Unterrichtseinheit zum Hadamard-Code

Daniel Oehry, Januar 2022

Inhalt und Lernziele

Beim Hadamard-Code handelt es sich um eine Kodierung von Nachrichten, welche es erlaubt – abhängig von der Länge der Code-Wörter – beliebig viele Fehler zu korrigieren. Unter anderem wurde ein Hadamard-Code von der NASA in der 70er Jahren zur Übermittlung der Bilddaten vom Mars zur Erde verwendet.

Die hier vorgestellte Unterrichtseinheit hat folgende Ziele:

- Kurze Repetition des Begriffs „Abstand einer Kodierung“ und des Zusammenhangs zur Fehlererkennung und -korrektur
- Vorstellung des Verfahrens zur Erzeugung des Hadamard-Codes
- Einübung des Verfahrens anhand eines Beispiels und Untersuchung der Eigenschaften der so gewonnenen Kodierung
- Finden einer allgemeinen Begründung für die Korrektheit der gefundenen Eigenschaften
- Präsentation eines kurzen geschichtlichen Abrisses über die Verwendung des Hadamard-Codes bei der Mariner 9 Marsmission der NASA
- Vergleich des Hadamard-Codes mit der mehrfachen Wiederholung anhand verschiedener Kenngrössen
- Interpretation des Hadamard-Codes als Kodierung mit Nachrichten- und Kontrollbits und damit verbunden eine allgemeinere Betrachtung fehlerkorrigierender Kodierungen

Vorwissen

Für die Bearbeitung der Aufgaben ist es notwendig zu wissen, was mit der „Kodierung einer endlichen Menge von Nachrichten“ gemeint ist. Insbesondere folgende Punkte müssen bereits bekannt sein:

- Zusammenhang zwischen Bitlänge einer Nachricht und der Anzahl Nachrichten, welche dann kodiert werden können
- Abstand von zwei Code-Wörtern und Abstand einer Kodierung
- Zusammenhang zwischen dem Abstand einer Kodierung und der Fehlererkennung
- Zusammenhang zwischen dem Abstand einer Kodierung und der Fehlerkorrektur

Wenn Kontrollbits, Fehlermeldungstabellen und Hamming-Code bereits bekannt sind, schafft die letzte Aufgabe eine Verbindung dazu. Sind diese Begriffe nicht bekannt, kann die Aufgabe als Einstieg dienen.

Hadamard-Code

Aufgabe 1

Wir wollen ein Bild übermitteln und verwenden für die einzelnen Pixel acht verschiedene Graustufen. Dazu reichen drei Bits, weil es $2^3 = 8$ unterschiedliche Bitfolgen der Länge 3 gibt.

Es sollen nun zwei verschiedene Kodierungen bezüglich ihrer Fehlererkennungsmöglichkeiten verglichen werden: Kodierung 1 wiederholt einfach die Darstellung zweimal, Kodierung 2 sieht auf den ersten Blick etwas seltsam und willkürlich aus. Wir werden erst später erläutern, wie sie entwickelt wurde und untersuchen vorerst nur ihre Eigenschaften.

Bestimmen Sie den Abstand der beiden Kodierungen. Was bedeutet das für die Fehlererkennungs- und Fehlerkorrekturmöglichkeiten der Kodierungen?

Graustufe	Darstellung	Kodierung 1	Kodierung 2
	000	000000	0000
	001	001001	0101
	010	010010	0011
	011	011011	0110
	100	100100	1111
	101	101101	1010
	110	110110	1100
	111	111111	1001

Ein Verfahren zur Erzeugung von Kodierungen

In diesem Abschnitt soll nun erklärt werden, wie Kodierung 2 entstanden ist. Wir beginnen damit, dass wir uns anschauen, welche Wörter mit 2 Bits gebildet werden können. Es sind dies die vier Wörter 00, 01, 10 und 11. Sortieren wir diese etwas anders und schreiben sie untereinander, so entsteht folgendes 4×2-Feld.

A_2	0 0
	0 1
$\overline{A_2}$	1 1
	1 0

In der oberen Hälfte (rot markiert) erhalten wir ein 2×2-Feld, welches wir mit A_2 bezeichnen. In der unteren Hälfte (blau markiert) befindet sich das Komplement von A_2, welches wir mit $\overline{A_2}$ bezeichnen. Den Index 2 verwenden wir deshalb, weil es sich um eine Kodierung für Wörter der Länge 2 handelt.

Nun werden wir mit Hilfe von A_2 und $\overline{A_2}$ eine neue Kodierung erzeugen. Wir platzieren dazu die Felder nach dem unten links dargestellten Muster und erhalten so das 8×4-Feld rechts.

