Selbstkorrigierende Kodierungen: Erweiterungen des Kartentricks auf 2- und 3-fehlerkorrigierend

C. Busenhart, L. Diana, L. Moser

1 Einleitung

Nachdem wir zuletzt eine 1-fehlerkorrigierende Kodierung kennengelernt hatten in Form des Kartentricks, wollen wir in den nächsten beiden Doppelstunden diesen Kartentrick zuerst auf 2-fehlerkorrigierend erweitern und in einem weiteren Schritt sogar auf 3-fehlerkorrigierend.

Dazu werdet ihr als SuS zuerst selber versuchen, den Kartentrick auf 2-fehlerkorrigierend zu erweitern. Im Anschluss daran werden wir Vor- und Nachteile der gefundenen Lösungen diskutieren. In der zweiten Doppelstunde werden wir die beste Möglichkeit der Erweiterung auf 2-fehlerkorrigierend erklären. Im Anschluss daran werden wir eine Erweiterung auf 3-fehlerkorrigierend diskutieren.

Das Ziel ist einerseits, dass ihr diese 2-fehlerkorrigierende Kodierung versteht und eine Intuition für die 3-fehlerkorrigierende Kodierung entwickelt. Andererseits sollt ihr ein Bewusstsein dafür entwickeln, wie der Aufwand für n-fehlerkorrigierend ist und mit zunehmendem n ansteigt. Dies ist ein genereller Sachverhalt bei der Realisierung von jeglichen technischen Systemen. Zusätzliche Robustheit oder Redundanz führt immer zu einem zusätzlichen Aufwand: *"There is no such thing as a free lunch."*

Als weiteren Effekt erhöhen die zusätzlichen Bits, die der Fehlerkorrektur dienen, wiederum die Wahrscheinlichkeit, dass ein Code-Wort Fehler enthält.

Wie ihr gleich sehen werdet, ist die Erweiterung auf n-fehlerkorrigierend wichtig (für $n > 1$), da wir in der Realität nicht einfach davon ausgehen können, dass nur 1 Fehler passiert. Dazu aber gleich mehr. Ich bin sicher, dass wir bei der Eigenentwicklung und Diskussion von möglichen 2-fehlerkorrigierenden Kodierungen auch unseren Spass in der Klasse haben werden. Aktive Mitarbeit und Out-of-the-Box-Denken ist ausdrücklich

erwünscht, und vielleicht findet ja jemand die Lösung, die gemeinhin als ideal angeschaut wird.

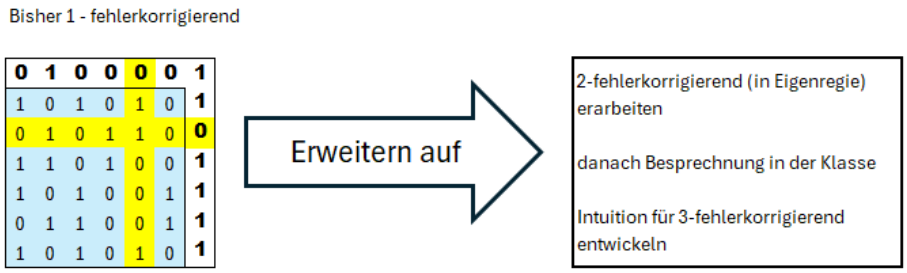


Abbildung 1: Zusammenfassung unseres geplanten Vorgehens

Repetitionsaufgaben

Repetitionsaufgabe 1) *Wenn man eine Kodierung haben will, die* ***1-fehlererkennend*** *ist, wie gross muss der Hamming-Abstand mindestens sein? Begründe deine Antwort kurz.*

Repetitionsaufgabe 2) *Wenn man eine Kodierung haben will, die* ***n-fehlererkennend*** *ist, wie gross muss der Hamming-Abstand mindestens sein? Begründe deine Antwort kurz.*

Repetitionsaufgabe 3) *Wenn man eine Kodierung haben will, die* ***1-fehlerkorrigierend*** *ist, wie gross muss der Hamming-Abstand mindestens sein? Begründe deine Antwort kurz.*

Repetitionsaufgabe 4) *Wenn man eine Kodierung haben will, die* ***n-fehlerkorrigierend*** *ist, wie gross muss der Hamming-Abstand mindestens sein? Begründe deine Antwort kurz.*

Repetitionsaufgabe 5) *Finde das falsche Bit mittels des Kartentricks:*

0	**1**	**0**	**0**	**0**	**0**	**1**
1	0	1	0	1	0	**1**
0	1	0	1	0	0	**0**
1	1	0	1	0	0	**1**
1	0	1	0	0	1	**1**
0	1	1	1	0	1	**1**
1	0	1	0	1	0	**1**

Für Lösungen zu allen Aufgaben, siehe Abschnitt 4.

Motivationsaufgaben

Motivationsaufgabe 1) *Wenn man eine Kodierung, die **1-fehlererkennend** ist, mittels Repetitionen des Nachrichtenwortes realisieren will, wie oft muss das Nachrichtenwort gesendet werden? Begründe deine Antwort kurz.*

Motivationsaufgabe 2) *Wenn man eine Kodierung, die **n-fehlererkennend** ist, mittels Repetitionen des Nachrichtenwortes realisieren will, wie oft muss das Nachrichtenwort gesendet werden? Begründe deine Antwort kurz.*

Motivationsaufgabe 3) *Wenn man eine Kodierung, die **1-fehlerkorrigierend** ist, mittels Repetitionen des Nachrichtenwortes realisieren will, wie oft muss das Nachrichtenwort gesendet werden? Begründe deine Antwort kurz.*

Motivationsaufgabe 4) *Wenn man eine Kodierung, die **n-fehlerkorrigierend** ist, mittels Repetitionen des Nachrichtenwortes realisieren will, wie oft muss das Nachrichtenwort gesendet werden? Begründe deine Antwort kurz.*

Für Lösungen zu allen Aufgaben, siehe Abschnitt 4.

Korrektur von mehr als 1 Fehler - Motivation

Aufgabe 1) *Nehmen wir an, ein Nachrichtenwort sei 16 Bit lang, also 2 Byte, und wir haben ein Netzwerk, bei dem ein übertragenes Bit mit einer Wahrscheinlichkeit von $p = 0.01$ einen Bit-Flip erleidet: Wie hoch ist die Wahrscheinlichkeit, dass **mindestens 2 Bit** in diesem Nachrichtenwort geflippt sind?*
Überlege schrittweise:

- *Ein übertragenes 16-Bit-Nachrichtenwort kann entweder 0 oder 1 oder 2, ..., oder 15 oder 16 Bit haben, die geflippt sind. Eines dieser 17 möglichen Ergebnisse wird garantiert eintreten, und damit ist die Wahrscheinlichkeit dass eines dieser 17 Ergebnisse eintritt 1.*
- *Jetzt überlege dir, wie hoch die Wahrscheinlichkeit ist, dass kein Bit geflippt ist und wie hoch die Wahrscheinlichkeit ist, dass genau 1 Bit geflippt ist.*
- *Berechne aus diesen Resultaten die Wahrscheinlichkeit, dass mindestens 2 Bit geflippt sind.*

Wenn wir diese 16 Bit eines Nachrichtenwortes mittels des bereits bekannten 1-fehlerkorrigierenden Kartentricks absichern, erhalten wir Code-Worte der Länge 25 Bit.

- *Berechne nach derselben Systematik erneut die Wahrscheinlichkeit, dass von den 25 Bit eines Code-Wortes **mindestens 2 Bit** geflippt sind.*

Nun hat jemand die Idee, anstelle des bisher kennengelernten 1-fehlerkorrigierenden Kartentricks stattdessen mittels simpler Verdreifachung des Nachrichtenwortes eine 1-fehlerkorrigierende Kodierung zu realisieren. Das Code-Wort besteht also nun aus dreimal dem Nachrichtenwort und ist 48 Bit lang. Ein Fehler wird korrigiert, indem das Code-Wort in die 3 Nachrichtenworte rückaufgeteilt wird und geschaut wird, welche beiden Nachrichtenworte übereinstimmen. Wenn das Code-Wort nur 1 Fehler hat, kann auch nur eines der drei Nachrichtenworte einen Fehler haben, und die anderen beiden Nachrichtenworte müssen übereinstimmen und fehlerfrei sein. Damit ist das korrekte Nachrichtenwort eindeutig identifizierbar.

- *Berechne für diese Kodierung wieder nach derselben Systematik die Wahrscheinlichkeit, dass von den 48 Bit eines Code-Wortes* ***mindestens 2 Bit*** *geflippt sind.*

Schaut man sich die Resultate der obigen Berechnungen an (vgl. Lösungen in Abschnitt 4), stellen wir zwei wichtige Sachverhalte fest:

- Es besteht ein Interesse, mehr als 1 Fehler korrigieren zu können
- Die Korrekturmethode sollte mit möglichst wenig zusätzlichen Bits auskommen - nicht nur, um die Übertragungsbandbreite zu schonen, sondern auch, um die Wahrscheinlichkeit zu minimieren, dass ein Code-Wort mehr Fehler enthält als die gewählte Kodierung korrigieren kann.

