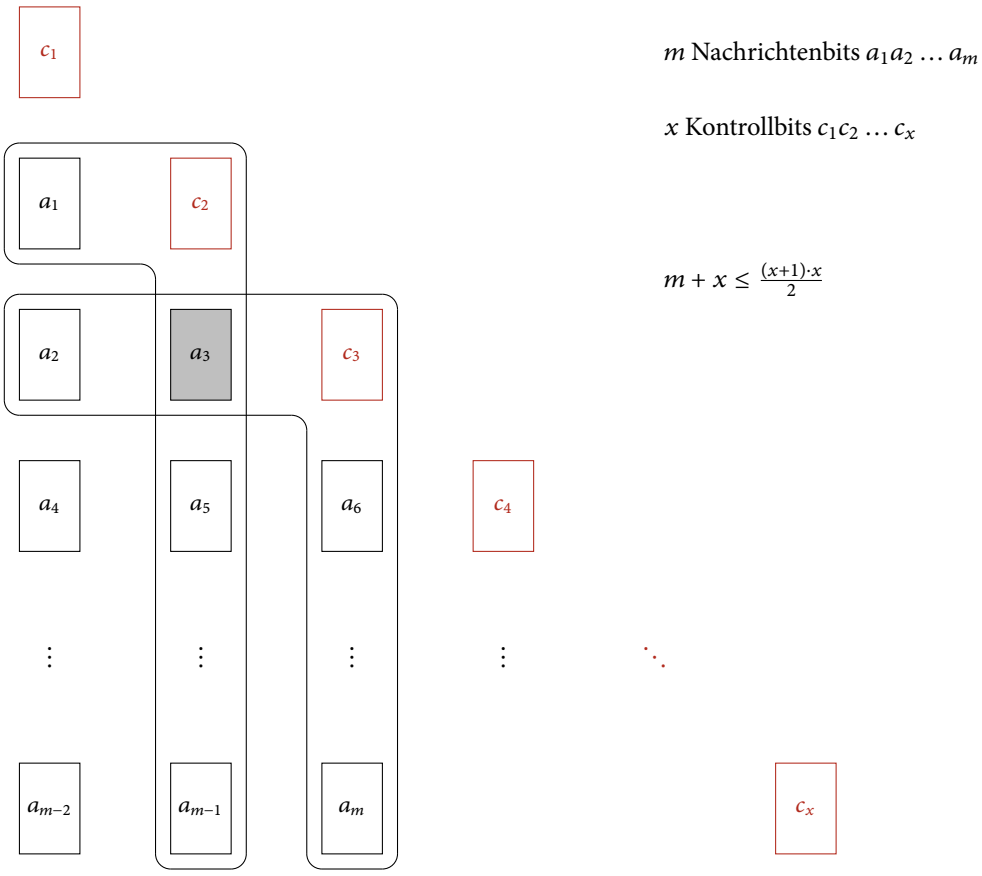
Selbstkorrigierende Kodierungen

Eine alternative Kartentrick-Kodierung

Claudio Müller (claudio.mueller@kzo.ch)

Januar 2022



m Nachrichtenbits $a_1 a_2 \dots a_m$

x Kontrollbits $c_1 c_2 \dots c_x$

$m + x \leq \frac{(x+1) \cdot x}{2}$

Diese Unterlagen entstanden im Rahmen der Vorlesung *Fachdidaktik Informatik I* an der ETH Zürich unter der Betreuung von Prof. Dr. Juraj Hromkovič.

1 Vorwissen

Die vorliegende Unterrichtseinheit kann als Ergänzung zum Unterrichtsmaterial im Kapitel «4 Selbstkorrigierende Kodierung» im Lehrmittel *Informatik - Daten verwalten, schützen und auswerten* betrachtet werden. Die Unterrichtssequenz ist so aufgebaut, dass sie vorzugsweise im Anschluss an **Aufgabe 4.29** und **vor der Lernaufgabe/Projektaufgabe 4.30** durchgeführt wird.

Wesentlich für die Unterrichtseinheit sind die Textpassagen ab Unterkapitel «Effizient Kodierungen für viele Nachrichten finden» und insbesondere der **Kartentrick aus Beispiel 4.4**.

Ferner wird gegen Ende der Unterrichtseinheit Bezug zu Gauss'schen Summenformel in der **Aufgabe 4.11** und zu ihrer Musterlösung genommen.

Die Aufgaben D und E in diesen Unterlagen sind anspruchsvoll und benötigen einiges an mathematischem Wissen: hierzu gehören Fertigkeiten wie das Lösen von Ungleichungen und quadratischen Gleichungen sowie dem qualitativen Skizzieren von Graphen quadratischer Funktionen.