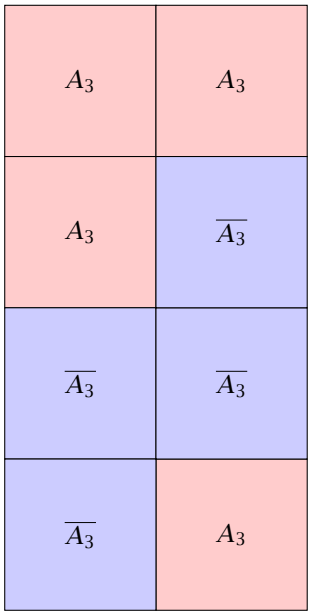
A_2	A_2
A_2	$\overline{A_2}$
$\overline{A_2}$	$\overline{A_2}$
$\overline{A_2}$	A_2

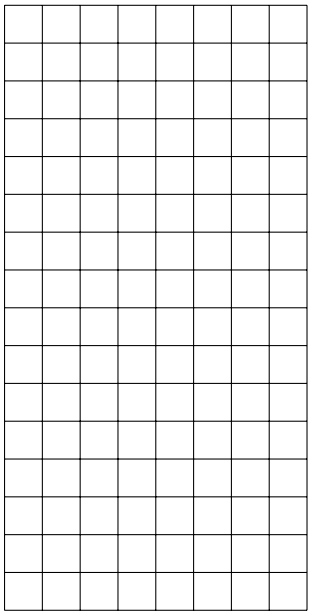
0	0	0	0
0	1	0	1
0	0	1	1
0	1	1	0
1	1	1	1
1	0	1	0
1	1	0	0
1	0	0	1

Es handelt sich exakt um unsere Kodierung 2 aus Aufgabe 1! Wir können damit acht verschiedene Graustufen (Nachrichten) kodieren, es handelt sich also um eine Kodierung für Wörter der Länge 3.

Nun wenden wir das gleiche Verfahren nochmals an, d. h., wir bezeichnen den oberen Teil mit A_3. Nach unserer Konstruktion ist der untere Teil das Komplement $\overline{A_3}$. Diese Blöcke werden nun nach dem identischen Muster nochmals platziert und wir erhalten eine weitere Kodierung.

A_3	0	0	0	0
	0	1	0	1
	0	0	1	1
	0	1	1	0
$\overline{A_3}$	1	1	1	1
	1	0	1	0
	1	1	0	0
	1	0	0	1





Aufgabe 2

a) Füllen Sie die Felder oben rechts aus. Wie viele verschiedene Graustufen (Nachrichten) können damit kodiert werden? Welche Bitlänge haben die Nachrichten?
b) Bestimmen Sie den Abstand der so gewonnenen Kodierung.
c) Geben Sie Beispiele von Bitfolgen an, welche Fehler enthalten, jedoch trotzdem eindeutig zugeordnet werden können. Gibt es auch Bitfolgen, bei welchen zwar erkennbar ist, dass sie Fehler enthalten, die jedoch nicht eindeutig zugeordnet werden können?
d) Fassen Sie allgemein zusammen, wie viele Fehler mit dieser Kodierung erkannt und wie viele sogar korrigiert werden können.

Aufgabe 3

Das vorgestellte Verfahren kann weitergeführt werden und wir erhalten immer neue Kodierungen.

a) Wie verändern sich folgende Kenngrössen mit jedem Schritt? Begründen Sie jeweils auch Ihre Antworten.
   - Anzahl der Graustufen (Nachrichten), welche kodiert werden können
   - Länge der Code-Wörter
   - Abstand der Kodierung
b) Sie möchten 64 verschiedene Graustufen kodieren. Wie oft müssen Sie das Verfahren anwenden und welche Eigenschaften (Länge der Code-Wörter, Abstand) hat die so gewonnene Kodierung?

Etwas Geschichtliches

Bei der vorgestellten Kodierung handelt es sich um einen *Hadamard-Code*. Er wurde nach dem französischen Mathematiker JACQUES HADAMARD (1865–1963) benannt.

Der Hadamard-Code aus Aufgabe 3b wurde 1971 in der Mariner 9 Mission der NASA verwendet. Mariner 9 startete im Mai 1971 und erreichte im November desselben Jahres den Mars. Die Sonde wurde dann auf eine Umlaufbahn um den Mars gebracht und sammelte während fast eines Jahres Daten. Insgesamt wurden 54 Milliarden Bits zur Erde gesendet, darunter auch 7239 Bilder der Marsoberfläche. Dank des verwendeten Hadamard-Codes konnten bei der Bildübertragung Fehler korrigiert werden.

