Das Schachbrett-Rätsel: Didaktische Anmerkungen und Lösungen

Vorwissen

Die SuS sollten über die grundlegenden Kenntnisse aus dem Kapitel „Selbstkorrigierende Kodierungen“ des Buches „Informatik: Daten verwalten, schützen und auswerten“ verfügen, insbesondere sollte die *Kartentrick-Kodierung* als Beispiel einer 1-fehlerkorrigierenden Kodierung bekannt sein.

Konzept

Mit einem neuen, verblüffenden Kartentrick – dem Schachbrett-Rätsel – erhalten die SuS eine anschauliche Einführung in die Hamming-Kodierung, entweder als Alternative zum allerletzten Abschnitt des Kapitels „Selbstkorrigierende Kodierungen“ oder als anschauliche Ergänzung dazu. Diese Lerneinheit zeigt zudem einen Zusammenhang zum Thema „binäre Suche“ auf.

Im letzten Abschnitt wird als Ergänzung untersucht, warum dieser Trick nicht für jedes Spielfeld funktioniert, dabei wird mit geometrischen Mitteln und kombinatorischen Zusammenhängen argumentiert. Der Kartentrick eröffnet also zahlreiche Einblicke.

Eine Möglichkeit zur konkreten Gestaltung der Lerneinheit

Der neue Kartentrick könnte erst einmal vorgeführt werden, eine Person müsste jedoch zuvor instruiert werden. Dies kann eine Lehrperson sein oder auch eine Schülerin oder ein Schüler. Die anfängliche Verblüffung ermuntert den Trick und den Bezug zur Hamming-Kodierung zu verstehen. Dazu kann selbstständig die Lernaufgabe bearbeitet werden (beiliegendes Dokument). Im Anschluss kann eine Gruppe den Trick vorführen (Aufgabe 1), eine Gruppe kann die Hamming-Kodierung erklären und eine Gruppe kann zeigen, warum diese Kodierung 1-fehlerkorrigierend ist (Aufgabe 2).

Quellen:

- How to send a self-correcting message (Hamming Codes): youtube.com/watch?v=X8jsijhllIA
- The impossible chessboard puzzle: youtube.com/watch?v=wTJI_WuZSwE&t=902s
- The almost impossible chessboard puzzle: youtube.com/watch?v=as7Gkm7Y7h4
- Hamming cods part 2, the elegance of it all: youtube.com/watch?v=b3NxrZOu_CE

Lösungen zu den Kontrollaufgaben:

1) a) 0000, dieses Feld ist in allen Komplementärmengen, also in keiner Teilmenge Q1 bis Q4.

   b) 1001, dieses Feld ist in Q1 und Q4, aber nicht in Q2 und Q3.

   c) 0010, dieses Feld liegt nur in Q2, nicht in Q1, Q3 und Q4.

2) Für das Schachbrett (8x8-Felder) betrachten wir analog folgende Teilmengen:

   Q1: Ist die Binärziffer des Feldes von der Form _ _ _ _ _ 1 ?
   Q2: Ist die Binärziffer des Feldes von der Form _ _ _ _ 1 _ ?
   Q3: Ist die Binärziffer des Feldes von der Form _ _ _ 1 _ _ ?
   Q4: Ist die Binärziffer des Feldes von der Form _ _ 1 _ _ _ ?
   Q5: Ist die Binärziffer des Feldes von der Form _ 1 _ _ _ _?
   Q6: Ist die Binärziffer des Feldes von der Form 1 _ _ _ _ _?