2 Zielsetzung und Mehrwert der Unterrichtssequenz

Die beschriebene Sequenz soll eine ausführliche Vorbereitung auf die Knobelaufgabe 4.31 und das Beispiel 4.5 im Lehrbuch bieten. Ab Beispiel 4.5 und bis und mit Aufgabe 4.33 im Lehrmittel geht es darum, dass die Lernenden die Kartentrick-Kodierung so weit optimieren, dass die benötigte Kontrollbitslänge möglichst klein wird, was letztlich auf eine optimale Spaltenwahl in den Fehlermeldetabellen (Beispiel 4.5) führen soll.

Die Unterrichtseinheit kann bei der Suche nach effizienteren Kodierungen einen Beitrag leisten. Ziel ist es, dass die Lernenden anhand einer alternativen Kartentrick-Kodierung erfahren, dass sich die Kontrollbitslänge gegenüber der im Lehrbuch vorgestellten Kartentrick-Kodierung verbessern lässt. Im besten Fall werden die Schülerinnen und Schüler dazu angeregt, selber nach effizienteren Kodierungen zu suchen.

Aus fachwissenschaftlicher Sicht steckt hinter dem neuen Kartentrick - im Folgenden mit ◺-Trick (sprich *Dreieckstrick*) bezeichnet - keine wesentlich neue Erkenntnis: das Bestimmen der Kontrollkarten im ◺-Trick ist so angeleitet, dass es weniger Kontrollkarten als im Kartentrick aus Beispiel 4.4 braucht. Dies liegt daran, dass durch die einzelnen Kontrollkarten im ◺-Trick «mehr» kontrolliert wird als beim klassischen Kartentrick. In der Sprache der Fehlermeldetabellen bedeutet dies, dass der ◺-Trick (bewusst!) keine optimale, aber zumindest bessere Wahl für die Verteilung der Kreuzchen auf die Spalten in der Fehlermeldetabelle bietet als der Kartentrick aus Beispiel 4.4.

Der Mehrwert der Unterrichtseinheit liegt also weniger in der fachwissenschaftlichen Erkenntnis als viel mehr in ihrem fachdidaktischen Nutzen:

- *Vertiefung und Erweiterung:* Die Unterrichtssequenz ist so aufgebaut, dass sie für den ◺-Trick fast identische Aufgaben anbietet wie das Lehrbuch für den Trick in Beispiel 4.4.

  Insofern kann das Vorliegende einerseits als eine Vertiefung von bereits vorhandenen Konzepten im Unterkaptitel «Effizient Kodierungen für viele Nachrichten finden» angesehen werden. Die dort eingeübten Fertigkeiten werden in dieser Unterrichtseinheit mit dem ◺-Trick gefestigt.

  Andererseits stellt es auch eine Erweiterung von bereits Vorhandenem dar, da eine neue Form der Kodierung präsentiert wird und die Frage nach kürzerer Kontrollbitslänge mehr Gewicht erhält.

  Letztlich wird am Rande auch die Gauss'sche Summenformel und ihre Herleitung aus Aufgabe 4.11 aufgegriffen und vertieft.

- *Übergang zu einem neuen Konzept:* Der Grundgedanke der vorgestellten Unterrichtsequenz ist, dass die Lernenden realisieren, dass es hinsichtlich der Kontrollbitslänge effizienter als beim klassichen Kartentrick geht. Mit dem ◺-Trick soll eben dies bei den Lernenden ausgelöst werden. Idealweise stellen sich die Lernenden am Ende der Sequenz die Frage, ob sich hinsichtlich der Kontrollbitslänge nicht weitere Verbesserungen finden lassen. Damit wäre für die Aufgabe 4.31 bzw. das Beispiel 4.5 die ideale Vorbereitung gegeben.

Nicht zuletzt nimmt die Ungleichung

$$m + x \leq \frac{(x+1) \cdot x}{2}$$

in der Lösung zur Aufgabe E die berühmte Ungleichung

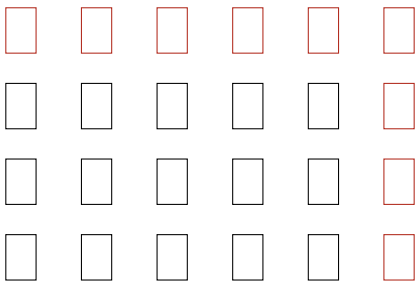
$$m + x \leq 2^x + 1$$

aus Beispiel 4.5 vorweg. In dieser Unterrichtseinheit wird die Länge der kodierten Nachricht $m + x$ nicht durch eine Exponentialfunktion sondern durch eine - bezüglich Abschätzung etwas ungünstigere - quadratische Funktion abgeschätzt. Die Besprechung der Lösung zu Aufgabe E kann einen spannenden Weg eröffnen, um im Anschluss Beispiel 4.5 zu thematisieren.

3 Unterrichtseinheit

In der nachfolgenden Unterrichtseinheit geht es darum, eine weitere Möglichkeit kennen zu lernen, wie effiziente Kodierungen generiert werden können. Die neue Kodierung liefert kürzere Code-Wörter als die Kartentrick-Kodierung im Lehrmittel. Die Unterrichtseinheit soll einerseits als Motivation dienen, effizientere Kodierungen suchen zu wollen. Andererseits liefert sie Hinweise, wie weitere Optimierungen hinsichtlich der Kontrollbitslänge umgesetzt werden könnten. Sie dient als Vorbereitung für die Knobelaufgabe 4.31 und das Beispiel 4.5 im Lehrmittel *Informatik - Daten verwalten, schützen und auswerten*.

