Diskrete Mathematik II

Alexander May

Fakultät für Mathematik
Ruhr-Universität Bochum

Sommersemester 2009

Organisatorisches

- Vorlesung: **Mo 12-14** in HIA , **Di 09-10** in NA 3/99
  (3+1 SWS, 6.75 CP)
- Übung: **Di 10-12** in ND2/99
  - Assistent: **Maike Ritzenhofen**, Korrektor: **Alexander Meurer**
  - Übung ist zweiwöchentlich: gerade/ungerade Woche
  - Übungsaufgaben werden korrigiert.
  - Gruppenabgaben bis 3 Personen
  - Bonussystem:
    1/3-Notenstufe für 50%, 2/3-Notenstufe für 75%
    Gilt nur, wenn man die Klausur besteht!
  - Musterlösungen
- Klausur: voraussichtlich am Mi. den 26. August 2009

Themengebiete

1. Kodierungstheorie
   - Komprimierende Codes
   - Beispiel Anwendungen: Kommunikation (Mobilfunk, Internet), Speicher (MP3)
   - Fehlererkennende Codes
   - Ausfalltolerante Codes
   - Beispiel Anwendungen: Mobilfunk, Internet, CD, Secret Sharing, Kryptosystem
2. Komplexitätstheorie
   - Klassen P und NP
   - Reduktionen
   - Anwendung: Sicherheitsbeweise in der Kryptographie
3. Algorithmische Zahlentheorie
   - Quadratische Reste
   - Beispiel Anwendungen: Zufallszahlengenerator, Identity-Based Encryption

Weiterführende Referenzen

- Steven Roman, “Introduction to Coding and Information Theory”, Springer Verlag, 1996
- Michael R. Garey, David S. Johnson, “Computers and Intractability”, Freeman, 2000
- J. Blömer, “Einführung in Algorithmen und Komplexität”, Vorlesungsskript Universität Paderborn, 2002
- N. Koblitz, “A Course in Number Theory and Cryptography”, Springer Verlag, 1994

Unser Modell

- Shannon 1948: Informationstheorie und Mathematik der Kommunikation
- Hamming 1950: Erste Arbeit über fehlerkorrigierende Codes

Modell:
Sender $\longrightarrow$ Kodierer $\longrightarrow$ Kanal $\longrightarrow$ Dekodierer $\longrightarrow$ Empfänger

- Kanal ist bandbreitenbeschränkt (Kompression)
- Kanal ist fehleranfällig (Fehlerkorrektur)
  - Bits können ausfallen: $0 \longrightarrow \epsilon$, $1 \longrightarrow \epsilon$
  - Bits können kippen: $0 \longrightarrow 1$, $1 \longrightarrow 0$

Motivierendes Bsp: Datenkompression

Szenario:

- Kanal ist **fehlerfrei**.
- Übertragen gescannte Nachricht:
  Wahrscheinlichkeiten: 99% weißer, 1% schwarzer Punkt.
- Weiße Punkte erhalten Wert 0, schwarze Wert 1.

Kodierer:

- Splitten Nachricht in Blocks der Größe 10.
- Wenn Block x=0000000000, kodiere mit 0, sonst mit 1x.
- 1 dient als Trennzeichen beim Dekodieren.

Dekodierer:

- Lese den Code von links nach rechts.
- Falls 0, dekodiere 0000000000.
- Falls 1, übernehme die folgenden 10 Symbole.

Erwartete Codelänge

Sei $q := \text{Ws}[\text{Block ist } 0000000000] = (0.99)^{10} \geq 0.9$.
Sei Y Zufallsvariable für die Codewortlänge eines 10-Bit Blocks:

$$E[Y] = \sum_{y \in \{0,1x\}} |y| \cdot \text{Ws}(Y = y) = 1 \cdot q + 11 \cdot (1 - q) = 11 - 10q.$$

- D.h. erwartete Länge der Kodierung eines 10-Bit Blocks ist
$$11 - 10q \leq 2 \text{ Bit}.$$
- Datenkompression der Nachricht auf 20%.
- Können wir noch stärker komprimieren?
- Entropie wird uns Schranke für Komprimierbarkeit liefern.