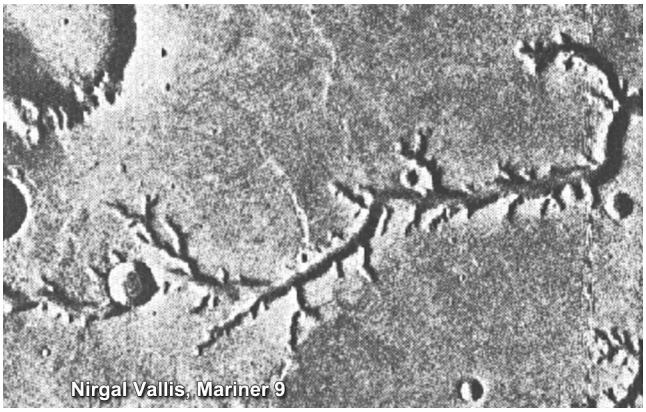


Foto: NASA/JPL-Caltech

Aufgabe 4

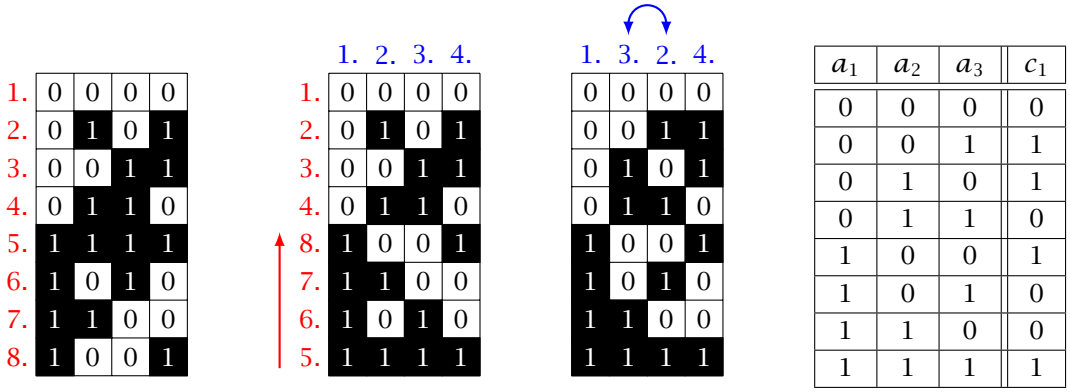
In Aufgabe 3b haben Sie die Eigenschaften der von Mariner 9 verwendeten Kodierung untersucht. Alternativ hätte sich die NASA auch für eine simple mehrfache Wiederholung der Nachricht entscheiden können.

a) Wie viele Wiederholungen hätten verwendet werden können unter der Bedingung, dass die Länge der resultierenden Code-Wörter nicht länger sein darf als beim verwendeten Hadamard-Code?
b) Wie viele Fehler hätten so erkannt und wie viele korrigiert werden können?
c) Vergleichen Sie die beiden Kodierungen und überlegen Sie sich, was wohl die Gründe dafür waren, dass bei Mariner 9 ein Hadamard-Code verwendet wurde.

hadamardcode.tex 16.1.2022 oda

Ein etwas anderer Blick auf den Hadamard-Code

Wir gehen nochmals zurück zum Hadamard-Code für Wörter der Länge 3 und stellen ihn etwas anders dar. In einem ersten Schritt verändern wir die Reihenfolge der Zeilen 5 bis 8 so, dass diese von unten nach oben geschrieben wird. Danach vertauschen wir die Spalten 2 und 3. In dieser neuen Darstellung erkennen wir in den ersten drei Spalten die kodierten Wörter (000, 001, 010, …, 111). Die vierte Spalte enthält ein Kontrollbit. Dieses ist einfach ein Paritätsbit, d. h., es schaut, ob die Anzahl Einsen in jeder Zeile gerade oder ungerade ist. Wir schreiben $c_1 = a_1 \oplus a_2 \oplus a_3$ für die Addition modulo 2.



a_1	a_2	a_3	c_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Aufgabe 5

Eine Umsortierung der Zeilen und Spalten in dieser Art lässt sich auch beim Hadamard-Code für Wörter der Länge 4 durchführen.

a) Wie viele Kontrollbits werden Sie so erhalten?
b) Beschreiben Sie, wie Sie umsortiert haben. Können Sie angeben, wie die so erhaltenen Kontrollbits berechnet werden?
c) Wir haben gesehen, dass mit diesem Hadamard-Code ein Fehler korrigiert werden kann. Könnten Sie ein Kontrollbit (oder sogar mehrere) entfernen, so dass immer noch ein Fehler korrigiert werden kann?

Quellen

[1] Computerphile. „64 Shades of Martian Grey". *YouTube*, https://youtu.be/NRv3HMEyuDE.

[2] Dewdney, Alexander K. *Der Turing Omnibus*. Springer, 1995. S. 82–86.

[3] „Hadamard-Code". *Wikipedia*, https://de.wikipedia.org/wiki/Hadamard-Code.