Der im Lehrbuch vorgestellte Kartentrick (Beispiel 4.4) bedient sich eines speziellen Kartenlege-Schemas, bei dem ein Freiwilliger eine Anzahl Karten *in Form eines Rechtecks* arrangieren soll. Anschliessend werden entlang des halben Rechteckumfanges Kontrollkarten gelegt, die die Karten innerhalb des Rechtecks überwachen.



Auf der nächsten Seite lernen Sie einen zweiten Kartentrick - im Folgenden ◺-Trick genannt - kennen, der sich eines anderen Lege-Schemas bedient. Die Karten werden nicht rechteckig angeordnet, sondern *in Form eines Dreiecks*.

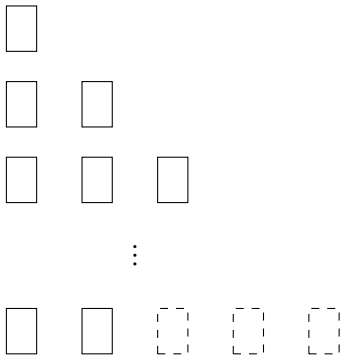
Der Dreieckstrick

Hier wird Ihnen der alternative ◺-Trick vorgestellt.

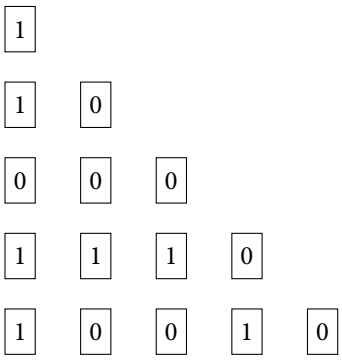
Der Ablauf des neuen Tricks ist grundsätzlich derselbe wie jener im Beispiel 4.4. Der Unterschied zwischen den beiden Tricks liegt in der speziellen Anordnung, wie die Karten gelegt werden, und im Verfahren, wie die Kontrollkarten bestimmt werden. In der linken Spalte wird das allgemeine Verfahren beschrieben, rechts ein möglicher konkreter Trickablauf.

① Nachdem der erste Magier den Raum verlassen hat, fordert der zweite Magier einen Freiwilligen auf, m Karten nach folgendem dreieckigem Schema zu legen. Der Freiwillige kann die Karten nach seinem Gutdünken offen oder verdeckt hinlegen:

Allgemein



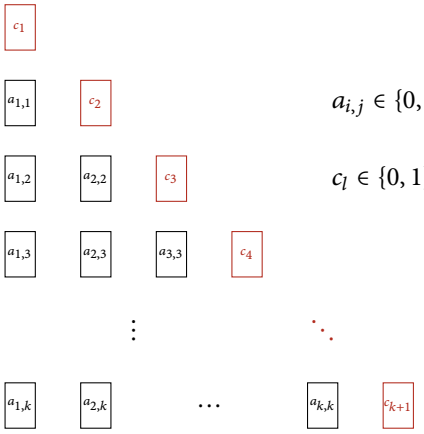
Konkret für $m = 15$



Falls der Freiwillige nicht selber die letzte Zeile vervollständigt, ergänzt der zweite Magier die letzte Zeile mit Karten, wobei diese beliebig offen oder verdeckt gelegt werden können. Wir können also davon ausgehen, dass das dreieckige Lege-Schema vollständig mit Karten ausgelegt ist.

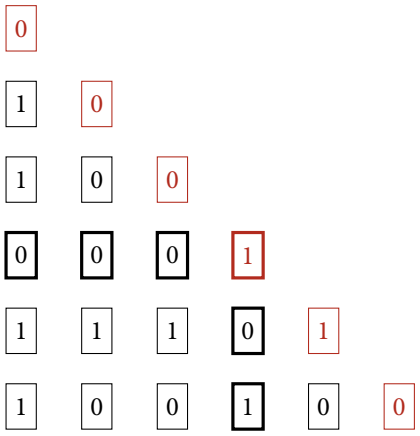
② Nun ergänzt der zweite Magier das gelegte Schema durch passende Kontrollkarten, die jeweils gewisse «Spalten und Zeilen» kontrollieren. Wir nummerieren die Spalten von links nach rechts mit $1 \le i \le k$ und die Zeilen von oben nach unten mit $1 \le j \le k$ durch. Die Kontrollkarten werden an das obere Ende jeder Spalte bzw. ans rechte Ende jeder Zeile gelegt. Die oberste Kontrollkarte kontrolliert nur eine Spalte, die unterste Kontrollkarte nur eine Zeile.