Ausblick: fehlerkorrigierende Codes

Szenario: Binärer symmetrischer Kanal

- Bits 0,1 kippen mit Ws $p, p < \frac{1}{2}$ zu 1,0. (Warum $< \frac{1}{2}$?)
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Kodierer:

- Verdreifache jedes Symbol, d.h. $0 \mapsto 000$, $1 \mapsto 111$
- Repetitionscode der Länge 3.

Dekodierer:

- Lese den Code in 3er-Blöcken.
- Falls mindestens zwei Symbole 0 sind, dekodiere zu 0.
- Sonst dekodiere zu 1.

Ws Dekodierfehler

Symbol wird falsch dekodiert, falls mind. zwei der drei Bits kippen.

$$\begin{aligned} & \text{Ws}(\text{Bit wird falsch dekodiert}) \\ = \; & \text{Ws}(\text{genau 2 Bits kippen}) + \text{Ws}(\text{genau 3 Bits kippen}) \\ = \; & 3 * p^2 * (1-p) + p^3 = 3 * 10^{-2} * (1 - 10^{-1}) + 10^{-3} \end{aligned}$$

- Ohne Kodierung Fehlerws von 0.1.
- Mit Repetitionscode Fehlerws von ≈ 0.03.
- Nachteil: Kodierung ist dreimal so lang wie Nachricht.
- **Ziel:**
  Finde guten Tradeoff zwischen Fehlerws und Codewortlänge.

Ausblick: fehlertolerante Codes

Szenario: Binärer Ausfallkanal

- Bits 0,1 gehen mit Ws $p, p < \frac{1}{2}$ verloren, d.h. $0 \mapsto \epsilon$ bzw. $1 \mapsto \epsilon$.
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Kodierer: Repetitionscode der Länge 3.

Dekodierer:

- Lese den Code in 3er-Blöcken.
- Falls 3er-Block Zeichen $x \in \{0, 1\}$ enthält, Ausgabe x.

Fehler beim Dekodieren: Alle drei Symbole gehen verloren.

- $\text{Ws}(\text{Bit kann nicht dekodiert werden}) = p^3 = 0.001$.
- Fehlerws kleiner beim Ausfallkanal als beim sym. Kanal.

Definition Code

- Alphabet $A = \{a_1, \ldots, a_n\}$, Menge von Symbolen a_i
- Nachricht $m \in A^*$

Definition Code

Sei A ein Alphabet. Eine (binäre) *Codierung C des Alphabets A* ist eine injektive Abbildung

$$\begin{aligned} C: \quad & A \to \{0,1\}^* \\ & a_i \mapsto C(a_i). \end{aligned}$$

Die *Codierung einer Nachricht* $m = a_{i_1} \ldots a_{i_\ell} \in A^*$ definieren wir als

$$C(m) = C(a_{i_1}) \ldots C(a_{i_\ell}) \quad \text{(Erweiterung von } C \text{ auf } A^*\text{)}.$$

Die Abbildung C heißt *Code*.

Bezeichnungen Code

- Die Elemente $c_i := C(a_i)$ bezeichnen wir als *Codeworte*.
- Wir bezeichnen sowohl die Abbildung von Nachrichten auf Codeworte als auch die *Menge der Codeworte* mit dem Buchstaben C.
- Falls $C \subseteq \{0,1\}^n$ spricht man von einem *Blockcode* der Länge n. In einem Blockcode haben alle Codeworte die gleiche Länge.

Entschlüsselbarkeit von Codes

Szenario: Datenkompression in fehlerfreiem Kanal

Definition eindeutig entschlüsselbar

Ein Code heißt eindeutig entschlüsselbar, falls jedes Element aus $\{0,1\}^*$ Bild höchstens einer Nachricht ist. D.h. die Erweiterung der Abbildung C auf A^* muss injektiv sein.

