1 Abzählungen

Es gibt nur zwei Arten von Aufgaben:

- Anzahl Aufteilungen von einer Menge N von Kugeln in eine Menge R von Fächern
- Aus einer Menge N mit n Elementen sollen alle oder k Elemente ausgewählt werden

1.1 Anzahl Aufteilungen von einer Menge N von Kugeln in eine Menge R von Fächern

$\lvert N\rvert = n, \lvert R\rvert = r$	beliebig	injektiv	surjektiv	bijektiv
N unterscheidbar R unterscheidbar	r^n	$r \geq n : r^{\underline{n}}$ $r < n : 0$	$n \geq r : r! S_{n,r}$ $n < r : 0$	$r = n : n!$ $r \neq n : 0$
N nicht unterscheidbar R unterscheidbar	$\binom{r+n-1}{n} = \frac{r^{\overline{n}}}{n!}$	$r \geq n : \binom{r}{n}$ $r < n : 0$	$n \geq r : \binom{n-1}{r-1}$ $n < r : 0$	$r = n : 1$ $r \neq n : 0$
N unterscheidbar R nicht unterscheidbar	$\sum_{k=1}^{r} S_{n,k}$	$r \geq n : 1$ $r < n : 0$	$n \geq r : S_{n,r}$ $n < r : 0$	$r = n : 1$ $r \neq n : 0$
N nicht unterscheidbar R nicht unterscheidbar	$\sum_{k=1}^{r} P_{n,k}$	$r \geq n : 1$ $r < n : 0$	$n \geq r : P_{n,r}$ $n < r : 0$	$r = n : 1$ $r \neq n : 0$

Beliebige Aufteilung:

- Keine speziellen Regeln bei der Aufteilung (Man kann die Kugeln verteilen wie man möchte)

  $\Rightarrow$ Ein Fach darf leer sein, kann eine Kugel haben oder mehrere Kugeln haben
- Alle Kugeln müssen benutzt werden

Injektive Aufteilung:

- Jedes Fach darf höchstens eine Kugel enthalten (Kein Fach darf mehr als eine Kugel enthalten)

  $\Rightarrow$ Fächer dürfen leer bleiben (wenn Anzahl Fächer > Anzahl Kugeln)
- Nicht jede Kugel muss benutzt werden

Surjektive Aufteilung:

- Jedes Fach muss mindestens eine Kugel enthalten

  $\Rightarrow$ Kein Fach darf leer sein

  $\Rightarrow$ Einige Fächer können mehr Kugeln haben als andere (wenn Anzahl Fächer < Anzahl Kugeln)
- Nicht jede Kugel muss benutzt werden

Bijektive Aufteilung:

- Jedes Fach muss genau eine Kugel enthalten

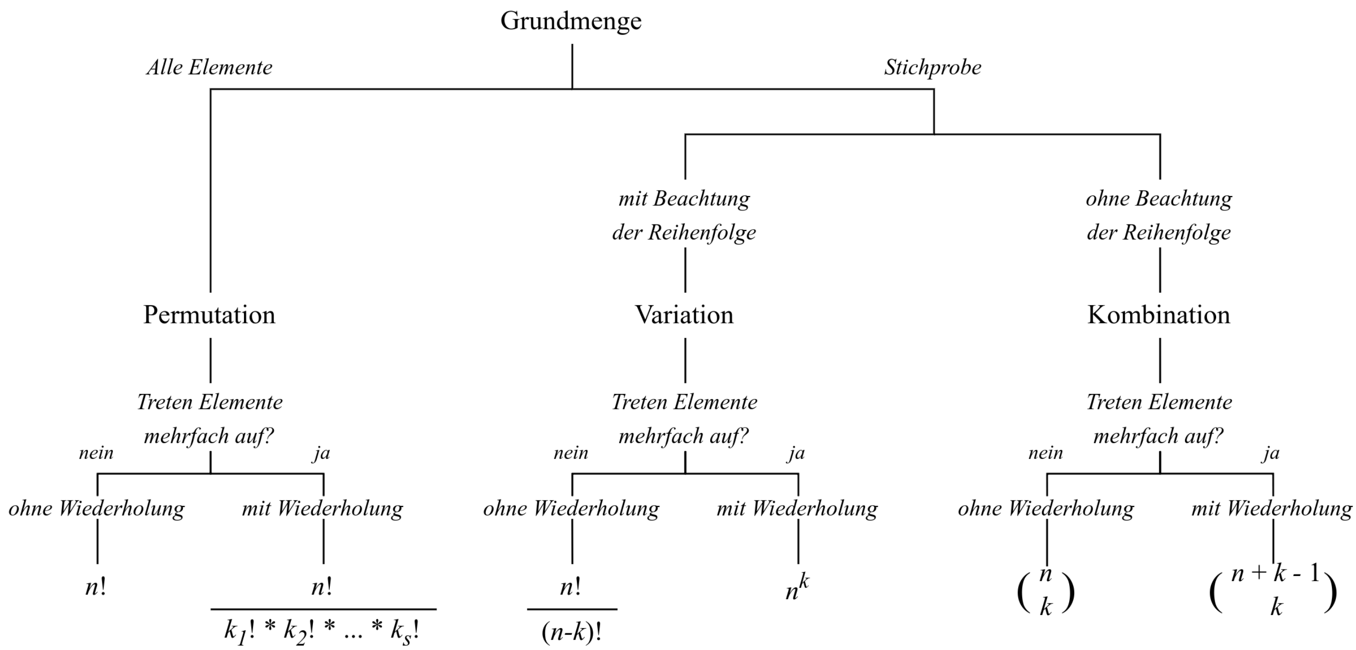
  $\Rightarrow$ Kein Fach darf leer sein

  $\Rightarrow$ Kein Fach darf mehre Kugeln enthalten

$\Rightarrow$ Anzahl Kugeln = Anzahl Fächer

- Jede Kugel muss somit benutzt werden

1.2 Aus einer Menge N mit n Elementen sollen alle oder k Elemente ausgewählt werden



- *Mit Beachtung der Reihenfolge:* ABC $\neq$ ACB $\neq$ BAC $\neq$ BCA $\neq$ CAB $\neq$ CBA
- *Ohne Beachtung der Reihenfolge:* ABC = ACB = BAC = BCA = CAB = CBA

Kombination:

Exakte Auswahl: Aus einer Menge mit n Elementen wählt man (*genau*) k aus.

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

Mindestauswahl (Allgemein): Wie viele Möglichkeiten gibt es *mindestens* k von n (z.B mindestens 7 von 10) richtig zu beantworten?

- Man rechet immer mit Plus (+)

$$\binom{10}{7} + \binom{10}{8} + \binom{10}{9} + \binom{10}{10} = 176$$

Mindestauswahl (mit getrennter Betrachtung): Wie viele Möglichkeiten gibt es, falls von den ersten 5 Fragen *genau* 3 und von den nächsten 5 *genau* 4 richtig beantwortet werden müssen?