Der Wert der entsprechenden Kontrollkarte ergibt sich als Summe der Spalten- und Zeileneinträge ***modulo*** 2*, wobei jeweils die Zeile links der Kontrollkarte bzw. die Spalte unterhalb der Kontrollkarte betrachtet wird.*



$a_{i,j} \in \{0,1\}, 1 \le i, j \le k$

$c_l \in \{0,1\}, 1 \le l \le k+1$



Die allgemeine Vorschrift zum Setzen der Kontrollkarte c_l lautet:

$$c_l = a_{1,l-1} \oplus a_{2,l-1} \oplus \ldots \oplus a_{l-1,l-1} \oplus a_{l,l} \oplus a_{l,l+1} \oplus \ldots \oplus a_{l,k}$$

(Werte mit Indizes < 1 oder $> k$ werden auf 0 gesetzt.)

c_4 wird ermittelt durch:

$$c_4 = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

③ Wenn nun eine (1) beliebige Karte umgedreht wurde, so findet der zweite Magier die geflippte Karte, denn: Jede vom Freiwilligen gelegte Karte (Nachrichtenkarte) ist durch genau zwei Kontrollkarten kontrolliert. Genauer, jede Position $(i, j) \in \{1, \ldots, k\}^2$ mit $i \le j$ ist durch c_i und c_{j+1} kontrolliert. Jede Kontrollkarte kontrolliert sich selber.

Wurde eine Nachrichtenkarte geflippt, so stimmen genau zwei Kontrollkarten, sagen wir c_i und c_j mit $i < j$, nicht. Entsprechend muss die Nachrichtenkarten an der Position $(i, j-1)$ die gesuchte Karte sein. Wurde eine Kontrollkarte geflippt, so stimmt genau eine Kontrollkarte nicht. Ebene diese Kontrollkarte muss dann die geflippte Karte sein.

Bemerkung

Den ◺-Trick können wir verwenden, um Nachrichten für beliebige Bitlängen m zu kodieren. Die Rolle der Nachrichtenkarten spielen die Nachrichtenbits. Kontrollkarten entsprechen Kontrollbits.
Falls die Länge m der zu kodierenden Nachricht so ist, dass die letzte Zeile des Dreiecks nicht ausgefüllt wird, so, wird die letzte Zeile im Schema einfach durch eine beliebige aber bekannte (!) Bitfolge ergänzt.[1] Hier bedarf es einer Absprache: Wir wählen jeweils ein Folge von Nullen.

Aufgabe A

Wir haben $2^{12} = 4096$ Nachrichten der Länge 12 Bits. Nutzen Sie den ◺-Trick, um Code-Wörter für die folgenden Nachrichten zu generieren:

i) 010111101010

ii) 000000000000

iii) 111100101100

Aufgabe B

Man verwendet für die Nachrichten die Nachrichtenbits $a_1a_2a_3a_4a_5a_6$ der Länge 6. Die folgenden Nachrichten sind beschädigt. Bestimmen Sie die ursprüngliche Nachricht (das gesendete Code-Word), wenn Sie annehmen, dass genau 1 Bit geflippt wurde.

i) 0111010000

ii) 0111010010

iii) 1110100100

iv) 1110100000

Aufgabe C

Wir wollen 1-fehlerkorrigierende Kodierungen so kurz wie möglich konstruieren, d. h. so wenig Kontrollbits verwenden, wie es nur geht.

Mit den beiden Kartentricks haben wir nun zwei Verfahren an die Hand erhalten, um solche 1-fehlerkorrigierende Kodierungen für beliebige Bitlängen m zu generieren.

Untersuchen Sie die Qualität der beiden Verfahren hinsichtlich der *Bitlänge* der entstehenden Kodierungen. Für den Kartentrick aus Beispiel 4.4 liefern die Aufgaben 4.25 und 4.29b zwei Optimierungen.

i) Ergänzen Sie die beigelegte[2] Tabelle.

ii) Was können Sie hinsichtlich der Qualität der beiden Kodierungsverfahren aussagen?

[1] Das künstliche Ergänzen der letzten Zeile ist nicht notwendig; die Selbstkorrektur wäre auch sonst möglich. Um die Analogie zum Kartentrick zu erhalten, behalten wir die Ergänzung jedoch bei.

[2] Vgl. Anhang.

Aufgabe D

Die Anzahl Karten, die benötigt werden, um ein komplettes Dreieck zu legen, entspricht einer sogenannten *Dreieckszahl*. Die ersten Dreieckszahlen sind $1, 3, 6, 10, 15, 21, 28, 36, 45, 55, \ldots$.
Allgemein formuliert ist eine Dreieckszahl eine Zahl, die der Summe aller natürlicher Zahlen von 1 bis zu einer Obergrenze n entspricht:

$$\Delta_n = 1 + 2 + \ldots + n\,.$$

Bestimmen Sie eine (explizite) Formel für Δ_n.

