Algorithmen für Gruppen und Codes

Inhaltsverzeichnis

I Gruppen 3

1 Grundlagen der Gruppentheorie 3

2 Graphen und Bahnen 8

3 Transversalen und der Satz von Schreier 13

4 Stabilisatorketten 16

5 Der Schreier-Sims-Algorithmus 21

6 Basiswechsel 25

7 Kurze Faktorisierungen 29

8 Faktorisierung mit Freiheitsgraden 36

9 Endliche Körper und ihre Darstellungen 38

10 Matrixgruppen über endlichen Körpern 41

II Codes 44

11 Lineare Blockcodes 44

12 Das Minimalgewicht linearer Codes 46

13 Zyklische und quasizyklische Codes 55

14 Kongruenzen für die Gewichte im Code 59

15 Effizientes Aufzählen von Codeworten 63

Literatur 64

Index 65

Teil I

Gruppen

1 Grundlagen der Gruppentheorie

Definition 1.1 Eine **Gruppe** G ist eine Menge mit einer Verknüpfung $\cdot : G \times G \to G$, so dass gilt:

1. Die Verknüpfung $\cdot$ ist **assoziativ**, d.h. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
2. Es gibt ein **neutrales Element** 1_G mit den Eigenschaften $x \cdot 1_G = x = 1_G \cdot x$.
3. Zu jedem $x \in G$ gibt es ein **inverses Element** $x^{-1} \in G$ mit $x \cdot x^{-1} = 1_G = x^{-1} \cdot x$.

Man nennt die Gruppe **abelsch** oder **kommutativ**, wenn zusätzlich $x \cdot y = y \cdot x$ für alle $x, y \in G$ gilt.

Definition 1.2 Eine Teilmenge $U \subseteq G$, die mit der Verknüpfung $\cdot$ selbst wieder eine Gruppe bildet, nennt man **Untergruppe**, schreibe $U \leq G$.

Beispiel 1.3

- Die Einheitengruppe eines Rings bzw. eines Körpers ist eine multiplikative Gruppe.
- Die additive Gruppe eines Rings; z.B. $(\mathbb{Z}, +)$ mit dem neutralen Element 0.
- Permutationsgruppen: Vertauschung von Elementen einer Menge.
- Matrixgruppen: Invertierbare $n \times n$-Matrizen, etwa Rotationen im $\mathbb{R}^3$.
- Affine Transformationen (Basiswechsel mit Translation):

  $$x \mapsto Ax + b$$

  Gruppenelemente: $(A, b) \in \mathrm{GL}_n(\mathbb{K}) \times \mathbb{K}^n$.
  Neutrales Element: $(I_n, 0)$.

  $$\begin{aligned}((A_1, b_1) \cdot (A_2, b_2))\, x &= (A_1, b_1) \cdot (A_2 x + b_2) \\ &= A_1 A_2 x + (A_1 b_2 + b_1) \\ &= (A_1 A_2, A_1 b_2 + b_1) x\end{aligned}$$

Definition 1.4 Eine Abbildung $\Phi : G \to H$ zwischen zwei Gruppen G, H heißt **Homomorphismus**, wenn für alle $x, y \in G$ gilt:

$$\Phi(x \cdot y) = \Phi(x) \cdot \Phi(y).$$

Ist Φ bijektiv, so spricht man von einem **Isomorphismus**.

Definition 1.5 Sei $x \in G$. Die Menge

$$\langle x \rangle = \{x^n : n \in \mathbb{Z}\}$$

heißt die von x **erzeugte Untergruppe**. Allgemeiner heißt für eine Teilmenge $S \subseteq G$ die Menge

$$\langle S \rangle = \bigcap_{H \leq G \text{ mit } S \subseteq H} H$$

die von S erzeugte Untergruppe, S ist die **Erzeugermenge** von $\langle S \rangle$. Die Gruppe G heißt **zyklisch**, wenn es ein $x \in G$ gibt mit $G = \langle x \rangle$.

Definition 1.6 Die **Ordnung** von $x \in G$ ist $\mathrm{ord}(x) := |\langle x \rangle|$. Allgemeiner bezeichnet man als **Ordnung** von G auch die Anzahl $|G|$ der Elemente von G.

Definition 1.7 Sei G eine Gruppe, $S \subset G$. Die **freie Gruppe** oder **Wortgruppe** $F(S)$ von S besteht aus den Termen, die durch formales Hintereinanderschreiben (d.h. Konkatenation) der Elemente von S enstehen. Das neutrale Element ist das leere Wort, das inverse Element eines Wortes $w_1 \cdots w_k \in F(S)$ ist durch $w_k^{-1} \cdots w_1^{-1}$ gegeben.

Definition 1.8 Sei H eine Untergruppe von G und $g \in G$. Dann heißt die Menge

$$gH = \{g \cdot x : x \in H\}$$

die (Links-)**Nebenklasse** von g bzgl. H. Entsprechend ist die Rechtsnebenklasse definiert. Die Anzahl der Linksnebenklassen ist gleich der Anzahl der Rechtsnebenklassen, sie heißt **Index** von H in G, schreibe $[G : H]$.

Satz 1.9 (Satz von Lagrange)
Ist G endlich, so ist

$$|G| = |H| \cdot [G : H].$$

Definition 1.10 Eine Untergruppe $N \leq G$ heißt **Normalteiler**, wenn für alle $g \in G$ und $x \in N$ gilt: $gxg^{-1} \in N$. Schreibe dafür $N \trianglelefteq G$.

Definition 1.11 Eine Gruppe G **operiert** auf einer Menge M (von links), wenn es eine Verknüpfung $* : G \times M \to M$ gibt mit:

1. $(g_1 \cdot g_2) * m = g_1 * (g_2 * m)$.
2. $1_G * m = m$.

Entsprechend ist die Operation von rechts definiert. Schreibweise:

$$m^g := m * g$$

Beispiel 1.12 Eine **Permutationsgruppe** ist eine Untergruppe der **symmetrischen Gruppe** S_M einer Menge M:

$$\mathsf{S}_M = \{\sigma : M \to M : \sigma \text{ bijektiv}\}.$$

Permutationsgruppen operieren auf der Menge M durch Abbilden. Meistens ist $M = \{1, ..., n\}$, schreibe dann auch $\mathsf{S}_n = \mathsf{S}_M$. Die Gruppe S_n hat die Ordnung $n!$.

Mögliche Darstellungen der Elemente der Permutationsgruppe S_n:

1. Liste aller Bilder unter der Permutation, z.B.

   $$\pi = [2, 1, 3, 4, 5, 6, 9, 8, 7],$$

   d.h. $\pi[i] =$ Bild von i unter der Permutation π.

2. Zykelschreibweise:

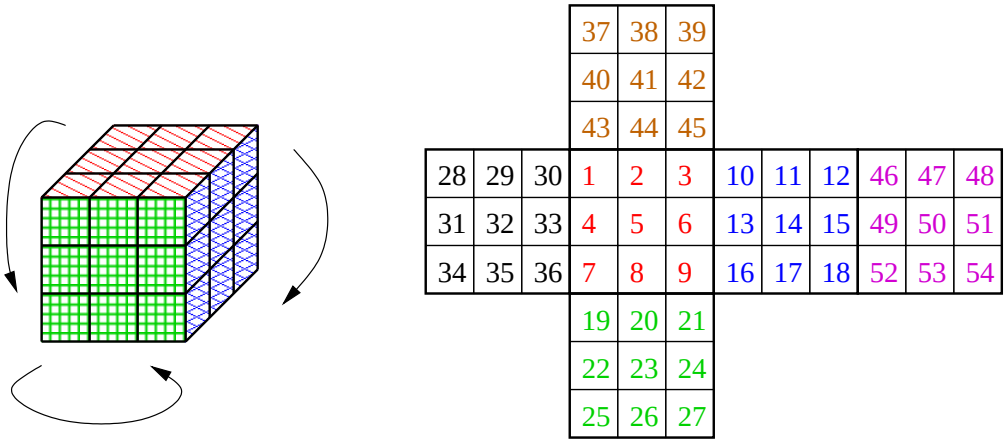
   $$\begin{aligned}\pi &= (1\ 2)(3)(4)(5)(6)(7\ 9)(8)\\ &= (1\ 2)(7\ 9)\end{aligned}$$

   Verknüpfung: $\pi_1 = (1\ 2), \pi_2 = (2\ 3)$:

   $\pi_1 \cdot \pi_2 = (1\ 3\ 2)$ falls „π_1 zuerst“, d.h. Operations von rechts
   $\pi_1 \cdot \pi_2 = (1\ 2\ 3)$ falls „π_2 zuerst“, d.h. Operations von links

   Konvention: Operation von rechts.

Beispiel 1.13 Numeriert man die Seitenflächen beim **Rubik's Cube** von 1 bis 54, so kann man die zulässigen Spielzüge als Erzeuger einer Untergruppe der S_{54} auffassen.



Definition 1.14 Sei G eine Gruppe, die auf einer Menge M operiert. Die **Bahn** oder der **Orbit** eines Elements $m \in M$ ist die Menge

$$m^G := \{m^g : g \in G\},$$

d.h. alle Bilder von m bzgl. der Operation von G auf M.

Definition 1.15 Die **Fixgruppe** (oder auch der **Stabilisator**) eines Elementes ist die Teilmenge G_m von G, die m **stabilisiert**, d.h.

$$G_m := \{g \in G : m^g = m\}.$$

Bemerkung 1.16 G_m ist eine Untergruppe von G.

Lemma 1.17 Es gilt folgende **Bahnbilanz**

$$|m^G| = \frac{|G|}{|G_m|} = [G : G_m],$$

d.h. Gruppenordnung = Bahnlänge · Ordnung der Fixgruppe.

BEWEIS: Jede Gruppe kann als disjunkte Vereinigung von Nebenklassen einer Untergruppe geschrieben werden:

$$G = G_m \dot\cup G_m\tau_1 \dot\cup ... \dot\cup G_m\tau_{l-1}.$$

Aus $G_m\tau_i \cap G_m\tau_j = \emptyset$ für $i \neq j$ folgt auch $m^{G_m\tau_i} \cap m^{G_m\tau_j} = \emptyset$, also gilt für die Bahn:

$$\begin{aligned} m^G &= m^{G_m} \dot\cup m^{G_m\tau_1} \dot\cup ... \dot\cup m^{G_m\tau_{l-1}} \\ &= \{m\} \dot\cup \{m^{\tau_1}\} \dot\cup ... \dot\cup \{m^{\tau_{l-1}}\} \\ &= \{m^{\tau_i} : i = 0, ..., l-1\} \end{aligned}$$

Somit ist $|m^G| = l$ bei l Nebenklassen. Die Behauptung folgt nun aus dem Satz von Lagrange. ■

Definition 1.18 Die Untergruppe $K \leq G$, die alle Elemente von M punktweise fest lässt, also

$$K := \{g \in G : \forall m \in M : m^g = m\},$$

heißt **Kern** der Operation.

Grundidee: Untersuche die Gruppe anhand ihrer Operation auf M bzw. auf einzelnen Elementen von M.

Problem: Ist die Menge M „zu klein“, so erhält man nicht alle Informationen über die Gruppe, indem man ihre Operation auf M untersucht.
Extremfall: $M = \{1\}$, also $m^g = m$ für alle $g \in G$ (triviale Operation).

Voraussetzung: Die Operation von G auf M muss **treu** sein, d.h. zu jedem Gruppenelement $g \in G, g \neq 1_G$, gibt es mindestens ein $m \in M$ mit $m^g \neq m$ (also $K = \{1_G\}$). Ansonsten erhält man nur partielle Informationen.

Lemma 1.19 Zu jeder Gruppe G gibt es eine Menge M, auf der G treu operiert.

BEWEIS: Operation der Gruppe auf sich selbst ($G = M$) durch Rechts- bzw. Linksmultiplikation (meistens von links):

$$g * m := gm$$

mit $g, m \in G$. Das neutrale Element von G ist eindeutig ⇒ Kern der Operation trivial. ■

Bemerkung 1.20 Eine Gruppe operiert auch durch **Konjugation** auf sich selbst:

$$g^h := h^{-1}gh$$

mit $g, h \in G$.

Bemerkung 1.21 Nummeriert man die Gruppenelemente $g_i \in G$, so liefert die Operation der Gruppe auf sich selbst per (Links-/Rechts-)Multiplikation einer **Permutationsdarstellung** von G auf $|G|$ Punkten.

In den folgenden Kapiteln werden alle Gruppen, sofern nichts anderes gesagt wird, als *endlich* vorausgesetzt.

2 Graphen und Bahnen

Ausgangssituation: Die (Unter-)Gruppe G ist gegeben durch eine Menge S von Erzeugern aus einer bekannten Gruppe G_0 (z.B. S_n oder $\mathrm{GL}_n(\mathbb{K})$).

Fragestellung:

- Wieviele Elemente hat die von einer Menge $S = \{s_1, ..., s_n\} \subset G_0$ erzeugte Untergruppe $G = \langle s_1, ..., s_n \rangle$? Bei Matrixgruppen über unendlichen Körpern: Ist G endlich?
- Gegeben ein Element $g \in G_0$. Teste, ob $g \in G$!
- Stelle $g \in G$ als Produkt der Erzeuger $s_1, ..., s_n$ dar!

(naiver) Ansatz: Bestimmung der Anzahl der Elemente einer Gruppe: Zähle alle Elemente auf.

Beispiel 2.1 Es sei $G_0 = \mathsf{S}_4$ (Permutationen von $\{1, 2, 3, 4\}$) und $s_1 = \sigma = (1\ 2)(3\ 4)$, $s_2 = \tau = (1\ 2\ 3)$.

$\cdot$	σ	τ
id	σ	τ
$(1\ 2)(3\ 4)$	id	$(1\ 3\ 4)$
$(1\ 3\ 4)$	$(1\ 4\ 2)$	$(2\ 3\ 4)$
$(2\ 3\ 4)$	$(1\ 2\ 4)$	$(1\ 2)(3\ 4)$
$(1\ 2\ 3)$	$(2\ 4\ 3)$	$(1\ 3\ 2)$
$(2\ 4\ 3)$	$(1\ 2\ 3)$	$(1\ 2\ 4)$
$(1\ 2\ 4)$	$(2\ 3\ 4)$	$(1\ 3)(2\ 4)$
$(1\ 3)(2\ 4)$	$(1\ 4)(2\ 3)$	$(2\ 4\ 3)$
$(1\ 3\ 2)$	$(1\ 4\ 3)$	id
$(1\ 4\ 3)$	$(1\ 3\ 2)(1\ 4)$	$(2\ 3)$
$(1\ 4)(2\ 3)$	$(1\ 3)(2\ 4)$	$(1\ 4\ 2)$
$(1\ 4\ 2)$	$(1\ 3\ 4)$	$(1\ 4\ 3)$

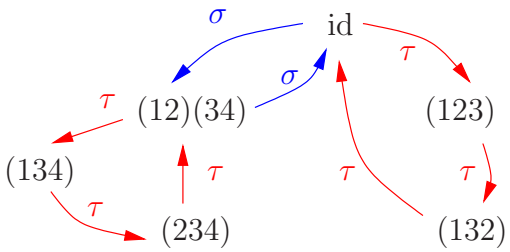
Man erhält $|G| = 12$. Aufwand: $|G| \cdot |S|$ Gruppenoperationen. Bei Permutationsgruppen: $|\mathsf{S}_n| = n! = \mathcal{O}(n^n)$.

Definition 2.2 Der **Cayley-Graph** zu einer von der Menge S erzeugten Gruppe G ist ein gerichteter Graph mit:

- Knoten: alle Gruppenelemente.
- Kanten: beschriftet mit den Erzeugern; Kante $(g, g \cdot s_i)$: Multiplikation eines Gruppenelements g mit dem Erzeuger s_i liefert den Zielknoten $g \cdot s_i$.

Beispiel 2.3 (Fortsetzung von Beispiel 2.1) $\sigma = (1\ 2)(3\ 4), \tau = (1\ 2\ 3)$.

(unvollständiger) Cayley-Graph



...

Bemerkung 2.4 Die Faktorisierung (d.h. Zerlegung) eines Gruppenelements g in ein Produkt von Erzeugern entspricht der Suche eines Weges vom Knoten id zum Knoten g. Der kürzeste Weg im Cayley-Graphen entspricht der kürzesten Faktorisierung.

Vorteil: Die gesamte Information über Gruppe und Erzeuger ist im Cayley-Graphen vorhanden $\Rightarrow$ optimale Faktorisierung.

Nachteil: Speicherplatz für den Cayley-Graphen: $\#\text{Knoten} = |G|$, $\#\text{Kanten} = |G| \cdot |S|$.

Man kann sich mit einem *partiellen* Cayley-Graphen behelfen: Der Cayley-Graph wird dabei über Breitensuche aufgebaut. Beschränkung der Tiefe entspricht Anzahl der Faktoren einer Zerlegung.

Bemerkung 2.5 Der Durchmesser des Cayley-Graphen ist die maximale Länge der Faktorisierung eines Gruppenelements bzgl. der Erzeugermenge S.

Konvention: Meist betrachtet man die Erzeugermenge bzgl. Inversion abgeschlossen, d.h. sowohl s_i als auch s_i^{-1} sind Erzeuger. Damit wird der Cayley-Graph ungerichtet (aber mit beschrifteten Kanten).

Bemerkung 2.6 Kreise im Cayley-Graphen liefern Relationen zwischen den Erzeugern (z.B. $\sigma^2 = \mathrm{id}$ oder $\tau^3 = \mathrm{id}$ in Beispiel 2.1). So erhält man eine „Darstellung der Gruppe in Erzeugern und Relationen“.

Problem: Es ist unentscheidbar, ob eine durch Erzeuger und Relationen dargestellte Gruppe die triviale Gruppe ist.

Beispiel 2.7 $G = \langle s, t\rangle$, wobei diese Relationen gelten sollen: $t^2 = \mathrm{id}, s^n = \mathrm{id}, sts = t$. Verknüpfungen in der Wortgruppe (siehe Definition 1.7): Konkatenation der Erzeuger, Inversenbildung:

$$\begin{aligned} t^2 = \mathrm{id} &\Rightarrow \text{„maximal ein } t\text{“} \\ t^2 = \mathrm{id}, st = ts^{-1} &\Rightarrow \text{„}t \text{ immer links von } s\text{“} \\ s^n = \mathrm{id} &\Rightarrow \text{„maximal } (n-1)\text{-mal } s \text{ hintereinander“} \end{aligned}$$

$\Rightarrow$ insgesamt $2n$ Elemente $t^i s^j, i = 0, 1, j = 0, ..., n-1$.

Bemerkung 2.8 Die Faktorisierung von Gruppenelementen in Erzeuger aus S entspricht formal einem Homomorphismus von G in die Wortgruppe $F(S)$ von S modulo der Relationen (diese entsprechen gewissen Produkten von Symbolen s_i aus S).

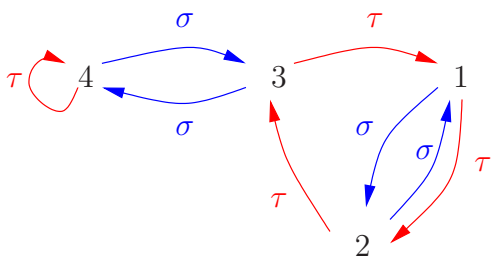
Problem: Die Faktorisierung über den Cayley-Graphen ist zu aufwendig.

Idee: Um den Aufwand zu reduzieren, betrachte die Operationen der Gruppe auf einer Menge M mit $|M| \ll |G|$. Erhalte Informationen aus der Bahn von $m \in M$: analog zum Cayley-Graphen kann man den **Graphen der Bahn** m^G bilden:

- Knoten = Elemente der Bahn.
- Kantenbeschriftung = Erzeuger $s_i \in S$; Kante $(m_j, m_j^{s_i})$.

Beispiel 2.9 (Fortsetzung von Beispiel 2.1) Betrachte $m = 4$:

Graph der Bahn von 4



Bemerkung 2.10 Der Graph der Bahn m^G liefert Information modulo des Stabilisators G_m, d.h. Information über die Nebenklassen von G_m in G.

Idee: Codiere die Information über die Nebenklassen in die Bahn.

Algorithmus 2.11 Berechnung der Bahn

Eingabe: $m_0 \in M, S = \{s_1, ..., s_n\}$
Ausgabe: die Bahn m_0^G (und Zusatzinformation, s.u.)

Bahn $:= \{m_0\}$
neu $:=$ Bahn
while neu $\neq \emptyset$ **do**
    $A := \{m^{s_i} : m \in$ neu$, s_i \in S\}$
    neu $:= A \backslash$ Bahn
    Bahn $:=$ Bahn $\cup A$
return Bahn

Aufwand: $|m^G| \cdot |S| \leq |M| \cdot |S|$ (im Vergleich zu $|S| \cdot |G|$ für den Cayley-Graphen).