  Oder in anderen Worten: Haben 2 unterschiedliche Kodierungen beide die Eigenschaft, bei Nachrichtenworten gleicher Länge maximal n Fehler in einem Code-Wort korrigieren zu können, hat diejenige Kodierung, die längere Code-Wörter nutzt, eine höhere Wahrscheinlichkeit, dass nicht-korrigierbare Code-Wörter (= mit n+1 oder mehr Fehlern) beim Empfänger ankommen [unter der Annahme, dass bei beiden Kodierungen die Bit-Flip-Wahrscheinlichkeit p identisch ist].

Es ist offensichtlich, dass die Wahrscheinlichkeit solcher nicht korrigierbarer falscher Code-Wörter minimiert werden sollte. Eine Korrektur mittels mehrfachen Sendens des Nachrichtenwortes ist in diesem Sinne also keine gute Kodierung, weil sie viele zusätzliche Bits benötigt. Andere Kodierungen, die dieselbe "Korrekturfähigkeit"haben, aber weniger zusätzliche Bits benötigen, sind zu bevorzugen.

Die beiden obigen Anforderungen (möglichst viele Fehler in einem Code-Wort korrigieren zu können vs. möglichst wenig zusätzliche Bits) stehen im Widerspruch zueinander, und es braucht in der Realität eine Abwägung, wie man diese beiden Anforderungen austariert. Dies weiter auszuführen würde aber an dieser Stelle zu weit führen. Deswegen soll es nun weitergehen mit dem 2-fehlerkorrigierenden Kartentrick.

Für Lösungen zu allen Aufgaben, siehe Abschnitt 4.

2 2-fehlerkorrigierender Kartentrick

Wir haben gelernt, dass eine 2-fehlerkorrigierende Kodierung einen Abstand von mindestens $2 * 2 + 1 = 5$ haben muss. Die Kartentrick-Kodierung hat jedoch nur einen Abstand von 4. Das heisst, es gibt ungültige Code-Wörter mit 2 Fehlern, welche der Kartentrick nicht eindeutig einem nächstgelegenen (im Sinne des Hamming-Abstandes) gültigen Code-Wort zuordnen und dadurch korrigieren kann.

Aufgabe 2) *2-Fehler im Kartentrick*
Öffne das Excel-Kartentrick-Tool und navigiere zum Tabellenblatt für die 1-Fehlerkorrektur. Dazu kannst Du entweder Punkt 1 der Navigation im Inhaltsverzeichnis verwenden oder direkt das Tabellenblatt 1-Fehlerkorrektur anwählen. In der linken Kartentrick-Illustration, welche mit ***Übermittelte Nachricht*** *bezeichnet ist, kannst Du einzelne Bits flippen, also entweder eine* 0 *mit einer* 1 *oder eine* 1 *mit einer* 0 *überschreiben. Nun werden die Zeilen und Spalten, bei welchen die Kontrollbits, bei welchen die Parität nun verletzt ist, gelb eingefärbt.*

Baue auf diese Art 2 Fehler in die linke Abbildung ein. Frage nun Deinen Tischnachbarn, ob er die Fehler eindeutig identifizieren kann. Dabei soll die ursprüngliche Nachricht, also die Darstellung auf der rechten Seite, nicht betrachtet werden.
Dein Beispiel könnte etwa wie in der untenstehenden Illustration aussehen.

0	**1**	**0**	**0**	**0**	**0**	**1**
1	0	1	0	1	0	**1**
0	0	0	1	0	0	**0**
1	1	0	1	0	0	**1**
1	0	1	0	0	1	**1**
0	1	1	0	1	1	**1**
1	0	1	0	1	0	**1**

Ist es möglich, dass dieselbe Konstellation durch jeweils zwei Fehler, welche sich bei unterschiedlichen ursprünglichen Nachrichten eingeschlichen haben, entstehen?
Versucht dies gegenseitig mit unterschiedlichen Fehlerkonstellationen. Natürlich dürfen auch Fehler in den Kontrollbits eingebaut werden. Was fällt Euch dabei auf?
Begründe, warum der Kartentrick 2 Fehler nie eindeutig korrigieren kann.

Im Folgenden möchten wir den Kartentrick so erweitern, dass dieser auch 2 Fehler korrigieren kann. Die Fehlerkorrektur im Kartentrick funktioniert mittels Paritäten über die Zeilen und Spalten der Nachricht. Wir wollen uns nun überlegen, mit welcher zusätzlichen Redundanz wir genügend Informationen erhalten, um auch 2 Fehler zu korrigieren.

Aufgabe 3) *Diskutiere mit deinem Banknachbarn, wie der Kartentrick erweitert werden könnte, um 2-fehlerkorrigierend zu sein. Betrachtet dabei nochmals die fehlerhaften Code-Wörter aus Aufgabe 2. Habt ihr eine Idee, mit welchen zusätzlichen Informationen bzw. Kontrollbits die fehlerhaften Bits eindeutig bestimmt werden könnten?*
Tipp: Wir haben, weiter oben gesehen, dass der 1-fehlerkorrigierende Kartentrick einen Abstand von 4 aufweist. Dieser Abstand kann auch wie folgt als Funktion der überwachenden Bits verstanden werden: Wenn sich 2 Nachrichtenworte in nur 1 Bit unterscheiden, unterscheiden sich deren Code-Wörter zusätzlich in den drei Kontrollbits, welche das Nachrichtenbit überwachen, was in einem Abstand von 4 resultiert. Eine mögliche Strategie, um eine 2-fehlerkorrigierende Kodierung mit Abstand 5 zu finden, wäre es also die Kodierung so anzupassen, dass jedes Nachrichtenbit von 4 Kontrollbits überwacht wird.

Aufgabe 4) *Einer anderen Klasse wurde dieselbe Aufgabe gestellt. Die SuS hatten dabei einige Ideen für Erweiterungen des Kartentricks. Begründe für die nachfolgenden Erweiterungen, ob diese Erweiterungen korrekt, d.h. 2-fehlerkorrigierend sind: Falls du glaubst, dass sie korrekt sind, versuche zu erklären, warum. Falls du glaubst, dass sie falsch sind, kannst du ebenfalls begründen, warum, oder es reicht auch 2 Code-Wörter mit Abstand $d < 5$ zu finden. Falls es unter den Beispielen mehr als eine korrekte Kodierung gibt, versuche diese zu vergleichen: Wieviele Bits werden für die Code-Wörter gebraucht?*

Beispiel 1: Adam schlägt vor, zusätzlich eine Kopie des Kartentrick-Code-Worts mitzusenden und somit die Länge des Code-Worts zu verdoppeln.

Beispiel 2: Bettina schlägt vor, nicht das ganze Code-Wort, sondern nur eine zusätzliche Kopie der Kontrollbits zu senden.

Beispiel 3: Christian schlägt vor, dass wir uns auf Nachrichten, welche in Rechtecke mit geraden Seitenzahlen a, b passen, beschränken. Dann könnte das Rechteck in vier Unter-Rechtecke mit Seitenlängen $a/2$ und $b/2$ unterteilt werden. Für diese vier Unter-Rechtecke werden dann Kontrollbits berechnet und zusätzlich übermittelt. Siehe dazu Abbildung 2, welche den Fall für ein $6x6$-Quadrat zeigt.

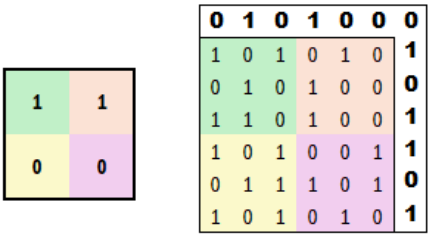


Abbildung 2: Ein mögliches Code-Wort in Beispiel 3

Beispiel 4: Doris hat die Idee, dass zusätzlich noch Kontrollbits für die Linksdiagonalen, d.h. von links unten nach rechts oben, berechnet werden könnten.