Hinweis: In Aufgabe 4.11 im Lehrmittel wurde eine Formel s_n für die Anzahl notwendiger Vergleiche von n Code-Wörtern zur Bestimmung des Abstands einer Kodierung hergeleitet:

$$s_n = \frac{n \cdot (n-1)}{2}\,.$$

In der Lösung zu Aufgabe 4.11 wurde die Formel für s_n auf drei verschiedene Möglichkeiten gefunden. Insbesondere mit der erstgenannte Methode lässt sich eine Formel für Δ_n ableiten.

Aufgabe E

In Aufgabe C ii) haben wir vermutet, dass der ◺-Trick hinsichtlich Kontrollbitslänge etwas besser abschneidet als der Kartentrick im Beispiel 4.4.
In dieser Aufgabe soll *bewiesen* werden, dass für jede Nachrichtenlänge m der ◺-Trick höchstens so viele Kontrollbits braucht wie der Kartentrick im Lehrbuch.

i) Leiten Sie eine Formel für die Anzahl benötigter Kontrollbits x im ◺-Trick in Abhängigkeit der Nachrichtenbitslänge m her.

ii) Beweisen Sie, dass der ◺-Trick hinsichtlich der Kontrollbitslänge besser ist als der Kartentrick aus Beispiel 4.4.

Aufgabe F

Zum Abschluss dieser Unterrichtseinheit sollen Sie sich nochmals einige Gedanken zum Gelernten machen.[3] Versuchen Sie hierzu prägnant die folgenden Kontrollfragen für sich zu beantworten:

i) Habe ich gut verstanden, wie der ◺-Trick abläuft und könnte ich den Trick selber mit einem/einer Gehilfen/Gehilfin durchführen?

ii) Kann ich den ◺-Trick verwenden, um für Nachrichtenlängen der Länge m Kodierung herzustellen?

iii) Kann ich an einem Beispiel erläutern, wie ein einzelner Fehler in einer erhaltenen Nachricht korrigiert werden kann?

iv) Zwischen den benötigten Anzahl Kontrollbits x und der Nachrichtenlänge m besteht folgender Zusammenhang:

$$m + x \leq \frac{(x+1) \cdot x}{2}\,.$$

Kann ich erklären, wie es zu dieser Ungleichung kommt?

v) Welche Vorteile/Nachteile sehe ich in der Kodierung mittels dem ◺-Trick im Vergleich zum ursprünglichen Kartentrick aus Beispiel 4.4?

vi) Ergeben sich aus dem Vorangestellten für mich neue Fragen im Zusammenhang mit effizienten Kodierungen, über die es sich lohnen könnte nachzudenken?

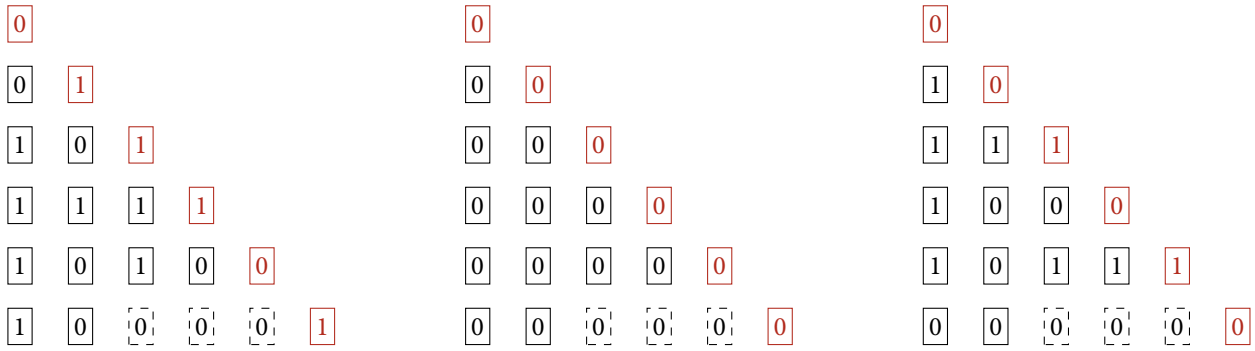
[3] Die Antworten zu den gestellten Fragen finden sich z.T. in anderen Lösungen oder Ausführungen. Oder es handelt sich eher um metakognitive Fragen. Auf Lösungen zu diesen Fragen wird verzichtet.

4 Lösungen

Lösung zu Aufgabe A

Mit der Zahl 12 lässt sich das Schema nicht komplett legen. Es braucht 15 Karten/Nachrichtenbits. Unsere Code-Wörter werden also jeweils künstlich mit 3 Nullen ergänzt: Das Lege-Schema wird *zeilenweise von links nach rechts* mit den Nachrichtenbits «belegt».

i) 010111101010 ii) 000000000000 iii) 111100101100



Nun Bedarf es wiederum einer Abmachung, in welcher Reihenfolge die Kontrollbits $c_1, c_2, \ldots, c_6$ hinter die Nachricht angehängt werden sollen. Wir wählen die naheliegende Reihenfolge entsprechend der Nummerierung der c_i.