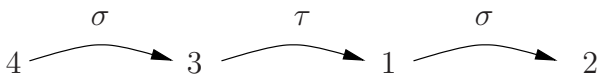
Zusatzinformation: Für ein festes m_0 speichere zu jedem Element m_i der Bahn ein Gruppenelement g_i mit $m_i = m_0^{g_i}$. Das Element g_i wird als Produkt von Erzeugern generiert, d.h. man kann die Faktorisierung speichern (z.B. die Nummern der Erzeuger). Es genügt, einen Spannbaum für den Graphen der Bahn von m_0 mit Wurzel m_0 zu berechnen.

- Teste für jeden Erzeuger s_i, ob m^{s_i} schon in der Bahn liegt. Falls nein, speichere den neuen Punkt m^{s_i} und die „Kante s_i“ (das entspricht (m, m^{s_i}) beschriftet mit s_i).
- Speichere zum Punkt m^{s_i} nur den Erzeuger s_i.

Definition 2.12 Der Spannbaum für den Graphen der Bahn eines Elementes $m_0 \in M$ mit Wurzel m_0 heißt **Schreier-Baum** (zu m_0). Der **Schreier-Vektor** v_{m_0} hat an der Stelle m als Eintrag den Generator s_i, mit dem der Punkt m in der Bahn erreicht wurde. Falls $m \notin m_0^G$, schreibe das Symbol $\perp$ und für die Identität schreibe den Index 0.

Beispiel 2.13 (Fortsetzung von Beispiel 2.1)

Schreier-Baum zu $4^{\langle\sigma,\tau\rangle}$



Position	1	2	3	4	5
Erzeuger	τ	σ	σ	id	$\perp$
Nr. des Erz.	2	1	1	0	$\perp$

Speicheraufwand: $|M| \cdot \mathcal{O}(\log |S|)$ (bei binärer Codierung).

Algorithmus 2.14 Test auf Enthaltensein in der Bahn und Darstellung in Erzeugern

Teste, ob ein $\mu \in M$ in der Bahn von m_0 liegt, d.h. löse die Gleichung $m_0^g = \mu$ für $g \in G = \langle S \rangle$. Stelle g als Produkt von Erzeugern s_j aus S dar.

Eingabe: μ und der Schreiervektor v_{m_0}
Ausgabe: g mit $m_0^g = \mu$

$g := \mathrm{id}$
if $v_{m_0}(\mu) = \perp$ **then** error : „μ nicht in der Bahn enthalten“
while $\mu \neq m_0$ **do**
    $s := v_{m_0}(\mu)$
    $\mu := \mu^{s^{-1}}$ *// Inversenbildung und Operation auf M*
    $g := s \cdot g$ *// Gruppenmultiplikation*
return g

Aufwand: $\mathcal{O}(|M|)$ Gruppenoperationen.
„worst case“: Schreier-Baum degeneriert zur linearen Kette.

Alternativer Aufwand $\mathcal{O}(1)$, wenn man zu jedem Punkt im Schreier-Baum den Weg speichert. Dabei wächst natürlich der Speicheraufwand.

Bemerkung 2.15 Fasst man in Algorithmus 2.14 die Anweisung $g := s \cdot g$ als Konkatenation von Erzeugerworten auf, so liefert der Algorithmus die Darstellung von g als Produkt von Erzeugern aus S.

3 Transversalen und der Satz von Schreier

Sei $m \in M$, $H = G_m$. Zerlege G in Nebenklassen von H:

$$G = \bigcup_{g \in G} Hg = \dot{\bigcup_{g \in T}} Hg$$

Die Menge T enthält Repräsentanten der Nebenklassen von H in G (i.A. ist T keine Untergruppe).

Definition 3.1 Eine Menge T von Gruppenelementen aus G heißt **Transversale** von H in G, falls gilt:

1. $H \cap T = \{\mathrm{id}\}$.
2. $G = HT$.
3. $|T| = [G : H] = \frac{|G|}{|H|}$.

Definition 3.2 Um zu bestimmen, in welcher Nebenklasse ein Gruppenelement liegt, definieren wir die **kanonische Projektion**

$$\tau : G \to T, \quad g \mapsto \tau(g)$$

so, dass $H\tau(g) = Hg$. Die Funktion τ wählt also einen Repräsentanten der Nebenklasse von g aus.

Bemerkung 3.3

1. $g \in H \Leftrightarrow \tau(g) = \mathrm{id}$.
2. Es ist $g \cdot \tau(g)^{-1} \in H$, denn es gibt $h \in H$ mit $g = h \cdot \tau(g)$, also $g \cdot \tau(g)^{-1} = h$.

Algorithmus 3.4 Berechnung von τ

Sei $H = G_m$ und v_m der Schreier-Vektor für $m \in M$. Berechne $\tau(g)$:

1. $\mu := m^g$.
2. Finde über den Schreier-Vektor ein $x \in G$ mit $\mu = m^x$ (Algorithmus 2.14).
3. x ist ein Element der Transversalen von G_m in G, d.h. $\tau(g) = x$.

Bemerkung 3.5 Die Punkte der Bahn m^G entsprechen den Nebenklassen von G_m in G.

$$G_m t_1 = G_m t_2 \quad \Leftrightarrow \quad m^{t_1} = m^{t_2}.$$

Die Berechnung einer Transversalen entspricht also der Berechnung der Bahn (Algorithmus 2.11 mit Zusatzinformation).

Bemerkung 3.6 $g \cdot \tau(g)^{-1} = h$ ist die Projektion von g auf G_m. Damit erhalten wir eine Darstellung von $g \in G$ als Produkt eines Elements $h \in G_m$ und einem Produkt von Erzeugern von G:

$$g = \underbrace{g \cdot \tau(g)^{-1}}_{=h} \cdot \underbrace{\tau(g)}_{=s_{i_1} \cdots s_{i_l}}$$

Die Darstellung von $\tau(g)$ durch die Erzeuger s_{i_j} liefert Algorithmus 2.14. Die Idee ist nun, dieses Vorgehen rekursiv für h auf G_m fortzusetzen, um eine vollständige Faktorisierung von g zu erhalten.

Problem: Bestimme Erzeuger für G_m.

Satz 3.7 (Satz von Schreier)
Seien $G = \langle S \rangle$, H eine Untergruppe von G, T eine (Rechts-)Transversale von H in G (also $\mathrm{id} \in T$) und $\tau : G \to T$ die kanonische Projektion. Dann erzeugt die Menge

$$S_1 := \{t \cdot s \cdot \tau(t \cdot s)^{-1} : t \in T, s \in S\}$$

die Untergruppe H. Es ist $|S_1| \leq |T| \cdot |S|$.

BEWEIS: Die Abbildung $g \mapsto g \cdot \tau(g)^{-1}$ liefert nur Elemente in H, d.h. $S_1 \subseteq H$ und damit $\langle S_1 \rangle \leq H$.

Nun sei $h \in H$ beliebig, $h = s_1 \cdots s_k$ für gewisse $s_1, ..., s_k \in S$, die nicht notwendigerweise paarweise verschieden sind. (Bei unendlichen Gruppen sei S bzgl. Inversion abgeschlossen, d.h. mit s ist auch s^{-1} in S, um zu garantieren, dass jedes Element $h \in H$ (bzw. $g \in G$) eine endliche Darstellung besitzt.)

Ziel: Stelle h als Produkt von Elementen aus S_1 dar. Setze

$$\begin{aligned} y_1 &:= s_1 (= \mathrm{id} \cdot s_1) \\ y_j &:= \underbrace{\tau(y_{j-1})}_{\in T} \cdot \underbrace{s_j}_{\in S} \qquad (j = 2, ..., k) \end{aligned}$$

Nach Definition von S_1 ist $y_j \cdot \tau(y_j)^{-1} \in S_1$. Also:

$$\begin{aligned} h &= s_1 \cdot (\tau(y_1)^{-1} \cdot \tau(y_1)) \cdot s_2 \cdot (\tau(y_2)^{-1} \cdot \tau(y_2)) \cdot s_3 \cdots \\ &= s_1 \cdot \tau(y_1)^{-1} \cdot (\tau(y_1) \cdot s_2) \cdot \tau(y_2)^{-1} \cdot (\tau(y_2) \cdot s_3) \cdots \\ &\underset{\text{Def.}}{=} \underbrace{y_1 \cdot \tau(y_1)^{-1}}_{\in S_1} \cdot \underbrace{y_2 \cdot \tau(y_2)^{-1}}_{\in S_1} \cdots y_k \end{aligned}$$

Dies ist fast ein Produkt von Elementen aus S_1; der letzte Faktor ist y_k. Sei $a := h \cdot \tau(y_k)^{-1}$. a ist Produkt von Elementen aus $S_1 \Rightarrow a \in H$. Mit $h \in H$ folgt:

$$\begin{aligned} & \tau(y_k) = a^{-1} \cdot h \in H \\ \Rightarrow\ & \tau(y_k) = \mathrm{id} \\ \Rightarrow\ & a = h. \end{aligned}$$

Somit lässt sich h als Produkt von Elementen aus S_1 darstellen. ■

Bemerkung 3.8 Der Satz von Schreier liefert eine Möglichkeit, sogenannte **Schreier-Generatoren** für die Untergruppe H zu bestimmen. Insbesondere kann man nun rekursiv Erzeuger für G_m bestimmen und die Bahn eines Punktes m_2 unter G_m berechnen.

4 Stabilisatorketten

Um die Idee aus Bemerkung 3.6 umzusetzen, brauchen wir eine „geeignete“ Folge von Punkten aus M.

Definition 4.1 Eine Teilmenge $B = \{\beta_1, ..., \beta_k\} \subseteq M$ heißt **Basis** der Gruppe G bzgl. der Operation von G auf M, falls der **punktweise Stabilisator**

$$G_{\beta_1,...,\beta_k} = \{g \in G : \forall \beta_i \in B : \beta_i^g = \beta_i\}$$

aller Elemente $\beta_i \in B$ trivial ist. Die Basis sei angeordnet, schreibe $B = (\beta_1, ..., \beta_k)$.

Definition 4.2 Die **Stabilisatorkette** zu einer Basis $B = (\beta_1, ..., \beta_k)$ ist die Kette der Untergruppen

$$G \geq G_{\beta_1} \geq G_{\beta_1,\beta_2} \geq ... \geq G_{\beta_1,...,\beta_{k-1}} \geq G_{\beta_1,...,\beta_k} = \{\mathrm{id}\}.$$

Bemerkung 4.3

1. Die symmetrische Gruppe S_n hat die Basis $B = (1, ..., n)$.
2. Ein Element β_j mit Bahnlänge 1 bzgl. $G_{\beta_1,...,\beta_{j-1}}$ heißt **redundant** und liefert keine echte Untergruppe in der Stabilisatorkette. Also ist $B \backslash \{\beta_j\}$ auch eine Basis.
3. Ist die Basis minimal in dem Sinne, dass sie keine redundanten Elemente enthält, so gilt:
   $$|B| = \mathcal{O}(\log |G|),$$
   denn jede echte Untergruppe hat mindestens den Index 2 (also höchstens halb soviele Elemente wie G).

Definition 4.4 Sei $B = (\beta_1, ..., \beta_k)$ eine Basis der Gruppe G. Eine Menge S von Erzeugern von G heißt **starke Erzeugermenge** bzgl. der Basis B (engl.: *(base) strong generating set*, kurz (B)SGS), wenn gilt:

1. $\langle S \rangle = G$.
2. $G_{\beta_1,...,\beta_j} = \langle S \cap G_{\beta_1,...,\beta_j} \rangle$.

S enthält also Erzeuger für jede Untergruppe der Stabilisatorkette.

Bemerkung 4.5 $S \cap G_{\beta_1,...,\beta_j}$ ist einfach zu berechnen: Ein $s_i \in S$ liegt genau dann im Schnitt, wenn es $\beta_1, ..., \beta_j$ fix lässt.

Beispiel 4.6 $G = \mathsf{S}_4$ mit Basis $B = (1, 2, 3)$. Sei $S = \{s_1 := (1\ 2\ 3\ 4), s_2 := (3\ 4)\}$:

$$1 \xrightarrow{s_1} 2 \xrightarrow{s_1} 3 \xrightarrow{s_1} 4$$

$S \cap G_1 = \{(3\ 4)\}, G_1 = \langle (2\ 3\ 4), (3\ 4)\rangle \neq \langle S \cap G_1\rangle$, also ist S keine starke Erzeugermenge, aber $\tilde{S} = S \cup \{(2\ 3\ 4)\}$ ist eine.

$$\begin{aligned} & G > G_1 > G_{1,2} > G_{1,2,3} = \{\mathrm{id}\} \\ \hat{=}\ & \mathsf{S}_4 > \mathsf{S}_3 > \mathsf{S}_2 > \{\mathrm{id}\} \end{aligned}$$

Ziel: Nutze die Stabilisatorketten zur Lösung unserer Grundprobleme: Bestimmung der Gruppenordnung $|G|$, Test auf Enthaltensein in G, Faktorisierung von Gruppenelementen.

Uns steht bereits Folgendes zur Verfügung:

- Codierung der Nebenklassen von G_{β_1} in G über die Bahn von β_1 unter G. Dies führt zum Schreier-Vektor (Algorithmus 2.11).
- Berechnung eines Repräsentanten $\tau(g)$ jeder Nebenklasse (Algorithmus 3.4).
- Der Satz von Schreier zeigt, dass man Erzeuger für die Untergruppe G_{β_1} von der Form $t \cdot s \cdot \tau(t \cdot s)^{-1}$ bestimmen kann.

Daraus erhalten wir einen iterativen/rekursiven Algorithmus: ersetze G durch G_{β_1}, dann G_{β_1} durch G_{β_1,β_2} usw.

Bemerkung 4.7 Sei $T^{(i)}$ die Transversale von $G_{\beta_1,\ldots,\beta_i}$ in $G_{\beta_1,\ldots,\beta_{i-1}}$. Damit können wir die Gruppenordnung von G angeben:

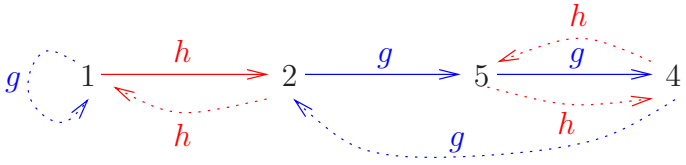
$$\begin{aligned} |G| &= |T^{(1)}| \cdot |G_{\beta_1}| \\ &= |T^{(1)}| \cdot (|T^{(2)}| \cdot |G_{\beta_2}|) \\ &\vdots \\ &= |T^{(1)}| \cdots |T^{(k)}|. \end{aligned}$$

Beispiel 4.8 $S = \{g, h\}$ mit $g = (2\ 5\ 4), h = (1\ 2)(4\ 5)$.

Fragestellung: Welche Ordnung hat $G = \langle S\rangle$?

Vorgehen:

1. Wähle einen Punkt, etwa $\beta_1 := 1$.
2. Berechne die Bahn von β_1 unter $\langle S\rangle$.



$\Rightarrow$ Bahnlänge: $|1^G| = 4$.
$\Rightarrow$ Bahnbilanz: $|G| = 4 \cdot |G_1|$.

Der Schreier-Vektor des Punktes 1:

	1	2	3	4	5
	id	h	$\perp$	g	g
Pfad	id	h	$\perp$	hg^2	hg

$\Rightarrow$ Transversale $T^{(1)} = \{\mathrm{id}, h, hg, hg^2\}$ von G_1.

3. Bestimme mit dem Satz von Schreier die Erzeuger von G_1:
Elemente der Form $t \cdot s \cdot \tau(t \cdot s)^{-1}$ mit $s \in S = \{g, h\}, t \in T^{(1)} = \{\mathrm{id}, h, hg, hg^2\}$:

$x = t \cdot s$	g	hg	hg^2	hg^3	h	h^2	hgh	hg^2h
1^x	1	5	4	2	2	1	4	5
$\tau(x)$	id	hg	hg^2	h	h	id	hg^2	hg
$x \cdot \tau(x)^{-1}$	g $(2\ 5\ 4)$	id -	id -	id -	id -	id -	$hghg^{-2}h^{-1}$ $(2\ 4\ 5)$	$hg^2hg^{-1}h^{-1}$ $(2\ 5\ 4)$

$\Rightarrow$ Erzeuger für G_1 sind $a = (2\ 5\ 4), b = (2\ 4\ 5)$.

4. Wähle neuen Punkt, etwa $\beta_2 := 2$.

5. Bahn der 2:

$$5 \xleftarrow{\ a\ } 2 \xrightarrow{\ b\ } 4$$

$\Rightarrow$ Bahnlänge: $|2^{G_1}| = 3$.
$\Rightarrow$ Bahnbilanz: $|G_1| = 3 \cdot |G_{1,2}| \Rightarrow |G| = 4 \cdot 3 \cdot |G_{1,2}|$.

6. $G_{1,2} = \{\mathrm{id}\}$.

Bemerkung 4.9 Die Schreier-Generatoren der Form $t \cdot s \cdot \tau(t \cdot s)^{-1}$ können entweder rein symbolisch (d.h. als Elemente der Wortgruppe) oder auch explizit vorliegen (weitere Multiplikationen in der Gruppe, aber weniger Erzeuger).

Bemerkung 4.10 Anzahl der Erzeuger:

$$\begin{array}{ccccccc} G & \geq & G_{\beta_1} & \geq & G_{\beta_1,\beta_2} & \cdots \\ |S| & & |S| \cdot |T^{(1)}| & & (|S| \cdot |T^{(1)}|) \cdot |T^{(2)}| & \cdots \end{array}$$

Also erhält man eine obere Schranke für die Anzahl der Erzeuger eines Stabilisators:

$$|S| \cdot (\text{Produkt der Längen der Transversalen}) \leq |S| \cdot |G|.$$

Dies ist linear in der Gruppenordnung, aber die Abschätzung ist viel zu grob!

Bemerkung 4.11 Wortlänge der Erzeuger bzw. Schreier-Generatoren (d.h. die Anzahl der Faktoren bei der Darstellung als Produkt von Erzeugern der Gruppe G):
Schreier-Generatoren haben die Form $t \cdot s \cdot \tau(t \cdot s)^{-1} = t \cdot s \cdot \tilde{t}$. Folglich verdreifacht sich die Länge beim Übergang zu G_{β_1} (falls keine Relationen berücksichtigt werden; vgl. Beispiel 2.7).

Problem: Die Länge der Schreier-Generatoren kann exponentiell anwachsen!
Ausblick: Heuristiken zur Wahl der Basis und zur Wahl von Erzeugern helfen, dieses Problem zu vermeiden (mehr dazu in Kapitel 7).

Satz 4.12 Sei $T^{(i)}$ die Transversale von $G_{\beta_1,\ldots,\beta_i}$ in $G_{\beta_1,\ldots,\beta_{i-1}}$ mit der zugehörigen Projektion $\tau_i : G_{\beta_1,\ldots,\beta_{i-1}} \to T^{(i)}$. Dann lässt sich jedes $g \in G$ darstellen als

$$g = t_k \cdot t_{k-1} \cdots t_2 \cdot t_1$$

mit $t_i \in T^{(i)}$.

BEWEIS: Setze $g_0 := g$, $g_i := g_{i-1} \cdot \tau_i(g_{i-1})^{-1} \in G_{\beta_1,\ldots,\beta_i}$ und $t_i := \tau_i(g_{i-1}) \in T^{(i)}$. Wendet man nun Bemerkung 3.6 rekursiv auf jede Stufe i an, so folgt die Behauptung. ■

Algorithmus 4.13 Faktorisierung

Sind für alle i die Erzeuger von $G_{\beta_1,\ldots,\beta_i}$ bekannt (etwa wenn S eine starke Erzeugermenge ist), so kann man die t_i aus Satz 4.12 mit Algorithmus 3.4 bestimmen und mit Hilfe des Schreier-Vektors (bzw. Algorithmus 2.14) als Worte von Erzeugern darstellen.

Eingabe: $g \in G$.
Ausgabe: Faktorisierung von g als Wort in den Erzeugern.

$w := \mathrm{id}$
$h := g$
for $i = 1$ **to** k **do**
    $t := \tau_i(h)$ *// als Wort von Erzeugern aus S aufgefasst*
    $w := t \cdot w$ *// neues Wort t_i anhängen*
    $h := h \cdot t^{-1}$
return w

Beispiel 4.14 Anwendung der Faktorisierung bei der Berechnung von Gruppenhomomorphismen: Ein Gruppenhomomorphismus Φ wird eindeutig festgelegt durch die Bilder $\Phi(s_i)$ der Erzeuger s_i:

$$\Phi(g) = \Phi(s_1 \cdots s_n) = \Phi(s_1) \cdots \Phi(s_n).$$

Bemerkung 4.15 Satz 4.12 liefert ein Kriterium zu testen, ob g überhaupt in der Gruppe enthalten ist. Ist dies nämlich nicht der Fall, so können wir g in der Form

$$g = h \cdot t_j \cdots t_1$$

schreiben, wobei $\beta_{j+1}^h \notin \beta_{j+1}^G$.

