässige Nummern (Code-Worte) erfüllen eine gegebene (lineare!) Bedingung.

z.B., gerade Parität des Wortes $a_0 a_1 \dots a_n$, $a_i \in \{0,1\}$ heißt, daß die Summe $a_0+a_1+\cdots+a_n$ gerade ist, also Rest 0 bei Division durch 2 besitzt:

$$\sum_{i=0}^{n} a_i \equiv 0 \pmod 2.$$

Durch Parität ist aber keine Vertauschung von Zeichen erkennbar. Ausweg bei Nicht-Dualzahlen: gewichtete Summe. Z.B., haben

ISBN-Nummern 10 Stellen $x_1, \dots, x_{10}$ und zulässige Nummern erfüllen die Bedingung

$$\sum_{i=1}^{10} i\, x_i \equiv 0 \pmod{11}$$

die Summe ist also Vielfaches der Primzahl 11.

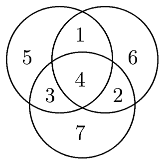
Beispiel: ISBN=	3 5 2 8 0 7 2 6 8 7	
PrüfS. in $\mathbb{N}$:	3+10+6+32+0+42+14+48+72+70	=297=$27 \cdot 11$
PrüfS. in $\mathbb{Z}_{11}$:	3+10+6+10+0 +9 +3 +4 +6 +4	=55$\equiv$ 0

Die Rechnung kann also auf die mit den Divisionsresten bzgl. 11 zurückgeführt werden. Da 11 Primzahl ist, ist die Menge $\mathbb{Z}_{11}$ dieser Divisionsreste algebraisch ein *Körper* ($\rightarrow$ §3.1). Für die Informatik sehr praktische Primzahlen sind

$$127 = 2^7 - 1,\ 2^{31} - 1 \text{ (Mersenne)},\ 257 = 2^8 + 1,\ 65537 = 2^{16} + 1 \text{ (Fermat)}.$$

Durch Hinzufügen weiterer Prüfziffern so, daß die zulässigen Codewörter mehrere *linear unabhängige* Bedingungen erfüllen, können sogar einzelne Fehler korrigiert werden.

Hamming Code $x_1 x_2 \dots x_7$
16 Codeworte, 3 Prüfziffern,
1 Fehler behebbar



die Summe der Stellen in jedem Kreis ist gerade: 3 lin. unabh. Bedingungen

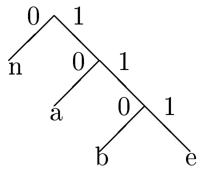
$\rightarrow$ $\boxed{\textbf{B}\text{ Algebra, §3}}$

$\boxed{\text{Zu 2.}}$ Wenn in der Datenquelle Datenwerte mit unterschiedlicher Häufigkeit auftreten, ist der *Informationsgehalt* der verschiedenen Werte nicht gleich. In Texten vieler Sprachen ist dies der Fall: der Text DISKRT MATHMATIK ist noch lesbar, da 'E' der häufigste Buchstabe im (deutschen) Alphabet ist und daher nur geringen Informationsgehalt hat. Der Anteil von 'e' ist ca. $0.147 > 1/7$.

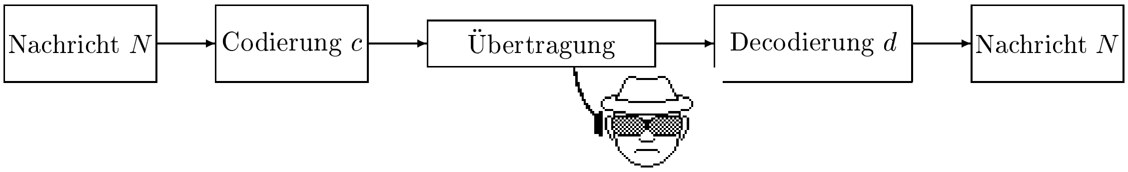
Beispiel: Die Nachricht $\boxed{\text{bananen}}$ besteht aus 7 Stellen mit 4 unterschiedlichen Zeichen. Bei Standardcodierung mit 2 Bit, $a = 00$, $b = 01$, $c = 10$, $n = 11$, benötigt die codierte Nachricht $2 \cdot 7 = 14$ Bit: $\boxed{01001100111011}$.
Einfache Decodierung: Zerschneiden in 2-Bit-Stücke.

Günstiger: Kurze Codeworte für häufige Buchstaben, z.B., $n = 0$, $a = 10$, $b = 110$, $e = 111$ (Länge 3!). Codierte Nachricht $\boxed{1101001001110}$ hat Länge 13 Bit.

Decodierung?? Kein Codewort ist Anfang eines anderen!
$\to$ Durchgehen durch Nachricht bis ein Zeichen erkannt ist.
Hintergrund *Codebaum* ($\cong$ Suchbaum): Start an der *Wurzel* des Baums oben, je nach Bitwert wird rechter oder linker Ast verfolgt bis zum *Blatt*



$\boxed{\text{Zu 3.}}$ Kryptographie, Risiken im Übertragungsweg:



Ziel: Die codierte Nachricht $c(N)$ ist von Mithörer nicht lesbar.
Nur der Empfänger kennt die inverse Decodier-Abbildung $d = c^{-1}$ und kann die Nachricht $N = d(c(N))$ rekonstruieren.
Meist gilt: Die Abbildungen hängen von einem Parameter S ab, genannt 'Schlüssel'.
Klassisch: einfache Abbildungen c, d mit gleichem Schlüssel für c und d, z.B., stellenweise Addition mod 26:

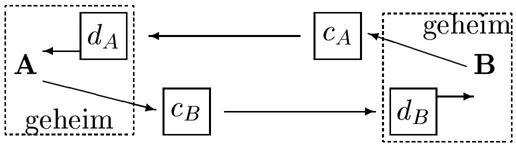
$$\begin{array}{r|l} N = & \text{D I S K R E T E M A T H E M A T I K} \\ S = & \text{A P R I L A P R I L A P R I L A P R} \\ \hline c(N) = & \text{E Y K T D F N W V M U X W V M U Y B} \end{array}$$

Die Sicherheit solcher *symmetrischer* Verfahren hängt von der vorherigen(!) sicheren Übermittlung des Schlüssels ab.
Weitere Bedingung: Schlüsselzeichen sind zufällig verteilt, keine Wiederholung ($\to$ *Einmalschlüssel* beweisbar sicher).

Asymmetrische Verfahren

Entscheidende Idee [Diffie&Hellmann]: Empfänger an Verschlüsselung beteiligen!
Voraussetzung: d ist **nicht** aus der Kenntnis von c herleitbar ('Falltür-Funktion').
Sichere Kommunikation zwischen A und B:

B kennt d_B, sendet c_B offen an A (an alle)
  A codiert Nachricht mit c_B
A kennt d_A, sendet c_A offen an B (an alle),
  B codiert Nachricht mit c_A



Der Besitzer des Codes muß natürlich (in Kenntnis einer geheimen Hintergrundinformation) d zu c konstruieren können, nicht aber *Andere*. Die gängigsten Falltürfunktionen beruhen auf algebraischen Operationen in endlichen Zahl-Ringen /-Körpern, z.B., das

RSA-Verfahren [Rivest-Shamir-Adleman]:
Der Besitzer wählt eine große Zahl m ($\cong$ 200 Dezimalen) und Exponenten $s, g \in \{1, \ldots, m-1\}$. Die Nachricht wird codiert als Zahl in $\mathbb{Z}_m$. Damit werden die Abbildungen definiert

$$c : \left\{ \begin{array}{ccc} \mathbb{Z}_m & \to & \mathbb{Z}_m \\ N & \mapsto & N^s (\operatorname{mod} m) =: X \end{array} \right. \qquad d : \left\{ \begin{array}{ccc} \mathbb{Z}_m & \to & \mathbb{Z}_m \\ X & \mapsto & X^g (\operatorname{mod} m) =: N \end{array} \right.$$

Geheimer Hintergrund: $m = pq$ ist das Produkt von zwei großen Primzahlen. Der geheime Exponent (Decodier-Schlüssel) g ist so gewählt, daß gilt

$$sg \equiv 1 \bmod (p-1)(q-1).$$

Diese Konstruktion erfordert die Kenntnis der Faktoren p und q. Die Faktorisierung $m \to p, q$ ist für andere bei $m > 10^{200}$ auf absehbare(?) Zeit undurchführbar (Im August 99 wurde mit massivem Computereinsatz ein 512-Bit-RSA-Verfahren 'geknackt', frühere Schätzungen dafür beliefen sich auf 50 Mio Jahre).

Weitere Anwendungen: Digitale Unterschrift (EU-/Bundesgesetz, Trust-Center), E-mails mit Verfallsdatum (Microsoft-Prozess). $\to$ $\boxed{\textbf{B} \text{ Algebra, §3.3}}$

Teil A

Kombinatorik

1 Endliche Mengen

1.1 Elementare Regeln

Folgende Bezeichnungen werden im Zusammenhang mit Operationen von Mengen $A_1, A_2, \ldots, A_k$ eingeführt:

- *Vereinigung* der Mengen
$$\bigcup_{i=1}^{k} A_i := A_1 \cup A_2 \cup \ldots \cup A_k.$$
Wenn dabei die Mengen *paarweise disjunkt* sind, wird diese disjunkte Vereinigung auch als Summe geschrieben:
$$\sum_{i=1}^{k} A_i := \bigcup_{i=1}^{k} A_i, \quad \text{wenn} \quad A_i \cap A_j = \emptyset \quad \forall i \neq j.$$

- Das *Cartesische Produkt* der Mengen ist die Menge aller k-Tupel $(a_1, \ldots, a_k)$:
$$\prod_{i=1}^{k} A_i := A_1 \times A_2 \times \ldots \times A_k = \{(a_1, \ldots, a_k) : \ a_i \in A_i, \, i = 1, \ldots, k\}.$$
Im Gegensatz zur Vereinigung spielt hier natürlich die Reihenfolge der Faktoren ein Rolle.

- *Größe* von endlichen Mengen: Für eine endliche Menge A bezeichnet
$$|A| := \text{Anzahl der Elemente von } A.$$

Bis auf wenige Ausnahmen werden im folgenden nur endliche Mengen betrachtet. Für diese gelten folgende Rechenregeln.

Satz 1.1 *a) Die Mengen $A_1, \ldots, A_k$, seien paarweise disjunkt, $A_i \cap A_j = \emptyset \forall i \neq j$. Für ihre Vereinigung gilt*
$$|\sum_{i=1}^{n} A_i| = \sum_{i=1}^{n} |A_i|.$$

b) Für das Cartesische Produkt beliebiger Mengen $A_1, \ldots, A_k$ gilt
$$|A_1 \times A_2 \times \ldots \times A_k| = \prod_{i=1}^{k} |A_i|.$$

Der Beweis ist trivial und kann durch Induktion über die Zahl der Mengen und die Anzahl der Elemente geführt werden.

Bemerkung: Der allgemeine Fall von Teil a) wird später im §1.5 behandelt.

Beispiel: Zu a) In vielen Anwendungen sind die Mengen A_i Teilmengen einer Gesamtmenge A, die durch sich gegenseitig ausschließende Eigenschaften E_i charakterisiert werden:

$$A_i := \{x \in A : \ x \text{ erfüllt } E_i\}.$$

Als Beispiel sei $A(n)$ die Menge aller Permutationen der Menge $N := \{1, \ldots, n\} \subseteq \mathbb{N}$, d.h.,

$$A(n) := \{(a_1, \ldots, a_n) : \ a_i \in N, \ a_i \neq a_j \forall i \neq j\}.$$

Zur Bestimmung der Zahl $P_n = |A(n)|$ wird $A(n)$ zerlegt in

$$A_i(n) := \{(a_1, \ldots, a_n) \in A : \ \underbrace{a_i = n}_{=E_i}\}, \quad i = 1, \ldots, n,$$

d.h. in allen Elementen von A_i steht in der Position i der Wert n. Da in den restlichen $n-1$ Positionen alle Permutationen (der restl. $n-1$ Werte) möglich sind, gilt $|A_i(n)| = P_{n-1}$ und mit Satz 1.1 folgt

$$P_n = |A(n)| = |\sum_{i=1}^{n} A_i(n)| = \sum_{i=1}^{n} |A_i(n)| = nP_{n-1}.$$

Dies ist eine (triviale) *Rekursionsformel*, die mit dem Anfangswert $P_1 = 1$ auf das bekannte Ergebnis $P_n = n(n-1) \cdots 2 \cdot 1 = n!$ führt.

Zu b) Die Produktregel des Satzes ist immer dann anwendbar, wenn man bei *mehrstufigen* Entscheidungen voneinander *unabhängige* Alternativen hat. Um z.B. von Marburg über Gießen nach Frankfurt zu fahren gebe es

3 mögl. Wege MR → GI: a, b, c
5 mögl. Wege GI → F: $\alpha, \beta, \gamma, \delta, \epsilon$

Daher existieren $3 \cdot 5 = 15$ mögliche Wege MR $\to F$. Denn in einer ‘Wegbeschreibung’, d.h., einem Paar (x, ξ) können

für x 3 verschiedene Werte aus $\{a, b, c\}$ und
für ξ 5 verschiedene Werte aus $\{\alpha, \beta, \gamma, \delta, \epsilon\}$

unabhängig voneinander eingetragen werden. Es gibt also eine *bijektive* (eineindeutige) Zuordnung von Wegen und 2-Tupeln aus dem cartesischen Produkt $\{a, b, c\} \times \{\alpha, \beta, \gamma, \delta, \epsilon\}$.

Dies ist ein Beispiel für die Anwendung von

Satz 1.2 *Wenn eine bijektive Abbildung $A \to B$ zwischen Mengen A, B existiert, gilt $|A| = |B|$.*

Daher kann, z.B., bei n-elementigen Mengen auch oBdA $A = N$ gesetzt werden.

1.2 k-Stichproben

Viele Betrachtungen von Auswahlmöglichkeiten können (vgl. §1.1) auf die Auswahl von k Elementen aus einer n-elementigen Grundmenge A (Alphabet) zurückgeführt werden. Die tatsächlich auftretenden Anzahlen hängen dabei von den einzuhaltenden *Regeln* ab:

R Reihenfolge: die k gewählten Elemente können als k-Tupel (Reihung beachtet) oder als k-Menge (Reihenfolge unwichtig, bei $A = N$ etwa werden die Zahlen oBdA nach der Größe angeordnet) betrachtet werden.

W Wiederholung: Elemente von A dürfen beliebig oft (Ziehen aus einem Topf mit Zurücklegen) oder höchstens einmal (ohne Zurücklegen) vorkommen.

Für diese k-Stichproben verwendet man in Abhängigkeit von den Regeln unterschiedliche Bezeichnungen:

Bezeichnung	Reihenfolge!	Reihenf. egal
Wiederholung	Worte	Multimengen
keine Wiederh.	k-Permutationen	Teilmengen

Die möglichen Anzahlen unterschiedlicher k-Stichproben müssen in den vier Fällen getrennt diskutiert werden:

1. k-Worte: Die Menge der k-Worte ist einfach das cartesische Produkt
$$\{(a_1, \ldots, a_k) : \ a_i \in A\} = \prod_{i=1}^{k} A,$$
nach Satz 1.1 ist die Anzahl der k-Worte: n^k.

2. k-Permutationen: Eine Beschreibung der Gesamtmenge ist
$$\{(a_1, \ldots, a_k) : \ a_i \in A, \, a_i \neq a_j \forall i \neq j\}.$$
Die Größe dieser Menge bestimmt man rekursiv:

Zur Wahl von	a_1	gibt es	n	Möglichkeiten
Zur Wahl von	a_2	gibt es	$n-1$	Möglichkeiten (Wert von a_1 vergeben)
..	..	..	..	
Zur Wahl von	a_k	gibt es	$n-k+1$	Möglichkeiten

Die Gesamtanzahl ist daher
$$n(n-1)\cdots(n-k+1) = \frac{n!}{(n-k)!} =: n^{\underline{k}} \qquad \text{‘\textit{fallende Faktorielle}’} \tag{1}$$
$$\text{Analog:} \quad n(n+1)\cdots(n+k-1) = \frac{(n+k-1)!}{(n-1)!} =: n^{\overline{k}} \qquad \text{‘\textit{wachsende Faktorielle}’} \tag{2}$$
Grenzfälle dieser Definitionen:

- $n^{\underline{n}} = n!$, für $n = 0$ wird definiert $0! := 1$.
- $n^{\underline{n+k}} = 0$ für $k > 0$: A hat zu wenige Elemente, in dem Produkt tritt Faktor 0 auf.

3. k-Teilmengen: Im Unterschied zum letzten Fall stehen in der folgenden Beschreibung

$$\{\{a_1, \ldots, a_k\} : \ a_i \in A, \, a_i \neq a_j \forall i \neq j\} =: T$$

Mengenklammern, daher ist die Bedingung $a_i \neq a_j$ eigentlich überflüssig. Die Anzahl der k-Teilmengen wird durch folgenden Wert angegeben:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n^{\underline{k}}}{k!}, \quad \textit{Binomialkoeffizient.} \tag{3}$$

Später wird diese Behauptung nochmal direkt durch Induktion bewiesen, hier erfolgt dies durch Zurückführung auf die Menge S aller k-Permutationen.

Man betrachte die Abbildung $f : S \to T$,

$$f : \ (a_1, \ldots, a_k) \mapsto \{a_1, \ldots, a_k\},$$

die eine k-Permutation auf die Menge der auftretenden Elemente abbildet. f ist surjektiv, aber nicht injektiv, da alle $k!$ Permutationen eines bestimmten k-Tupels $(a_1, \ldots, a_k)$ auf die gleiche Menge abgebildet werden. Aus Teil 2 folgt daher die Anzahl $n^{\underline{k}}/k!$.

4. k-Multimengen: Nach Definition sind die Elemente einer Menge verschieden. Wenn man dagegen Wiederholungen zuläßt, redet man von einer Multimenge. Auch Multimengen werden mit den üblichen Mengenklammern bezeichnet.

Z.B. ist $\{1, 3, 3, 4, 4, 4\}$ eine Multimenge über $\{1, 2, 3, 4\}$, wobei das Element 1 mit *Vielfachheit* 1, die 3 mit Vielfachheit 2 und die 4 mit Vielfachheit 3 auftritt.

Sei nun S die Menge der k-Multimengen über N. Für ein Element von S,

$$\{a_1, \ldots, a_k\} \in S \text{ gelte } a_1 \leq a_2 \leq \ldots \leq a_k.$$

Mit der Menge T der k-Teilmengen von $\{1, \ldots, n+k-1\}$ wird die Abbildung $f : S \to T$ definiert durch

$$f : \ \{a_1, \ldots, a_k\} \mapsto \{a_1, a_2 + 1, a_3 + 2, \ldots, a_k + k - 1\}.$$

Das Bild $\{b_1, \ldots, b_k\} := f(\{a_1, \ldots, a_k\}) \in T$ ist tatsächlich eine Menge, da $b_1 < b_2 < \ldots < b_k$ gilt. Da explizit die Umkehrabbildung $g : T \to S$ angegeben werden kann durch

$$g(\{b_1, \ldots, b_k\}) = \{b_1, b_2 - 1, \ldots, b_k - k + 1\}$$

(nachrechnen!), ist f bijektiv und nach Satz 1.2 gilt daher

$$|S| = |T| = \binom{n+k-1}{k} = \frac{n(n+1)\cdots(n+k-1)}{k!} = \frac{n^{\overline{k}}}{k!}.$$

Der folgende Satz faßt die Ergebnisse zusammen.

Satz 1.3 *Aus einer Grundmenge A, $|A| = n$, werden jeweils k Elemente gewählt. Die Anzahl der dabei möglichen Fälle ist je nach Zählregel:*

Anzahl	*Reihenfolge!*	*Reihenf. egal*
Wiederholung	n^k *(Worte)*	$\frac{n^{\overline{k}}}{k!} = \binom{n+k-1}{k}$ *(Multimengen)*
keine Wiederh.	$n^{\underline{k}}$ *(Permut.)*	$\frac{n^{\underline{k}}}{k!} = \binom{n}{k}$ *(Teilmengen)*

Anwendung:

Die Anzahlen aus Satz 1.3 treten natürlicherweise auf bei der Betrachtung bestimmter Klassen von Abbildungen, z.B., der Aufteilung von Bällen auf Schubladen (Hash-Funktionen!). Dazu sei $K := \{1, \ldots, k\}$ (Bälle) und wieder $N := \{1, \ldots, n\}$ (Fächer), es werden Abbildungen $K \to N$ diskutiert. Einfache Fälle:

Anzahl	*aller*	Abbildungen ist	n^k	(jeder Wert hat n mögl.Bilder)
Anzahl	der *injektiven*	Abbildungen ist	$n^{\underline{k}}$	(belegte Fächer sind blockiert)
Anzahl	der *surjektiven*	Abbildungen	??	(alle Fächer belegt)

Bei dieser Frage ist natürlich $k \geq n$ vorauszusetzen. Bei einer surjektiven Abbildung gehört zu jedem $b \in N$ (mind.) ein Urbild, d.h., es gilt

$$f^{-1}(b) := \{a \in K : \ f(a) = b\} \neq \emptyset.$$

Als Abbildung vermittelt f eine eindeutige und vollständige Zuordnung, d.h.,

$$f^{-1}(b) \cap f^{-1}(c) = \emptyset \text{ für } b \neq c, \quad \text{und} \quad K = \sum_{b \in N} f^{-1}(b).$$

Daher stellt das Mengensystem $\{f^{-1}(b) : \ b \in N\}$ eine disjunkte Zerlegung von K in n Teilmengen dar, eine sogenannte

$$n - \textit{Partition} \text{ der Menge } K \text{ mit } |K| = k \geq n.$$

Die Anzahl dieser n-Partitionen wird mit S_{kn} bezeichnet. Da für jede Partition die n Bildwerte permutierbar sind, ist die Anzahl der surjektiven Abbildungen

$$|Surj(K, N)| = n! \, S_{kn}.$$

Die Zahlen S_{kn} heißen *Stirling-Zahlen 2. Art* und haben mehrere Bedeutungen ($\to$§1.3).

1.3 Identitäten & Rekursionen für elementare Koeffizienten

Bei den Binomialkoeffizienten

$$\binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!}$$

bestimmt k die Anzahl der Faktoren, an der Stelle von n kann aber eine beliebige reelle oder komplexe Variable auftreten. Auch der Fall $k = 0$ läßt sich mit $0! = 1$ und $n^{\underline{0}} := 1$ erklären, denn $\binom{n}{0} = 1$ ist die Anzahl der leeren Teilmengen, auch bei $n = 0$: $\binom{0}{0} = 1$.

Defin. 1.4 *Für $x \in \mathbb{C}$, $k \in \mathbb{Z}$ werden* Binomialkoeffizienten *(als Polynome) definiert:*

$$\binom{x}{k} := \frac{1}{k!}\, x^{\underline{k}} = \frac{1}{k!}\, x(x-1)\cdots(x-k+1),$$

und die Faktoriellen $x^{\underline{k}} := x(x-1)\cdots(x-k+1)$, $x^{\overline{k}} := x(x+1)\cdots(x+k-1)$.

Für $k < 0$ setzt man $\binom{x}{k} = 0$. Elementare Identitäten:

Satz 1.5 *Mit $x \in \mathbb{C}$, $k, n \in \mathbb{N}_0$ gilt*

$$\begin{array}{ll}
a) & \binom{n}{k} = \binom{n}{n-k} \quad (= 0, k > n), \\
b) & \binom{-x}{k} = (-1)^k \binom{x+k-1}{k} \\
c) & \binom{x}{k} = \binom{x-1}{k-1} + \binom{x-1}{k}, \quad k \geq 1 \\
d) & \sum_{m=0}^{n} \binom{m}{k} = \binom{n+1}{k+1}, \\
e) & \sum_{k=0}^{n} \binom{x+k}{k} = \binom{x+n+1}{n}
\end{array}$$

Beweis

a) $\displaystyle \binom{n}{k} = \frac{n^{\underline{k}}}{k!} = \frac{n!}{k!(n-k)!} = \binom{n}{n-k}$, $n^{\underline{k}} = 0$ für $k > n$.

b) $(-x)^{\underline{k}} = (-x)(-x-1)\cdots(-x-k+1) = (-1)^k x(x+1)\cdots(x+k-1) = (-1)^k x^{\overline{k}}$.

c) $\displaystyle \frac{1}{k!}x^{\underline{k}} = \frac{1}{k!}x(x-1)^{\underline{k-1}} = \frac{1}{k!}(k + (x-k))(x-1)^{\underline{k-1}} = \frac{1}{(k-1)!}(x-1)^{\underline{k-1}} + \frac{1}{k!}(x-1)^{\underline{k}}$.

d) Induktion über n, $n = 0$: $\binom{0}{0} = 1 = \binom{1}{1}$; $k > 0$: $\binom{0}{k} = 0 = \binom{1}{k+1}$. Schritt:

$$\sum_{m=0}^{n} \binom{m}{k} + \binom{n+1}{k} \overset{I.V.}{=} \binom{n+1}{k+1} + \binom{n+1}{k} \overset{c)}{=} \binom{n+2}{k+1}.$$

e) Induktion über n, $n = 0$: $\binom{x}{0} = \binom{x+1}{0} = 1$; Schritt:

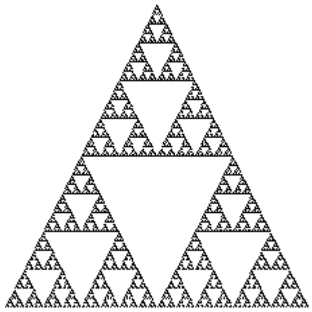
$$\sum_{k=0}^{n} \binom{x+k}{k} + \binom{x+n+1}{n+1} \overset{I.V.}{=} \binom{x+n+1}{n} + \binom{x+n+1}{n+1} \overset{c)}{=} \binom{x+n+2}{n+1}. \quad \blacksquare$$

Die wichtige Regel c) kann für $x = n \in \mathbb{N}$ als Rekursionsformel zur Berechnung der Binomialkoeffizienten $\binom{n}{k}$ verwendet werden mit den Randbedingungen

$$\binom{n}{0} = 1 \;\forall n \geq 0, \quad \binom{n}{k} = 0 \;\forall k > n.$$

Die führt auf das *Pascal-Dreieck*:

$k=$	0	1	2	3	4	5	6	7
$n=0$	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	



Die Regel a) beschreibt die Symmetrie bzgl. der Mittelachse, die Regeln d),e) behandeln darin Summen längs Diagonalen bzw. Vertikalen. Die Tatsache, daß die Gesamtzahl der Teilmengen einer n-Menge durch Vereinigung aller k-Teilmengen, $0 \leq k \leq n$, zustandekommt,

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

die Summe in Zeile n des Pascal-Dreiecks also gerade 2^n ist, ist ein Spezialfall im

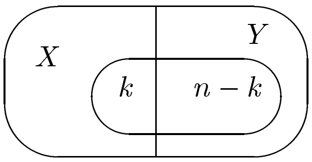
Satz 1.6 *Für $x, y \in \mathbb{C}$, $n \in \mathbb{N}_0$ gilt*

$$\text{a)} \qquad (x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} \qquad \textit{(Binomischer Satz)}$$

$$\text{b)} \qquad \binom{x+y}{n} = \sum_{k=0}^{n} \binom{x}{k} \binom{y}{n-k} \qquad \textit{(Vandermonde-Identität)}$$

Beweis a) Mit Induktion über n, Summationsverschiebung und Satz 1.5 c), vgl. Analysis 1.
b) Nur für $x, y \in \mathbb{N}$. Es seien X, Y disjunkte Mengen mit $|X| = x$, $|Y| = y$. Der Binomialkoeffizient gibt die Anzahl der n-Teilmengen A von $X \cup Y = X + Y$ an. Für jedes $0 \leq k \leq n$ können k Elemente einer n-Teilmenge A aus X und $n-k$ aus Y gewählt werden. Dazu gibt es

für	$A \cap X$	mit	$\lvert A \cap X \rvert = k$:	$\binom{x}{k}$	Möglichk.
für	$A \cap Y$	mit	$\lvert A \cap Y \rvert = n-k$:	$\binom{y}{n-k}$	Möglichk.
		d.h., zusammen		$\binom{x}{k}\binom{y}{n-k}$	Möglichk.



Summation über k führt zur Behauptung. ■

Die Verallgemeinerung des Binomischen Satzes auf m Summanden ist der Multinomische Satz:

$$(x_1 + \ldots + x_m)^n = \sum_{n_1+..+n_m=n} \frac{n!}{n_1! \cdots n_m!} x_1^{n_1} \cdots x_m^{n_m}$$

$$= n! \sum_{|\alpha|=n} \frac{x^\alpha}{\alpha!} \qquad \text{(Multi-Index-Schreibweise Analysis 2, Satz v.Taylor, } \alpha = (n_1, \ldots, n_m))$$

Beispiel: $(a+b+c)^3 = 6(\frac{a^3}{3!} + \frac{a^2b}{2!1!} + \frac{a^2c}{2!1!} + \frac{ab^2}{1!2!} + \frac{ac^2}{1!2!} + \frac{abc}{1} + \frac{b^3}{3!} + \frac{b^2c}{2!1!} + \frac{bc^2}{1!2!} + \frac{c^3}{3!}) = a^3 + 3a^2b + 3a^2c + 3ab^2 + 3ac^2 + 6abc + b^3 + 3b^2c + 3bc^2 + c^3$.