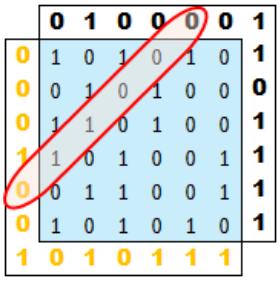


Abbildung 3: Eine mögliches Code-Wort in Beispiel 4

2.1 Erweiterung mit Diagonalen

In der Folge wollen wir die Idee von Doris mit der Erweiterung durch Diagonalen weiter verfolgen. Um sich damit etwas vertrauter zu machen, wurde Euch ein kleines Excel-Kartentrick-Tool zur Verfügung gestellt. Macht euch doch erst mal 2 Minuten damit vertraut. Das Excel-Kartentrick-Tool zeigt eine Nachricht (blau hinterlegt) auf der rechten Seite. Die Nachricht ist fest mit 0 und 1 kodiert. Die Kontrollbits des ursprünglichen Kartentricks sind in schwarzer Fettschrift dargestellt und die Diagonal-Kontrollbits der Erweiterung sind in oranger Fettschrift dargestellt. Auf der linken Seite sieht man eine Kopie dieses Code-Worts, in welches nun Fehler eingebaut werden können (durch Überschreiben der entsprechenden Zellen mit 0 oder 1). Um Euch Rechenarbeit zu ersparen, werden folgende Sachverhalte direkt hervorgehoben:

- Zellen mit Bit-Flips werden rot hinterlegt,

- Fehlerzeilen- und Spalten des ursprünglichen Kartentricks werden gelb hinterlegt und
- Fehlerdiagonalen der Erweiterung werden mit roter Fettschrift hervorgehoben.

Mittels des *Reset*-Buttons kann das ursprüngliche Code-Wort wiederhergestellt werden.
Untenstehend zur Veranschaulichung ein Beispiel mit einem Fehler.

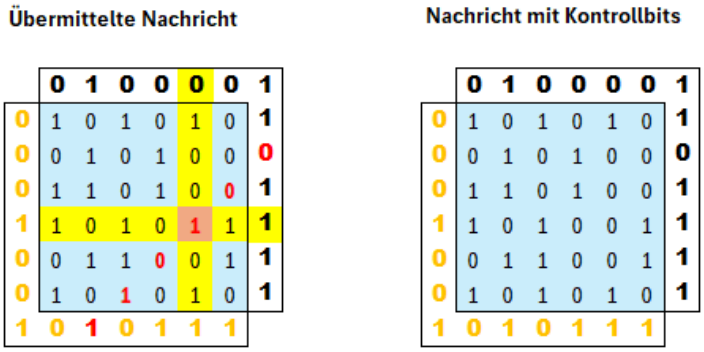


Abbildung 4: Ansicht Excel-Kartentrick-Tool

Diesen Fehler hätten Sie selbstverständlich mit dem ursprünglichen Kartentrick sofort zuverlässig korrigiert. Wir wollen uns jetzt aber der Situation mit 2 Fehlern zuwenden.

Aufgabe 5) *Verteilung von 2 Fehlern*
Überlege Dir, wie sich 2 Fehler in der Nachricht verteilen können. Denke dabei auch an Aufgabe 3, d.h., welche Fehlersituationen im ursprünglichen Kartentrick nicht korrigiert wurden. Beschreibe in diesen unterschiedlichen Fällen, wie die Kontrollbits auf der Diagonalen helfen, die Fehler zu entdecken.

Damit ist uns die Erweiterung des Kartentricks auf 2 Fehler bereits gelungen.

Aufgabe 6) *3 Fehler im erweiterten Kartentrick*
Linus weist darauf hin, dass mit der Erweiterung auf die Diagonalen selbst 3 Fehler korrigiert werden können. Er führt dazu die beiden folgenden Beispiele an.

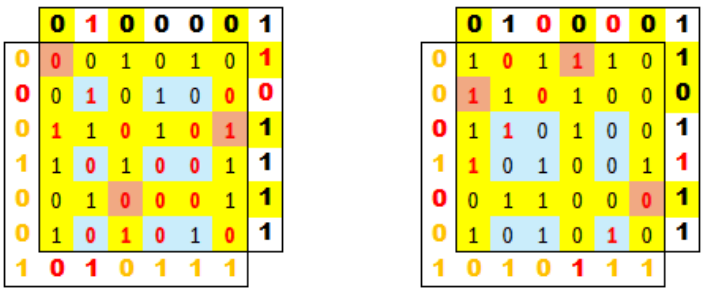


Abbildung 5: 3 Fehler durch Kartentrick erkannt

Loris wendet ein, dass dies vielleicht nur funktioniert, weil alle 3 Fehler auf unterschiedlichen Zeilen und Spalten auftreten, dies aber nicht allgemein der Fall sein muss. Christian entdeckt aber sogar ein Beispiel, in welchem die Fehler sogar in diesem Spezialfall (alle Fehler auf unterschiedlichen Zeilen und Spalten und somit auch Diagonalen) nicht korrigiert werden können. Versuche im Excel-Kartentrick-Tool ebenfalls eine Konstellation zu finden, wie sie Christian entdeckt hat.

Die vorherige Aufgabe zeigt, dass die Erweiterung des Kartentricks auf die Linksdiagonalen nicht ausreicht, um 3 Fehler zu korrigieren. Wir hatten schon gesehen, dass der Abstand der Erweiterung mindestens 5 ist. Nun wissen wir auch, dass er höchstens 6 sein kann:

$$\text{2-fehlerkorrigierend} \Rightarrow \text{Abstand} \geq 2 * 2 + 1 = 5$$
$$\text{nicht 3-fehlerkorrigierend} \Rightarrow \text{Abstand} < 2 * 3 + 1 = 7$$

Der Abstand des erweiterten Kartentricks ist damit entweder 5 oder 6. Das wollen wir noch etwas genauer untersuchen. Dazu gibt es im Excel-Kartentrick-Tool ein eigenes Tabellenblatt, welches mit *Abstandsbetrachtungen* beschriftet ist.
Wir betrachten einen Fall, in dem sich zwei Code-Wörter lediglich in einem Nachrichtenbit unterscheiden.

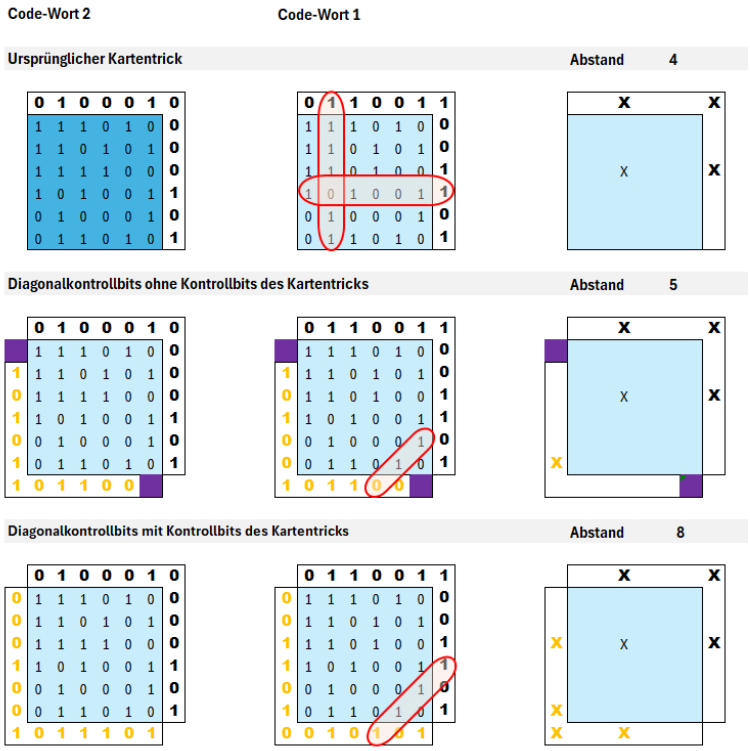


Abbildung 6: Abstand bei Code-Wörtern, mit einem Unterschied in den Nachrichtenbits

Die roten Umrahmungen zeigen an, welche Bits bei der Berechnung der Kontrollbits berücksichtigt werden. Während dies für den ursprünglichen Kartentrick klar ist, kann man bei der Erweiterung auf die Diagonale zwei Fälle unterscheiden. Im ersten Fall werden die Diagonalkontrollbits nur aus den Nachrichtenbits berechnet. Im zweiten Fall werden auch die Kontrollbits des ursprünglichen Kartentricks mit einbezogen. Abbildung 6 zeigt die Situation bei einem Code-Wort-2, welches sich in einem Nachrichtenbit von Code-Wort-1 unterscheidet.
Im Fall des ursprünglichen Kartentricks ergibt sich wie bereits untersucht Abstand 4. Interessant sind die beiden Fälle mit der Diagonalerweiterung. Falls die Diagonalkontrollbits nur die Nachrichtenbits kontrollieren, ergibt sich in der Situation Abstand 5, während sich bei Mitberücksichtigung der Kontrollbits des ursprünglichen Kartentricks bei der Berechnung der Diagonalkontrollbits ein Abstand von 8 ergibt.

Aufgabe 7) *Abstand 8 in oben dargestellter Version der Erweiterung des Kartentricks, liesse ja sogar auf die Möglichkeit einer 3-Fehlerkorrektur hoffen. Aufgrund des Gegenbeispiels aus Abbildung 5 wissen wir aber bereits,*

dass dies nicht sein kann. Es müssen also 2 Code-Wörter existieren, welche Abstand kleiner oder gleich 6 haben.
Versuche eine solche Konstellation im Excel-Kartentrick-Tool darzustellen.