- Man rechet immer mit Mal ($\cdot$)

$$\binom{5}{3} \cdot \binom{5}{4} = 50$$

Mindestauswahl (mit kombinierter Betrachtung): Wie viele Möglichkeiten gibt es, falls von den ersten 5 Fragen *mindestens* 3 richtig beantwortet und insgesamt *genau* 7 richtig beantwortet werden müssen?

- Man rechet immer mit Plus (+) und Mal (·)

$$\binom{5}{3} \cdot \binom{5}{4} + \binom{5}{4} \cdot \binom{5}{3} + \binom{5}{5} \cdot \binom{5}{2}$$

1.3 Rekursionsgleichungen

1. Gleichung umstellen:

- Rekursionsgleichung in einen homogenen (links) und einen inhomogenen Teil (rechts) teilen
- **Homogener Teil:** Überall wo ein a vorhanden ist
- **Inhomogener Teil:** Rest wo kein a vorhanden ist

$$homogener\ Teil = inhomogener\ Teil$$

2. Allgemeine Lösung des homogenen Teils berechnen:

- Das Charakteristisches Polynom aufstellen indem man alle a_n mit λ^n ersetzt

$$p(\lambda) = x_m\lambda^{n_m} + x_{m-1}\lambda^{n_{m-1}} + \ldots x_0 + \lambda^{n_0}$$

- Das Charakteristisches Polynom lösen, durch finden der Nullstellen
  - Bei Polynom zweiten Gerades (λ^2) pq-Formel verwenden: $-\frac{-p}{2} \pm \sqrt{(\frac{p}{2})^2 - q}$

$$x_m\lambda^{n_m} + x_{m-1}\lambda^{n_{m-1}} + \ldots x_0 + \lambda^{n_0} = 0$$

- Allgemeine Lösung für homogenen Teil setzt sich aus den Nullstellen $\lambda_1, \lambda_2, \ldots, \lambda_k$ zusammen
  - Falls die Nullstellen verschieden (und reelle Zahlen) sind ist die allgemeine Lösung des homogenen Teils:

$$a_{h,n} = c_1\lambda_1^n + c_2\lambda_2^n + \cdots + c_k\lambda_k^n$$

  - Falls die Nullstellen gleich (und relle Zahlen) sind ist die allgemeine Lösung des homogenen Teils:

$$a_{h,n} = c_1\lambda_1^n + c_2n\lambda_2^n + c_3n^2\lambda_3^n + \cdots + c_kn^{k-1}\lambda_k^n$$

3. Ansatz für Spezielle Lösung des inhomogenen Teils herleiten:

- Mit Hile der Tabelle lässt sich ein Ansatz für die spezielle Lösung des inhomogenen Teils herleiten (vermutete spezielle Lösung)
- Dieser wird mit $a_{s,n}$ bezeichnet

	Ansatzfunktion: $y_{s,n}$
$b\ (Konstante)$	$B\ (Konstante)$
n	$B_1n + B_0$
$n^t, t \in \mathbb{N}$	$B_tn^t + B_{n-1}n^{t-1} + \cdots + B_1n + B_0$
$r^n, r \in R$	Br^n
n^tr^n	$r^n(B_tn^t + B_{n-1}n^{t-1} + \cdots + B_1n + B_0)$

4. Ansatz der Spezielle Lösung in den homogenen Teil einsetzen, um die spezielle Lösung zu erhalten:

- Ansatz der Spezielle Lösung in den homogenen Teil einsetzen
- Koeffizientenverlgeich zum bestimmen der speziellen Lösung

5. a_n bestimmen

- Kombiniere die allgemeine Lösung des homogenen Teils und die spezielle Lösung des inhomogenen Teils:

$$a_n = a_{h,n} + a_{s,n}$$

6. Gegebenes a_0 und a_1 benuten

- Gegebenen Anfangswerte in die allgemeine Lösung eingesetzt, um die spezifischen Werte der Konstanten zu bestimmen, die benötigt werden, um die endgültige Lösung zu finden.

Spezialfälle:

1. Falls in der allgemeinen Lösung des homogenen Teils, es gleichen Stellen wie in dem Ansatz der speziellen Lösung der inhomogenen Lösung gibt, dann wird ein n im Ansatz der speziellen Lösung hinzugefügt

- **Beispiel:**

$$a_{h,n} = c_1 3^n + c_2(-2)^n$$

$$\textit{und}$$

$$a_{s,n} = b_1 3^n + b_2 n + b_3$$

$$\textit{Vor } 3^n \textit{ wird deshalb ein } n \textit{ hinzugefügt}$$

$$a_{s,n} = b_1 n 3^n + b_2 n + b_3$$

2. Falls es ein 1^n in der allgemeine Lösung des homogenen Teils gibt mit $n \geq 0$, dann wird ein n im speziellen Teil hinzugefügt.

- **Beispiel:**

$$a_{h,n} = c_1 3^n + c_2(-2)^n$$

$$\textit{und}$$

$$a_{s,n} = b_1 1^n + b_2 n + b_3$$

$$\textit{Vor } 3^n \textit{ wird deshalb ein } n \textit{ hinzugefügt}$$

$$a_{s,n} = b_1 n 3^n + b_2 n + b_3$$

Beispiel: $a_{n+2} = 49a_n + 48n - 98, \qquad n \geq 0 \qquad a_0 = 5, a_1 = 8$

1. Gleichung umstellen

$$a_{n+2} - 49a_n = 48n - 98$$

2. Allgemeine Lösung des homogenen Teil berechnen

- Ersetze alle a_n mit λ^n
- $a_{n+2} \to \lambda^2$ und $-49a_{n+0} \to -49\lambda^0 = -49$

$$p(\lambda) = \lambda^2 - 49$$

- Nullstelle bestimmen (λ_1 und λ_2)

$$\lambda^2 - 49 = 0 \to \lambda_{1,2} = \pm 7$$

- Allgemeine Lösung für homogenen Teil:

$$\boxed{a_{h,n} = c_1 7^n + c_2(-7)^n}$$

3. Ansatz für spezielle Lösung des inhomogenen Teil

- Nach Tabelle Ansatz für spezielle Lösung aufstellen

$$a_{s,n} = b_1 n + b_2$$

	Ansatzfunktion: $y_{s,n}$
$b\ (Konstante)$	$B\ (Konstante)$
n	$B_1 n + B_0$
$n^t, t \in \mathbb{N}$	$B_t n^t + B_{n-1} n^{t-1} + \dots + B_1 n + B_0$
$r^n, r \in R$	$B r^n$
$n^t r^n$	$r^n(B_t n^t + B_{n-1} n^{t-1} + \dots + B_1 n + B_0)$

4. Ansatz für spezielle Lösung in homogenen Teil einfügen

- homogener Teil: $a_{n+2} - 49a_n$
- Ansatz spezielle Lösung: $a_{s,n} = b_1 n + b_2$
- inhomogener Teil: $48n - 98$

$$b_1(n+2) + b_2 - 49(b_1 n + b_2) = 48n - 98$$
$$b_1 n + 2b_1 + b_2 - 49b_1 n - 49b_2 = 48n - 98$$