Algorithmus 4.16 Membership Test und **Word Strip**

Es sei H eine Untergruppe der Permutationsgruppe G. Sei $B = (\beta_i, ..., \beta_l)$ eine Basis von H mit starker Erzeugermenge, $\tau_j : H_{\beta_i,...,\beta_{j-1}} \to T^{(j)}$ die Projektion in die Transversale $T^{(j)}$ von $H_{\beta_i,...,\beta_j}$ in $H_{\beta_i,...,\beta_{j-1}}$ und $O^{(j)}$ bezeichne die Bahn von β_j unter $H_{\beta_i,...,\beta_{j-1}}$.

Eingabe: Eine Permutation $g \in G$; H; i.
Ausgabe: Antwort, ob $g \in H$; Stufe $j \in \mathbb{N}$, auf der der Algorithmus terminiert hat.
(für Word Strip: $h =$ ziehe Transversalelemente von g ab; $j \in \mathbb{N}$)

$h := g$
$j := i$ *// in der Regel wird $i = 1$ sein*
repeat until $j > l$ **or** $\beta \notin O^{(j)}$
  $\beta := \beta_j^h$ *// Bild von β_j unter h*
  if $\beta \in O^{(j)}$ **then** *// β liegt in der Bahn von β_j unter $H_{\beta_i,...,\beta_{j-1}}$*
    $h := h \cdot \tau_j(h)^{-1}$ *// Projektion auf neues h im Stabilisator von β_j*
    $j := j + 1$
return $(h \stackrel{?}{=} \mathrm{id}), j$
(Word Strip: **return** h, j)

Membership Test bei partieller Datenstruktur

Ausgangssituation: Die Menge $B = (\beta_1, ..., \beta_l)$ ist nicht unbedingt eine Basis, und die Bahnen $O^{(j)}$ sind nicht unbedingt vollständig.

Test eines Elements g, von dem bekannt ist, dass es in der Gruppe G liegt:

1. Fall: Auf einer Stufe j liegt der Punkt β nicht in der „Bahn“ $\Rightarrow$ neues Element in der Bahn $O^{(j)}$ und neuer Nebenklassenrepräsentant in $T^{(j)}$ benötigt.

2. Fall: $g \in G_{\beta_1,...,\beta_l}$, aber am Ende des Membership Test: $h \neq \mathrm{id}$.
$\Rightarrow G_{\beta_1,...,\beta_l} \gneq \{\mathrm{id}\}$
$\Rightarrow (\beta_1, ..., \beta_l)$ ist keine Basis
$\Rightarrow$ neuer Basispunkt β_{l+1} benötigt.

Bemerkung 4.17 Diese Datenstruktur ermöglicht eine effiziente Codierung von Gruppenelementen als Vektor der Bilder der Basiselemente:

$$(\beta_1^{t_1}, \beta_2^{t_2 t_1}, ..., \beta_l^{t_l \cdots t_1}),$$

mit den t_i aus Satz 4.12. Diese Darstellung ermöglicht ein gleichverteiltes Ziehen von Gruppenelementen: Wähle einen Punkt aus jeder Bahn und bestimme das Gruppenelement.

Ist etwa $l = 2$ und sind $\gamma \in O^{(1)}$, $\delta \in O^{(2)}$ zufällig gezogene Punkte der Bahnen, so kann man das entsprechende Gruppenelement wie folgt ermitteln:
Löse $\gamma = \beta_1^{t_1}$ nach t_1 (Algorithmus 2.14). Dann löse $\delta^{t_1^{-1}} = \beta_2^{t_2}$ nach t_2. Das gesuchte Gruppenelement ist dann $g = t_2 t_1$.

5 Der Schreier-Sims-Algorithmus

Ziel: Berechnung einer Basis und einer Menge von starken Erzeugern sowie der zugehörigen Bahnen und Transversalen.

Lemma 5.1 Sei G eine Gruppe, $B = (\beta_1, ..., \beta_k)$ eine „potentielle Basis“ und $S \subset G$, $\mathrm{id} \notin S$. Setze $S^{(0)} := S, S^{(j)} := S \cap G_{\beta_1,...,\beta_j}$. Falls gilt

1. $G = \langle S \rangle$,
2. $S^{(k)} = \emptyset$,
3. $\langle S^{(j-1)} \rangle_{\beta_j} = \langle S^{(j)} \rangle$ für alle j,

so ist B eine Basis und S eine Menge von starken Erzeugern bzgl. B.

BEWEIS: Induktion über j:

$j = 1$: $G_{\beta_1} \overset{1.}{=} \langle S \rangle_{\beta_1} \overset{3.}{=} \langle S^{(1)} \rangle = \langle S \cap G_{\beta_1} \rangle$.

Induktionsannahme: $G_{\beta_1,...,\beta_j} = \langle S \cap G_{\beta_1,...,\beta_j} \rangle$.

$j \rightsquigarrow j+1$: Es gilt

$$\begin{aligned} G_{\beta_1,...,\beta_j,\beta_{j+1}} &= (G_{\beta_1,...,\beta_j})_{\beta_{j+1}} \\ &\overset{(*)}{=} \langle S \cap G_{\beta_1,...,\beta_j} \rangle_{\beta_{j+1}} \\ &= \langle S^{(j)} \rangle_{\beta_{j+1}} \\ &\overset{3.}{=} \langle S^{(j+1)} \rangle \\ &= \langle S \cap G_{\beta_1,...,\beta_{j+1}} \rangle, \end{aligned}$$

wobei die Gleichheit bei $(*)$ wegen der Induktionsannahme gilt. ■

Schema:

$$\begin{array}{ccccccc} G & & = & & H^{(0)} & := & \langle S \rangle \\ \vee & & & & \vee & & \\ G_{\beta_1} & \geq & H^{(0)}_{\beta_1} & \geq & H^{(1)} & := & \langle S^{(1)} \rangle \\ \vee & & \vee & & \vee & & \\ G_{\beta_1,\beta_2} & \overset{(*)}{\geq} & H^{(1)}_{\beta_2} & \overset{(**)}{\geq} & H^{(2)} & := & \langle S^{(2)} \rangle \\ \vdots & & \vdots & & \vdots & & \end{array}$$

Grob gesprochen besagt Lemma 5.1, dass aus Gleichheit bei $(**)$ auf jeder Stufe i auch die Gleichheit bei $(*)$ auf jeder Stufe i folgt. Angenommen, auf einer Stufe i der Stabilisatorkette ist $S^{(i)}$ bereits eine starke Erzeugermenge für $H^{(i)}$. Die Idee ist nun, solange weitere Schreier-Generatoren zu $H^{(i)}$ hinzuzufügen, bis $H^{(i-1)}_{\beta_i} = H^{(i)}$ gilt. Da es dabei passieren kann, dass ein neu hinzugefügter Schreier-Generator alle Basispunkte fix lässt,

müssen ggf. weitere Punkte zur Basis hinzugenommen werden (und die Datenstrukturen auf den Stufen $> i$ müssen aktualisiert werden).

Nach diesem Schema arbeitet man sich vom unteren Ende der Stabilisatorkette nach oben, bis man eine vollständige Basis mit starker Erzeugermenge für $G = H^{(0)}$ gefunden hat.

Algorithmus 5.2 Schreier-Sims-Algorithmus

Es sei $H^{(j)} = \langle S^{(j)} \rangle$ und $T^{(j)}$ die Transversale von $H_{\beta_j}^{(j-1)}$ in $H^{(j-1)}$ mit zugehöriger Projektion $\tau_j : H^{(j-1)} \to T^{(j)}$. Außerdem sei $O^{(j)}$ die Bahn von β_j unter $H^{(j-1)}$.

Der Algorithmus startet unter der *Induktionsannahme*, dass B und S für $H^{(i)}$ bereits eine Basis bzw. starke Erzeugermenge sind. Er führt den *Induktionsschritt* durch, so dass bei Terminierung B und S auch für $H^{(i-1)}$ eine Basis bzw. starke Erzeugermenge sind.

Eingabe: $B = (\beta_i, ..., \beta_l)$, S, i: Basis und starke Erzeugermenge für $H^{(i)}$.
Ausgabe: B, S: Basis und starke Erzeugermenge für $H^{(i-1)}$.

for all $t \in T^{(i)}$ **do**
  for all $s \in S^{(i-1)}$ **do**
    $g := t \cdot s \cdot \tau_i(t \cdot s)^{-1}$ *// neuer Schreier-Generator g aus $H_{\beta_i}^{(i-1)}$*
    $(\mathsf{flag}, \mu) := \mathrm{MembershipTest}(g, H^{(i)}, i+1)$ *// Algorithmus 4.16: ist $g \in H^{(i)}$?*
    if not flag **then**
      bestimme maximales ν mit $\beta_k^g = \beta_k$ für alle $k = 1, ..., \nu$
      $S := S \cup \{g, g^{-1}\}$ *// g ist neuer starker Erzeuger in $S^{(i)}, S^{(i+1)}, ..., S^{(\nu)}$*
      if $\nu = l$ **then**
        finde neuen Basispunkt β_{l+1} mit $\beta_{l+1}^g \neq \beta_{l+1}$
        $B := B \cup \{\beta_{l+1}\}$
        $l := l + 1$
      // die Bahnen müssen ergänzt werden
      for $j := \mu$ **to** l **do** aktualisiere den Schreier-Vektor von β_μ
      // stelle die Induktionsvoraussetzungen für die neue Basis wieder her
      for $j := \nu + 1$ **to** i **by** -1 **do** $(B, S) := \mathrm{SchreierSims}(B, S, \mu)$
return B, S

Um für G eine starke Erzeugermenge zu erhalten, wird der Schreier-Sims-Algorithmus wie folgt aufgerufen:

suche Punkte $\beta_1, ..., \beta_l$ mit $S \cap G_{\beta_1,...,\beta_l} = \emptyset$
$B := (\beta_1, ..., \beta_l)$
for $i := l$ **to** 1 **by** -1 **do** $(B, S) := \mathrm{SchreierSims}(B, S, i)$

Analyse:

Korrektheit des Algorithmus: Folgt direkt aus Lemma 5.1, da auf jeder Stufe ein Membership Test für alle Schreier-Generatoren durchgeführt wird und damit bei Terminierung

jeder Erzeuger von $\langle S^{(j-1)} \rangle_{\beta_j}$ sicher in $\langle S^{(j)} \rangle$ enthalten ist. Dies bedeutet nichts anderes, als dass die 3. Bedingung des Lemmas erfüllt ist: $\langle S^{(j-1)} \rangle_{\beta_j} = \langle S^{(j)} \rangle$. Bedingung 1. ist trivialerweise erfüllt, da der Algorithmus auf die von S erzeugte Gruppe angewandt wird. Die 2. Bedingung ist für Permutationsgruppen immer erfüllbar, da man im Schreier-Sims-Algorithmus immer einen neuen Basispunkt β_{l+1} mit $\beta_{l+1}^g \neq \beta_{l+1}$ finden kann. Allgemeiner gilt für eine beliebige Operation einer Gruppe G auf einer Menge M: Die 2. Bedingung ist nur erfüllbar, wenn die Operation treu ist.

Terminierung: Für Permutationsgruppen ist $G \leq \mathsf{S}_n$ und damit endlich. Spätestens bei $S = G$ liefert der Membership Test keine neuen Elemente. Allgemeiner:

- Falls G eine endliche Gruppe ist, so terminiert der Algorithmus immer.
- Falls M endlich ist, so wird durch die Gruppenoperation auf M eine Permutationsdarstellung von G definiert, also ein Homomorphismus von G in die symmetrische Gruppe $\mathsf{S}_{|M|}$: Definiere $\Phi(g) \in \mathsf{S}_n$ durch

  $$M = \{m_1, ..., m_n\}, g * m_i = m_j =: m_{\Phi(g)(i)}.$$

  Folglich terminiert der Algorithmus und liefert Informationen über G modulo $G_{\beta_1,...,\beta_l}$.

Aufwand:

Erzeuger: Erinnerung: Die Anzahl der Schreier-Generatoren ist $|T^{(j)}| \cdot |S^{(j-1)}| \geq |S^{(j)}|$, also potentiell exponentielles Wachstum der Anzahl der Schreier-Generatoren. Aber: Es werden nur die wirklich benötigten Schreier-Generatoren in die Menge S aufgenommen. Folglich:

$$|S^{(j)}| = \mathcal{O}(|S^{(0)}| + \log |G|).$$

Dies gilt nur, wenn man auch bei der initialen Menge $S = S^{(0)}$ nur die „benötigten“ Erzeuger berücksichtigt.

Anzahl der Basispunkte: Jede Gruppe in der Stabilisatorkette ist eine echte Untergruppe. Folglich ist die Länge der Stabilisatorkette gleich der Anzahl der Basispunkte $(+1)$, also nach Bemerkung 4.3

$$\mathcal{O}(\log |G|).$$

Aktualisierung der Bahn: Wende dafür jeden Erzeuger auf jeden Punkt der Bahn an. Wende dann jeden Erzeuger auf die neuen Punkte an. Das ergibt

$$\begin{aligned} |O^{(j)}| &\leq |M| = n \\ \Rightarrow n \cdot &\text{ Anzahl Erzeuger} \cdot \text{Anzahl Bahnen} \\ &= \mathcal{O}(n(\log |G|)^2 + n|S^{(0)}| \log |G|) \end{aligned}$$

Operationen von G auf M.
Speziell für Permutationsgruppen: Speichere das Bild eines Punktes unter der Permutation:

$$\begin{aligned} \Rightarrow &\mathcal{O}(n \log n) \text{ Speicher}, \\ &\mathcal{O}(1) \text{ Zeit}. \end{aligned}$$

Test der Schreier-Generatoren: Es ist $|T^{(i)}| \leq n$. Damit kann man die Anzahl der Tests abschätzen:

$$\sum_j |T^{(j)}| \cdot |S^{(j-1)}| = \mathcal{O}(n(\log |G|)^2 + n|S^{(0)}| \log |G|).$$

Membership Test: Zwei Varianten zur Berechnung des Nebenklassenvertreters $\tau_j(h)$ (vgl. Algorithmus 2.14):

1. Über den Schreier-Vektor: maximal $|O^{(j)}|$ viele Multiplikationen von Gruppenelementen (falls der Schreier-Baum eine lineare Kette ist, wie etwa bei zyklischen Gruppen). Das ergibt $\mathcal{O}(n)$ viele Gruppenmultiplikationen (bei Permutationsgruppen $\mathcal{O}(n^2)$ viele Elementaroperationen, da die Multiplikation in $\mathcal{O}(n)$ liegt). Für alle Schleifendurchläufe ergibt sich damit ein Gesamtaufwand von $\mathcal{O}(n \log |G|)$ Gruppenoperationen (bzw. $\mathcal{O}(n^2 \log |G|)$ Elementaroperationen in Permutationsgruppen).
2. Über Speicherung des kompletten Pfades: $\mathcal{O}(1)$, aber der Speicherbedarf wächst um den Faktor n. Für alle Schleifendurchläufe ergibt sich damit ein Gesamtaufwand von $\mathcal{O}(\log |G|)$ Gruppenoperationen (bzw. $\mathcal{O}(n \log |G|)$ Elementaroperationen).

Mit diesen zwei Varianten ergibt sich der Gesamtaufwand des Algorithmus:

Satz 5.3 Der Schreier-Sims-Algorithmus hat folgenden Gesamtaufwand für Laufzeit und Speicherbedarf bei

1. Berechnung von τ_j über den Schreier-Vektor:

$$\begin{aligned} \text{Zeit: } & \mathcal{O}(n^3(\log |G|)^3 + n^3|S^{(0)}|(\log |G|)^2) \\ \text{Speicher: } & \mathcal{O}(n(\log |G|)^2 + n|S^{(0)}| \log |G|) \end{aligned}$$

2. Berechnung von τ_j über den gespeicherten Pfad:

$$\begin{aligned} \text{Zeit: } & \mathcal{O}(n^2(\log |G|)^3 + n^2|S^{(0)}|(\log |G|)^2) \\ \text{Speicher: } & \mathcal{O}(n^2(\log |G|)^2 + n^2|S^{(0)}| \log |G|) \end{aligned}$$

6 Basiswechsel

Ausgangssituation: Sei G eine Permutationsgruppe mit starkem Erzeugendensystem S bzgl. der Basis B.

Ziel: Das Lösen von Gleichungen der folgenden Formen:

- Für $a, b \in M$ finde $g \in G$ mit $a = b^g$.
- Simultane Gleichungen: zu $\boldsymbol{a} = (a_1, ..., a_l), \boldsymbol{b} = (b_1, ..., b_l)$ mit $a_i, b_i \in M$ finde $g \in G$ mit $\boldsymbol{a} = \boldsymbol{b}^g$, d.h. $a_i = b_i^g$ für $i = 1, ..., l$.
  Beachte: $\boldsymbol{a}, \boldsymbol{b}$ sind geordnete Sequenzen und keine Mengen.

Bemerkung 6.1 Ein besonders einfacher Fall ist gegeben, wenn $b = \beta_1$ ist, wobei β_1 der erste Basispunkt aus B sei. Dann gilt nämlich:

$$[\exists g \in G : a = b^g] \Leftrightarrow \left[a \in \beta_1^G = O^{(1)}\right]$$

Eine mögliche Lösung ist durch ein Element der Transversalen $T^{(1)}$ gegeben und wird ggf. durch Algorithmus 2.14 geliefert.

Bemerkung 6.2 Die Lösungen der Gleichung $a = b^g$ sind von der Form $a = hg_0$ mit $h \in G_b$ und $a = b^{g_0}$. Also sind die Elemente der Nebenklasse $G_b g_0$ Lösungen.

Bemerkung 6.3 Der einfachste Fall der simultanen Gleichungen ist gegeben, wenn $\boldsymbol{b} = (\beta_1, ..., \beta_l)$ ein Anfangsstück der Basis B ist:

1. Löse $a_1 = \beta_1^{g_1}$ mit $g_1 \in G$.
2. Löse $a_2 = \beta_2^{g'}$ mit $g' = g_2 g_1 \in G_{\beta_1} g_1$.
   $[a_2 = \beta_2^{g_2 g_1}$ mit $g_2 \in G_{\beta_1}] \Leftrightarrow [a_2^{g_1^{-1}} = \tilde{a}_2 = \beta_2^{g_2}$ mit $g_2 \in G_{\beta_1}]$, die Lösung kann also mit Hilfe der Transversalen $T^{(2)}$ von G_{β_1,β_2} in G_{β_1} bestimmt werden.

Algorithmus 6.4 Schema des Algorithmus zur Lösung von $(a_1, ..., a_l) = (\beta_1^g, ..., \beta_l^g)$.

1. Bestimme eine Lösung $g_1 \in G$ mit $a_1 = \beta_1^{g_1}$.
2. Löse die Gleichungen

   $$(a_2^{g_1^{-1}}, ..., a_l^{g_1^{-1}}) = (\beta_2^{g_2}, ..., \beta_l^{g_2})$$

   in der Gruppe G_{β_1}.
3. Die Lösung ist $g = g_2 g_1$.

Problem: Wie löst man $\boldsymbol{a} = \boldsymbol{b}^g$, falls $\boldsymbol{b}$ nicht das Anfangsstück der bekannten Basis ist? Ähnliches Problem: Wie berechnet man den Stabilisator eines Punktes $b \in M$ bzw. einer Sequenz $\boldsymbol{b} = (b_1, ..., b_l)$? (Die Lösung ist einfach, wenn $\boldsymbol{b}$ ein Anfangsstück der Basis B ist: $G_{\boldsymbol{b}} = \langle S \cap G_{\boldsymbol{b}} \rangle$.)

Ziel: Verwende die gegebenen Informationen (bekannte Basis und starkes Erzeugendensystem), um eine neue Basis B' zu berechnen, die $\boldsymbol{b}$ als Anfangsstück hat. Dabei soll keine Neuberechnung von B' und S' mit dem Schreier-Sims-Algorithmus stattfinden.

Wir greifen auf die Gruppentheorie zurück:

Definition 6.5 Der **Normalisator** einer Permutationsgruppe $G < \mathsf{S}_n$ ist der Normalteiler

$$N(G) := N_{\mathsf{S}_n}(G) := \{h \in \mathsf{S}_n : \forall g \in G : h^{-1}gh \in G\}.$$

Wir verhält sich eine starke Erzeugermenge unter Konjugation?

Lemma 6.6 Sei S eine starke Erzeugermenge von G bzgl. der Basis B. Dann ist für $h \in N(G)$ auch

$$S^h = \{h^{-1}sh : s \in S\}$$

eine starke Erzeugermenge von G bzgl. der Basis $B^h = (\beta_1^h, ..., \beta_l^h)$.

BEWEIS: Skizze: Es sei $(\beta_i^h)^{s'_j} = (\beta_i^h)^{h^{-1}s_jh} = (\beta_i^{s_j})^h$. Dann folgt: Falls s_j den Punkt β_i stabilisiert, so stabilisiert s'_j den Punkt β_i^h. ■

Basiswechsel durch Konjugation: Finde ein $h \in N(G)$ mit $(\beta_1^h, ..., \beta_l^h) = (b_1, ..., b_l)$. Dies ist das ursprüngliche Problem, aber in der größeren Gruppe $N(G)$.

