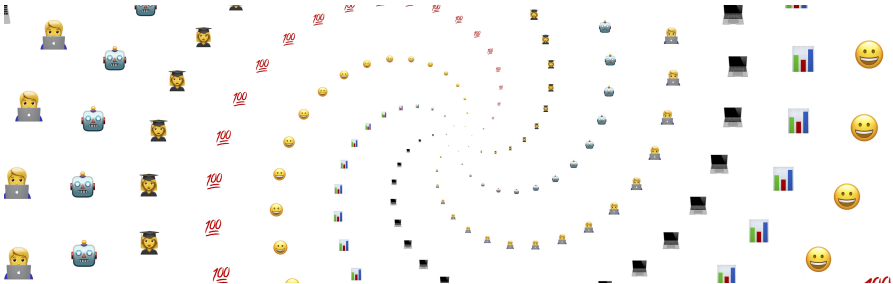
Datenintegrität

Optimale Codes und Hamming-Codes

Cyril Blum

Fachschaft Informatik
Kantonsschule im Lee



Rückblick: Kartentrick-Methode

Nachricht: 0101 1111 0010 1011

c_1	c_2	c_3	c_4		
0	0	1	1	0	c_9
0	1	0	1	0	c_5
1	1	1	1	0	c_6
0	0	1	0	1	c_7
1	0	1	1	1	c_8

- 16 Nachrichtenbits $a_1, \ldots, a_{16}$ (zeilenweise), 9 Kontrollbits $c_1, \ldots, c_9$



Rückblick: Kartentrick-Methode

Nachricht: 0101 1111 0010 1011

c_1	c_2	c_3	c_4		
0	0	1	1	0	c_9
0	1	0	1	0	c_5
1	1	1	1	0	c_6
0	0	1	0	1	c_7
1	0	1	1	1	c_8

- 16 Nachrichtenbits $a_1, \ldots, a_{16}$ (zeilenweise), 9 Kontrollbits $c_1, \ldots, c_9$
- c_1 überwacht die erste Spalte: a_1, a_5, a_9, a_{13} und sich selbst
- $0 + 1 + 0 + 1 + \underbrace{0}_{c_1}$: 2 Einsen, gerade



Rückblick: Kartentrick-Methode

Nachricht: 0101 1111 0010 1011

c_1	c_2	c_3	c_4		
0	0	1	1	0	c_9
0	1	0	1	0	c_5
1	1	1	1	0	c_6
0	0	1	0	1	c_7
1	0	1	1	1	c_8

- 16 Nachrichtenbits $a_1, \ldots, a_{16}$ (zeilenweise), 9 Kontrollbits $c_1, \ldots, c_9$
- c_1 überwacht die erste Spalte: a_1, a_5, a_9, a_{13} und sich selbst
- $0 + 1 + 0 + 1 + \underbrace{0}_{c_1}$: 2 Einsen, gerade
- Codewort: $16 + 9 = 25$ Bit. Geht es mit weniger Kontrollbits?



Was heisst „überwachen“?

Definition

Ein Kontrollbit **überwacht** eine Gruppe von Bits (seinen Überwachungsbereich), wenn es so gewählt wird, dass die Anzahl Einsen im Bereich inklusive des Kontrollbits selbst gerade ist.



FreeFlower

Was heisst „überwachen“?

Definition
Ein Kontrollbit **überwacht** eine Gruppe von Bits (seinen Überwachungsbereich), wenn es so gewählt wird, dass die Anzahl Einsen im Bereich inklusive des Kontrollbits selbst gerade ist.

- Bereich von c_1: a_1, a_5, a_9, a_{13} und c_1



Was heisst „überwachen“?

Definition
Ein Kontrollbit **überwacht** eine Gruppe von Bits (seinen Überwachungsbereich), wenn es so gewählt wird, dass die Anzahl Einsen im Bereich inklusive des Kontrollbits selbst gerade ist.

- Bereich von c_1: a_1, a_5, a_9, a_{13} und c_1
- Kippt ein Bit im Bereich, ist die Anzahl Einsen ungerade $\rightarrow$ c_1 *meldet* einen Fehler



Überwacht ein Kontrollbit sich selbst?

0	0	1	1	0
0	1	0	1	0
1	1	1	1	0
0	0	1	0	1
1	0	1	1	1

- Ja: c_1 prüft a_1, a_5, a_9, a_{13} und sich selbst



Überwacht ein Kontrollbit sich selbst?

1	0	1	1	0
0	1	0	1	0
1	1	1	1	0
0	0	1	0	1
1	0	1	1	1

- Ja: c_1 prüft a_1, a_5, a_9, a_{13} und sich selbst
- Beispiel: c_1 kippt von 0 auf 1

Überwacht ein Kontrollbit sich selbst?

1	0	1	1	0
0	1	0	1	0
1	1	1	1	0
0	0	1	0	1
1	0	1	1	1

- Ja: c_1 prüft a_1, a_5, a_9, a_{13} und sich selbst
- Beispiel: c_1 kippt von 0 auf 1
- Erste Spalte: $0 + 1 + 0 + 1 + 1$, 3 Einsen $\rightarrow$ c_1 meldet
- Sonst meldet nur c_9 (zählt alle Bits)



Überwacht ein Kontrollbit sich selbst?

1	0	1	1	0
0	1	0	1	0
1	1	1	1	0
0	0	1	0	1
1	0	1	1	1

- Ja: c_1 prüft a_1, a_5, a_9, a_{13} und sich selbst
- Beispiel: c_1 kippt von 0 auf 1
- Erste Spalte: $0 + 1 + 0 + 1 + 1$, 3 Einsen $\rightarrow$ c_1 meldet
- Sonst meldet nur c_9 (zählt alle Bits)
- Kein a_i hat dieses Muster $\rightarrow$ c_1 selbst ist gekippt
- Tabelle: Kreuz in Zeile c_1, Spalte c_1

Von der Tabelle zur Überwachungstabelle

c_1	c_2	c_3	c_4	c_9
a_1	a_2	a_3	a_4	c_5
a_5	a_6	a_7	a_8	c_6
a_9	a_{10}	a_{11}	a_{12}	c_7
a_{13}	a_{14}	a_{15}	a_{16}	c_8



Von der Tabelle zur Überwachungstabelle

c_1	c_2	c_3	c_4	c_9
a_1	a_2	a_3	a_4	c_5
a_5	a_6	a_7	a_8	c_6
a_9	a_{10}	a_{11}	a_{12}	c_7
a_{13}	a_{14}	a_{15}	a_{16}	c_8