- Nun vergleicht man die Koeffizienten der speziellen Lösung (n^1, n^0)
- inhomogener Teil: $48n-98$

$$n^1 : b_1 - 49b_1 = -48b_1 = 48 \to b_1 = -1$$

$$n^0 : 2b_1 + b_2 - 49b_2 = -98 \to b_2 = 2$$

- Nun setzt man die ermittelten Werte in den Ansatz der speziellen Lösung ein und erhält die spezielle Lösung

$$\boxed{a_{s,n} = -1n + 2}$$

a_n bestimmen

$$\begin{aligned} a_n &= a_{h,n} + a_{s,n} \\ a_n &= c_1 7^n + c_2(-7)^n - 1n + 2 \end{aligned}$$

a_0 und a_1 benutzen um c_1 und c_2 zu bestimmen

- $a_0 = 5, a_1 = 8$

$$\begin{aligned} a_0 : c_1 7^0 + c_2(-7)^0 - 1 \cdot 0 + 2 &= 5 \\ c_1 - c_2 + 2 &= 5 \end{aligned}$$

$$\begin{aligned} a_1 : c_1 7^1 + c_2(-7)^1 - 1 \cdot 1 + 2 &= 8 \\ 7c_1 - 7c_2 - 1 + 2 &= 8 \end{aligned}$$

- a_0 nach c_1 umstellen un in a_1 einsetzen um c_2 zu bestimmen
- dann c_2 in c_1 einsetzen um c_1 zu erhalten
- Werte von c_1 und c_2 in a_n einsetzen, das Ergebnis ist die allgemeine Lösung

$$\begin{aligned} a_0 : c_1 - c_2 + 2 &= 5 \\ c_1 &= 3 + c_2 \end{aligned}$$

$$\begin{aligned} a_1 : 7(3 + c_2) - 7c_2 - 1 + 2 &= 8 \\ &\dots \end{aligned}$$

$$\boxed{a_n = c_1 7^n + c_2(-7)^n - 1n + 2}$$

Hier c_1 und c_2 mit eigentlichen Werten ersetzen

1.4 Typische Klausurfragen zu Rekursionsgleichungen:

1. Typ der Rekursionsgleichung bestimmen

- **Homogene lineare Rekursion:**
  - Besteht nur aus a's (kann auch ein anderer Buchstabe sein) aber es werden keine Koeffizienten vorgegeben
  - **Aufbau:**

$$a_{n+k} = y_{k-1}a_{n+k-1} + y_{k-2}a_{n+k-2} + \cdots + y_1 a_{n+1} + y_0 a_n \qquad n \geq 0$$

  - **Beispiel:**

$$a_{n+2} = 3a_{n+1} - 2a_n \qquad n \geq 0$$

- **Homogene lineare Rekursion mit konstanten Koeffizienten:**
  - Besteht nur aus a's (kann auch ein anderer Buchstabe sein)
  - **Aufbau:**

$$a_{n+k} = y_{k-1}a_{n+k-1} + y_{k-2}a_{n+k-2} + \cdots + y_1 a_{n+1} + y_0 a_n \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

  - **Beispiel:**

$$a_{n+2} = 3a_{n+1} - 2a_n \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

- **Inhomogene lineare Rekursionsgleichungen mit konstanten Koeffizienten:**
  - Hat zusätzliche inhomogene Komponente, entweder eine Funktion von n (n, n^2, n^3, ..., x^n, ...) oder eine konstante Zahl
  - **Aufbau:**

$$a_{n+k} = y_{k-1}a_{n+k-1} + y_{k-2}a_{n+k-2} + \cdots + y_1 a_{n+1} + y_0 a_n + f(n) \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

  - **Beispiel:**

$$a_{n+2} = 3a_{n+1} - 2a_n + 4n \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

oder

$$a_{n+2} = 3a_{n+1} - 2a_n + 6 \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

oder

$$a_{n+2} = 3a_{n+1} - 2a_n + 3^n \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

oder

$$a_{n+2} = 3a_{n+1} - 2a_n + 4n + 6 + 3^n \qquad n \geq 0 \quad a_0 = 1, a_1 = 4$$

2. Grad der Rekursionsgleichung bestimmen

- Der Grad einer Rekursionsgleichung bezieht sich auf den höchsten Exponenten der Rekursionsvariable
- **Beispiel:** $a_{n+2} = 3a_{n+1} - 2a_n + 6 \rightarrow \text{Grad} = 2$

2 Codierung

2.1 Allgemeines

- Linearer (n, m)-Code C
- ***a*** **x** ***b*** **Generatormatrix:** $n = b$ und $m = a$
- ***a*** **x** ***b*** **Kontrollmatrix:** $n = b$ und $m = a - b$

2.2 Wichtige Formeln

- **Blocklänge/Codewortlänge:** n
- **Linear unabhängige Wörter/Dimension des Unterrraum/Länge Wörters** C**:** m
- **Anzahl Codewörter:** $|C| = q^m$, wobei q Anzahl Elemente in C
- **Anzahl Wörter in Standardfeld:** q^n
- **Anzahl Wörter/Zeilen in Syndromtabelle:** q^{n-m}
- **Hamming Code:** $n = \dfrac{q^{n-m} - 1}{q - 1}$
- **Schätzen der Codedistanz (Singleton-Schranke):** $d(C) \leq n - m + 1$ (obere Schranke)
- Wenn Zeilen von Kontrollmatrix linear unabhängig ist Codedistanz immer $d(C) \geq 3$ (Konstruktionssatz) (untere Schranke)?
- **Fehlererkennend:** $(n - m)$
- **Fehlerkorrigierend:** $\left\lfloor \dfrac{(n - m)}{2} \right\rfloor$
- **t-fehlererkennend:** $d(C) \geq t + 1$, das ausgewählte t ist, wie viel Fehler der Code erkennt
- **t-fehlerkorrigierend:** $d(C) \geq 2t+1$, das ausgewählte t ist, wie viele Fehler der Code korrigiert
- **t-ausfällekorrigierend:** $d(C) \geq t + 1$

2.3 Allgemeines zu Matrizen

Aufbau Generatormatrix:

- $(m \times n)$ Generatormatrix
- Beispiel: $m = 2, n = 5$
  $\rightarrow$ (2×5) Generatormatrix

Aufbau Kontrollmatrix:

- $(n \times n - m)$ Kontrollmatrix
- Beispiel: $m = 2, n = 5$
  $\rightarrow$ $(5 \times 5 - 2) = (5 \times 3)$ Kontrollmatrix

$$G = \begin{array}{cc} & n \longrightarrow \\ m \downarrow & \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix} \end{array}$$

$$H = \begin{array}{cc} & n\text{-}m \longrightarrow \\ n \downarrow & \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \end{array}$$

wort · matrix:

$$1010010 \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot 1 & 1 \cdot 1 & 1 \cdot 1 \\ 1 \cdot 0 & 0 \cdot 0 & 1 \cdot 0 \\ 0 \cdot 1 & 1 \cdot 1 & 1 \cdot 1 \\ 1 \cdot 0 & 1 \cdot 0 & 0 \cdot 0 \\ 1 \cdot 0 & 0 \cdot 0 & 0 \cdot 0 \\ 0 \cdot 1 & 1 \cdot 1 & 0 \cdot 1 \\ 0 \cdot 0 & 0 \cdot 0 & 1 \cdot 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & | & 1 & | & 1 \\ +0 & | & +0 & | & +0 \\ +0 & | & +1 & | & +1 \\ +0 & | & +0 & | & +0 \\ +0 & | & +0 & | & +0 \\ +0 & | & +1 & | & +0 \\ +0 & \downarrow & +0 & \downarrow & +0 \end{pmatrix} = 110$$

$$1010010 \cdot \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}^T = \begin{pmatrix} 1 \cdot 1 & 1 \cdot 0 & 0 \cdot 1 & 1 \cdot 0 & 1 \cdot 0 & 0 \cdot 1 & 0 \cdot 0 \\ 1 \cdot 1 & 0 \cdot 0 & 1 \cdot 1 & 1 \cdot 0 & 0 \cdot 0 & 1 \cdot 1 & 0 \cdot 0 \\ 1 \cdot 1 & 1 \cdot 0 & 1 \cdot 1 & 0 \cdot 0 & 0 \cdot 0 & 0 \cdot 1 & 1 \cdot 0 \end{pmatrix}^T =$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T = \begin{pmatrix} 1 & +0 & +0 & +0 & +0 & +0 & +0 \\ - & - & - & - & - & - & \rightarrow \\ 1 & +0 & +1 & +0 & +0 & +1 & +0 \\ - & - & - & - & - & - & \rightarrow \\ 1 & +0 & +1 & +0 & +0 & +0 & +0 \end{pmatrix}^T = 110$$

2.4 Codewörter sind gegeben

n und m bestimmen

$$n = \text{Länge der Codewörter} \Rightarrow n = 5$$

00000
01101
10111
11010

$$m = \text{Dimension} \Rightarrow m = 2$$

00000
01101
10111
11010

Hamming-Distanz/Code-Distanz/$d(c)$

- Hemming-Distanz ist die minimale Änderung des Gewichts
- Wird mit $d(C)$ bezeichnet

Beispiel:

- Anzahl Einsen in Codewörter Zählen
- $0 \ldots 0$ wird dabei nicht beachtet

01101	$\rightarrow$	Gewicht: 3
10111	$\rightarrow$	Gewicht: 4
11010	$\rightarrow$	Gewicht: 3

- Gewicht der Codewörter vergleichen und minimalstes auswählen

$\Rightarrow$ $d(C) = 3$, da es minimal ist

Kanonische Generatormatrix

- Definition: Aus der Generatormatrix kann man alle möglichen Codewörter der Sprache erzeugen.
- Falls die Generatormatrix gegeben ist, lässt sich die die kanonische Generatormatrix durch das Anwenden des Gaußschen Verfahrens auf die Generatormatrix erstellen.
- Größe: $(m \times n)$
- Aufbau: $G = \begin{pmatrix} E & G' \end{pmatrix}$
  - E: Einheitsmatrix
  - G': Linear unabhängiger Rest aus Codewörtern

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

Kanonische Kontrollmatrix

- Definition: Erfüllt ein Wort $wort \cdot kontrollmatrix = 0$ ist es ein richtiges Codewort.
- Größe: $n \times (n - m)$
- Aufbau: $H = \begin{pmatrix} -G \\ E \end{pmatrix}$
  - E: Einheitsmatrix
  - $-G$: Linear unabhängiger Rest aus Codewörtern

$$H = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Modulo mit negativen Zahlen:

	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
$\mathbb{Z}_2$									0	1	[0	1]								
$\mathbb{Z}_3$								0	1	2	[0	1	2]							
$\mathbb{Z}_4$							0	1	2	3	[0	1	2	3]						
$\mathbb{Z}_5$						0	1	2	3	4	[0	1	2	3	4]					
$\mathbb{Z}_6$					0	1	2	3	4	5	[0	1	2	3	4	5]				
$\mathbb{Z}_7$				0	1	2	3	4	5	6	[0	1	2	3	4	5	6]			
$\mathbb{Z}_8$			0	1	2	3	4	5	6	7	[0	1	2	3	4	5	6	7]		
$\mathbb{Z}_9$		0	1	2	3	4	5	6	7	8	[0	1	2	3	4	5	6	7	8]	
$\mathbb{Z}_{10}$	0	1	2	3	4	5	6	7	8	9	[0	1	2	3	4	5	6	7	8	9]

2.5 Syndromtabelle

- Definition: Identifiziert und korrigiert Fehler in Codewörter durch Vergleich mit erwarteten Werten.
- Anzahl Zeilen: q^{n-m}
- Anzahl Spalten: n

Beispiel: (7×3)-Kontrollmatrix, $n = 7, m = 4$

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

a	$S = a \cdot H$
0000000	000
1000000	111
0100000	101
0010000	011
0001000	110
0000100	100
0000010	010
0000001	001

- Anzahl Zeilen: $2^{7-4} = 8$
- Anzahl Spalten: 7
- Wenn über 0000001 hinaus geht egal ob 1000001, 1100000, ...

1. Überprüfen ob es ein empfangenes Codewort fehlerfrei ist($wort \cdot kontrollmatrix = 0$)
   - Empfangenes Wort: $y = 1010010$

$$1010010 \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = 110 \neq 0 \Rightarrow \text{Fehler im empfangenen Codewort}$$

2. Codewort durch Syndromtabelle korrigieren
   - Klassenanführer von 110 aus Syndromtabelle ablesen: $a = 0001000$
   - $empfangenes\ Codewort - Klassenanfuehrer = korrigiertes\ Codewort$

$$\begin{array}{ccc} y & & 1010010 \\ a & - & 0001000 \\ \hline & & 1011010 \end{array}$$

   - korrigiertes Codewort: 1011010

3. Nachricht extrahieren
   - Letzte Stellen des korrigierten Codewort entfernen, um die Nachricht zu erhalten (Anzahl entfernte Stellen entspricht Länge des Syndrom).
   - Nachricht: $1011\cancel{010} = 1011$

2.6 Reed-Solomon-Codes

- $RS(q, m, n)$
- **Hamming-/Code-Distanz:** $d(C) = n - m + 1$
- **Max. Anzahl Fehlerkorrigierend:** $\left\lfloor \frac{(n-m)}{2} \right\rfloor$
- **Max. Anzahl Ausfällekorrigierend:** $(n - m)$

Nachricht codieren/Codewort erzeugen:

- Da $q = 5$ alles $mod\ 5$
- **Nachricht:** $a = (3, 4, 2)$
- **Polynom:** $a(x) = a_0 + a_1x + a_2x^2 \xRightarrow{a \text{ einsetzen}} a(x) = 3 + 4x + 2x^2$
- **Stützstellen:** $u_1 = 0, u_2 = 1, u_3 = 2, u_4 = 3, u_5 = 4$