Stirling-Zahlen

Die erste Bedeutung der Stirlingzahlen 2. Art wurde schon in §1.3 angesprochen:

Defin. 1.7 *Es sei A eine Menge mit $|A| = n$ Elementen. Die Anzahl der Zerlegungen von A in k disjunkte Teilmengen (k-Partitionen) wird mit S_{nk} bezeichnet. S_{nk} heißt* Stirling-Zahl *2. Art.*

Triviale Werte von S sind: $S_{n0} = 0$ $(n > 0)$, $S_{nk} = 0$ $(k > n)$, weiter wird gesetzt $S_{00} := 1$. Eine Rekursionsformel für diese Stirlingzahlen und ihre zweite Bedeutung enthält der

Satz 1.8 *a) Es gilt die Rekursionsformel*

$$S_{nk} = S_{n-1,k-1} + kS_{n-1,k}, \quad k, n \in \mathbb{N}.$$

b) Es gilt die Polynom-Identität

$$x^n = \sum_{k=0}^{n} S_{nk}\, x^{\underline{k}}, \quad x \in \mathbb{C},\ n \in \mathbb{N}_0.$$

Beweis a) Sei $|A| = n$ und $a \in A$ fest gewählt.
Fall 1: Aus jeder der $(k-1)$-Partitionen der Restmenge $A \setminus \{a\}$ kann durch Hinzunahme der Menge $\{a\}$ eine k-Partition von A gemacht werden, die Anzahl ist $S_{n-1,k-1}$.
Fall 2: Aus jeder der k-Partitionen von $A \setminus \{a\}$ kann durch Hinzufügen von a in einer der Teilmengen eine k-Partition von A erzeugt werden. Da a in jede von k Teilmengen eingefügt werden kann, führt dieser Fall auf $k \cdot S_{n-1,k}$ Partitionen.
Die Gesamtanzahl ergibt sich aus der Summe der beiden Fälle.

b) Zunächst sei $x = m \in \mathbb{N}$ (triviale Fälle werden für $m \geq n$ vermieden). Dann ist m^n die Anzahl der n-Worte gebildet mit einem m-Alphabet M, $|M| = m$. Die Menge dieser Worte wird jetzt disjunkt zerlegt in die Mengen von n-Worten, die genau k verschiedene *Werte* haben, $0 \leq k \leq n$. Für jedes solche k werde die Indexmenge $N = \{1, \ldots, n\}$ zerlegt in k disjunkte Teile:

n-Tupel: $(a_1, a_2, a_3, \ldots, a_n)$

$$N_1 + \ldots + N_k = N$$

$N_1 \quad N_2 \quad N_k$

(Für diese Aufteilung ex. S_{nk} Möglichkeiten)
Komponenten innerhalb einer Menge N_j haben dabei alle den gleichen Wert. Zur Auswahl der k verschiedenen Werte aus M zu $N_1, \ldots, N_k$ gibt es $m(m-1)\cdots(m-k+1) = m^{\underline{k}}$ Möglichkeiten (k-Permutationen!). Aus der Summenformel von Satz 1.1 folgt daher

$$m^n = \sum_{k=0}^{n} S_{nk}\, m^{\underline{k}}.$$

Mit $m^0 = 1 = S_{00} m^{\underline{0}}$ ist dabei auch der Fall $n = 0$ abgedeckt. Aus dieser Identität folgt, dass das *Polynom*

$$x^n - \sum_{k=0}^{n} S_{nk}\, x^{\underline{k}}, \quad x \in \mathbb{C},$$

vom Grad $n \in \mathbb{N}_0$ auf allen natürlichen Zahlen verschwindet, also mehr als n Nullstellen hat. Nach dem Fundamentalsatz der Algebra ist es daher identisch Null. ■

Mit der Rekursion aus Teil a) können die Zahlen wieder berechnet werden:

$k=$	0	1	2	3	4	5	6	
$n=0$	1							
1	0	1						
2	0	1	1					
3	0	1	3	1				
4	0	1	7	6	1			
5	0	1	15	25	10	1		
6	0	1	31	90	65	15	1	
7	0	1	63	301	350	140	21	1

$\boxed{S_{nk}}$

Für einige Indexwerte lassen sich Formeln für die Stirlingzahlen angeben:

$S_{n1} = S_{nn} = 1$, da $S_{n1} = S_{n-1,0} + S_{n-1,1} = S_{n-1,1}$ und $S_{nn} = S_{n-1,n-1} + n \cdot 0$.

$S_{n2} = 2^{n-1} - 1$, da $S_{n2} = S_{n-1,1} + 2S_{n-1,2} = 1 + 2S_{n-1,2}$.

$S_{n,n-1} = \binom{n}{2} = S_{n-1,n-2} + (n-1)S_{n-1,n-1} \overset{Ind.}{=} \binom{n-1}{2} + n - 1$.

Bemerkung: Nach Satz 1.8b sind die S_{nk} Koeffizienten eines Basiswechsels, da die einfachen Potenzfunktionen (Monome) x^n durch die Basis $\{x^{\underline{k}}\}$ der Faktoriellen (*Newton-Polynome*, vgl. Satz 2.2) dargestellt werden. Den umgekehrten Wechsel beschreibt

Defin. 1.9 *Die* Stirling-Zahlen 1.Art *sind die Koeffizienten s_{nk} in der Darstellung*

$$x^{\underline{n}} = \sum_{k=0}^{n} (-1)^{n-k} s_{nk} x^k, \quad x \in \mathbb{C},\ n \in \mathbb{N}_0 .$$

Triviale Werte sind dabei: $s_{00} := 1$, $s_{nk} = 0$ $(k > n)$ und

$$s_{n0} = 0 \ (n > 0), \text{ da dann } \lim_{x \to 0} x^{\underline{n}} = 0.$$

Auch für die Stirlingzahlen 1. Art gibt es eine Rekursionsformel und einen kombinatorischen Hintergrund, für den eine Anwendung auf Sortierverfahren diskutiert wird.

Satz 1.10 *Es gilt die Rekursion*

$$s_{nk} = s_{n-1,k-1} + (n-1)s_{n-1,k}, \quad k, n \in \mathbb{N}.$$

Beweis Induktion über n: Für $n = 0$ ist $x^{\underline{0}} = x^0 = 1$, also $s_{00} = 1$ und für $n = 1$ ist $x^{\underline{1}} = 1 \cdot x^1 + 0 \cdot x^0$. Für $n > 1$ gilt

$$\begin{aligned} x^{\underline{n}} &= x^{\underline{n-1}}(x - n + 1) = x \cdot x^{\underline{n-1}} - (n-1)x^{\underline{n-1}} \\ &\overset{I.V.}{=} \sum_{j=0}^{n-1} (-1)^{n-1-j} s_{n-1,j} x^{j+1} + \sum_{k=1}^{n} (-1)^{n-k} (n-1) s_{n-1,k} x^k \end{aligned}$$

$$= \sum_{k=1}^{n}(-1)^{n-k}s_{n-1,k-1}x^k + \ldots$$

$$= \sum_{k=1}^{n}(-1)^{n-k}\Big(s_{n-1,k-1} + (n-1)s_{n-1,k}\Big)x^k \overset{Def}{=} \sum_{k=0}^{n}(-1)^{n-k}s_{nk}x^k.$$

In der 2. Zeile wurde der Summationsbereich um die trivialen Elemente $s_{n-1,0} = s_{n-1,n} = 0$ modifiziert und zur 3. Zeile erfolgte die Indexverschiebung $k = j + 1$. ∎

Wertetabelle: $\boxed{s_{nk}}$

$k =$	0	1	2	3	4	5	6	
$n = 0$	1							
1	0	1						
2	0	1	1					
3	0	2	3	1				
4	0	6	11	6	1			
5	0	24	50	35	10	1		
6	0	120	274	225	85	15	1	
7	0	720	1764	1624	735	175	21	1

Spezialfälle: $s_{n1} = (n-1)!$, $s_{n,n-1} = \binom{n}{2}$, $s_{nn} = 1$.

Bemerkung: Faßt man die Stirlingzahlen bis zu einer gewissen Ordnung m als (untere Dreieck-) Matrizen auf, vgl. die Tabellen, dann zeigen Satz 1.8 und die letzte Definition, dass die Matrizen $\Big(S_{nk}\Big)_{n,k=0}^{m}$ und $\Big((-1)^{n-k}s_{nk}\Big)_{n,k=0}^{m}$ zueinander invers sind, da sie jeweils den umgekehrten Basiswechsel beschreiben. Daher gilt

$$\sum_{k=0}^{n} S_{nk}(-1)^{k-j}s_{kj} = \delta_{nj} = \sum_{k=0}^{n}(-1)^{n-k}s_{nk}S_{kj}.$$

Kombinatorischer Hintergrund n-Permutationen:

Die Menge aller n-Permutationen von $N = \{1, \ldots, n\}$ entspricht der Menge aller *bijektiven* Abbildungen $N \to N$. Schreibweise:

$$\pi : N \to N \quad \text{als} \quad \begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ \pi(1) & \pi(2) & \pi(3) & \ldots & \pi(n) \end{pmatrix}.$$

Wichtige spezielle Permutationen sind die *Zyklen* (zyklische Vertauschungen).

Defin. 1.11 *Ein Zyklus der Länge $k \le n$, geschrieben als $(a_1, a_2, \ldots, a_k)$ ist die Permutation $\pi : N \to N$ mit*

$$\pi(a_1) = a_2,\ \pi(a_2) = a_3, \ldots, \pi(a_k) = a_1,\ \textit{sowie } \pi(i) = i \textit{ für } i \notin \{a_1, \ldots, a_k\}.$$

Unveränderte Elemente einer Permutation π nennt man wie üblich Fixpunkte.

Fixpunkte können als Zyklen der Länge 1 interpretiert werden. Von zentraler Bedeutung sind die Zyklen, da sich jede Permutation als *Produkt disjunkter Zyklen* darstellen läßt. Die Zyklendarstellung ist aber nur eindeutig, wenn man bestimmte Standardisierungen vornimmt [Knuth 1, S. 176].

Beispiel: $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 5 & 8 & 3 & 1 & 9 & 7 & 6 & 2 & 4 \end{pmatrix} = (1,5,9,4)(2,8)(6,7)(3)$. π besteht also aus einem Fixpunkt, 2 Vertauschungen und einem Viererzyklus $(1,5,9,4) = (5,9,4,1) = \ldots$.

Die kombinatorische Bedeutung der Stirling-Zahlen 1.Art behandelt der folgende Satz. Bei der Anzahl der Zyklen werden dabei die Fixpunkte (Einerzyklen) mitgezählt.

Satz 1.12 *Die Anzahl der Permutationen von $N = \{1, \ldots, n\}$ mit genau k Zyklen ist s_{nk}.*

Beweis Ähnlich zu Satz 1.8a, zunächst gilt $s_{0k} = 0\,(k > 0)$, $s_{n0} = 0\,(n > 0)$, setze $s_{00} := 1$. Für $n > 0$ wird wieder ein Element $a \in N$ fest gewählt und es sei $0 \le k \le n$.

Fall 1: (a) ist Fixpunkt der Permutationen. Die Permutationen der restlichen Elemente bestehen dann aus $k-1$ Zyklen. Daher tritt dieser Fall $s_{n-1,k-1}$ mal auf.

Fall 2: a tritt in einem nichttrivialen Zyklus auf, d.h., $N \setminus \{a\}$ besteht auch aus k Zyklen. Dann kann a in der Zyklendarstellung an jeder von $n-1$ Positionen auftreten (der Eintrag am Ende eines Zyklus ergibt keine neue Permutation). Dieser zweite Fall tritt $(n-1)s_{n-1,k}$ mal auf.
Die Rekursion stimmt also mit der aus der Definition der s_{nk} überein. ■

Anwendung Permutationen treten in der Informatik auf, wenn Daten am Platz umgeordnet werden, etwa beim Bewegen eines Blocks innerhalb eines Speicherbereichs oder beim Sortieren. Für den erforderlichen Bewegungsaufwand (große Einheiten auf Platten?!) ist die Anzahl und Länge der Zyklen der erforderlichen Permutation entscheidend. Für das folgende kann man davon ausgehen, dass die gewünschte Reihenfolge der Datensätze (Records) durch ein Indexfeld $p[1..n]$ der Ausgangspositionen gegeben ist (d.h. die inverse Permutation). Beim Bewegen eines Blocks ist das klar, beim Sortieren kann p durch Vorsortieren der Schlüssel bestimmt werden. Die minimale Anzahl von Bewegungen im Datenfeld $x[1..n]$ wird durch folgenden Algorithmus verwendet, der die Permutation durch Verfolgung der einzelnen Zyklen erzeugt. Die Zahl der Bewegungen ist dabei

$$\Big(\text{Anzahl der nichttrivialen Zyklen}\Big) + \Big(\text{Summe der Zyklenlängen} > 1\Big).$$

Algorithmus P: Umordnung $x[1..n]$ als $x[i] := x[p[i]]$, $i = 1, \ldots, n$.

```
i := 1;
while i ≤ n do begin
  if p[i] <> i then              Fixpunkte ignorieren
  begin h := x[i]; k := i;       einen Platz freimachen
    repeat j := k; k := p[j];    Zyklus verfolgen
      x[j] := x[k]; p[j] := j;   umspeichern & markieren
    until k = i;                 Zyklus fertig
    x[j] := h;                   Zyklus schließen
  end; i := i + 1;               nächster Zyklus
end;
```

Zur Bestimmung des Aufwands für diesen Algorithmus wird die *mittlere* Anzahl B_n der ausgeführten Bewegungen im Algorithmus **P** bestimmt. Die Mittelung wird dabei über alle Permutationen von N durchgeführt. Da pro Zyklus nur eine zusätzliche Bewegung erforderlich ist, gilt

$$B_n = \frac{1}{n!}\sum_{\pi}\Big((\text{Anzahl nichttriv.Zyklen}) + (\text{Summe Zyklenlängen} > 1)\Big).$$

Die Gesamtlänge aller Zyklen ist n, daher kann der zweite Summand auch durch $n - f(\pi)$ ersetzt werden, wobei $f(\pi)$ die Anzahl der Fixpunkte der Permutation π ist. Der erste Summand kann analog umformuliert werden als *(Anzahl der nichttriv.Zyklen)=(Anzahl aller Zyklen)*$-f(\pi)$. Daher ist also auch

$$B_n = \frac{1}{n!}\sum_{\pi}\Big((\text{Anzahl aller Zyklen}) + n - 2f(\pi)\Big).$$

Nach Satz 1.12 ist die Anzahl der Permutationen mit genau k Zyklen bekannt, es folgt daher

$$\begin{aligned} B_n &= \frac{1}{n!}\Big(\sum_{k=0}^{n} ks_{nk} + n!\,n - 2\sum_{\pi} f(\pi)\Big) \\ &= n + \frac{1}{n!}\sum_{k=0}^{n} ks_{nk} - 2\underbrace{\frac{1}{n!}\sum_{\pi} f(\pi)}_{=1}. \end{aligned}$$

Der unterklammerte Ausdruck ist die mittlere Anzahl der Fixpunkte und hat den Wert 1 ([Knuth-1,S.177]). Da er mit dem negativen Vorzeichen auftritt, spielt er für die Abschätzung auch keine Rolle. Hauptproblem ist die Berechnung der ersten Summe. Hierzu wird jetzt die Technik der *Erzeugenden Funktionen* (vgl. §2.3) angewendet. Die gesuchte Summe hat nämlich den Wert $\sum_k ks_{nk} = G'(1)$, wenn man das mit den Koeffizienten s_{nk} gebildete Polynom betrachtet. Nach Definition ist dieses

$$G(x) := \sum_{k=0}^{n} s_{nk}x^k = \sum_{k=0}^{n}(-1)^k s_{nk}(-x)^k = (-1)^n(-x)^{\underline{n}}.$$

Aus Satz 1.5b folgt $G(x) = (-1)^n(-x)^{\underline{n}} = x^{\overline{n}} = x(x+1)\cdots(x+n-1)$ und $G(1) = n!$. Nach der Produktregel gilt

$$\begin{aligned} G'(x) &= (x+1)\cdots(x+n-1) + x(x+2)\cdots(x+n-1) + \ldots + x\cdots(x+n-2) \\ &= G(x)\Big(\frac{1}{x} + \frac{1}{x+1} + \ldots + \frac{1}{x+n-1}\Big) \end{aligned}$$

und an der Stelle $x = 1$ also

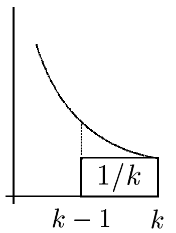
$$G'(1) = G(1)\Big(1 + \frac{1}{2} + \ldots + \frac{1}{n}\Big) = n!\,H_n$$

Für die hier eingeführte harmonische Zahl gilt $H_n \leq 1 + \ln n$ (s.u.). Als Ergebnis folgt, dass die Anzahl der Bewegungen im Algorithmus **P** im Mittel

$$B_n = n + H_n - 2 \leq n + \ln n - 1$$

ist. Im Vergleich dazu benötigen die besten Sortierverfahren (Quicksort) im Mittel $n \cdot \ln n$ Bewegungen, also das $\ln n$-fache (z.B. $n = 1024$: $\ln n \doteq 6.93$). Wenn also der Bewegungsaufwand erheblich teurer ist als der Schlüsselvergleich, lohnt sich ein Vorsortieren der Schlüssel und anschließende Ausführung von Algorithmus **P**.

Defin. 1.13 *Für $n \in \mathbb{N}$ ist die* harmonische Zahl $\quad H_n := \sum\limits_{k=1}^{n} \frac{1}{k}$.



H_n kann einfach mit dem Integralkriterium abgeschätzt werden. Da $\frac{1}{k} \leq \int_{k-1}^{k} \frac{dx}{x} = \ln k - \ln(k-1)$ gilt für $k > 1$, folgt

$$H_n = 1 + \sum_{k=2}^{n} \frac{1}{k} \leq 1 - \ln 1 + \ln 2 - \ln 2 + \ldots + \ln n = 1 + \ln n.$$

Mit H_n läßt sich eine weitere Reihe der Stirlingzahlen bestimmen, es gilt $s_{n,2} = (n-1)!H_{n-1}$. Denn mit Satz 1.10 und $s_{n1} = (n-1)!$ folgt

$$\frac{s_{n2}}{(n-1)!} = \frac{s_{n-1,1}}{(n-1)!} + \frac{(n-1)s_{n-1,2}}{(n-1)!} = \frac{1}{n-1} + \frac{s_{n-1,2}}{(n-2)!} = \frac{1}{n-1} + \frac{1}{n-2} + \ldots + 1 = H_{n-1}.$$

1.4 Erzeugung aller k-Stichproben

Als Grundlage einfacher Abzähl- oder Optimierungsalgorithmen im Bereich der k-Stichproben können Verfahren zur Erzeugung aller möglichen Elemente dienen. Zunächst ist es dabei hilfreich, eine Reihenfolge durch Anordnung der Figuren zu definieren. Dazu werden alle k-Stichproben als k-Tupel geschrieben, als Grundmenge (Alphabet) wird $N = \{1, \ldots, n\}$ zugrundegelegt.

Defin. 1.14 *Das k-Tupel $a = (a_1, \ldots, a_k)$ heißt* lexikographisch kleiner *als $b = (b_1, \ldots, b_k)$ (kurz $a < b$), wenn*

$$\exists\, m \in \{1, \ldots, k\} : \quad a_1 = b_1, \ldots, a_{m-1} = b_{m-1},\ a_m < b_m.$$

Diese Definition entspricht der üblichen Sortierreihenfolge in Lexika, Telefonbüchern und bei Dezimalzahlen.

Die Gesamtmenge aller zu einer Vorschrift gehörigen k-Tupel kann dann, z.B., in aufsteigender Reihenfolge erzeugt werden. Dazu beginnt man mit dem lexikalisch kleinsten (zulässigen) Element. Der Übergang zum nächstgrößeren kann durch die Nachbarfunktion ON erklärt werden, wobei $b := ON(a)$ das zu a nächstgrößere Element angibt. Bei den vier Stichprobenarten ist das entscheidende Problem bei der Konstruktion von ON die Bestimmung des Index m mit $(a_1, \ldots, a_{m-1}) = (b_1, \ldots, b_{m-1})$ für $b = ON(a)$.

1. k-Worte: Dies ist der einfachste Fall, Startwort ist $(1, \ldots, 1)$. $b = ON(a)$ entspricht dem Zählen mit Übertrag:
$$m = \max\{i : \ a_i < n\}, \quad b_m := a_m + 1, \ (b_{m+1}, \ldots, b_k) = (1, \ldots, 1).$$
Bemerkung: Dies ist zwar die übliche Zählweise, hat aber für Traversen-Anwendungen den Nachteil, dass sich beim Übertrag sehr viele Stellen gleichzeitig ändern. Abzählmethoden in $\{0, 1\}^k$ (Traversen im 'Hypercube'/Parallelrechner), bei denen sich in jedem Schritt nur eine Stelle ändert, werden durch Gray-Codes realisiert (stetige Hypercube-Traversen).

2. n-Permutationen ($k = n$): Startwort ist die wachsende Folge $(1, 2, \ldots, n)$. Zur Motivation wird das Zählverfahren am Beispiel $n = 4$ erläutert.
$$12\bar{3}\underline{4},\ 1\bar{2}\underline{43},\ 13\bar{2}\underline{4},\ 1\bar{3}\underline{42},\ 14\bar{2}\underline{3},\ \bar{1}\underline{432},\ 21\bar{3}\underline{4}, \ldots, 4321$$
Die Worte ändern sich jeweils ab der überstrichenen Position $(= m)$. Diese Position ist dadurch definiert, dass der folgende unterstrichene Teil die längste *fallende* Ziffernfolge am Wortende ist. Der überstrichene Wert wird dabei mit dem nächstgrößeren aus dem fallenden Rest getauscht. Also ist $b = ON(a)$ gegeben durch
$$\begin{aligned} m = \max\{i : a_i < a_{i+1}\}, \qquad & b_m := \min\{a_i : a_i > a_m, i > m\}, \\ & (b_{m+1}, \ldots, b_k) = \mathrm{sort}_{\nearrow}(\{a_m, \ldots, a_n\} \setminus \{b_m\}). \end{aligned}$$
Anmerkung: Die aufsteigende Sortierung ist hier einfach durchzuführen, da nach Definition von m gilt $a_{m+1} > \ldots > a_n$. Daher ist das Restwort nur zu spiegeln und a_m mit dem nächstgrößeren Wert darin zu tauschen. Dazu wird die Stelle l mit $a_l > a_m > a_{l+1}$ bestimmt und dann die Elemente umgeordnet:
$$\begin{array}{llllllll} a = (a_1, \ldots, a_{m-1}, & a_m, & a_{m+1} & \longleftarrow & a_l & \longleftarrow & a_n) \\ b = (a_1, \ldots, a_{m-1}, & a_l, & a_? & \longrightarrow & a_m & \longrightarrow & a_?) \end{array}$$
Durch Streichen der letzten Stelle sind auch die $n-1$-Permutationen abgedeckt.

Bemerkung: Wie im letzten Fall existieren bei Verzicht auf die lexikalische Reihenfolge Verfahren, die alle Permutationen nur durch einzelne Vertauschungen erzeugen.

3. k-Permutationen, $k < n - 1$: Startwort $(1, 2, \ldots, k)$. Erhöht wird die am weitesten rechts stehende Stelle a_m, für die nicht alle größeren Werte aus N schon links von ihr vertreten sind. Rechts von a_m werden möglichst kleine Werte gewählt. Formal definiert man dazu
$$A_i := \{a \in N : a > a_i, a \notin \{a_1, \ldots, a_{i-1}\}\}, \quad 2 \leq i \leq k,$$
$A_1 := \{a \in N : a > a_1\}$. Damit ergibt sich ON aus
$$\begin{aligned} & m := \max\{i : \ A_i \neq \emptyset\}, \ b_m := \min A_m, \\ & (b_{m+1}, \ldots, b_k) := \text{min-sort}_{\nearrow}(N \setminus \{a_1, \ldots, a_{m-1}, b_m\}). \end{aligned}$$

Die letzte Operation bedeutet Auswahl der kleinsten Elemente, steigend sortiert.

Beispiel: Die 3-Permutationen aus $\{1, \ldots, 5\}$, ihre Anzahl ist $5^{\underline{3}} = 5 \cdot 4 \cdot 3 = 60$:

$$12\bar{3},\ 12\bar{4},\ 1\bar{2}5,\ 13\bar{2},\ 13\bar{4},\ 1\bar{3}5,\ 14\bar{2},\ 14\bar{3},\ 1\bar{4}5,\ 15\bar{2},\ 15\bar{3},\ \bar{1}54,\ 21\bar{3},\ 21\bar{4},\ 2\bar{1}5,\ 231, \ldots$$

4. k-Teilmengen ($k < n$) werden als k-Tupel mit monoton wachsenden Elementen interpretiert, das Startwort ist $(1, 2, \ldots, k)$. Da das größte k-Tupel $(n-k+1, \ldots, n-1, n)$ ist und die Monotonie erhalten bleiben muß, wird jeweils die letzte Stelle inkrementiert, für die noch $a_i < n - k + i$ gilt:

$$m = \max\{i : a_i < n-k+i\}, \quad (b_m, b_{m+1}, \ldots, b_k) := (a_m + 1, a_m + 2, \ldots, a_m + k - m + 1).$$

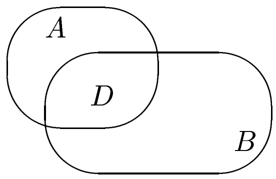
Beispiel: Alle 3-Teilmengen aus $\{1, \ldots, 5\}$, die Anzahl ist $\binom{5}{3} = 10$.

$$12\bar{3},\ 12\bar{4},\ 1\bar{2}5,\ 13\bar{4},\ 1\bar{3}5,\ \bar{1}45,\ 23\bar{4},\ 2\bar{3}5,\ \bar{2}45,\ 345.$$

In allen Fällen endet die Zählung, wenn das jeweils definierte m nicht existiert.

1.5 Inklusion-Exklusion

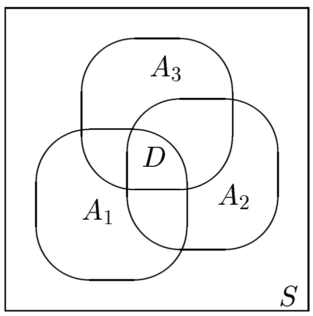
Hier wird die allgemeine Version der Summenformel aus Satz 1.1a für nicht-disjunkte Vereinigungen behandelt und einige Anwendungen dazu.



Beispiel: 1) Bei Vereinigung von zwei nicht-disjunkten Teilmengen A, B werden in der Summe $|A| + |B|$ die Elemente aus dem Durchschnitt $D := A \cap B$ offensichtlich doppelt gezählt. Daher ist

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

2) Problem: *Wieviele Zahlen zwischen 1 und 30 sind teilerfremd zu 30?*
Die beschriebene Menge wird S genannt. Da 30 die Faktorzerlegung $30 = 2 \cdot 3 \cdot 5$ besitzt, kommen keine der Zahlen in Frage, die in einer der Mengen



$$\begin{aligned} A_1 &= \{2i : \ i = 1, \ldots, 15\}, \\ A_2 &= \{3i : \ i = 1, \ldots, 10\}, \\ A_3 &= \{5i : \ i = 1, \ldots, 6\} \end{aligned}$$

liegen. Die Menge S ist also das Komplement von $A_1 \cup A_2 \cup A_3$.

Eine naive Verallgemeinerung der letzten Formel führt für $|A_1 \cup A_2 \cup A_3|$ auf die Anzahl

$$|A_1 \cup A_2 \cup A_3| \stackrel{?}{=} |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3|.$$

Dabei wurde also die Doppelzählung für Elemente aus den Schnitten $A_i \cap A_j$ korrigiert. Allerdings werden Elemente aus dem gemeinsamen Schnitt $D := A_1 \cap A_2 \cap A_3$, die in $|A_1| + |A_2| + |A_3|$

dreimal gezählt wurden, jetzt auch dreimal wieder abgezogen. Daher muß die letzte Formel noch einmal um $|D|$ korrigiert werden:

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|.$$

Für die einleitende Frage ergibt sich daher folgende Antwort: $|A_1| = 15$, $|A_2| = 10$, $|A_3| = 6$, $|A_1 \cap A_2| = |\{6i : i = 1,..,5\}| = 5$, $|A_2 \cap A_3| = |\{15i : i = 1,2\}| = 2$, $|A_1 \cap A_3| = 3$, $|D| = 1$. Damit folgt

$$|A_1 \cup A_2 \cup A_3| = 15 + 10 + 6 - 5 - 2 - 3 + 1 = 22, \text{ d.h., } |S| = 30 - 22 = 8.$$

Den allgemeinen Fall behandelt der folgende Satz, der unter den Bezeichnungen 'Ein- Ausschalt-Formel', 'Siebtheorem', 'Inklusions-Exklusions-Prinzip' bekannt ist.