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	×				×				×				×				×								
c_2		×				×				×				×				×							
c_3			×				×				×				×				×						
c_4				×				×				×				×				×					
c_5	×	×	×	×																	×				
c_6					×	×	×	×														×			
c_7									×	×	×	×											×		
c_8													×	×	×	×								×	
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- a_1 liegt in Spalte c_1 und Zeile c_5, c_9 zählt alles $\rightarrow$ Spalte a_1: c_1, c_5, c_9



Von der Tabelle zur Überwachungstabelle

c_1	c_2	c_3	c_4	c_9
a_1	a_2	a_3	a_4	c_5
a_5	a_6	a_7	a_8	c_6
a_9	a_{10}	a_{11}	a_{12}	c_7
a_{13}	a_{14}	a_{15}	a_{16}	c_8

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	×				×				×				×				×								
c_2		×				×				×				×				×							
c_3			×				×				×				×				×						
c_4				×				×				×				×				×					
c_5	×	×	×	×																	×				
c_6					×	×	×	×														×			
c_7									×	×	×	×											×		
c_8													×	×	×	×								×	
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- a_1 liegt in Spalte c_1 und Zeile c_5, c_9 zählt alles $\rightarrow$ Spalte a_1: c_1, c_5, c_9
- c_1 prüft die erste Spalte $\rightarrow$ Zeile c_1: $a_1, a_5, a_9, a_{13}, c_1$



Überwachungstabelle

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	×				×				×				×				×								
c_2		×				×				×				×				×							
c_3			×				×				×				×				×						
c_4				×				×				×				×				×					
c_5	×	×	×	×																	×				
c_6					×	×	×	×														×			
c_7									×	×	×	×											×		
c_8													×	×	×	×								×	
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- Zeile: Kontrollbit, Spalte: Bit des Codeworts, Kreuz: wird überwacht



Überwachungstabelle

Wert	0	1	0	1	1	1	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
c_1	×				×				×				×				×								
c_2		×				×				×				×				×							
c_3			×				×				×				×				×						
c_4				×				×				×				×				×					
c_5	×	×	×	×																	×				
c_6					×	×	×	×														×			
c_7									×	×	×	×											×		
c_8													×	×	×	×								×	
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- Zeile: Kontrollbit, Spalte: Bit des Codeworts, Kreuz: wird überwacht



Überwachungstabelle

Wert	0	1	0	1	1	1	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								2 ✓
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				4 ✓
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	14 ✓

- Zeile: Kontrollbit, Spalte: Bit des Codeworts, Kreuz: wird überwacht
- Zeile c_1: unter den Kreuzen steht `0,1,0,1,0` $\rightarrow$ 2 Einsen



Überwachungstabelle

Wert	0	1	0	1	1	1	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								2 ✓
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				4 ✓
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	14 ✓

- Zeile: Kontrollbit, Spalte: Bit des Codeworts, Kreuz: wird überwacht
- Zeile c_1: unter den Kreuzen steht `0,1,0,1,0` $\rightarrow$ 2 Einsen
- In jeder Zeile gerade: die ***Anzahl Einsen unter den Kreuzen*** (die Anzahl Kreuze ist egal)



Signatur eines Bits

Wert	0	1	0	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	

0	0	1	1	0
0	1	0	1	0
1	0	1	1	0
0	0	1	0	1
1	0	1	1	1

- Empfangen: a_6 ist von 1 auf 0 gekippt



Signatur eines Bits

Wert	0	1	0	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								**1** ×
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				**3** ×
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	**13** ×

0	0	1	1	0
0	1	0	1	0
1	0	1	1	0
0	0	1	0	1
1	0	1	1	1

- Empfangen: a_6 ist von 1 auf 0 gekippt
- Ungerade, melden also: c_2, c_6, c_9



Signatur eines Bits

Wert	0	1	0	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								**1** ×
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				**3** ×
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	**13** ×

0	0	1	1	0
0	1	0	1	0
1	0	1	1	0
0	0	1	0	1
1	0	1	1	1

- Empfangen: a_6 ist von 1 auf 0 gekippt
- Ungerade, melden also: c_2, c_6, c_9
- Spalte a_6: Kreuze genau bei c_2, c_6, c_9



Signatur eines Bits

Wert	0	1	0	1	1	0	1	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	1	1	0	
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	*Einsen*
c_1	×				×				×				×				×									2 ✓
c_2		×				×				×				×				×								**1** ×
c_3			×				×				×				×				×							4 ✓
c_4				×				×				×				×				×						4 ✓
c_5	×	×	×	×																	×					2 ✓
c_6					×	×	×	×														×				**3** ×
c_7									×	×	×	×											×			2 ✓
c_8													×	×	×	×								×		4 ✓
c_9	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	**13** ×

0	0	1	1	0
0	1	0	1	0
1	0	1	1	0
0	0	1	0	1
1	0	1	1	1

- Empfangen: a_6 ist von 1 auf 0 gekippt
- Ungerade, melden also: c_2, c_6, c_9
- Spalte a_6: Kreuze genau bei c_2, c_6, c_9
- ***Signatur*** eines Bits: seine Spalte, also wer bei seinem Fehler meldet



Wann ist eine Codierung 1-fehlerkorrigierend?

Bemerkung (Merksatz)

Eine Codierung ist genau dann 1-fehlerkorrigierend, wenn die Spalten ihrer Überwachungstabelle

- paarweise verschieden und
- nicht leer sind.



Wann ist eine Codierung 1-fehlerkorrigierend?

Bemerkung (Merksatz)

Eine Codierung ist genau dann 1-fehlerkorrigierend, wenn die Spalten ihrer Überwachungstabelle

- paarweise verschieden und
- nicht leer sind.

- Gleiche Spalten: Zwei Bits haben dieselbe Signatur $\rightarrow$ Fehler nicht lokalisierbar. Beispiel: Hätte a_2 wie a_6 die Signatur c_2, c_6, c_9, wüsste man beim obigen Fehler nicht, ob a_2 oder a_6 gekippt ist



Wann ist eine Codierung 1-fehlerkorrigierend?

Bemerkung (Merksatz)

Eine Codierung ist genau dann 1-fehlerkorrigierend, wenn die Spalten ihrer Überwachungstabelle

- paarweise verschieden und
- nicht leer sind.