$a(0) = 3 + 4 \cdot 0 + 2 \cdot 0^2 = 3$

$a(1) = 3 + 4 \cdot 1 + 2 \cdot 1^2 = 9\ mod\ 5 = 4$

$a(2) = 3 + 4 \cdot 2 + 2 \cdot 2^2 = 19\ mod\ 5 = 4$

$a(3) = 3 + 4 \cdot 3 + 2 \cdot 3^2 = 33\ mod\ 5 = 3$

$a(4) = 3 + 4 \cdot 4 + 2 \cdot 4^2 = 51\ mod\ 5 = 1$

$c(a) = (a(0), a(1), a(2), a(3), a(4)) = (3, 4, 4, 3, 1)$

Ausfällekorrigieren: Fehlerhaftes Codewort bestimmen

- Achtung rechnen im restklassenring

- Da $q = 5$ alles $mod\ 5$
- **Fehlerhaftes Codewort:** $y = (3, *, 4, 3, *)$
- **Polynom:** $a(x) = a_0 + a_1x + a_2x^2$
- **Stützstellen:** $u_1 = 0, u_2 = 1, u_3 = 2, u_4 = 3, u_5 = 4$

1. Stützstellen und empfangenen (nicht fehlerhafte) Werte in das Polynom einsetzen um die Koeffizienten a_0, a_1, a_2 zu finden

$a(0) = a_0 + a_1 \cdot 0 + a_2 \cdot 0^2 = 3 \Rightarrow \boxed{a_0 = 3}$

$$\begin{aligned} a(2) &= 3 + a_1 \cdot 2 + a_2 \cdot 2^2 = 4 \\ &= 3 + 2a_1 + 4a_2 = 4 \quad | + 2 \mid + a_2 \mid \cdot 3 \\ &= a_1 = 3 + a_2 \end{aligned}$$

$$\begin{aligned} a(3) &= 3 + (3 + a_2) \cdot 3 + a_2 \cdot 3^2 = 3 \\ &= 3 + 3(3 + a_2) + 4a_2 = 3 \\ &= 3 + 4 + 4a_2 + 4a_2 = 3 \\ &= 2 + 3a_2 = 3 \quad | + 3 \mid \cdot 2 \Rightarrow \boxed{a_2 = 2} \Rightarrow \boxed{a_1 = 0} \end{aligned}$$

2. Mit den gefunden Koeffizienten a_0, a_1, a_2 lassen sich nun die fehlenden Stellen herleiten

$a(1) = 3 + \frac{3}{2} \cdot 1 - \frac{1}{2} \cdot 1^2 = \underline{\underline{4}}$

$a(4) = 3 + \frac{3}{2} \cdot 4 - \frac{1}{2} \cdot 4^2 = \underline{\underline{1}}$

Das Codewort ist somit $y = (3, 4, 4, 3, 1)$

Generatormatrix erzeugen:

- Beispiel: $m = 3, n = 5, q = 5$
- $u_1 = 0, u_2 = 1, u_3 = 2, u_4 = 3, u_5 = 4$

$$G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ u_1 & u_2 & \dots & u_n \\ u_1^2 & u_2^2 & \dots & u_n^2 \\ u_1^3 & u_2^3 & \dots & u_n^3 \\ \vdots & \vdots & & \vdots \\ u_1^{m-1} & u_2^{m-1} & \dots & u_n^{m-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 9 & 16 \end{pmatrix}$$

Kontrollmatrix erzeugen:

- Beispiel: $m = 2, n = 5, q = 5$
- $u_1 = 0, u_2 = 1, u_3 = 2, u_4 = 3, u_5 = 4$
- Stützpunkte: $u_1 = 0,\ u_2 = 1??$
- wird als $RS(q, q - m, q)$ aufgefasst

$$H = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & u_3 & u_3^2 & \dots & u_3^{q-m-1} \\ 1 & u_4 & u_4^2 & \dots & u_4^{q-m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u_q & u_q^2 & \dots & u_q^{q-m-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 2 & 4 \\ 0 & 3 & 9 \\ 0 & 4 & 16 \end{pmatrix}$$

2.7 Kontrollmatrix zu Generatormatrix/Generatormatrix zu Kontrollmatrix

- Die Kontrollmatrix eines linearen (n,m)-Codes C über dem Körper $\mathbb{Z}_5$
- Da $\mathbb{Z}_5$ alles *mod* 5

$$H = \begin{pmatrix} 4 & 1 \\ 1 & 1 \\ 2 & 4 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 1 & 2 & 3 \\ 1 & 1 & 4 & 4 \end{pmatrix}^T$$

- Um aus einer Kontrollmatrix eine Generatormatrix abzuleiten machen wir uns die Eigenschaft $G \cdot H = 0$ zu nutze
- Um $G \cdot H = 0$ zu lösen stellen wir eine Tabelle auf die unser Gleichungssystem repräsentiert

g_1	g_2	g_3	g_4	
4	1	2	3	0
1	1	4	4	0

- Nun wählt man 2 beliegt Variablen (Anzahl Variablen: $|g| - 2$)
- $g_3 = \lambda$, $g_4 = \mu$
- Nun löst man das Gleichungssystem für die restlichen g's (g_1 und g_2)
- **Achtung:** Wir rechnen in $\mathbb{Z}_5$
  - Wenn man neative Zahlen hat gilt $-a \ mod \ 5 = b$ und man rechnet mit der positive Zahl weiter
  - Wenn man dividiert immer das multiplikative Inverse der Zahl (z.B a) nehmen, und mit dem Inversen (z.B b) multiplizieren ($a \cdot b \ mod \ 5 = 1$)

Nach g_1 lösen:

$$\begin{aligned} 0 &= 4g_1 + g_2 + 2g_3 + 3g_4 \\ 4g_1 &= -g_2 - 2g_3 - 3g_4 \\ 4g_1 &= 4g_2 + 3g_3 + 2g_4 \mid \textit{variablen einsetzen} \\ 4g_1 &= 4g_2 + 3\lambda + 2\mu \mid \cdot 4 \ (\textit{mult. Inverses von } 4) \end{aligned}$$

$$\boxed{g_1 = g_2 + 2\lambda + 3\mu} \xrightarrow{\text{Nach dem lösen von } g_2} \boxed{g_1 = 4\lambda + 2\mu}$$

Nach g_2 lösen:

$$\begin{aligned} 0 &= g_1 + g_2 + 4g_3 + 4g_4 \\ g_2 &= -g_1 - 4g_4 - 4g_4 \ | g_1 \ \textit{und variablen einsetzen} \\ &= -(g_2 + 2\lambda + 3\mu) - 4\lambda - 4\mu \\ &= -g_2 + -2\lambda - 3\mu - 4\lambda - 4\mu \\ &= 4g_2 + 3\lambda + 2\mu + 1\lambda + 1\mu \\ &= 4g_2 + 4\lambda + 3\mu \mid -4g_2 \\ 2g_2 &= 4\lambda + 3\mu \mid \cdot 3 \ (\textit{mult. Inverses von } 2) \end{aligned}$$

$$\boxed{g_2 = 2\lambda + 4\mu}$$

- Nun lässt sich G bestimmen
- G hat die Form (g_1, g_2, g_3, g_4)

$$G = (4\lambda + 2\mu, 2\lambda + 4\mu, \lambda, \mu)$$

- Nun muss mann für alle Spalten λ und μ so bestimmen das $G \cdot H = 0$ aufgeht
- **Für die ersten Spalte:** $\lambda = 1, \mu = 0$
- **Für die zweite Spalte:** $\lambda = 0, \mu = 1$

$$G = \begin{pmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 4 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}^T$$

2.8 Codes mit Unbekannten

Jedem Wort $x \in \mathbb{Z}_3^2$ wird durch die Abbildung

$$C : x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1 \\ 2x_2 \\ x_1 + x_2 \\ 3x_1 \\ 4x_2 \end{pmatrix}$$

ein Codewort aus $x \in \mathbb{Z}_5^5$ zugeordnet wird

a) Zeigen Sie, dass dadurch ein linearer (5,2)-Code erzeugt wird.