Bemerkung 6.7 Ein Ansatz zur Lösung von $\boldsymbol{a} = \boldsymbol{b}^g$:

1. Bestimme eine Lösung von $\boldsymbol{b} = \boldsymbol{\beta}^{g_1}$.
2. Bestimme eine Lösung von $\boldsymbol{a} = \boldsymbol{\beta}^{g_2}$.

Dann ist $\boldsymbol{a} = \boldsymbol{\beta}^{g_2} = \boldsymbol{b}^{g_1^{-1}g_2}$. Dieser Ansatz ist gut, falls immer eine Lösung existiert.

Bemerkung 6.8 Ein anderer Ansatz: *Basiswechsel durch Vertauschen der Basiselemente.* Transformiere direkt die Basis B und das starke Erzeugendensystem, so dass man eine neue Basis $B' = (b_1, ..., b_l, b_{l+1}, ..., b_k)$ erhält. Dann kann man die Gleichung $\boldsymbol{a} = \boldsymbol{b}^g$ wie in Algorithmus 6.4 lösen.

Ist $B = (\beta_1, ..., \beta_m)$ eine Basis, so auch $B' = (\beta_1, ..., \beta_m, b_1, ..., b_l)$.
Idee: Vertausche die Basiselemente, bis man

$$B'' = (b_1, ..., b_l, \beta_1, ..., \beta_m)$$

erhält. Es genügt, benachbarte Elemente der Basis zu vertauschen.

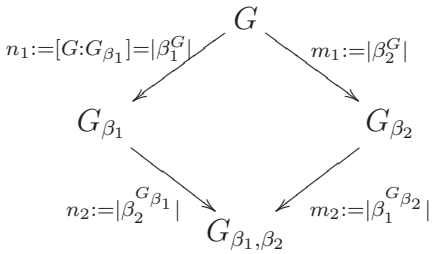
Transposition der Basiselemente:

$$B = (\beta_1, ...,\beta_i, \beta_{i+1}, ...)$$
$$\updownarrow$$
$$B' = (\beta_1, ...,\beta_{i+1}, \beta_i, ...)$$

In den Stabilisatorketten sieht das so aus:

$$G > G_{\beta_1} > ... > G_{\beta_1,...,\beta_i} > G_{\beta_1,...,\beta_i,\beta_{i+1}} > ...$$
$$\updownarrow \text{ (einzige Änderung!)}$$
$$G > G_{\beta_1} > ... > G_{\beta_1,...,\beta_{i-1},\beta_{i+1}} > G_{\beta_1,...,\beta_i,\beta_{i+1}} > ...$$

Ohne Einschränkung reicht es, den Fall $B = (\beta_1, \beta_2)$, $B' = (\beta_2, \beta_1)$ zu betrachten.



Wegen der Bahnbilanz gilt $m_2 = \frac{n_1 \cdot n_2}{m_1}$. Folgendes ist zu tun:

1. Berechne die Bahn von β_2 unter G (das ist einfach: Algorithmus 2.11).
2. Bestimme die Erzeuger von G_{β_2} und die Bahn von β_1 unter G_{β_2}. (Variante mit Schreier-Generatoren; Abbruch, falls die Bahnlänge $= m_2$.)

Bemerkung 6.9 Es ist $\beta_1^{G_{\beta_2}} \subseteq \beta_1^G$. Die Bahn von β_1 unter G zerfällt in disjunkte Bahnen unter $G_{\beta_2} \leq G$.

Algorithmus 6.10 Transposition von Basiselementen

Eingabe: Starke Erzeugermenge S bzgl. $B = (\beta_1, \beta_2)$, die Bahnen β_1^G und $\beta_2^{G_{\beta_1}}$.
Ausgabe: Starke Erzeugermenge S' bzgl. $B' = (\beta_2, \beta_1)$ und die Bahn $\beta_1^{G_{\beta_2}}$.

$S' := S \cap G_{\beta_1,\beta_2}$ *// S' soll ein starkes Erzeugendensystem für G_{β_2} werden*
$O^{(1)} := \{\beta_1\}$ *// $O^{(1)}$ soll die Bahn von β_1 unter G_{β_2} werden*
$C := \beta_1^G \backslash O^{(1)}$ *// enthält die Kandidaten für die Bahnelemente*
while $|O^{(1)}| < m_2$ **do**
  $\alpha :=$ Element aus C
  finde $u \in G$ mit $\beta_1^u = \alpha$
  $\gamma := \beta_2^{u^{-1}}$
  if $\gamma \in \beta_2^{G_{\beta_1}}$ **then**
    finde $w \in G_{\beta_1}$ mit $\beta_2^w = \gamma$
    $S' := S' \cup \{wu, (wu)^{-1}\}$
    aktualisiere die Bahn $O^{(1)} = \beta_1^{\langle S' \rangle}$
    $C := C \backslash O^{(1)}$
  else $C := C \backslash \alpha^{\langle S' \rangle}$

Analyse:

Korrektheit des Algorithmus: Sei $\alpha = \beta_1^u$ mit $u \in G$. Gesucht ist $x \in G$ mit

$$\beta_1^x = \alpha,\ \beta_2^x = \beta_2,$$

d.h. $x \in G_{\beta_2}$ (denn es soll ja geprüft werden, ob α in der Bahn von β_1 unter G_{β_2} liegt). Alle x mit $\beta_1^x = \alpha$ sind von der Form $x = wu$ mit $w \in G_{\beta_1}$, $\beta_1^u = \alpha, u \in G$.
Zu lösen ist also

$$\beta_2^x = \beta_2^{wu} \stackrel{!}{=} \beta_2,$$

was äquivalent ist zu

$$\beta_2^w = \beta_2^{u^{-1}} = \gamma$$

mit $w \in G_{\beta_1}$, u wie oben. Gibt es so ein w, so liegt α in der Bahn von β_1 unter G_{β_2} und $x = wu$ ist ein weiterer Erzeuger von G_{β_2}; gibt es so ein w nicht, so können die Elemente von $\alpha^{\langle S' \rangle}$ als Bahnelemente ausgeschlossen werden.

7 Kurze Faktorisierungen

Erinnern wir uns an die Anwendungsmöglichkeiten einer Basis B mit starker Erzeugermenge S:

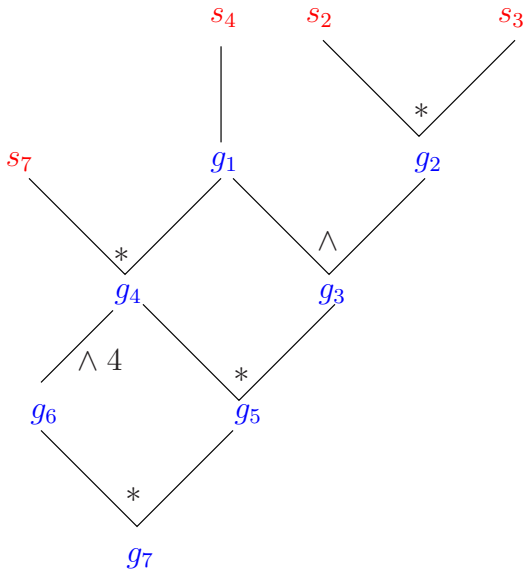
- Test, ob ein Element g in der von S erzeugten Gruppe liegt: Benötigt die Transversale bzw. zu jedem Punkt in der Bahn ein Gruppenelement, das den Basispunkt auf diesen Punkt abbildet (Algorithmus 4.16).
  Speichere die Transversalelemente explizit als (Permutations-)Gruppenelemente.

- Faktorisierung von Gruppenelementen als Produkt von Erzeugern (d.h. Elemente von S und ihre Inversen); etwa für die Auswertung von Homomorphismen, die eindeutig durch die Bilder der Erzeuger festgelegt sind.
  Speichere für jedes Element der Transversale eine Darstellung als Produkt von „bekannten“ Elementen.

Definition 7.1 Ein **straight line program**, kurz SLP, ist eine geordnete Sequenz $(g_1, g_2, ..., g_n)$, in der jedes g_i eines der folgenden Kriterien erfüllt:

- g_i ist ein Element der Erzeugermenge $\{s_1, ..., s_m\}$ der Gruppe.
- g_i ist ein Produkt der Form $g_i = g_j g_k$ mit $j, k < i$ oder von zwei Erzeugern.
- g_i ist eine Potenz $g_i = g_j^m$ mit $j < i, m \in \mathbb{Z}\backslash\{0\}$.
- g_i ist entstanden durch Konjugation, d.h. $g_i = g_j^{g_k} := g_k^{-1} g_j g_k$ mit $j, k < i$.

Beispiel 7.2 Ein SLP liefert einen Berechnungsgraphen für das Element g_n.



Als „Programm“ sieht der Berechnungsgraph so aus:
$g_1 := s_4, \quad g_2 := s_2 \cdot s_3,$
$g_3 := s_1^{s_2}, \quad g_4 := s_7 \cdot g_1,$
$g_5 := g_4 \cdot g_3, \quad g_6 := g_4^4, \quad g_7 := g_6 \cdot g_5$

Man verwendet nun für die Darstellung eines Elements nicht die Erzeugermenge S, sondern das SLP.

Speichere während des Schreier-Sims-Algorithmus zu jedem neuen starken Erzeuger und jedem Element der Transversalen, wie sie aus den vorhandenen Elementen berechnet werden, d.h. aktualisiere das „globale“ SLP.

Eine Faktorisierung eines Elements wählt eine Teilsequenz aus dem „globalen“ SLP (nur diejenigen Elemente, die wirklich benötigt werden).
Im Allgemeinen erhält man so eine effiziente Berechnungsvorschrift für das Bild unter einem Homomorphismus.

Aufwand (grobe Abschätzung): Im ungünstigsten Fall ist die Bahn eine lineare Kette der Länge n. Folglich ist das Transversalelement ein Produkt von höchstens n Elementen.
Insgesamt: maximal $\mathcal{O}(\log |G|)$ viele Stufen und $\mathcal{O}(n \log |G|)$ viele Multiplikationen.
Schreier-Generatoren der Form $ts\tau(ts)^{-1}$: Konstante Anzahl von Operationen.

Aber: Die Anzahl der Faktoren eines Schreier-Generators als Produkt von Erzeugern der Gruppe kann exponentiell mit der Anzahl der Stufen anwachsen.

Ziel: Finde eine kurze Faktorisierung mit Hilfe von Stabilisatorketten.

Bemerkung 7.3 Eine optimale Faktorisierung erhält man über den Cayley-Graphen.

Definition 7.4 Sei $G = \langle s_1, ..., s_m \rangle$ eine endliche Gruppe und $F = \langle f_1, ..., f_m \rangle$ eine freie Gruppe über den Erzeugern $f_1, ..., f_m$. Eine **Faktorisierung** von $g \in G$ ist ein Element $f \in F$ mit $\Phi(f) = g$, wobei Φ der kanonische Homomorphismus $\Phi : F \to G, f_i \mapsto s_i$, ist. Die **Länge** einer Faktorisierung von g ist die Wortlänge $|f|$, wobei gilt:

- $|f_i| = 1$.
- $|f_i^{-1}| = 1$ (nach Konvention).
- $|f\tilde{f}| = |f| + |\tilde{f}|$, falls sich „nichts kürzt“, d.h. falls $f = w_1 \cdots w_k$, $\tilde{f} = v_1 \cdots v_l$ und $w_k \neq v_1^{-1}$ (im Allgemeinen gilt nur $|f\tilde{f}| \leq |f| + |\tilde{f}|$).

Problem: Zu $g \in G = \langle s_1, ..., s_m \rangle$ finde das kürzeste $f \in F$ mit $\Phi(f) = g$.

Bemerkung 7.5 Zu einer festen Basis B existiert immer ein optimales Transversalensystem, d.h. die Länge der Faktorisierung aller Transversalelemente ist minimal.

BEWEIS: Die Basis B legt die Stabilisatorkette $G \geq G_{\beta_1} \geq ... \geq \{\mathrm{id}\}$ eindeutig fest. Zu jedem Element der Bahn gibt es ein Gruppenelement mit minimaler Faktorisierung. ■

Problem: Wie findet man ein Element $g \in G_{\beta_1,...,\beta_{j-1}}$ minimaler Länge mit $\beta_j^g = \gamma$?
Wie wählt man eine gute Basis?

Idee: Wähle die Basis so, dass möglichst kurze Worte in den Erzeugern viele Basispunkte festlassen, d.h. man hat per Konstruktion schon einige kurze Erzeuger für die Untergruppen in der Stabilisatorkette (siehe dazu auch Egner und Püschel [4]).

Die folgende Definition 7.6 ist bewusst vage gehalten.

Definition 7.6 Sei $G = \langle s_1, ..., s_m \rangle \leq \mathsf{S}_n$, $F = \langle f_1, ..., f_m \rangle$ eine freie Gruppe und $\Phi(f_i) = s_i$, der kanonische Homomorphismus.

1. Als **Paar** bezeichnen wir ein Tupel $(f, g) := (f, \Phi(f)) \in F \times G$.
2. Als **faules Paar** (f, g) bezeichnen wir Gruppenelemente f, g mit vielen Fixpunkten, d.h. es gibt nur wenige Elemente $m \in M = \{1, ..., n\}$, die von g bewegt werden.
3. Ein **kurzes Paar** ist ein Paar (f, g) mit kleiner Wortlänge $|f|$.
4. **Konjugierte Paare** sind Paare der Form $(\tilde{f}, \tilde{g}) = (f'^f, g'^g) = (f^{-1}f'f, g^{-1}g'g)$. Die Anzahl der Fixpunkte von $(\tilde{f}, \tilde{g})$ ist gleich der Anzahl der Fixpunkte von (f', g'), aber im Allgemeinen sind die Fixpunkte verschieden.

Definition 7.7 Wir definieren eine **Paarordnung**: Es sei $(f, g) < (\tilde{f}, \tilde{g})$ genau dann, wenn die Anzahl der Fixpunkte von g größer als die Anzahl der Fixpunkte von $\tilde{g}$ ist, oder, bei gleicher Anzahl, falls $|f| < |\tilde{f}|$.

Für ein Paar (f, g) identifizieren wir im Folgenden f und g miteinander.

Algorithmus 7.8 Wahl einer Basis

Sei $G \leq \mathsf{S}_n$ eine Permutationsgruppe, $\mathrm{Fix}(u)$ bezeichne die Menge der Fixpunkte von u.

1. Generiere eine Liste L von kurzen Paaren und dazu konjugierten Paaren, die bzgl. der Paarordnung $<$ geordnet ist (etwa über Breitensuche von id ausgehend).
2. Finde eine geeignete Partition $\boldsymbol{B}$ von $\{1, ..., n\}$:

   $\boldsymbol{B} := (\{1, ..., n\})$ *// i.A.* $\boldsymbol{B} = (B_1, ..., B_j)$ *mit* $B_i \subseteq \{1, ..., n\}$
   while $L \neq \emptyset$ **do**
   $u := \min_<\{v \in L\}$ *// das kürzeste Wort aus* L*, für das* $|\mathrm{Fix}(u) \cap B_1|$ *maximal ist*
   $\boldsymbol{B} := (B_1 \cap \mathrm{Fix}(u), B_1 \backslash \mathrm{Fix}(u), B_2, ..., B_j)$
   $L := \{v \in L : B_1 \not\subseteq \mathrm{Fix}(v)\}$
3. Verfeinere die Partition $\boldsymbol{B}$ zu einer geordneten Basis B durch Aneinanderhängen der Blöcke B_j:

$$B := (\underbrace{\beta_1, ..., \beta_i}_{B_1}, \underbrace{\beta_{i+1}, ..., \beta_{i+k}}_{B_2}, ...).$$

Die Idee hinter dieser Heuristik ist es, in die vorderen Blöcke der Partition diejenigen Punkte zu sortieren, die von vielen kurzen Paaren fest gelassen werden. Diese kurzen Paare verteilen sich daher über viele Stufen der Stabilisatorkette und können in den entsprechenden Untergruppen als Erzeuger wirken, so dass man nicht (nur) auf die immer länger werdenden Schreier-Generatoren angewiesen ist.

Als nächstes werden wir uns überlegen, wie man die Transversalen der Stabilisatorkette mit kurzen Worten vervollständigen kann.

Im ersten Schritt werde eine Basis so festgelegt, dass es zumindest einige kurze Paare gibt, die viele Basispunkte fix lassen. Die Basis legt die Stabilisatorkette fest:

$$G \geq G_{\beta_1} \geq \ldots \geq \underbrace{G_{\beta_1,\ldots,\beta_j}}_{\text{einige Erzeuger mit kurzer Faktorisierung}} \geq \{\mathrm{id}\}$$

Die Berechnung eines optimalen Transversalensystems kann z.B. mittels Breitensuche erfolgen. Aber der Aufwand ist zu groß!

Bemerkung 7.9

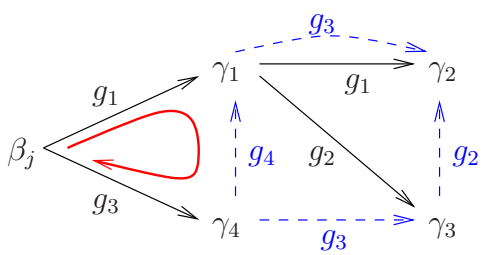
1. Oft kennt man die Gruppenordnung $|G|$ und hat damit ein Abbruchkriterium für die Breitensuche: Produkt der Bahnlängen $\stackrel{?}{=} |G|$.
2. Die Berechnung eines optimalen Transversalensystems ist zwar aufwendig, aber man muss nicht den kompletten Cayley-Graphen speichern, um das optimale Transversalensystem zur Faktorisierung zu verwenden.
   $\Rightarrow$ Speicheraufwand: $\mathcal{O}(p(\log |G|))$ mit einem Polynom p.

Es wurden für die Basiswahl im ersten Schritt von Algorithmus 7.8 viele kurze Paare berechnet. Füge diese in die Datenstruktur ein. Dabei sei $O(\beta_j)$ die Bahn von β_j in $G_{\beta_{j-1}}$.

- Teste, welche Basispunkte von einem kurzen Paar g fix gelassen werden.
- Bestimme die Stufe j mit $\beta_i^g = \beta_i$, $i < j$ und $\gamma := \beta_j^g \neq \beta_j$.
  Falls $\gamma \notin O(\beta_j)$: verwende g auf jeden Fall als Element der Transversalen $T^{(j)}$.
  Falls $\gamma \in O(\beta_j)$: verwende g, falls es kürzer ist als das bisher gespeicherte Wort $\tilde{g}$ mit $\gamma = \beta_j^{\tilde{g}}$.
  Speichere ggf. die Kante $\beta_j \stackrel{g}{\to} \gamma$ im Bahngraphen.

Wir betrachten nun einige Heuristiken zur Vervollständigung der Transversalen.

Strategie 7.10 Nutze „Kollisionen“ im Bahngraphen:



Speichere zusätzlich die redundanten Kanten (im Bild gestrichelt) im Bahngraphen ab, eventuell sogar mehrfache Kanten. Kreise im Bahngraphen (im Bild fett) liefern Elemente des Stabilisators (z.B. ist im Bild $g_1 g_4^{-1} g_3^{-1} \in G_{\beta_j}$). Diese Elemente können in den Bahnen von $\beta_{j+1}, \beta_{j+2}, ...$ von Nutzen sein.
Vorgehen: Finde möglichst kurze Kreise im Bahngraphen und füge diese Elemente auf tieferen Stufen der Stabilisatorkette in die Datenstruktur ein.

Strategie 7.11 *Orbit Graph Completion*: Verwende alle vorhandenen kurzen Elemente in jedem Bahngraphen, um möglichst viele Kanten mit kurzen Faktorisierungen zu haben. Auch Elemente $g \in G_{\beta_1,...,\beta_j}$ können durch Verknüpfung mit anderen Gruppenelementen in den Graphen zu $\beta_1, ..., \beta_{j-1}$ neue Transversalelemente liefern.

Strategie 7.12 Vervollständigung mit zufällig gewählten kurzen Worten (größere Worte als die initialen kurzen Paare).

Strategie 7.13 Verwende Schreier-Generatoren.

Bei der konkreten Implementierung dieser Strategien gibt es einige Freiheitsgrade:

- Wieviele initiale kurze Paare?
- Wieviele konjugierte Paare?
- Wieviele Kanten werden gespeichert?
- Welche Reihenfolge bei der Anwendung mehrerer Strategien?

Bemerkung 7.14 Die maximale Wortlänge ist die Summe der maximalen Wortlängen in jeder Transversale.

Sei nun eine Stabilisatorkette mit einer Faktorisierungsfunktion wie z.B. in Algorithmus 4.13 gegeben. Zu $g \in G$ bezeichne $w(g)$ das durch diesen Algorithmus geliefert Wort in den Erzeugern von G.

Definition 7.15 Wir definieren die **Verteilungsfunktion der Wortlänge** (engl. *length distribution function*, kurz LDF) als formale Potenzreihe

$$\begin{aligned} \mathrm{ldf}(Z) &= \sum_{g \in G} Z^{|w(g)|} \\ &= \sum_{m \geq 0} |\{g : |w(g)| = m\}| \cdot Z^m. \end{aligned}$$

Der Koeffizient p_m von Z^m gibt an, wieviele Worte mit der Wortlänge m es in der Gruppe gibt.