Ohne hier eine genaue Beweisführung darzustellen, lässt sich in der Tat zeigen, dass die Erweiterung, welche in den Diagonalkontrollbits lediglich die Nachrichtenbits berücksichtigt, Abstand 5 hat, während die Erweiterung, welche die ursprünglichen Kontrollbits miteinbezieht, Abstand 6 aufweist und somit einen Fehler mehr erkennen kann. Dabei ist aber auch zu bemerken, dass die Erweiterung mit Abstand 5 mit 2 Kontrollbits weniger auskommt, welche in der Abbildung 6 violett gekennzeichnet sind, um zwei Fehler zu erkennen und also effizienter ist oder auch der grössere Abstand bei Berücksichtigung der ursprünglichen Kontrollbits bei den Diagonalbits mit zusätzlicher Redundanz einhergeht.

Die folgenden Ausführungen sind für speziell interessierte SuS gedacht. Bei einer schnelleren Bearbeitung kann man gleich mit Kapitel 2.2 fortfahren.

Nun erinnern wir uns, dass der ursprüngliche Kartentrick auch ohne das Kontrollbit in der rechten oberen Ecke auskommt, dann aber nur Abstand 3, was für 1-Fehlerkorrektur ja ausreichend ist, aufweist. Dies lässt vermuten, dass wir durch Weglassen dieses Kontrollbits, die Erweiterung des Kartentricks noch effizienter gestalten könnten.

Untenstehende Abbildung 7 soll diese Situation verdeutlichen.

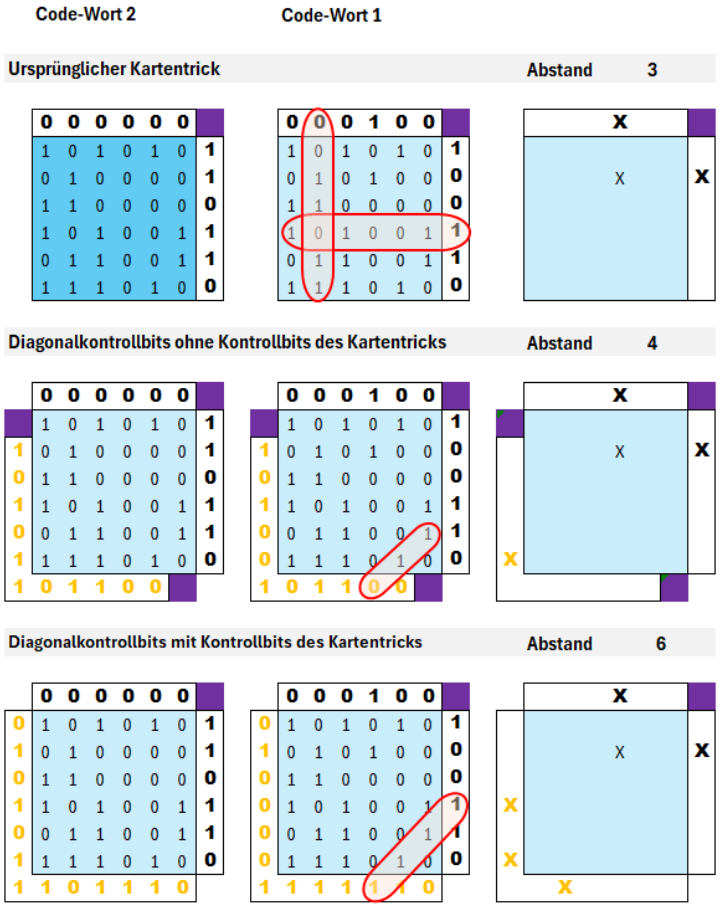


Abbildung 7: Abstand ohne Berücksichtigung des Eck-Kontrollbits des ursprünglichen Kartentricks

Wir erkennen hier, dass der zusätzliche Verzicht auf das Eck-Kontrollbit des ursprünglichen Kartentricks, bei der Version der Erweiterung, welche bei den neuen Kontrollbits nur die Nachrichtenbits berücksichtigt, nun lediglich mehr Abstand 4 aufweist und also nicht mehr 2-fehlerkorrigierend sein kann.

Aufgabe 8) *Finde zwei Code-Wörter mit je 2 Fehlern, welche durch die entsprechende Erweiterung ohne Eck-Kontrollbit nicht erkannt werden können.*
Tipp: *Betrachte die entsprechenden Code-Wörter 1 und 2 aus Abbildung 7 und versuche, die Fehler je auf die unterschiedlichen Bits der beiden Code-Wörter zu verteilen.*

Die vorangehenden Überlegungen sollten ein vertiefteres Verständnis zur

Bedeutung des Abstands in Bezug auf die Fehlerkorrektur vermittelt haben. Bewaffnet mit diesen Erkenntnissen, wollen wir uns nun aber noch der 3-Fehlerkorrektur zuwenden. Wir müssen also mit der Erweiterung noch mehr Abstand schaffen, bzw. für die Kodierung zumindest Abstand 7 erreichen.

2.2 Zweite Erweiterung mit Links- und Rechtsdiagonalen

Wir betrachten die folgende Erweiterung des Kartentricks auf die zweite Diagonale:

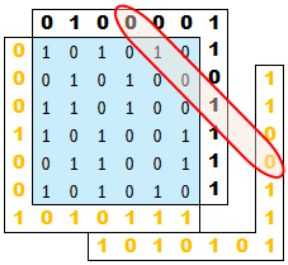


Abbildung 8: Zweite Erweiterung

Um 3 Fehler korrigieren zu können, muss eine Kodierung mindestens Abstand $2 * 3 + 1 = 7$ aufweisen. Wir wollen an dieser Stelle darauf verzichten, genau darzulegen, dass die vorgeschlagene Erweiterung diese Bedingung immer erfüllt. Wir geben uns damit zufrieden, folgende Abstandsbetrachtung vorzunehmen:

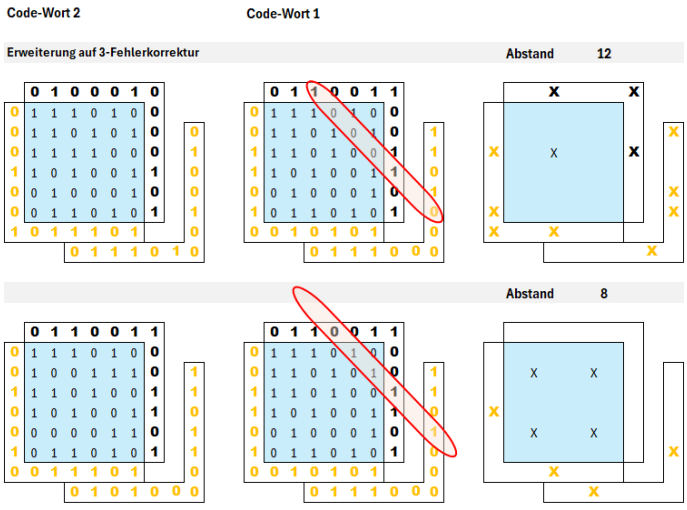


Abbildung 9: Abstand zweite Erweiterung

Zwei Code-Wörter, welche sich nur in einem Nachrichtenbit unterscheiden, weisen Abstand 12 auf. Die dargestellte Situation mit 4 Unterschieden in den Nachrichtenbits führt dazu, dass sich keine Unterschiede bei den Kontrollbits des ursprünglichen Kartentricks ergeben und sich zusätzlich die Unterschiede auf den Diagonal-Kontrollbits teilweise aufheben. Dies soll uns hier genügen, um uns davon zu überzeugen, mit der zweiten Erweiterung genug Abstand für die 3-Fehlerkorrektur geschaffen zu haben.
3 Fehler würden sich in dieser Erweiterung wie folgt präsentieren. Es handelt sich dabei um das Beispiel aus der vorherigen Aufgabe, bei welchem die erste Erweiterung des Kartentricks die Fehler nicht eindeutig identifizieren konnte.

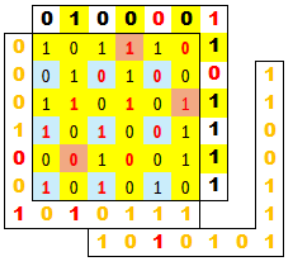


Abbildung 10: Drei Fehler in zweiter Erweiterung

Mithilfe der zusätzlichen Rechtsdiagonalen können sämtliche Fehlerkonstellationen mit 3 Fehlern korrigiert werden. Die Korrektur von 3 Fehlern teilt sich aber in verschiedene Fälle auf und ist nicht mehr so übersichtlich. Wir gehen deswegen gemeinsam dieses Beispiel durch.