- **Lösung:** beliebige Werte aus $\mathbb{Z}_3$ einsetzen, sodass ein Codewort aus $\mathbb{Z}_5$ rauskommt

$$x_{1,2} = \begin{pmatrix} x_1 = 2 \\ x_2 = 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \cdot 1 \\ 2 + 1 \\ 3 \cdot 2 \\ 4 \cdot 1 \end{pmatrix} = 22314$$

b) Bestimmen Sie die Generatormatrix G und eine Kontrollmatrix des Codes

- **Lösung:** Erstens $x_1 = 1$ und $x_2 = 0$ und zweitens $x_1 = 0$ und $x_2 = 3$, sodass eine kanonische Kontrollmatrix entsteht

$$G = \begin{pmatrix} 1 & 0 & 1 & 3 & 0 \\ 0 & 1 & 3 & 0 & 2 \end{pmatrix} \rightarrow H = \begin{pmatrix} -1 & -3 & -0 \\ -3 & -0 & -2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 2 & 0 \\ 2 & 0 & 3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

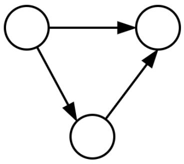
3 Graphentheorie

3.1 Grundbegriffe

$G(V, E)$, V = Vertices = Knoten, E = Edges = Kanten

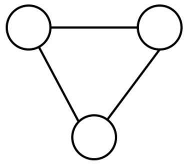
Gerichteter Graph

- Kante geht nur in *eine* Richtung



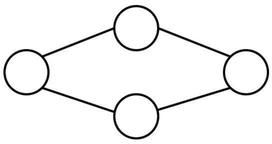
Ungerichteter Graph

- Kante geht nur in *beide* Richtung



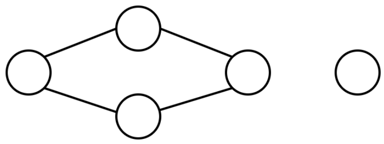
Zusammenhängender Graph

- Verbindung von einem Knotenpunkt zu allen anderen
- Verbindungen müssen nicht direkt sein
- Gibt keine isolierten Knoten



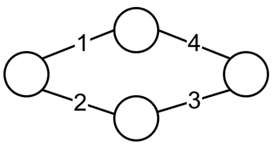
Nichtzusammenhängender Graph

- Wenn ein Knoten isoliert ist und keine Verbindung herrscht



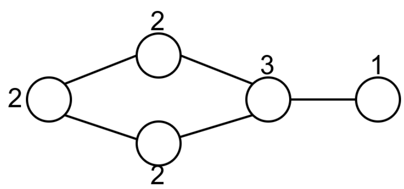
Gewichteter Graph

- Die Kanten bekommen Gewichte zugewiesen



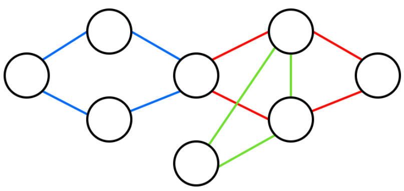
Knotengrad

- Anzahl Kanten die von einem Knoten ausgehen



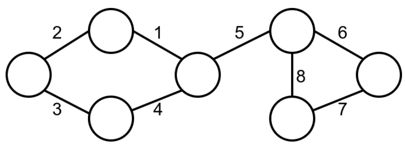
Euler-Zyklus/Eulersch

- Jede Kante wird genau einmal durchlaufen
- Kanten dürfen somit nicht mehrmals durchlaufen werden
- Anfangszustand = Endzustand
- Alle Knoten haben einen geraden Grad
- Ist bei Adjazenzmatrix Außengrad $\neq$ Innengrad dann kein Euler-Zyklus



Euler-Pfad

- Jede Kante wird genau einmal durchlaufen
- Keine Kante wird mehrmals durchlaufen
- Startknoten muss nicht gleich Endknoten sein
- Genau zwei Knoten haben einen ungeraden Grad



Hamilton-Zyklus

- Jeder Knoten wird genau einmal besucht
- Anfangsknoten = Startknoten

Hamilton-Pfad

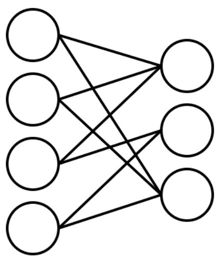
- Jeder Knoten wird genau einmal durchlaufen
- Nicht jede Kante muss durchlaufen werden
- Keine Kante wird mehrmals durchlaufen
- Startknoten muss nicht gleich Endknoten sein

Isomorph

- Bijektive Abbildung
- Kanten, Knoten und Knotengrad müssen gleich sein

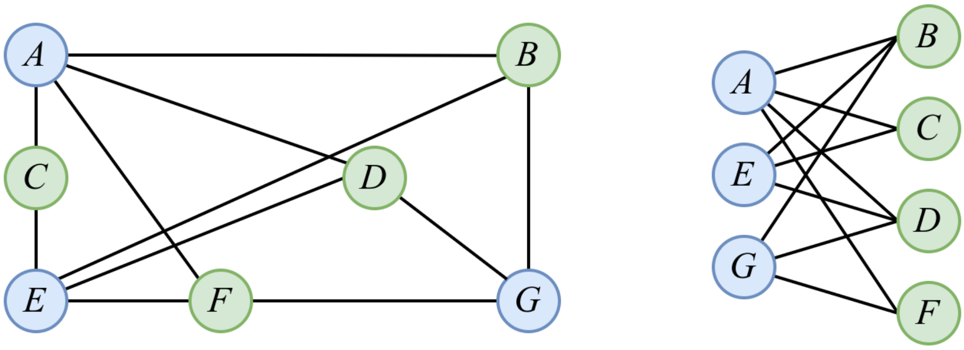
Bipartit

- Man kann alle Knoten in zwei (disjunkte) Teilmengen/Gruppen aufteilen
- Jede Verbindung geht dabei von einer Teilmenge/Gruppe in die andere
- Es darf allerdings keine Verbindung der Knoten innerhalb der eigenen Teilmenge/Gruppe geben
- **Hamilton-Zyklus:**
  - Wenn beide Teilmenge (A und B) die gleiche Anzahl an Knoten haben ($|A| = |B|$)
  - Graph vollständig ist
- **Hamilton-Pfad:**
  - Wenn gilt $|A| \leq |B| + 1$ *und* $|A| \geq |B| - 1$



Beispiel: Zeigen ob ein Graph bipartit ist

- Beliebigen Knoten auswählen und färben (z.B blau).
- Alle benachbarte Knoten mit einer anderen Farbe färben (z.B grün)
- Widerholen: Alle Knoten die an einen blauen Knoten angrenzen, grün und alle Knoten, die an einen grünen Knoten angrenzen, blau färben.
- Prüfung: Wenn man auf einen Knoten stößt, der bereits gefärbt ist und dessen Farbe sich von der Farbe unterscheidet, die man ihm zuweisen möchten, dann ist der Graph *nicht bipartit*. Wenn man alle Knoten ohne solche Konflikte färben konnte, ist der Graph *bipartit*.
- Die blauen Knoten sind am Ende die eine Teilmenge/Gruppe und die grünen Knoten sind am Ende die andere Teilmenge/Gruppe



Planar

- Ein Graph ist planar, wenn man ihn so zeichnen kann, dass sich seine Kanten nirgendwo kreuzen.