Bemerkung 7.16 Die einfacher zu berechnende Funktion

$$\mathrm{ldf}^*(Z) = \prod_{j=1}^{l} \left(\sum_{t \in T^{(j)}} Z^{|w(t)|} \right)$$

liefert eine Abschätzung für $\mathrm{ldf}(Z)$ im folgenden Sinne: Für $\mathrm{ldf}^*(Z) = \sum_{m\geq 0} p_m^* Z^m$ und $\mathrm{ldf}(Z) = \sum_{m\geq 0} p_m Z^m$ gilt

$$\sum_{m=0}^{k} p_m \geq \sum_{m=0}^{k} p_m^*, \quad 1 \leq k \leq l.$$

Diese Funktionen hängen natürlich von dem vorher ermittelten Transversalensystem ab, da der Faktorisierungsalgorithmus darauf zurückgreift.

Bemerkung 7.17 Werden Elemente aus verschiedenen Transversalen miteinander multipliziert, so wird das Wort nicht länger als die Summe der Wortlängen:

$$|t_j t_{j+1}| \leq |t_j| + |t_{j+1}|.$$

Dies zeigt, dass die Koeffizienten von $\mathrm{ldf}(Z)$ und $\mathrm{ldf}^*(Z)$ nicht unbedingt identisch sind, da ldf^* die ungekürzten Worte zählt.

Beobachtung: Nur relativ wenige Elemente haben eine sehr große Wortlänge.
Idee: Faktorisiere leicht modifizierte Elemente, nämlich das Produkt von g und einem kurzen Wort: *Trembling*.

Algorithmus 7.18 Trembling

Eingabe: g soll faktorisiert werden, $k \in \mathbb{N}$ ist Anzahl der initialen kurzen Paare $\mathsf{sp}_1 \leq \ldots \leq \mathsf{sp}_k$ (*short pair*) aus Schritt 1 von Algorithmus 7.8
Ausgabe: Eine voraussichtlich kürzere Faktorisierung w von g

$w := \mathrm{factor}(g)$ // *Algorithmus 4.13*
for $i = 1$ **to** k
  $v := \mathsf{sp}_i^{-1} \cdot \mathrm{factor}(\Phi(\mathsf{sp}_i) \cdot g)$ // Φ *kanonischer Homomorphismus*
  if $|v| < |w|$ **then** $w := v$
return w

Bemerkung 7.19 Die Abhängigkeit der Länge von $\Phi(\mathsf{sp}_i) \cdot g$ von $|\mathsf{sp}_i|$ ist im Allgemeinen „sehr unstetig“, d.h sie variiert stark.

Eine genauere Analyse findet sich bei Egner und Püschel [4]: Die Verteilung eines *random walk* auf dem Cayley-Graphen bzw. einem Teil davon konvergiert schnell für Gleichverteilung auf der Gruppe. Man beobachtet dabei, dass mit relativ wenigen kurzen Paaren die Wortlänge auf ca. 66% der urspünglichen Länge reduziert werden kann.

8 Faktorisierung mit Freiheitsgraden

Bisher haben wir für eine gegebene Gruppe $G = \langle S \rangle \leq \mathsf{S}_n$ und ein gegebenes $\sigma \in \mathsf{S}_n$ folgende Fragestellungen betrachtet:

- Prüfe, ob σ in G enthalten ist
- Faktorisiere σ als $\sigma = s_{i_1} \cdots s_{i_k}$ mit möglichst wenigen Faktoren $s_{i_j} \in S$

Jetzt wollen wir *Freiheitsgrade* über eine zusätzliche Untergruppe $H \leq G$ einführen. Dies ist gerechtfertigt, wenn die Untergruppe H „unsichtbar“ in dem Sinne ist, dass Permutationen aus H den Ausgangszustand eines Systems (z.B. eines Puzzles) in nicht unterscheidbarer Weise transformieren. Als verdeutlichendes Beispiel kann das *Brezel*-Puzzle von Egner und Püschel [4] dienen.

Problem: „Faktorisiere modulo H“, d.h. finde ein $h \in H$ und $s = s_{i_1} \cdots s_{i_l}$, so dass

- $\sigma = h \cdot s$ gilt
- die Wortlänge $|s|$ möglichst klein ist

Bemerkung 8.1 Im Idealfall ist H ein Element der Stabilisatorkette, d.h.

$$G \geq G_{\beta_1} \geq ... \geq G_{\beta_1,...,\beta_j} = H \geq ... \geq \{\mathrm{id}\}.$$

Dann ist eine partielle Faktorisierung dadurch gegeben, dass man nur die Faktoren aus den Transversalen bis zu $H = G_{\beta_1,...,\beta_j}$ verwendet:

$$\sigma = h \cdot t_j \cdots t_1.$$

Problem: Wie kann man erreichen, dass $H = G_{\beta_1,...,\beta_j}$ ist (falls überhaupt möglich)?

Ansatz: Initialisiere die Partition $\boldsymbol{B}_0 = (\{\beta_1, ..., \beta_j\}, \{\beta_{j+1}, ...\})$ im Algorithmus 7.8 für die Basiswahl mit den Fixpunkten bzw. Nichtfixpunkten von H.
Aber: Im Allgemeinen hat H wenig Fixpunkte, so dass dieser Ansatz nicht zum Erfolg führt.

Dennoch kann man eventuell erreichen, dass es eine relativ große Untergruppe von H gibt, die in der Stabilisatorkette auftaucht:
Für eine feste Basis B gelte etwa:

$$\begin{array}{ccccccc} G & \geq & G_{\beta_1} & \geq & G_{\beta_1,\beta_2} & \geq & ... \geq G_{\beta_1,...,\beta_j} \geq ... \geq \{\mathrm{id}\} \\ \vee & & \vee & & & & \\ H & = & H_{\beta_1} & \gneq & H_{\beta_1,\beta_2} = H_{\beta_1,\beta_2,\beta_3} & \geq & ... \geq \{\mathrm{id}\} \end{array}$$

Falls die Basis gut gewählt werden kann, so sind in der zugehörigen Stabilisatorkette einige Untergruppen $H_{\beta_1,...,\beta_k}$ und $H_{\beta_1,...,\beta_{k+1}}$ identisch und außerdem $H_{\beta_1,...,\beta_j} = G_{\beta_1,...,\beta_j}$ für ein kleines j. So kann man erreichen, dass ein Teil der Freiheitsgrade ausgenutzt werden kann.

Problem: Wie wählt man eine gute Basis?

Strategie 8.2 Heuristik:

1. Wähle eine Basis B_0 für die Untergruppe H.
2. Füge weitere bzgl. H redundante Basispunkte ein, so dass man eine Basis B für G erhält.

Es kann sein, dass es bei Permutationsgruppen keine Basis gibt, die es erlaubt, die Freiheitsgrade von H auszunutzen. Also ein anderer Ansatz:

Strategie 8.3 Betrachte eine neue Operation der Gruppe, beispielsweise

1. Operation auf (kleinen) Mengen von Punkten. H hat eventuell keine Fixpunkte, aber kleine Bahnen, etwa $|1^H| = k \ll n = |G|$. Betrachte die Operationen auf k-Teilmengen der Gruppe G, wähle etwa $\beta_i = 1^H$. Die Anzahl der „Punkte“ (k-Teilmengen) ist $\binom{n}{k}$.
2. Operation von G auf H durch Konjugation oder Rechts-/Linksmultiplikation.

Konjugation	Multiplikation
evntl. nichttrivialer Kern operiert auf Menge der konjugierten Untergruppen	treu operiert auf Menge aller Nebenklassen

Die Stabilisatorkette hängt von der Basis ab und damit auch von der Operation der Gruppe. Optimierungen durch weitere Gruppenoperationen (auch gemischt) sind möglich, etwa Operationen auf Teilmengen bei Permutationsgruppen.

Es gibt allerdings Gruppen mit einem Untergruppenverband, der „schlecht“ ist für die Methode der Faktorisierung mit Stabilisatorketten, da stets ein Paar von Untergruppen mit großem Index (also großen Bahnen) existiert.

9 Endliche Körper und ihre Darstellungen

Definition 9.1 Als **endlichen Körper** $\mathbb{F}_q$ bezeichnet man einen Körper mit q Elementen, wobei $q = p^m$ für eine Primzahl p und $m \in \mathbb{N}$ ist.

Bemerkung 9.2 Eigenschaften endlicher Körper.

1. Zu jedem Körper $\mathbb{F}_q$ mit $q = p^m$ gibt es einen **Primkörper** $\mathbb{F}_p \cong \mathbb{Z}/p\mathbb{Z}$.
2. $\mathbb{F}_q$ mit $q = p^m$ hat die Charakteristik p, d.h. $\underbrace{1 + ... + 1}_{p\text{-mal}} = 0$. Folglich gilt auch für jedes $a \in \mathbb{F}_q$:
$$\underbrace{a + ... + a}_{p\text{-mal}} = a \cdot (1 + ... + 1) = 0.$$
3. Die endlichen Körper entstehen aus dem Primkörper als **Erweiterungskörper** der Form
$$\mathbb{F}_{p^m} \cong \mathbb{F}_p[X]/\langle f \rangle,$$
wobei f ein normiertes und irreduzibles Polynom vom Grad m ist. Das Inverse von $g \in \mathbb{F}_p[X]/\langle f \rangle$, $g \neq 0$, lässt sich über den Euklidischen Algorithmus bestimmen, denn es gilt für gewisse a, b:
$$\begin{aligned} & f \text{ irreduzibel} \\ \Rightarrow\ & 1 = \mathrm{ggT}(f, g) = af + bg \\ \Rightarrow\ & b \equiv g^{-1} \bmod f. \end{aligned}$$
4. Die **Einheitengruppe** $\mathbb{F}_q^\times = \mathbb{F}_q \backslash \{0\}$ aller invertierbaren Elemente ist zyklisch, d.h. es gibt ein **primitives Element** $\alpha \in \mathbb{F}_q^\times$, so dass jedes Element als $\beta = \alpha^j$ geschrieben werden kann für ein $j \in \{0, ..., q-2\}$.

Satz 9.3 Für $q = p^m$ definieren wir den **Frobenius-Automorphismus** durch
$$F : \mathbb{F}_q \to \mathbb{F}_q, \quad \gamma \mapsto \gamma^p.$$
Es gilt (wegen Charakteristik p):
$$\begin{aligned} F(ab) &= (ab)^p = a^p b^p, \\ F(a+b) &= (a+b)^p = a^p + b^p. \end{aligned}$$

Bemerkung 9.4 Der Frobenius-Automorphismus ist ein Körperautomorphismus der Ordnung m, d.h. für alle $\gamma \in \mathbb{F}_q$ gilt
$$F^m(\gamma) = \gamma^{p^m} = \gamma^q = \gamma \quad \text{bzw.} \quad \gamma^{q-1} = 1.$$
Daraus folgt, dass das Polynom f aus Bemerkung 9.2(3) ein Teiler von $X^q - X = X(X^{q-1} - 1)$ ist (da $X^q - X \equiv 0 \bmod f$), und da f irreduzibel ist, ist es sogar ein Teiler von $X^{q-1} - 1$.

Bemerkung 9.5 Mit einem primitiven Element α kann man in $\mathbb{F}_q^\times$ „bequem“ rechnen: Die Multiplikation wird zur Addition der Exponenten:

$$\alpha^i \cdot \alpha^j = \alpha^{i+j},$$

im Exponenten findet eine Addition in $\mathbb{Z}/(q-1)\mathbb{Z}$ statt.
Die Addition kann mit einem Trick auf die Multiplikation zurückgeführt werden:

$$\alpha^i + \alpha^j = \alpha^i \cdot \underbrace{(1+\alpha^{j-i})}_{=\alpha^k} = \alpha^{i+k},$$

falls $\alpha^{j-i} \neq -1$. Um dies schnell berechnen zu können, benötigt man eine Tabelle der $1+\alpha^l$, den sogenannten **Zech-Logarithmus**.

Frage: Auf welche Weise(n) kann man endliche Körper darstellen?

Satz 9.6 (Polynomdarstellung)
Die endlichen Körper erhält man als Quotienten von $\mathbb{F}_p[X]$:

$$\mathbb{F}_{p^m} \cong \mathbb{F}_p[X]/\langle f \rangle,$$

wobei f ein normiertes und irreduzibles Polynom vom Grad m ist, siehe Bemerkung 9.2(3).

Satz 9.7 Alle endlichen Körper mit q Elementen sind isomorph.

Es gibt aber keine kanonischen Isomorphismus, d.h. der Isomorphismus zwischen zwei verschiedenen Darstellungen muss explizit angegeben werden.

Beispiel 9.8 Die Polynome $f_1 = X^3 + X + 1 \in \mathbb{F}_2[X]$ und $f_2 = Y^3 + Y^2 + 1 \in \mathbb{F}_2[Y]$ liefern beide eine Darstellung von $\mathbb{F}_8$. Man könnte etwa über primitive Elemente α_1, α_2 einen Isomorphismus $\alpha_1^k \mapsto \alpha_2^k$ konstruieren.

Bemerkung 9.9 Eine Nullstelle des definierenden Polynoms f ist nicht unbedingt ein primitives Element: $f = X^2 + 1 \in \mathbb{F}_3[X]$ liefert $\mathbb{F}_9$. Eine Nullstelle sei λ. Dann ist $\lambda^2 = -1$, also $\lambda^4 = 1$. Die Ordnung von λ ist zu klein, sie müsste $q - 1 = 8$ sein für ein primitives Element.

Die Darstellung von $\mathbb{F}_q$ durch Polynome vom Grad $< m$ (vgl. Bemerkung 9.2 bzw. Satz 9.6) liefert für $\gamma \in \mathbb{F}_q$ eine Darstellung als Koeffizientenvektor:

$$\gamma \;\cong\; \gamma(X) = \gamma_0 + \gamma_1 X + ... + \gamma_{m-1} X^{m-1} \;\cong\; (\gamma_0, \gamma_1, ..., \gamma_{m-1}) \in \mathbb{F}_p^m.$$

Satz 9.10 (Vektorraumdarstellung)
Der Körper $\mathbb{F}_q$ ist ein $\mathbb{F}_p$-Vektorraum der Dimension m, d.h. jedes Element $\gamma \in \mathbb{F}_q$ entspricht einem Vektor in $\mathbb{F}_p^m$:

$$\mathbb{F}_q = \mathbb{F}_{p^m} \cong \mathbb{F}_p^m \quad \text{(als } \mathbb{F}_p\text{-Vektorraum)}$$

Ein Basiswechsel liefert eine andere Darstellung der Elemente, d.h. allgemein hat man für eine Basis $B = \{b_1, ..., b_m\}$ von $\mathbb{F}_p^m$ und $\gamma \in \mathbb{F}_q$ die Darstellung

$$\gamma = \sum_{i=1}^{m} \tilde{\gamma}_i b_i$$

mit $\tilde{\gamma}_i \in \mathbb{F}_p$.

Ziel beim Basiswechsel ist es, eine Darstellung zu erhalten, die Rechenoperationen vereinfacht, also z.B. eine schnellere Multiplikation ermöglicht.

Die Operation des Körpers auf sich selbst durch Multiplikation, $x \mapsto x \cdot \gamma$, ist eine $\mathbb{F}_p$-lineare Abbildung. Es gibt also für jedes $\gamma \in \mathbb{F}_{p^m}$ eine Matrixdarstellung $M_\gamma \in \mathbb{F}_p^{m \times m}$.

Satz 9.11 (Matrixdarstellung)
$\iota : \mathbb{F}_{p^m} \hookrightarrow \mathbb{F}_p^{m \times m}, \gamma \mapsto M_\gamma$, ist eine injektive Einbettung in $\mathbb{F}_p^{m \times m}$. Sie ist ein Isomorphismus auf das Bild $\iota(\mathbb{F}_{p^m})$:

$$M_{\beta+\gamma} = M_\beta + M_\gamma,$$
$$M_{\beta\gamma} = M_\beta M_\gamma.$$

Bemerkung 9.12 Die Wahl der Darstellung muss in jedem Anwendungsfall neu getroffen werden:

- Welche Operationen werden ausgeführt?
- z.B. Kryptographie: Diskreter Logarithmus über $\mathbb{F}_q$: Keine Darstellung bzgl. des primitiven Elementes.

10 Matrixgruppen über endlichen Körpern

Definition 10.1 Mit

$$\begin{aligned}\mathrm{GL}_n(\mathbb{F}_q) &= \{A \in \mathbb{F}_q^{n\times n} : \text{ es gibt } A^{-1} \text{ mit } A^{-1}A = I_n = AA^{-1}\}\\ &= \{A \in \mathbb{F}_q^{n\times n} : \det(A) \neq 0\}\end{aligned}$$

bezeichnen wir die Gruppe der invertierbaren $n \times n$-Matrizen über $\mathbb{F}_q$.

Bemerkung 10.2 Als „natürliche Operation“ von $\mathrm{GL}_n(\mathbb{F}_q)$ auf dem Vektorraum $\mathbb{F}_q^n$ betrachten wir die Multiplikation von $x \in \mathbb{F}_q^n$ mit $A \in \mathrm{GL}_n(\mathbb{F}_q)$. Diese Operation ist treu, da eine Vektorraumbasis nur von der Einheitsmatrix I_n festgelassen wird.

Problem: Wahl der Basispunkte für eine starke Erzeugermenge.

Sei $G \leq \mathrm{GL}_n(\mathbb{F}_q)$. Für $|G| < q^n - 1$ gibt es einen Vektor $v \in \mathbb{F}_q^n$, der nur von der Einheitsmatrix festgelassen wird. Also: $|v^G| = |G|$ und $G_v = \{I_n\}$. Die (ziemlich kurze) Stabilisatorkette ist

$$G \geq G_v = \{I_n\},$$

die Bahnlänge wäre $|G|$, also maximal.

Ziel: Finde Vektoren, die von möglichst vielen Gruppenelementen festgelassen werden, um eine große Untergruppe in der Stabilisatorkette zu erhalten. Daraus erhält man eine kurze Bahn.

Idee: Verwende nicht nur Vektoren, sondern auch (Unter-)Vektorräume als Basispunkte.

Bemerkung 10.3 Für eindimensionale Vektorräume der Form

$$\langle v\rangle = \{\gamma v : \gamma \in \mathbb{F}_q\}, \quad v \neq 0,$$

ist $1 \leq [G_{\langle v\rangle} : G_v] \leq q - 1$ (eine Nebenklasse für jeden möglichen Eigenwert aus $\mathbb{F}_q$). Verwendet man $\langle v\rangle$ und v in der Stabilisatorkette,

$$\ldots \geq G_{\langle v\rangle} \geq G_v \geq \ldots,$$

so kann die Bahnlänge um einen Faktor bis zu $q - 1$ reduziert werden.
Beachte: $A \in G_{\langle v\rangle}$ bedeutet, dass v Eigenvektor von A ist; $A \in G_v$ bedeutet, dass v Eigenvektor zum Eigenwert 1 von A ist.

Bemerkung 10.4 Aus der linearen Algebra ist bekannt, dass man zu einer Matrix A eine *Normalform* N_A bestimmen kann. Dafür schreibt man das *charakteristische Polynom* f_A von A als Produkt von irreduziblen Faktoren g_i:

$$f_A(\lambda) = \det(\lambda I_n - A) = g_1(\lambda)g_2(\lambda)\cdots g_k(\lambda).$$

Die Normalform ist dann

$$N_A = \begin{pmatrix} \boxed{G_1} & & & \\ & \boxed{G_2} & & \\ & & \ddots & \\ & & & \boxed{G_k} \end{pmatrix},$$

wobei die G_i die *Begleitmatrizen* der Faktoren g_i (mit $\deg(g_i) = d_i$) sind:

$$G_i = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ -g_{i,0} & -g_{i,1} & \cdots & -g_{i,d_i-2} & -g_{i,d_i-1} \end{pmatrix}.$$

Jedes G_i beschreibt die Aktion von A auf einem *verallgemeinerten Eigenraum* (oder *Hauptraum*) der Dimension d_i.

Wähle als Basispunkte verallgemeinerte Eigenräume mit kleiner Dimension, denn wie wir im folgenden Satz 10.6 sehen werden, verbessert dies die Chancen, kurze Bahnen zu erhalten.