Satz 1.15 *(Ein-Ausschalt-Formel von Sylvester) Für jede Familie $\{A_i : i = 1, \ldots, n\}$, $n \in \mathbb{N}$, endlicher Mengen gilt*

$$\begin{aligned}
\Big|\bigcup_{i=1}^{n} A_i\Big| &= \sum_{i=1}^{n} |A_i| - \sum_{1 \le i_1 < i_2 \le n} |A_{i_1} \cap A_{i_2}| + \sum_{1 \le i_1 < i_2 < i_3 \le n} |A_{i_1} \cap A_{i_2} \cap A_{i_3}| \pm \ldots \\
&\quad + (-1)^{n-1} |A_1 \cap \ldots \cap A_n| \\
&= \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \le i_1 < .. < i_k \le n} |A_{i_1} \cap \ldots \cap A_{i_k}|.
\end{aligned}$$

Bemerkung: Im k-ten Summanden wird über alle k-Teilmengen $\{i_1, \ldots, i_k\}$ der Indexmenge N summiert, vgl. §1.4-4.

Beweis Induktion über n. Der Fall $n = 1$ gilt trivialerweise. Ansonsten wird die Basisregel

$$|A \cup B| = |A| + |B| - |A \cap B| \tag{4}$$

angewendet. Die EA-Formel gelte nun für n, bei Vereinigung von $n + 1$ Mengen gilt dann

$$\begin{aligned}
Z &:= \Big|\bigcup_{i=1}^{n+1} A_i\Big| = \Big|\Big(\bigcup_{i=1}^{n} A_i\Big) \cup A_{n+1}\Big| \\
&\overset{(4)}{=} \Big|\bigcup_{i=1}^{n} A_i\Big| + \Big|A_{n+1}\Big| - \Big|\Big(\bigcup_{i=1}^{n} A_i\Big) \cap A_{n+1}\Big| \\
&= \Big|\bigcup_{i=1}^{n} A_i\Big| + \Big|A_{n+1}\Big| - \Big|\bigcup_{i=1}^{n} (A_i \cap A_{n+1})\Big|.
\end{aligned}$$

Nun betreffen beide auftretenden Vereinigungen genau n Mengen, mit der Induktionsvoraussetzung folgt

$$\begin{aligned}
Z &= \sum_{k=1}^{n} (-1)^{k-1} \sum_{i_1 < .. < i_k \le n} |A_{i_1} \cap \ldots \cap A_{i_k}| + |A_{n+1}| \\
&\quad - \sum_{j=1}^{n} (-1)^{j-1} \sum_{i_1 < .. < i_j \le n} |A_{i_1} \cap \ldots \cap A_{i_j} \cap A_{n+1}|.
\end{aligned}$$

Der Summand $|A_{n+1}|$ ergänzt die erste Summe bei $k = 1$. In der letzten Summe stehen alle Durchschnitte von je $j + 1(=: k)$ Mengen mit $i_1 < \ldots < i_j \leq n < n + 1 =: i_{j+1}$. Im j-ten Summanden ist daher das Vorzeichen $(-1)^j$ korrekt. Außerdem treten hier alle Summanden auf, die in der ersten Summe bei $k = j + 1$ fehlen, nämlich die mit $i_k = n + 1$. ■

Falls die Größe der Durchschnittsmengen nur von der Anzahl der schneidenden Mengen abhängt, vereinfacht sich die Formel wesentlich.

Korollar *Für jedes $k \leq n \in N$ gelte*

$$|A_{i_1} \cap \ldots \cap A_{i_k}| = d_k \quad \text{für beliebige } 1 \leq i_1 < .. < i_k \leq n,$$

die Größe der k-Durchschnitte sei also d_k, unabhängig von der k-Teilmenge $\{i_1, \ldots, i_k\}$. Dann ist

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} d_k.$$

Beweis Für jedes k tritt der Summand d_k genau $\binom{n}{k}$ mal auf, dies ist die Anzahl der k-Teilmengen von N, vgl. Satz 1.3. ■

Anwendung Anzahl surjektiver Abbildungen: Es sei $Surj(M,N)$ die Menge der surjektiven Abbildungen $M \to N$, wobei (oBdA) $M = \{1, \ldots, m\}$, $N = \{1, \ldots, n\}$ und $m \geq n$ ist. In §1.2 wurde dazu festgestellt, dass bei jeder Abbildung $f \in Surj(M,N)$ die Familie der Urbildmengen $\{f^{-1}(b) : \ b \in N\}$ eine n-Partition von M bildet. Daher ist $|Surj(M,N)| = n!\, S_{mn}$.

Mit der EA-Formel wird dieser Wert nun explizit berechnet. Dazu wird das Komplement der gesuchten Menge $Surj(M,N)$ in der Menge aller Abbildungen, $Abb(M,N) = N^M$ dargestellt,

$$N^M \setminus Surj(M,N) = A_1 \cup \ldots \cup A_n.$$

Die Anzahl aller Abbildungen ist dabei $|N^M| = n^m$, vgl. §1.2. Die Mengen A_i werden definiert durch

$$A_i := \{f \in Abb(M,N) : i \notin f(M)\}, \quad i = 1, \ldots, n.$$

A_i enthält alle Abbildungen, bei denen $i \in N$ nicht als Bild auftritt. In den Durchschnitten solcher Mengen treten immer weniger Bildwerte auf, daher ist

$$|A_{i_1} \cap \ldots \cap A_{i_k}| = \Big|\Big\{f : M \to N \setminus \{i_1, \ldots, i_k\}\Big\}\Big| = (n-k)^m =: d_k.$$

Aus dem Korollar folgt nun

$$\begin{aligned} |Surj(M,N)| &= |N^M| - |A_1 \cup \ldots \cup A_n| = n^m - \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} (n-k)^m \\ &= \sum_{k=0}^{n} (-1)^k \binom{n}{n-k} (n-k)^m = \sum_{j=0}^{n} (-1)^{n-j} \binom{n}{j} j^m. \end{aligned}$$

Insbesondere gilt also die explizite Darstellung für die Stirlingzahlen 2.Art

$$S_{mn} = \frac{1}{n!}\sum_{j=0}^{n}(-1)^{n-j}\binom{n}{j}j^m. \tag{5}$$

Anwendung Anzahl fixpunktfreier n-Permutationen (Derangements) D_n. *Wieviele Möglichkeiten gibt es, Briefe in adressierte Umschläge zu stecken so, dass keiner richtig eingeordnet ist?*

Weiterer Hintergrund: Beim Sortierproblem in §1.3 (Algorithmus **P**) spielte die mittlere Zahl der Fixpunkte von Permutationen eine Rolle. Die Anzahl der Permutationen mit genau k Fixpunkten ist $\binom{n}{k}D_{n-k}$, da bei der Wahl der Fixpunkte $\binom{n}{k}$ Fälle möglich sind, der Rest aber fixpunktfrei sein muß.

Wie bei der vorherigen Anwendung wird das Komplement zur Menge aller Permutationen betrachtet, entfernt wird die Vereinigung der Mengen

$$A_i := \{\pi : \pi(i) = i\}, \quad i = 1,..,n,$$

wobei bei den Permutationen in A_i der Wert i Fixpunkt ist. Da bei Fixierung von k Werten $i_1,\ldots,i_k$ für die restlichen Werte $(n-k)!$ Permutationen übrig bleiben, gilt für die Durchschnitte

$$|A_{i_1}\cap\ldots\cap A_{i_k}| = (n-k)! =: d_k.$$

Mit dem Korollar folgt

$$\begin{aligned} D_n &= n! - |A_1\cup\ldots\cup A_n| = n! - \sum_{k=1}^{n}(-1)^{k-1}\binom{n}{k}(n-k)! \\ &= \sum_{k=0}^{n}(-1)^k\frac{n!}{k!} = n!\sum_{k=0}^{n}\frac{(-1)^k}{k!}. \end{aligned}$$

Die letzte Summe ist der Beginn der Reihe für e^{-1}. Für größere n ist daher die mittlerer Anzahl fixpunktfreier Permutationen

$$\frac{1}{n!}D_n \cong \frac{1}{e} \doteq 0.367879.$$

Für die ersten Werte erhält man

$n =$	1	2	3	4	5	6
$D_n =$	0	1	2	9	44	265
$n! =$	1	2	6	24	120	720
$D_n/n! \doteq$	0	0.5	0.3333	0.375	0.3666	0.3681

Im Briefbeispiel gibt es daher für große n eine Wahrscheinlichkeit von 63%, dass wenigstens einer der Briefe richtig ankommt. Allgemein ist die Anzahl der Permutationen mit genau k Fixpunkten

$$\binom{n}{k}D_{n-k} = \frac{n^{\underline{n-k}}}{(n-k)!}D_{n-k} = n^{\underline{n-k}}\sum_{j=0}^{n-k}(-1)^j\frac{1}{j!} \cong \frac{n^{\underline{n-k}}}{e}.$$

Zur (praktischen) Berechnung der Derangement-Zahlen gibt es aber auch einen anderen Ansatz, der im nächsten Paragraphen eine Rolle spielt:

Satz 1.16 *Die Anzahl D_n fixpunktfreier n-Permutationen (Derangements) ist*

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

Außerdem gilt mit $D_1 = 0,\ D_2 = 1$ die lineare Rekursionsformel

$$D_n = (n-1)(D_{n-1} + D_{n-2}), \quad n = 2, 3, \ldots .$$

Beweis Für $n > 2$ sei π eine fpf-Permutation und $i = \pi(1)$ bezeichne den ersten Bildwert $(i \neq 1)$. Fallunterscheidung:

Fall 1, $\pi(i) = 1$: Dann ist $(1, i)$ ein 2-Zyklus (Transposition), für die restlichen Werte $\{2, \ldots, n\} \setminus \{i\}$ gibt es D_{n-2} fpf Permutationen.

Fall 2, $\pi(i) \neq 1$: Durch Entfernen von i aus π und folgende Ergänzung

$$\sigma = \begin{pmatrix} 2 & \ldots & i-1 & 1 & i+1 & \ldots & n \\ \pi(2) & \ldots & \pi(i-1) & \pi(i) & \pi(i+1) & \ldots & \pi(n) \end{pmatrix}$$

entsteht eine Permutation von $N \setminus \{i\}$ mit $\sigma(1) = \pi(i) \neq 1$. Von den so konstruierten Permutationen σ sind D_{n-1} fixpunktfrei und auch die ursprüngliche mit $1 \mapsto i = \pi(1) \mapsto \pi(i)$ war daher eine fpf-Permutation von N.

Beide Fälle zeigen, dass für jedes $i \neq 1$ also $D_{n-1} + D_{n-2}$ fpf Permutationen existieren mit $\pi(1) = i$. Daher ist D_n das $(n-1)$-fache dieser Summe. ■

2 Folgen und Rekursionen

2.1 Summen- und Differenzenrechnung

Bei geeigneter Übertragung der Begriffe können viele Regeln der Differential- und Integralrechnung von Funktionen auf Differenzen bzw. Summen von Folgenwerten (Funktionswerten) übertragen werden.

Defin. 2.1 *Für Folgen $f : \mathbb{Z} \to \mathbb{C}$ bzw. Funktionen $f : \mathbb{R} \to \mathbb{C}$ werden folgende Abbildungen definiert*

$$\begin{array}{llll} S: & f \mapsto f(\cdot+1), & \textit{d.h.}, \ (Sf)(x) = f(x+1) & \textit{Schiebe-Operator/Translation}, \\ \Delta: & f \mapsto f(\cdot+1) - f(\cdot), & \textit{d.h.}, \ (\Delta f)(x) = f(x+1) - f(x), & \textit{Vorwärts-Differenzenoperator}, \\ \nabla: & f \mapsto f(\cdot) - f(\cdot-1), & \textit{d.h.}, \ (\nabla f)(x) = f(x) - f(x-1) & \textit{Rückwärts-Differenzenoperator}. \end{array}$$

Zur Herleitung von Rechenregeln sind die Darstellungen $\Delta = S - I$, $\nabla = I - S^{-1}$ hilfreich. Die in der Definition noch verwendete Klammer um die Bildfunktion wird dabei meist weggelassen, z.B., wie in

$$(\nabla f)(x) = \nabla f(x) = f(x) - S^{-1}f(x) = f(x) - f(x-1).$$

Außerdem wird bei Folgen das Argument weiterhin als Index geschrieben, die gleiche Regel lautet bei (a_n) etwa

$$\nabla a_k = a_k - S^{-1}a_k = a_k - a_{k-1}.$$

Die erste Summationsregel ist eine Analogie zum Hauptsatz der Differential-/Integral-Rechnung und betrifft Teleskopsummen:

$$\begin{aligned} \sum_{k=0}^{n-1} \Delta f(k) &= f(1) - f(0) + f(2) - f(1) + \ldots + f(n) - f(n-1) = f(n) - f(0) =: f\Big|_0^n, \quad (1) \\ \sum_{k=1}^{n} \nabla f(k) &= f(1) - f(0) + f(2) - f(1) + \ldots + f(n) - f(n-1) = f(n) - f(0) =: f\Big|_0^n. \end{aligned}$$

<u>Achtung</u>: In dieser und anderen Formeln sind die Auswertungspunkte bei $f\Big|$ nicht identisch mit den Summationsgrenzen.

Eine Analogie zur partiellen Integration ist

Satz 2.2 *Für $u, v : \mathbb{R} \to \mathbb{C}$ (bzw. $\mathbb{Z} \to \mathbb{C}$) gilt die Regel der* partiellen Summation

$$\begin{aligned} \sum_{k=0}^{n-1} u(k)\Delta v(k) &= uv\Big|_0^n - \sum_{k=0}^{n-1} (\Delta u(k))Sv(k) \\ &= u(n)v(n) - u(0)v(0) - \sum_{k=0}^{n-1} (\Delta u(k))v(k+1) \end{aligned}$$

Beweis Die Behauptung folgt als Teleskopsumme (1) aus der Produktregel

$$\begin{aligned}\Delta(u(k)v(k)) &= u(k+1)v(k+1) - u(k)v(k) = (u(k+1) - u(k))v(k+1) + u(k)(v(k+1) - v(k)) \\ &= (\Delta u(k))Sv(k) + u(k)\Delta v(k). \quad \blacksquare\end{aligned}$$

Beide Regeln sind dann von Interesse, wenn man genügend viele Paare von Funktionen/Folgen kennt mit $f = \Delta u$. Man nennt u dabei eine *diskrete Stammfunktion* von f.

Bei der Differentiation waren die Regeln für Monome x^m besonders einfach, $(x^m)' = mx^{m-1}$, $m \in \mathbb{N}$. Bei Differenzenbildung gilt dies für die in Defin. 1.4 eingeführten Faktoriellen:

$$\begin{aligned}\Delta x^{\underline{m}} &= (x+1)^{\underline{m}} - x^{\underline{m}} = (x+1)x^{\underline{m-1}} - x^{\underline{m-1}}(x-m+1) = mx^{\underline{m-1}} \\ \nabla x^{\overline{m}} &= x^{\overline{m}} - (x-1)^{\overline{m}} = x^{\overline{m-1}}(x+m-1) - (x-1)x^{\overline{m-1}} = mx^{\overline{m-1}}.\end{aligned}$$

Insbesondere reduziert sich auch bei Differenzbildung der Polynomgrad um eins. Durch Umkehrung folgt mit (1) die Summationsregel

$$\sum_{k=0}^{n-1}(a+k)^{\underline{m}} = \sum_{k=0}^{n-1}\Delta\frac{(a+k)^{\underline{m+1}}}{m+1} = \frac{x^{\underline{m+1}}}{m+1}\Big|_{x=a}^{x=a+n} = \frac{(a+n)^{\underline{m+1}} - a^{\underline{m+1}}}{m+1}.$$

Mit Hilfe der geometrischen Summe erhält man für festes $b \in \mathbb{R}$ die

$$\begin{array}{llcll}\text{Differenzen-Regel} & \Delta b^x & = & b^{x+1} - b^x & = (b-1)b^x, \\ \text{Summations-Regel} & \sum\limits_{k=0}^{n-1} b^{a+k} & = & & = \dfrac{b^x}{b-1}\Big|_{x=a}^{x=a+n}.\end{array}$$

In der Analysis war hierbei der Fall $b = e$ besonders einfach, jetzt ist es der Fall $b = 2$:

$$\Delta 2^x = 2^x, \qquad \sum_{k=0}^{n-1} 2^{a+k} = 2^x\Big|_{x=a}^{x=a+n}.$$

Anwendung 1) Arithmetische Summen, Satz 1.8 zeigt:

$$\begin{aligned}\sum_{k=0}^{n-1} k^m &= \sum_{k=0}^{n-1}\sum_{i=0}^{m} S_{mi}k^{\underline{i}} = \sum_{i=0}^{m} S_{mi}\sum_{k=0}^{n-1} k^{\underline{i}} = \sum_{i=0}^{m} S_{mi}\frac{1}{i+1}x^{\underline{i+1}}\Big|_0^n \\ &= \sum_{i=0}^{m} S_{mi}\frac{n^{\underline{i+1}}}{i+1}.\end{aligned}$$

2) Gemischt arithmetisch-geometrische Summe $\sum k2^k$. In Satz 2.2 wähle man $u(x) = x$, $v(x) = 2^x$, dann gilt $2^x = \Delta 2^x$, $\Delta u = 1$. Es folgt

$$\begin{aligned}\sum_{k=0}^{n-1} k2^k &= x2^x\Big|_0^n - \sum_{k=0}^{n-1}(\Delta u(k))(Sv(k)) = n2^n - \sum_{k=0}^{n-1} 1 \cdot 2^{k+1} = n2^n - 2^{x+1}\Big|_0^n \\ &= n2^n - 2^{n+1} + 2 = (n-2)2^n + 2.\end{aligned}$$

3) Die Regeln c), d) in Satz 1.5 für Binomialkoeffizienten sind Spezialfälle der Regeln für $x^{\underline{m}}$:

$$\Delta\binom{x}{m} = \binom{x}{m-1}, \qquad \sum_{k=0}^{n-1}\binom{k}{m} = \binom{x}{m+1}\Big|_0^n = \binom{n}{m+1}.$$

Höhere Differenzen werden induktiv definiert,

$$\Delta^n f := \Delta(\Delta^{n-1} f), \quad n \in \mathbb{N}.$$

Auch hierfür ergeben sich besonders einfache Regeln bei den Faktoriellen, $\Delta^n x^{\underline{m}} = m\Delta^{n-1}x^{\underline{m-1}} = \ldots = m(m-1)\cdots(m-n+1)x^{\underline{m-n}}$, also

$$\Delta^n x^{\underline{m}} = m^{\underline{n}}\, x^{\underline{m-n}}. \tag{2}$$

Man beachte, dass in der Formel nur Faktorielle auftreten, für $n > m$ ist das Ergebnis dabei null. Die genaue Gestalt der höheren Differenz kann durch formale Anwendung des Binomischen Satzes auf $\Delta^n = (S - I)^n$ berechnet werden. Für Polynome f gibt es auch eine, zum Satz von Taylor analoge, Umkehr-Beziehung.

Satz 2.3 *a) Für $n \in \mathbb{N}$ und $f : \mathbb{R} \to \mathbb{C}$ gilt*

$$\Delta^n f(0) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} f(k).$$

b) Sei f ein Polynom vom Grad n. Dann gilt die Newton-Darstellung

$$f(x) = \sum_{k=0}^{n} \binom{x}{k} \Delta^k f(0) = \sum_{k=0}^{n} \frac{\Delta^k f(0)}{k!}\, x^{\underline{k}}, \quad x \in \mathbb{R}.$$

Beweis a) Die formale Anwendung der binomischen Formel ergibt

$$\Delta^n f(x) = (S - I)^n f(x) = \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} S^k f(x) = \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} f(x+k).$$

b) Nach §1.3 ist klar, dass die Faktoriellen $x^{\underline{k}}$, $k = 0, \ldots, n$, eine Basis des Raums aller Polynome vom Maximalgrad n bilden. Daher existiert eine Darstellung $f(x) = \sum_{i=0}^{n} b_i x^{\underline{i}}$. Bei Anwendung der Differenzenregel (2) darauf erhält man

$$\Delta^k f(x) = \sum_{i=k}^{n} b_i i^{\underline{k}}\, x^{\underline{i-k}} \quad \Rightarrow \quad \Delta^k f(0) = b_k\, k!\,,$$

da $x^{\underline{i-k}}$ nur für $k = i$ im Punkt Null nicht verschwindet. ■

Beispiel: Die Formeln für die ersten höheren Differenzen sind $\Delta^2 f(0) = \Delta(f(1) - f(0)) = f(2) - 2f(1) + f(0)$, $\Delta^3 f(0) = f(3) - 3f(2) + 3f(1) - f(0)$.

Anwendung Identifikation von Folgen, durch Differenzenbildung erkennt man, dass das Bildungsgesetz der gegebenen Folge (f_n) ein Polynom dritten Grades ist, $f_n = p(n)$. Mit Satz 2.3 kann sie daher aus den 4 Werten $f(0), \ldots, f(3)$ vollständig rekonstruiert werden.

$n =$	0	1	2	3	4	5	6	$\ldots$
$f_n =$	2	-1	0	11	38	87	164	$\ldots$
$\Delta f_n =$	-3	1	11	27	49	77	$\ldots$	
$\Delta^2 f_n =$	4	10	16	22	28	$\ldots$		
$\Delta^3 f_n =$	6	6	6	6	$\ldots$			
$\Delta^4 f_n =$			0					

Anwendung Bei $f(x) = x^n$ gilt nach Satz 1.8 $f(x) = x^n = \sum_{k=0}^n S_{nk} x^{\underline{k}}$. Nach Teil b) von Satz 2.3 folgt daher $\Delta^k f(0) = k! S_{nk}$ und Teil a) bestätigt daher die Formel (1.5)

$$S_{nk} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} f(j) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$

Bemerkung: Die Teile a) und b) von Satz 2.3 sind wieder zueinander inverse Formeln (vgl. Nachsatz zu Satz 1.10), mit $x = n$ gilt

$$\sum_{k=0}^{n} \binom{n}{k} (-1)^{k-j} \binom{k}{j} = \delta_{nj}, \quad j, n \in \mathbb{N}.$$

Dies bedeutet, dass die Matrizen von Binomialkoeffizienten bis zu einer bestimmten Ordnung zueinander invers sind, z.B.,

$$\begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 2 & 1 & & \\ 1 & 3 & 3 & 1 & \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ 1 & -2 & 1 & & \\ -1 & 3 & -3 & 1 & \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}.$$

Durch eine andere Aufteilung der Vorzeichen ergibt sich sogar eine selbst-inverse (involutorische) Matrix:

$$P := \begin{pmatrix} 1 & & & & \\ 1 & -1 & & & \\ 1 & -2 & 1 & & \\ 1 & -3 & 3 & -1 & \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix} = P^{-1}, \quad \text{d.h.,} \quad \begin{cases} a_n = \sum_{k=0}^n (-1)^k \binom{n}{k} b_k \ \forall n \\ \iff \\ b_n = \sum_{k=0}^n (-1)^k \binom{n}{k} a_k \ \forall n \end{cases}$$

2.2 Lineare Differenzengleichungen

Für die meisten bisher behandelten Größen existieren lineare Rekursionen (vgl. Binomialkoeff., Stirlingzahlen, Derangement-Zahlen). Für lineare Rekursionen von Folgen (einfach indizierte Größen) gibt es Aussagen zur Lösungsstruktur und ein Lösungsverfahren für wichtige Spezialfälle.

Beispiel: Tilgungs-/Zinsgleichung: Ein Startkapital K_0 vermehrt sich durch Verzinsung mit dem Faktor $a = 1 + p$ nach der (einstufigen) Zins-Rekursion

$$K_n = aK_{n-1}, \ n \geq 1 \quad \Rightarrow \quad K_n = a^n K_0.$$

Ein aufwendigeres Beispiel trat in Satz 1.16 für die Derangement-Zahlen auf $D_n = (n-1)(D_{n-1} + D_{n-2})$. Dies war eine lineare *zweistufige Rekursion.*

Beispiel: Wieviele Binärworte der Länge n gibt es, wo nie aufeinander folgende Nullen auftreten?

Sei W_n diese gesuchte Anzahl und $a = (a_1, \ldots, a_n)$ ein Wort der Länge n. Falls $a_n = 1$ ist, kann der Rest $(a_1, \ldots, a_{n-1})$ auf W_{n-1} Weisen gewählt werden. Für $a_n = 0$ ist dagegen $a_{n-1} = 1$ und nur $(a_1, \ldots, a_{n-2})$ ist wählbar mit W_{n-2} Möglichkeiten. Mit den Werten $W_1 = 2$ und $W_2 = 3$ folgt also

$$W_n = W_{n-1} + W_{n-2}, \ n \geq 3.$$

Diese Rekursion ist die gleiche wie bei der folgenden, klassischen Folge.

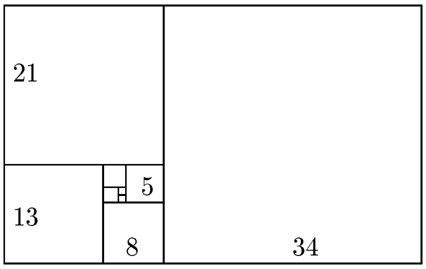
Beispiel: Die Fibonacci-Folge wird mit $F_0 := 0, \ F_1 := 1$, definiert durch die zweistufige Rekursion

$$F_n = F_{n-1} + F_{n-2}, \quad n \geq 2.$$

Wertetabelle:

$n =$	0	1	2	3	4	5	6	7	8	9	...
$F_n =$	0	1	1	2	3	5	8	13	21	34	...

Quadrat-Spirale:



Regel: An längere Rechteckseite wird Quadrat angesetzt

Der Standard-Lösungsansatz ist hier analog zur Zinsgleichung $F_n = c \cdot z^n$, $z \in \mathbb{C}$ (mit $cz \neq 0$). Durch Einsetzen in die Rekursion erhält man

$$F_n - F_{n-1} - F_{n-2} = (z^2 - z - 1)z^{n-2}c \stackrel{!}{=} 0 \quad \forall n \geq 2.$$

Um hier null zu erreichen, muß der Vorfaktor verschwinden,

$$z^2 - z - 1 = 0, \quad \text{also} \quad z = \frac{1}{2}(1 \pm \sqrt{5}) = \begin{cases} \ \ 1.618034 =: z_1, \\ -0.618034 =: z_2. \end{cases}$$

Es gibt also zwei mögliche Lösungen (auch c ist noch beliebig), welche ist die gesuchte?

Da die Rekursion *linear* ist, können verschiedene Lösungen linear überlagert werden, daher erfüllt auch die Folge $F_n := c_1 z_1^n + c_2 z_2^n$ für beliebige c_1, c_2 die Rekursion:

$$\begin{aligned} F_n - F_{n-1} - F_{n-2} &= c_1 z_1^n + c_2 z_2^n - c_1 z_1^{n-1} - c_2 z_2^{n-1} - c_1 z_1^{n-2} - c_2 z_2^{n-2} \\ &= c_1 \underbrace{(z_1^2 - z_1 - 1)}_{=0} z_1^{n-2} + c_2 \underbrace{(z_2^2 - z_2 - 1)}_{=0} z_2^{n-2} \equiv 0 \quad \forall n \geq 2\,(!). \end{aligned}$$

Um auch die Startwerte F_0, F_1 zu beschreiben, sind die Koeffizienten c_1, c_2 aus diesen zu bestimmen:

$$0 = F_0 = c_1 z_1^0 + c_2 z_2^0 = c_1 + c_2, \quad 1 = F_1 = c_1 z_1 + c_2 z_2$$

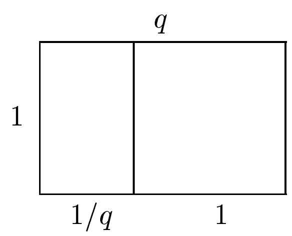
$\Rightarrow c_2 = -c_1$, $c_1 = 1/\sqrt{5}$. Für die *Fibonacci-Folge* ergibt sich also die explizite Darstellung

$$F_n = \frac{1}{\sqrt{5}}(z_1^n - z_2^n) = \frac{1}{\sqrt{5}}\Big[\Big(\frac{1+\sqrt{5}}{2}\Big)^n - \Big(\frac{1-\sqrt{5}}{2}\Big)^n\Big] \ \forall n \in \mathbb{N}_0 .$$

Für die obige Folge W_n ergeben sich mit dem gleichen Ansatz $W_n = c_1 z_1^n + c_2 z_2^n$ (gleiche Rekursion!) wegen der anderen Startwerte dagegen $c_{1/2} = \frac{1}{2} \pm \frac{3}{10}\sqrt{5}$. An der Darstellung kann man direkt das Verhalten der Folge (F_n) für große n ablesen. Da $|z_1| > |z_2|$ gilt, wächst der erste Anteil $c_1 z_1^n$ in

$$F_n = c_1 z_1^n + c_2 z_2^n = z_1^n(c_1 + c_2 x^n), \ |x| = \frac{|z_2|}{z_1} = \frac{2}{3+\sqrt{5}} < 0.4,$$

wesentlich schneller als der zweite, für große n gilt also $F_n \cong \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \cong z_1 F_{n-1}$. Das Seitenverhältnis bei der Quadratspirale nähert sich daher dem *Goldenen Schnitt* $z_1 = q$ mit $q = 1 + \frac{1}{q}$, bei dem beide Rechtecke gleiches Seitenverhältnis haben.