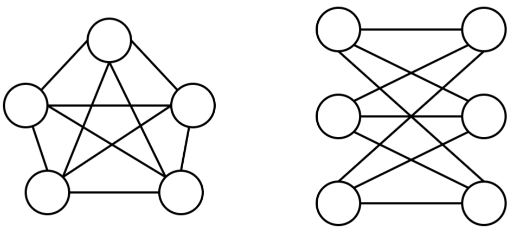
(Nicht) Planarität prüfen:

1 Methode:

- $n = |V|$ (Anzahl Knoten) und $m = |E|$ (Anzahl Kanten) bestimmen
- Hat der Graph einen Zyklus der Länge 3?
  - $\rightarrow$ **Ja:** $m \leq 3n - 6$
  - $\rightarrow$ **Nein:** $m \leq 2n - 4$
  - Gilt $m \not\leq 3n - 6$ oder $m \not\leq 2n - 4$ ist der Graph *nicht planar*
  - **Achtung:** Die Formel macht nur eine Aussage darüber, ob der Graph nicht planar ist, aber nicht, ob ein Graph planar ist. Gilt also $m \leq 3n - 6$ oder $m \leq 2n - 4$ heißt es nicht, dass der Graph planar ist. Man kann also nur die Nichtplanarität überprüfen.

2 Methode:

- Ist der K_5 oder $K_{3,3}$ der Graph oder ein Teilgraph ist der ein Graph immer *nicht planar*



K_5 (links), $K_{3,3}$ (rechts)

3 Methode:

- Graphen versuchen so zu zeichnen das ein planarer Graph rauskommt (umständlich)

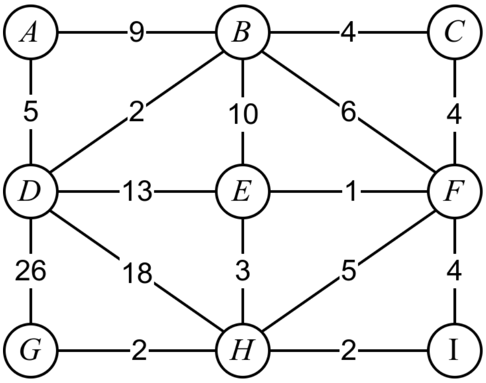
3.2 Anmerkung Adjazenzmatrix

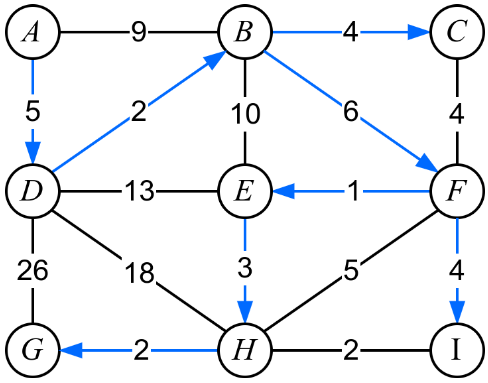
- Der Außdengrad d^+ eines Knoten ist die Anzahl Kanten ihn verlassen
- Der Innengrad d^- eines Knoten ist die Anzahl Kanten die zu ihn führen
- Ein Euler-Zyklus kann nur dann existieren, wenn für jeden Knoten gilt eines gerichteten Graphens gilt: $Außengrad = Innengrad$
  - **Achtung:** Es ist eine notwendige Bedingung, d.h falls dies zutriftt kann aber muss kein Euler-Zyklus existieren

	a	b	c	d^+
a	0	1	2	3
b	2	3	0	5
c	3	0	1	4
d^-	5	4	3	12

3.3 Kürzester Weg

Dijkstra

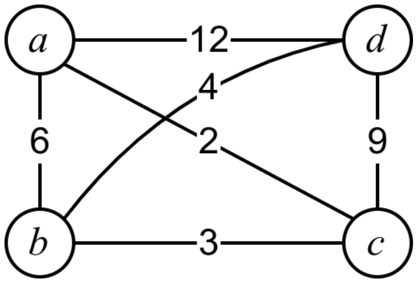




A	B	C	D	E	F	G	H	I	S
0(A)									A
0(A)	9(A)		5(D)						A, D
0(A)	7(D)		5(D)	18(E)		31(D)	23(D)		A, D, B
0(A)	7(D)	11(B)	5(D)	17(E)	13(B)	31(D)	23(D)		A, D, B, C
0(A)	7(D)	11(B)	5(D)	17(E)	13(B)	31(D)	23(D)		A, D, B, C, F
0(A)	7(D)	11(B)	5(D)	14(F)	13(B)	31(D)	18(F)	17(F)	A, D, B, C, F, E
0(A)	7(D)	11(B)	5(D)	14(F)	13(B)	31(D)	17(E)	17(F)	A, D, B, C, F, E, H
0(A)	7(D)	11(B)	5(D)	14(F)	13(B)	31(D)	17(E)	17(F)	A, D, B, C, F, E, H, I
0(A)	7(D)	11(B)	5(D)	14(F)	13(B)	31(D)	17(E)	17(F)	A, D, B, C, F, E, H, I, G

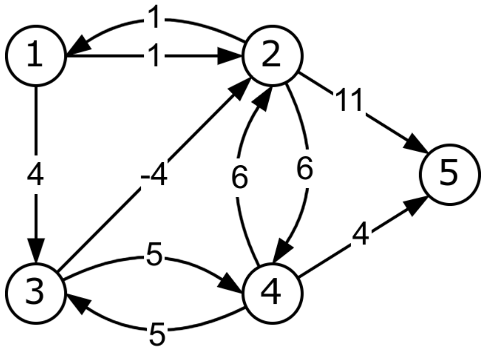
- Kann bei negativen Kanten ein falsches Ergebnis liefern, da er davon ausgeht, dass einmal besuchte Knoten nicht mehr verbessert werden können. Bei negativen Kanten ist dies jedoch möglich.
- Funktioniert nicht bei negativen Kanten Gewichte deshalb lieber Bellman-Ford-Algorithmus