Lemma 10.5 Für Vektoren v aus dem verallgemeinerten Eigenraum zu g_i gilt:

$$g_i(A) \cdot v = 0.$$

BEWEIS: vgl. Satz von Cayley-Hamilton aus der linearen Algebra. ■

Daraus erhält man eine Abschätzung der Bahnlänge:

Satz 10.6 Es sei v ein Vektor aus dem verallgemeinerten Eigenraum zu g_i und $d = \deg(g_i)$. Dann gilt:

$$|v^{\langle A \rangle}| \leq q^d - 1.$$

BEWEIS: $g_i(\lambda)$ teilt $\lambda^{q^d-1} - 1$. Daraus folgt $(A^{q^d-1} - I_n) \cdot v = f_i(A) \cdot g_i(A) \cdot v = 0$, denn nach Lemma 10.5 ist $g_i(A) \cdot v = 0$. ■

Algorithmus 10.7 Basiswahl für Matrixgruppen

1. Berechne (verallgemeinerte) Eigenräume zu einigen zufällig gewählten Gruppenelementen.
2. Berechne die Schnitte dieser Eigenräume.
3. Wähle daraus eine Basis nach folgenden (heuristischen) Kriterien:
   (a) kleinster Grad des zugehörigen Faktors g_i
   (b) kleine Anzahl von Termen in g_i
   (c) große Anzahl von gemeinsamen Eigenräumen
   (d) kleine Anzahl von Einträgen in den Vektoren ($\neq 0$)
   (e) kleinste Dimension

Ausführliche Informationen dazu findet man bei Murray und O'Brien [10].

Teil II

Codes

11 Lineare Blockcodes

In diesem Abschnitt sei $\mathbb{F}_q$ ein endlicher Körper mit $q = p^m$, p prim. Unter einem (allgemeinen) **Code** der Länge n verstehen wir hier eine Untermenge $C \subseteq \mathbb{F}_q^n$.

Definition 11.1 Als **linearen Blockcode** $C[n, k, d, q]$ bezeichnen wir einen Untervektorraum C von $\mathbb{F}_q^n$ mit folgenden Eigenschaften:

1. $\dim_{\mathbb{F}_q}(C) = k$, d.h. C enthält $|C| = q^k$ Codeworte.
2. Die **Minimaldistanz** des Codes ist

$$\min\{\mathrm{dist}(\boldsymbol{c}, \boldsymbol{c'}) : \boldsymbol{c}, \boldsymbol{c'} \in C, \boldsymbol{c} \neq \boldsymbol{c'}\} = d,$$

wobei

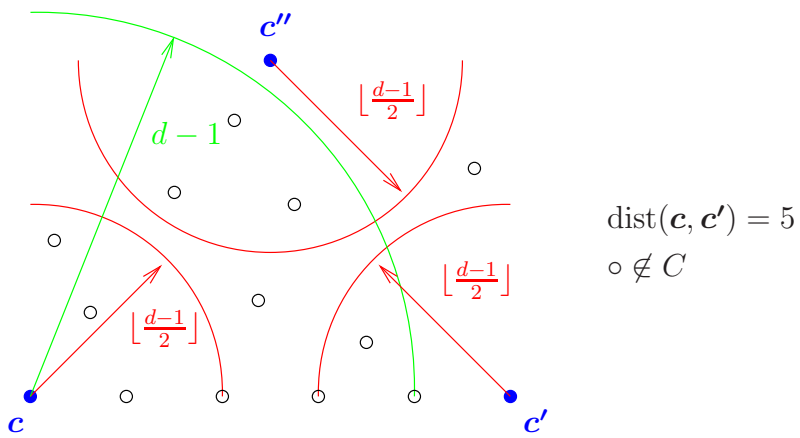
$$\mathrm{dist}(\boldsymbol{c}, \boldsymbol{c'}) = |\{j : c_j \neq c'_j\}| \quad \text{für} \quad \boldsymbol{c} = (c_1, ..., c_n), \boldsymbol{c'} = (c'_1, ..., c'_n)$$

die **Hamming-Distanz** zweier Codewörter aus C ist (also die Anzahl der Stellen von $\boldsymbol{c}$ und $\boldsymbol{c'}$, die verschieden sind).

Bemerkung 11.2 Ist $m = m_1 m_2$, so kann man C auch als $\mathbb{F}_{p^{m_1}}$-Untervektorraum von $\mathbb{F}_{p^{m_1 m_2}}^n \cong \mathbb{F}_{p^{m_1}}^{nm_2}$ auffassen. $\mathbb{F}_{p^{m_1}}$ ist ein Teilkörper von $\mathbb{F}_{p^{m_1 m_2}}$.

Die Minimaldistanz d ist eines der Kriterien für die Fehlerkorrekturfähigkeit des Codes.

Lemma 11.3 Ein Code mit Minimaldistanz d kann *entweder* bis zu $d-1$ Fehler erkennen *oder* bis zu $\lfloor \frac{d-1}{2} \rfloor$ Fehler korrigieren.



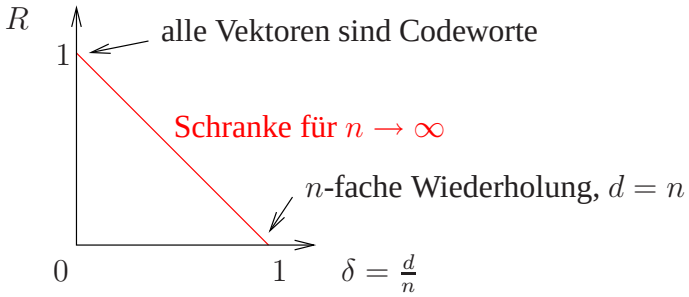
BEWEIS: Skizze: Die Kugeln mit dem Radius $\lfloor \frac{d-1}{2} \rfloor$ um die Codeworte sind disjunkt. Folglich korrespondieren alle Vektoren in einer Kugel zu genau einem Codewort. Kugeln mit Radius $d-1$ enthalten genau ein Codewort. ■

Ziele beim Entwurf von Codes

1. Die **Rate**
$$R = \frac{k}{n}$$
sollte möglichst groß sein (auch: $R_2 = \log_2(q\frac{k}{n}) =$ Bits pro Symbol bzw. Codewort).

2. Die Minimaldistanz sollte möglichst groß sein.

Dies sind konkurrierende Ziele:



Dies stellt für Codes endlicher Länge ein diskretes Optimierungsproblem dar.

12 Das Minimalgewicht linearer Codes

Sei $C[n, k, d, q]$ ein linearer Code.

Frage: Wie aufwendig ist es, einen Code zu beschreiben?

- Für einen allgemeinen Code, der durch eine Teilmenge von $\mathbb{F}_q^n$ mit m Elementen gegeben ist, müssen alle Codeworte angegeben werden. Die Beschreibung ist *linear* in der Anzahl der Codeworte.
- Da ein linearer Code ein Vektorraum der Dimension k über $\mathbb{F}_q$ ist, genügt es, eine Basis mit k linear unabhängigen Vektoren anzugeben, um die q^k Codeworte festzulegen. Die Beschreibung ist *logarithmisch* in der Anzahl der Codeworte.

Definition 12.1 Die **Codierungsabbildung** von C ist eine lineare Transformation

$$\varphi : \mathbb{F}_q^k \to \mathbb{F}_q^n, \quad \boldsymbol{i} \mapsto \boldsymbol{i} \cdot G \qquad (n > k).$$

Sie ist durch die **Generatormatrix** $G \in \mathbb{F}_q^{k \times n}$ gegeben, deren Zeilen[1)] eine Basis von C bilden, d.h. $\mathrm{Bild}(\varphi) = C$.
Die Codierungsabbildung bildet ein **Informationswort** $\boldsymbol{i}$ auf ein **Codewort** $\boldsymbol{c}$ ab.
Aufwand für die Codierung: $\mathcal{O}(nk) = \mathcal{O}(n^2)$.

Problem: Ein Absender schickt ein Codewort $\boldsymbol{c}$ an einen Empfänger, der $\boldsymbol{v} \in \mathbb{F}_q^n$ empfängt. Bei der Übertragung können **Fehler** auftreten, d.h. es ist $\boldsymbol{v} \neq \boldsymbol{c}$.

Frage: Wie kann man das Auftreten von Fehlern erkennen?

Teste, ob ein Vektor $\boldsymbol{v} \in \mathbb{F}_q^n$ ein Codewort ist:

Strategie 12.2 Löse die lineare Gleichung

$$\boldsymbol{i}G = \boldsymbol{v}$$

für festes $\boldsymbol{v}$ und Variablenvektor $\boldsymbol{i}$.

Strategie 12.3 Der Code C ist der Bildraum der Matrix G. Gleichzeitig ist jeder Vektorraum der Lösungsraum eines homogenen linearen Gleichungssystems. Folglich gibt es eine Matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ vom Rang $n - k$ mit

$$GH^\top = 0$$

oder äquivalent dazu $HG^\top = 0$. Somit ist G einen Basismatrix für $\mathrm{Kern}(H^\top)$.

Diese Matrix H ist die sogenannte **Prüfmatrix** zum linearen Code C, der von der Generatormatrix G erzeugt wird.

Mit Hilfe der Prüfmatrix können wir ein empfangenes Wort auf Fehler prüfen.

[1)] In der Codierungstheorie werden üblicherweise Zeilenvektoren verwendet.

Lemma 12.4 Sei $\boldsymbol{c} \in C$ das gesendete Codewort und $\boldsymbol{v}$ das empfangene Wort.

1. Es gilt:
$$\boldsymbol{v} \in C \quad \Leftrightarrow \quad \boldsymbol{v}H^\top = \boldsymbol{0}.$$

2. Der Vektor
$$\boldsymbol{s} = \boldsymbol{v}H^\top \in \mathbb{F}_q^{n-k}$$
heißt **Fehlersyndrom** von $\boldsymbol{v}$ (bzgl. H). Das Fehlersyndrom $\boldsymbol{s}$ hängt nur von einem additiven Fehler $\boldsymbol{e}$ ab, aber nicht vom Element $\boldsymbol{c}$, denn:
$$\begin{aligned} &\boldsymbol{v} = \boldsymbol{c} + \boldsymbol{e} \\ \Rightarrow\ &\boldsymbol{v}H^\top = \underbrace{\boldsymbol{c}H^\top}_{=\boldsymbol{0}} + \boldsymbol{e}H^\top = \boldsymbol{e}H^\top = \boldsymbol{s}. \end{aligned}$$

Aus dem bekannten $\boldsymbol{v}$ kann man also $\boldsymbol{c} = \boldsymbol{v} - \boldsymbol{e}$ rekonstruieren, wenn man $\boldsymbol{e}$ mit $\boldsymbol{e}H^\top = \boldsymbol{s}$ finden kann.

Definition 12.5 Das **Hamming-Gewicht** $\mathrm{wgt}(\boldsymbol{v})$ eines Vektors $\boldsymbol{v} \in \mathbb{F}_q^n$ ist definiert als die Anzahl der von 0 verschiedenen Einträge von $\boldsymbol{v}$.

Satz 12.6 Die Fehlerkorrektur ist NP-schwer (Berlekamp, McEliece, van Tilborg [1]). Entscheidungsproblem in NP:
Gegeben: Eine binäre Matrix $H \in \mathbb{F}_2^{m \times n}$, ein Vektor $\boldsymbol{s} \in \mathbb{F}_2^m$ und eine ganze Zahl $w > 0$.
Problem: Existiert ein Vektor $\boldsymbol{e} \in \mathbb{F}_2^n$ mit Hamming-Gewicht $\mathrm{wgt}(\boldsymbol{e}) \leq w$ und $\boldsymbol{e}H^\top = \boldsymbol{s}$?

Ein eng verwandtes Problem ist die Bestimmung der Fehlerkorrektureigenschaften.

Bemerkung 12.7 Die Minimaldistanz
$$d = \min\{\mathrm{dist}(\boldsymbol{c}, \boldsymbol{c'}) : \boldsymbol{c}, \boldsymbol{c'} \in C, \boldsymbol{c} \neq \boldsymbol{c'}\}$$
ist ein Kriterium dafür, wieviele Fehler erkannt bzw. korrigiert werden können (siehe Lemma 11.3).

- Bei einem allgemeinen Code K sind die Abstände für alle $\binom{|K|}{2}$ Paare von Codeworten zu bestimmen. Der Aufwand ist $\mathcal{O}(|K|^2)$.
- Für einen linearen Code C gilt:
$$\begin{aligned} \mathrm{dist}(\boldsymbol{c}, \boldsymbol{c'}) &= \mathrm{dist}(\boldsymbol{c} - \boldsymbol{c'}, \boldsymbol{c'} - \boldsymbol{c'}) \\ &= \mathrm{dist}(\boldsymbol{c} - \boldsymbol{c'}, \boldsymbol{0}) \\ &= \mathrm{wgt}(\boldsymbol{c} - \boldsymbol{c'}). \end{aligned}$$
Also gilt $d = \min\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C, \boldsymbol{c} \neq \boldsymbol{0}\}$, für lineare Codes ist also die Minimaldistanz gleich dem **Minimalgewicht**. Der Aufwand für das Aufzählen und Prüfen aller Codeworte ist $\mathcal{O}(|C|) = \mathcal{O}(q^k)$.

Satz 12.8 Die Bestimmung der Minimaldistanz (bzw. des Minimalgewichts) ist NP-schwer (Vardy [13]). Entscheidungproblem in NP:
Gegeben: Eine binäre Matrix $H \in \mathbb{F}_2^{m\times n}$ und eine ganze Zahl $w > 0$.
Problem: Gibt es einen Vektor $\boldsymbol{c} \in \mathbb{F}_q^n \backslash \{\mathbf{0}\}$ mit $\mathrm{wgt}(\boldsymbol{c}) \leq w$ und $\boldsymbol{c}H^\top = \mathbf{0}$?

Bemerkung 12.9 Die Berechnung des Minimalgewichts ist mit Hilfe des Entscheidungsproblems möglich: Teste für $w = 1, 2, ...$, ob es ein Codewort $\neq \mathbf{0}$ vom Gewicht w gibt.

Algorithmus 12.10 Berechnung des Minimalgewichts (naiver Ansatz)

Zähle alle Vektoren aus $\mathbb{F}_q^n$ mit aufsteigendem Hamming-Gewicht w auf, bis ein Codewort $\boldsymbol{c} \neq \mathbf{0}$ gefunden wurde.
Aufwand: mindestens

$$\sum_{w=1}^{d-1} \binom{n}{w} (q-1)^w$$

Vektoren müssen aufgezählt werden. Dabei ist $\binom{n}{w}$ die Anzahl der Möglichkeiten, w Stellen mit einem Eintrag $\neq 0$ zu finden, und $(q-1)^w$ ist die Anzahl der möglichen Werte aus $\mathbb{F}_q$ an diesen Stellen.

Bemerkung 12.11 Die Summanden wachsen (auch für $q = 2$) sehr schnell in w. Dennoch kann für „kleine“ d der Aufwand kleiner als q^k sein (was dem Aufzählen aller Codeworte entspräche).

Strategie 12.12 Systematische Codierung
Sei G eine Generatormatrix für den linearen Code C. Ist $S \in \mathrm{GL}_k(\mathbb{F}_q)$, so ist auch $S \cdot G$ eine Generatormatrix von C. Es ändert sich nur die Zuordnung von Informationsworten $\boldsymbol{i} \in \mathbb{F}_q^k$ und Codeworten $\boldsymbol{c} \in C$.
Das Vertauschen der Spalten von G ändert die Fehlerkorrektureigenschaften von C nicht. Also liefert (zeilenweise) Gauß-Elimination eine Generatormatrix G' der Form

$$G' = (I_k|A), \qquad A \in \mathbb{F}_q^{k\times(n-k)}.$$

Die Matrix G' heißt **systematische Generatormatrix**. Die Codierungsabbildung zu G' ist

$$\boldsymbol{i} \cdot G' = (\boldsymbol{i}I_k|\boldsymbol{i}A) = (\boldsymbol{i}|\boldsymbol{i}A).$$

Das Informationswort $\boldsymbol{i}$ ist also das Anfangsstück des Codeworts $\boldsymbol{c}$. Damit gilt:

$$\mathrm{wgt}(\boldsymbol{c}) \geq \mathrm{wgt}(\boldsymbol{i}).$$

Definition 12.13 $\mathcal{I} = \{i_1, ..., i_k\}$ heißt **Informationsmenge** zu C, falls die zugehörigen Spalten in G linear unabhängig sind. Zur Matrix G' der systematischen Codierung 12.12 gehört also die Informationsmenge $\{1, ..., k\}$.

Bemerkung 12.14 Allgemein gilt: Das Minimalgewicht einer Teilmenge des linearen Codes C liefert eine obere Schranke d_{up} für die Minimaldistanz d.
Problem: Finde eine untere Schranke d_{low}.

Algorithmus 12.15 Berechnung des Minimalgewichts mit systematischer Codierung

Sei G' die systematische Generatormatrix von C.
Zähle alle Informationsworte $\boldsymbol{i} \in \mathbb{F}_q^k$ mit aufsteigendem Gewicht auf.

- Die Menge $C_0 \subseteq C$ der bereits aufgezählten Codeworte liefert eine obere Schranke d_{up} für die Minimaldistanz d, d.h.

$$\begin{aligned} d &= \min\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C \backslash \{0\}\} \\ &\leq \min\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C_0 \backslash \{0\}\} =: d_{\text{up}}. \end{aligned}$$

- Das Gewicht von $\boldsymbol{i}$ liefert eine untere Schranke d_{low} für d:

$$\mathrm{wgt}(\boldsymbol{i}) =: d_{\text{low}} \leq \min\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C \backslash C_0, \boldsymbol{c} \neq 0\}.$$

  Es gilt sogar: Hat man alle Informationsworte $\boldsymbol{i}$ mit Gewicht $\leq w$ codiert, so ist das Gewicht der Codeworte, die man noch nicht aufgezählt hat, mindesten $w+1$.

Ende des Algorithmus, falls $d_{\text{low}} \geq d_{\text{up}}$ (oder falls alle Codeworte aufgezählt wurden). Dann gilt $d = d_{\text{up}}$.

Analyse

Aufwand: Es müssen

$$\text{mindestens} \sum_{w=1}^{\min\{k,d-1\}} \binom{k}{w} (q-1)^w$$

$$\text{höchstens} \sum_{w=1}^{\min\{k,d\}} \binom{k}{w} (q-1)^w$$

Worte betrachtet werden ($\binom{k}{w}(q-1)^w$ ist die Anzahl der Informationsworte der Länge k vom Gewicht w). Im Extremfall hat ein Wort minimalen Gewichts die Form $(\boldsymbol{i}|\boldsymbol{0})$ mit $\mathrm{wgt}(\boldsymbol{i}) = d$.

Verbesserung für $q > 2$: C ist linear abgeschlossen, insbesondere gilt

$$\boldsymbol{c} \in C, \alpha \in \mathbb{F}_q^\times \;\Rightarrow\; \alpha\boldsymbol{c} \in C, \mathrm{wgt}(\boldsymbol{c}) = \mathrm{wgt}(\alpha\boldsymbol{c}).$$

Es genügt, nur „normierte“ Vektoren $\boldsymbol{i} \in \mathbb{F}_q^k$ zu codieren, d.h. der erste von 0 verschiedene Eintrag von $\boldsymbol{i}$ ist eine 1. Damit ist der Aufwand höchstens

$$\sum_{w=1}^{\min\{k,d\}} \binom{k}{w} (q-1)^{w-1}.$$

Der Aufwand zur Codierung des Informationsworts, d.h. der Berechnung von $\boldsymbol{i} \mapsto (\boldsymbol{i}|\boldsymbol{i}A)$, $A \in \mathbb{F}_q^{k\times(n-k)}$, ist in $\mathcal{O}(n^2)$. Es müssen nur $\mathrm{wgt}(\boldsymbol{i}) = w$ viele Zeilen der Matrix A linear kombiniert werden. Das ergibt $\mathcal{O}(w(n-k))$ Körperoperationen.

Der Gesamtaufwand kann reduziert werden, wenn man bessere untere Schranken berechnen kann. Die Theorie liefert für manche Codekonstruktionen untere Schranken.

Bemerkung 12.16 Nehmen wir nun $2k \leq n$ an. Weiter seien $\mathcal{I}_1 = \{1, ..., k\}$ und $\mathcal{I}_2 = \{n-k+1, ..., n\}$ Informationsmengen zum Code $C[n, k, d, q]$, d.h. es gibt Generatormatrizen G_1, G_2 der Form

$$G_1 = (I_k | A_1) \quad \text{und} \quad G_2 = (A_2 | I_k).$$

G_1 und G_2 erzeugen denselben Code C (ohne Permutation der Spalten kommt man von G_1 zu G_2 durch den Gauß-Algorithmus). Codierung von $\boldsymbol{i} \in \mathbb{F}_q^k$ mit G_1 und G_2:

$$\begin{aligned} \boldsymbol{i}G_1 &= (\boldsymbol{i} | \boldsymbol{i}A_1) = \boldsymbol{c}_1 \\ \boldsymbol{i}G_2 &= (\underbrace{\boldsymbol{i}}_{k} \underbrace{A_2}_{n-2k} | \underbrace{\boldsymbol{i}}_{k}) = \boldsymbol{c}_2 \end{aligned}$$

Das Gewicht von $\boldsymbol{c}_1$ an den ersten k Positionen ist w, das Gewicht von $\boldsymbol{c}_2$ an den letzten k Positionen ist w.