Das hier im Spezialfall angewendete Lösungsverfahren beruht auf zwei Eigenschaften, die noch einmal allgemein formuliert werden. Die Aussagen sind völlig analog zu denen bei linearen Differentialgleichungen k-ter Ordnung.

Satz 2.4 *Zu $k \in \mathbb{N}$ und $n \geq k$ seien Koeffizienten $a_{n1}, \ldots, a_{nk} \in \mathbb{C}$ gegeben, $a_{nk} \neq 0 \ \forall n \geq k$. Dann ist die Menge der Lösungen der k-stufigen homogenen* linearen Differenzengleichung

$$x_n + a_{n1}x_{n-1} + \ldots + a_{nk}x_{n-k} = 0 \quad \textit{für } n = k, k+1, k+2, \ldots \tag{3}$$

ein k-dimensionaler linearer Unterraum der Menge aller komplexen Folgen Abb$(\mathbb{N}_0, \mathbb{C})$.

Beweis Es sei L die Lösungsmenge der Differenzengleichung.

a) Linearität: Für zwei Folgen $(x_n), (y_n) \in L$ und beliebige $\alpha, \beta \in \mathbb{C}$ gilt für die Folge $(\alpha x_n + \beta y_n)$:

$$\begin{aligned} &(\alpha x_n + \beta y_n) + a_{n1}(\alpha x_{n-1} + \beta y_{n-1}) + \ldots + a_{nk}(\alpha x_{n-k} + \beta y_{n-k}) \\ = \ & \alpha \underbrace{(x_n + a_{n1}x_{n-1} + \ldots + a_{nk}x_{n-k})}_{=0} + \beta \underbrace{(y_n + a_{n1}y_{n-1} + \ldots + a_{nk}y_{n-k})}_{=0} = 0, \end{aligned}$$

b) Jede Folge, die Lösung der Differenzengleichung ist, ist durch ihre k Anfangswerte $x_0, \ldots, x_{k-1}$ eindeutig bestimmt. Eine Basis $(y_n^{(0)}), \ldots, (y_n^{(k-1)})$ von L kann durch die Startwerte

$$y_n^{(j)} = \delta_{nj}, \quad 0 \leq n, j \leq k-1$$

definiert werden. Da schon die Anfangsabschnitte $(y_0^{(j)}, \ldots, y_{k-1}^{(j)})$ nach Konstruktion linear unabhängig sind, sind dies auch die ganzen Folgen $(y_n^{(j)})$. Daher ist $\dim L \geq k$. Sei nun $(x_n) \in L$ beliebig. Für den Abschnitt $(x_0, \ldots, x_{k-1})$ gibt es Konstanten $\beta_0, \ldots, \beta_{k-1}$ (nämlich $\beta_j = x_j$) mit

$$x_n = \beta_0 y_n^{(0)} + \ldots + \beta_{k-1} y_n^{(k-1)}, \quad n = 0, \ldots, k-1.$$

Nach Teil a) überträgt sich diese Linearkombination über die Differenzengleichung induktiv auf die restlichen Elemente $n \geq k$:

$$x_n = -\sum_{i=1}^{k} a_{ni}x_{n-i} = -\sum_{i=1}^{k} a_{ni} \sum_{j=0}^{k-1} \beta_j y_{n-i}^{(j)} = \sum_{j=0}^{k-1} \beta_j \Big(-\sum_{i=1}^{k} a_{ni} y_{n-i}^{(j)} \Big) = \sum_{j=0}^{k-1} \beta_j y_n^{(j)}.$$

Daher ist $\dim L = k$. ■

Bei Kenntnis einer Basis von L kann also jede Lösung explizit konstruiert werden.

Beispiel: Bei den Derangementzahlen sind die Koeffizienten in (3) $a_{n1} = a_{n2} = -(n-1)$. Eine Lösung der Dfz-Gl. wurde in Satz 1.16 angegeben, eine andere (lin. unabh.) ist $(x_n) = (n!)$, denn

$$x_n = (n-1)\Big((n-1)! + (n-2)!\Big) = (n-1)(n-1+1)(n-2)! = n!\ .$$

Bei den Fibonacci-Zahlen waren die Koeffizienten $a_{n1} = a_{n2} = -1$ unabhängig von n. Für diesen Fall gibt es ein allgemeines Lösungs-*Verfahren.*

Satz 2.5 *Die Koeffizienten der Differenzengleichung (3) seien* konstant, *d.h., unabhängig von n ($a_{nj} \equiv a_j \forall n$). Das* charakteristische Polynom

$$p(z) := z^k + a_1 z^{k-1} + \ldots + a_{k-1}z + a_k, \quad z \in \mathbb{C},$$

dieser Gleichung besitze k verschiedene Nullstellen $z_1, \ldots, z_k \in \mathbb{C}$. Dann ist eine Basis des Lösungsraums L der Differenzengleichung gegeben durch die Folgen $(z_1^n)_n, \ldots, (z_k^n)_n$, d.h., jede ihrer Lösungen $(x_n)_n$ läßt sich mit $\beta_j \in \mathbb{C}$ darstellen in der Form

$$x_n = \beta_1 z_1^n + \ldots + \beta_k z_k^n, \quad n \in \mathbb{N}_0\,.$$

Beweis Jede der k Folgen $y_n^{(j)} := z_j^n$, $n \in \mathbb{N}_0$, erfüllt (mit $a_0 := 1$) für $n \ge k$

$$\sum_{i=0}^k a_i y_{n-i}^{(j)} = \sum_{i=0}^k a_i z_j^{n-i} = z_j^{n-k}\Big(\sum_{i=0}^k a_i z_j^{k-i}\Big) = z_j^{n-k}p(z_j) = 0.$$

Also ist $(y_n^{(j)}) \in L$. Dabei sind die Folgen $(y_n^{(j)})$, $j = 1, \ldots, k$, linear unabhängig, da dies schon für ihre Anfangsabschnitte gilt:

$$\det\begin{pmatrix} y_0^{(1)} & \ldots & y_{k-1}^{(1)} \\ \vdots & & \vdots \\ y_0^{(k)} & \ldots & y_{k-1}^{(k)} \end{pmatrix} = \det\begin{pmatrix} 1 & z_1 & z_1^2 & \ldots & z_1^{k-1} \\ \vdots & \vdots & & & \vdots \\ 1 & z_k & z_k^2 & \ldots & z_k^{k-1} \end{pmatrix} = \prod_{i>j}(z_i - z_j) \ne 0.$$

Diese *Vandermonde-Determinante* verschwindet nicht, da die z_j n.V. verschieden sind. ■

Beispiel: 1) Fibonacci-Folge $x_n = x_{n-1} + x_{n-2}$ (vgl. oben).

$$p(z) = z^2 - z - 1,\ z_{1/2} = \frac{1}{2} \pm \frac{1}{2}\sqrt{5},\ \Rightarrow\ x_n = \beta_1(\frac{1}{2} + \frac{1}{2}\sqrt{5})^n + \beta_2(\frac{1}{2} - \frac{1}{2}\sqrt{5})^n.$$

2) Rekursion $x_n = x_{n-1} + 2x_{n-2}$, als Differenzengleichung $x_n - x_{n-1} - 2x_{n-2} = 0$.

$$p(z) = z^2 - z - 2,\ z_{1/2} = \frac{1}{2} \pm \frac{3}{2},\ \Rightarrow\ x_n = \beta_1 2^n + \beta_2(-1)^n.$$

Die Koeffizienten sind aus den Anfangswerten x_0, x_1 zu bestimmen:

$$\left.\begin{array}{lll} \beta_1 2^0 & +\beta_2(-1)^0 & = x_0 \\ \beta_1 2 & -\beta_2 & = x_1 \end{array}\right\} \Rightarrow 3\beta_1 = x_0 + x_1\ \Rightarrow \beta_2 = \frac{2}{3}x_0 - \frac{1}{3}x_1.$$

3) Bei der reellen Differenzengleichung $x_n - 2x_{n-1} + 2x_{n-2} = 0$ treten komplexe Nullstellen auf:

$$p(z) = z^2 - 2z + 2 = (z-1)^2 + 1,\ z_{1/2} = 1 \pm \mathbf{i} \in \mathbb{C} \ \Rightarrow \ x_n = \beta_1(1+\mathbf{i})^n + \beta_2(1-\mathbf{i})^n.$$

Diese Folge ist wegen $z_2 = \bar{z}_1$ reell, wenn $\beta_2 = \bar{\beta}_1$ gilt. Mit den Anfangswerten $x_0 = 0, x_1 = 1$ folgt, z.B.,

$$\begin{array}{rlll} 0 = & \beta_1 & +\beta_2 & \\ 1 = & \beta_1(1+\mathbf{i}) & +\beta_2(1-\mathbf{i}) & = \underbrace{(\beta_1+\beta_2)}_{=0} + \mathbf{i}(\beta_1-\beta_2) \ \cdot \end{array}$$

Also ist $\beta_1 - \beta_2 = 2\beta_1 = -2\beta_2 = -\mathbf{i}$ und daher mit $1 + \mathbf{i} = \sqrt{2}\exp(\mathbf{i}\pi/4)$

$$x_n = -\frac{\mathbf{i}}{2}(1+\mathbf{i})^n + \frac{\mathbf{i}}{2}(1-\mathbf{i})^n = \mathrm{Re}(\,-\,\mathbf{i}(1+\mathbf{i})^n) = 2^{n/2}\sin\frac{n\pi}{4}.$$

Bemerkung: Im Satz wird der Fall mehrfacher Nullstellen von p nicht behandelt.

Inhomogene Gleichungen: Wird bei der Zinsgleichung eine konstante Ratenzahlung b eingeführt, ändert sich die Rekursion zu $K_n = aK_{n-1} + b$. Die Bestimmung der Lösung ist $(a \neq 1)$ analog zu früher.

$$\begin{aligned} K_n &= aK_{n-1} + b = a^2K_{n-2} + ab + b = a^3K_{n-3} + a^2b + ab + b = \ldots \\ &= a^nK_0 + b(a^{n-1} + a^{n-2} + \ldots + 1) = a^nK_0 + b\,\frac{a^n - 1}{a - 1}. \end{aligned}$$

Der erste Bestandteil, a^nK_0, ist die Lösung der homogenen Gleichung, dazu wird der von b herstammende Anteil addiert. Dies ist ein allgemeines Prinzip:

Satz 2.6 *Die allgemeine Lösung der* inhomogenen linearen Differenzengleichung

$$x_n + a_{n1}x_{n-1} + \ldots + a_{nk}x_{n-k} = b_n \quad \textit{für } n = k, k+1, k+2, \ldots$$

hat die Form $x_n = x_n^H + \widehat{x_n}$, $n \in \mathbb{N}_0$, *wobei* $(\widehat{x_n})$ *eine spezielle Lösung der inhomogenen Gleichung ist, und* (x_n^H) *die allgemeine Lösung der homogenen Gleichung aus Satz 2.4 ist. Unter den spezielleren Voraussetzungen von Satz 2.5 ist also*

$$x_n = \widehat{x_n} + \beta_1 z_1^n + \ldots + \beta_k z_k^n, \quad n \in \mathbb{N}_0.$$

Beweis Es seien $(\widehat{x_n})$ und (x_n) zwei Lösungen der inhomogenen Gleichung. Dann ist die Differenzfolge $(x_n - \widehat{x_n})$ eine Lösung der homogenen Gleichung:

$$\begin{aligned} & x_n - \widehat{x_n} + a_{n1}(x_{n-1} - \widehat{x_{n-1}}) + \ldots + a_{nk}(x_{n-k} - \widehat{x_{n-k}}) \\ &= (x_n + a_{n1}x_{n-1} + \ldots + a_{nk}x_{n-k}) - (\widehat{x_n} + a_{n1}\widehat{x_{n-1}} + \ldots + a_{nk}\widehat{x_{n-k}}) \\ &= b_n - b_n = 0 \quad \forall n \geq k. \quad \blacksquare \end{aligned}$$

Bei Differenzengleichungen mit konstanten Koeffizienten können *spezielle Inhomogenitäten* durch geeigneten Ansatz behandelt werden:

a) Konstante, $b_n \equiv c \; \forall n$: Für $p(1) \neq 0$ führt der Ansatz $\widehat{x_n} \equiv d$ zum Erfolg:

$$\widehat{x_n} + \ldots + a_k \widehat{x_{n-k}} = (1 + a_1 + \ldots + a_k)d \stackrel{!}{=} c$$

ist lösbar, wenn $1 + a_1 + \ldots + a_k = p(1) \neq 0$ gilt. Die spezielle Lösung ist dann $\widehat{x_n} \equiv c/p(1)$.

b) Allgemeine Polynome, $b_n = \sum_{j=0}^{m} \gamma_j n^j$. Ansatz $\widehat{x_n} = \sum_{j=0}^{m'} \delta_j n^j$ mit $m' \geq m$.
Für $p(1) \neq 0$ und $m' = m$ ergibt sich ein lineares Gleichungssystem für die Koeffizienten $\delta_0, \ldots, \delta_m$. Im Fall $p(1) = 0$ ist ein höherer Polynomgrad $m' > m$ im Ansatz erforderlich.

Das Verfahren wird am Beispiel $x_n = 3x_{n-1} - 2x_{n-2} + 1$ erläutert. Es gilt $p(z) = z^2 - 3z + 2 = (z-1)(z-2)$ und $b_n \equiv 1$. Wegen $p(1) = 0$ wird der Ansatz $\widehat{x_n} = \delta n$ gewählt, es folgt

$$\widehat{x_n} - 3\widehat{x_{n-1}} + 2\widehat{x_{n-2}} = \delta(n - 3(n-1) + 2(n-2)) = \delta(n - 3n + 3 + 2n - 4) = -\delta \stackrel{!}{=} 1.$$

Also ist $\widehat{x_n} = -n$ und die allgemeine Lösung der Rekursion lautet $x_n = -n + \beta_1 + \beta_2 2^n$.

c) Exponentialfunktionen $b_n = c\, q^n$: Ansatz $\widehat{x_n} = \delta\, q^n$ erfolgreich für $p(q) \neq 0$ (für $p(q) = 0$ ist (q^n) dagegen schon Lösung der homogenen Gleichung).

2.3 Erzeugende Funktionen

Einige der behandelten Folgen (Binomialkoeffizienten, Stirlingzahlen) sind auch die Koeffizienten spezieller Polynome. Aus diesem Zusammenhang ergaben sich bestimmte Rechenregeln mit diesen Zahlen. Analog läßt sich zu jeder Folge (a_n) die zugehörige Potenzreihe zumindestens formal (d.h., ohne Konvergenzdiskussion) betrachten, um zusätzliche Hilfsmittel zugänglich zu machen.

Defin. 2.7 *Zur Folge* $(a_n)_{n\geq 0}$ *wird die* erzeugende Funktion *definiert durch die formale Reihe*

$$A(z) := \sum_{n\geq 0} a_n z^n, \quad z \in \mathbb{C}.$$

'Formal': Rechenregeln unabhängig von Konvergenzfragen. Aus der Analysis ist, z.B., das Cauchy-Produkt bekannt:

$$\begin{aligned} A(z)B(z) &= \sum_{n\geq 0} a_n z^n \sum_{n\geq 0} b_n z^n = \sum_{n\geq 0} \Big(\sum_{j=0}^{n} a_j b_{n-j}\Big) z^n = C(z), \text{ mit} \\ c_n &= \sum_{j=0}^{n} a_j b_{n-j}, \; n \geq 0. \end{aligned} \tag{4}$$

Indexverschiebung ($\rightarrow$ Differenzen-Rechnung, -Gleichungen):

$$\begin{aligned} (b_n) = (0, a_0, a_1, \ldots) \quad &\Rightarrow \quad B(z) = \sum_{n\geq 1} a_{n-1} z^n = zA(z), \\ (a_n) = (b_1, b_2, \ldots) \quad &\Rightarrow \quad A(z) = \sum_{n\geq 0} b_{n+1} z^n = (B(z) - b_0)/z. \end{aligned}$$

Eine wichtige Anwendung betrifft spezielle gewichtete Summen von Folgenelementen, die etwa bei mittleren Anzahlen, Erwartungswerten (vgl. Algorithmus **P**) auftreten. Dazu gehört

$$\sum_{n\geq 1} na_n = A'(1), \quad \text{da} \quad A'(z) = \sum_{n\geq 1} na_n z^{n-1}.$$

Summen mit höheren n-Potenzen lassen sich aus denen mit Faktoriellen zusammensetzen, es gilt

$$\sum_{n\geq j} n^{\underline{j}} a_n = A^{(j)}(1),\ j \in \mathbb{N}.$$

Beispiele:

1. $a_n \equiv 1$: $A(z) = \sum_{n\geq 0} z^n = \frac{1}{1-z}$ (formale geometrische Reihe). Die Funktion
$$\frac{1}{1-z}B(z) = \frac{1}{1-z}\sum_{n\geq 0} b_n z^n = b_0 + (b_0+b_1)z + (b_0+b_1+b_2)z^2 + \ldots,$$
nach (4), gehört zur *diskreten Stammfunktion* der Folge (b_n).

2. $m \in \mathbb{N}$ fest, $a_n = \binom{m}{n}$: $A(z) = \sum_{n\geq 0} \binom{m}{n} z^n = (1+z)^m$, Binomial-Summe,
$x \in \mathbb{R}$ fest, $a_n = \binom{x}{n}$: $A(z) = \sum_{n\geq 0} \binom{x}{n} z^n = (1+z)^x$, Binomial-**Reihe**.

3. Stirlingzahlen ($m \in \mathbb{N}$ fest), nach Definition in §1.3 gilt: Zu $a_n = (-1)^{m-n} s_{mn}$ gehört das Polynom $A(z) = z^{\underline{m}}$ und zu $b_n = s_{mn}$ das Polynom $B(z) = z^{\overline{m}}$.

4. Harmonische Zahlen: (H_n) ist die diskrete Stammfunktion zur Folge $(b_n) = (\frac{1}{n})_{n\geq 1}$ und es gilt $B(z) = \sum_{n\geq 1} \frac{1}{n} z^n = \int_0^z \frac{dx}{1-x} = -\ln(1-z)$. Mit 1) folgt
$$H(z) = \sum_{n\geq 1} H_n z^n = \frac{1}{1-z} \ln \frac{1}{1-z}.$$

Die erzeugenden Funktionen von linearen Rekursionen sind rationale Funktionen. Bei der Fibonacci-Folge sei $F(z) := \sum_{n\geq 0} F_n z^n$ mit $F_0 = 0, F_1 = 1$. Aus der Rekursion $F_n = F_{n-1} + F_{n-2}$ folgt

$$\begin{aligned} F(z) &= z + \sum_{n\geq 2} F_n z^n = z + z\sum_{n\geq 2} F_{n-1} z^{n-1} + z^2 \sum_{n\geq 2} F_{n-2} z^{n-2} \\ &= z + (z+z^2)F(z) \Rightarrow \quad F(z) = \frac{z}{1-z-z^2}. \end{aligned} \tag{5}$$

Das Beispiel wird später weitergeführt, die allgemeine Problemklasse behandelt

Satz 2.8 *Die erzeugende Funktion $X(z) = \sum_{n\geq 0} x_n z^n$ für jede Lösung der linearen Differenzengleichung mit konstanten Koeffizienten*

$$x_n + a_1 x_{n-1} + \ldots + a_k x_{n-k} = b_n, \quad n \geq k,$$

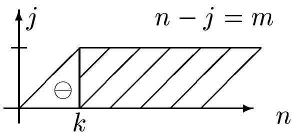
($a_0 := 1$) hat die Form

$$X(z) = \frac{B(z) + q(z)}{A(z)}.$$

Dabei hängt $A(z) = \sum_{n=0}^k a_n z^n$ mit dem charakteristischen Polynom p in Satz 2.5 über $A(z) = z^k p(1/z)$ zusammen. Außerdem ist $q(z) = \sum_{n=0}^{k-1} \sum_{j=0}^n a_j x_{n-j} z^n$ und $B(z) = \sum_{n\geq k} b_n z^n$.

Beweis Mit $a_0 := 1$ folgt aus der Differenzengleichung

$$\begin{aligned}
B(z) &= \sum_{n\geq k} b_n z^n = \sum_{n\geq k}\Big(\sum_{j=0}^{k} a_j x_{n-j}\Big) z^n = \sum_{n\geq k}\sum_{j=0}^{k} x_{n-j} z^{n-j} a_j z^j \\
&= \sum_{j=0}^{k}\Big(\sum_{n\geq j} x_{n-j} z^{n-j} a_j z^j - \sum_{n=j}^{k-1} x_{n-j} a_j z^n\Big) \\
&= \sum_{m\geq 0} x_m z^m \sum_{j=0}^{k} a_j z^j - q(z) \\
&= X(z)A(z) - q(z). \quad \blacksquare
\end{aligned}$$



Den Zusammenhang mit der Darstellung aus Satz 2.5, also die Koeffizienten von $X(z)$, bekommt man durch Partialbruchzerlegung der rationalen Funktion und den Einsatz der geometrischen Reihe. Bei der Fibonacci-Folge ergibt sich aus (5) mit $z_{1/2} = (1 \pm \sqrt{5})/2$ und $1 - z - z^2 = (1 - z_1 z)(1 - z_2 z)$ der Ansatz

$$F(z) = \frac{z}{1 - z - z^2} = \frac{\alpha_1 + \beta_1 z}{1 - z_1 z} + \frac{\alpha_2 + \beta_2 z}{1 - z_2 z}.$$

Durch Übergang zum Hauptnenner und Vergleich der Zähler folgt

$$z = \alpha_1 + \alpha_2 + (\beta_1 + \beta_2 - \alpha_1 z_2 - \alpha_2 z_1) z - (\beta_1 z_2 + \beta_2 z_1) z^2,$$

woraus sich durch Koeffizientenvergleich etwa $\alpha_1 = \alpha_2 = 0$ und insgesamt die folgende Darstellung ergibt, die der vom Anfang des Paragraphen entspricht:

$$F(z) = \frac{z_1 z}{\sqrt{5}(1 - z_1 z)} - \frac{z_2 z}{\sqrt{5}(1 - z_2 z)} = \frac{1}{\sqrt{5}} \sum_{n\geq 1} \Big(z_1^n - z_2^n\Big) z^n.$$

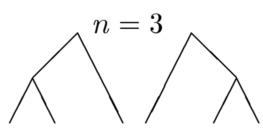
Anwendung Catalansche Zahlen C_n:

Bedeutung: C_n sei die Anzahl der zulässigen Klammerungen bei Auswertung des (nicht kommutativen) Produkts $a_1 a_2 \cdots a_n$ von n Größen. Diese Anzahl ist identisch mit der Zahl binärer (Auswertungs-) Bäume mit n Blättern bzw. $n-1$ Verzweigungsknoten. Es gilt ($C_0 := 0$)

$C_1 = 1$: a_1

$C_2 = 1$: $(a_1 a_2)$

$C_3 = 2$: $(a_1 a_2) a_3 = a_1 (a_2 a_3)$



Allgemein sind für jedes j, $1 \leq j \leq n-1$, innerhalb der äußeren Klammerung bei

$$a_1 \cdots a_n = \underbrace{(a_1 \cdots a_j)}_{C_j} \cdot \underbrace{(a_{j+1} \cdots a_n)}_{C_{n-j}}$$

zusätzlich $C_j \cdot C_{n-j}$ weitere Klammerungen möglich. Daher gilt für $n \geq 2$ (mit $C_0 = 0$):

$$C_n = C_1 C_{n-1} + C_2 C_{n-2} + \ldots + C_{n-1} C_1 = \sum_{j=1}^{n-1} C_j C_{n-j} = \sum_{j=0}^{n} C_j C_{n-j}.$$

Dies ist eine *nichtlineare* Rekursion. Allerdings entspricht die darin auftretende Summe einer Faltung wie beim Cauchy-Produkt von Potenzreihen. Daher gilt für die Erzeugende Funktion

$$C(z) = z + \sum_{n\geq 2} C_n z^n = z + \sum_{n\geq 2}\sum_{j=0}^{n} C_j C_{n-j} z^n \overset{(4)}{=} z + C(z)^2.$$

Also gilt $C^2(z) - C(z) + z = 0 \Rightarrow C(z) = \frac{1}{2} \pm \frac{1}{2}\sqrt{1-4z}$. Von diesen beiden Lösungen kommt aber wegen der Startbedingung $C_0 = C(0) = 0$ nur die mit dem negativen Vorzeichen in Frage. Schließlich liefert die allgemeine Binomialreihe die Darstellung

$$C(z) \quad = \quad \frac{1}{2} - \frac{1}{2}(1-4z)^{1/2} = \frac{1}{2} - \frac{1}{2}\sum_{n\geq 0}\binom{1/2}{n}(-4z)^n = \sum_{n\geq 1}\frac{1}{n}\binom{2n-2}{n-1}z^n.$$

Die letzte Umformung folgt bei $(\frac{1}{2})^{\underline{n}} = \frac{1}{2}(-\frac{1}{2})(-\frac{3}{2})\cdots(\frac{3}{2}-n) = (-1)^{n-1}2^{-n}\cdot 1\cdot 3\cdot 5\cdots(2n-3)$ durch Ergänzung der geraden Faktoren von $(2n-2)!$.

Ergebnis: Für $n \geq 1$ existieren $C_n = \frac{1}{n}\binom{2n-2}{n-1}$ Möglichkeiten zur Klammerung von n-Produkten.

Anwendung Formale Sprachen: Eine Sprache $\mathcal{L}$ über einem Alphabet A ist eine Teilmenge der Menge $A^* = \bigcup_{n\geq 0} A^n$ aller Worte endlicher Länge. Das leere Wort heißt ε. Bezeichnet ℓ_n die Anzahl der Worte in $\mathcal{L}$ mit Länge n, so definiert man die Erzeugende Funktion von $\mathcal{L}$ durch

$$L(z) := \sum_{n\geq 0} \ell_n z^n, \ z \in \mathbb{C}.$$

Die Eigenschaften einer Sprache spiegeln sich in ihrer Erzeugenden Funktion wieder:

- Reguläre Sprachen $\mathcal{L}$ besitzen rationale Erzeugende Funktionen $L(z)$.
- Kontextfreie Sprachen $\mathcal{L}$ besitzen algebraische Erzeugende Funktionen $L(z)$.

Insbesondere gelten einfache Rechenregeln bei der Vereinigung und Verkettung (vgl. Übungen):

$$\begin{array}{lll} \mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2, & \text{mit } \mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset & \Rightarrow \quad L(z) = L_1(z) + L_2(z), \\ \mathcal{L} = \mathcal{L}_1\mathcal{L}_2, & \text{(eindeutige Zerlegbarkeit)} & \Rightarrow \quad L(z) = L_1(z)L_2(z), \\ \mathcal{L} = \mathcal{L}_1^*, & \text{(eindeutige Zerlegbarkeit)} & \Rightarrow \quad L(z) = \dfrac{1}{1-L_1(z)}. \end{array}$$

Die Verkettung $\mathcal{L}_1\mathcal{L}_2 = \{u_1u_2 : u_j \in \mathcal{L}_j\}$ entspricht dem 'Hintereinanderschreiben' aller Worte aus $\mathcal{L}_1$ und $\mathcal{L}_2$, die Einschränkung *eindeutige Zerlegbarkeit* heißt umgekehrt, dass bei einem Wort $w = u_1u_2 \in \mathcal{L}_1\mathcal{L}_2$ die Zerlegung in $u_1 \in \mathcal{L}_1, u_2 \in \mathcal{L}_2$ eindeutig ist. Analoges fordert man bei $\mathcal{L}_1^* = \{\varepsilon\} \cup \mathcal{L}_1 \cup \mathcal{L}_1\mathcal{L}_1 \cup \mathcal{L}_1\mathcal{L}_1\mathcal{L}_1 \cup \ldots$ sowie $\mathcal{L}_1^j \cap \mathcal{L}_1^k = \emptyset \forall j \neq k$, also insbesondere $\varepsilon \notin \mathcal{L}_1$.

Beispiel: Die Menge $\mathcal{L}$ aller Palindrom-Worte (wie *Reliefpfeiler*) über A wird aus dem Startsymbol S generiert durch die kontextfreie Grammatik

$$S \to aSa, \ S \to b, \ S \to \varepsilon \ \forall a, b \in A.$$

Für $|A| = m$ ist $\ell_0 = 1, \ell_1 = m$ und aus der ersten Regel folgt $\ell_{n+2} = m\ell_n$, $n \geq 0$. Also gilt $L(z) = 1 + mz + mz^2L(z) \iff$

$$L(z) = \frac{1+mz}{1-mz^2}, \quad \ell_n = m^{\lceil n/2 \rceil}.$$

L ist hier rational, da auch reguläre Sprachen mit dieser Erzeugenden Funktion existieren.

