

Algorithmen und Datenstrukturen in C/C++

Prof. Dr. rer. nat. Peter Gerwinski

24. Mai 2018

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 Einführung in C++

3 Datenorganisation

4 Datenkodierung

4.0 Parität

4.1 $(x^2 - 1)$ Der Herr der Ringe: Manchmal ist $1 + 1 = 0$.

4.1 Fehlererkennung durch CRC

4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

4.3 Verschlüsselung

5 Hardwarenahe Algorithmen

6 Optimierung

7 Numerik



Änderungen
vorbehalten

4 Datenkodierung

4.0 Parität

4.0.1 Fehlererkennung durch Parität

Zusätzlich zu den Daten: Paritäts-Bit

Quersumme einschließlich Paritäts-Bit muß gerade/ungerade sein

—→ *gerade/ungerade Parität*

Wenn nicht: Fehler erkannt

Beispiel:

Nachricht: 11010110

gerade Parität: Paritäts-Bit: 1

kodierte Nachricht: 110101101

ungerade Parität: Paritäts-Bit: 0

kodierte Nachricht: 110101100

1 falsches Bit wird erkannt.

2 falsche Bits werden nicht erkannt.

3 falsche Bits werden erkannt.

...

Eine ungerade Anzahl falscher Bits wird erkannt.

4 Datenkodierung

4.0 Parität

4.0.2 Ausfallsicherheit durch Parität: RAID

Daten mit Parität

1 Bit fehlt, ansonsten korrekt

→ fehlendes Bit rekonstruierbar

Anwendung: RAID

zusätzliche Platte speichert (gerade) Parität

Platte 1	Platte 2	Platte 3
A1	A2	$A3 = A1 \text{ xor } A2$

Parität berechnen:

$$\begin{array}{rcl} & 0101 & A1 \\ \text{xor} & 0011 & A2 \\ \hline & 0110 & A3 \end{array}$$

Verlorene Bits rekonstruieren:

$$\begin{array}{rcl} & 0101 & A1 \\ \text{xor} & 0110 & A3 \\ \hline & 0011 & A2 \end{array}$$

4 Datenkodierung

4.0 Parität

4.0.2 Ausfallsicherheit durch Parität: RAID

- RAID 3* Speicherung bitweise
RAID 4 Speicherung blockweise
RAID 5 Speicherung blockweise, Parität wandert
- } Paritätsplatte ist „Flaschenhals“

Platte 1	Platte 2	Platte 3
A1	A2	$A3 = A1 \text{ xor } A2$
$B1 = B2 \text{ xor } B3$	B2	B3
C1	$C2 = C1 \text{ xor } C3$	C3
D1	D2	$D3 = D1 \text{ xor } D2$
...

4 Datenkodierung

4.0 Parität

4.0.3 Fehlerkorrektur durch Hamming-Code

Nachricht: 11010110

defekte Nachricht: 11000110 *Hamming-Abstand 1*

- „zu viele“ Symbole senden

Beispiel: 7 Bit senden \rightarrow 128 Symbole,
davon nur 16 Symbole gültig \rightarrow 4 Bit

- gültige Symbole so anordnen,
daß Hamming-1-Nachbarn zugerechnet werden können

\rightarrow 1 fehlerhaft übertragenes Bit kann korrigiert werden.

4 Datenkodierung

4.0 Parität

4.0.3 Fehlerkorrektur durch Hamming-Code

(3, 1)-Hamming-Code

Nachricht:	1	1	0	1	0	1	1	0
kodierte Nachricht:	1	1	1	1	1	0	0	0
defekte Nachricht:	1	1	1	1	1	0	1	0
Rekonstruktion:	1	1	0	1	0	1	1	0

Analog: (5, 1)-Hamming-Code

4 Datenkodierung

4.0 Parität

4.0.3 Fehlerkorrektur durch Hamming-Code

(7, 4)-Hamming-Code

p p d p d d d

- 1. Paritäts-Bit: Parität über jedes zweite Bit
- 2. Paritäts-Bit: Parität über jedes dritte und vierte von 4 Bit
- 3. Paritäts-Bit: Parität über jedes fünfte bis achte von 8 Bit