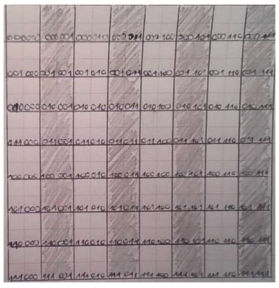
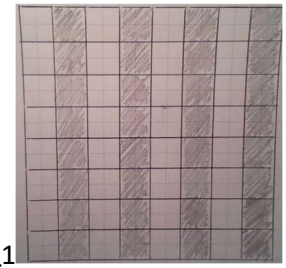
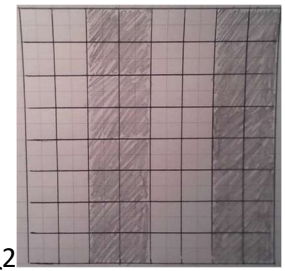
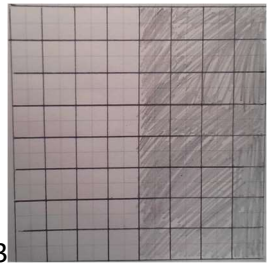


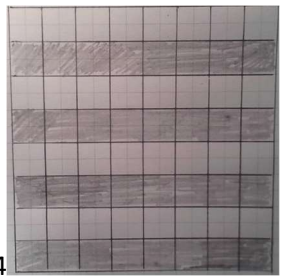
Bild: Schachbrett mit Binärziffern, wobei Q1 hervorgehoben ist.

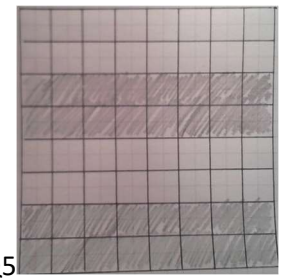
Analog alle Teilmengen Q1 bis Q6:

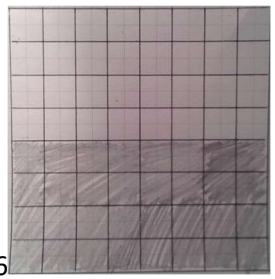
Q1 

Q2 

Q3 

Q4 

Q5 

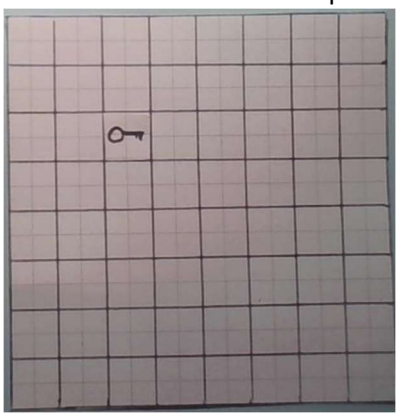
Q6 

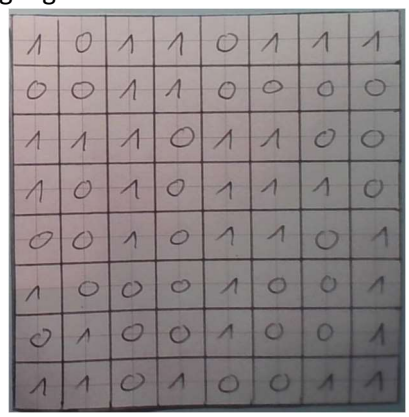
Die Strategie hinter dieser Konstruktion der Qs ist:
Jede Teilmenge sollte die Hälfte der Bits (32) beinhalten. Die Kreuzung von beliebigen zwei soll die Grösse halbieren, also 16 Positionen beinhalten. Eine Kreuzung von beliebigen drei sollte 8 Positionen beinhalten, eine Kreuzung von vier enthält 4 Positionen und von fünf 2 Positionen. Endziel ist, dass der Schnitt von 6 Gebieten aus den 12 möglichen Gebieten (6 gewählte Q1 bis Q6 und ihre Komplemente) immer genau eine Position enthält.

Beispiel 1: 000000 ist Position der Schnitte $\overline{Q6} \cap \overline{Q5} \cap \overline{Q4} \cap \overline{Q3} \cap \overline{Q2} \cap \overline{Q1}$.

Beispiel 2: 011101 ist Position der Schnitte $\overline{Q6} \cap Q5 \cap Q4 \cap Q3 \cap \overline{Q2} \cap Q1$.