Definition Präfixcode

Ein Code $C = \{c_1, \ldots, c_n\}$ heißt Präfixcode, falls es keine zwei Codeworte $c_i \neq c_j$ gibt mit

$$c_i \text{ ist Präfix (Wortanfang) von } c_j.$$

Beispiel

	a_1	a_2	a_3
C_1	0	0	1
C_2	0	1	00
C_3	0	01	011
C_4	0	10	11

- C_1 ist kein Code, da $C_1 : A \to \{0,1\}^*$ nicht injektiv.
- C_2 ist nicht eindeutig entschlüsselbar, da $C_2 : A^* \to \{0,1\}^*$ nicht injektiv.
- C_3 ist eindeutig entschlüsselbar, aber kein Präfixcode.
- C_4 ist ein Präfixcode.

Präfixcodes sind eindeutig entschlüsselbar.

Satz: Präfixcode eindeutig entschlüsselbar

Sei $C = \{c_1, \ldots, c_n\}$ ein Präfixcode. Dann kann jede kodierte Nachricht $C(m)$ in Zeit $\mathcal{O}(|C(m)|)$ eindeutig zu m decodiert werden.

- Zeichne binären Baum
  - Kanten erhalten Label 0 für linkes Kind, 1 für rechtes Kind.
  - Codewort $c_i = c_{i_1} \ldots c_{i_k}$ ist Label des Endknoten eines Pfads von der Wurzel mit den Kantenlabeln $i_1, \ldots, i_n$
- **Präfixeigenschaft:** Kein einfacher Pfad von der Wurzel enthält zwei Knoten, die mit Codeworten gelabelt sind.
- Codewort c_i ist Blatt in Tiefe c_i

Algorithmus Dekodierung Präfix

Algorithmus Dekodierung Präfix

1. Lese $C(m)$ von links nach rechts.
2. Starte bei der Wurzel. Falls 0, gehe nach links. Falls 1, gehe nach rechts.
3. Falls Blatt mit Codewort $c_i = C(a_i)$ erreicht, gib a_i aus und iteriere.

Laufzeit: $\mathcal{O}(|C(m)|)$

Woher kommen die Nachrichtensymbole?

Modell

- *Quelle Q* liefert Strom von Symbolen aus A.
- Quellwahrscheinlichkeit: $\mathrm{Ws}(\text{ Quelle liefert } a_i) = p_i$
- Ws p_i ist unabhängig von der Zeit und vom bisher produzierten Strom (erinnerungslose Quelle)
- X_i: Zufallsvariable für das Quellsymbol an der i-ten Position im Strom, d.h.

$$\mathrm{Ws}(X_i = a_j) = p_j \quad \text{für } j = 1, \ldots, n \text{ und alle } i.$$

Ziel: Kodiere Elemente a_j mit großer Ws p_j mit kleiner Codewortlänge.

Kompakte Codes

Definition Erwartete Codewortlänge

Sei Q eine Quelle mit Alphabet $A = a_1, \ldots, a_n$ und Quellwahrscheinlichkeiten $p_1, \ldots, p_n$. Die Größe

$$E(C) := \sum_{i=1}^{n} p_i |C(a_i)|$$

bezeichne die erwartete Codewortlänge.

Definition Kompakter Code

Ein Code C heißt kompakt bezüglich einer Quelle Q, falls er *minimale erwartete Codewortlänge* besitzt.

Wann sind Codes eindeutig entschlüsselbar?

Definition Suffix

Sei C ein Code. Ein Folge $s \in \{0,1\}^*$ heißt Suffix in C falls

1. $\exists c_i, c_j \in C : c_i = c_j s$ oder
2. $\exists c \in C$ und einen Suffix s' in C: $s' = cs$ oder
3. $\exists c \in C$ und einen Suffix s' in C: $c = s's$.

- Bedingung 1: Codewort c_j lässt sich zu Codewort c_i erweitern.
- Bedingung 2: Codewort c lässt sich zu Suffix s' erweitern.
- Bedingung 3: Suffix s' lässt sich zu Codewort c erweitern.