2.4 Aufwandsanalyse bei rekursiven Algorithmen

Beim Einsatz von Algorithmen ist bei *großen Problemen* das asymptotische Wachstum des Aufwands in Abhängigkeit von einem geeigneten Größenparameter n entscheidend. Zum Vergleich des *Wachstums* von Funktionen für große $n \gg 1$ gibt es unterschiedlich präzise Begriffe:

Defin. 2.9 *Gegeben seien Funktionen $f, g : \mathbb{N} \to \mathbb{R}$ ($g(n) = 0$ nur endlich oft). Man sagt, es gilt (für $n \to \infty$)*

$$\begin{array}{llllll}
a) & f(n) = o(g(n)) & :\iff & \forall \varepsilon > 0 \exists n_0 > 0: & |f(n)| \le \varepsilon |g(n)| & \forall n \ge n_0. \\
b) & f(n) = O(g(n)) & :\iff & \exists C, n_0 > 0: & |f(n)| \le C|g(n)| & \forall n \ge n_0. \\
c) & f(n) = \Theta(g(n)) & :\iff & \exists C_1, C_2, n_0 > 0: & C_1|g(n)| \le |f(n)| \le C_2|g(n)| & \forall n \ge n_0. \\
d) & f(n) \sim g(n) & :\iff & & f(n)/g(n) \to 1, & n \to \infty.
\end{array}$$

Sprechweisen:	a)	'f wächst langsamer als g',	'f ist klein-o von g'
	b)	'f wächst nicht schneller als g',	'f ist groß-O von g'
	c)	'f und g wachsen gleich schnell',	'f ist Theta von g'
	d)	'f und g sind asymptotisch gleich'.	

Beispiel: p und q seien Polynome in n. Dann gilt für $n \to \infty$:
a) $p(n) = o(q(n)) \iff \operatorname{Grad} p < \operatorname{Grad} q$;
b) $p(n) = O(q(n)) \iff \operatorname{Grad} p \le \operatorname{Grad} q$
c) $p(n) = \Theta(q(n)) \iff \operatorname{Grad} p = \operatorname{Grad} q$.
d) $p(n) \sim q(n) \iff \operatorname{Grad} p = \operatorname{Grad} q$ und die höchsten Koeffizienten sind gleich.

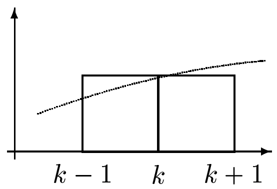
Außerdem gilt (vgl. Analysis I): $p(n) = o(e^{an})$ für $a > 0$; $\log n = o(p(n))$ für $\operatorname{Grad} p > 0$.

Für den Wachstumsvergleich mit Fakultäten ist der folgende Satz entscheidend.

Satz 2.10 *(Stirling-Formeln)*

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}, \qquad \ln(n!) \sim n \ln n.$$

Beweis (für die 2. Formel) Elementar gilt $\ln n! = \ln(2 \cdot 3 \cdots n) = \sum_{k=2}^n \ln k$.



Da der Logarithmus eine monoton wachsende Funktion ist, gilt

$$\int_{k-1}^{k} \ln x \, dx \le \ln k \le \int_{k}^{k+1} \ln x \, dx = x(\ln x - 1)\Big|_k^{k+1}.$$

Durch Summation folgt:

$$n \ln n - n + 1 = x(\ln x - 1)\Big|_1^n \le \ln n! \le x(\ln x - 1)\Big|_2^{n+1} = (n+1)\ln(n+1) - n - 2\ln 2 + 1.$$

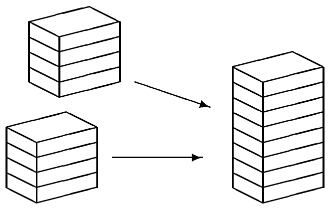
Im Grenzübergang $n \to \infty$ gilt daher

$$1 \leftarrow \frac{n \ln n - n + 1}{n \ln n} \le \frac{\ln n!}{n \ln n} \le \frac{(n+1)\ln(n+1) - n - 2\ln 2 + 1}{n \ln n} \to 1. \quad \blacksquare$$

Wachstum bei Rekursionen

Bei stark strukturierten Algorithmen kann der Aufwand oft durch eine Rekursion über die Problemgröße beschrieben werden. Ähnliches gilt für Größenangaben rekursiv definierter Objekte (Sprachen). Dazu ist die Wachstumsabschätzung solcher Rekursionen hilfreich.

1. In der Situation von Satz 2.5 gilt für jede Lösung (x_n) der homogenen *linearen Rekursion* $x_n = O(\phi^n)$, wobei $\phi = \max\{|z_j| : 1 \le j \le k\}$ ist, denn
$$|x_n| = |\sum_{j=1}^{k} \beta_j z_j^n| = \phi^n \underbrace{\Big|\sum_{j=1}^{k} \beta_j (\frac{z_j}{\phi})^n\Big|}_{\text{glm. beschr.}}.$$
Für die Fibonacci-Zahlen können diese Aussagen noch präzisiert werden:
$$F_n = \Theta(\phi^n), \quad F_n \sim \frac{1}{\sqrt{5}}\phi^n, \quad \phi =: (1+\sqrt{5})/2.$$

2. Teile-und-Herrsche-Algorithmen (divide and conquer): Gemeinsames Prinzip solcher Algorithmen ist die (rekursive) Aufteilung des Gesamtproblems in kleinere Teile und die anschließende Zusammenführung der Teillösungen.
Beispiel: Sortieren durch Mischen (merge-sort), zwei gleichartig sortierte Kartenstapel der Länge $n/2$ können mit n Vergleichen zu einem sortierten vereinigt werden. Das Prinzip ist auch schon auf die Teilstapel anwendbar. Aufwand:

$$T(n) = 2\,T(\frac{n}{2}) + n, \quad T(1) = 0.$$
Für $n = 2^m$ ist dies eine einstufige lineare Rekursion, wenn man $x_m := T(2^m)$ betrachtet. Dann gilt $x_m = 2x_{m-1} + 2^m = 2^k x_{m-k} + (m-k)2^m = m2^m = n\log_2 n$. Die Größenordnung ändert sich nicht, wenn n keine Zweierpotenz ist: $T(n) = \Theta(n\log_2 n)$. Für die Anzahl der Runden hat man trivialerweise $R(n) = R(n/2) + 1$. Mit der Bezeichnung $\lceil x\rceil := \min\{n \in \mathbb{Z} : n \ge x\}$ gilt allgemein:

Satz 2.11 *Für die positive, monoton wachsende Funktion T gelte mit $a \ge 1$, $b > 1$ und $g \ge 0$ die Beziehung*
$$T(n) = a\,T\Big(\lceil\frac{n}{b}\rceil\Big) + g(n), \quad n \in \mathbb{N}.$$
Mit $q := \log a/\log b = \log_b a$ gilt dann
$$\begin{aligned} T(n) &= \Theta(n^q) && \textit{wenn} \quad g(n) = O(n^p),\ p < q,\\ T(n) &= \Theta(n^q \log n) && \textit{wenn} \quad g(n) = \Theta(n^q),\\ T(n) &= O(n^p) && \textit{wenn} \quad g(n) = O(n^p),\ p > q. \end{aligned}$$

Beweis Zunächst werden wieder die Werte $x_m := T(b^m)$ in $n = b^m$ betrachtet. Für diese läßt sich die Rekursion induktiv lösen:

$$\begin{aligned} x_m &= ax_{m-1} + g(b^m) = a^2 x_{m-2} + g(b^m) + ag(b^{m-1}) = \dots \\ &= a^m x_0 + \sum_{j=0}^{m-1} a^j g(b^{m-j}). \end{aligned} \tag{6}$$

Nach Definition gilt $a = b^q$. Das Wachstum von x_m hängt nun davon ab, ob $a^m = b^{mq} = n^q$ oder $g(b^m)$ schneller wachsen.

a) Für $g(n) \le Cn^p$ und $p < q$ folgt $g(b^i) \le Cb^{ip}$ und in (6) daher

$$x_m \le a^m x_0 + C \sum_{j=0}^{m-1} a^j b^{p(m-j)} = a^m \Big(x_0 + C \sum_{j=0}^{m-1} (b^p/a)^{m-j} \Big).$$

Da n.V. $b^p < b^q = a$ gilt, ist die Summe Teil einer konvergenten geom. Reihe und daher beschränkt. Also ist $a^m x_0 \le x_m \le C' a^m$. Das Ergebnis folgt nun mit $a^m = b^{mq} = n^q$.

b) Für $C_1 n^q \le g(n) \le C_2 n^q$ ist

$$x_m - a^m x_0 \ge C_1 \sum_{j=0}^{m-1} a^j b^{q(m-j)} = C_1 a^m \sum_{j=0}^{m-1} 1 = C_1 m a^m = C_1 n^q \log_b n.$$

Die Abschätzung nach oben folgt analog mit C_2 statt C_1.

c) g wächst schneller als $n^q = a^m$, da $p > q$:

$$x_m \le a^m x_0 + C \Big(\sum_{j=0}^{m-1} a^j b^{-pj} \Big) b^{mp} \le n^p \Big(x_0 + \frac{c}{1 - a/b^p} \Big).$$

Da hier $a = b^q < b^p$ gilt, ist die Summe wieder glm. beschränkt.
Wegen der vorausgesetzten Monotonie von T kann der bisher ausgesparte Fall $n \notin b^{\mathbb{N}}$ durch 'Aufrunden' abgedeckt werden: $T(n) \le T(b^m)$ mit $m = \lceil \log_b n \rceil$. ■

Der dritte Fall im Satz ist bei der Analyse von rekursiven **Parallelen Algorithmen** wichtig, hier ist $a \cong 1$ (Teilprobleme parallel gelöst), mit $b \ge 2$ gilt $0 \le q \ll 1$. Bei einer genügenden Anzahl von Prozessoren tritt in der Regel daher der letzte Fall ein, $T(n) = O(g(n))$, d.h., die Vereinigung der Teillösungen bestimmt den Aufwand.

Die Komplexität eines Verfahrens begrenzt die Größe der Probleme, die damit praktisch lösbar sind. Dies gilt auch bei den heute und zukünftig verfügbaren Rechnerleistungen. Der zur Zeit (März 2002) schnellste Rechner auf der Welt hat eine Nominalleistung von 40 TeraFLOPS, also von $40 \cdot 10^{12}$ Gleitpunkt-Operationen pro Sekunde (vgl. die Top-500 -Liste installierter Rechner bei *www.top500.org*). Die folgende Tabelle zeigt für verschiedene Funktionen T und

Größenordnungen n die Zahlwerte $T(n)$, sowie die Problemgröße $n_{\max}$, die auf einem 28-Teraflop-Rechner in 1 Stunde lösbar ist (d.h. mit 10^{17} Operationen).

	$n =$	10	100	10^3	10^6	10^9	$n_{\max}$
	$n \lg n$	10	200	3000	$6 \cdot 10^6$	10^{10}	$8 \cdot 10^{15}$
Auf-	n^2	100	10^4	10^6	10^{12}	10^{18}	$3 \cdot 10^8$
wand	n^3	1000	10^6	10^9	10^{18}	10^{27}	$5 \cdot 10^5$
$T(n)$	2^n	1000	10^{30}	10^{301}	xxx	xxx	56
	$n!$	$4 \cdot 10^6$	10^{158}	10^{2568}	xxx	xxx	18

Mit Verfahren vom exponentiellem Wachstum des Aufwands sind nur sehr kleine Probleme lösbar. Auf diesem Argument beruhen, z.B., Sicherheitsüberlegungen in der im nächsten Kapitel behandelten Kryptographie.

Teil B

Algebra

3 Endliche algebraische Strukuren

3.1 Endliche Körper

Nichtnumerische Algorithmen mit algebraischem Hintergrund können in endlichen Zahlkörpern exakt durchgeführt werden (keine Rundungsfehler wie bei Gleitpunktrechnung). Solche Algorithmen sind die Basis fehlererkennender Codierungen und moderner kryptographischer Verfahren. Zu endlichen Körpern kommt man durch Betrachtung von Restklassen.

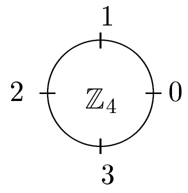
Zu $m \in \mathbb{N}$ stellt die Relation

$$x \equiv y \quad :\iff \quad x - y \in m\mathbb{Z}, \quad \text{d.h.,} \quad \exists k \in \mathbb{Z} : x - y = km$$

eine *Äquivalenzrelation* dar (symmetrisch, reflexiv, transitiv). Es gibt m Äquivalenzklassen $[0], [1], \ldots, [m-1]$. Zahlen $x, y \in \mathbb{Z}$ liegen dann in der gleichen Klasse, wenn sie gleichen Divisionsrest bzgl. m haben, d.h., wenn sie *kongruent modulo* m sind:

$$[x] = [y] \quad \iff \quad x \bmod m = y \bmod m \quad \iff \quad (x - y) \equiv 0 \pmod{m}.$$

Der Divisionsrest läßt sich durch $x \bmod m = x - m\lfloor x/m \rfloor$ berechnen. Die Rechenregeln für Äquivalenzklassen (Restklassen) entsprechen denen der ganzen Zahlen:



$$[x] + [y] = [(x + y) \bmod m], \quad [x][y] = [(xy) \bmod m],$$

allerdings wie auf einem Kreis (*Ring!*), wo die Werte $m, m+1, \ldots$ mit $0, 1, \ldots$ identifiziert werden.

Bezeichnung: Quotienten-**Ring** $\mathbb{Z}_m := \mathbb{Z}/(m\mathbb{Z})$ mit dem Vertretersystem $\{0, 1, \ldots, m-1\}$ der Restklassen. Diese werden im folgenden ohne Klammern geschrieben mit der Null 0, der Eins 1.

Anwendung Quersummenregel: 9 teilt $n \in \mathbb{N} \iff$ 9 teilt Quersumme.
Aus der Beziehung $10 \equiv 1 \pmod 9$ folgt für die Dezimaldarstellung der Zahl $n = \sum_{i=0}^{k} a_i 10^i$:

$$n \bmod 9 = \Big(\sum_{i=0}^{k} a_i 10^i\Big) \bmod 9 \equiv \Big(\sum_{i=0}^{k} (a_i \bmod 9) \underbrace{(10^i \bmod 9)}_{=1}\Big) \bmod 9 = \Big(\sum_{i=0}^{k} a_i\Big) \bmod 9.$$

Bezeichnung: $\mathbb{P}$=Menge der Primzahlen, $\mathbb{P} \subseteq \mathbb{N}$.

Satz 3.1 *Für $m \in \mathbb{N}$ ist $\mathbb{Z}_m$ ein kommutativer Ring mit Eins. Für $p \in \mathbb{P}$ ist $\mathbb{Z}_p$ ein Körper.*

Erläuterung: In Z_m gilt immer $[m] = [0]$. Wenn nun $m = pq$ ist, folgt $[pq] = [p][q] = [0]$ und daher hat p (analog q) kein multiplikativ Inverses. Denn wäre y ein Element mit $[yp] = [1]$ folgte ein Widerspruch aus $[q] = [1][q] = [ypq] = [y][pq] = [0]$.

Beispiel, Körper $\mathbb{Z}_3$:

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Der folgende Satz ist als probabilistischer Primzahltest verwendbar (Test mit vielen n).

Satz 3.2 *(Kleiner Satz von Fermat) Sei $p \in \mathbb{P}$ und $n \in \mathbb{N}$ kein Vielfaches von p. Dann gilt*

$$n^{p-1} \equiv 1\,(mod\,p).$$

Beweis In der multiplikativen Gruppe $\mathbb{Z}_p \setminus \{0\}$ ist für festes $n \in \mathbb{N}$ die Abbildung

$$\begin{aligned} \{1, \ldots, p-1\} &\to \{1, \ldots, p-1\} \\ x &\mapsto (nx)\,(\mathrm{mod}\,p) \end{aligned}$$

bijektiv, also eine Permutation. Daher gilt

$$u := 1 \cdot 2 \cdots (p-1) \equiv n(2n) \cdots ((p-1)n) \equiv u \cdot n^{p-1}\,(\mathrm{mod}\,p).$$

Mit $v := u^{-1}$ folgt nun $1 \equiv vu \equiv v(un^{p-1}) \equiv n^{p-1}\,(\mathrm{mod}\,p)$. ■

Weitere endliche Körper K existieren nur mit $|K| = p^n$, $p \in \mathbb{P}$, $n \in \mathbb{N}$.
Für jede solche Primzahlpotenz $q = p^n$ existiert (bis auf Isomorphie) genau ein Körper mit $|K| = q = p^n$, das *Galoisfeld* $K = GF(q)$.
Bemerkung: Insbesondere existiert auf den Binärworten ($p = 2$) der Länge n eine Körperstruktur!
Zur Konstruktion: Man geht aus vom *Polynomring*

$$\mathbb{Z}_p[x] := \{a_0 + a_1 x + a_2 x^2 + \ldots :\ a_i \in \mathbb{Z}_p\}$$

über dem Körper Z_p mit der üblichen Addition und Multiplikation (Cauchy-Produkt!). Nun betrachtet man wieder Divisionsreste bei der Polynomdivision und wählt dazu ein festes Polynom $f(x) \in \mathbb{Z}_p[x]$ mit Grad n. Für $g(x), h(x) \in \mathbb{Z}_p[x]$ wird die Relation

$$g(x) \equiv h(x)\,(\,\mathrm{mod}\,f(x)) \quad :\Longleftrightarrow \quad \exists\, q(x) \in \mathbb{Z}_p[x] :\ g(x) - h(x) = q(x)f(x)$$

erklärt. Zu jedem Polynom $g(x)$ kann der Divisionsrest $r(x)$ mit Grad $r \leq n-1$ durch Polynom-Division (s.u.) bestimmt werden. Die so definierte Relation '$\equiv$' ist wieder eine Äquivalenzrelation, und Restklassenbildung führt auf die Rechenregeln

$$[g(x)] + [h(x)] = [g(x) + h(x)], \quad [g(x)][h(x)] = [g(x)h(x)],$$

also ist $\mathbb{Z}_p[x]\,\mathrm{mod}\,f(x)$ ein Ring, der Quotientenring. Dieser kann mit den n-Worten $(a_0, \ldots, a_{n-1})$ über $\mathbb{Z}_p$ identifiziert werden, denn er besitzt das Vertretersystem

$$\{a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} :\ a_j \in \mathbb{Z}_p\}, \tag{1}$$

da die Restklassen $[g(x)] = [r(x)]$ gleich sind, wenn $g(x) = q(x)f(x) + r(x)$ gilt mit Grad $r < n$. Umgekehrt folgt aus der Verschiedenheit von zwei Polynomen $r_1(x), r_2(x)$ vom Grad$< n$, daß natürlich auch $r_1(x) \not\equiv r_2(x) \,(\bmod f(x))$ ist, da $f(x)$ den größeren Grad n hat, also $[r_1(x)] \neq [r_2(x)]$. Daher ist (1) ein Vertretersystem und dies zeigt

$$\Big| \mathbb{Z}_p[x] \bmod f(x) \Big| = p^n.$$

Dieser Ring ist dann wieder ein Körper, wenn $f(x)$ in einem geeigneten Sinn unzerlegbar ist.

Defin. 3.3 *Das Polynom $f \in \mathbb{Z}_p[x]$ heißt irreduzibel, wenn gilt*

$$f(x) = g(x)h(x),\ g(x), h(x) \in \mathbb{Z}_p[x] \ \Rightarrow\ \mathit{Grad}\, g = 0 \ \mathit{oder}\ \mathit{Grad}\, h = 0.$$

Beispiel: $f(x) = x^2 + 1 \in \mathbb{Z}_3[x]$ ist irreduzibel. Denn aus $x^2 + 1 = (x + a)(x + b) = x^2 + (a + b)x + ab$ folgt $b = -a = 2a$ und $1 = ab = 2a^2$. Die letzte Gleichung ist aber unlösbar, da (vgl. Multiplikationstabelle oben) sowohl $2 \cdot 1^2 = 2$ als auch $2 \cdot (2^2) = 2 \cdot 1 = 2$ ist.

Satz 3.4 *$f(x) \in \mathbb{Z}_p[x]$ sei ein irreduzibles Polynom vom Grad n über $\mathbb{Z}_p$, $p \in \mathbb{P}$. Dann ist*

$$\mathbb{Z}_p[x] \ \mathit{mod}\, f(x) \quad \textit{Körper mit } p^n \textit{ Elementen.}$$

(o.Beweis)
Beispiel: $\mathbb{Z}_3[x] \bmod (x^2 + 1)$ hat $3^2 = 9$ Elemente. Die Addition operiert stellenweise:

$$[a_0 + a_1 x] + [b_0 + b_1 x] = [(a_0 + b_0) + (a_1 + b_1)x].$$

Bei der Multiplikation zweier Polynome hat das normale Produkt $(a_0 + a_1 x)(b_0 + b_1 x) = a_0 b_0 + (a_0 b_1 + a_1 b_0)x + a_1 b_1 x^2$ i.a. den Grad 2. Mit $[-1] = [2] \in \mathbb{Z}_3$ führt Polynom-Division hier auf

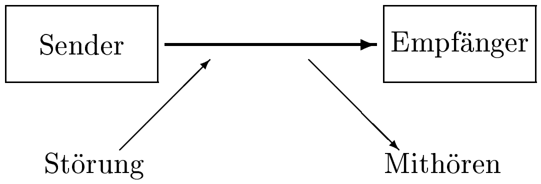
$$\begin{array}{llll} (a_1 b_1 x^2 & +(a_0 b_1 + a_1 b_0)x & +a_0 b_0) : (x^2 + 1) & = a_1 b_1 \\ \ \ a_1 b_1 x^2 & & +a_1 b_1 & \\ \hline & +(a_0 b_1 + a_1 b_0)x & +a_0 b_0 + 2a_1 b_1 & = \text{Divisions-Rest} \end{array}$$

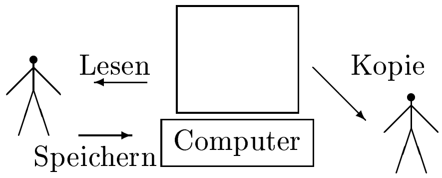
Die Mulitplikation lautet also $[a_0 + a_1 x] \cdot [b_0 + b_1 x] = [(a_0 b_0 + 2a_1 b_1) + (a_0 b_1 + a_1 b_0)x]$. Die vollständige Multiplikationstabelle ist rechts für Koeffizientenpaare (a_0, a_1), (b_0, b_1) ohne Klammern abgedruckt.
Es gilt, z.B.: Einselement $1 = (1, 0)$ und $1 = (2, 0)^2 = (0, 1)(0, 2) = (1, 1)(2, 1) = (1, 2)(2, 2)$.
Anmerkung: Mit $2 = -1$ entspricht die Produktregel in der Form $[(a_0 b_0 - a_1 b_1) + (a_0 b_1 + a_1 b_0)x]$ der der komplexen Zahlen $\mathbb{C}$.

·	00	10	20	01	11	21	02	12	22
00	00	00	00	00	00	00	00	00	00
10	00	10	20	01	11	21	02	12	22
20	00	20	10	02	22	12	01	21	11
01	00	01	02	20	21	22	10	11	12
11	00	11	22	21	02	10	12	20	01
21	00	21	12	22	10	01	11	02	20
02	00	02	01	10	12	11	20	22	21
12	00	12	21	11	20	02	22	01	10
22	00	22	11	12	01	20	21	10	02

3.2 Fehlerkorrigierende Codierungen

Problematik: Bei der Übertragung oder Speicherung von Daten ist, je nach Umfeld, eine Sicherung gegen Verfälschung ($\to$ Fehlerkorrektur) oder Kopieren ($\to$ Kryptographie) erwünscht.





Eine *Codierung* eines (Signals bzw.) Alphabets X ist eine injektive Abbildung

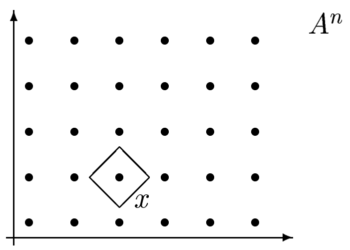
$$c : X \to A^* := \bigcup_{n \geq 0} A^n$$

auf die Menge aller Worte über einem Alphabet A, $|A| = q \geq 2$. Bei Binärcodes ist $A = \{0, 1\}$. Die *Injektivität* ist erforderlich um überhaupt eine eindeutige Decodierung möglich zu machen.

Hier werden nur Codes fester Länge n betrachtet, sogenannte *Blockcodes* mit $c : X \to A^n$. Eine Erkennung oder sogar Korrektur von Fehlern ist dann möglich, wenn c *nicht surjektiv* ist, d.h. $C := c(X) \neq A^n$. Denn nur dann können zulässige ($\in c(X)$) und verfälschte ($\in A^n \setminus c(X)$) Codeworte unterschieden werden.
Beispiel: $X = A^2$, $A = \{0, 1\}$, $m = 2$, Wiederholungscode der Länge $n = 2m$, bei dem jedes Wort zweimal gesendet wird:

$$c : \begin{cases} A^2 \to A^4 & C = c(X) = \{0000, 0101, 1010, 1111\}, \\ a \mapsto aa & \text{nur 4 von } 2^4 = 16 \text{ Codeworten zulässig.} \end{cases}$$

Allgemein konstruiert man Codes oft so, daß zu m Informationszeichen k Kontroll- bzw. Prüf-Zeichen hinzukommen: $n = m + k$.
Ziel: Mit möglichst wenig Kontrollzeichen Sicherheit gegen *zu erwartende* Verfälschungen erreichen (hängt ab vom Umfeld, Art und Häufigkeit von Störungen).
Die Informations*rate* des Codes $\frac{m}{n} = \frac{m}{m+k}$ sollte dabei möglichst hoch sein, damit bei einer bestimmten Sicherheitsstufe möglichst viele zulässige Codeworte existieren.



Prinzip: Decodierung erkennt ein falsches Wort x an der Eigenschaft $x \notin C$, sie kann Verfälschung korrigieren, wenn der Abstand $\min\{d(x, a) : a \in C\}$ zum nächstgelegenen Wort $a \in C$ klein genug.

Fragen: Metrik? Günstige Verteilung von $C \subseteq A^n$?

Begriffe: Zu Worten $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in A^n$ sei

$$d(a, b) := |\{i : a_i \neq b_i\}| \quad \text{die } \textit{Hamming-Distanz}.$$

Diese hat die Eigenschaften einer Metrik in A^n.

Bei fehlertoleranten Codierungen geht man davon aus, daß auf dem betrachteten Übertragungsweg höchstens r Zeichen pro Codewort gestört werden und weitergehende Störungen ein genügend kleines Restrisiko darstellen. Solche Störungen können dann *entdeckt* werden, wenn zulässige Codeworte immer eine größere Hamming-Distanz als r besitzen. Beim doppelten Abstand $2r$ von Codeworten können Fehler sogar *korrigiert* werden. Daher wird definiert

$$d(C) := \min\{d(a,b):\ a \neq b,\ a,b \in C\}, \quad \text{die Code-Distanz.}$$

Der Code C heißt dann

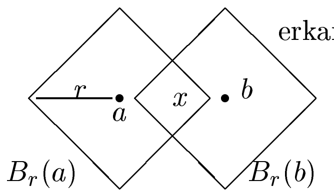
$$\begin{array}{lll} r\text{-fehlererkennend}, & \text{wenn} & d(C) \geq r+1, \\ r\text{-fehlerkorrigierend}, & \text{wenn} & d(C) \geq 2r+1. \end{array}$$

Erläuterung: Die Störung eines Wortes $a \in C$ in maximal r Stellen heißt, daß das verfälschte Wort x in der Kugel

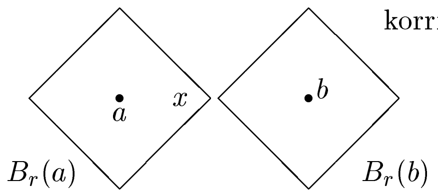
$$B_r(a) := \{x \in A^n : d(a,x) \leq r\}$$

liegt. Wenn kein anderes Wort $b \in C$ in $B_r(a)$ liegt, $B_r(a) \cap C = \{a\}$, ist $x \notin C$ und der Fehler wird *erkannt*. Dieser Fehler kann sogar *korrigiert* werden, wenn das nächste Wort aus C sogar weiter als $2r$ von a entfernt ist:

r-Fehler wird erkannt



r-Fehler wird korrigiert



Der folgende Satz gibt an, wieviele Worte ein fehlerkorrigierender Code enthalten kann.