[4] „Mariner 9". *NASA*, https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1971-051A.

Lösungen

1. Ziel dieser Aufgabe ist es, die zur Bearbeitung der weiteren Aufgaben notwendigen Begriffe und Konzepte zu wiederholen. Für den Abstand einer Kodierung müssen die gegenseitigen Abstände aller Paare von Code-Wörtern untersucht werden. Das Minimum entspricht dem Abstand der Kodierung. Für Kodierung 1 ist das rasch beantwortet. Durch die zweifache Wiederholung erhalten wir als Abstand der Kodierung zwei.

   Der Abstand der Kodierung 2 ist ebenfalls zwei. Dies lässt sich überprüfen, indem alle möglichen Paare angeschaut werden. Wer eine etwas allgemeinere Erklärung sucht, könnte sich auch überlegen, dass ausser bei 0000 und 1111 überall genau zweimal die 1 vorkommt. D. h., der Abstand zwischen 0000 (bzw. 1111) und einem dieser Wörter ist auf jeden Fall zwei. Der Abstand zwischen 0000 und 1111 ist natürlich vier. Bleibt zu untersuchen, wie gross der Abstand zwischen zwei beliebigen Wörtern mit zwei Einsen ist. Da alle Wörter unterschiedlich sind, bleiben nur zwei Möglichkeiten: Entweder sie haben eine 1 gemeinsam, dann unterscheiden sie sich aber in zwei Stellen, oder sie haben keine 1 gemeinsam und unterscheiden sich in allen vier Stellen. Resümierend formuliert: Es gibt keine zwei Wörter, welche einen Abstand kleiner als zwei haben. Der Abstand der Kodierung ist zwei.

   Wenn der Abstand einer Kodierung zwei beträgt, bedeutet das, dass Fehler in zwei Stellen nicht mehr erkannt werden können, weil dann z. B. aus der 0000 die 1001 wurde. Ein weisses Pixel wurde also wegen dieser zwei Fehler als schwarzes zugeordnet. D. h., wir können maximal einen Fehler erkennen. Korrigiert werden kann dieser allerdings nicht: 1000 könnte z. B. einen Fehler in der 1. Stelle haben und damit eigentlich 0000 sein, es könnte aber auch ein Fehler in der 4. Stelle und damit eigentlich 1001 sein.

   Allgemein formuliert ist es so, dass eine Kodierung den Abstand $k + 1$ haben muss, damit k Fehler erkannt werden können. Für die Korrektur der Fehler muss der Abstand grösser sein. Sollen k Fehler korrigiert werden können, muss der Abstand der Kodierung mindestens $2k + 1$ sein.

2. a) Das ausgefüllte Raster sieht wie unten abgebildet aus. Zur besseren Übersicht wurden die Felder mit den Einsen schwarz eingefärbt. Es hat 16 Zeilen, womit sich ebenso viele Graustufen kodieren lassen. Weil $16 = 2^4$ ist, werden für die Kodierung von 16 Graustufen 4 Bits benötigt.

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

Graustufe	Darstellung	Kodierung
	0000	00000000
	0001	01010101
	0010	00110011
	0011	01100110
	0100	00001111
	0101	01011010
	0110	00111100
	0111	01101001
	1000	11111111
	1001	10101010
	1010	11001100
	1011	10011001
	1100	11110000
	1101	10100101
	1110	11000011
	1111	10010110

b) Die Kodierung hat den Abstand vier.

c) Bei der Zuordnung von Bitfolgen entscheiden wir uns für diejenige Zuordnung, bei welcher der Abstand am geringsten ist. Z. B. wird das Wort 00010000 der Bitfolge 00000000 zugeordnet. Bei zwei Fehlern bekommen wir ein Problem. 00101000 könnte sowohl 00000000 oder 10101010 zugeordnet werden.

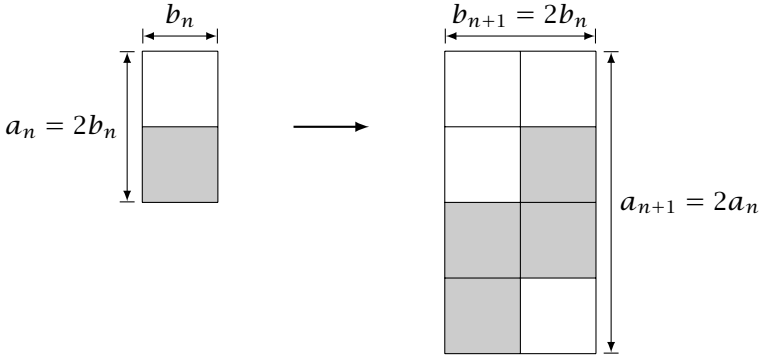
0 0 1 0 1 0 0 0	0 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0	1 0 1 0 1 0 1 0
× ×	× ×

Noch schlimmer wird es bei drei Fehlern. Es wird zwar erkannt, dass etwas nicht stimmt, die Bitfolge wird jedoch falsch zugeordnet. 10010100 wird 10010110 zugeordnet, könnte bei drei Fehlern aber ursprünglich auch 00000000 gewesen sein.