Effiziente Berechnung von Suffixen

Algorithmus Berechnung Suffix

EINGABE: $C = \{c_1, \ldots, c_n\}$

1. Setze $S := \emptyset, T := \emptyset$.
2. Für alle $c_i, c_j \in C \times C$: Falls es ein $s \in \{0,1\}^*$ gibt mit $c_i = c_j s$, füge s in S und T ein.
3. Solange $T \neq \emptyset$
   1. Entferne ein beliebiges s' aus T.
   2. Für alle $c \in C$: Falls es ein $s \in \{0,1\}^* \setminus S$ gibt mit $s' = cs$ oder $c = s's$, füge s zu S und T hinzu.

AUSGABE: Menge S der Suffixe von C

Laufzeit Suffixberechnung

Laufzeit:

- Schritt 2: $\mathcal{O}(n^2)$ Codewortpaare
- Suffixlänge ist durch $\max_i\{|c_i|\}$ beschränkt.
- Es kann höchstens $n \cdot \max_i\{|c_i|\}$ Suffixe geben.
- Schritt 3: $\mathcal{O}(n^2 \cdot \max_i\{|c_i|\})$
- **Polynomiell in der Eingabelänge**: n, $max_i\{|c_i|\}$

Beispiele Suffixberechnung

- Code $C_2 = \{0, 1, 00\}$
  - Suffix $s_1 = 0$, denn $c_3 = c_1 0$.
- Code $C_3 = \{0, 01, 011\}$
  - Suffix $s_1 = 1$, denn $c_2 = c_1 1$.
  - Suffix $s_2 = 11$, denn $c_3 = c_1 11$.
- Code $C_4 = \{0, 10, 110\}$
  - Keine Suffixe, da Präfixcode.
- Code $C_5 = \{1, 110, 101\}$
  - Suffix $s_1 = 10$, denn $c_2 = c_1 10$.
  - Suffix $s_2 = 01$, denn $c_3 = c_1 01$.
  - Suffix $s_3 = 0$, denn $s_3 = c_1 0$.
  - Suffix $s_4 = 1$, denn $c_3 = s_1 1$.

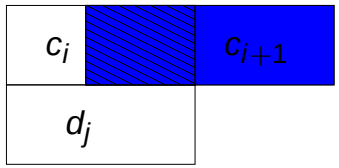
Kriterium für eindeutig entschlüsselbar

Eindeutig entschlüsselbar

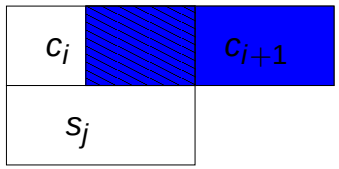
C ist ein eindeutig entschlüsselbarer Code $\Leftrightarrow$ Kein Suffix ist Codewort in C.

z.z.: C nicht eindeutig entschlüsselbar $\Rightarrow$ Suffix ist Codewort

- Zwei gleiche Folgen $c_1 \ldots c_n$ und $d_1 \ldots d_m$ von verschiedenen Codeworten
- Fall 1: Codewort c_i lässt sich zu d_j erweitern

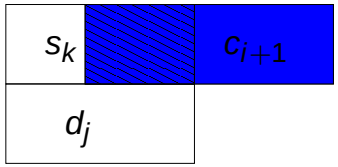


- Fall 2: Codewort c_i lässt sich zu Suffix s_j erweitern



Suffix ist Codewort

- Fall 3: Suffix s_k lässt sich zu Codewort d_j erweitern



- Nach jedem Schritt beginnt der konstruierte Suffix mit einem Codewortpräfix.
- Der zuletzt konstruierte Suffix ist identisch mit dem letzten Codewort von beiden Sequenzen.