- Gleiche Spalten: Zwei Bits haben dieselbe Signatur $\rightarrow$ Fehler nicht lokalisierbar. Beispiel: Hätte a_2 wie a_6 die Signatur c_2, c_6, c_9, wüsste man beim obigen Fehler nicht, ob a_2 oder a_6 gekippt ist
- Leere Spalte: Ein Fehler in diesem Bit wird nie bemerkt



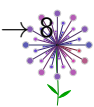
Wann ist eine Codierung 1-fehlerkorrigierend?

Bemerkung (Merksatz)

Eine Codierung ist genau dann 1-fehlerkorrigierend, wenn die Spalten ihrer Überwachungstabelle

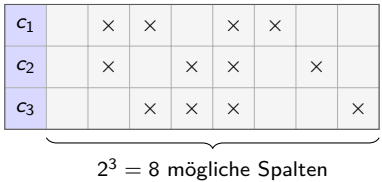
- paarweise verschieden und
- nicht leer sind.

- Gleiche Spalten: Zwei Bits haben dieselbe Signatur $\rightarrow$ Fehler nicht lokalisierbar. Beispiel: Hätte a_2 wie a_6 die Signatur c_2, c_6, c_9, wüsste man beim obigen Fehler nicht, ob a_2 oder a_6 gekippt ist
- Leere Spalte: Ein Fehler in diesem Bit wird nie bemerkt
- Kartentrick: Auch ohne c_9 bleiben alle Spalten verschieden und nicht leer $\rightarrow$ 8 Kontrollbits genügen.
  Beispiel: a_6 hätte dann die Signatur c_2, c_6



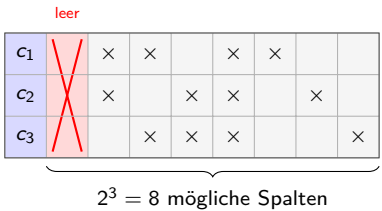
FreeFlower

Wie viele Kontrollbits sind mindestens nötig?



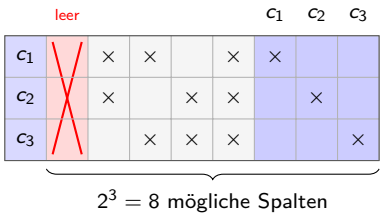
- $K = 3$ Kontrollbits $\rightarrow$ 3 Zeilen, jede Zelle mit oder ohne Kreuz $\rightarrow 2^3 = 8$ mögliche Spalten

Wie viele Kontrollbits sind mindestens nötig?



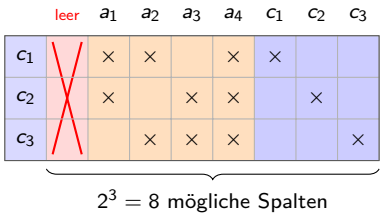
- $K = 3$ Kontrollbits $\rightarrow$ 3 Zeilen, jede Zelle mit oder ohne Kreuz $\rightarrow 2^3 = 8$ mögliche Spalten
- Die leere Spalte ist verboten $\rightarrow 2^3 - 1 = 7$ nutzbare Spalten

Wie viele Kontrollbits sind mindestens nötig?



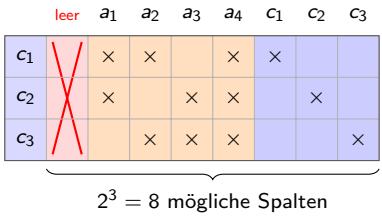
- $K = 3$ Kontrollbits $\rightarrow$ 3 Zeilen, jede Zelle mit oder ohne Kreuz $\rightarrow 2^3 = 8$ mögliche Spalten
- Die leere Spalte ist verboten $\rightarrow 2^3 - 1 = 7$ nutzbare Spalten
- Die Kontrollbits brauchen 3 davon

Wie viele Kontrollbits sind mindestens nötig?



- $K = 3$ Kontrollbits $\rightarrow$ 3 Zeilen, jede Zelle mit oder ohne Kreuz $\rightarrow 2^3 = 8$ mögliche Spalten
- Die leere Spalte ist verboten $\rightarrow 2^3 - 1 = 7$ nutzbare Spalten
- Die Kontrollbits brauchen 3 davon
- Bleiben 4 Spalten für Nachrichtenbits: $N + K = 4 + 3 \leq 7$

Wie viele Kontrollbits sind mindestens nötig?



- $K = 3$ Kontrollbits $\rightarrow$ 3 Zeilen, jede Zelle mit oder ohne Kreuz $\rightarrow 2^3 = 8$ mögliche Spalten
- Die leere Spalte ist verboten $\rightarrow 2^3 - 1 = 7$ nutzbare Spalten
- Die Kontrollbits brauchen 3 davon
- Bleiben 4 Spalten für Nachrichtenbits: $N + K = 4 + 3 \leq 7$
- Allgemein: K Kontrollbits, N Nachrichtenbits

$$2^K - 1 \geq N + K$$

Wie viele Kontrollbits braucht eine 16-Bit-Nachricht?