1 0 0 1 0 1 0 0	1 0 0 1 0 1 0 0
1 0 0 1 0 1 1 0	0 0 0 0 0 0 0 0
×	× × ×

d) Da der Abstand der Kodierung vier ist, können maximal drei Fehler erkannt werden. Liegen zwei Fehler vor, so gibt es bei der Zuordnung zwei mögliche Kandidaten, welche den gleichen Abstand haben. Erst bei einem Fehler ist eine eindeutige Zuordnung möglich. Somit kann mit dieser Kodierung ein Fehler korrigiert werden.

3. a) Aufgrund der Konstruktion ist es einfach eine Antwort auf die ersten beiden Punkte zu geben. Das Feld verdoppelt sich in jedem Schritt sowohl in der Länge als auch in der Breite. Die Länge der Code-Wörter im $n$ten Schritt beträgt deshalb $b_n = 2^{n-1}$ und es können $b_n = 2^n$ Graustufen kodiert werden. Damit ist auch klar, dass die Nummer n der Bitlänge der Nachricht (Graustufe) entspricht.

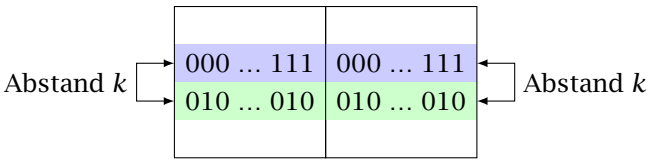


Etwas schwieriger gestaltet sich die Beantwortung des dritten Punkts nach dem Abstand der Kodierung. Aufgrund der Ergebnisse von Aufgabe 1 und 2 liegt es nahe anzunehmen, dass sich auch der Abstand in jedem Schritt verdoppelt. Wir wollen dies aber auch begründen. Dazu resümieren wir zunächst, was wir bisher entdeckt haben:

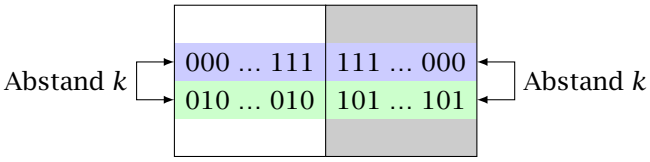
Bitlänge n der Nachricht	2	3	4
Länge der Code-Wörter	2	4	8
Abstand der Kodierung	1	2	4

Für $n \in \{2, 3, 4\}$ haben wir also gesehen, dass unsere Vermutung stimmt. Ausserdem können wir auch feststellen, dass der Abstand der Kodierung der halben Länge der Code-Wörter entspricht. Wir wollen zeigen, dass diese Eigenschaft bei jedem weiteren Schritt erhalten bleibt. Ausgangslage bildet dabei eine Kodierung mit Code-Wörtern der Länge $2k$, welche den Abstand k hat, aus welcher mit Hilfe der vorgestellten Konstruktion eine Kodierung mit Code-Wörtern der Länge $4k$ gebildet wird.

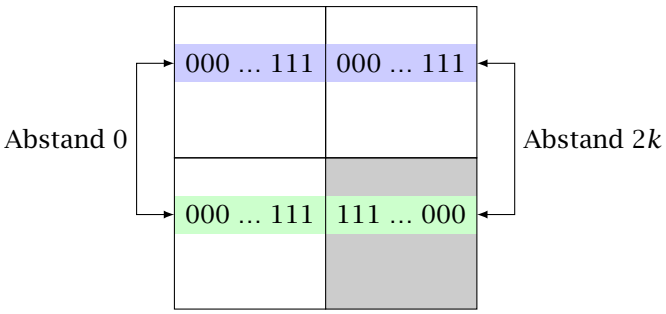
Wir müssen den Abstand von zwei beliebig ausgewählten Zeilen betrachten. Liegen beide Zeilen im obersten Viertel, liegt eine zweifache Wiederholung der Bitfolge vor. Damit verdoppelt sich der Abstand auf $2k$.