Floyd



	i	a	b	c	d	a	b	c	d
	a	0	6	2	12	a	a	a	a
	b	6	0	3	4	b	b	b	b
	c	2	3	0	9	c	c	c	c
	d	12	4	9	0	d	d	d	d
1	a	0	6	2	12	a	a	a	a
1	b	6	0	3	4	b	b	b	b
1	c	2	3	0	9	c	c	c	c
1	d	12	4	9	0	d	d	d	d
2	a	0	6	2	10	a	a	a	b
2	b	6	0	3	4	b	b	b	b
2	c	2	3	0	7	c	c	c	b
2	d	10	4	7	0	b	d	b	d
3	a	0	5	2	9	a	c	a	b
3	b	5	0	3	4	c	b	b	b
3	c	2	3	0	7	c	c	c	b
3	d	9	4	7	0	c	d	b	d
4	a	0	5	2	9	a	c	a	b
4	b	5	0	3	4	c	b	b	b
4	c	2	3	0	7	c	c	c	b
4	d	9	4	7	0	c	d	b	d

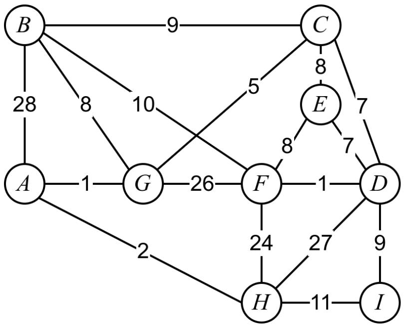
FIFO

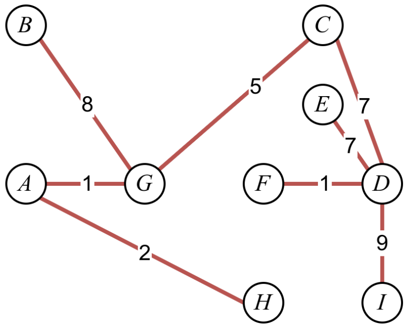


D(s,1)	D(s,2)	D(s,3)	D(s,4)	D(s,5)	R(1)	R(2)	R(3)	R(4)	R(5)	S
0										1
0	1	4				1	1			2 < 3
0	1	4	7	12		1	1	2	2	3 < 4 < 5
0	0	4	7	12		3	1	2	2	4 < 5 < 2
0	0	4	6	11		3	1	2	4	5 < 2
0	0	4	6	11		3	1	2	4	2
0	0	4	6	10		3	1	2	4	4
0	0	4	6	10		3	1	2	4	5
0	0	4	6	10		3	1	2	4	

3.4 Prim

- **Ziel:** Erstellung eines minimalen Spannbaum, von einem gewichteten ungerichteten Graphen.
- Die Kantengewichte sind im vorhinein nicht bekannt
- Es wird immer die Kante mit dem kleinsten Gewicht (welches aktuell bekannt ist) benutzt

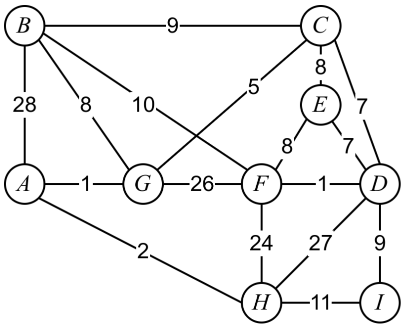


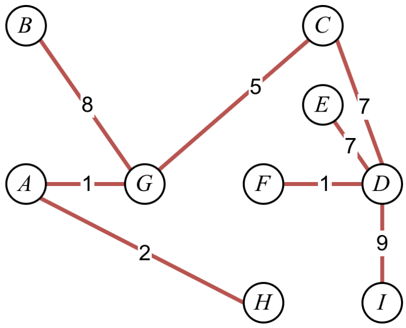


- Start: A, G
- Füge hinzu: A, H
- Füge hinzu: G, C
- Füge hinzu: C, D
- Füge hinzu: D, F
- Füge hinzu: D, E
- Füge hinzu: B, G
- Füge hinzu: D, I

3.5 Kruskal

- **Ziel:** Erstellung eines minimalen Spannbaum, von einem gewichteten ungerichteten Graphen.
- Alle Kantengewichte sind im vorhinein bekannt

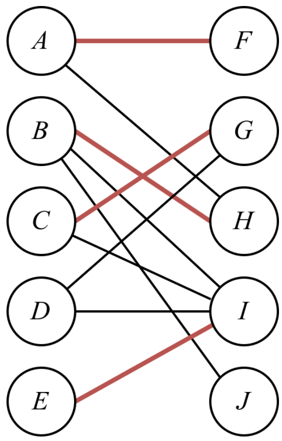




- Start: A, G
- Füge hinzu: F, D
- Füge hinzu: A, H
- Füge hinzu: G, C
- Füge hinzu: F, D
- Füge hinzu: C, D
- Füge hinzu: D, E
- Füge hinzu: G, B
- Füge hinzu: D, I

3.6 Matching

Unter Matching eines Graphen versteht man eine Menge von Kanten von G, sodass keine zwei Kanten einen Knoten gemeinsam haben.



$M = (A, F), (B, H), (C, G), (E, I)$ ist ein Matching der Mächtigkeit 4

Heiratssatz

Im bipartiten Graphen $G = (S + T, E)$ gilt:

$$m(G) = |S| \Leftrightarrow |A| \leq |N(A)| \text{ für alle } A \subseteq S$$

Gesättigtes Matching

Ein gesättigtes Matching ist ein Matching, bei dem man keinen Kante hinzufügen kann, ohne die Matching-Regeln verletzt werden.

Maximales Matching

Ein maximales Matching in einem Graphen ist ein Matching, das die maximale/größtmögliche Anzahl von Kanten und Knotenpaare verbindet, ohne dass zwei Kanten einen Knoten gemeinsam haben. In einem Graph existiert also kein Matching, dass mehr Kanten hat als ein maximales Matching. Jedes maximales Matching ist somit auch ein gesättigtes Matching.

Vollständiges Matching

Ein vollständiges/perfektes Matching ist ein Matching, das alle Kanten des Graphen abdeckt, sodass jeder Knoten des Graphen mit genau einer Kante des Matchings verbunden ist.

Hopcraft-Karp Algorithmus

3.7 Hierholzer für ungerichteten/gerichtete Graphen

Vorraussetzungen:

- Der Graph muss zusammenhängend sein.
- Alle Knoten außer zwei haben denselben Innengrad wie Außengrad. (Mit ausnahme von zwei außnahmen)
- Einer der beiden ausnahmsweisen Knoten hat einen Außengrad, der um eins größer ist als sein Innengrad (dies ist der Startknoten).
- Der andere ausnahmsweise Knoten hat einen Innengrad, der um eins größer ist als sein Außengrad (dies ist der Endknoten).
  - Der Endknoten wird automatisch zum Endknoten

Ablauf:

- **Startknoten:** Außengrad eins größer als Innengrad
- Beginne beim Startknoten und folge den Kanten des Graphen, bis du zu einem Knoten kommst, von dem aus du keine weiteren unbesuchten Kanten mehr folgen kannst.
- Füge den gewählten Weg zum Euler-Pfad hinzu
- Solange es noch Knoten mit nicht markierten Kanten in Weg gibt
  - Wähle als Startknoten einen mit noch freien Kanten
  - Finde einen Zyklus vom gewählten Startknoten bis wieder zurück zum Startknoten
  - Verschmelze Weg und Zyklus