Hamming-Codes

Marc Schumann

6. Januar 2023

1. Vorwissen

- Daten als Zahlen/Bits
- Binärzahlen
- Parität

2. Zielsetzungen

In dieser Unterrichtseinheit soll die Hamming-Kodierung exemplarisch für fehlerkorrierende Kodierungen bearbeitet werden. Es soll versucht werden, diese Kodierung anschaulich den Schülerinnen und Schülern näher zu bringen. Die benötigte Zeit liegt bei ca. 3 bis 4 Lektionen.

Die grundsätzliche Idee hierzu beruht auf den Videos von Grant Sanderson (3Blue-1Brown) zu dem Thema:

- How to send a self-correcting message (Hamming codes): `https://www.youtube.com/watch?v=X8jsijhllIA`
- Hamming codes part 2, the elegance of it all `https://www.youtube.com/watch?v=b3NxrZOu_CE&t=663s`

Die Unterrichtseinheit bezieht sich hauptsächlich auf den Inhalt des ersten Videos. Das zweite geht auf die Eleganz des Algorithmus ein, die diese Arbeit zu umfangreich machen würde und weitergehende Programmierkenntnisse erfordert (List Comprehension).

Im folgenden Abschnitt wird der Ablauf der Unterrichtseinheit dargelegt. In Abschnitt 4 werden Aufgaben für die Schülerinnen und Schüler gegeben, in Abschnitt 5 die Lösungen dazu. Diese Die Arbeit schliesst mit einem Fazit.

Zusätzlich zu den fertigen Aufgaben liegt ein Python-Script bei, das in der jetzigen Form Hamming-Codes in der hier genutzten Matrix-Schreibweise generiert und Fehler einbaut. Dieses Script wurde bewusst nicht optimiert geschrieben, sondern gemäss des

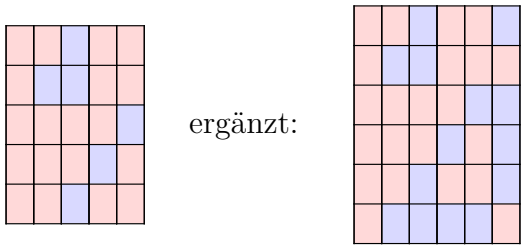
im anderen Teil der Fachdidaktik beworbenen Stils des modularen Entwurfs und der Parametrisierung. Die Ausgabe erfolgt als LaTeX-Code. Zur Benutzung benötigt man das Paket „nicematrix“.

3. Rätsel, Aufgaben, Beispiele, Projekte, Lernaufgaben

3.1. Einstieg

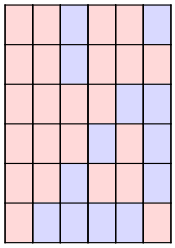
Als Einstieg könnte der vielen bekannte „Zaubertrick“ gewählt werden, bei dem ein Quadrat aus 5×5 Objekten mit zwei Kennzeichen (z.B. Münzen[1], Spielkarten aufgedeckt oder zugedeckt, schwarze und weisse Mühlesteine, oder selbst angefertigen Karten mit 0 und 1) gelegt wird. Der „Zauberer“ legt dann unter einem Vorwand weitere Elemente hinzu, so dass ein 6×6 grosses Quadrat entsteht. Dabei werden die hinzugefügten Karten so gedreht, dass die Parität der Zeile bzw. der Spalte, die ergänzt wird, gerade ist.[2]

Beispiel mit Farben:



Der Zauberer gibt die Anweisung eine Karte umzudrehen, dies aber erst dann zu tun, wenn er de Raum verlassen hat. Wenn er wieder hereinkommt, kann er die umgedrehte Karte daran erkennen, dass in der betroffenen Spalte und Zeile die Parität nicht mehr gerade ist. Die Karte unten rechts ist nur für die Parität in der letzten Spalte zuständig. Sie macht keine Aussage zur Parität der untersten Zeile.

Beispiel:



Die Parität ist in Zeile 2 (5 rote, 1 blaue Zelle) und Spalte 2 (5 rote, 1 blaue Zelle) verletzt. Somit ist die Zelle mit den Koordinaten $(2,2)$ geändert worden.

Daran anschliessend folgt eine kurze Besprechung des Vorgehens. Je nach Vorwissen kann dann das Prinzip der Parität erklärt oder wiederholt werden. Insbesondere sollte

[1] Münzen sind wahrscheinlich zu klein, allenfalls geeignet für Kleingruppen oder Gruppenarbeit.