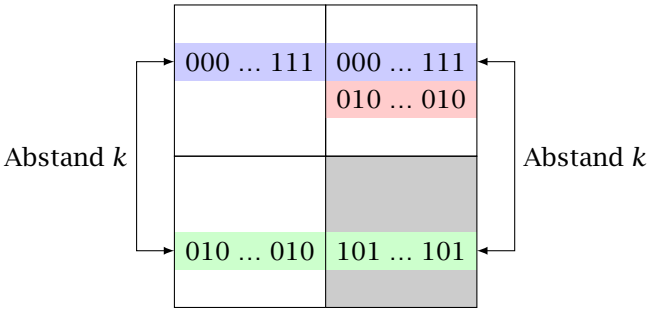
Liegen beide Zeilen im zweiten Viertel, verhält sich die Sache ähnlich. Es liegt nun zwar keine zweifache Wiederholung der Bitfolge mehr vor, weil sich im rechten Teil das Komplement befindet, aber auch dort haben die Zeilen den Abstand k, weshalb der Abstand insgesamt ebenfalls $2k$ ist.



Interessant wird es, wenn eine Zeile im ersten und die andere im zweiten Viertel liegt. Es sind zwei Fälle zu unterscheiden. Wenn der linke Teil identisch ist, hat dieser einen Abstand von 0. Der rechte Teil enthält dann aber das Komplement, unterscheidet sich also in allen Stellen, weshalb der Abstand insgesamt wiederum $2k$ beträgt.

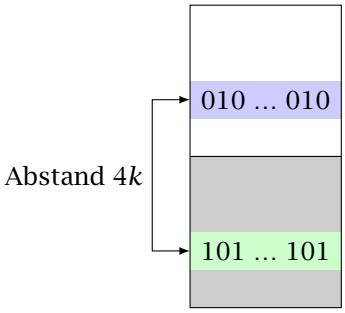


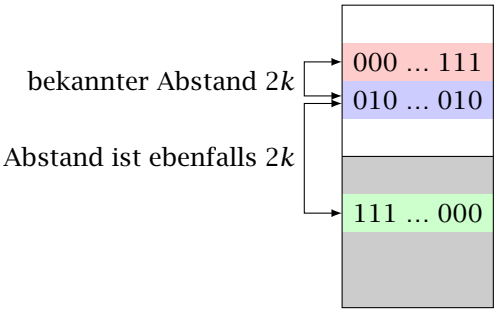
Ist der linke Teil nicht identisch hat dieser einen Abstand von k. In diesem Fall hat auch der rechte Teil den Abstand k. Um dies einzusehen betrachten wir in der rechten Hälfte den rot eingefärbten Teil. Dieser hat den Abstand k zur blau eingefärbten Zeile, d. h., die rote Zeile entsteht aus der blauen durch Flippen von k Bits. Weil es sich bei der grün eingefärbten Zeile um das Komplement der roten handelt, entsteht diese aus der blauen durch Flippen der anderen k Bits. Diese hat also wie behauptet den Abstand k. Insgesamt haben die betrachteten Zeilen den Abstand $2k$.



Was jetzt noch fehlt, ist eine Begründung dafür, dass der Abstand auch $2k$ ist, wenn im dritten und vierten Viertel noch das Komplement des ersten und zweiten Viertels hinzugefügt wird. Wir haben bisher gezeigt, dass der Abstand $2k$ ist, wenn beide Zeilen aus der oberen Hälfte

gewählt werden. Werden beide Zeilen aus der unteren Hälfte gewählt, welche das Komplement ist, funktioniert die Argumentation genau gleich, d. h., der Abstand ist ebenfalls $2k$. Wird eine Zeile aus der oberen und die andere aus der unteren Hälfte gewählt, gibt es zwei Fälle zu unterscheiden. Der einfache Fall liegt vor, wenn aus dem oberen und unteren Teil sich entsprechende Zeilen gewählt werden. Weil in diesem Fall das untere Wort das Komplement des oberen ist, unterscheiden sich die Wörter in allen Stellen. Der Abstand entspricht also der Länge der Code-Wörter und somit $4k$.



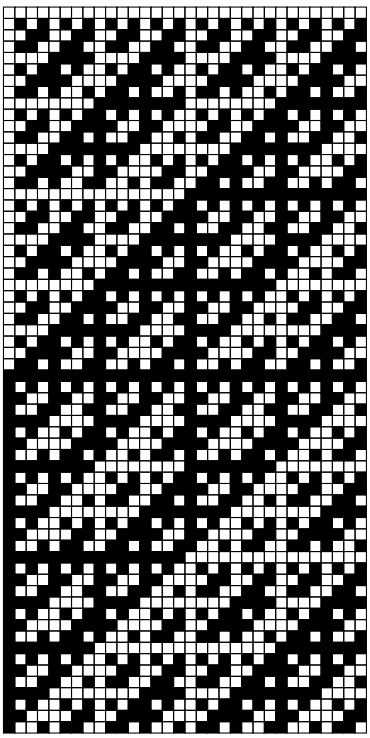


Werden aus dem oberen und unteren Teil sich nicht entsprechende Zeilen gewählt, hilft die gleiche Überlegung wie weiter oben. Wir betrachten das rot eingefärbte Komplement der unteren Zeile. Der Abstand zwischen dieser Zeile und der blauen ist $2k$, d. h., sie entsteht aus der blauen durch Flippen von $2k$ Bits. Werden die anderen $2k$ Bits geflippt, erhalten wir die grün eingefärbte Zeile. Der Abstand zwischen der blauen und grünen Zeile ist also ebenfalls $2k$.