Es seien

$$\begin{aligned} C_1 &= \{\boldsymbol{i}G_1 : \boldsymbol{i} \in \mathbb{F}_q^k, \mathrm{wgt}(\boldsymbol{i}) < w\} \subseteq C, \\ C_2 &= \{\boldsymbol{i}G_2 : \boldsymbol{i} \in \mathbb{F}_q^k, \mathrm{wgt}(\boldsymbol{i}) < w\} \subseteq C. \end{aligned}$$

Dann gilt mit obigen Annahmen:

$$\min\{\mathrm{wgt}(\boldsymbol{c}) : \boldsymbol{c} \in C, \boldsymbol{c} \notin C_1 \cup C_2\} \geq 2w = d_{\text{low}}.$$

Ist nämlich $\boldsymbol{c} \notin C_1 \cup C_2$, so muss $\boldsymbol{c}$ an den ersten und letzten k Stellen jeweils Gewicht $\geq w$ haben, insgesamt also $\mathrm{wgt}(\boldsymbol{c}) \geq 2w$.

Führt man nun Algorithmus 12.15 mit beiden systematischen Generatormatrizen G_1 und G_2 durch, so hat man eine untere Schranke, die doppelt so schnell anwächst, aber auch doppelten Aufwand bei der Codierung. Im Allgemeinen gilt $C_1 \cap C_2 \neq \emptyset$, d.h. es werden Codeworte doppelt aufgezählt. Insgesamt müssen (höchstens)

$$2 \sum_{w=1}^{\lfloor \frac{d}{2} \rfloor} \binom{k}{w} (q-1)^{w-1}$$

Codeworte aufgezählt werden.

Bemerkung 12.17 Nun gelte $mk \leq n$. Seien $\mathcal{I}_1, ..., \mathcal{I}_m$ disjunkte Informationsmengen zu einem linearen Code $C[n, k, d, q]$ mit zugehörigen Generatormatrizen $G_1, ..., G_m$, d.h. G_j hat an den Positionen $l \in \mathcal{I}_j$ eine Einheitsmatrix. Hat man alle Informationsworte $\boldsymbol{i} \in \mathbb{F}_q^k$ mit Gewicht $\mathrm{wgt}(\boldsymbol{i}) < w$ durch die Matrizen G_j codiert, so ist das Gewicht der noch nicht betrachteten Codeworte mindestens mw. Der Aufwand ist

$$m \sum_{w=1}^{\lfloor \frac{d}{m} \rfloor} \binom{k}{w} (q-1)^{w-1}.$$

Es werden also in jedem Schritt (d.h. für festes w) m-mal soviele Codeworte aufgezählt wie im Algorithmus 12.15. Andererseits wächst die untere Schranke m-mal schneller,

so dass das maximale Gewicht der aufgezählten Informationsworte sehr viel kleiner ist. Da der Aufwand vom letzten Term der Summe dominiert wird, ergibt sich insgesamt für größeres m eine Verbesserung.

Problem: Für $n > mk$ ist noch nicht garantiert, dass auch wirklich m disjunkte Informationsmengen existieren, betrachte z.B.

$$G_1 = \begin{pmatrix} 1 & & 0 & 1 & ... & 1 \\ & \ddots & & & 0 & \\ 0 & & 1 & & & \end{pmatrix}.$$

Hier sind alle Informationsmengen von der Form $\{2, ..., k\} \cup \{j\}, j = 1, k+1, ..., n$.

Algorithmus 12.18 Informationsmengen bestimmen

Ein Greedy-Algorithmus:

1. Berechne die systematische Generatormatrix G_1, Œ sei $G_1 = (I_k|A_1)$, d.h. $\mathcal{I}_1 = \{1, ..., k\}$.
2. Wende den Gauß-Algorithmus so auf G_1 an, so dass A_1 in Treppenform überführt wird:

$$A_1 \text{ ist ähnlich zu } \begin{pmatrix} I_{r_2} & A_2 \\ 0 & 0 \end{pmatrix}, \quad r_2 = \mathrm{Rang}(A_1)$$

$$G_1 \underset{\text{Gauß}}{\rightsquigarrow} G_2 = \begin{pmatrix} B_{11} & 0 & | & I_{r_2} & A_2 \\ B_{21} & I_{k-r_2} & | & 0 & 0 \end{pmatrix}$$

   Wir erhalten durch den Block

$$\begin{matrix} 0 & | & I_{r_2} \\ I_{k-r_2} & | & 0 \end{matrix}$$

   aus G_2 eine weitere Informationsmenge $\mathcal{I}_2$. Die Informationsmengen $\mathcal{I}_1$ und $\mathcal{I}_2$ sind disjunkt, falls die Teilmatrix A_1 maximalen Rang k hat. Ansonsten ist $|\mathcal{I}_1 \cap \mathcal{I}_2| = k - r_2$.
3. Setze das Verfahren entsprechend für A_2 fort, bis soviele Informationsmengen gefunden sind, dass alle Stellen $1, ..., n$ in mindestens einer Informationsmenge vorkommen.

Der Algorithmus liefert Informationsmengen $\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_m$ mit den Eigenschaften

- $\mathcal{I}_1 \cap ... \cap \mathcal{I}_m = \emptyset$, falls die Teilmatrizen A_j jeweils den maximalen Rang k haben.
- $\mathcal{I}_j$ ist maximal in der Teilmenge

$$\{1, ..., n\} \backslash \bigcup_{l=1}^{j-1} \mathcal{I}_l.$$

Bemerkung 12.19 Im Idealfall existieren für den Code C disjunkte Informationsmengen. Aber selbst wenn disjunkte Informationsmengen existieren, liefert Algorithmus 12.18 nicht unbedingt disjunkte Informationsmengen. Ist $G_1 = (I_k|A_1)$ und ist der Rang von A_1 kleiner als k, so erhält man nur $r_2 = \mathrm{Rang}(A_1)$ neue Informationspositionen für $\mathcal{I}_2$, die restlichen $k - r_2$ Positionen sind auch in $\mathcal{I}_1$ enthalten. Trotzdem erhält man durch die Verwendung von G_1 und G_2 eine verbesserte untere Schranke beim Aufzählen der Codeworte: Sind alle Informationsworte $\boldsymbol{i}$ mit $\mathrm{wgt}(\boldsymbol{i}) \leq w$ aufgezählt worden, so haben die noch nicht aufgezählten Codeworte an den Stellen von $\mathcal{I}_1$ und $\mathcal{I}_2$ jeweils $w + 1$ Einträge $\neq 0$. Da $\mathcal{I}_1$ und $\mathcal{I}_2$ sich überlappen, werden einige der Positionen doppelt gezählt, so dass man

$$2(w + 1) - |\mathcal{I}_1 \cap \mathcal{I}_2|$$

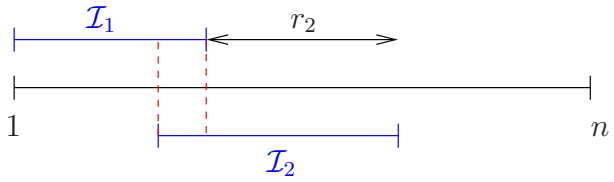
als untere Schranke erhält.

Dies kann man auf m Informationsmengen verallgemeinern:

Definition 12.20 Seien $\mathcal{I}_1, ..., \mathcal{I}_m$ Informationsmengen zum Code C. Der **relative Rang** r_j einer Informationsmenge $\mathcal{I}_j$ ist

$$r_j = k - \left|\mathcal{I}_j \cap \bigcup_{l=1}^{j-1} \mathcal{I}_l\right|,$$

d.h. k minus die Anzahl von Positionen in $\mathcal{I}_j$, die schon in einem $\mathcal{I}_l$ mit $l < j$ vorkommen.



Bemerkung 12.21 Der Greedy-Algorithmus 12.18 liefert eine monoton fallende Folge von relativen Rängen r_j. Optimal ist

$$r_1 = ... = r_{\lfloor \frac{n}{k} \rfloor} = k, \quad r_m = n \bmod k,$$

d.h. alle Informationsmengen (bis auf $\mathcal{I}_m$, wenn k kein Teiler von n ist) sind disjunkt.

Lemma 12.22 Verwendet man die Generatormatrizen $G_1, ..., G_m$ der Informationsmengen $\mathcal{I}_1, ..., \mathcal{I}_m$ und codiert damit alle Informationsworte $\boldsymbol{i}$ vom Gewicht $\mathrm{wgt}(\boldsymbol{i}) \leq w$, so ist die untere Schranke für das Gewicht der noch nicht aufgezählten Codeworte

$$d_{\text{low}} = \sum_{j=1}^{m} \max\{0, (w + 1) - (k - r_j)\}.$$

Die Matrix G_j trägt somit den Term $(w + 1) - (k - r_j)$ zur unteren Schranke bei, falls $w + 1 \geq k - r_j =$ Anzahl der überlappenden Positionen.

Bemerkung 12.23 Anfangs trägt mindestens eine Generatormatrix zur unteren Schranke bei, später dann mehrere. Folglich ist die untere Schranke als Funktion von w stückweise linear und die Steigung nimmt zu, da die Steigung abhängig von der Anzahl der Generatormatrizen mit $k - r_j < w$ ist.

Algorithmus 12.24 Gesamtalgorithmus zur Bestimmung des Minimalgewichts

Schematisch:

1. Bestimme eine Menge von systematischen Generatormatrizen und die zugehörige Folge r_j von relativen Rängen. (r_j sollte möglichst groß sein $\rightsquigarrow$ randomisierter Algorithmus: Permutation der Spalten)

2. Bestimme die Folge der unteren Schranken in Abhängigkeit von w und j:

$$d_{\text{low}}(w, j) = \sum_{i=1}^{j} \max\{0, (w+1) - (k - r_i)\}.$$

   Anhand der aktuellen oberen Schranke kann man vorhersagen, wann der Algorithmus spätestens terminiert, d.h. bei welchem Gewicht $w = w_0$. Damit kann entschieden werden, ob eine Matrix G_j überhaupt einen Beitrag zur unteren Schranke liefern wird (Generatormatrizen mit $k - w_0 > r_j$ können weggelassen werden).

3. Eventuell weitere Vorberechnungen, um die Codeworte schneller aufzuzählen (mehr dazu in Kapitel 15).

4. Für alle Gewichte $w = 1, 2, ...$:
   Für alle Generatormatrizen $G_j = G_1, G_2, ..., G_m$:

   - Zähle jedes Informationswort $\boldsymbol{i}$ mit Gewicht $\text{wgt}(\boldsymbol{i}) = w$ auf und bestimme das Gewicht des zugehörigen Codeworts $\boldsymbol{i}G_j$.
   - Bestimme das neue Minimum der Gewichte der aufgezählten Worte, d.h.

$$d_{\text{up}} := \min_{\boldsymbol{i}:\text{wgt}(\boldsymbol{i})=w} \{d_{\text{up}}, \text{wgt}(\boldsymbol{i}G_j)\}$$

     Abbruch, falls $d_{\text{up}} \leq d_{\text{low}}(w, j-1)$.
   - Falls d_{up} geändert wurde, können ggf. weitere Matrizen wie in Schritt 2 verworfen werden.

   Bei Terminierung ist $d = d_{\text{up}}$.

13 Zyklische und quasizyklische Codes

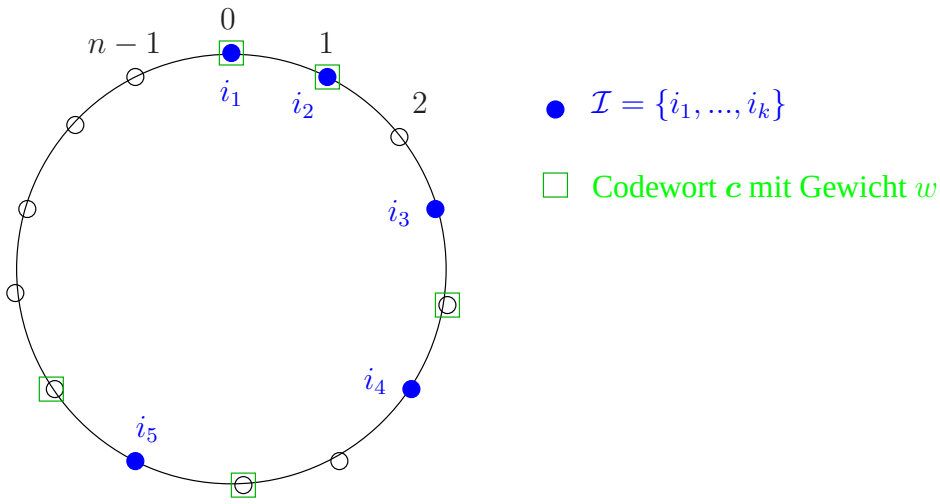
Definition 13.1 Ein linearer Code $C[n,k,d,q]$ heißt **zyklisch**, falls für jedes Codewort $\boldsymbol{c} = (c_0, ..., c_{n-1}) \in \mathbb{F}_q^n$ auch der zyklisch verschobene Vektor $\boldsymbol{c'} = (c_{n-1}, c_0, ..., c_{n-2})$ ein Codewort ist.

Bemerkung 13.2 Ist $\mathcal{I} = \{i_1, ..., i_k\}$ eine Informationsmenge, so auch

$$\mathcal{I}' = \{i_1 + l, ..., i_k + l\}$$

(Rechnung modulo n). Für k aufeinanderfolgende Positionen gibt es stets $\lfloor \frac{n}{k} \rfloor = m$ disjunkte Informationsmengen. Ist etwa $\mathcal{I}_1 = \{1, ..., k\}$, so ist $\mathcal{I}_2 = \{k+1, ..., 2k\}$ usw. . Will man in Algorithmus 12.24 die Codeworte ermitteln, die durch Codierung eines Informationsworts $\boldsymbol{i}$ durch die Generatormatrizen dieser Informationsmengen entstehen, so reicht es, ein Codewort $\boldsymbol{c} = \boldsymbol{i}G_1$ zu bestimmen und die anderen durch zyklische Verschiebung aus $\boldsymbol{c}$ zu erhalten, da eine Permutation das Hamming-Gewicht nicht ändert. Es können also m Informationsmengen genutzt werden, obwohl nur einmal codiert wird.

Zyklisch verschobene Informationsmenge:



Lasse das Codewort $\boldsymbol{c}$ fest und variiere die Informationsmenge durch zyklisches Verschieben: Jedes Stelle von $\boldsymbol{c}$ kommt in k verschiedenen Informationsmengen vor.
$\Rightarrow$ die Gesamtzahl der Stellen mit einem Eintrag $\neq 0$ in den k verschiedenen Informationsmengen ist $k\mathrm{wgt}(\boldsymbol{c})$
$\Rightarrow$ im Mittel ist die Anzahl der Stellen in einer Informationsmenge, an denen $\boldsymbol{c}$ einen Eintrag $\neq 0$ hat, $\frac{k}{n}\mathrm{wgt}(\boldsymbol{c})$
$\Rightarrow$ es gibt mindestens eine Informationsmenge, die höchstens $\lfloor \frac{k}{n}\mathrm{wgt}(\boldsymbol{c}) \rfloor$ Stellen enthält, an denen $\boldsymbol{c}$ einen Eintrag $\neq 0$ hat (denn hätten alle Informationsmengen mehr solche Stellen, so müssten sie auch im Durchschnitt mehr als $\frac{k}{n}\mathrm{wgt}(\boldsymbol{c})$ haben).

Daraus folgt:

Lemma 13.3 Für jedes Codewort $\boldsymbol{c}$ eines zyklischen Codes $C[n,k]$ gibt es eine Informationsmenge $\mathcal{I}'$ (mit Generatormatrix G'), so dass das Gewicht des zugehörigen Informationswortes $\boldsymbol{i}$ mit $\boldsymbol{i}G' = \boldsymbol{c}$ höchstens $\lfloor \frac{k}{n}\mathrm{wgt}(\boldsymbol{c})\rfloor$ ist.

Lemma 13.4 Hat man bei einem zyklischen linearen Code C bzgl. einer festen systematischen Generatormatrix G alle Informationsworte $\boldsymbol{i} \in \mathbb{F}_q^k$ mit Gewicht $\mathrm{wgt}(\boldsymbol{i}) \leq w$ codiert und zyklisch verschoben, so erhält man als untere Schranke für das Gewicht der noch nicht aufgezählten Vektoren

$$d_{\text{low}} = \left\lceil \frac{n}{k}(w+1) \right\rceil .$$

Definition 13.5 Wir geben zwei verschiedene Definitionen für *quasizyklische Codes* an. Es sei l ein Teiler von n.

1. Der Code C heißt **quasizyklisch**, wenn mit $\boldsymbol{c} \in C$ auch der um l Positionen verschobene Vektor

   $$\boldsymbol{c}^{(l)} = (c_{n-l}, c_{n-l+1}, ..., c_{n-1}, c_0, ..., c_{n-l-1})$$

   ein Codewort ist.

2. Der Code C heißt **quasizyklisch**, wenn mit $\boldsymbol{c} \in C$ auch der simultan in m Blöcken der Größe $l = \frac{n}{m}$ zyklisch verschobene Vektor $\tilde{\boldsymbol{c}}$ ein Codewort ist.

   $$\tilde{\boldsymbol{c}} = (c_{l-1}, c_0, ..., c_1, c_{l-2} | c_{2l-1}, c_l, c_{l+1}, ..., c_{2l-2} | ...).$$

Wir werden uns im Folgenden an die zweite Definitionsvariante halten.

Bemerkung 13.6 Die Generatormatrix G eines quasizyklischen Codes hat folgende Form:

$$G = \left(\begin{array}{cccc|cccc|c} c_0 & c_1 & \dots & c_{l-1} & c_l & c_{l+1} & \dots & c_{2l-1} & \dots \\ c_{l-1} & c_0 & \dots & c_{l-2} & c_{2l-1} & c_l & \dots & c_{2l-2} & \dots \\ & & \ddots & & & & \ddots & & \ddots \end{array} \right)$$

Eine Matrix dieser Form hat nicht unbedingt vollen Rang. In dieser Anordnung kann relativ leicht getestet werden, ob der erzeugte Code quasizyklisch mit m Blöcken ist:

- Bestimme die Teiler von n.
- Teste für jeden Teiler, ob das simultane Verschieben der Zeilen der Generatormatrix in $m = \frac{n}{l}$ Blöcken eine Generatormatrix für denselben Code liefert:
  Betrachte G^σ, wobei $\sigma = (1...l)(l+1...2l)\cdots(n-l+1...n)$ eine Permutation mit m Zykeln der Länge l ist. Verwende anschließend Gauß-Elimination, um zu testen, ob G und G^σ ähnlich sind.

Verallgemeinerung: Sei $\tau \in \mathsf{S}_n$ eine beliebige Permutation der Stellen des Codes mit Zykeln der Länge l_i, die den Code invariant lässt. Wende die Abschätzung für das mittlere Gewicht in einem permutierten „Informationsfenster“ auf die Bahnen der von τ erzeugten Gruppe an. Allgemein gilt: Ist eine Informationsmenge $\mathcal{I} = \{i_1, ..., i_k\}$ vollständig in der Bahn $\mathcal{I}^{\langle\tau\rangle}$ der zyklischen Gruppe $\langle\tau\rangle$ enthalten, so gilt die untere Schranke

$$\left\lceil (w+1)\frac{\tilde{n}}{k} \right\rceil$$

für das Gewicht der noch nicht aufgezählten Codeworte. Dabei ist $\tilde{n} = |\mathcal{I}^{\langle\tau\rangle}|$.

Bemerkung 13.7 Bei quasizyklischen Codes hat man m Blöcke der Länge l. Nimmt man zusätzlich $l \geq k$ und vollen Rang für jeden Block an, so existiert zu jedem Block eine Generatormatrix, die in diesem Block die Einheitsmatrix enthält.

$$\begin{aligned} G_1 &= (I_k, A_{11}|A_{12}|...|A_{1m}) \\ G_2 &= (A_{21}|I_k, A_{22}|...|A_{2m}) \\ &\vdots \\ G_m &= (A_{m1}|A_{m2}|...|I_k, A_{mm}) \end{aligned}$$

Wenn man die quasizyklische Struktur des Codes ausnutzen kann, benötigt man in der Regel weniger Generatormatrizen und hat eine verbesserte untere Schranke

$$m \left\lceil (w+1)\frac{n}{mk} \right\rceil,$$

im Vergleich zu $m(w+1)$ bei nichtzyklischen Codes.