[2] Hier wurde ein 5×5-Raster gewählt, daher ist immer eine Sorte ungerade und eine gerade und man kann dies schnell durch Hinzufügen eines weiteren Objekts der gleichen Art ausgleichen. Wenn man eine gerade Anzahl verwendet, muss man sich festlegen, welche Parität auf einen geraden Wert angepasst werden soll.

über die Notwendigkeit von Fehlerkorrekturen bei der Datenübertragung gesprochen werden.

Wichtige Begriffe:

Code-Wort Im Beispiel des Kartentricks wurde die Nachricht, die durch die jeweils oben liegende Seite der Karten/Objekte gegeben war, durch das Hinzufügen der zusätzlichen Karten in ein Code-Wort umgewandelt. Das hier verwendete Verfahren ist eine 1-Fehler korrigierende Kodierung, da durch die Kontrollbits genau ein Fehler korrigiert werden kann.

Kontrollbits Jede der Karten stellt ein Bit der Nachricht dar. Die hinzugefügten Bits nennt man Kontrollbits. Ihr Informationswert dient der Fehlererkennung und -korrektur. Die im Kartentrick hinzugefügten Karten kontrollieren jeweils die Korrektheit ihrer Zeile oder Spalte.

Länge der Nachricht Die Länge der Nachricht ist gegeben durch die Anzahl Bits, die übertragen werden sollen. Während der Kodierung werden Kontrollbits hinzugefügt.

Länge des Code-Wortes Die Anzahl Bits des Code-Wortes ergibt sich zusammen aus den Bits der Nachricht und den Kontrollbits. Dies ist die Länge des Code-Wortes.

Aufgabe 1: Nennen Sie einige Beispiele, bei denen bei der Datenübertragung Fehlerkorrekturen eingesetzt werden.

Aufgabe 2: Eine Möglichkeit der Fehlerkorrektur ist es, Daten mehrfach zu verschicken. Wie viele Kopien reichen aus? Begründen Sie Ihre Aussage.

Aufgabe 3: Stellen Sie einen Zusammenhang zwischen dem Zaubertrick und der Fehlerkorrektur her.

Nach diesen Vorüberlegungen kommen wir zurück auf den Zaubertrick. Dort wurden 11 Paritätsbits benötigt, um einen Fehler in 25 Datenbits zu finden.

Aufgabe 4: Vergleichen Sie die Länge der Code-Wörter bei dreifachen Kopien und bei dem Kartentrick. Bilden Sie die Quotienten aus der Anzahl der Kontrollbits und den Nachrichtenbits bei beiden Kodierungen. Gehen Sie von einer Nachrichtenlänge von k^2 Bits aus.

3.2. Richard Hamming

Richard Hamming war ein amerikanischer Mathematiker (1915–1998). Er hat während des Zweiten Weltkriegs am Manhattan-Projekt zum Bau der ersten Atombomben mitgearbeitet und wechselte anschliessend an die Bell Laboratorien. Dort hat er für seine Forschung die ersten Computer programmiert. Dies geschah damals noch mit Lochkarten. Dabei traten leicht Fehler beim Einlesen der Programme auf. Daher machte er sich Gedanken über eine effiziente Methode, Fehler automatisch zu korrigieren. Das System,

das er entwickelt hat, nennt man nun Hamming-Kodierung. Diese wird exemplarisch für fehlerkorrigierende Kodierungen in dieser Unterrichtseinheit das Hauptthema sein.

3.3. Hamming-Codes

Daten können als Aufreihung von Bits aufgefasst werden. Die Anordnung der Paritätsbits, die von Hamming erdacht wurde, soll an einem 4×4-Raster vorgestellt werden. Wir nummerieren die Zellen von 0 bis 15 durch:

0	1	2	3
4	5	8	7
8	9	10	11
12	13	14	15

Die Stellen, deren Nummer eine Zweierpotenz ist (hier also 1, 2, 4 und 8), reservieren wir für ein Paritätsbit. Die Zelle 0 ebenso. Es bleiben also von den 16 Bits noch 11 für Daten.

Wenn man die 11 Bits „10100101001“ in dieses Raster schreibt, so erhält man:

			1
	0	1	0
	0	1	0
1	0	0	1

Die Bits an den Stellen 1, 2, 4 und 8 werden nun auf eine bestimmte Weise ermittelt. Das Bit in Zelle 1 wird so gesetzt, dass die beiden Streifen der Spalten 2 und 4 eine gerade Parität besitzen.

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

Das Bit in Zelle 2 wird so gesetzt, dass der Block, der die letzten beiden Spalten beinhaltet, eine gerade Parität hat.