Satz 3.5 *(Hamming-Schranke) Für den Code $C \subseteq A^n$ mit $|A| = q$ sei $d(C) = 2r+1$ ungerade. Dann gilt*

$$|C| \leq \frac{q^n}{\sum_{j=0}^{r} \binom{n}{j}(q-1)^j}.$$

Beweis Die r-Kugeln um verschiedene Codeworte sind disjunkt, $B_r(a) \cap B_r(b) = \emptyset\ \forall a,b \in C$, $a \neq b$ nach Voraussetzung. Sei nun $a \in C$ beliebig. Dann hat a genau

$$\binom{n}{j}(q-1)^j \text{ Nachbarn } b \text{ mit } d(a,b) = j,$$

da die j Fehlerstellen in $\{1,\ldots,n\}$ auf $\binom{n}{j}$ Möglichkeiten wählbar und in jeder Fehlstelle $q-1$ Werte möglich sind. Also ist der Inhalt einer Kugel

$$|B_r(a)| = \sum_{j=0}^{r} \binom{n}{j}(q-1)^j.$$

Da die (disjunkte!) Vereinigung aller Kugeln Teil von A^n ist, folgt

$$\Big|\bigcup_{a\in C} B_r(a)\Big| = |C| \sum_{j=0}^{r} \binom{n}{j}(q-1)^j \leq |A^n| = q^n. \quad \blacksquare$$

Wenn die r-Kugeln eines r-fehlerkorrigierenden Codes A^n lückenlos überdecken, nennt man den

$$\text{Code } r-\text{perfekt} \quad \iff \quad \bigcup_{a\in C} B_r(a) = A^n.$$

Beispiel: Einfacher Wiederholungs-Code, $A=\{0,1\}$ mit $n=2r+1$: Es gilt $C=\{0\ldots0,1\ldots1\}$, also $|C|=2$ und $d(C)=n=2r+1$. Aus Satz 3.5 folgt

$$\sum_{j=0}^{r}\binom{n}{j}(2-1)^j=\sum_{j=0}^{r}\binom{2r+1}{j}=2^{2r}=2^{n-1}$$

der Code ist also r-perfekt. Interessantere Beispiele folgen hier:

Lineare Codes

Die Konstruktion basiert auf algebraischen Strukturen, daher sei das Alphabet $A=K:=GF(q)$ jetzt ein Körper, wobei $|K|=q$ Primzahlpotenz ist. Dann ist die Menge der n-Worte $A^n=K^n$ ein K-Vektorraum. Jeder lineare Unterraum $C\subseteq K^n$ heißt *linearer Code*, genauer linearer (n,m)-Code, wenn $\dim C=m$ ist. Wegen der Linearität vereinfachen sich die Aussagen zur Distanz:

$$\begin{aligned} d(a,b) &= g(b-a), \qquad g(w):=|\{i:\, w_i\neq 0\}| \text{ Gewicht von } w,\\ d(C) &= \min\{g(a):\, a\in C,\, a\neq 0\}. \end{aligned}$$

Die Hilfsmittel aus der Linearen Algebra führen auf folgende Begriffe (Worte $a\in K^n$ als Zeilenvektoren):

Defin. 3.6 *Es sei $C\subseteq K^n$ ein linearer (n,m)-Code. Eine $m\times n$-Matrix G, deren Zeilen eine Basis von C bilden, heißt* Generatormatrix *des Codes C. Der* duale Code *wird definiert durch*

$$C^\perp := \{w\in K^n:\ 0=wc^\mathsf{T}=:w\cdot c\ \forall c\in C\}.$$

Jede $n\times(n-m)$ Matrix H, deren Spalten eine Basis des dualen Codes $C^\perp$ bilden, heißt Kontrollmatrix *von C.*

Bedeutung: Bei Empfang eines Zeichens $z\in K^n$ ist es die erste Aufgabe, dessen Unversehrtheit zu prüfen. Dazu dient die *Kontrollmatrix.* Da gilt

$$a\in C \iff a\perp C^\perp \iff aH=0,$$

kann die Unversehrtheit von z anhand der Frage $\boxed{zH=0?}$ überprüft werden.

Jede *Generatormatrix* G kann zur Codierung dienen. Wenn die Nachricht im Alphabet $X=K^m$ vorliegt, ist die Abbildung

$$c:\begin{cases} K^m & \to C\\ w & \mapsto wG \end{cases}$$

eine lineare Codierung. Es existieren verschiedene Generatormatrizen (Basen) für C. Der Übergang zu einer anderen Generatormatrix von C entspricht einer Umcodierung, d.h., einem Basiswechsel, im Ausgangsraum X. Der Zusammenhang zwischen der Generator- und Kontrollmatrix eines Codes ist auch einfach:

$$a \in C \iff \{\exists w \in K^m : wG = a\} \iff aH = wGH = 0,$$

Also sind G und H gekoppelt durch die Bedingung

$$G\,H = 0 \ \in K^{m\times(n-m)}. \tag{2}$$

Bei allgemeinen Codes ist zur *Decodierung*, also zur Rekonstruktion einer Nachricht w aus einem Codewort a das lineare Gleichungssystem

$$w\,G = a \quad \text{zu lösen.}$$

Eine Generatormatrix heißt *systematisch*, wenn $G = (I_m, G')$ ist mit $G' \in K^{m\times k}$, $k = n - m$. Dann hat die Codierung für $w \in K^m$ die Form

$$c(w) = w\,G = (w, wG'),$$

d.h., die Codierung besteht aus der Nachricht w und $k = n - m$ Kontroll-/Prüfzeichen wG'. Die Lösung eines Gleichungssystems entfällt, es ist $w = (a_1, \ldots, a_m)$. Auch die Kontrollmatrix ist einfach aufgebaut:

$$H = \begin{pmatrix} -G' \\ I_k \end{pmatrix},$$

denn es gilt $GH = -I_m G' + G' I_k = 0$.

Beispiel: Der binäre $(n, n-1)$ Paritätscode hat (in $GF(2)$: $-1 = 1$) die

$$\text{Generatormatrix } G = \begin{pmatrix} 1 & & 0 & 1 \\ & \ddots & & \vdots \\ 0 & & 1 & 1 \end{pmatrix}, \quad \text{Kontrollmatrix } H = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in K^{n\times 1}.$$

Bei Konstruktion und Existenzfragen linearer Codes hilft folgender Satz. Wegen der linearen Struktur von C braucht man im Beweis oBdA nur die Störung des Null-Wortes zu betrachten.

Satz 3.7 *Es sei $H \in K^{n\times k}, k = n - m$, Kontrollmatrix eines (n,m)-Codes $C \subseteq K^n$. Wenn* **jeweils** *r Zeilen von H linear unabhängig sind, dann ist $d(C) \geq r + 1$.*

Beweis Zu zeigen ist, daß $g(a) \geq r + 1$ gilt für jede nichttriviale Lösung von $aH = 0$. Mit z_i wird die i-te Zeile von H bezeichnet. Da je r Zeilen von H linear unabhängig sind, gilt für alle Indexmengen $\{i_1, \ldots, i_\ell\} \subseteq N,\ \ell \leq r$, die Folgerung

$$a_{i_1} z_{i_1} + \ldots + a_{i_\ell} z_{i_\ell} = 0 \quad \Rightarrow \quad a_{i_1} = \ldots = a_{i_\ell} = 0.$$

Für $a \in C$ mit $g(a) \leq r$ wählt man die Indexmenge der nichttrivialen Komponenten und erhält so aus $aH = 0$ auch $a = 0$. ■

Bemerkung: a) In dieser Vorlesung wird nicht allgemein auf die Frage eingegangen, wie bei einem r-fehlerkorrigierenden Code zu einem verfälschten Wort $z \in K^n$ mit $0 < g(zH) \leq r$ das nächstgelegene Wort $a \in C$ rekonstruiert wird. Dazu müssen die möglichen Worte mit dem *Syndrom* $y := zH \neq 0$ betrachtet werden.

b) Eine Umkehrung des Satzes besagt, dass für $d(C) = r+1$ auch eine geeignete Kontrollmatrix zu Satz 3.7 existiert. Daraus folgt das

Korrolar: Der (n, m)-Code C sei r-fehlererkennend $\Rightarrow r \leq k = n - m$, da $d(C) \leq k + 1$.

Beweis Die Zeilen der Kontrollmatrix sind k-Vektoren. Da höchsten k linear unabhängige existieren, folgt aus Satz 3.7 $d(C) \leq k + 1$, der Code erkennt höchstens k Fehler. ■

Also: Zur Erkennung von r Fehlern sind mindestens r Prüfzeichen erforderlich.

Beispiel: Hamming-Codes sind 1-perfekte Codes über $K = GF(q)$, d.h., es gilt $d(C) = r+1 = 3$. Nach dem Korollar ist $k = n - m \geq r = 2$ erforderlich und in der Kontrollmatrix müssen je $r = 2$ Zeilen lin.unabh. sein, also nicht Vielfache anderer. Zunächst gibt es $q^k - 1$ nichttriviale Zeilen, von denen jede $q - 1$ nichttriviale Vielfache hat. Daher gibt es

$$\frac{q^k - 1}{q - 1} = 1 + q + \ldots + q^{k-1}$$

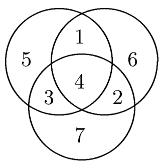
mögliche Zeilen von H, und es folgt $n \leq (q^k - 1)/(q - 1)$.

Bei Gleichheit, $n = (q^k - 1)/(q - 1) = m + k$, ist der Code dann 1-perfekt, da die Hamming-Schranke aus Satz 3.5 angenommen wird:

$$\frac{q^n}{|C|} = \frac{q^n}{q^m} = q^k = 1 + \frac{q^k - 1}{q - 1}(q - 1) = 1 + n(q - 1) = \sum_{j=0}^{1} \binom{n}{j} (q - 1)^j.$$

Für $q = 2, k = 3$ und $n = (2^3 - 1)/(2 - 1) = 7$ hat die Kontrollmatrix die rechts gezeigte Gestalt. Dieser Code ist auch systematisch mit $G = (I_4, G')$. Die Bedingung $aH = 0$ entspricht der Bedingung, daß in jedem der 3 gezeigten Kreise die Komponentensumme gerade ist. Zahlen in den Kreisen: Indizes.

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} G' \\ I_3 \end{pmatrix}$$



Fehlerbehandlung: Der Code kann einen Fehler korrigieren. Dessen Position kann aus den betroffenen Kreisen rekonstruiert werden. Bei Störung etwa von $a_1 \to z_1$ sind die beiden oberen Kreise betroffen. Bei einer geeigneten Vertauschung der Indizes gibt das Syndrom zH als Binärwort die Position des gestörten Zeichens an.

Codes mit speziellen Zielsetzungen:

- *Zyklische Codes:* Bei diesen sind mit jedem Wort $a = (a_1, \ldots, a_n) \in C$ auch alle zyklisch rotierten, z.B., $(a_2, \ldots, a_n, a_1) \in C$ Codeworte. Ihr Vorteil ist, daß die De-/Codierung einfach und mit schneller Hardware (Schieberegister $\sim$ Differenzengleichungen) durchführbar und eine Analyse im Polynomring $Z_2[x]$ möglich ist.

- *Fehlerbündel* (*burst*): Starke Störungen (Blitz, Schaltvorgang, Kratzer auf CD) betreffen oft mehrere aufeinanderfolgende Zeichen (Bit). Das Sicherheitsniveau gegen solche Störungen kann durch Codes über $K = GF(q)$ mit $q > 2$ verbessert werden. Zur Codierung von CDs wird, z.B., folgender Code verwendet.
  Reed-Solomon-Code über $GF(q)$: Sei
  $$K = GF(q) = \{0, 1, z_2, \ldots, z_{q-1}\}, \quad (z_0 = 0,\ z_1 = 1).$$
  Die Kontrollmatrix wird definiert durch
  $$H = \begin{pmatrix} 0 & 0 & 0 & \ldots & 1 \\ 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 1 & \ldots & 1 \\ 1 & z_2 & z_2^2 & \ldots & z_2^{k-1} \\ \vdots & \vdots & & & \\ 1 & z_{q-1} & z_{q-1}^2 & \ldots & z_{q-1}^{k-1} \end{pmatrix} = \begin{pmatrix} 0\,0 \ldots 0\,1 \\ (z_i^j)_{i,j=0}^{q-1,k-1} \end{pmatrix} \in K^{(q+1)\times k}$$
  Hier steht unterhalb der nullten Zeile eine *Vandermonde-Matrix,* daher sind je k Zeilen linear unabhängig und nach Satz 3.7 ist $d(C) = k + 1$, der Code ist k-fehlererkennend. Dieser $(n, m) = (q + 1, q + 1 - k)$-Code hat den Informationsgehalt
  $$\frac{m}{n} = \frac{q + 1 - k}{q + 1} = 1 - \frac{k}{q + 1}.$$
  Speziell für $q = 2^\ell$ hat der RS-Code eine größere Korrekturfähigkeit gegen Fehlerbündel als ein Binärcode der Länge ℓn. Als Beispiel sei $q = 2^8$, $n = 257$, der Code sei r-korrigierend, d.h. $k = 2r$. Ein Codewort $(a_1, \ldots, a_{257}) \in C$ kann mit $8n = 2056$ Bit codiert werden (induzierter Binärcode C^*). Fehler in $\leq 8r - 7$ (z.B. 33) aufeinanderfolgenden Bits betreffen höchsten r Byte (im Beispiel $r = 5$)

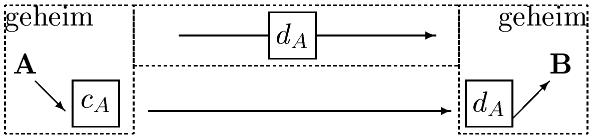
	xxxx	xxxx	xxxx	xxxx	x

  und können daher korrigiert werden mit $16r = 8k$ Prüf-Bit (im Zahlbeispiel $8k = 80$). Für einen reinen Binärcode B mit $8n$ Stellen, der $8r - 7 = 33$ (dann aber beliebig verteilte) Fehler korrigieren könnte, gilt nach der Hammingschranke (Satz 3.5) im Beispiel
  $$\frac{2^{8n}}{|B|} \geq \sum_{j=0}^{8r-7} \binom{2056}{j} \geq \binom{2056}{33} \geq \frac{(2024)^{33}}{33!} > 2^{239}.$$
  Dieser Binärcode benötigt also mehr als 239 Prüfbit gegenüber den 80 Prüfbit des Reed-Solomon-Codes.

3.3 Kryptographie, Asymmetrische Verfahren

Die Zielsetzung ist der Austausch von Nachrichten zwischen Personen A (Alice) und B (Bob) über eine öffentliche Verbindung so, daß Dritte den Inhalt nicht verstehen. Die Codierung der Nachricht ist wieder eine injektive (hier i.a. bijektive) Abbildung $c : X \to Y$ auf der Menge der Nachrichten X. Jetzt soll aber für Dritte die Decodierung $c^{-1} =: d$ nicht möglich sein. Oft wählt man c aus einer bestimmten Klasse von Funktionen, die von Parametern abhängt. Diese Parameter werden dann *Schlüssel* genannt.

Symmetrische Verfahren (klassisch): Absender A und Empfänger B treffen eine geheime Absprache über beide Funktionen c **und** d (Vorbedingung: *sicherer Schlüsselaustausch*).



Beispiel Substitutions-Code: Die Nachrichten sind Worte über einem Alphabet A (Bits, Buchstaben, Zeichen-Gruppen), also $X = A^n$. Zur Codierung wird jedes Zeichen durch ein anderes ersetzt, d.h., c hat mit Substitutionen $c_i : A \to A$ die Form

$$c: \; x = (x_1, \dots, x_n) \mapsto (c_1(x_1), c_2(x_2), \dots, c_n(x_n)).$$

Einfache Variante: $c_i(x) = (x+a_i) \bmod |A|$, Addition (zyklische) eines Schlüsselbegriffs $(a_1, \dots, a_n)$. Beim Schlüsselwort `APRIL` und Addition modulo 27:

```
DISKR ETE-M ATHEM ATIK
APRIL APRIL APRIL APRI
EYKTD FNW.. ..... ....
```

In der gezeigten Variante ist das Verfahren unsicher, da der Schlüssel wiederholt wird ($\Rightarrow$ AT trifft zweimal auf AP) und aus Klartext besteht ($\to$ Ansatz für *Attacken*). Umgekehrt ist das Verfahren beweisbar sicher, wenn die Zeichen des Schlüssels statistisch ideal gleichverteilt sind und der Schlüssel nur einmal verwendet wird ($\to$ Einmalschlüssel, *one-time-pad*). Das Verfahren basiert aber auf dem sicheren Austausch der Schlüssel.

Asymmetrische Verfahren (Diffie/Hellman 1976): Hier setzt man die Kenntnis von *Einweg-Funktion*en (Falltür-Funktionen) c voraus mit folgenden Eigenschaften

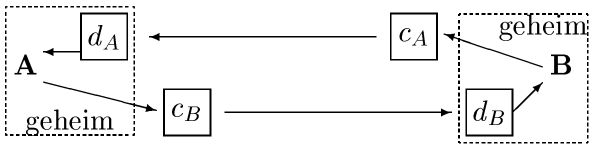
1. Die Auswertung $c(x)$ der Codierungsfunktion für eine Nachricht $x \in X$ ist effizient durchführbar (polynomieller Aufwand).

2. Aus Kenntnis einer geheimen Eigenschaft von c kann die Decodierungsfunktion $d = c^{-1}$ effizient konstruiert und ausgewertet werden.

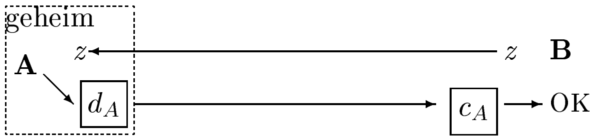
3. Bei Kenntnis der Abbildung c alleine ist c^{-1} praktisch nicht berechenbar (Aufwand exponentiell).

Beispiele von Einweg-Funktionen folgen später, ihr Einsatz ermöglicht folgende Anwendungen:

- Sichere Kommunikation ohne sicheren Schlüsselaustausch (public-key-Kryptosysteme): Jeder Teilnehmer A und B konstuiert *sein persönliches* Paar c_A, d_A bzw. c_B, d_B an Abbildungen und schickt nur die Beschreibung von c an den anderen (d.h., A veröffentlicht c_A, hält d_A geheim). Nun kann B seine Nachricht x_B verschlüsselt als $c_A(x_B)$ an A senden und nur A kann diese mit d_A lesen, umgekehrt schickt A die Nachricht x_A als $c_B(x_A)$ an B.



- Digitale Unterschrift, - Ausweis: A kann durch die Kenntnis von d_A seine Identität nachweisen, indem er eine beliebige Nachricht z mit d_A codiert und das Paar $(z, d_A(z))$ an B schickt. B prüft die Korrektheit durch Anwendung der bekannten Abbildung $c_A = d_A^{-1}$. Zur Absicherung gegen das Kopieren alter Nachweise kann z aus der zu signierenden Nachricht gebildet (Unterschrift) oder von B an A geschickt werden (Ausweis).



Spezielle Einwegfunktionen Die gängigen Verfahren operieren in Restklassenringen $\mathbb{Z}_m$, da diese Strukturen sehr gut untersucht sind. Zur Absicherung verwendet man dabei sehr große Zahlen m $(> 10^{200})$, interpretiert Nachrichten also als Zahlen mit $n > 200$ Dezimalstellen.

Diskreter Logarithmus Bei nichtlinearen Abbildungen c sind die Auswertungen von c und c^{-1} oft von unterschiedlicher Komplexität, z.B., bei Exponentiation und Wurzel- bzw. Logarithmus-Berechnung. Es sei $p \in \mathbb{P}$, $a \in \mathbb{Z}_p, a \neq 1$, und

$$\exp_a : x \mapsto y \equiv a^x \pmod p, \quad x \in \mathbb{Z}_p.$$

Die Exponentialfunktion $\exp_a : \mathbb{Z}_p \to \mathbb{Z}_p$ ist aber i.a. nicht injektiv. Nach dem Satz 3.2 von Fermat gilt für jedes $a \in \mathbb{Z}_p$

$$a^{p-1} \equiv 1 \pmod p.$$

Eine Zahl a, für die a^{p-1} die *erste* Potenz mit Divisionsrest 1 ist, heißt *Primitvwurzel* modulo p:

$$a \text{ Primitvwurzel modulo } p \;:\Longleftrightarrow\; \{a^x \equiv 1 \pmod p, x \in \mathbb{N} \Rightarrow x \geq p-1\}.$$

Für diesen Fall ist die Abbildung $\exp_a : x \mapsto a^x \pmod p$ tatsächlich bijektiv auf $\{1, \ldots, p-1\}$ und damit umkehrbar (d.h., eine Permutation). Die Umkehrung heißt *diskreter Logarithmus*. Die Anzahl der Primitivwurzeln ist gleich der Anzahl der zu $p-1$ teilerfremden Zahlen $\leq p-1$. *Beispiel:* $p = 11$, Primitivwurzeln sind $a = 2, 6, 7, 8$. Für $a = 7$ etwa ist die Folge $(7^i \pmod p : 1 \leq i \leq 10) = (7, 5, 2, 3, 10, 4, 6, 9, 8, 1)$.

Man kennt bisher keinen polynomiellen Algorithmus zur Berechnung des diskreten Logarithmus (aktuelle Abschätzung: $O(\exp(C[n^{1/3}(\log n)^{2/3}])$, $n = \log p$), daher ist Bedingung 3 für Einwegfunktionen erfüllt. Auch die erste Bedingung wird eingehalten, da der Aufwand für die Auswertung der Exponentialfunktion i.w. nur quadratisch mit der Stellenzahl wächst (Satz 3.8). Zunächst wird die Abwicklung der Kommunikation besprochen.

Das *El-Gamal-Verfahren* basiert auf der Sicherheit des diskreten Logarithmus.

Öffentlich bekannt:	(große) Primzahl p und Primitivwurzel a modulo p.
Nutzer A wählt:	$r < p - 1$ geheim, veröffentlicht seinen Schlüssel $s \equiv a^r \pmod p$
Nutzer B übermittelt x_B:	wählt $k < p - 1$ zufällig, hält $K \equiv s^k \pmod p$ geheim,
	sendet $w :\equiv a^k \pmod p$ und $y :\equiv K x_b \pmod p$.
Nur Nutzer A weiß:	$K \equiv s^k \pmod p \equiv a^{rk} \pmod p \equiv w^r \pmod p$;
	entschlüsselt durch Division $x_b \equiv (y/K) \pmod p$.

K hat hier die Funktion eines geheimen Einmalschlüssels. Ohne Kenntnis von r müßte der Logarithmus von s oder w berechnet werden, was in dem genannten Größenbereich aber undurchführbar ist. Zur Auswertung der Exponentialfunktion selbst gibt es aber ein einfaches Verfahren.

Satz 3.8 *Es seien $m, k \in \mathbb{N}$. Die Berechnung von m^k erfordert höchstens $2\lceil \log_2 k \rceil$ Multiplikationen, für $k = 2^n$ genau $n = \log_2 k$ Multiplikationen.*

Beweis a) Bei $k = 2^n$ erfordert n-faches Quadrieren $n = \log_2 k$ Multiplikationen:

$$m^k = m^{2^n} = \left(m^{2^{n-1}}\right)^2.$$

b) Allgemein sei $k = \sum_{j=0}^{n} b_j 2^j$, $b_j \in \{0, 1\}$, die Binärdarstellung von k. Dann gilt

$$m^k = m^{\sum b_j 2^j} = \prod_{j=0}^{n} m^{b_j 2^j} = \prod_{b_j = 1} m^{2^j}.$$

Parallel zur Berechnung von $m^2, \ldots, m^{2^n}$ benötigt so die von m^k maximal n weitere Multiplikationen. ■

Da jede Multiplikation (modulo p) von n-stelligen Zahlen $O(n^2)$ Ziffern-Operationen kostet, werden insgesamt $O(n^3)$ Ziffern-Operationen für $m^k \pmod p$ benötigt, wenn p (und $m, k < p$) höchstens n Stellen haben.

Das *RSA-Verfahren* (Rivest, Shamir, Adleman) basiert auf der Schwierigkeit der Faktorisierung großer Zahlen. Zur Ver- und Entschlüsselung wird auch hier die Potenzierung verwendet. Der öffentliche Schlüssel besteht aus der (großen) Basis m und dem Exponenten s, ein geheimer Exponent g dient zur Entschlüsselung. Also ist mit $x, y \in \mathbb{Z}_m$

$$\begin{aligned} c: &\quad x \mapsto y \equiv x^s \pmod m, \qquad (3)\\ d: &\quad y \mapsto x \equiv y^g \pmod m. \end{aligned}$$

Hintergründe:

— Die Basis m ist Produkt von zwei (geheimen) Primzahlen $p, q \in \mathbb{P}$: $m = pq$.

— Für die Exponenten s, g gilt $1 \leq s, g \leq r := (p-1)(q-1)$ und

$$sg \equiv 1 \,(\bmod r).$$

Dann sind s, g teilerfremd zu r und gehören zu zueinander inversen Abbildungen c und d:

Satz 3.9 *Es sei $m = pq$, $p, q \in \mathbb{P}$ und $sg \equiv 1 \,(\bmod r)$, $s, g < r := (p-1)(q-1)$. Dann gilt*

$$x^{sg} \equiv x \,(\bmod m) \quad \forall x \in \mathbb{Z}_m.$$

Beweis Nach Satz 3.2 (Fermat) gelten für $x \notin \{0, p, q\}$ jeweils $x^{p-1} \equiv 1(\bmod p)$ und $x^{q-1} \equiv 1(\bmod q)$. Mit $sg = 1 + kr = 1 + k(p-1)(q-1)$, $k \in \mathbb{N}$, folgt

$$x^{sg} = x \cdot (x^{p-1})^{k(q-1)} \equiv x \cdot 1 \,(\bmod p).$$

Analog gilt $x^{sg} \equiv x(\bmod q)$. Daher gibt es Faktoren $a_1, a_2, a_3 \in \mathbb{Z}$ mit $x^{sg} - x = a_1 p = a_2 q$, also $x^{sg} - x = a_3 pq$. Dies gilt auch für $x = p, q$. ■

Konstruktion: Jede Primzahl $g > \max\{p, q\}$ ist teilerfremd zu r. Den öffentlichen Schlüssel s berechnet man dazu mit dem euklidschen Algorithmus für $ggT(g, r)$ (Wert ist 1), der die Faktoren s, k in der Darstellung $1 = sg + kr$ liefert, vgl. Beispiel.

Beispiel: $p = 47, q = 59, m = pq = 2773$, $r = 46 \cdot 58 = 2668$. Wähle $g = 157 \in \mathbb{P}$.

$$\text{Euklid zu } ggT(2668, 157): \quad \begin{cases} 2668 = & 16 \cdot 157 & + \boxed{156}, \\ 157 = & 1 \cdot 156 & + \boxed{1}, \end{cases}$$

$\Rightarrow \boxed{1} = 157 - 1 \cdot \boxed{156} = 157 - 1 \cdot (2668 - 16 \cdot 157) = 17 \cdot 157 - 1 \cdot 2668$, d.h., $s = 17$.
Codierung: Buchstaben werden dezimal codiert, $\_ = 00, A = 01, \ldots, Z = 26$ und je zwei Zeichen zu einer Dezimalzahl $\leq 2626 < m$ zusammengefaßt:

Text:	KO	MM	E_	MO	RG	EN
Codierung:	1115	1313	0500	1315	1807	0514
Kryptogramm:	1379	2395	1655	0422	0482	1643

Attacken: Eine Entschlüsselung ist möglich, wenn die Faktoren p und q von m berechnet werden können. Wenn die Faktoren ungefähr gleich groß ($> 10^{100}$) sind, sich aber doch um einige Zehnerpotenzen unterscheiden, ist diese Faktorisierung allerdings ähnlich aufwendig wie die Berechnung des diskreten Logarithmus. Bei ungünstiger Wahl der Werte p, q, g kann man die Nachricht aber auch ohne Kenntnis von p, q entschlüsseln. Denn zu jeder Nachricht $x = c^{-1}(y)$ gibt es einen Index k so, daß die mehrfache Anwendung $c^k(y) = c(\cdots c(y) \cdots)$ der Codierfunktion (3) die Nachricht $x = c^{-1}(y) = c^{k+1}(x)$ aufdeckt, dieser Index sollte daher große Wert haben. Z.B. gilt im obigen Zahlbeispiel mit $y = 1787$ wieder $c^4(y) = y$, daher war $c^3(y) = 0518$ die Nachricht.

Wenn solche Fälle ausgeschlossen werden, geht man bisher davon aus, daß die Faktorisierung von m auch erforderlich ist, um das RSA-System zu brechen. Die Auswahl 'guter' öffentlicher Schlüssel ist daher eine weitere Aufgabe von Trust-Centern.

Abschließende Bemerkungen:

• Das RSA-Verfahren ist wegen der erforderlichen großen Schlüssellängen für viele Anwendungen zu aufwendig und wird daher oft nur zum sicheren Austausch von Schlüsseln für klassische Verfahren verwendet. In der Praxis wird häufiger das El-Gamal-Verfahren eingesetzt, z.B., beruht das digitale Unterschriftsverfahren DSA der amerikanischen Normbehörde NIST darauf. Kürzere Schlüssel sind auch dann noch sicher, wenn man algebraische Operationen auf *elliptischen Kurven* verwendet.

• Bei der Absicherung von Kommunikation darf man sich nicht nur auf die Sicherheit der kryptographischen Methoden verlassen, sondern muß die gesamte Abwicklung der Kommunikation (Protokoll) mit einbeziehen. Z.B. ist auch bei asymmetrischen Methoden eine reine bilaterale Verbindung nicht sicher, nur der Schlüsselaustausch über eine Treuhandstelle (Trust-Center) verhindert die Attacke durch eine dritte Person C, die eine direkte Verbindung zwischen A und B vortäuscht,

$$\boxed{A} \longleftrightarrow \boxed{C} \longleftrightarrow \boxed{B}$$

tatsächlich aber die zwischen A und B ausgetauschten Schlüssel durch eigene ersetzt.