Somit erhalten wir die folgenden Lösungen:

i) 010111010100000**11101** ii) 000000000000000**000000** iii) 111100101100000**001010**

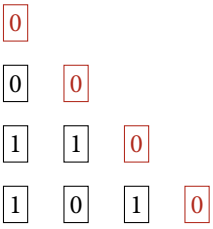
Lösung zu Aufgabe B

Da mit 6 Karten ein Dreieck komplett belegt werden kann, ist es nicht nötig, dass der Nachricht künstlich eine Bitfolge am Ende angehängt wird.

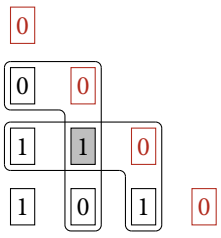
i) Wir stellen zwei Lösungswege vor:

Schema Eine etwas aufwändigere Methode führt über das Zeichnen des Schemas und dem Befüllen mit den gegebenen Werten.

Erhaltene (fehlerhafte) Nachricht 0111010000



Spalten und Zeilen mit ungerader Anzahl Einsen



Das geflippte Bit muss $a_3 = 1$ sein. Die korrekte Nachricht ist 0101010000.

Berechnung Eine schnellere Methode ohne das Schema zu zeichnen ist das Konzept der Überwachung anzuwenden. Wir nehmen an, dass 011101 die ursprüngliche (unverfälschte) Nachricht ist und *berechnen* die dazugehörigen Kontrollbits.

$$c_1 = a_1 \oplus a_2 \oplus a_4 = 0 \oplus 1 \oplus 1 = 0$$
$$c_2 = a_1 \oplus a_3 \oplus a_5 = 0 \oplus 1 \oplus 0 = 1$$
$$c_3 = a_2 \oplus a_3 \oplus a_6 = 1 \oplus 1 \oplus 1 = 1$$
$$c_4 = a_4 \oplus a_5 \oplus a_6 = 1 \oplus 0 \oplus 1 = 0$$

Danach vergleichen wir das Ausgerechnete mit der ehemaligen Nachricht:

Berechnete Kodierung für die Nachricht 011101	0111010110
Erhaltene Nachricht	0111010000

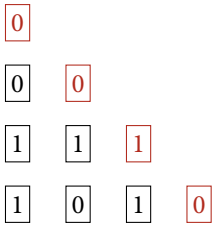
Wir stellen fest, dass sich die zwei Bitsequenzen an den Stellen c_2 und c_3 unterscheiden. Das einzige Nachrichtenbit, das durch c_2 und c_3 gemeinsam überwacht wird, ist a_3.

Somit wurde also a_3 geflippt und die korrekte Nachricht ist: 0101010000.

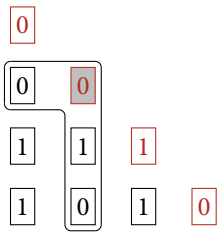
ii) Wir wenden beide Methoden an.

Schema

Erhaltene (fehlerhafte) Nachricht 0111010010



Spalte und Zeile mit ungerader Anzahl Einsen



Das geflippte Bit muss das Kontrollbit c_2 sein. Die korrekte Nachricht ist 0111010110.

Berechnung Für die zweiten Methode berechnen wir für $a_1a_2a_3a_4a_5a_6 = 011101$

$$c_1 = a_1 \oplus a_2 \oplus a_4 = 0 \oplus 1 \oplus 1 = 0$$
$$c_2 = a_1 \oplus a_3 \oplus a_5 = 0 \oplus 1 \oplus 0 = 1$$
$$c_3 = a_2 \oplus a_3 \oplus a_6 = 1 \oplus 1 \oplus 1 = 1$$
$$c_4 = a_4 \oplus a_5 \oplus a_6 = 1 \oplus 0 \oplus 1 = 0$$

und vergleichen das Ausgerechnete mit der ehemaligen Nachricht:

Berechnete Kodierung für die Nachricht 011101	0111010110
Erhaltene Nachricht	0111010010

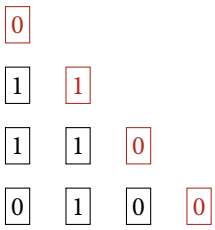
Wir stellen fest, dass sich die zwei Bitsequenzen an den Stellen c_2 unterscheiden.

Somit wurde also das Kontrollbit c_2 geflippt und die korrekte Nachricht ist: 0111010110.

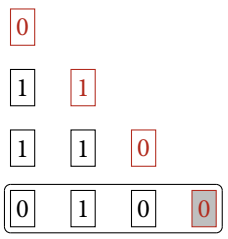
iii) Wir wenden beide Methoden an.

Schema

Erhaltene (fehlerhafte) Nachricht 1110100100



Zeile mit ungerader Anzahl Einsen



Das geflippte Bit muss das Kontrollbit c_4 sein. Die korrekte Nachricht ist 1110100101.