Zusammengefasst haben wir nun gezeigt, dass durch Anwendung der vorgestellten Konstruktion aus einer Kodierung mit Abstand k eine Kodierung mit Abstand $2k$ erzeugt werden kann. Wir können nun sogar allgemeine Formeln angeben.

Bitlänge der Nachricht	n
Anzahl der Nachrichten (Graustufen)	2^n
Länge der Code-Wörter	2^{n-1}
Abstand der Kodierung	2^{n-2}

b) Nach den vorherigen Überlegungen lässt sich diese Frage nun zum Glück leicht beantworten. Wir wollen 64 Graustufen kodieren. Weil $64 = 2^6$, benötigen wir dafür 6 Bits, d. h., wir müssen das Verfahren noch zwei weitere Male durchführen, bis wir Code-Wörter der Länge $2^5 = 32$ erhalten. Die so konstruierte Kodierung hat den Abstand 16. Das zugehörige Raster ist rechts abgebildet, wobei nicht ausgefüllte Felder die Null und ausgefüllte Felder die Eins repräsentieren.



4. a) Es wurden 64 Graustufen kodiert, was in der Darstellung einer Folge von 6 Bits entspricht. Beim verwendeten Hadamard-Code haben die kodierten Wörter eine Länge von 32. Wenn wir unter dieser Länge bleiben wollen, können wir also die Bitfolge fünfmal wiederholen.

   b) Der Abstand der Kodierung mit fünffacher Wiederholung ist fünf. Damit lassen sich vier Fehler erkennen und zwei korrigieren.

   c) Ein Vergleich dieser beiden Methoden für Nachrichten der Länge drei führt zu folgender Übersicht.

Kodierung	Hadamard-Code	fünffache Wiederholung
Länge der Code-Wörter	32	30
Abstand der Kodierung	16	5
Anzahl Fehler, welche erkannt werden	15	4
Anzahl Fehler, welche korrigiert werden	7	2

   Das Verhältnis von Bitlänge der Nachricht zur Länge der verwendeten Code-Wörter heisst *Informationsrate*. Diese ist bei beiden Kodierungen ähnlich hoch. Die *Korrekturrate* gibt das Verhältnis der korrigierbaren Fehler zur Länge der Code-Wörter an. Diese ist beim Hadamard-Code wesentlich höher.

Kodierung	Hadamard-Code	fünffache Wiederholung
Informationsrate	$\frac{6}{32} = 0.188 = 18.8\%$	$\frac{6}{30} = 0.2 = 20\%$
Korrekturrate	$\frac{7}{32} = 0.219 = 21.9\%$	$\frac{2}{30} = 0.067 = 6.7\%$

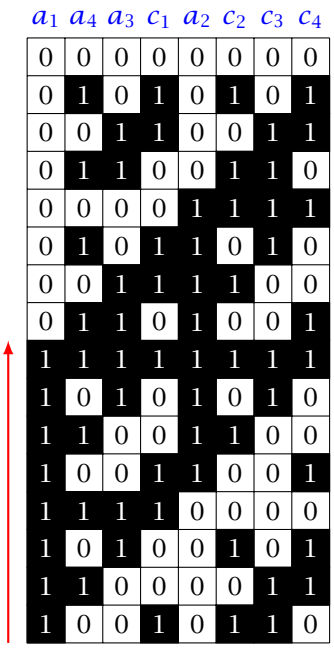
   Der Mars befindet sich von der Erde so weit entfernt, dass die Signale je nach relativer Position der beiden Planeten eine Laufzeit von zwischen 3 und 22 Minuten haben. Bei dieser Laufzeit ist es nicht möglich bei erkannten Fehlern die Daten einfach nochmals anzufordern, weshalb es wichtig war, möglichst viele Fehler korrigieren zu können. Die NASA ist davon ausgegangen, dass aufgrund von Störungen des Signals ohne Fehlerkorrektur etwa ein Viertel falsch empfangen wird. Damit erwies sich der Hadamard-Code mit seiner Korrekturrate von 22 % als gut geeignet, möglichst fehlerfreie Bilder zu empfangen.