Rückrichtung

z.z.: Suffix s ist ein Codewort $\Rightarrow$ C ist nicht eindeutig entschlüsselbar

- Suffix s ist aus Anwendungen der drei Regeln entstanden.
- Berechne die Kette zurück, aus der s entstanden ist.
  - Setze String $c^* \leftarrow s$. Iteriere:
  - 1. Fall $c_i = c_j s$: $c^* \leftarrow c_j c^*$, terminiere.
  - 2. Fall $s' = cs$: $c^* \leftarrow cc^*$, $s \leftarrow s'$.
  - 3. Fall $c = s's$: $c^* \leftarrow s'c^*$, $s \leftarrow s'$.
- Kette muss mit 1. Fall $c_i = c_j s'$ terminieren.
- Zwei verschiedene Entschlüsselungen:
  Eine beginnt mit c_i, die andere mit c_j.
- Beide sind gültig, da der letzte Suffix ein Codewort ist.

Beispiel: Für $C = 1, 110, 101$ erhalten wir für den Suffix 1 den String $c^* = 1101$ mit gültigen Dekodierungen $1|101$ und $110|1$.

Sätze von Kraft und McMillan

Satz von Kraft

Ein Präfixcode C für das Alphabet $A = \{a_1, \ldots, a_n\}$ mit Kodierungslängen $|C(a_j)| = \ell_j$ existiert gdw

$$\sum_{j=1}^{n} 2^{-\ell_j} \leq 1.$$

Satz von McMillan

Ein eindeutig entschlüsselbarer Code C für das Alphabet $A = \{a_1, \ldots, a_n\}$ mit Kodierungslängen $|C(a_j)| = \ell_j$ existiert gdw

$$\sum_{j=1}^{n} 2^{-\ell_j} \leq 1.$$

Präfixcodes genügen

Korollar

Ein Präfixcode C existiert gdw es einen eindeutig entschlüsselbaren Code C mit denselben Kodierungslängen gibt.

- Wir zeigen den Ringschluss für:
  $\sum_{j=1}^{n} 2^{-\ell_j} \leq 1 \Rightarrow$ Präfix $\Rightarrow$ Eindeutig entschlüsselbar

  (Präfix $\Rightarrow$ Eindeutig entschlüsselbar: letzte Vorlesung)
- Gegeben sind Kodierungslängen ℓ_j.
- Gesucht ist ein Präfixcode mit $\ell_j = |C(a_j)|$.
- Definiere $\ell := \max\{\ell_1, \ldots, \ell_n\}$, $n_i :=$ Anzahl ℓ_j mit $\ell_j = i$.

$$\sum_{j=1}^{n} 2^{-\ell_j} = \sum_{j=1}^{\ell} n_j 2^{-j} \leq 1.$$

Beweis: $\sum_{j=1}^{n} 2^{-\ell_j} \leq 1 \Rightarrow$ Präfix

Induktion über ℓ:

- **IA $\ell = 1$:** $n_1 \leq 2$
- Können Präfixcode $C \subseteq \{0,1\}$ für max. 2 Codeworte konstruieren.
- **IS $\ell - 1 \to \ell$:** $n_\ell \leq 2^\ell - n_1 2^{\ell-1} - n_2 2^{\ell-2} - \ldots n_{\ell-1} 2$.
- **IV:** Präfixcode C' mit n_i Worten der Länge i, $i = 1, \ldots, \ell - 1$.
- Anzahl der Worte der Länge ℓ: 2^ℓ
- Wir zählen die durch C' ausgeschlossenen Worte der Länge ℓ.
- Sei $c_i \in C'$ mit Länge ℓ_i. Dann enthalten alle $c_i s \in \{0,1\}^\ell$ mit beliebigem $s \in \{0,1\}^{\ell-\ell_i}$ den Präfix c_i.
- Durch Präfixe der Länge 1 ausgeschlossene Worte: $n_1 \cdot 2^{\ell-1}$.
- Durch Präfixe der Länge 2 ausgeschlossene Worte: $n_2 \cdot 2^{\ell-2}$.

$\vdots$

- Durch Präfixe der Länge $\ell - 1$ ausgeschlossene Worte: $n_{\ell-1} \cdot 2$.
- D.h. wir kodieren die n_ℓ Worte mit den verbleibenden $2^\ell - (n_1 2^{\ell-1} + \cdots + n_{\ell-1} 2)$ Worten der Länge ℓ als Präfixcode.