Berechnung Die Berechnung der Kontrollbits liefert für $a_1a_2a_3a_4a_5a_6 = 111010$

$$c_1 = a_1 \oplus a_2 \oplus a_4 = 1 \oplus 1 \oplus 0 = 0$$
$$c_2 = a_1 \oplus a_3 \oplus a_5 = 1 \oplus 1 \oplus 1 = 1$$
$$c_3 = a_2 \oplus a_3 \oplus a_6 = 1 \oplus 1 \oplus 0 = 0$$
$$c_4 = a_4 \oplus a_5 \oplus a_6 = 0 \oplus 1 \oplus 0 = 1$$

Der Vergleich des Ausgerechneten mit der ehemaligen Nachricht zeigt:

Berechnete Kodierung für die Nachricht 111010	1110100101
Erhaltene Nachricht	1110100100

Die zwei Bitsequenzen unterscheiden sich an den Stellen c_4.

Somit wurde also das Kontrollbit c_4 geflippt und die korrekte Nachricht ist: 1110100101.

d) Bei der erhaltenen Nachricht 1110100000 wurde das Nachrichtenbit a_5 geflippt.

Die korrekte Nachricht lautet: 1110000000.

Lösung zu Aufgabe C

i) Der Kartentrick aus Beispiel 4.4 lässt sich hinsichtlich Kontrollbitslänge so optimieren, dass für die Masse des Rechtecks zwei infrage kommenden Zahlen mit minimalem Abstand gewählt werden. (Aufgabe 4.25)

Weiter kann auf das Kontrollbit in der rechten oberen Ecke verzichtet werden. (Aufgabe 4.29b)

Unter Berücksichtigung dieser beider Optimierungen wurde auf der nächsten Seite die notwendige Länge der Kontrollbits für Nachrichtenlängen $m \in \{1, 2, \dots, 42\}$ bestimmt.

ii) Da die Werte in der letzten Spalte jeweils kleiner oder gleich den Werten in der zweitletzten Spalte sind, lässt sich folgende *Vermutung* aufstellen:

Die Kodierung mittels ◺-Trick ist für jedes m (≤ 42) hinsichtlich der Kontrollbitslänge mindestens so gut wie der Kartentrick aus Beispiel 4.4.

Bitlänge	Optimale Rechteckmasse		Kartentrick (Beispiel 4.4)	◺-Trick
m	a	b	Benötigte Anzahl Kontrollbits	Benötigte Anzahl Kontrollbits
1	1	1	2	2
2	1	2	3	3
3	1	3	4	3
4	2	2	4	4
5	1	5	6	4
6	2	3	5	4
7	1	7	8	5
8	2	4	6	5
9	3	3	6	5
10	2	5	7	5
11	1	11	12	6
12	3	4	7	6
13	1	13	14	6
14	2	7	14	6
15	3	5	8	6
16	4	4	8	7
17	1	17	18	7
18	3	6	9	7
19	1	19	20	7
20	4	5	9	7
21	3	7	10	7
22	2	11	13	8
23	1	23	24	8
24	4	6	10	8
25	5	5	10	8
26	2	13	15	8
27	3	9	12	8
28	4	7	11	8
29	1	29	30	9
30	5	6	11	9
31	1	31	32	9
32	4	8	10	9
33	1	33	34	9
34	2	17	19	9
35	5	7	12	9
36	6	6	12	9
37	1	37	38	10
38	2	19	21	10
39	1	39	40	10
40	5	8	13	10
41	1	41	42	10
42	6	7	13	10

Lösung zu Aufgabe D

Es gilt gemäss den Lösungen zu Aufgabe 4.11 für $n \geq 1$

$$s_n = \frac{n \cdot (n-1)}{2} \qquad \text{und}$$
$$s_n = (n-1) + (n-2) + \ldots + 2 + 1\,.$$

Indem wir die rechten Seiten der obigen beiden Gleichungen für $n+1$ gleichsetzen, erhalten wir für jede natürliche Zahl n:

$$n + (n-1) + \ldots + 2 + 1 = s_{n+1} = \frac{(n+1) \cdot n}{2}\,.$$

Daraus folgt für Δ_n:

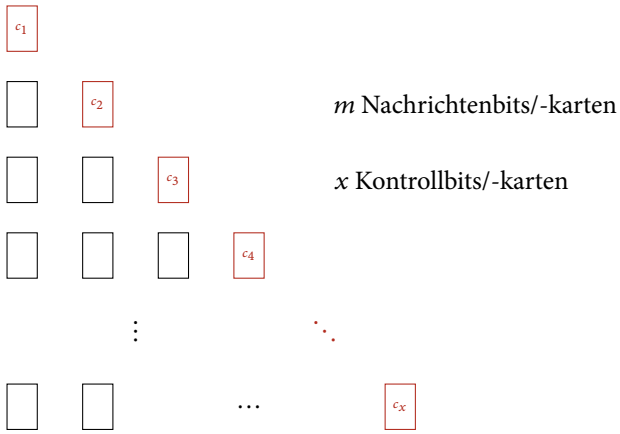
$$\Delta_n = 1 + 2 + \ldots + n = \frac{(n+1) \cdot n}{2} \qquad \text{für } n \geq 1\,.$$

Lösung zu Aufgabe E

Es sei m die Bitlänge der Nachricht. Mit x bezeichnen wir die benötigte Anzahl Kontrollbits.

i) Herleitung der Formel für die benötigte Anzahl Kontrollbits x

Wir betrachten das allgemeine Lege-Schema vom ◺-Trick:



Das obige Dreieck hat x Zeilen. Wir wählen x *minimal*, sodass mindestens eine Nachrichtenkarte in der letzten Zeile des Dreiecks liegt.)