5. a) Der Hadamard-Code für Wörter der Länge 4 hat acht Spalten. Es bleiben neben den vier Nachrichtenbits also vier Kontrollbits.

   b) Die Idee der Umsortierung lässt sich recht gut übertragen und ist auf der nächsten Seite abgebildet. Die untere Hälfte wird ebenfalls wieder von unten nach oben geschrieben. Danach lässt sich gut die 1er-Spalte finden. Es ist die zweite Spalte, bei welcher sich Einsen und Nullen abwechseln. Die 2er-Spalte findet sich in der dritten, wo sich 00 und 11 abwechseln, die 4er-Spalte in der fünften und die 8er-Spalte in der ersten Spalte. Die übrigen Spalten werden einfach der Reihe nach als Kontrollbits geschrieben.

   Bleibt die Frage, wie die Kontrollbits berechnet werden. Dazu ist etwas Kreativität und Ausprobieren notwendig. Letztendlich lässt sich feststellen, dass es klappt, wenn jedes Kontrollbit drei Nachrichtenbits kontrolliert. Probieren wir mögliche Kombinationen durch, so finden wir die Lösung:

$$c_1 = a_1 \oplus a_3 \oplus a_4$$
$$c_2 = a_1 \oplus a_2 \oplus a_4$$
$$c_3 = a_1 \oplus a_2 \oplus a_3$$
$$c_4 = a_2 \oplus a_3 \oplus a_4$$



a_1	a_4	a_3	c_1	a_2	c_2	c_3	c_4
0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
0	0	0	0	1	1	1	1
0	1	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0
1	0	0	1	1	0	0	1
1	1	1	1	0	0	0	0
1	0	1	0	0	1	0	1
1	1	0	0	0	0	1	1
1	0	0	1	0	1	1	0

a_1	a_2	a_3	a_4	c_1	c_2	c_3	c_4
0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	1
0	0	1	0	1	0	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	1	0	0
0	1	1	1	0	0	0	1
1	0	0	0	1	1	1	0
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	0	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	1	1	1	1	1

c) Bei der Beantwortung dieser Frage hilft folgende Tabelle. Sie dokumentiert, welches Kontrollbit welche Nachrichtenbits überwacht. Beispielsweise lässt sich ablesen, dass c_1 sich selbst sowie a_1, a_3 und a_4 überwacht. Für uns wichtiger sind aber die Spalten. Dort können wir ablesen, welche Prüfbits nicht stimmen, wenn ein Nachrichtenbit fehlerhaft ist. Haben wir z. B. einen Fehler in c_1, c_2 und c_3, wissen wir, dass a_1 umgeflippt wurde.

	a_1	a_2	a_3	a_4	c_1	c_2	c_3	c_4
c_1	×		×	×	×			
c_2	×	×		×		×		
c_3	×	×	×				×	
c_4		×	×	×				×

$$c_1 = a_1 \oplus a_3 \oplus a_4$$
$$c_2 = a_1 \oplus a_2 \oplus a_4$$
$$c_3 = a_1 \oplus a_2 \oplus a_3$$
$$c_4 = a_2 \oplus a_3 \oplus a_4$$

Weil alle Spalten der Tabelle paarweise unterschiedlich sind, kann die Kodierung einen Fehler korrigieren. Die Frage ist nun, ob ein Kontrollbit weggelassen werden kann. Bevor wir einfach ausprobieren, überlegen wir uns, ob das überhaupt gehen kann. Jede Spalte besteht aktuell aus vier Plätzen, wovon jeder entweder ein Kreuz haben oder leer sein kann. D. h., es gibt $2^4 = 16$ verschiedene Möglichkeiten die Kreuze zu setzen. Eine Möglichkeit müssen wir abziehen, weil eine Spalte gänzlich ohne Kreuze nicht zugelassen ist. Wir haben also $2^4 - 1 = 15$ verschiedene Spalten. Unsere Tabelle enthält jedoch nur acht Spalten.

Wir können also überlegen, ob es auch mit einem Kontrollbit weniger klappen könnte. Dann haben wir nur noch $2^3 - 1 = 8 - 1 = 7$ verschiedene Spalten, brauchen aber auch nur genauso viele. Das klappt also, wenn alle Möglichkeiten in die Tabelle aufgenommen werden. Wenn wir uns die Tabelle anschauen, stellen wir fest, dass es sogar egal ist, welches Kontrollbit wir streichen. Es wird immer klappen, dass die verbleibenden Spalten paarweise unterschiedlich sind und die Kodierung damit immer noch einen Fehler korrigiert. Beispielsweise können wir c_4 streichen und erhalten so eine Kodierung der Länge 7.

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

$$c_1 = a_1 \oplus a_3 \oplus a_4$$
$$c_2 = a_1 \oplus a_2 \oplus a_4$$
$$c_3 = a_1 \oplus a_2 \oplus a_3$$

Ein weiteres Kontrollbit kann nicht mehr weggelassen werden. Wir hätten dann nur noch $2^2 - 1 = 4 - 1 = 3$ verschiedene Spalten, bräuchten aber sechs.