Bevor wir dies jedoch in Angriff nehmen, wollen wir einer wichtigen Bemerkung von Loris Raum verschaffen.
Loris teilt nämlich eine wichtige Erkenntnis mit der Klasse: da wir ja bereits zwei Fehler korrigieren können, reicht es vollkommen aus, wenn wir mit einer zusätzlichen Erweiterung einen der 3 Fehler eindeutig identifizieren und damit korrigieren können. Ist dieser erste Fehler korrigiert, wissen wir ja inzwischen, wie die verbleibenden 2 Fehler korrigiert werden können.
Dies wollen wir in der Folge berücksichtigen und verwenden.

Christian hat bei Aufgabe 6 folgendes Code-Wort mit 3 Fehlern entdeckt, welches nur mit Linksdiagonalen nicht korrigiert werden konnte:

Abbildung 11: Code-Wort mit 3 Fehlern von Christian

Bei 3 Fehlern in unterschiedlichen Zeilen und Spalten, bilden diese aus Sicht der ursprünglichen Kartentrick-Kodierung eine Art Fehlergitter, bestehend aus 9 Kreuzungen, siehe Abbildung 12. Die 3 Fehler müssen sich innerhalb dieser 9 Kreuzungen befinden.

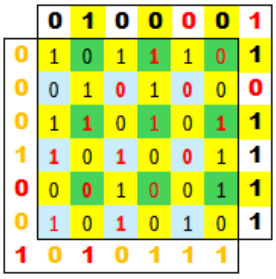


Abbildung 12: Kreuzungen (grün), wo sich Fehler befinden könnten.

Falls eine Linksdiagonale durch genau eine solche Kreuzung geht, kann ein Fehler lokalisiert werden. Dies ist in den beiden Beispielen in Aufgabe 6 der Fall. In diesem Beispiel gehen jedoch alle Linksdiagonalen immer durch mehrere Kreuzungen. Folgende zwei Fehlerkonstellationen haben dadurch die exakt gleichen Kontrollbits und können deswegen nicht unterschieden werden:

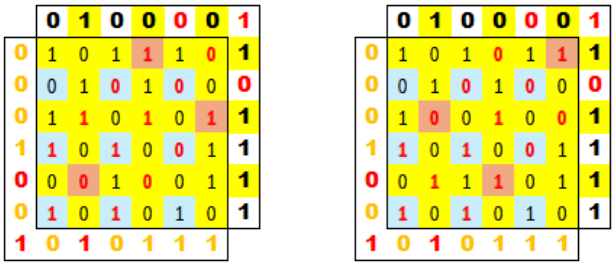


Abbildung 13: Fehlerkonstellationen mit gleichen Kontrollbits

Mit den zusätzlichen Rechtsdiagonalen erhalten wir nun weitere Kontrollbits, so dass die beiden Fehlerkonstellationen unterschieden werden können:

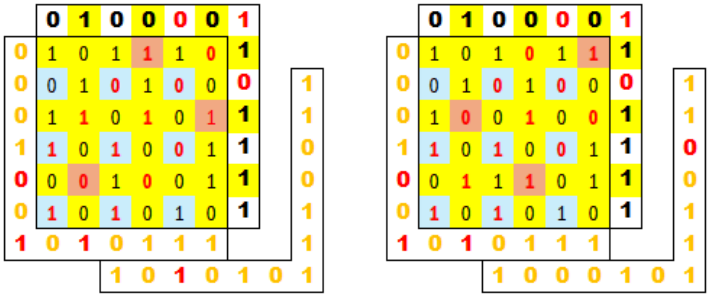


Abbildung 14: Gleiche Fehlerkonstellationen wie in Abbildung 13 mit zusätzlichen Rechtsdiagonalen

Für beide Fehlerkonstellationen läuft eine Rechtsdiagonale nämlich durch nur eine Kreuzung des Fehlergitters und kann dadurch diesen Fehler exakt lokalisieren. Nachdem dieser Fehler korrigiert ist, können die verbleibenden 2 Fehler mithilfe der ersten Erweiterung, also den Linksdiagonalen, korrigiert werden. Das Vorgehen für die Korrektur von 3 Fehlern auf jeweils unterschiedlichen Zeilen und Spalten ist also wie folgt:

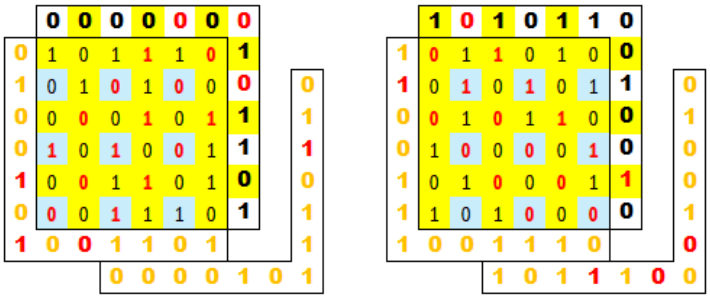
1. Fehlergitter mit 9 Kreuzungen identifizieren
2. Fehlersuche: Überprüfen, ob eine Linksdiagonale durch exakt 1 Kreuzung verläuft.

   Falls ja: mit 3. fortfahren.

   Falls nein: Fehler mittels Rechtsdiagonalen lokalisieren und dann mit 3. fortfahren.
3. Fehlerkorrektur: Lokalisierten Fehler korrigieren und restliche Fehler mittels Vorgehen der ersten Erweiterung korrigieren.

Abbildung 15: Vorgehen zur Fehlerkorrektur von 3 Fehlern auf unterschiedlichen Zeilen und Spalten.

Aufgabe 9) *In folgenden 2 Code-Wörtern haben sich 3 Fehler auf unterschiedlichen Zeilen und Spalten eingeschlichen. Versuche diese zu finden.*



Wir haben nun einen Fall gesehen, wo mithilfe von Rechtsdiagonalen 3 Fehler korrigiert werden können. Wir nehmen vorweg, dass diese zweite Erweiterung tatsächlich in allen Fällen 3 Fehler korrigieren kann. Für eine vollständige Betrachtung müsste selbstverständlich noch die Korrektur von weiteren unterschiedlichen Fehlerkonstellationen betrachtet werden, was wir hier weglassen.

3 Zusammenfassung und Ausblick

Wir haben gelernt, dass es bei den heutzutage grossen Datenmengen, die übermittelt werden, durchaus Sinn macht selbstkorrigierende Kodierungen zu verwenden, welche 2 oder mehr Fehler korrigieren. Mit dem erweiterten Kartentrick haben wir zum ersten Mal eine effiziente 2- bzw. 3-fehlerkorrigierende Kodierung für beliebig lange Nachrichten entwickelt und uns näher angeschaut. Unsere Erweiterungen haben, wie der ursprüngliche Kartentrick, mit Paritäten über verschiedene, anschauliche Teilmengen (Zeilen, Spalten und Diagonalen) der Nachricht gearbeitet. Es gibt auch bekannte Kodierungen, welche nach diesem Schema funktionieren. Der optimale, 1-fehlerkorrigierende Hamming-Code gehört zum Beispiel dazu. Für eine beliebige Nachrichtenlänge ist es sehr schwierig, eine 2- oder mehr fehlerkorrigierende Kodierung zu finden. Wenn wir uns an die Fehlermelde-Tabellen im Buch zurückerinnern, entspräche das für eine Nachrichtenlänge von n und k Kontrollbits dem Finden von n Spalten der Länge k, welche sowohl einzeln, als auch über 2 bzw. mehr aufsummiert, noch eindeutig Fehlern an spezifischen Stellen der Code-Wörter zugeordnet werden können. Heutzutage eher verbreitet sind eine andere Klasse von selbstkorrigierenden Codes, nämlich solche, welche mittels Interpolation von Polynomen arbeiten. Ein Beispiel dafür sind die Reed-Solomon-Codes, welche z.B. bei der Datenspeicherung auf CDs verwendet werden. Diese Kodierungen können einfacher, mit wenig Redundanz, für die Korrektur von zusätzlichen Fehlern erweitert werden. Bezahlt wird dies jedoch mit einem höheren Rechenaufwand beim Dekodieren eines Code-Worts.

4 Lösungen

Lösung zu Repetitionsaufgabe 1

Der Hamming-Abstand muss mindestens 2 sein, damit bei genau einem falschen Bit kein gültiges Code-Wort herauskommt.

Lösung zu Repetitionsaufgabe 2

Der Hamming-Abstand muss mindestens n+1 sein, damit auch bei n falschen Bits kein gültiges Code-Wort herauskommt.

Lösung zu Repetitionsaufgabe 3

Der Hamming-Abstand muss mindestens 3 sein, damit bei genau einem falschen Bit klar definiert ist, welches das nächstgelegene (im Sinne des Hamming-Abstandes) gültige Code-Wort ist.

Lösung zu Repetitionsaufgabe 4

Der Hamming-Abstand muss mindestens $2 * n + 1$ sein, damit auch bei n falschen Bits eindeutig bestimmt ist, welches das nächstgelegene (im Sinne des Hamming-Abstandes) Code-Wort ist.

