Aufgabensequenz Fachdidaktik zu selbstkorrigierenden Kodierungen: Tic-Tac-Toe

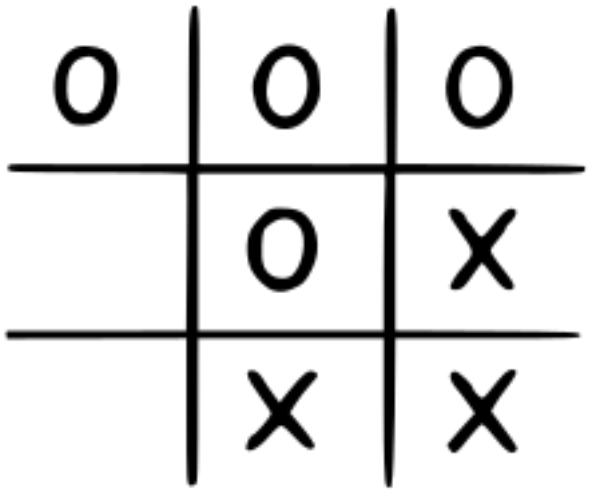
Voraussetzungen: Die Schüler*innen haben bereits Prüfsymbole, die Fehlerkorrektur und die Hamming-Codes kennengelernt.

Zeitumfang: 1-2 Lektionen inkl. Besprechung

Zielsetzung. Die Sequenz soll die Konzepte Fehlererkennung und -behebung vertiefen.

Situation:

Anna und Beat spielen auf Entfernung Tic-Tac-Toe. Bei diesem Spiel setzen die Spieler abwechselnd ihre Symbole (zuerst Kreis, dann Kreuz) in eines der 9 Felder. Ziel des Spiels ist es, drei gleiche Symbole in einer Zeile, einer Spalte oder auf einer der Diagonalen zu haben.



Alice und Beat teilen sich ihre Züge kodiert mit und möchten aber diese so absichern können, dass einzelne Fehler bei der Übertragung der Züge automatisch entdeckt und idealerweise auch korrigiert werden können.

In den Aufgaben unten wird es jeweils nur darum gehen, einen einzelnen Spielzug (das heisst: das Setzen von einem festgelegten Symbol in das Feld) zu machen.

Alice und Beat einigen sich auf die folgende **Kodierung**: Die Zeilen sind von 1-3 nummeriert und die Spalten ebenfalls. Wenn nun eine Person ihr Symbol ins Feld in Zeile 1 und Spalte 3 setzen möchte, so würde sie die Binärdarstellungen beider (Zeilen- und Spalten-) Zahlen mit je 2 Bit bestimmen und diese aneinanderhängen.
Es ist üblich, dass zuerst die Zeile und anschliessend die Spalte angegeben wird.

Beispiel:

Zeile 1 -> binär 01
Spalte 3 -> binär 11
Code-Wort für das gelb markierte Feld -> <u>0111</u>

	1	**2**	**3**
1			
2			
3			

Hinweis: Wir betrachten in der Folge ausschliesslich Fehler, welche das Flippen von einem Bit (von 1 auf 0 oder umgekehrt) betreffen (und nicht etwa, dass ein Bit gar nicht übermittelt wird).

Aufgaben:

1) Geben Sie die Code-Wörter der übrigen acht Felder an.
   Schreiben Sie diese direkt ins Gitter oben.

2) Wie viele Bit müsste man an ein Code-Wort anhängen, damit ein einzelner Fehler in der Übertragung mit Sicherheit bemerkt würde? Beschreiben Sie genau, wie diese(s) Kontrollbit(s) bestimmt werden kann/können.

3) Wie viele Kontrollbit werden für beliebige Bitfolgen der Länge 4 benötigt, um eine 1-fehlerkorrigierende Kodierung zu erhalten? Geben Sie Ihre Überlegung an.
   Wie lang wäre damit eine 1-fehlerkorrigierende Kodierung eines Spielzugs total?

Wir wollen uns jetzt überlegen, ob man für dieses Problem allenfalls auch mit weniger Kontrollbit eine 1-fehlerkorrigierende Kodierung erreichen könnte, indem wir einen Teil der Information durch die Länge der verschickten Nachricht kodieren. Dazu verwenden wir für die Fehlerkorrektur je nach Code-Wort eine **unterschiedliche Anzahl von Kontrollbit**.

Die Kodierung von einem Feld im Gitter ist ohne Kontrollbit von der Form $a_1 a_2 a_3 a_4$.

4) Gibt es einen Fehler beim Flippen eines Bit, den man sofort erkennen würde (auch wenn man ihn noch nicht sofort korrigieren kann)? Wenn ja, benennen Sie ihn.

5) Um nun Kontrollbit einzuführen wird die folgende Aufteilung der 9 Code-Wörter vorgeschlagen:
   a. Das Kode-Wort mit keiner Null: 1111
   b. Die Kode-Wörter mit genau einer Null: 0111, 1011, 1101, 1110
   c. Die Kode-Wörter mit genau zwei Nullen: 1010, 1001, 0101, 0110

   Überlegen Sie separat für die drei Kategorien, wie viele Kontrollbit jeweils mindestens nötig sind für eine 1-fehlerkorrigierende Kodierung und was die Bedeutung der Kontrollbit sein könnte.