$K = 3$ 

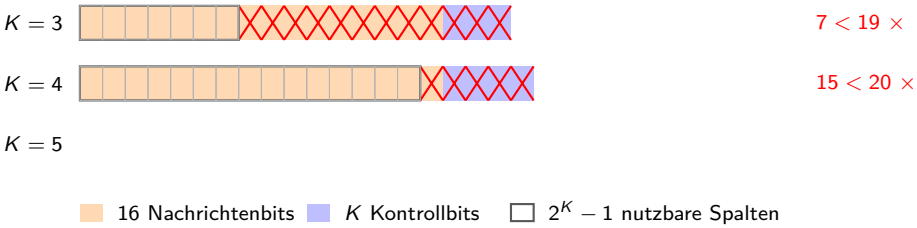
$7 < 19$ ×

$K = 4$

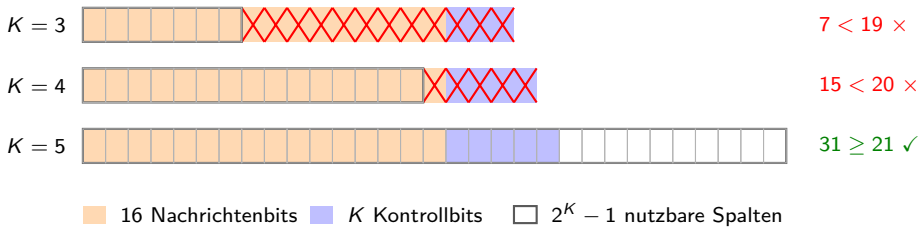
$K = 5$

16 Nachrichtenbits K Kontrollbits $2^K - 1$ nutzbare Spalten

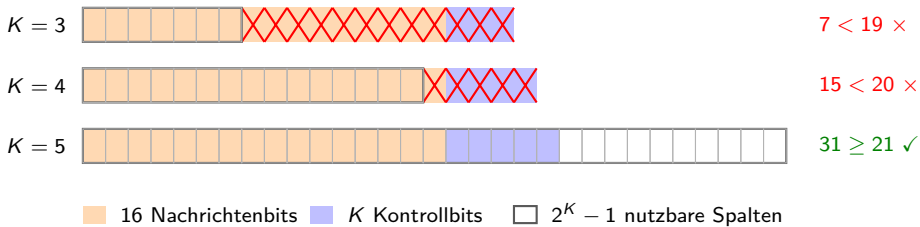
Wie viele Kontrollbits braucht eine 16-Bit-Nachricht?



Wie viele Kontrollbits braucht eine 16-Bit-Nachricht?



Wie viele Kontrollbits braucht eine 16-Bit-Nachricht?



- Mindestens 5 Kontrollbits (statt 9 beim Kartentrick)

Optimale Überwachungstabelle für 16 Bit

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5
c_1	×	×	×	×							×	×	×				×				
c_2	×				×	×	×				×	×	×	×	×			×			
c_3		×			×			×	×		×			×		×			×		
c_4			×			×		×		×		×		×	×	×				×	
c_5				×			×		×	×			×		×	×					×

- Kontrollbits: je ein Kreuz in der eigenen Zeile

Optimale Überwachungstabelle für 16 Bit

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5
c_1	×	×	×	×							×	×	×				×				
c_2	×				×	×	×				×	×	×	×	×			×			
c_3		×			×			×	×		×			×		×			×		
c_4			×			×		×		×		×		×	×	×				×	
c_5				×			×		×	×			×		×	×					×

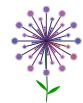
- Kontrollbits: je ein Kreuz in der eigenen Zeile
- Nachrichtenbits: Kreuze frei wählbar, solange alle Spalten verschieden und nicht leer sind



Optimale Überwachungstabelle für 16 Bit

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5
c_1	×	×	×	×							×	×	×				×				
c_2	×				×	×	×				×	×	×	×	×			×			
c_3		×			×			×	×		×			×		×			×		
c_4			×			×		×		×		×		×	×	×				×	
c_5				×			×		×	×			×		×	×					×

- Kontrollbits: je ein Kreuz in der eigenen Zeile
- Nachrichtenbits: Kreuze frei wählbar, solange alle Spalten verschieden und nicht leer sind
- Codewort: $16 + 5 = 21$ Bit



Optimale Überwachungstabelle für 16 Bit

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	c_1	c_2	c_3	c_4	c_5
c_1	×	×	×	×							×	×	×				×				
c_2	×				×	×	×				×	×	×	×	×			×			
c_3		×			×			×	×		×			×		×			×		
c_4			×			×		×		×		×		×	×	×				×	
c_5				×			×		×	×			×		×	×					×

- Kontrollbits: je ein Kreuz in der eigenen Zeile
- Nachrichtenbits: Kreuze frei wählbar, solange alle Spalten verschieden und nicht leer sind
- Codewort: $16 + 5 = 21$ Bit
- Beispiel: Die Spalte von a_6 hat Kreuze bei c_2 und c_4. Diese Signatur kommt beim Decodieren wieder vor



Codieren

Nachricht: 0101 1111 0010 1011

- c_1 überwacht $a_1, a_2, a_3, a_4, a_{11}, a_{12}, a_{13}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_1 = 0$



Codieren

Nachricht: 0101 1111 0010 1011

- c_1 überwacht $a_1, a_2, a_3, a_4, a_{11}, a_{12}, a_{13}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_1 = 0$
- c_2 überwacht $a_1, a_5, a_6, a_7, a_{11}, \ldots, a_{15}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_2 = 0$



Codieren

Nachricht: 0101 1111 0010 1011

- c_1 überwacht $a_1, a_2, a_3, a_4, a_{11}, a_{12}, a_{13}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_1 = 0$
- c_2 überwacht $a_1, a_5, a_6, a_7, a_{11}, \ldots, a_{15}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_2 = 0$
- Ebenso: $c_3 = 1$, $c_4 = 0$, $c_5 = 1$



Codieren

Nachricht: 0101 1111 0010 1011

- c_1 überwacht $a_1, a_2, a_3, a_4, a_{11}, a_{12}, a_{13}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_1 = 0$
- c_2 überwacht $a_1, a_5, a_6, a_7, a_{11}, \ldots, a_{15}$: 0101 1111 0010 1011
  Anzahl Einsen gerade $\rightarrow c_2 = 0$
- Ebenso: $c_3 = 1$, $c_4 = 0$, $c_5 = 1$
- Codewort:

$$\underbrace{0101\ 1111\ 0010\ 1011}_{\text{Nachricht}}\ \underbrace{00101}_{\text{Kontrollbits}}$$



Decodieren: Fehler finden

Empfangen: `0101 1011 0010 1011 00101`

- Für jedes Kontrollbit prüfen: Ist die Anzahl Einsen im Überwachungsbereich (samt Kontrollbit) gerade?



FreeFlower

Decodieren: Fehler finden

Empfangen: `0101 1011 0010 1011 00101`

- Für jedes Kontrollbit prüfen: Ist die Anzahl Einsen im Überwachungsbereich (samt Kontrollbit) gerade?
- Verletzt: c_2 und c_4. In Ordnung: c_1, c_3, c_5



Decodieren: Fehler finden

Empfangen: `0101 1011 0010 1011 00101`

- Für jedes Kontrollbit prüfen: Ist die Anzahl Einsen im Überwachungsbereich (samt Kontrollbit) gerade?
- Verletzt: c_2 und c_4. In Ordnung: c_1, c_3, c_5
- Signatur $(c_1, \ldots, c_5) = (\cdot, \times, \cdot, \times, \cdot)$



Decodieren: Fehler finden

Empfangen: `0101 1011 0010 1011 00101`

- Für jedes Kontrollbit prüfen: Ist die Anzahl Einsen im Überwachungsbereich (samt Kontrollbit) gerade?
- Verletzt: c_2 und c_4. In Ordnung: c_1, c_3, c_5
- Signatur $(c_1, \ldots, c_5) = (\cdot, \times, \cdot, \times, \cdot)$
- Die einzige Spalte mit dieser Signatur ist a_6



Decodieren: Fehler finden

Empfangen: `0101 1011 0010 1011 00101`

- Für jedes Kontrollbit prüfen: Ist die Anzahl Einsen im Überwachungsbereich (samt Kontrollbit) gerade?
- Verletzt: c_2 und c_4. In Ordnung: c_1, c_3, c_5
- Signatur $(c_1, \ldots, c_5) = (\cdot, \times, \cdot, \times, \cdot)$
- Die einzige Spalte mit dieser Signatur ist a_6
- a_6 umdrehen: `0101 1111 0010 1011 00101`



Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	1	1	?	?	?

Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	1	1	?	?	?

Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	1	1	0	?	?

c_1: 1+0+1 $=$ 2 Einsen $\rightarrow$ $c_1 = 0$



Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	1	1	0	?	?

c_1: 1+0+1 = 2 Einsen $\rightarrow$ $c_1 = 0$



Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	1	1	0	1	?

c_1: 1+0+1 $=$ 2 Einsen $\rightarrow$ $c_1 = 0$

c_2: 1+1+1 $=$ 3 Einsen $\rightarrow$ $c_2 = 1$

Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	1	1	0	1	?

c_1: 1+0+1 $=$ 2 Einsen $\rightarrow$ $c_1 = 0$

c_2: 1+1+1 $=$ 3 Einsen $\rightarrow$ $c_2 = 1$

Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	1	1	0	1	0

c_1: 1+0+1 $=$ 2 Einsen $\rightarrow$ $c_1 = 0$

c_2: 1+1+1 $=$ 3 Einsen $\rightarrow$ $c_2 = 1$

c_3: 0+1+1 $=$ 2 Einsen $\rightarrow$ $c_3 = 0$



Beispiel: Kontrollbits für 1011 berechnen

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	1	1	0	1	0

c_1: 1+0+1 = 2 Einsen $\rightarrow$ $c_1 = 0$

c_2: 1+1+1 = 3 Einsen $\rightarrow$ $c_2 = 1$

c_3: 0+1+1 = 2 Einsen $\rightarrow$ $c_3 = 0$

Codewort: 1011 010



Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	1	1	0	1	0

Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

c_1: 1+0+1+0 = 2, gerade ✓



Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

c_1: 1+0+1+0 = 2, gerade ✓



Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 010

c_1: 1+0+1+0 = 2, gerade ✓

c_2: 1+0+1+1 = 3, ungerade → c_2 meldet

Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 010

c_1: 1+0+1+0 = 2, gerade ✓

c_2: 1+0+1+1 = 3, ungerade → c_2 meldet

Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×

1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

c_1: 1+0+1+0 = 2, gerade ✓

c_2: 1+0+1+1 = 3, ungerade → c_2 meldet

c_3: 0+0+1+0 = 1, ungerade → c_3 meldet



Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	0	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

c_1: 1+0+1+0 = 2, gerade ✓

c_2: 1+0+1+1 = 3, ungerade → c_2 meldet

c_3: 0+0+1+0 = 1, ungerade → c_3 meldet

Signatur c_2, c_3: nur die Spalte von a_3 → a_3 zurückkippen



Beispiel: Ein Bit kippt

	a_1	a_2	a_3	a_4	c_1	c_2	c_3
c_1	×	×		×	×		
c_2	×		×	×		×	
c_3		×	×	×			×
	1	0	1	1	0	1	0

a_3 kippt: 1001 $\underline{010}$

c_1: 1+0+1+0 = 2, gerade ✓

c_2: 1+0+1+1 = 3, ungerade → c_2 meldet

c_3: 0+0+1+0 = 1, ungerade → c_3 meldet

Signatur c_2, c_3: nur die Spalte von a_3 → a_3 zurückkippen



Auftrag

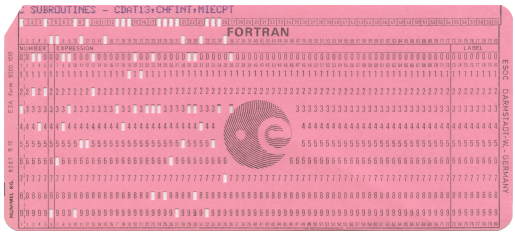
Skript „Datenintegrität“

- Abschnitt 1.6 bis und mit „Konstruktion einer optimalen Überwachungstabelle“ lesen
- 1.19, 1.20 (Anzahl Kontrollbits), 1.22 (Codieren und Decodieren)
- Challenge: 1.21 (Codierung entwerfen)



FreeFlower

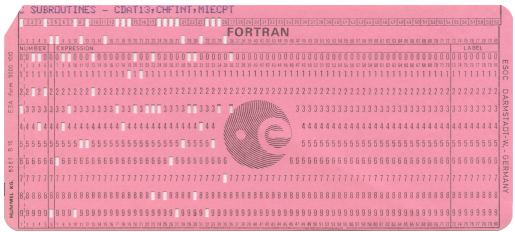
Richard Hamming (1915–1998)



Fortran-Lochkarte der ESA, 70er Jahre. Till Niermann, CC BY-SA 3.0, via Wikimedia Commons

- Mathematiker bei den Bell Laboratories

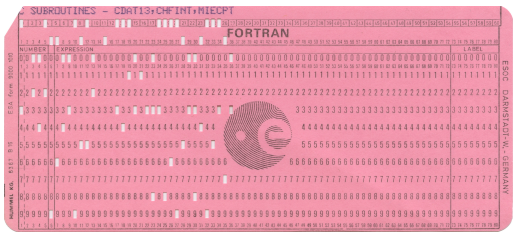
Richard Hamming (1915–1998)



Fortran-Lochkarte der ESA, 70er Jahre. Till Niermann, CC BY-SA 3.0, via Wikimedia Commons

- Mathematiker bei den Bell Laboratories
- Programme und Daten auf Lochkarten: Loch = 1, kein Loch = 0

Richard Hamming (1915–1998)

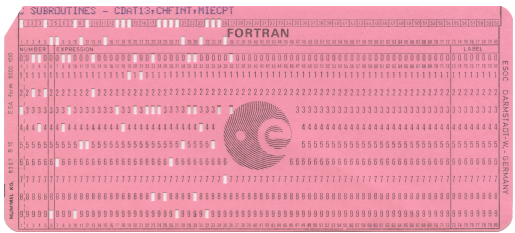


Fortran-Lochkarte der ESA, 70er Jahre. Till Niermann, CC BY-SA 3.0, via Wikimedia Commons