Lösung zu Repetitionsaufgabe 5

Das fehlerhafte Bit kann wie folgt ermittelt werden:

0	**1**	**0**	**0**	**0**	**0**	**1**
1	0	1	0	1	0	**1**
0	1	0	1	0	0	**0**
1	1	0	1	0	0	**1**
1	0	1	0	0	1	**1**
0	1	1	1	0	1	**1**
1	0	1	0	1	0	**1**

Lösung zu Motivationsaufgabe 1

Es reicht, das Nachrichtenwort 2x zu senden, um einen Unterschied in den beiden Nachrichtenworten und damit einen Fehler zu erkennen.

Lösung zu Motivationsaufgabe 2

Das Nachrichtenwort muss (n+1)-mal gesendet werden. Sind die n Fehler auf n verschiedene Nachrichtenworte verteilt, bleibt immer ein Nachrichtenwort, das sich von den andern n Nachrichtenworten unterscheidet und damit anzeigt, dass eine fehlerhafte Übertragung stattgefunden hat. Verteilen sich die n Fehler auf weniger als n Nachrichtenworte, gilt der Sachverhalt analog.

Würden nur n Nachrichtenworte gesendet, ist es möglich, dass sich die n Fehler auf die n Nachrichtenworte verteilen, und in jedem Nachrichtenwort an derselben Stelle sind. Damit hätten wir wieder n identische Nachrichtenworte, und eine fehlerhafte Übertragung wäre nicht zu erkennen.

Lösung zu Motivationsaufgabe 3

Es reicht, das Nachrichtenwort 3x zu senden, um einen Fehler zu korrigieren, weil dann 2 korrekte identische Nachrichtenworte und ein abweichendes fehlerhaftes Nachrichtenwort übertragen werden. Damit lassen sich die beiden identischen korrekten Nachrichtenworte erkennen.

Würden nur 2 Nachrichtenworte gesendet, würde man den Fehler bemerken, aber könnte nicht feststellen, in welchem der beiden Nachrichtenworte der Fehler ist.

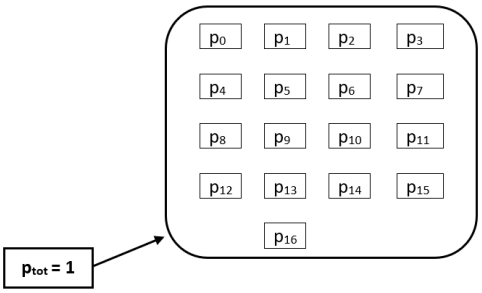
Lösung zu Motivationsaufgabe 4

Das Nachrichtenwort muss (2n+1)-mal gesendet werden. Sind die n Fehler auf n verschiedene Nachrichtenworte verteilt, bleiben trotzdem n+1 Nachrichtenworte, die korrekt und identisch sind und sich von den andern n Nachrichtenworten unterscheiden. Damit ist die Gruppe der n+1 korrekten Nachrichtenworte eindeutig identifizierbar. Verteilen sich die n Fehler auf weniger als n Nachrichtenworte, gilt der Sachverhalt analog.

Würden nur 2n Nachrichtenworte gesendet, ist es möglich, dass sich die n Fehler auf n verschiedene Nachrichtenworte verteilen, und in jedem Nachrichtenwort an derselben Stelle sind. Damit hätten wir n identische falsche Nachrichtenworte, und n identische korrekte Nachrichtenworte, und n Fehler wären nicht zu korrigieren, da man nicht entscheiden kann, welche der beiden Nachrichtenwort-Gruppen die korrekte ist.

Lösung zu Aufgabe 1

Wir nennen p_0 die Wahrscheinlichkeit, dass kein Bit geflippt ist. Wir nennen p_1 die Wahrscheinlichkeit, dass genau ein Bit geflippt ist. So kann man es für alle 17 möglichen Ergebnisse machen, also wäre dann p_{16} die Wahrscheinlichkeit, dass alle 16 Bits geflippt sind.
Jedes dieser 17 möglichen Ergebnisse hat eine eigene Wahrscheinlichkeit, und die Summe $p_0 + p_1 + \ldots + p_{16}$ all dieser 17 Wahrscheinlichkeiten (nennen wir sie p_{tot}) ergibt genau 1.
Dies kann man sich vorstellen wie ein Topf der alle 17 möglichen Ergebnisse enthält und damit auf eine Füllmenge von 1 kommt.
Die Wahrscheinlichkeit, dass kein Bit des Nachrichtenwortes geflippt ist, berechnen wir wie folgt: die Wahrscheinlichkeit, dass ein einzelnes Bit flippt



(= Basisexperiment), ist $p = 0.01$. Daraus folgt, dass die Wahrscheinlichkeit, dass ein einzelnes Bit nicht flippt $1 - p = 1 - 0.01$, also 0.99 ist (ein Bit kann entweder flippen oder nicht flippen, eine andere Möglichkeit gibt es nicht). Die Wahrscheinlichkeit, dass von den 16 Bit des Nachrichtenwortes keines flippt, ist also $p_0 = (1-p)^{16} = 0.99^{16} = 0.8515$ (gerundet). Wir haben einfach das Basisexperiment 16 mal wiederholt.
Nun zur Wahrscheinlichkeit, dass von den 16 Bits des Nachrichtenwortes genau ein Bit flippt: es gibt 16 verschiedene Möglichkeiten, dass genau ein Bit des Nachrichtenwortes flippt: das erste Bit, das 2. Bit, ..., das 16. Bit. Die Wahrscheinlichkeit, dass ein einzelnes Bit flippt, ist nach wie vor $p = 0.01$. Die Wahrscheinlichkeit, dass alle 15 anderen Bits nicht flippen, ist $(1-p)^{15} = 0.99^{15}$ Die Gesamtwahrscheinlichkeit, dass genau ein Bit flippt ist also $p_1 = 16 * p * (1-p)^{15} = 16 * 0.01 * 0.99^{15} = 0.1376$ (gerundet).
Nun können wir daraus die Wahrscheinlichkeit berechnen, dass mindestens 2 (oder mehr) Bits flippen: $1 - 0.8515 - 0.1376 = 0.0109 = 1.09\%$: wir haben einfach aus unserem Topf, der die Wahrscheinlichkeiten für alle 17 möglichen Ergebnisse enthält, die Wahrscheinlichkeiten für p_0 und p_1 entfernt und geschaut, "wieviel Wahrscheinlichkeit noch im Topf übrig ist".

Diese Zahl von 1.09% mag gering erscheinen. Man muss sie allerdings in Relation zu heute üblicherweise übermittelten Datenmengen setzen: Von 1'000'000 Nachrichtenworten (entspricht 16 Mbit = 16 Millionen Bits) enthalten im Durchschnitt also 10'900 Nachrichtenworte (=1.09%) mindestens 2 Fehler. Gemeinhin sagt man, dass für Streaming in 4K-Qualität eine Bandbreite von 50 Mbit/s notwendig ist. Das bedeutet bei unseren Annahmen, dass wir pro Sekunde $50/16 * 10'900 = 34'062$ Nachrichtenworte mit mindestens 2 Fehlern empfangen! Das Beispiel illustriert sehr schön, warum man sich Kodierungen überlegt hat, die mehr als nur 1 Fehler korrigieren können.

Führt man dieselbe Berechnung für ein Code-Wort von 25 Bit durch, erhält man folgende Resultate:
$p_0 = (1-p)^{25} = 0.99^{25} = 0.7778$ (gerundet).
$p_1 = 25 * p * (1-p)^{24} = 25 * 0.01 * 0.99^{24} = 0.1964$ (gerundet).
Wahrscheinlichkeit, dass mindestens 2 (oder mehr) Bits flippen: $1 - 0.7778 -$

$0.1964 = 0.0258 = 2.58\%$. Das ist mehr wie doppelt soviel wie für Code-Wörter der Länge 16 Bit.

Führt man dieselbe Berechnung für ein Code-Wort von 48 Bit durch, erhält man folgende Resultate:
$p_0 = (1-p)^{48} = 0.99^{48} = 0.6173$ (gerundet).
$p_1 = 48 * p * (1-p)^{47} = 48 * 0.01 * 0.99^{47} = 0.2993$ (gerundet).
Wahrscheinlichkeit, dass mindestens 2 (oder mehr) Bits flippen: $1 - 0.6173 - 0.2993 = 0.0834 = 8.34\%$. Die Wahrscheinlichkeit für mindestens 2 Fehler ist also nochmals deutlich erhöht.

Speziell Interessierte können sich dazu auch im Excel-Kartentrick-Tool das Tabellenblatt zu Aufgabe 1 anschauen. Dort sind die entsprechenden Berechnungen dargestellt und können auch für andere Fehlerwahrscheinlichkeiten als 0.01 untersucht werden.