Die Anzahl gelegter Karten kann nun auf zwei Arten dargestellt werden:

① Summe von Nachrichten- und Kontrollkarten:

$$m + x$$

② Die Anzahl des komplett belegten Dreiecks mit x Zeilen kann mittels der x-ten Dreieckszahl beschrieben werden:

$$\Delta_x = \frac{(x+1) \cdot x}{2}$$

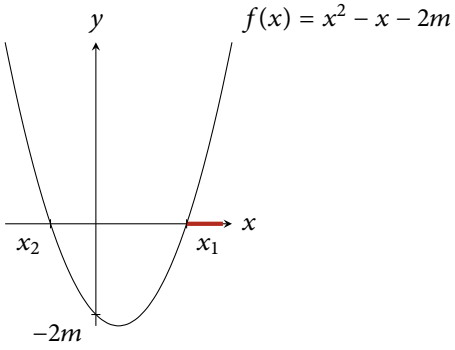
Je nach Wahl von m kann es sein, dass die letzte Zeile des Dreiecks nicht komplett mit Karten belegt ist. Das führt uns nun auf folgende Ungleichung:

$$\boxed{m + x \leq \frac{(x+1) \cdot x}{2}}$$

Wir lösen die Ungleichung nach x auf.

$$
\begin{aligned}
m + x &\leq \frac{(x+1) \cdot x}{2} \\
2m + 2x &\leq x^2 + x \\
0 &\leq x^2 - x - 2m \qquad (*)
\end{aligned}
$$

Der Ausdruck auf der rechten Seite des Ungleichheitszeichens beschreibt eine nach oben geöffnete Parabel 2. Ordnung mit negativem y-Achsenabschnitt $-2m$.



Die Funktion $f(x) = x^2 - x - 2m$ besitzt eine positive reelle Nullstelle x_1 und eine negative reelle Nullstelle x_2. Also müssen die infrage kommenden Lösungen der Ungleichung $(*)$ alle natürlichen Zahlen $\geq x_1$ sein.

Positive Nullstellen x_1 von $f(x)$: $x^2 - x - 2m = 0 \Rightarrow x_1 = \frac{1}{2}\left(\sqrt{8m+1} + 1\right)$

(Zweite Nullstelle $x_2 = \frac{1}{2}\left(-\sqrt{8m+1} + 1\right) < 0$ für $m \geq 1$)

Die gesuchte Anzahl Kontrollbits x für die Nachrichtenlänge m beträgt also:

$$\boxed{x = \left\lceil \frac{1}{2}\left(\sqrt{8m+1} + 1\right) \right\rceil}$$

(Hierbei stehen die eckigen Klammern $\lceil \ldots \rceil$ für das Aufrunden auf die nächst grössere natürliche Zahl.)

ii) Wir weisen nach, dass der ◺-Trick mindestens so gut ist wie der Kartentrick aus Beispiel 4.4, was die Länge der Kontrollbits anbelangt.

Für eine Nachricht der Bitlänge m benötigt der Kartentrick aus Beispiel 4.4 mindestens $2\sqrt{m}$ Bits.

Wir verifizieren folgende Ungleichung:

$$
\begin{aligned}
\frac{1}{2}\left(\sqrt{8m+1} + 1\right) &\leq 2\sqrt{m} \\
\sqrt{8m+1} + 1 &\leq 4\sqrt{m} \\
\sqrt{8m+1} &\leq 4\sqrt{m} - 1 \\
8m + 1 &\leq 16m - 8\sqrt{m} + 1 \\
8\sqrt{m} &\leq 8m \\
m &\leq m^2 \\
1 &\leq m
\end{aligned}
$$

Somit ist die erste Ungleichung für jedes $m \geq 1$ erfüllt. Wir haben gezeigt, dass der Kodierung mittels ◺-Trick nie mehr Kontrollbits braucht als über den Kartentrick in Beispiel 4.4.

5 Anhang

Tabelle zur Bestimmung der benötigten Anzahl Kontrollbits für den Kartentrick aus Beispiel 4.4 und für den ◺-Trick.

Bitlänge	Optimale Rechteckmasse		Kartentrick (Beispiel 4.4)	◺-Trick
m	a	b	Benötigte Anzahl Kontrollbits	Benötigte Anzahl Kontrollbits
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				