Bemerkung 13.8 („weniger ist mehr“)
Ein zyklischer Code der Länge $n = ml$ ist auch ein quasizyklischer Code mit m Blöcken der Länge l. Ersetze den zyklischen Automorphismus $\sigma = (1\ 2\ ...\ n)$ durch $\tau := \sigma^m$:

$$\tau = (1\ m+1\ 2m+1\ ...)(2\ m+2\ 2m+2\ ...)\cdots(m\ 2m\ 3m\ ...)$$

Vergleiche die unteren Schranken:

$$\text{zyklischer Code: } d_{\text{low}}^{\text{cyc}} = \left\lceil (w+1)\frac{n}{k} \right\rceil$$

$$\text{quasizyklischer Code: } d_{\text{low}}^{\text{qc}} = m \left\lceil (w+1)\frac{n}{km} \right\rceil \geq m(w+1)\frac{n}{km}$$

Beispiel 13.9 Betrachte $C[180, 30, d]$, $m = 5$ und $l = 36$:

$$\begin{aligned} d_{\text{low}}^{\text{cyc}} &= \left\lceil (w+1)\frac{180}{30} \right\rceil = 6(w+1) \\ d_{\text{low}}^{\text{qc}} &= 5 \cdot \left\lceil (w+1)\frac{6}{5} \right\rceil \geq 6(w+1). \end{aligned}$$

Zusammenfassung: Das Ausnutzen von Struktureigenschaften kann bessere untere Schranken liefern. Der Vergleich des a priori berechenbaren Aufwands, um eine untere Schranke d_{low} für die Minimaldistanz zu bestimmen, liefert Anhaltspunkte dafür, welche Strategie gewählt werden sollte. Ein weiteres Kriterium ist der relative Rang der einzelnen Generatormatrizen. Weiterführende Informationen findet man bei Grassl [5].

14 Kongruenzen für die Gewichte im Code

Sind alle Gewicht der Codeworte eines linearen Codes C ein Vielfaches von $t \in \mathbb{N}$, $t > 1$, so kann man die untere Schranke stets auf das nächste Vielfache von t aufrunden (falls sie nicht schon selbst ein Vielfaches von t ist).

Beispiel 14.1 Eine triviale Situation: Der Code wiederholt einen Block t-fach, d.h.

$$G = (\underbrace{G_0|G_0|...|G_0}_{t\text{-fach}}).$$

Definition 14.2 Ein Binärcode C heißt

- **even**, falls alle Gewichte der Codeworte durch 2 teilbar sind
- **doubly-even**, falls alle Gewichte durch 4 teilbar sind
- **singly-even**, falls alle Gewichte gerade sind und mindestens ein Gewicht $\mathrm{wgt}(\boldsymbol{c}) \equiv 2 \bmod 4$ ist (also nicht alle Gewichte durch 4 teilbar).

Lemma 14.3 Ein linearer Binärcode C ist genau dann even, wenn die Zeilen der Generatormatrix gerades Gewicht haben.

BEWEIS: Seien $\boldsymbol{v}, \boldsymbol{w} \in C$ mit geradem Gewicht. Œ gelte

$$\begin{aligned} \boldsymbol{v} &= (1...1|1...1|0...0|0...0) \\ \boldsymbol{w} &= (0...0|1...1|1...1|0...0) \\ \boldsymbol{v}+\boldsymbol{w} &= (1...1|0...0|1...1|0...0) \end{aligned}$$

Es gilt also

$$\mathrm{wgt}(\boldsymbol{v}+\boldsymbol{w}) = \mathrm{wgt}(\boldsymbol{v}) + \mathrm{wgt}(\boldsymbol{w}) - 2\mathrm{wgt}(\boldsymbol{v} \bullet \boldsymbol{w}),$$

wobei $\bullet$ die komponentenweise Multiplikation zweier Vektoren bezeichnet. Haben also alle Basisvektoren von C (also die Zeilen der Generatormatrix) gerades Gewicht, so gilt dies auch für alle anderen Vektoren aus C. ■

Frage: Wann sind die Gewichte aller Worte eines Codes durch 4 teilbar, also der Code ein doubly-even Code?

Wir führen ein „Skalarprodukt“ auf dem Vektorraum $\mathbb{F}_q^n$ ein:

$$\langle \boldsymbol{v}|\boldsymbol{w}\rangle := \sum_{i=1}^{n} v_i w_i \ \in \mathbb{F}_q,$$

mit $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{F}_q^n$.
Vorsicht: Über $\mathbb{F}_q$ gilt nicht:

$$[\langle \boldsymbol{v}|\boldsymbol{v}\rangle = 0] \Rightarrow [\boldsymbol{v} = 0].$$

Beispiel 14.4 Sei p prim, $q = p^m$ und $\boldsymbol{v} = (1, ..., 1) \in \mathbb{F}_q^p$. Dann gilt

$$\langle \boldsymbol{v} | \boldsymbol{v} \rangle = \sum_{i=1}^{p} v_i v_i = \sum_{i=1}^{p} 1 = 0.$$

Definition 14.5 Sei C ein linearer Code über $\mathbb{F}_q$. Dann ist der **duale Code**

$$C^\perp = \{\boldsymbol{v} \in \mathbb{F}_q^n : \forall \boldsymbol{c} \in C : \langle \boldsymbol{c} | \boldsymbol{v} \rangle = 0\}.$$

Bemerkung 14.6 Wird $C[n, k, d, q]$ von der Generatormatrix G erzeugt, so wird $C^\perp[n, k, d, q]$ von der Prüfmatrix H mit $GH^\top = 0$ erzeugt (siehe Strategie 12.3).

Definition 14.7 Ein linearer Code C heißt **selbstorthogonal**, wenn $C \subseteq C^\perp$ gilt, d.h. für alle $\boldsymbol{c}, \boldsymbol{c'} \in C$ gilt

$$\langle \boldsymbol{c} | \boldsymbol{c'} \rangle = \sum_{i=1}^{n} c_i c_i' = 0.$$

In diesem Fall ist $k \leq \frac{n}{2}$.

Definition 14.8 Ein linearer Code C heißt **selbstdual**, wenn $C = C^\perp$ gilt. In diesem Fall ist $n = 2k$.

Satz 14.9 Ein selbstorthogonaler Binärcode C ist doubly-even, falls alle Zeilen der Generatormatrix ein durch 4 teilbares Gewicht haben.

BEWEIS: Aus $C \subseteq C^\perp$ folgt, dass je zwei Codeworte $\boldsymbol{v}, \boldsymbol{w} \in C$ an einer geraden Anzahl von Positionen beide gleich 1 sind, d.h.

$$\mathrm{wgt}(\boldsymbol{v} \bullet \boldsymbol{w}) = 0 \bmod 2.$$

Setzen wir nun $\mathrm{wgt}(\boldsymbol{v}), \mathrm{wgt}(\boldsymbol{w}) \equiv 0 \bmod 4$ voraus, so folgt

$$\mathrm{wgt}(\boldsymbol{v} + \boldsymbol{w}) = \mathrm{wgt}(\boldsymbol{v}) + \mathrm{wgt}(\boldsymbol{w}) - \underbrace{2 \underbrace{\mathrm{wgt}(\boldsymbol{v} \bullet \boldsymbol{w})}_{\equiv 0 \bmod 2}}_{\equiv 0 \bmod 4} \equiv 0 \bmod 4$$

analog zu Lemma 14.3. ■

Algorithmus 14.10 Test auf doubly-even

1. Sind alle Gewichte der Zeilen der Generatormatrix G durch 4 teilbar?
2. Gilt $C \subseteq C^\perp$, d.h. ist $GG^\top = 0$?

Betrachten wir nun speziell die Codes über $\mathbb{F}_3$ und $\mathbb{F}_4$.

Lemma 14.11 Ist C ein selbstorthogonaler linearer Code über $\mathbb{F}_3$, so sind die Gewichte aller Codeworte durch 3 teilbar.

BEWEIS: Es ist $\mathbb{F}_3 = \{-1, 0, 1\}$. Wegen $C \subseteq C^\perp$ ist insbesondere $\langle \boldsymbol{c}|\boldsymbol{c}\rangle = 0$ für alle Codeworte $\boldsymbol{c}$. Aus

$$0 = \langle \boldsymbol{c}|\boldsymbol{c}\rangle = \sum_{i=1}^{n} c_i^2 = |\{i : c_i \neq 0\}| \bmod 3 = \mathrm{wgt}(\boldsymbol{c}) \bmod 3$$

folgt die Behauptung. ■

Ziel: Teste, ob ein gegebener Code über $\mathbb{F}_4$ even ist.

Aus Bemerkung 9.2 wissen wir, dass

$$\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\} \cong \mathbb{F}_2[X]/\langle X^2 + X + 1\rangle$$

gilt. Es ist also $1 + \alpha + \alpha^2 = 0$.

$\mathbb{F}_4$ ist ein $\mathbb{F}_2$-Vektorraum der Dimension 2. Also identifizieren wir $\mathbb{F}_4 \cong \mathbb{F}_2^2$:

$\mathbb{F}_4$	$\mathbb{F}_2^2$	mit Prüfbit
0	$(0,0)$	$(0,0,0)$
1	$(0,1)$	$(0,1,1)$
α	$(1,0)$	$(1,0,1)$
α^2	$(1,1)$	$(1,1,0)$

Fügen wir zu der Darstellung in $\mathbb{F}_2^2$ noch ein **Prüfbit** hinzu, so dass immer eine gerade Anzahl von Einsen vorliegt, so erhalten wir den Code $C_2[3, 2, 2, 2]$. Die dadurch definierte Abbildung $\mathbb{F}_4 \to C_2$ ist $\mathbb{F}_2$-linear und kann zu einer Abbildung

$$\Phi : \mathbb{F}_4^n \to \mathbb{F}_2^{3n}$$

von Vektoren der Länge n über $\mathbb{F}_4$ erweitert werden, indem jede Komponente auf ein Binärwort der Länge 3 abgebildet wird. Für $\boldsymbol{v} \in \mathbb{F}_4^n$ gilt:

$$\mathrm{wgt}(\Phi(\boldsymbol{v})) = 2\mathrm{wgt}(\boldsymbol{v}).$$

Damit erhalten wir:

Lemma 14.12 Die Gewichte in einem linearen Code C über $\mathbb{F}_4$ sind gerade, falls alle Gewichte im Binärcode $\Phi(C)$ durch 4 teilbar sind.

Frage: Wie sieht die Generatormatrix G_0 von $\Phi(C)$ aus?

C_2 hat die Generatormatrix

$$G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

$\mathbb{F}_4$ kann auch als Algebra über $\mathbb{F}_2$ aufgefasst werden, d.h. zu jedem $\beta \in \mathbb{F}_4$ gibt es eine Matrix $M_\beta \in \mathbb{F}_2^{2\times 2}$ (dies entspricht der linearen Abbildung durch „Multiplikation mit festem β“, vgl. Satz 9.11). So erhalten wir folgende Darstellung:

$$0 \cong \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad 1 \cong \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\alpha \cong \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad \alpha^2 \cong \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Sei G eine Generatormatrix von $C[n, k, d, 4]$.

- Ersetze jeden Eintrag g_{ij} in G durch $M_{g_{ij}}$ und erhalte so $\tilde{G} \in \mathbb{F}_2^{2k\times 2n}$.
- Multipliziere mit der Matrix

$$\begin{pmatrix} G_2 & & & \\ & G_2 & & \\ & & \ddots & \\ & & & G_2 \end{pmatrix} = I_n \otimes G_2.$$

  Erhalte so die Generatormatrix $G_0 \in \mathbb{F}_2^{2k\times 3n}$ von $\Phi(C)$.

Man erhält dies auch durch direkte Berechnung:

$$0 \cong 0 \cdot G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad 1 \cong I_2 G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$
$$\alpha \cong M_\alpha G_2 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$
$$\alpha^2 \cong M_{\alpha^2} G_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$

und ersetze jeden Eintrag g_{ij} in G durch $M_{g_{ij}} G_2$.

15 Effizientes Aufzählen von Codeworten

Gegeben:

- Eine Generatormatrix G mit Zeilen $\boldsymbol{g}_1, ..., \boldsymbol{g}_k$.
- Das Gewicht w der zu codierenden Informationsworte $\boldsymbol{i} \in \mathbb{F}_q^k$.

Generiere alle Codeworte $\boldsymbol{i}G$ für alle $\boldsymbol{i} \in \mathbb{F}_q^k$ mit $\mathrm{wgt}(\boldsymbol{i}) = w$. Œ sei $\boldsymbol{i}$ „normiert“, d.h. der erste Eintrag ungleich 0 ist 1.

Vorgehen:

- Generiere alle w-Teilmengen von $\{\boldsymbol{g}_1, ..., \boldsymbol{g}_k\}$.
- Bilde für jede w-Teilmenge alle Linearkombinationen

$$\sum_{j=1}^{w} \gamma_j \boldsymbol{g}_{i_j}$$

  mit $\gamma_j \neq 0$ und Œ $\gamma_1 = 1$. Durchlaufe dafür alle w-Tupel $(\gamma_1, ..., \gamma_w)$ über der Menge $\mathbb{F}_q \backslash \{0\}$.

Bemerkung 15.1 Bei der Bestimmung der Teilmenge $\{\boldsymbol{g}_{i_1}, ..., \boldsymbol{g}_{i_w}\}$ sowie des nächsten Tupels $(\gamma_1, ..., \gamma_w)$ kann man gleichzeitig die entsprechenden Vielfachen der Vektoren $\boldsymbol{g}_{i_j}$ bestimmen und die Summe aktualisieren.

Wir skizzieren zwei Algorithmen für diese Vorgehen, eine genauere Beschreibung findet man bei Knuth [6].

Algorithmus 15.2 Verwende einen *verallgemeinerten Gray-Code* zur Generierung der Tupel $(\gamma_1, ..., \gamma_w)$. Ein **Gray-Code** ist eine Sequenz aller Tupel, in der sich je zwei aufeinanderfolgende Elemente in genau einer Position unterscheiden.

Algorithmus 15.3 Revolving door
Aufzählen von w-Teilmengen einer Menge $\{1, ..., k\}$. Analog zum Gray-Code wird in jedem Schritt genau ein Element der aktuellen Teilmenge ausgetauscht: „Einer geht, einer kommt“, daher der engl. Name für diesen Algorithmus.

$$\sum_{j=0}^{w} \boldsymbol{g}_{i_j} + (\underbrace{\boldsymbol{g}_\nu}_{\text{„kommt“}} - \underbrace{\boldsymbol{g}_{i_0}}_{\text{„geht“}}).$$

Bei einer Vorberechnung der Differenzen $\boldsymbol{g}_i - \boldsymbol{g}_j$ ist nur noch eine Vektoraddition notwendig, um $\boldsymbol{g}_j$ durch $\boldsymbol{g}_i$ zu ersetzen.

Literatur

[1] BERLEKAMP, MCELIECE, VAN TILBORG:
On the Inherent Intractability of Certain Coding Problems.

[2] BUTLER:
Fundamental algorithms for permutation groups, Springer.

[3] BOSMA, CANNON:
An Introduction to Permutation Group Algorithms,
iaks-www.ira.uka.de/home/grassl/Academia/Algo-Gruppen-Codes/references/cwi.ps.gz

[4] EGNER, PÜSCHEL:
Solving Puzzles related to Permutation Groups.

[5] GRASSL:
Searching for Good Linear Codes,
iaks-www.ira.uka.de/home/grassl/Academia/Algo-Gruppen-Codes/references/BKLC_Grassl.ps

[6] KNUTH:
The Art of Computer Programming, Pre-Fascicle 3A: Generating all Combinations,
www-cs-faculty.stanford.edu/∼knuth/news.html

[7] KÖHLER:
Gruppentheoretische Algorithmen,
www.uni-giessen.de/∼gc1079/gruppen/gruppen.ps

[8] MACWILLIAMS, SLOANE:
The Theory of Error-Correcting Codes, North Holland.

[9] MURRAY:
The Schreier-Sims-Algorithm,
www.win.tue.nl/∼smurray/research/essay.pdf

[10] MURRAY, O'BRIEN:
Selecting base points for the Schreier-Sims algorithm for matrix groups,
www.win.tue.nl/∼smurray/research/base-points.pdf

[11] SERESS:
Permutation group algorithms, Cambridge University Press.

[12] SIMS:
Computation with Permutation Groups.

[13] VARDY:
The Intractability of Computing the Minimum Distance of a Code.

Index

$C[n,k,d,q]$, 40
$C^{\perp}[n,k,d,q]$, 54
$N \trianglelefteq G$, 4
$[G : G_m]$, 6
$\mathbb{F}_q$, 35
$\mathrm{GL}_n(\mathbb{F}_q)$, 38
$\langle x \rangle$, $\langle S \rangle$, 4
S_n, 5
m^g, m^G, 6

abelsche Gruppe, 3
affine Transformation, 3
Algorithmus
    Bahnberechnung, 10
    Basiswahl, 29, 39
    Berechnung von τ, 12
    Codeworte aufzählen, 57
    doubly-even Test, 54
    Enthaltensein in der Bahn, 11
    Faktorisierung, 17
    Informationsmengen bestimmen, 47
    Membership Test, 18
    Minimalgewicht berechnen, 45, 49
    Minimalgewicht berechnen (naiv), 44
    Revolving door, 57
    Schreier-Sims, 20
    Transposition von Basiselementen, 25
    Trembling, 32
    Word Strip, 18
Aufzählen der Codeworte, 57

Bahn, 5
    Berechnung, 10
    Graph, 10
Bahnbilanz, 6
Basis, 14
    geschickte Wahl, 29
Basiswechsel, 24
Begleitmatrix, 39
Blockcode
    linearer, 40
Brezel, 33

Cayley-Graph, 8
    partieller, 9
charakteristisches Polynom, 38
Code, 40
    allgemeiner, 40
    doubly-even, 53
    dual, 54
    even, 53
    Gray-, 57
    linearer, 40
    quasizyklisch, 51
    selbstdual, 54
    selbstorthogonal, 54
    singly-even, 53
    zyklisch, 50
Codewort, 42
    aufzählen, 57
Codierung
    Abbildung, 42
    systematische, 44
Codierungsabbildung, 42

doubly-even Code, 53
dualer Code, 54

Eigenraum, 39
    verallgemeinerter, 39
Eigenvektor, 38
endlicher Körper, 35
Erweiterungskörper, 35
Erzeugermenge, 4
    starke, 14
erzeugte Untergruppe, 4
even Code, 53

Faktorisierung, 28
    Algorithmus, 17
    Freiheitsgrade, 33
    Länge, 28
faules Paar, 29
Fehler, 42
Fehlersyndrom, 43
Fixgruppe, 6
freie Gruppe, 4, 28
Frobenius-Automorphismus, 35

Generatormatrix, 42
    systematische, 44
Graph
    Bahn, 10
    Cayley-, 8

Gray-Code, 57
Gruppe, 3
    abelsche, 3
    Fix-, 6
    freie, 4, 28
    Matrix-, 38
    Permutations-, 5
    symmetrische, 5
    Wort-, 4
Gruppenoperation, 4
    Bahn, 5
    Kern, 6
    treu, 6

Hamming-Distanz, 40
Hamming-Gewicht, 43
Hauptraum, 39
Homomorphismus, 3
    kanonischer, 28, 29

Index, 4
Informationsmenge, 44
Informationswort, 42
inverses Element, 3
Isomorphismus, 3

Körper
    endlicher, 35
    Erweiterungs-, 35
    Prim-, 35
kanonischer Homomorphismus, 28, 29
Kern, 6
Konjugation, 7
konjugierte Paar, 29
Kontrollmatrix (siehe Prüfmatrix), 42
kurzes Paar, 29

Länge einer Faktorisierung, 28
LDF, 31
linearer Blockcode, 40

Matrixgruppe, 38
Minimaldistanz, 40, 43
Minimalgewicht, 43
    berechnen, 44, 45, 49

Nebenklasse, 4
    Index, 4
neutrales Element, 3
Normalform, 38
Normalisator, 24
Normalteiler, 4

Operation (siehe Gruppenoperation), 4
Orbit (siehe Bahn), 5
Orbit Graph Completion, 31
Ordnung, 4

Paar, 29
    faul, 29
    konjugiert, 29
    kurz, 29
    Ordnung, 29
Paarordnung, 29
Permutationsdarstellung, 7
Permutationsgruppe, 5
    Basis, 14
Prüfbit, 55
Prüfmatrix, 42, 54
primitives Element, 35
Primkörper, 35
Projektion
    kanonische, 12
punktweiser Stabilisator, 14

quasizyklischer Code, 51

random walk, 32
Rang
    relativer, 48
Rate, 41
redundant, 14
relativer Rang, 48
Revolving door, 57
Rubik's Cube, 5

Satz
    Lagrange, 4
    Schreier, 13
Schreier-Baum, 11
Schreier-Generator, 13, 16, 20
Schreier-Sims-Algorithmus, 20
Schreier-Vektor, 11
selbstdualer Code, 54
selbstorthogonaler Code, 54
simultane Gleichungen, 23
singly-even Code, 53
SLP, 27
Stabilisator, 6
    Kette, 14
    punktweiser, 14
Stabilisatorkette, 14

starke Erzeugermenge, 14
straight line program, 27
symmetrische Gruppe, 5
systematische Codierung, 44
systematische Generatormatrix, 44

Transversale, 12
Trembling, 32
treu, 6

Untergruppe, 3
    erzeugte, 4

verallgemeinerter Eigenraum, 39
Verteilungsfunktion der Wortlänge, 31

weniger ist mehr, 52
Wortgruppe, 4
Wortlänge, 17, 28
    Verteilungsfunktion, 31

Zech-Logarithmus, 36
Zykelschreibweise, 5
zyklisch, 4
zyklischer Code, 50