Betrachten wir nun als Beispiel die Ausgangssituation:





So ergeben sich die Paritäten: Q1 ist 1, Q2 ist 0, Q3 ist 1, Q4 ist 1, Q5 ist 0, Q6 ist 1. Also:

Aktuell: 101101

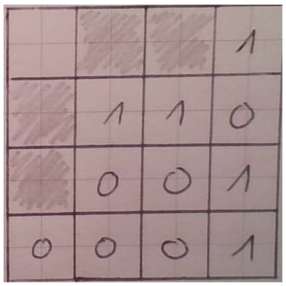
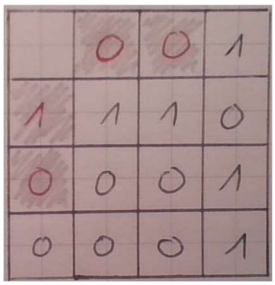
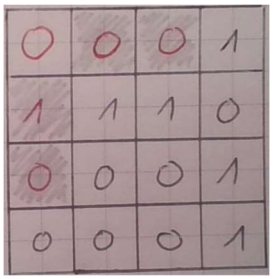
Wir müssen ändern: 111111 (weil xor bzw. mod 2, ergibt 1010 + 1000 = 0010)

Ziel: 010010

Der Helfer müsste also das Feld ganz unten rechts flippen. Der Magier wird dann die Paritäten 010010 ablesen und damit korrekt das Feld des Schlüssels identifizieren.

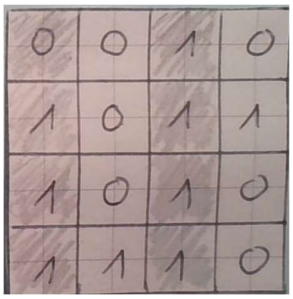
3) Auch dort galt: Wenn z.B. c1 umgeflippt wurde, dann hat die Fehlermeldung nur c1 betroffen.

4) Zuerst werden die 9 Nachrichtenbits in den 4x4-Codeblock eingetragen, dann bestimmt man für die Teilmengen Q1 bis Q4 die Paritätsbits und trägt diese ebenfalls ein, zuletzt bestimmt man das Paritätsbit für den ganzen 4x4-Codeblock.

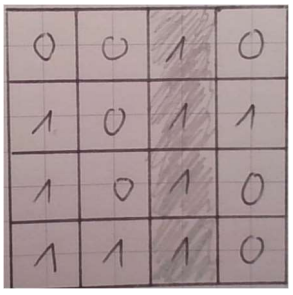




Hier ist wieder die Wahl der Kontrollpositionen wesentlich: Die Kontrollpositionen sind so gewählt, dass man die Werte für ihre Bits in beliebiger Reihenfolge berechnen kann, weil kein Inhalt einer Kontrollposition durch den Wert einer anderen Kontrollposition beeinflusst ist.

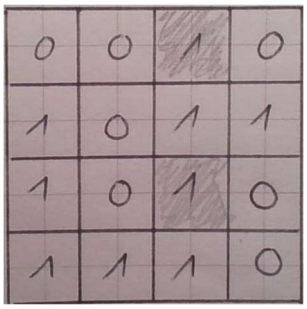
5) Für Q1 erhalten wir eine gerade Anzahl an Einsen, d.h.: Falls es genau einen Fehler gibt, dann ist er in den geraden Spalten.



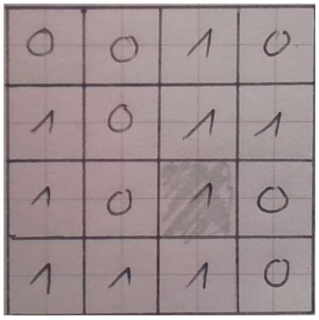
Für Q2 erhalten wir eine ungerade Anzahl an Einsen, d.h.: Es gibt mindestens einen Fehler in diesen beiden letzten Spalten.



Für Q3 erhalten wir eine gerade Anzahl an Einsen, d.h.: Falls es genau einen Fehler gibt, dann in den geraden Zeilen.



Für Q4 erhalten wir eine ungerade Anzahl an Einsen, d.h.: Es gibt mindestens einen Fehler in den letzten beiden Zeilen.



(Die gesamte Anzahl Nullen ist ungerade, das bestätigt: Es gibt also eine ungerade Anzahl Fehler, in diesem Fall gibt es, gemäss Annahme, genau einen Fehler.)

Der Fehler ist identifiziert, die korrekte Nachricht war: 00110001110