Wir beobachten, dass mit zunehmender Code-Wort-Länge die Wahrscheinlichkeit, dass man mindestens 2 falsche Bits hat, deutlich ansteigt! Man muss also versuchen, eine Fehlerkorrektur mit möglichst wenigen Korrekturbits zu realisieren.

Lösung zu Aufgabe 2

Der Kartentrick schafft es nie, zwei Fehler zu korrigieren. Die 2 Fehler können entweder auf unterschiedlichen Zeilen und Spalten auftreten, oder sie sind auf derselben Zeile oder auf derselben Spalte. Einer dieser 3 Fälle tritt immer ein.
Abbildung 4 zeigt für alle diese 3 Fälle jeweils 2 Fehler in zwei Code-Wörtern, welche sich nicht unterscheiden lassen.

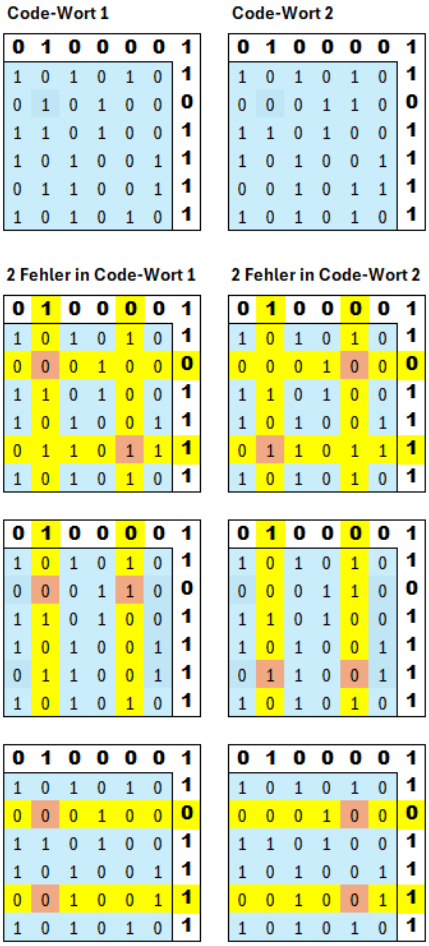


Abbildung 16: 2 Fehler, nicht korrigierbar durch Kartentrick

Lösung zu Aufgabe 4

Beispiel 1

Das funktioniert.
Begründung: Die neuen Code-Wörter haben die Form $c_{neu} = c_{original}c_{kopie}$
Wir machen eine Fallunterscheidung, wo die beiden Fehler auftreten können.
Es gibt 2 Fälle:

1. je ein Fehler in $c_{original}$ und in c_{kopie}
2. beide Fehler innerhalb nur eines Code-Wort-Bestandteils, d.h. nur in

$c_{original}$ oder nur in c_{kopie}

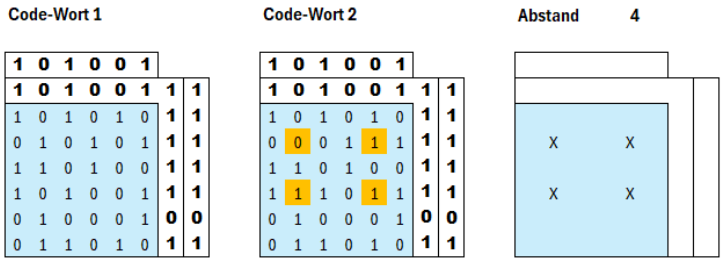
Fall 1 Wenn das Code-Wort in seine Bestandteile aufgesplittet wird und diese individuell angeschaut werden, wird in beiden Teilen erkannt werden, dass diese fehlerhaft sind. Da die Kartentrick-Kodierung bereits 1-fehlerkorrigierend ist können die Fehler können die beiden Code-Wort-Bestandteile sich auch individuell korrigieren.

Fall 2 Da die Kartentrick-Kodierung Abstand 4 hat, kann sie bis zu 3 Fehler erkennen. Wir können also feststellen, dass in einem Bestandteil die beiden Fehler liegen und der andere fehlerfrei ist. Entsprechend wird einfach der fehlerfreie Bestandteil zur Dekodierung verwendet.

Diese Kodierung ist sogar 3-fehlerkorrigierend. Siehst du warum?

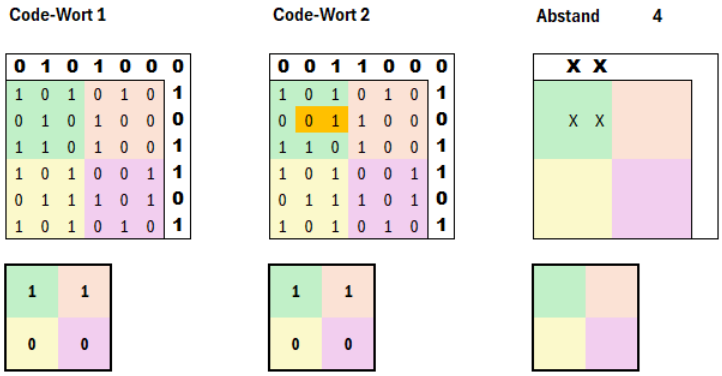
Beispiel 2

Das funktioniert nicht.
Begründung: Das Problem ist, dass die zusätzlichen Kontrollbits keine weiteren Informationen zur Fehlerkorrektur liefern. Wir können also einen beliebigen Fall von 2 Fehlern nehmen, welcher der Kartentrick nicht korrigieren kann und auch diese Kodierung wird ihn nicht korrigieren können. Wenn z.B. 2 horizontal nebeneinander liegende Nachrichtenbits geflippt werden, schlagen immer die gleichen 2 Kontrollbits der beiden Spalten an, egal, in welcher Zeile die Nachrichtenbits sich befinden. Daher können wir den Fehler keiner Zeile zuordnen. Genauso könnte man argumentieren, dass 2 solche Code-Wörter weiterhin den Hamming-Abstand 4 haben, d.h. die Kodierung somit maximal 1-fehlerkorrigierend sein kann.



Beispiel 3

Das funktioniert nicht.
Begründung: Diese Unterrechtecke helfen, falls die beiden Fehler in unterschiedlichen Unter-Rechtecken liegen. Wenn jedoch innerhalb eines Unter-Rechteckes z.B. zwei horizontal nebeneinander liegende Bits geflippt werden, wird das vom Kontrollbit des Unter-Rechtecks und der Zeile nicht "entdeckt" und es schlagen nur die Kontrollbits der beiden Spalten an, womit wir die Fehler wieder keiner Zeile zuordnen können.
Wieder kann man sich auch überlegen, dass der Abstand höchstens 4 ist.



Beispiel 4

Das funktioniert.
Begründung: Wir begründen hier, dass die Kodierung mit Diagonalbits einen Abstand von mindestens 5 aufweist, woraus folgt, dass die Kodierung 2-fehlerkorrigierend ist. Wir brauchen somit einfach zu zeigen, dass 1, 2, 3, oder 4 Unterschiede in den Nachrichtenbits zu Code-Wörtern mit Abstand mindestens 5 kodiert werden. Bei 5 oder mehr Unterschieden in der Nachricht ist bei dieser Kodierung, welche die Nachricht unverändert beinhaltet ein Abstand von 5 bereits trivialerweise gegeben. (Anm.: Wir reden hier bewusst von Unterschieden und nicht von Fehlern. Hier geht es um eine Eigenschaft (den Abstand) von gültigen Code-Wörtern der Kodierung, nicht um zufällig auftretende Fehler. Reine Unterschiede in den Kontrollbits, die bei fehlerhafter Übertragung passieren können, müssen hier somit nicht betrachtet werden, da diese nur in ungültigen Code-Wörtern vorhanden sind.)

Egal, wie wir 1, 2 oder 3 geänderte Nachrichtenbits anordnen, es befindet sich immer eines alleine in der linkesten Spalte oder in der untersten Zeile mit geänderten Bits. Das Kontrollbit dieses Nachrichtenbits muss entsprechend ebenfalls geflippt werden und auf einer eigenen Diagonale liegen, da links davon bzw. darunter kein anderes geflipptes Bit mehr liegen kann. Dadurch

hat es in all diesen Fällen mindestens ein zusätzlich geflipptes Diagonalbit. Bei 4 geänderten Nachrichtenbits muss nur der Fall angesehen werden, wo sich keine Kontrollbits ändern, da wir ansonsten bereits einen Abstand von mindestens 5 haben. Ändern sich keine Kontrollbits, müssen die 4 geflippten Bits in einem Viereck angeordnet sein. Dabei kann aber ebenfalls argumentiert werden, dass die Bits links oben und rechts unten in diesem Viereck in eigenen Diagonalen liegen müssen, womit wieder ein Abstand von mindestens 5 resultiert.
Damit ist gezeigt, dass die Erweiterung um Linksdiagonalen zu einer Kodierung mit Abstand mindestens 5 führt und somit 2-fehlerkorrigierend ist. Wir verweisen hier zusätzlich noch auf das Excel-Kartentrick-Tool, welches einen spielerischen Umgang und eine interaktive Verifikation dieser Kodierung erlaubt.