4 Datenkodierung

4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

- Polynome über endlichen Körpern
- n Zahlen \longleftrightarrow Polynom vom Grad $n - 1$
Beispiel: 3 Stützstellen \longrightarrow Parabel
- mehr als n Zahlen \longrightarrow Polynom überbestimmt

\longrightarrow Ausfall- und Fehlerkorrektur möglich \longrightarrow *Reed-Solomon-Code*

4 Datenkodierung

4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

- Polynome über endlichen Körpern
- n Zahlen \longleftrightarrow Polynom vom Grad $n - 1$
Beispiel: 3 Stützstellen \longrightarrow Parabel
- mehr als n Zahlen \longrightarrow Polynom überbestimmt
- Variante 1 (BCH):
Nachricht = Koeffizienten des Polynoms
Übertragung = Funktionswerte an Stützstellen
- Variante 2 (systematische Kodierung):
Übertragung enthält Nachricht = Funktionswerte an Stützstellen
Koeffizienten berechnen \longrightarrow Funktionswerte an zusätzlichen Stützstellen

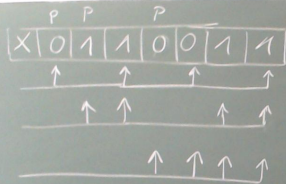
4 Datenkodierung

4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

- Polynome über endlichen Körpern
- n Zahlen \longleftrightarrow Polynom vom Grad $n - 1$
Beispiel: 3 Stützstellen \longrightarrow Parabel
- mehr als n Zahlen \longrightarrow Polynom überbestimmt
- Ausfallkorrektur: trivial. ; -)
- Fehlererkennung: Polynom hat zu hohen Grad
Beispiel: 4 Punkte liegen nicht auf Parabel
- Fehlerkorrektur (theoretisch):
alle Kombinationen durchprobieren, Mehrheitsentscheid
- Fehlerkorrektur (praktisch): spezielle Algorithmen

4 Datenkodierung

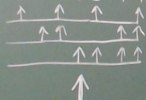
4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code



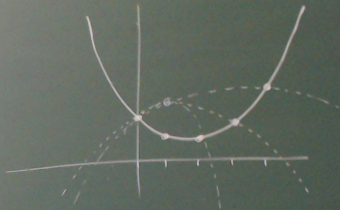
Nachricht: 1011

kodiert: 0110011

defekt: x0111011



✓ alle richtig
✓ alle richtig
✗ eins falsch



Statt über \mathbb{R} :

Polynome über endlichem Körper,
z.B. über $\{0, 1, \dots, p-1\}$ (p prim)
oder mit 2^m Elementen,
z.B. $\{0, 1, i, 1+i\}$ oder so ähnlich
($i = \sqrt{-1}$)

4 Datenkodierung

4.3 Verschlüsselung

4.3.0 OpenPGP in der Praxis

E-Mail

Malware, geschrieben in HTML

- abgefangene E-Mail, verschlüsselt
→ entschlüsseln
- an berechtigten Empfänger senden,
damit er sie unbemerkt entschlüsselt

sende:

Hausaufgabe

Verschlüsselte E-Mail an PG,
die auch verschlüsselt beantwortet wird

- Software installieren:
GnuPG, Kleopatra, GPG4Win,
Enigmail, GPA, mutt
- Schlüsselpaar erzeugen
- Schlüsselaustausch
- benutzen

Schlüssel von PG: Fingerprint

2CFD EA6C 31E6 12A0 BFFF

2457 D955 0227 6224 E7F5

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 Einführung in C++

3 Datenorganisation

4 Datenkodierung

4.0 Parität

4. $(x^2 - 1)$ Der Herr der Ringe: Manchmal ist $1 + 1 = 0$.

4.1 Fehlererkennung durch CRC

4.2 Ausfall- und Fehlerkorrektur durch Reed-Solomon-Code

4.3 Verschlüsselung

5 Hardwarenahe Algorithmen

6 Optimierung

7 Numerik



Änderungen
vorbehalten