- Mathematiker bei den Bell Laboratories
- Programme und Daten auf Lochkarten: Loch = 1, kein Loch = 0
- Lesefehler wurden oft erst nach langer Rechenzeit bemerkt



Richard Hamming (1915–1998)

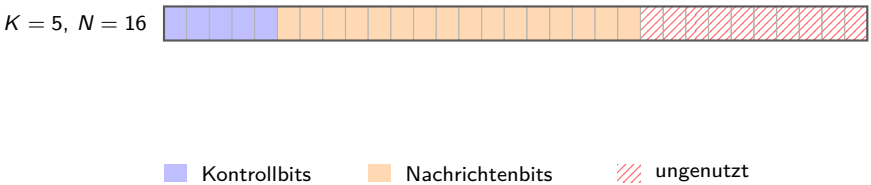


Fortran-Lochkarte der ESA, 70er Jahre. Till Niermann, CC BY-SA 3.0, via Wikimedia Commons

- Mathematiker bei den Bell Laboratories
- Programme und Daten auf Lochkarten: Loch = 1, kein Loch = 0
- Lesefehler wurden oft erst nach langer Rechenzeit bemerkt
- Hamming-Codes (1950): einzelne Bitfehler mit wenigen Kontrollbits korrigieren



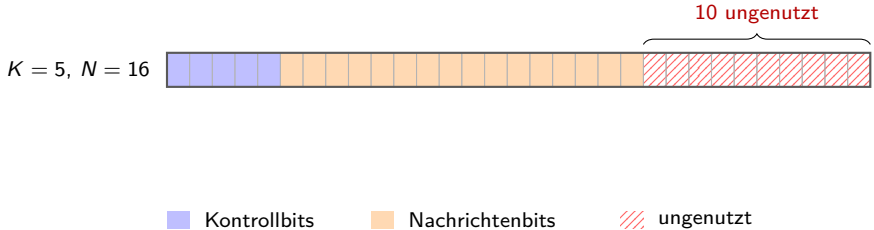
Welche Spalten nehmen wir?



- 16 Nachrichtenbits, 5 Kontrollbits: 16 der 26 übrigen Spalten frei wählbar

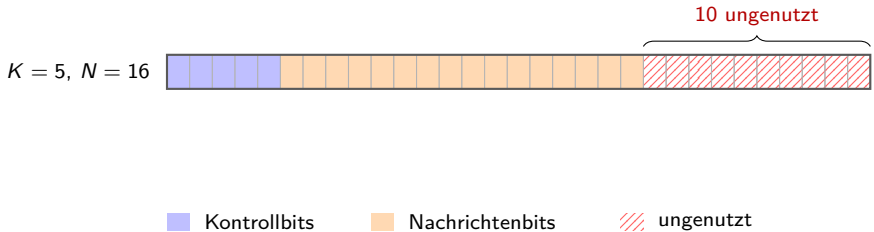


Welche Spalten nehmen wir?



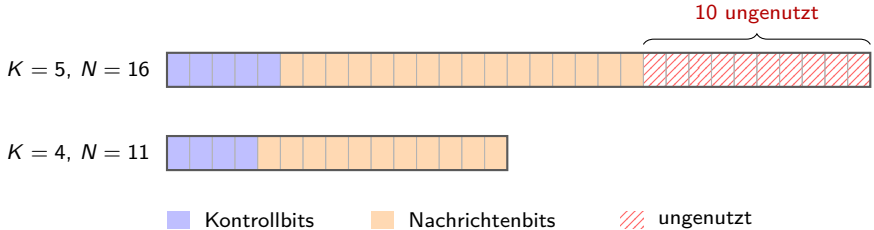
- 16 Nachrichtenbits, 5 Kontrollbits: 16 der 26 übrigen Spalten frei wählbar
- 10 Spalten ungenutzt: 5 Kontrollbits reichen für 26 Nachrichtenbits

Welche Spalten nehmen wir?



- 16 Nachrichtenbits, 5 Kontrollbits: 16 der 26 übrigen Spalten frei wählbar
- 10 Spalten ungenutzt: 5 Kontrollbits reichen für 26 Nachrichtenbits
- Zuordnung beliebig: Beide Seiten brauchen dieselbe Tabelle, Decodieren heisst nachschlagen

Welche Spalten nehmen wir?



- 16 Nachrichtenbits, 5 Kontrollbits: 16 der 26 übrigen Spalten frei wählbar
- 10 Spalten ungenutzt: 5 Kontrollbits reichen für 26 Nachrichtenbits
- Zuordnung beliebig: Beide Seiten brauchen dieselbe Tabelle, Decodieren heisst nachschlagen
- Hamming mit 4 Kontrollbits: 11 Nachrichtenbits füllen alle 15 Spalten



Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1																
2																
4																
8																
0																



Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2																
4																
8																
0																

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$

Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4																
8																
0																

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$

Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4					×	×	×	×					×	×	×	×
8																
0																

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$

Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4					×	×	×	×					×	×	×	×
8									×	×	×	×	×	×	×	×
0																

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$



Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4					×	×	×	×					×	×	×	×
8									×	×	×	×	×	×	×	×
0																

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$
- Kontrollbits an 1, 2, 4, 8: genau ein Kreuz in ihrer Spalte
- Positionen 1 bis 15 belegen alle $2^4 - 1 = 15$ nicht leeren Spalten



Hamming-Code als Überwachungstabelle

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4					×	×	×	×					×	×	×	×
8									×	×	×	×	×	×	×	×
0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$
- Kontrollbits an 1, 2, 4, 8: genau ein Kreuz in ihrer Spalte
- Positionen 1 bis 15 belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Position 0 überwacht alle Bits, wie c_9 bei der Kartentrick-Methode



Hamming-Code als Überwachungstabelle

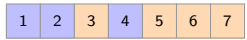
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×		×		×		×		×		×		×		×
2			×	×			×	×			×	×			×	×
4					×	×	×	×					×	×	×	×
8									×	×	×	×	×	×	×	×
0	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

- Spalte von Position $p = p$ als Binärzahl, z. B. $13 = 8 + 4 + 1$
- Kontrollbits an 1, 2, 4, 8: genau ein Kreuz in ihrer Spalte
- Positionen 1 bis 15 belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Position 0 überwacht alle Bits, wie c_9 bei der Kartentrick-Methode
- Fehler an Position 10: Es melden 2 und 8, also $2 + 8 = 10$



Was bedeuten (7,4), (15,11) und (16,11)?