• Die Abschätzung der Sicherheit einzelner Verfahren ist sehr schwierig, wenn sie auf Einschätzungen über die (jetzigen und zukünftigen) Fähigkeiten Dritter basiert, da diese in der Regel geheim gehalten werden. So war wahrscheinlich das Prinzip der Einwegverfahren verschiedenen Geheimdiensten schon lange bekannt. Auch die Angaben über die Sicherheit des RSA-Verfahrens müssen laufend nach unten korrigiert werden (vgl. Links auf DiMa-Seite). Kürzlich wurde, z.B., mit erheblichem Rechnereinsatz das RSA-155-Verfahren 'geknackt'. Ursprüngliche Schätzungen hatten dafür einen Zeitaufwand von 50 Millionen Jahren genannt.

Teil C

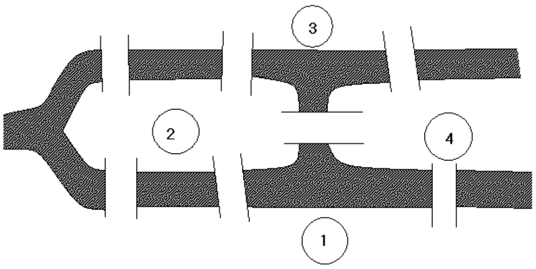
Graphen & Bäume

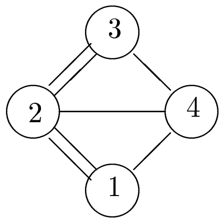
4 Graphentheorie

Viele Fragestellungen, in denen unregelmäßige Beziehungen zwischen Objekten oder Zuständen auftreten, können abstrakt als Probleme in Graphen interpretiert und behandelt werden. Bei Computeranwendungen ist dies v.a. dann angebracht, wenn komplexe, dynamische Datenstrukturen (Zeiger) eingesetzt werden, auch die Links im WWW erzeugen einen Graphen. Auch durch die Verwendung von Diagrammen zur Veranschaulichung von Beziehungen (*Relationen*) werden Graphen benutzt.

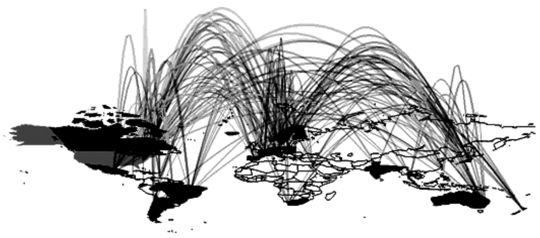
Beispiele:

• *Klassisch:* Eulers Königsberger Brückenproblem (1736): Gibt es einen Rundgang, der alle sieben Brücken genau einmal benutzt (Euler-Zug)? Situation als Graph:





• *Aktuell:* Die Links im Internet erzeugen einen Graphen mit den WWW-Seiten als Knoten. Moderne Suchmaschinen (Google) werten nicht nur die Seiteninhalte, sondern auch die Struktur dieses Graphen aus.



4.1 Bezeichnungen

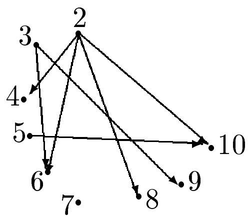
Defin. 4.1 *Ein* Graph $G = G(P, L)$ *besteht aus einer Menge* $P = P(G)$ *von* Ecken *(Punkten) und* $L = L(G) \subseteq \binom{P}{2}$ *von 2-Mengen* $\{u, v\}, u, v \in P, u \neq v$*, die* Kanten *(Linien) heißen. Bei einem* gerichteten Graphen *(Digraph) ist* $L \subseteq P \times P$*, Kanten sind Paare* (u, v)*,* $u, v \in P$*,* $u \neq v$*.*

Erläuterungen:

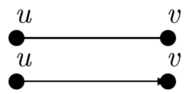
• Diese Definition entspricht der eines schlichten Graphen und schließt Schlingen $\{u, u\}$ und Mehrfachkanten wie in Eulers Brückenproblem aus ($\rightarrow$ Multigraph).

- Die Menge der Kanten L kann als eine Relation auf P interpretiert werden (symmetrisch bei ungerichteten Graphen), umgekehrt definiert jede (un)symmetrische Relation auf P einen (Di)Graphen.

  Z.B. $P = \mathbb{N}$ und Teilbarkeitsrelation:



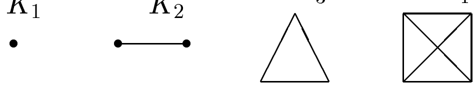
- Zur graphischen Darstellung von Graphen zeichnet man Ecken als Punkte, evtl. mit Bezeichnung, und verbindet diese bei
  ungerichteten Kanten $\{u, v\}$ durch Linien
  gerichteten Kanten (u, v) in Digraphen durch Pfeile



Spezielle einfache Graphen:

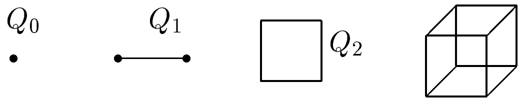
- Vollständiger Graph K_n: $|P| = n$, $L = \binom{P}{2}$, $|L| = \binom{n}{2}$.

K_1 K_2 K_3 K_4

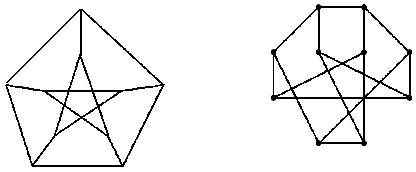


- Hyperwürfel Q_n: $|P| = 2^n$, die Ecken sind als n-Worte über $A := \{0, 1\}$ interpretierbar, mit der Hamming-Distanz d ist $L = \{\{u, v\} : u, v \in A^n, d(u, v) = 1\} \Rightarrow |L| = n2^{n-1}$.

Q_0 Q_1 Q_2 Q_3



- Petersen-Graph, $|P| = 10$, $|L| = 15$, verschiedene Darstellungen(!):



Da es jeweils viele unterschiedliche graphische Darstellungen des gleichen Graphen gibt, ist es ein wichtiges Problem, zu entscheiden, ob zwei Graphen "gleich", d.h., isomorph sind.

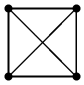
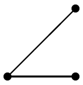
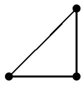
Begriffe I bei Graphen $G(P, L)$, Kanten werden ab jetzt ohne Klammern geschrieben:

Ordnung	von G ist $\|P\|$
Größe	von G ist $\|L\|$
Teilgraph	Ein Graph $G'(P', L')$ ist Teilgraph von $G(P, L)$, wenn $P' \subseteq P$, $L' \subseteq L$ ist. G' heißt von P' *induzierter* Teilgraph, wenn $L' = L \cap \binom{P'}{2}$.
Isomorphie	$G(P, L) \cong G'(P', L')$, wenn eine Bijektion $f : P \to P'$ existiert mit $uv \in L \iff f(u)f(v) \in L'$
Weg	ist eine Folge $W = (u_1, u_2, \ldots, u_n)$ von verschiedenen Ecken $u_i \in P$ mit $u_i u_{i+1} \in L\ \forall i = 1, \ldots, n-1$. Länge des Wegs ist $n - 1$= Anzahl der Kanten
Kreis	(Zyklus) ist ein geschlossener Weg $Z = (u_1, u_2, \ldots, u_n)$ mit $u_n u_1 \in L$; die Länge des Kreises ist n
Abstand	$d(u, v)$ von Ecken $u, v \in P$ ist die Länge des kürzesten Wegs $u \leftrightarrow v$; speziell wird gesetzt $d(u, u) := 0$, sowie $d(u, v) := \infty$, wenn kein Weg $u \leftrightarrow v$ existiert

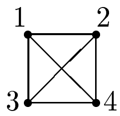
Durchmesser	von G: $D(G) := \max\{d(u,v) : u,v \in P\}$.
Zusammenhang	G heißt zusammenhängend $\iff$ $\forall u,v \in P \, \exists$ Weg $u \leftrightarrow v$, d.h., $D(G) < \infty$
Komponenten	die Eigenschaft $d(u,v) < \infty$ (Erreichbarkeit) definiert eine Äquivalenzrelation auf P, Äquivalenzklassen heißen Komponenten von G
Nachbarn	Knoten $u,v \in P$ heißen benachbart (adjazent) $\iff$ $uv \in L$; die Kante uv heißt *inzident* zu u und v; u,v sind End-Ecken von uv.
Ecken-Grad	sei $N(u)$ die Menge der Nachbarn von $u \in P$, dann heißt $g(u) := \|N(u)\|$ Grad von u. Eine Ecke u heißt *isoliert*, wenn $g(u) = 0$.

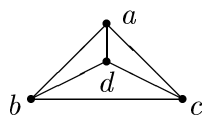
Beispiel:

- Teilgraph, die Graph-Eigenschaft wird bei G' vorausgesetzt, z.B.

  $G =$  $G' =$  von P' induziert: 

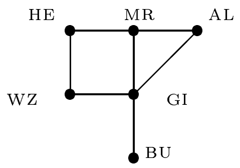
- Isomorphie:

  $G =$ 

  $G' =$ 

  $L = \{12, 13, 14, 23, 24, 34\}$, $L' = \{ab, ac, ad, bc, bd, cd\}$. Beide Graphen sind isomorph mit $f(1) = a$, $f(2) = b$, $f(3) = c$, $f(4) = d$

- Weg, Kreis, Straßenverbindungen um Marburg:

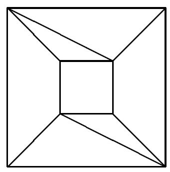
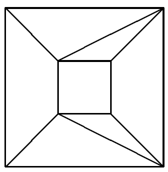
  $MR - GI - BU$ ist ein Weg, $MR - GI - WZ - HE$ ein Kreis:

- Abstand, Durchmesser: $d(MR, BU) = 2$, der Durchmesser des Straßengraphen ist $D(G) = 3 = d(HE, BU)$.

- Nachbarn, Eckengrad, im Straßengraphen:

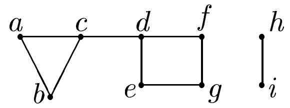
u	MR	AL	GI
$N(u)$	$\{HE, GI, AL\}$	$\{MR, GI\}$	$\{MR, AL, BU, WZ\}$
$g(u)$	3	2	4

  Der Vergleich der Eckengrade von zwei Graphen ist ein einfacher Test auf Nicht-Isomorphie. Bei Anwendung einer Bijektion bleiben die Eckengrade aller Ecken gleich und auch die Eckengrade der End-Ecken jeder Kante:

  G  $\stackrel{?}{\cong}$  G'

  Die Eckengrade sind in beiden Graphen $3, 3, 3, 3, 4, 4, 4, 4$, allerdings treten nur in G' Kanten mit 2 End-Ecken vom Grad 3 auf.

- G ist nicht zusammenhängend, hat 2 Komponenten $\{a, b, .., g\}$ und $\{h, i\}$. Erreichbarkeit (Transitivität der Äquivalenzrelation): $d(a, c) < \infty$, $d(c, e) < \infty \Rightarrow d(a, e) \leq d(a, c) + d(c, e) < \infty$.



Satz 4.2 *Für jeden Graph $G(P, L)$ gilt*

$$\sum_{u \in P} g(u) = 2|L|.$$

Beweis Zu jeder Kante $k = ab \in L$ werden die beiden Paare (a, k) und (b, k) gebildet. Die Gesamtzahl dieser Paare ist $2|L|$ (Zählung über $k \in L$). Die Anzahl aller Paare (u, k) mit $u = a$ ist $|N(a)| = g(a)$. Bei Summation über alle Ecken ergibt sich der gleiche Summenwert $2|L|$. ■

Korollar *Die Anzahl der Ecken ungeraden Grades in einem Graphen ist gerade.*

Beweis Da die Gradsumme in Satz 4.2 gerade ist, gilt bei Summation modulo 2 also $0 \equiv \sum_{u \in P} g(u) (\bmod 2)$. Daher tritt eine gerade Anzahl von Summanden $\equiv 1 (\bmod 2)$ auf. ■

Für Graphen mit konstantem Eckengrad $r = g(u) \, \forall u \in P$ (sogen. *r-reguläre* Graphen, etwa die vollständigen mit $r = n - 1$) gilt daher

$$\sum_{u \in P} g(u) = r|P| = 2|L|.$$

Bei den schon betrachteten Beispielen war bei K_n die Ordnung $|P| = n$, $r = n - 1$ und daher $|L| = n(n-1)/2$ und bei Q_n war $|P| = 2^n$, $r = n$, also $|L| = n2^{n-1}$.

4.2 Darstellung von Graphen

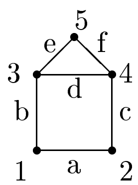
Zur Handhabung von 'dünnen' Graphen (mit $|L| \ll |P|^2/2$) in Algorithmen sind in der Regel dynamische Speicher-Strukturen angebracht (Zeiger, Index-Zugriff auf Felder). Die vollständige Speicherung der jetzt eingeführten Matrizen ist daher selten sinnvoll. Die folgenden Begriffe schaffen aber eine Querverbindung zur Linearen Algebra machen dadurch viele neue Hilfsmittel verfügbar.

Defin. 4.3 *Sei $G(P, L)$ ein Graph mit $P = \{u_1, \ldots, u_n\}$ und $L = \{k_1, \ldots, k_m\}$. Man definiert dazu die*

$$\textit{Adjazenzmatrix} \quad A = (a_{ij}) \in \mathbb{N}_0^{n \times n}, \qquad a_{ij} = \begin{cases} 1 & \textit{wenn } u_i u_j \in L \\ 0 & \textit{sonst} \end{cases}$$

$$\textit{Inzidenzmatrix} \quad B = (b_{ij}) \in \mathbb{N}_0^{n \times m}, \qquad b_{ij} = \begin{cases} 1 & \textit{wenn } u_i \in k_j \\ 0 & \textit{sonst} \end{cases}$$

Beispiel: Ungerichteter Graph mit $n = 5$, $m = 6$, $L = \{a, b, c, d, e, f\}$.



$$B=\begin{pmatrix} 1 & 1 & . & . & . & . \\ 1 & . & 1 & . & . & . \\ . & 1 & . & 1 & 1 & . \\ . & . & 1 & 1 & . & 1 \\ . & . & . & . & 1 & 1 \end{pmatrix},\; BB^{\mathsf{T}}=\begin{pmatrix} 2 & 1 & 1 & . & . \\ 1 & 2 & . & 1 & . \\ 1 & . & 3 & 1 & 1 \\ . & 1 & 1 & 3 & 1 \\ . & . & 1 & 1 & 2 \end{pmatrix},\; A=\begin{pmatrix} . & 1 & 1 & . & . \\ 1 & . & . & 1 & . \\ 1 & . & . & 1 & 1 \\ . & 1 & 1 & . & 1 \\ . & . & 1 & 1 & . \end{pmatrix}\begin{array}{c} 2 \\ 2 \\ 3 \\ 3 \\ 2 \end{array}\left\{\begin{array}{l} \text{Zeilen-} \\ \text{sum-} \\ \text{men=} \\ \text{Ecken-} \\ \text{grade} \end{array}\right\}$$

Die Gestalt der Matrizen hängt noch von der Numerierung von Ecken (und Kanten) ab.

Satz 4.4 *Es sei $G(P,L)$ ein Graph mit $P=\{u,\ldots,u_n\}$, $|L|=m$, A die Adjazenz- und B die Inzidenzmatrix. Dann gilt:*

$$\begin{aligned} &a) && A=A^{\mathsf{T}},\quad a_{ii}=0\ \forall i=1,\ldots,n. \\ &b) && \sum_{j=1}^n a_{ij}=\sum_{j=1}^m b_{ij}=g(u_i)\,\forall i,\quad \sum_{i=1}^n b_{ij}=2\,\forall j \\ & && \sum_{i,j=1}^n a_{ij}=\sum_{i=1}^n\sum_{j=1}^m b_{ij}=2m=2|L|. \\ &c) && A=BB^{\mathsf{T}}-diag(g(u_1),\ldots,g(u_n)). \end{aligned}$$

d) Es sei $A^k=:(a_{ij}^{(k)})_{i,j}$, $k\in\mathbb{N}$. Dann ist $a_{ij}^{(k)}$ die Anzahl der u_i und u_j verbindenden Kantenzüge $(u_{i_0},\ldots,u_{i_k})$ der Länge k, mit $u_{i_0}=u_i$, $u_{i_k}=u_j$, $u_{i_{l-1}}u_{i_l}\in L$, $l=1,\ldots,k$.

In der letzten Aussage können Kanten und Ecken mehrfach auftreten, z.B., gilt $a_{ii}^{(2)}=g(u_i)$.

Beweis Da Elemente von A und B in $\{0,1\}$ liegen, entsprechen Summationen Abzählungen.

a) Trivial: Die Symmetrie $a_{ij}=a_{ji}$ gilt bei ungerichteten Graphen, und da Schlingen verboten sind, ist $a_{ii}=0$.

b) Für die Summen gilt

$$\begin{aligned} &\sum_{j=1}^n a_{ij}=|\{u_j:\,u_iu_j\in L\}|=g(u_i), && \text{Anzahl Nachbarn,} \\ &\sum_{j=1}^m b_{ij}=|\{q\in L:\,u_i\in q\}|=g(u_i), && \text{Anzahl der Kanten mit } u_i, \\ &\sum_{i=1}^n b_{ij}=2, && \text{jede Kante hat 2 Endpunkte.} \end{aligned}$$

Die restlichen Aussagen entsprechen denen aus Satz 4.2.

c) Mit den Einheitsvektoren e_i bekommt man die Matrixelemente $a_{ij}=e_i^{\mathsf{T}}Ae_j$ im Fall

$$\begin{aligned} i\neq j:\quad e_i^{\mathsf{T}}BB^{\mathsf{T}}e_j &=\sum_{l=1}^m b_{il}b_{jl}=|\{l:\,b_{il}\neq 0\wedge b_{jl}\neq 0\}|=|\{q\in L:\,u_i\in q\wedge u_j\in q\}|=a_{ij}, \\ i=j:\quad e_i^{\mathsf{T}}BB^{\mathsf{T}}e_i &=\sum_{l=1}^m b_{il}^2=\sum_{l=1}^m b_{il}=g(u_i),\quad \text{vgl. b).} \end{aligned}$$

d) Induktiv, für $k = 1$ ist natürlich $a_{ij} = a_{ij}^{(k)}$. Schritt $k \to k+1$:

$$a_{ij}^{(k+1)} = e_i^\mathsf{T} A^{k+1} e_j = e_i^\mathsf{T} A \cdot A^k e_j = \sum_{l=1}^{n} a_{il} a_{lj}^{(k)} = \sum_{a_{il} \neq 0} a_{lj}^{(k)} = \sum_{u_l \in N(u_i)} a_{lj}^{(k)}. \tag{1}$$

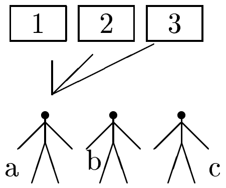
Die Bedingung $a_{il} \neq 0$ trifft genau für die *Nachbarn* $u_l \in N(u_i)$ zu. Daher summiert der letzte Ausdruck die Anzahl aller Kantenzüge der Länge k, die von diesen Nachbarn nach u_j gehen. Dies ist aber genau die Anzahl der Kantenzüge $u_i \leftrightarrow u_j$ mit Länge $k+1$. ■

Spezialfall **Bipartite Graphen**

Ein Graph $G(P, L)$ heißt *bipartit*, wenn gilt

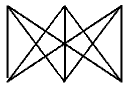
$$P = S + T \quad (\text{d.h. } S \cap T = \emptyset) \text{ und } \{uv \in L \Rightarrow u \in S,\ v \in T\}.$$

Alle Kanten laufen zwischen S und T, es gibt keine Kanten innerhalb von S oder T.



Solche Graphen treten bei Zuordnungsproblemen auf, z.B., mit Personen $S = \{a, b, c\}$ und Aufgaben T:

Für $|S| = n_1$, $|T| = n_2$ wird mit K_{n_1,n_2} der vollstängige bipartite Graph mit $|L| = n_1 n_2$ Kanten bezeichnet.

$K_{3,3}$ 

Kompakte Matrix-Darstellung bipartiter Graphen mit $S = (u_1, \ldots, u_{n_1})$, $T = (v_1, \ldots, v_{n_2})$:

$$D \in \mathbb{N}_0^{n_1 \times n_2}, \quad d_{ij} = \begin{cases} 1 & \text{wenn } u_i v_j \in L \\ 0 & \text{sonst} \end{cases}$$

Offensichtlich gilt (für $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$):

$$\sum_{j=1}^{n_2} d_{ij} = g(u_i), \quad \sum_{i=1}^{n_1} d_{ij} = g(v_j).$$

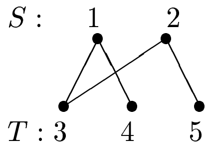
Durch Summation über alle dabei auftretenden Ecken führt dies auf

$$\sum_{u \in S} g(u) = \sum_{v \in T} g(v).$$

Hintergrund: Mit $P := S + T$, $n = n_1 + n_2$, hat die Adjazenzmatrix des Graphen die Struktur

$$A = \begin{pmatrix} 0 & D \\ D^\mathsf{T} & 0 \end{pmatrix}.$$

Beispiel:



$$A = \begin{pmatrix} . & . & 1 & 1 & 0 \\ . & . & 1 & 0 & 1 \\ 1 & 1 & . & . & . \\ 1 & 0 & . & . & . \\ 0 & 1 & . & . & . \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Satz 4.5 *Ein Graph G mit $n \geq 2$ Ecken ist genau dann bipartit, wenn alle Kreise gerade Länge haben (z.B., keine Kreise existieren).*

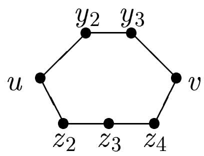
Beweis '$\Rightarrow$' Sei G bipartit mit $S = \{u_i\}$ und $T = \{v_j\}$ und $Z = (z_1, z_2, \dots, z_m)$ ein Kreis der Länge m mit $z_1 \in S$ (oBdA). Da G bipartit ist, folgt $z_2 \in T$, $z_3 \in S$ und induktiv $z_{2k-1} \in S$, $z_{2k} \in T$, $k \geq 1$. Daher ist m gerade.

'$\Leftarrow$' Es sei $u \in P$ mit $g(u) > 0$ fest gewählt. Da alle Kreise gerade Länge besitzen, hat jeder Kreis $Z = (u, z_2, \dots, z_{2k})$ gerade Länge $2k$, es sei $z_1 = u$. Die beiden Mengen T und S werden gebildet durch die Einteilung

$$z_i \in \begin{cases} S, & i \text{ ungerade}, \\ T, & i \text{ gerade}. \end{cases}$$

Es ist nun zu zeigen, dass S und T so wohldefiniert sind, d.h., dass diese Einteilung unabhängig vom gewählten Kreis ist (Eigenschaft $d(u,v)$ gerade ist Äquivalenzrelation). Gegenannahme: Für $v \in P$ gebe es zwei Kreise

$$(z_1 = u, z_2, \dots),\ (y_1 = u, y_2, \dots) \text{ mit } z_{2k-1} = v = y_{2l}.$$

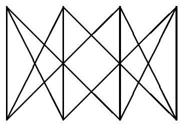


Dann könte aber aus beiden ein neuer Kreis $(z_1, z_2, \dots, z_{2k-1}, y_{2l-1}, \dots, y_2)$ mit *ungerader* Länge $2k - 1 + 2l - 2 = 2(k+l) - 3$ gebildet werden. **W**!

Die Einteilung betrifft nur Punkte, die von u aus 'erreichbar' sind. Das Verfahren kann aber ohne Widerspruch (s.o) mit anderen Startpunkten wiederholt werden. ■

Im Beweis ergab sich sogar, daß alle Wege zwischen Ecken von S (oder T) gerade Länge haben.

Beispiel: Der Hyperwürfel Q_n ist bipartit. Der Nullpunkt $u = (0, \dots, 0) \in \mathbb{Z}_2^n$ hat gerades Hamming-Gewicht, alle Nachbarn aber ein ungerades Code-Gewicht. Einteilung: $S = \{a \in \mathbb{Z}_2^n : g_H(a) \text{ gerade}\}$, $T = \{a \in \mathbb{Z}_2^n : g_H(a) \text{ ungerade}\}$.

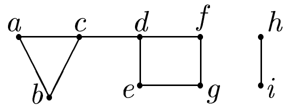


Q_3

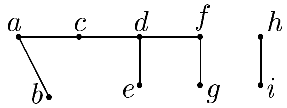
Begriffe II:

Brücke	eine Kante heißt Brücke, wenn ihre Streichung die Zahl der Komponenten erhöht.
Wald	zyklenfreier Graph
Baum	zusammenhängender Wald
Gewichteter	Graph, jede Kante $q \in L$ hat ein Gewicht $w(q)$, d.h., es ist eine Abbildung $w : L \to \mathbb{R}$ gegeben.

Beispiel:



• Die Kanten cd und hi sind Brücken.



• Durch Entfernung von bc und eg wird G zyklenfrei, also ein Wald mit den beiden Bäumen $\{a, b, .., g\}$, $\{h, i\}$.

• Gewicht: In vielen Anwendungen kann man den Kanten Werte zuordnen, etwa Weglängen, Kosten, Kapazitäten, etc. In einer Zeichnung schreibt man die Gewichte $w(k)$ an die Kanten. Das Gewicht eines Weges ist die Summe der Gewichte der auftretenden Kanten. Der oben definierte Abstand zweier Punkte stimmt dann mit dem Gewicht des kürzesten Wegs zwischen ihnen überein, wenn alle Kanten Gewicht 1 besitzen.

5 Bäume in Graphen

Bei vielen (Optimierungs-) Aufgaben in Graphen geht es darum, auf möglichst sparsame Weise (mit möglichst wenigen Kanten) alle Ecken zu erreichen. Der erzeugte Suchgraph in G enthält dann keine Kreise, ist also ein Baum.

Defin. 5.1 *Der Graph $G(P, L)$ sei zusammenhängend. Ein Untergraph B mit Ordnung $|P|$, der ein Baum ist, heißt* aufspannender Baum *von G.*

Aufspannende Bäume zu einem Graphen kann man konstruieren, indem man iterativ aus allen Kreisen eine Kante streicht ($\Rightarrow$ Existenz). Für Bäume gelten einige einfache Zusammenhänge.

Satz 5.2 *Sei $G(P, L)$ ein Graph. Dann sind äquivalent:*
a) G ist ein Baum.
b) Je zwei Ecken verbindet genau ein Weg.
c) G ist zusammenhängend und jede Kante ist eine Brücke.
d) G ist zusammenhängend und $|L| = |P| - 1$.

Beweis *a)* $\iff$ *b)*: Gäbe es zwei Wege, die Ecken u,v verbinden, ließe sich aus beiden ein Kreis konstruieren. Und auf einem Kreis gibt es immer zwei Wege zwischen verschiedenen Ecken.

b) $\iff$ *c)*: Sei $uv \in L$. Nach Teil b) ist diese Kante die einzige Verbindung zwischen u und v, durch Streichen von uv würde G zerfallen. Wäre umgekehrt uv keine Brücke, gäbe es einen weiteren Weg $u \leftrightarrow v$, der mit uv zu einem Kreis würde.

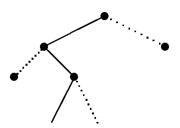
a) $\iff$ *d)*: Sei $W = (u_1, u_2, \ldots, u_k)$, $k \leq n := |P|$, ein längster Weg im Baum G. Dann sind auch alle Nachbarn von u_1 in W, da der Weg sonst verlängert werden könnte. Da G keine Kreise enthält, gilt $u_1 u_i \in L$ nur für $i = 2$, daher ist $N(u_1) = \{u_2\}$ und $g(u_1) = 1$. Durch Entfernen von u_1 und $u_1 u_2$ ist der Restgraph $G_2(P_2, L_2)$ mit $P_2 = P \setminus \{u_1\}$, $L_2 = L \setminus \{u_1 u_2\}$, daher auch NICHT-zerfallend und ist wieder ein Baum (keine Kreise) mit $|P_2| - |L_2| = |P| - |L|$. Da die Methode $n - 1$ mal angewendet werden kann und P_n dann nur noch eine Ecke enthält, gilt $|P| - |L| = |P_n| - |L_n| = 1 - 0$. ■

Bäume sind somit die sparsamste Datenstruktur zur Verbindung aller Ecken P eines Graphen. Da sich Methoden zur Durchquerung und Erzeugung von Bäumen stark ähneln, werden die Traversen zuerst behandelt. Man startet an einer Anfangsecke $u_0 \in P$ ('Wurzel'), die aber im Graph in der Regel keine besondere Bedeutung hat.

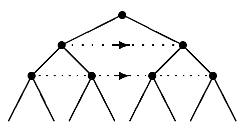
5.1 Baum-Traversen

Zur erschöpfenden Bearbeitung von Bäumen kommen zwei alternative Strategien in Frage.

- *Tiefen-Suche:* In der Startecke u_0 wird eine der Möglichkeiten ausgesucht, in der nächsten eine der Folgemöglichkeiten weiterverfolgt. In einer Sackgasse kehrt man zur jüngsten, noch nicht betrachteten Alternative zurück ('backtrack').
  Vorteile: Aufeinander folgende Ecken sind miteinander verbunden, weisen (je nach Problem) evtl. Ähnlichkeiten auf, die eine billige Anpassung der Eckenbeschreibung ermöglicht.
  Nachteile: Ungünstige Äste werden evtl. sehr weit untersucht.