Vergleich von Beispiel 1 und Beispiel 4
Die Länge der Code-Wörter berechnen sich für Nachrichten der Länge n für die beiden Kodierungen wie folgt:

B1: $|c_{B1}| = 2 * |c_{Kartentrick}| = 2(n + 2\sqrt{n} + 1))) = 2n + 4\sqrt{n} + 2$

B4: $|c_{B4}| = |c_{Kartentrick}| + \#_{\text{Diagonal-Paritäten}} = (n+2\sqrt{n}+1)+(2\sqrt{n}+1) = n + 4\sqrt{n} + 2$

In folgender Tabelle werden diese Formeln für einige konkrete n ausgewertet

n	**B1** $(2n + 4\sqrt{n} + 2)$	**B4** $(n + 4\sqrt{n} + 2)$	**Differenz** $(B1 - B4)$	**Rel. Differenz** $(B1/B4)$
1	8	7	1	1.142857
4	18	14	4	1.285714
9	32	23	9	1.391304
25	72	47	25	1.531915
100	242	142	100	1.704225
10000	20402	10402	10000	1.961354
1000000	2004002	1004002	1000000	1.996014

Tabelle 1: Werte und Differenzen für Code-Wort-Längen resultierend aus Nachrichten-Länge n

Die Tabelle 1 veranschaulicht, wie der Unterschied in der Länge der Code-Wörter aus *Beispiel 1* verglichen mit *Beispiel 4* für längere Nachrichten immer grösser wird. Für grosse n kann man die Hälfte an Daten einsparen, wenn man auf die bessere Kodierung aus *Beispiel 4* setzt. Die Wahrscheinlichkeit für Fehler, insbesondere mehr Fehler, als korrigiert werden können, ist zudem bei langen Code-Wörtern wesentlich grösser. Dies kann Kodierungen mit langen Code-Wörtern komplett unbrauchbar machen, da sich in

jedes übertragene Code-Wort mehr Fehler als korrigiert werden können, einschleichen können. Es lohnt sich also definitiv, gute Kodierungen zu wählen.

Lösung zu Aufgabe 5

Die 2 Fehler können entweder in unterschiedlichen Zeilen und Spalten, in derselben Zeile oder in derselben Spalte auftreten.
Im Fall von Fehlern in unterschiedlichen Zeilen und Spalten ist noch zu unterscheiden, ob die Fehler auf derselben Diagonalen liegen oder nicht.
Wir betrachten zuerst den Fall mit 2 Fehlern in unterschiedlichen Zeilen und Spalten. Dies präsentiert sich etwa wie unten dargestellt:

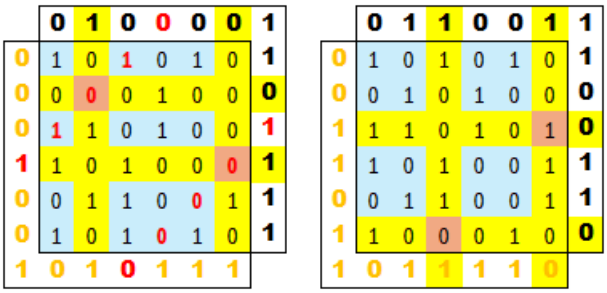


Abbildung 17: 2 Fehler in unterschiedlichen Zeilen und Spalten

Im Fall, in welchem die Fehler nicht auf einer gemeinsamen Diagonalen liegen, können die Fehler aufgrund der Kontrollbits der Diagonalen eindeutig zugewiesen werden. Sind die beiden Fehler auf einer gemeinsamen Diagonalen, schlägt kein Diagonal-Kontrollbit an, womit die Fehler ebenfalls eindeutig identifiziert werden können, nämlich auf der Diagonale liegend, die durch 2 Fehler-Kreuze hindurchgeht.
Untenstehend die Darstellung von 2 Fehlern in derselben Zeile oder Spalte:

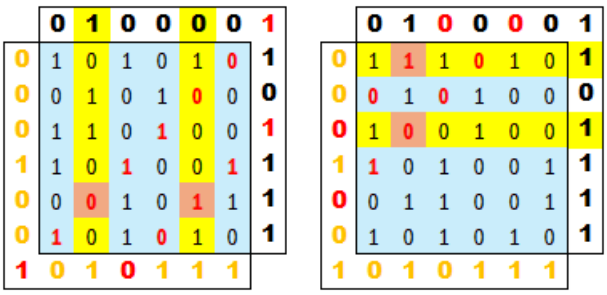


Abbildung 18: 2 Fehler in derselben Zeile oder Spalte

Liegen beide Fehler in derselben Zeile, schlagen keine Kontrollbits, welche die Zeilen überwachen an. Die Fehldiagonalen schneiden die Fehlspalten in derselben Zeile. Diese Schnittpunkte stellen die Fehler dar.
Die Überlegung für den Fall, in welchem die beiden Fehler in derselben Spalte liegen, ist analog.

Lösung zu Aufgabe 6

Die folgende Abbildung zeigt zwei unterschiedliche Anordnungen von 3 Fehlern, welche zu denselben Fehlermeldungen führen (Fehlzeilen, -spalten und -diagonalen), was gleichbedeutend damit ist, dass die Fehler durch die Erweiterung des Kartentricks mit den Linksdiagonalen nicht eindeutig identifiziert werden können.

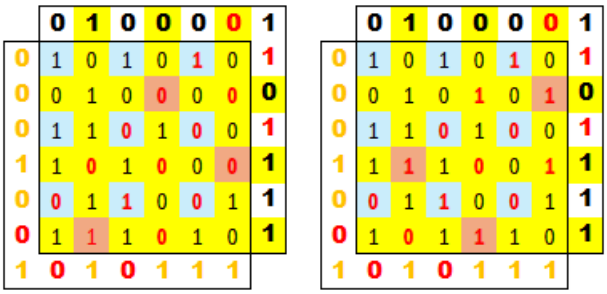
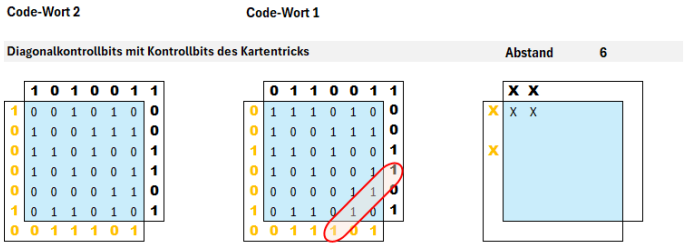


Abbildung 19: 3 Fehler mit denselben Fehlermeldungen

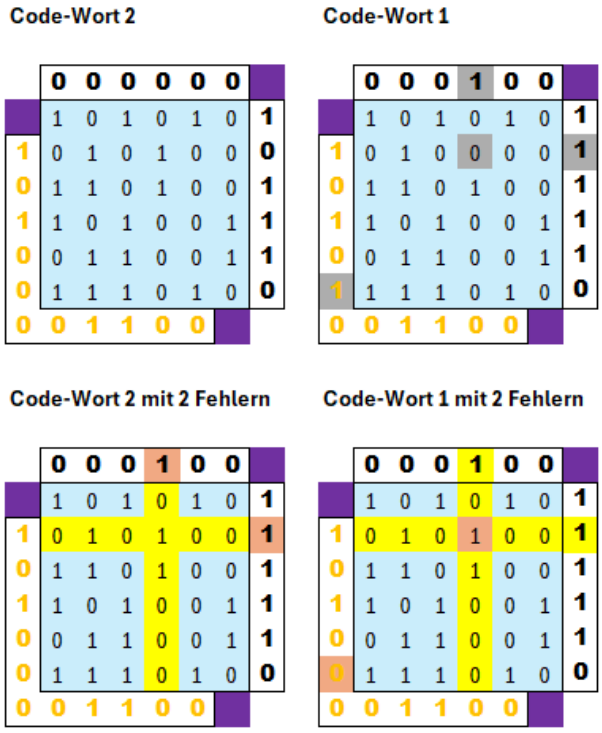
Lösung zu Aufgabe 7

Die folgenden 2 Code-Wörter weisen voneinander lediglich Abstand 6 auf.



Lösung zu Aufgabe 8

Die folgenden 2 Code-Wörter weisen Abstand 4 auf. Die unterschiedlichen Nachrichtenbits sind grau hinterlegt. Darunter sind jeweils Konstellationen dieser Code-Wörter mit 2 Fehlern dargestellt, welche zu identischen Code-Wörtern führen. Die Fehler können also nicht identifiziert werden.



Lösung zu Aufgabe 9

Die Fehler sind in untenstehenden Code-Wörtern rot hinterlegt.