(7,4)



Länge 7, davon 4 Nachrichtenbits

Kontrollbits

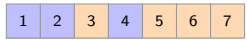
Nachrichtenbits

- Erste Zahl: Länge des Codeworts, zweite Zahl: Anzahl Nachrichtenbits



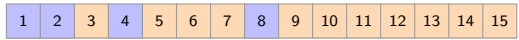
Was bedeuten (7,4), (15,11) und (16,11)?

(7,4)



Länge 7, davon 4 Nachrichtenbits

(15,11)



Länge 15, davon 11 Nachrichtenbits

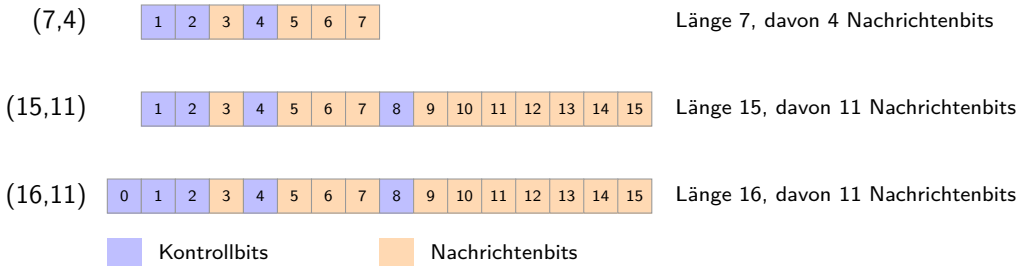
Kontrollbits

Nachrichtenbits

- Erste Zahl: Länge des Codeworts, zweite Zahl: Anzahl Nachrichtenbits
- Differenz = Anzahl Kontrollbits, an den Positionen 1, 2, 4, 8, ...



Was bedeuten (7,4), (15,11) und (16,11)?

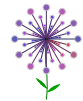


- Erste Zahl: Länge des Codeworts, zweite Zahl: Anzahl Nachrichtenbits
- Differenz = Anzahl Kontrollbits, an den Positionen 1, 2, 4, 8, ...
- $(16, 11)$: zusätzlich Position 0 für die Gesamtparität

Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111



Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111

- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)
- Position 1 (0001) überwacht alle Positionen ...1: 2. und 4. Spalte



Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111

- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)
- Position 1 (0001) überwacht alle Positionen ...1: 2. und 4. Spalte
- Position 2 (0010) überwacht alle Positionen ..1.: 3. und 4. Spalte



Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111

- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)
- Position 1 (0001) überwacht alle Positionen `...1`: 2. und 4. Spalte
- Position 2 (0010) überwacht alle Positionen `..1.`: 3. und 4. Spalte
- Position 4 (0100) überwacht alle Positionen `.1..`: 2. und 4. Zeile



Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111

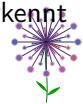
- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)
- Position 1 (0001) überwacht alle Positionen `...1`: 2. und 4. Spalte
- Position 2 (0010) überwacht alle Positionen `..1.`: 3. und 4. Spalte
- Position 4 (0100) überwacht alle Positionen `.1..`: 2. und 4. Zeile
- Position 8 (1000) überwacht alle Positionen `1...`: 3. und 4. Zeile



Hamming-Codes: Kontrollbits an Zweierpotenz-Positionen

0 0000	1 0001	2 0010	3 0011
4 0100	5 0101	6 0110	7 0111
8 1000	9 1001	10 1010	11 1011
12 1100	13 1101	14 1110	15 1111

- 16 Bit: 11 Nachrichtenbits, Kontrollbits an den Positionen 1, 2, 4, 8 (und 0)
- Position 1 (0001) überwacht alle Positionen `...1`: 2. und 4. Spalte
- Position 2 (0010) überwacht alle Positionen `..1.`: 3. und 4. Spalte
- Position 4 (0100) überwacht alle Positionen `.1..`: 2. und 4. Zeile
- Position 8 (1000) überwacht alle Positionen `1...`: 3. und 4. Zeile
- Position 0: Parität über alles, erkennt zusätzlich zwei Fehler



Auftrag

Skript „Datenintegrität“

- Beispiel 1.8 (Hamming-Code) im Skript durchgehen
- 1.23 (Hamming-Code)



FreeFlower

Hamming-Codes sind perfekt

- Ohne Position 0: 15 Positionen belegen alle $2^4 - 1 = 15$ nicht leeren Spalten



Hamming-Codes sind perfekt

- Ohne Position 0: 15 Positionen belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Jede Meldung der Kontrollbits gehört zu genau einem Bit



Hamming-Codes sind perfekt

- Ohne Position 0: 15 Positionen belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Jede Meldung der Kontrollbits gehört zu genau einem Bit
- Jede empfangene Bitfolge wird zu genau einem Codewort korrigiert: Der Code ist ***perfekt***



Hamming-Codes sind perfekt

- Ohne Position 0: 15 Positionen belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Jede Meldung der Kontrollbits gehört zu genau einem Bit
- Jede empfangene Bitfolge wird zu genau einem Codewort korrigiert: Der Code ist ***perfekt***
- Zum Vergleich 16 Nachrichtenbits, 5 Kontrollbits: 21 von 31 Spalten benutzt, 10 Meldungen gehören zu keinem Bit



Hamming-Codes sind perfekt

- Ohne Position 0: 15 Positionen belegen alle $2^4 - 1 = 15$ nicht leeren Spalten
- Jede Meldung der Kontrollbits gehört zu genau einem Bit
- Jede empfangene Bitfolge wird zu genau einem Codewort korrigiert: Der Code ist ***perfekt***
- Zum Vergleich 16 Nachrichtenbits, 5 Kontrollbits: 21 von 31 Spalten benutzt, 10 Meldungen gehören zu keinem Bit
- Perfekt genau dann, wenn $2^K - 1 = N + K$



Meldung = Fehlerposition als Binärzahl

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

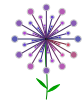


FreeFlower

Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

- Position binär = Zeile, dann Spalte:
  $13 \rightarrow 11\,01$



Meldung = Fehlerposition als Binärzahl

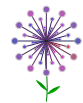
	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1

Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1
- Kontrollbit 2: zweitletzte Ziffer ist 1



Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1
- Kontrollbit 2: zweitletzte Ziffer ist 1
- Kontrollbit 4 und 8: dasselbe mit den Zeilen



Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

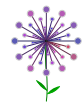
- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1
- Kontrollbit 2: zweitletzte Ziffer ist 1
- Kontrollbit 4 und 8: dasselbe mit den Zeilen



Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

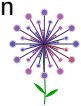
- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1
- Kontrollbit 2: zweitletzte Ziffer ist 1
- Kontrollbit 4 und 8: dasselbe mit den Zeilen
- Fehler an 13: 8, 4, 1 melden, 2 nicht $\rightarrow 1101 = 13$



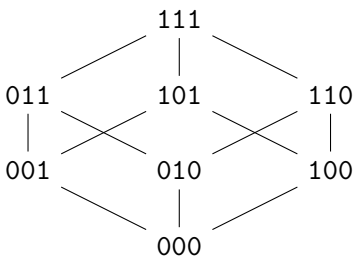
Meldung = Fehlerposition als Binärzahl

	00	01	10	11
00	0 0000	1 0001	2 0010	3 0011
01	4 0100	5 0101	6 0110	7 0111
10	8 1000	9 1001	10 1010	11 1011
11	12 1100	13 1101	14 1110	15 1111

- Position binär = Zeile, dann Spalte: $13 \rightarrow 11\,01$
- Kontrollbit 1: letzte Ziffer ist 1
- Kontrollbit 2: zweitletzte Ziffer ist 1
- Kontrollbit 4 und 8: dasselbe mit den Zeilen
- Fehler an 13: 8, 4, 1 melden, 2 nicht $\rightarrow 1101 = 13$
- Freie Tabelle: Spalte suchen, Hamming: Meldung = Position

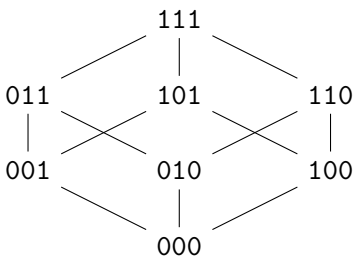


Exkurs: Geht es ohne Überwachungstabelle besser?



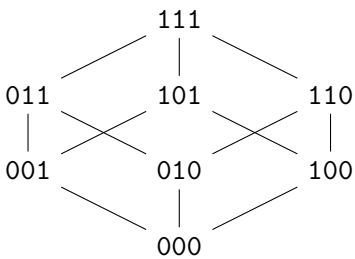
- Gilt für jeden Code, nicht nur für Codes mit Überwachungstabelle

Exkurs: Geht es ohne Überwachungstabelle besser?



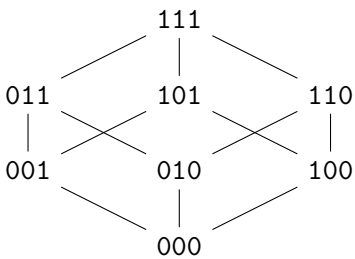
- Gilt für jeden Code, nicht nur für Codes mit Überwachungstabelle
- Punkt: Bitfolge der Länge L

Exkurs: Geht es ohne Überwachungstabelle besser?



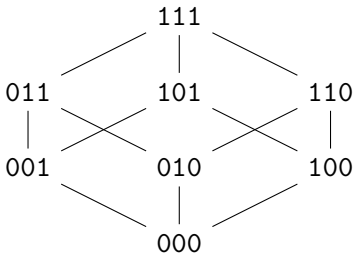
- Gilt für jeden Code, nicht nur für Codes mit Überwachungstabelle
- Punkt: Bitfolge der Länge L
- Kante: Die Bitfolgen unterscheiden sich in genau einem Bit

Exkurs: Geht es ohne Überwachungstabelle besser?

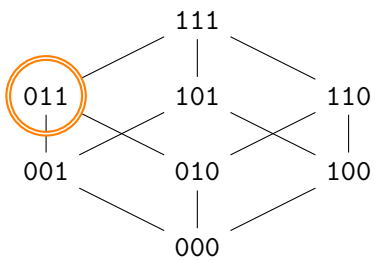


- Gilt für jeden Code, nicht nur für Codes mit Überwachungstabelle
- Punkt: Bitfolge der Länge L
- Kante: Die Bitfolgen unterscheiden sich in genau einem Bit
- Ein Bitfehler = ein Schritt entlang einer Kante

Zonen eines 1-fehlerkorrigierenden Codes

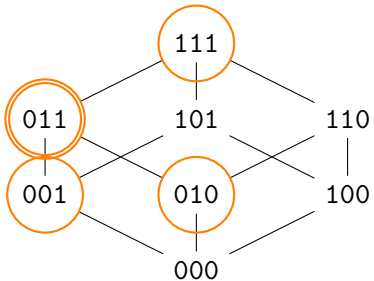


Zonen eines 1-fehlerkorrigierenden Codes



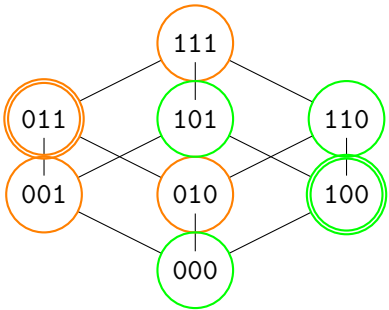
- Codewort 011 (doppelter Kreis)

Zonen eines 1-fehlerkorrigierenden Codes



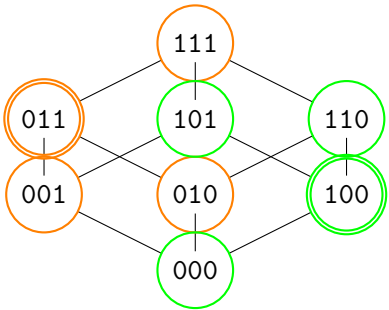
- Codewort 011 (doppelter Kreis)
- Seine Zone: Codewort und seine L Nachbarn

Zonen eines 1-fehlerkorrigierenden Codes



- Codewort 011 (doppelter Kreis)
- Seine Zone: Codewort und seine L Nachbarn
- Zweites Codewort 100 mit Zone. Zonen dürfen sich nicht überlappen

Zonen eines 1-fehlerkorrigierenden Codes



- Codewort 011 (doppelter Kreis)
- Seine Zone: Codewort und seine L Nachbarn
- Zweites Codewort 100 mit Zone. Zonen dürfen sich nicht überlappen
- Die Zonen füllen den Würfel lückenlos aus: Der Code ist ***perfekt***