   Hinweis: Sie sollten so vorgehen, dass Sie für jede der drei Gruppen a.-c. eine unterschiedliche Anzahl an Kontrollbit verwenden.

6) Lesen Sie nun die Lösung der Aufgabe 5) durch und vergleichen Sie diese mit Ihrer Lösung in Bezug auf die durchschnittliche Länge der Code-Wörter. Für die restlichen Aufgaben verwenden Sie bitte die Kodierung aus der Lösung.

7) Bestimmen Sie die Code-Wörter zu den folgenden Feldern auf dem Spielfeld:

   a. Feld unten rechts
   b. Feld oben rechts
   c. Feld Mitte Links

8) Sie erhalten die folgenden Bitfolgen. Entscheiden Sie, ob die Bitfolge korrekt übermittelt wurde oder ob ein Fehler gemacht wurde. Falls es einen Fehler gab, entscheiden Sie, an welcher Stelle er aufgetreten ist und korrigieren Sie ihn. Bestimmen Sie damit, welches Feld gemeint war.

    a. 1011
    b. 1001110
    c. 101101

9) Bei den folgenden Nachrichten wurden zwei Bit geflippt. Geben Sie sämtliche möglichen ursprünglichen (d.h. korrekten) Nachrichten an.
    a. 1011101
    b. 1110110
    c. 111000

10) Zum Schluss betrachten wir noch eine andere Art der Kodierung für die Felder:
    Die Felder werden mit den Zahlen 0, 1, 2, ..., 8 durchnummeriert.
    Die Kodierung können Sie der folgenden Tabelle entnehmen:

Feldnummer	0	1	2	3	4	5	6	7	8
Nummer binär	0	1	10	11	100	101	110	111	1000
Kodierung	0	01	010	0011	00000	01011	10101	11110	001000

    Begründen Sie, weshalb die Kodierung 1-fehlerkorrigierend ist und bestimmen Sie die durchschnittliche Länge einer übermittelten Nachricht.
    Erkennen Sie in den Kodierungen der Felder 4 bis 7 ein bekanntes Muster?

Lösungen:

1) Die 9 Kode-Wörter sind direkt in der Tabelle angegeben.

0101	0110	0111
1001	1010	1011
1101	1110	1111

2) Zur Fehlererkennung einer beliebigen Bitfolge reicht **ein** Paritätsbit. Dieses ergänzt die Bitfolge so, dass die Anzahl Ziffern '1' in der erweiterten Folge gerade ist. Angewendet auf unser 3x3-Gitter sehen die erweiterten Folgen so aus:

0101**0**	0110**0**	0111**1**
1001**0**	1010**0**	1011**1**
1101**1**	1110**1**	1111**0**

Hinweis: Diese Aufgabe ist für das Verständnis des Folgenden nicht unbedingt nötig und könnte auch weggelassen werden.

3) 9 Nachrichten kann man mit $n = 4$ Nachrichtenbit kodieren. Nun wird ein x so gesucht, dass $4 + x \leq 2^x - 1$. Die kleinste Zahl, welche diese Ungleichung erfüllt, ist

$$x = 3.\ (7 \leq 7)$$

Es werden also 3 Kontrollbit benötigt, um einen 1-fehlerkorrigierenden Code zu erhalten.

4) Die Ziffern 00 kommen in keinem Code-Wort in derselben Hälfte vor.
Es gibt also zwei Fehler, die sofort erkannt, aber nicht behoben werden können, nämlich wenn in der vorderen oder hinteren Hälfte des Code-Wortes **01 zu 00** oder **10 zu 00** geflippt wird.

5) Wir betrachten die drei Fälle separat:
   a. Wenn keine Null vorkommt, so ist auch kein Kontrollbit nötig. Wir könnten aufgrund der Länge (4 Bit) der übertragenen Nachricht auf das Feld schliessen (dazu würde eigentlich auch nur 1 Bit ausreichen).
   b. Wenn im Code-Wort genau eine Null vorkommt, so benötigt man zwei Kontrollbit. Das erste Kontrollbit könnte angeben, ob die Null in der vorderen (=0) oder hinteren (=1) Hälfte ist und das zweite, ob es sich um das erste (=0) oder zweite (=1) Bit der Hälfte handelt; zum Beispiel: 1011 -> 101101 (die Null ist in der vorderen Hälfte, zweites Bit)

Diese Kodierung führt zu insgesamt 6 Bit (Nachricht 4 Bit und 2 Kontrollbit). In der Länge des Code-Wortes ist die Anzahl der Nullen kodiert. Die Überlegung, dass die Kodierung wirklich fehlerkorrigierend ist, ist einfach: Wenn in den ersten vier Bit ein Fehler passiert, so wird das aufgrund der falschen Anzahl Nullen bemerkt und kann mit den Kontrollbit korrigiert werden (so könnte man auch mehrere Flips von 1 zu 0 korrigieren). Passiert aber ein Fehler in den Kontrollbit, so kann man wie im Beispiel unten vorgehen:

Der Empfänger sieht 101100, er rekonstruiert aus den Kontrollbit (00 -> die Null ist an erster Stelle) 0111 und vergleicht mit 1011 und bemerkt den Fehler in den Kontrollbit. Er weiss also, dass die ersten vier Bit korrekt übermittelt wurden.

c. Bei zwei Nullen muss die erste Null in der ersten Hälfte und die zweite Null in der zweiten Hälfte sein. Um die beiden Positionen in der jeweiligen Hälfte anzugeben, reichen eigentlich zwei Kontrollbit aus. Ein drittes Kontrollbit dient dazu anzugeben, dass in der ursprünglichen Nachricht genau zwei Nullen waren. **Der Wert des dritten Kontrollbit ist immer Null.**

Beispiel: 1001 -> 1001**100**

- Das erste Kontrollbit ist 1, denn die erste Null ist an der zweiten Stelle der vorderen Hälfte.
- Das zweite Kontrollbit ist 0, denn die zweite Null ist an der ersten Stelle der hinteren Hälfte.
- Das dritte Kontrollbit ist immer Null.

Aus den drei Fällen a.-c. erhalten wir also die folgende 1-fehlerkorrigierende Kodierung:

0101000	0110010	011100
1001100	1010110	101101
110110	111011	1111

6) Die durchschnittliche Länge entspricht dem Mittelwert der Längen, also:
$\frac{1}{9} * (4 * 6 + 4 * 7 + 1 * 4) = \frac{56}{9} \approx 6.22$ Bit
Diese Länge könnte noch einmal reduziert werden, indem man für die Nachricht 1111 nur ein einzelnes Bit verwendet.

7) s. Tabelle oben

8)

a. 1011 - > Länge 4, also richtig: 1111
b. 1001110 -> Länge 7, es gibt zwei Nullen innerhalb der ersten vier Bit, also muss der Fehler im zweiten Kontrollbit sein. -> 1001100
c. 101001 -> Länge 6, also eine Null in den ersten vier Bit; es kommen aber zwei Nullen vor, also sind die Kontrollbit richtig, diese sagen 1011 voraus.

0101	0110	0111
1001	1010	1011
1101	1110	1111

9)

a. 1011101
   7 Bit, das hinterste Bit ist per Konvention eine Null, das ist der erste Fehler.
   In den ersten vier Bit müssten zwei Nullen vorkommen, es gibt aber nur eine, also muss die zweite Null in der hinteren Hälfte ergänzt werden.
   Korrekt ist also **1001100**.

b. 1110110
   7 Bit, das hinterste ist korrekt.
   In den ersten vier Bit müssten zwei Nullen vorkommen, es gibt aber nur eine.
   Nach den Kontrollbit müssten die ersten vier Bit 1010 lauten, dann hätte die übertragene Nachricht aber nur einen Fehler. Das bedeutet, dass eines der Kontrollbit falsch ist, es muss das vordere sein (das Flippen eines korrekten Kontrollbit hätte zwei Korrekturen in den vier ersten Bit zur Folge).
   Die richtigen Kontrollbit sind also 010 -> vordere vier Bit: 0110
   Damit lautet die Nachricht korrekt: **0110010**

c. 111000
   6 Bit, also muss in den ersten vier Bit eine Null vorkommen.
   Es gibt zwei Möglichkeiten:
   - die Null in den ersten vier Bit ist am falschen Ort (zwei Fehler), der korrekte Ort wird durch die Kontrollbit 00 kodiert, also wäre **011100** korrekt;
   - die Null in den ersten vier Bit ist richtig, dann sind die Fehler bei den Kontrollbit und die korrekte Variante lautet **111011**.

Angesichts der Erkenntnisse aus Beispiel c. sieht man, dass die Kodierung nur 1-fehlerkorrigierend und nicht 2-fehlerkorrigierend ist.

10) Die Felder 0 bis 3 und das Feld 8 werden mit je einer eindeutigen Länge kodiert, daher sind diese Kodierungen 1-fehlerkorrigierend.
Für die Felder 4 (binär 100) bis 7 (binär 111) wurde die Kodierung folgendermassen erstellt:

Alle binären Nummern sind von der Form 1xy. Die Ziffer 1 kann also bei der Kodierung weggelassen werden.

Die Kodierung für 1xy lautet xyxyp, wobei p ein Paritätsbit für die Folge xy ist.
Damit ist die Kodierung dieser 4 Felder ebenfalls 1-fehlerkorrigierend.
Wird nämlich eines der ersten 4 Bit geflippt, so erkennt man aufgrund des Paritätsbit, wo der Fehler ist (die ersten 4 Bit sollten zwei identische Kopien sein).
Gibt es einen Fehler im Paritätsbit, so kann man das aufgrund der korrekten ersten vier Bit erkennen (die Korrektur ist trivial).
Die durchschnittliche Länge der Kodierung beträgt nur noch:

$$\frac{1}{9} * (1 + 2 + 3 + 4 + 4 * 5 + 6) = \frac{36}{9} = 4 \text{ Bit}$$