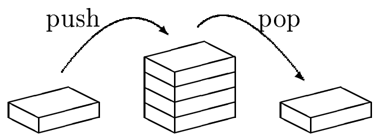


- *Breiten-Suche:* Alle Alternativen in einem Knoten werden gleichrangig behandelt, die Traverse steigt durch die 'Baumebenen' ab.
  Vorteile: Günstige Alternativen können bevorzugt werden.
  Nachteile: Alle die Ecke beschreibenden Daten sind zu speichern, da aufeinanderfolgende Ecken nicht verbunden sind. Da i.w. alle Knoten der aktuellen Baumebene gespeichert werden, ist auch der Speicherbedarf für die Ablaufverwaltung evtl. sehr hoch.



Die **Abwicklung** dieser beiden dualen Strategien erfordert die dynamische Verwaltung von Aufgaben-Listen. In beiden Fällen ist dies aber mit sehr einfachen Datenstrukturen möglich.

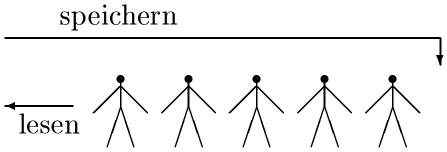
- *Tiefen-Suche: Keller* (Stapel, stack, FILO). Diese Verwaltung wird vom Rechner bei rekursiver Programmierung automatisch durchgeführt (Unterprogramm-Stapel).
  Speicheroperationen:
  push (speichern) und *pop* (lesen), gelesen wird jeweils das jüngste gespeicherte Element, das dabei im Keller gelöscht wird.



  Für Traversen wird jeweils die aktuelle Ecke u und der zuletzt bearbeitete Nachbar k gespeichert. Im folgenden Algorithmus sind alle Ecken oberhalb der aktuellen im Keller.
  Vollständige Tiefensuche, Start in Ecke u_0:

```
u := nil; k := u0;                                         {Wurzel}
repeat
   if k ≠ nil then
   begin bearbeite(k); push(u,k); u := k; k := nil end     {weiter nach unten}
   else pop(u,k);                                          {zurück nach oben}
   if k = nil then k :=ersternachfolger(u)
      else k :=nächsternachbar(k)                          {nächst.Nachfolger}
until kellerleer and (k = nil);
```

- *Breiten-Suche: Schlange* (queue, FIFO).
  Speicheroperationen:
  speichern (ans Ende der Schlange)
  lesen (vom Kopf), gelesen wird das am längsten in der Schlange befindliche Element und dann gelöscht.

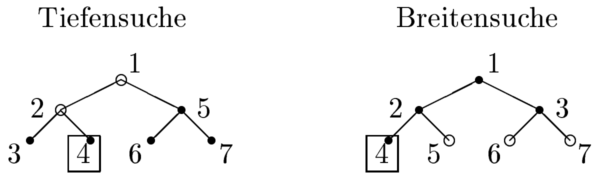


  Vollständige Breitensuche, Start in Ecke u_0:

  ```
  speichre(u0);
  repeat lies(u); bearbeite(u);
     forall v :=nachfolger(u) do speichre(v)
  until schlangeleer;
  ```

  In der Schlange befinden sich jeweils die restlichen Ecken der aktuellen Baumebene und ein Teil der Ecken aus der nächsten.

Die Nummern an den Knoten des folgenden Baums beschreiben den unterschiedlichen Ablauf. Bei Bearbeitung der Knoten Nr. 4 befinden sich die mit offenen Kreisen markierten im Speicher.



Eine direkte Übertragung dieser Traversen auf allgemeine Graphen ist möglich durch Einführung einer **Eckenmarkierung**

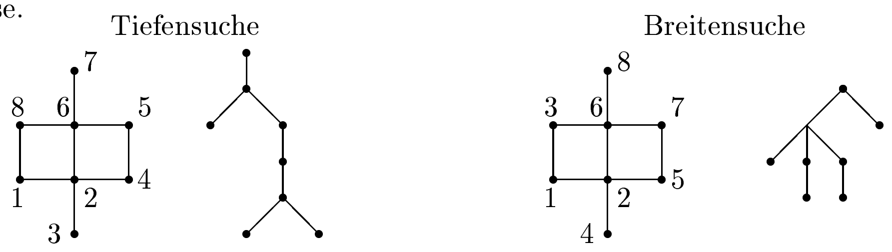
$$m : P \to M, \quad u \mapsto m(u),$$

z.B., mit $M = \{0, 1\}$ (ja/nein), $M = \mathbb{N}_0$ (Nummer), $M = \mathbb{R}$ (Kosten). In den obigen Algorithmen wird mit der Markierung besuchter Ecken *dynamisch* eine Baumstruktur B erzeugt durch Einschränkung der möglichen Nachfolger auf unmarkierte Nachbarn ($m = nil$ heißt unmarkiert):

$$N_B(u) := N(u) \cap \{v \in P : \; m(v) = nil\}.$$

Die Gestalt des Baums hängt dann allerdings vom Ablauf ab, ist also insbesondere unterschiedlich bei Tiefen- und Breitensuche.

Beispiel: Die gemeinsame Startecke ist 1, die Eckenmarkierung bezeichnet die Reihenfolge bei der Traverse.



Beide Traversier-Algorithmen erzeugen bei Einschränkung auf N_B offensichtlich dann einen aufspannenden Baum, wenn G zusammenhängend ist (Testverfahren!).

5.2 Kürzeste Wege

Gegeben sei ein zusammenhängender Graph $G(P, L)$ mit Gewichtsfunktion $w : L \to \mathbb{R}_+$ ('Entfernung' $w \geq 0$) und eine Startecke $u_0 \in P$ (und evtl. Zielecke $z \in P$).
Gesucht ist ein Weg $S = (u_0, u_1, \ldots, z)$ mit minimalem Gesamtgewicht

$$d_w(u_0, z) := w(S) = \min\{w(T) : T \text{ ist Weg } u_0 \leftrightarrow z\},$$

wobei die Weglänge definiert ist durch

$$w(T) := \sum_{j=2}^{l} w(v_{j-1}v_j) \quad \text{für den Weg } T = (v_1, v_2, \ldots, v_l).$$

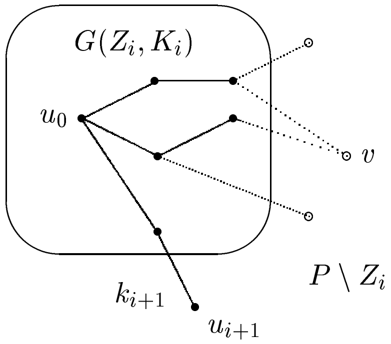
In dem folgenden Algorithmus wird dazu sogar das erweiterte Problem gelöst, optimale Teilwege $u_0 \leftrightarrow u$ zu allen Punkten $u \in P$ zu finden. Daher fällt dieses Verfahren in die folgende Kategorie.

Bezeichnung: Ein Algorithmus, der in jedem Schritt die aktuell optimale Maßnahme durchführt, wird *gieriger* Algorithmus ('greedy method') genannt.

Prinzip (**Dijkstra-Algorithmus**)
Beginnend mit $Z_0 = \{u_0\}$, $K_0 = \emptyset$, wird schrittweise ein Zentrumsbaum $G(Z, K)$ der zu u_0 nächstgelegenen Ecken und der dahin führenden Kanten aufgebaut. Im i-ten Schritt sei der Teilbaum $Z(G_i, K_i)$ mit

$$Z_i = \{u_0, u_1, \ldots u_i\}, \quad K_i = \{k_1, \ldots, k_i\},$$

$|Z_i| = |K_i| + 1$, vorhanden. Dieser Baum wird um die zu u_0 nächstgelegene Ecke $u \notin Z_i$ und die aus $Z(G_i, K_i)$ dorthin führende Kante erweitert, d.h., um $u_{i+1} \in P \setminus Z_i$ mit



$$d_w(u_0, u_{i+1}) = \min\{d_w(u_0, v) : \ v \in P \setminus Z_i\}.$$

Durchführung: In den schon gefundenen Ecken von $G(Z_i, K_i)$ wird als *Markierung*

$$m(u_j) := d_w(u_0, u_j), \quad u_j \in Z_i,\ j = 1, \ldots, i,$$

der (kürzeste) Abstand zu u_0 vermerkt. Für alle Nachbarn von Z_i,

$$N(Z_i) := \{N(u_j) \setminus Z_i : \ j = 0, \ldots, i\},$$

kann dann die kürzeste Entfernung zu u_0 durch Betrachtung der aus Z_i herausführenden Kanten u_jv berechnet werden, für $v \in N(Z_i)$ gilt

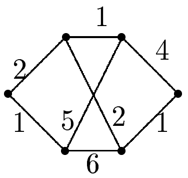
$$d_w(u_0, v) = \min\{m(u_j) + w(u_jv) : \ u_j \in N(v) \cap Z_i\}.$$

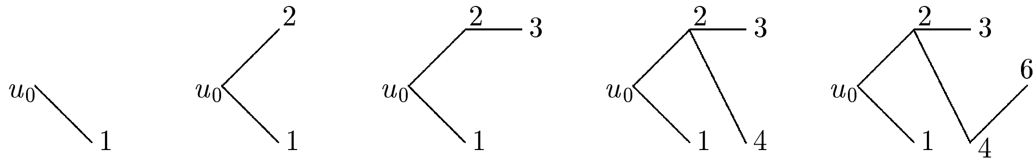
Algorithmus **D**: Aufspannender Baum kürzester Wege (Dijkstra).

$Z := \{u_0\}$; $K := \emptyset$; $m(u_0) := 0$;
for $i := 1$ to $|P| - 1$ do
begin $minab := \infty$; $x := nil$; $q := nil$;
  forall $uv \in L$ do if $(u \in Z)$ and $(v \notin Z)$ then $\boxed{1}$
  begin $a := m(u) + w(uv)$;
    if $a < minab$ then begin $minab := a$; $x := v$; $q := uv$ end;
  end; $m(x) := minab$;
  $Z := Z \cup \{x\}$; $K := K \cup \{q\}$;
end;

Im markierten Schritt $\boxed{1}$ sollte die Suche auf die aus Z_i hinausführenden Kanten eingeschränkt werden. Dazu kann evtl. eine Liste der Randecken $N(P \setminus Z_i)$ von Z_i mitgeführt werden. (Datenstruktur?)

Beispiel: Graph mit Kantengewichten:
Schritte im Dijkstra-Algorithmus
mit Entfernungen $m(u_i)$:





Satz 5.3 *Bei einem zusammenhängenden Graphen $G(P, L)$ mit Gewicht $w : L \to \mathbb{R}_+$ erzeugt der Dijkstra-Algorithmus einen aufspannenden Baum, in dem alle Wege von der Startecke u_0 zu anderen Ecken in G Wege minimalen Gewichts sind. Der Aufwand des Algorithmus ist $O(|P|^2)$.*

Beweis a) Induktiv ist nachzuweisen, daß die Formel für die Gewichtsfunktion im Algorithmus korrekt ist, daß also gilt

$$d_w(u_0, u_i) = \min\{m(u_l) + w(u_l v) : \ u_l \in N(v),\ v \in N(Z_{i-1})\} = m(u_j) + w(u_j u_i).$$

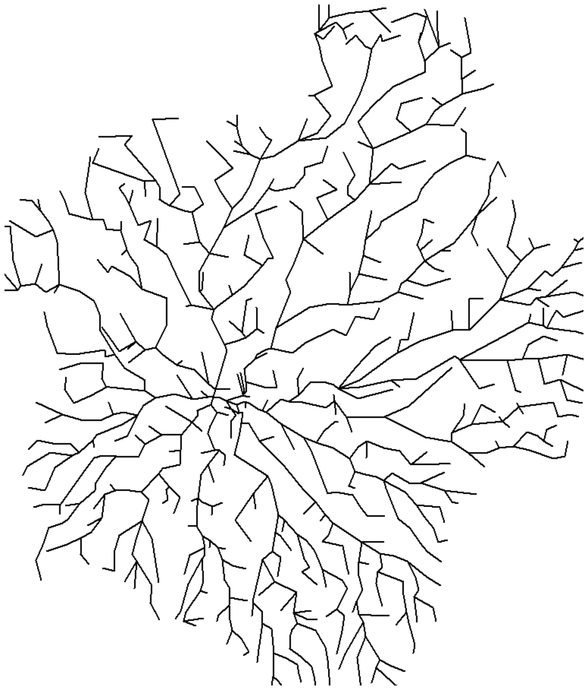
Sei dazu $W = (u_0, \ldots, y, v, \ldots, u_i)$ ein kürzester Weg mit $y \in Z_{i-1}$, $v \in P \setminus Z_{i-1}$. Dann gilt

$$d_w(u_0, u_i) = \underbrace{m(y) + w(yv)} + d_w(v, u_i) \overset{Def}{\geq} m(u_j) + w(u_j u_i) + 0 \geq d_w(u_0, u_i).$$

b) Im i-ten Schritt ist $|P \setminus Z_i| = n - i$, $n = |P|$, der Aufwand ist hier daher

$$(n - i)(\text{Additionen für } d_w) + (n - i)(\text{Min-Vergleiche}).$$

Durch Summation folgt ein Gesamtaufwand von $2(n^2/2) + \ldots \sim n^2$ Operationen. Außerdem sind im Schritt $\boxed{1}$ die aus Z_{i-1} hinausführenden Kanten mit einem Gesamtaufwand von $O(n^2)$ bestimmbar. ∎



Beispiel: Baum kürzester (Rad-) Wege der Marburger Umgebung, Start: Rudolphsplatz

5.3 Minimale aufspannende Bäume

Anwendung kostenminimale Versorgungsnetze: Eine bestimmte Anzahl von Teilnehmern soll durch mindestens eine Verbindung erreichbar sein (Wege, Fluglinien, Leitungen, Rechner-Vernetzung). Die Teilnehmer denkt man sich als Ecken eines Graphen, die möglichen Verbindungen als Kanten. Die Kantengewichte entsprechen den Kosten der Einzelverbindung.
Ein aufspannender Baum mit minimalem Gesamtgewicht ist durch ein *gieriges Verfahren*, den Algorithmus von Kruskal, berechenbar mit dem Aufwand $O(|L|^2)$. Dieser erzeugt einen Wald, in dem Teilbäume schrittweise durch Kanten minimalen Gewichts zusammengefaßt werden.

Prinzip: $L_0 := \emptyset$, der Wald $G_0 := G(P, L_0)$ hat $|P|$ Komponenten. Der Schritt i, $1 \le i < |P|$, bestimmt die neue Kante $k_i \in L \setminus L_{i-1}$ von $G(P, L_i)$ und $L_i = L_{i-1} \cup \{k_i\}$ gemäß

$$w(k_i) = \min\{w(\ell) : G(P, L_{i-1} \cup \{\ell\}) \text{ kreisfrei}\}.$$

Durchführung: Die beiden wesentlichen Schritte sind die Suche nach der jeweils minimalen Kante und die Entscheidung, ob die Hinzunahme dieser Kante zu einem Kreis führen würde.
Die wiederholte Minimumsuche ist trivial nach dem Sortieren einer Tabelle der Kanten nach fallendem Gewicht (Aufwand $O(|L| \log |L|)$), jede ausgewählte Kante wird in der Tabelle gelöscht. Weiterhin lassen sich Zyklen vermeiden, wenn die neue Kante Ecken aus *verschiedenen Bäumen* verbindet. Diese Frage kann nach Markierung der Ecken mit einer Baum-Nummer leicht überprüft werden, zu Beginn ist $m(u_j) = j$, $j = 1, \dots, |P|$, in G_0. Nach Auswahl der neuen Kante $k_i = uv$ werden die Baumnummern angeglichen, z.B., alle auf $m(u)$ gesetzt durch

$$\text{setze } m(e) := m(u) \text{ für alle } e \in P \text{ mit } m(e) = m(v).$$

Zur Konkretisierung des Verfahrens sei $|P| = n$, die Kanten seien zu Beginn als Feld $L[1..?]$ gespeichert, mit den Verweisen $L[k].u$, $L[k].v$ auf die Endpunkte der Kante $L[k]$.

Algorithmus **K***:* Minimaler aufspannender Baum (Kruskal)

$r := \lvert L\rvert$; $k := r$; for $i := 1..n$ do $m(u_i) := i$;	{Init.: n triviale Bäume}
sortiere(L);	{sortieren, Gewicht fallend}
repeat	
$\quad q := m(L[k].u)$; $p := m(L[k].v)$;	{kürzeste Kante}
$\quad$ for $i := 1..n$ do if $m(u_i) = p$ then $m(u_i) := q$;	{Bäume zusammenfassen}
$\quad L[k..r-1] := L[k+1..r]$; $r := r-1$; $k := r$;	{Kante löschen}
$\quad$ while $(k > 0)$ and $(m(L[k].u) \neq m(L[k].v))$ do $k := k-1$;	{nächste Brücke}
until $k = 0$;	

Beispiel mit $|P| = 500$, $|L| \cong 4000$ vgl. §6.

6 Euler- und Hamilton-Touren

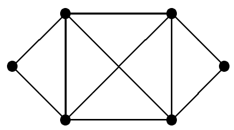
Zwei Arten von vollständigen Rundreisen durch Graphen haben in der Graphentheorie eine besondere Bedeutung. Sie werden in ungewichteten Graphen als Existenzfrage, in gewichteten Graphen als Optimierungsprobleme behandelt.

	ungewichtet, Existenz?	gewichtet, Optimierungsaufgabe
$\boxed{\text{E}}$	Ein *Eulerzug* ist eine zusammenhängende Kantenfolge $(k_1, \ldots, k_q)$, $q = \lvert L\rvert$, die jede Kante genau einmal enthält und zur Startecke zurückkehrt.	Briefträgerproblem (Kanten=Straßen): Eulerzug minimalen Gewichts.
$\boxed{\text{H}}$	Ein *Hamiltonkreis* ist ein Kreis (der Länge $n = \lvert P\rvert$), der jede Ecke genau einmal enthält.	Problem des Handlungsreisenden (TSP= *traveling salesman problem*): Hamiltonkreis minimalen Gewichts.

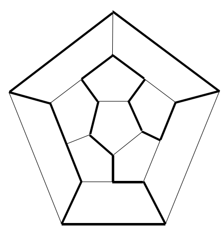
Beim Eulerproblem betrachtet man meist Multigraphen (Mehrfachkanten, Schlingen). Die Graphen heißen *eulersch* bzw. *hamiltonsch*, wenn sie einen Eulerzug bzw. Hamiltonkreis enthalten.

Beispiel:

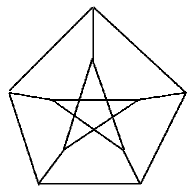
Eulersch



Hamiltonsch



Petersen-Graph:
nicht hamiltonsch



Beide Fragestellungen können in zwei Formen auftreten, als die der

- *Existenz:* Hat der Graph die Eigenschaft $\boxed{\text{E}}$ bzw. $\boxed{\text{H}}$?
- *Konstruktion* einer Tour minimalen Gewichts in gewichteten Graphen. Bei diesen Optimierungsproblemen wird in die Frageestellung oft eine geeignete Erweiterung des Graphen eingeschlossen ($\boxed{\text{E}}$), damit überhaupt Rundreisen existieren.

Die Existenz kann bei $\boxed{\text{E}}$ einfach charakterisiert werden, bei $\boxed{\text{H}}$ ist diese Frage schwer und nur im Einzelfall zu klären.

Satz 6.1 *Ein zusammenhängender Multigraph $G(P, L)$ mit $L \neq \emptyset$ ist genau dann* eulersch, *wenn es keine Ecken ungeraden Grades gibt.*

Konstruktion einer Euler-Tour: vgl. Literatur (Aigner).

Die Lösung zum Problem des Handlungsreisenden (TSP) ist sehr schwer, es können allerdings Näherungslösungen mit Hilfe von besprochenen Verfahren konstruiert werden. Daher wird jetzt das **TSP** (traveling-salesman-problem) behandelt in der folgenden Form. Es sei $P = (u_1, \ldots, u_n)$, $|P| = n$ und $G(P, L) = K_n$ vollständig. Die Gewichtsfunktion w entspricht dann einer symmetrischen Matrix mit Einträgen

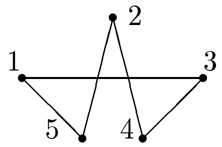
$$w_{ij} = w_{ji} = w(u_i, u_j) \geq 0.$$

Durch Anordnung nach dem ersten Index der Kantengewichte im Gesamtgewicht $w(H) = \sum_{l=1}^{n} w_{i_l i_{l+1}}$, $i_{n+1} := i_1$, eines Hamiltonkreises $(u_{i_1}, \ldots, u_{i_n})$ reduziert sich das TSP auf folgendes kombinatorische Minimierungsproblem

$$w(H) = w(\pi) := \sum_{i=1}^{n} w_{i,\pi(i)} \stackrel{!}{=} \min, \quad \pi \text{ zyklisch.} \tag{1}$$

Zugelassen sind hier nur zyklische Permutationen, welche aus genau einem Zyklus bestehen.

Beispiel: $w(H) = w_{13} + w_{34} + w_{42} + w_{25} + w_{51}$
$= w_{13} + w_{25} + w_{34} + w_{42} + w_{51}$.
mit $\pi = \binom{1\,2\,3\,4\,5}{3\,5\,4\,2\,1}$.



Nach Satz 1.12 gibt es $s_{n,1} = (n-1)!$ zyklische Permutationen von $\{1, \ldots, n\}$. Eine erschöpfende Untersuchung aller Möglichkeiten ist daher für größere n nicht durchführbar (vgl. §2.4 mit Stirlingformel Satz 2.10). Es ist auch kein Algorithmus bekannt, der das (strenge) Minimierungsproblem wesentlich schneller löst.

Daher behilft man sich mit *Heuristiken* zur Konstruktion 'guter' Rundreisen. Diese beziehen sich oft auf das *metrische* TSP, bei dem die Kostenmatrix *metrisch* ist, d.h., die Dreieckungleichung für die Gewichte gilt,

$$w_{ij} \leq w_{il} + w_{lj} \quad \forall\, 1 \leq i, j, l \leq n.$$

Da die kürzeste Einwegverbindung aller Ecken der minimale aufspannende Baum B aus §5.3 ist, kann kein Hamiltonkreis H kürzer sein: $w(H) \geq w(B)$. Interpretiert man jede Kante des Baums als Doppelkante, ist darin ein Eulerzug mit Länge $2w(B)$ enthalten. Daraus kann man durch 'Abkürzen' einen Hamiltonkreis $\tilde{H}$ machen, indem man über schon besuchte Ecken hinwegspringt. Wegen der Dreieckungleichung ist er nicht länger als der Eulerzug: $w(B) \leq w(\tilde{H}) \leq 2w(B)$.

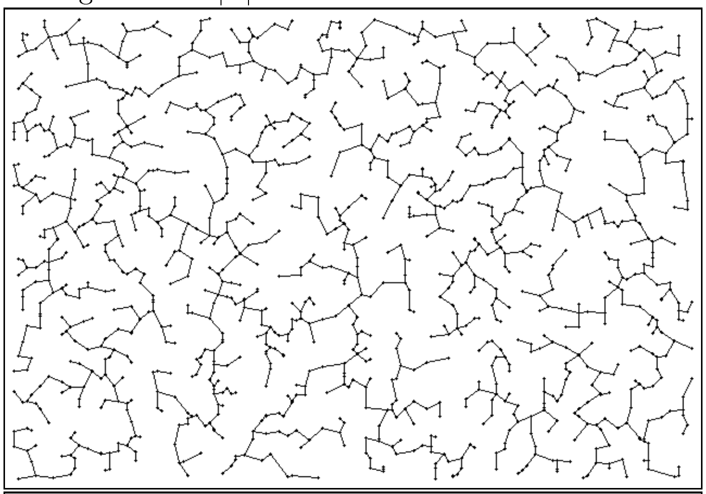
Durchführung der MST-Heuristik (minimal spanning tree):

- Bestimme minimalen aufspannenden Baum mit Algorithmus **K** (§5.3)
- Wähle Wurzel u_0 und führe Baumtraverse mit Tiefensuche durch (§5.1). Markiere besuchte Ecken und nimm Kante zwischen den beiden zuletzt markierten Ecken in den Kreis $\tilde{H}$ auf. Rücksprung zu u_0.

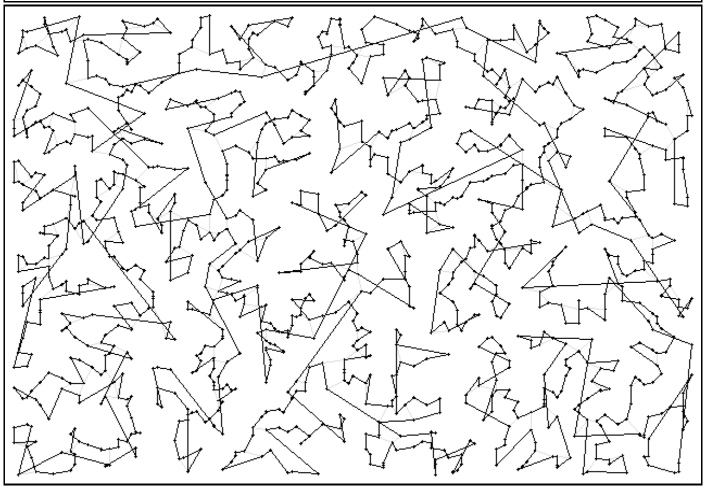
Beispiel: zum Ablauf, links der aufspannende Baum, rechts die daraus abgeleitete Tour.

Da auch für den optimalen Kreis $\hat{H}$ gilt $w(B) \leq w(\hat{H})$, wird dessen Länge durch $w(\tilde{H}) \leq 2w(\hat{H})$ um nicht mehr als 100% überschritten. Andere Heuristiken (Christofides) können den Verlust auf 50% reduzieren.

Praktisches Beispiel: $|P| = 1200$ Punkte sind zufällig in der Ebene verteilt, das Kantengewicht zwischen zwei Punkten entspricht dem Euklid-Abstand, allerdings werden nur Kanten bis zu einer Länge R eingeführt $\Rightarrow |L| = 10^5$.



Minimaler aufspannender Baum,



und Rundreise-Heuristik MST.

Die Komplexitätsklassen *P* und *NP*

Die Probleme mit Hamiltonkreisen sind Paradebeispiele für besonders schwere Aufgaben, die nur für kleine Problemgrößen n exakt gelöst werden können, vgl. §2.4. Zur Präzisierung gibt es eine Einteilung in Komplexitätsklassen für Entscheidungsprobleme (ja/nein-Ergebnis). Das *Hamiltonproblem* (HP), die Frage, ob ein Graph hamiltonsch ist, ist offensichtlich ein Entscheidungsproblem. Auch das TSP kann durch Einführung einer Schranke $M \in \mathbb{R}$ zu einem solchen gemacht werden, indem man nach der Existenz eines Hamiltonkreises H mit Länge $w(H) \leq M$ fragt. Entscheidend ist beim Studium der einsetzbaren Verfahren die Entwicklung des Aufwands in Abhängigkeit von einer Problemgröße n. Insbesondere interessiert man sich dafür, ob der Aufwand in annehmbarer Form (polynomiell) oder astronomisch (exponentiell) anwächst.

<u>Standardisierungen</u>:
Problemgröße n: Darunter versteht man die zur Formulierung des Problems erforderliche Bit-Anzahl.
Komplexitätsklasse $\boxed{\text{P}}$: Entscheidungsprobleme, für die ein Algorithmus mit *polynomieller* Laufzeit existiert, d.h., ein $r \in \mathbb{R}$ so, daß der Aufwand in Abhängigkeit von n wächst wie $O(n^r)$.

Beim Hamiltonproblem kann die Adjazenzmatrix bei $|P| = m$ Ecken mit $n = m(m-1)/2$ Bit gespeichert werden. Für die Durchmusterung aller $(m-1)!$ zyklischen Permutationen verhält sich der Aufwand wie die Funktion $(\sqrt{n})!$, die stärker als jedes Polynom anwächst, vgl. Satz 2.10. Andererseits kann für einen Weg der Länge n leicht geprüft werden, ob er ein Hamiltonkreis ist, eine mögliche Lösung kann also schnell verifiziert werden. Dies ist charakteristisch für die nächste Problemklasse.
Komplexitätsklasse $\boxed{\text{NP}}$ (Nichtdeterministisch polynomiell): Entscheidungsprobleme, für die man *keinen* polynomiellen Lösungs-Algorithmus kennt, für die eine Lösung aber in polynomieller Zeit verifizierbar ist.

Ein bisher ungelöstes Problem der Komplexitätstheorie ist die Frage, ob die beiden Klassen gleich oder verschieden sind, $\boxed{\text{P}}=\boxed{\text{NP}}$? Allerdings kann man für die Klasse NP besonders repräsentative Probleme identifizieren, die NP-vollständigen Probleme. Dies sind solche, aus deren Lösung in polynomieller Zeit die polynomielle Lösbarkeit aller NP-Probleme folgen würde. Die NP-Vollständigkeit des Hamilton-Problems wurde 1972 nachgewiesen. Da auch andere bekannt schwere Problem NP-vollständig sind, und für keines polynomielle Algorithmen bekannt sind, vermutet man, daß tatsächlich gilt $\boxed{\text{P}} \neq \boxed{\text{NP}}$.

Index