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

Diese beiden Bits genügen, um die fehlerhafte Spalte zu identifizieren. Angenommen wir ändern das Bit in Zelle 11 von einer 0 zu einer 1. Das Raster sieht dann so aus:

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

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

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

Man sieht, dass die Parität der Spalten als auch die Parität des Blocks ungerade ist. Daher ist der Fehler in der Schnittmenge der beiden rot markierten Bereiche zu orten, also in der letzten Spalte.

Aufgabe 5: Ändern Sie ein Bit in der 2. oder 3. Spalte und vergewissern Sie sich, dass man auch in diesen Fällen die fehlerhafte Spalte erkennen kann.

Wenn ein Fehler in der ersten Spalte auftritt, welche Parität haben die beiden Bereiche? Wie kann man in diesem Fall den Fehler erkennen?

Wenn man sich nun klargemacht hat, dass dieses Verfahren für die Spalten funktioniert, kann man für die Zeilen genauso vorgehen und so auch Fehler in der ersten Spalte erkennen (eine Ausnahme bleibt die Zelle 0).

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

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

Schliesslich wird noch der Wert der Zelle 0 so gesetzt, dass die Parität des gesamten Rasters gerade ist.

Das präparierte Datenpaket ist nun:

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

Das heisst, die Nachricht „10100101001“ wird mit dem Code-Wort: „1001101000101001“ kodiert.

Aufgabe 6: Erstellen Sie ein präpariertes Datenpaket aus den gegebenen Bits: „11100011100“.

Lösung:

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

Code-Wort: „0011011010011100“.

Aufgabe 7: Finden Sie den Fehler.

a)

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

b)

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

Was ist an diesem Beispiel speziell? Beachten Sie, welches Bit als falsch erkannt wird.

In Teil b) wird ein Paritätsbit als falsch erkannt.

Wir stellen fest, dass auch ein Fehler an einem Paritätsbit erkannt wird.

Ausserdem kann man bemerken, dass bei einem Fehler im Code-Wort auch das Kontrollbit an Position 0 falsch ist, da die Gesamtparität durch die Änderung eines einzelnen Bits von gerade auf ungerade geändert wird.

Aufgabe 8: Wie erkennt man einen falschen Wert im Bit 0? Z.B. hier:

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

Die Paritätsbits in Position 1, 2, 4 und 8 sind alle korrekt. Das heisst, entweder liegt kein Fehler vor oder er befindet sich in Position 0. Wie man leicht sieht, ist das Bit 0 in der Tat falsch, da die Gesamtparität ungerade ist.

Dies ist eine mögliche Information, die man aus dem Bit 0 ziehen kann. Ein weiterer wichtiger Punkt ist, dass man erkennen kann, ob 1 oder 2 Fehler vorliegen (unter der Annahme, dass höchstens 2 Fehler auftreten können). Wenn 1 Fehler vorliegt, wird das Bit 0 ebenfalls falsch sein. Bei zwei Fehlern wird es trotz des erkannten Fehlers richtig sein. Es folgt ein Beispiel mit zwei Fehlern:

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

Als Aufgabe formuliert folgt daraus die Frage:

Aufgabe 9: Bisher haben Sie einen Fehler erkannt. Das folgende Code-Wort hat zwei Fehler. Woran erkennen Sie dies?

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

Es ist aber nur ein Paritätsbit falsch (das markierte). Wenn es nur einen Fehler gäbe, wäre dieser genau dieses Bit. Aber Bit 0 zeigt keinen Fehler an, d.h., es gibt eine gerade Anzahl Fehler. Da aber mindestens ein Fehler vorliegt, sind es also 2 (oder mehrere). Das Bit 0 lässt uns also unterscheiden, ob ein oder zwei Fehler vorliegen.

3.4. Wahl der Paritätsbits

Es können also Fehler in einem der 11 bzw. 16 Bits erkannt werden. Die Wahl der Position der Paritätsbits und der Regionen, die sie kontrollieren, scheint momentan aber recht willkürlich gewählt.

Aufgabe 10: Vergleichen Sie die Nummerierung der einzelnen Zellen in Binärdarstellung in den einzelnen Streifen und Blöcken, die bei der Hamming-Kodierung verwendet wurden. Was gilt insbesondere für die Paritätsbits?

0	1	2	3
4	5	8	7
8	9	10	11
12	13	14	15

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Wir stellen fest, dass die Paritätsbits an den Stellen stehen, an denen nur ein einziges Bit gesetzt ist. Genau deswegen stehen sie an den Positionen, deren Nummer eine Zweierpotenz ist ($0001_2 = 2^0_{10}$, $0010_2 = 2^1_{10}$ usw.).

Die Zellen, die zum Paritätsbit in Position 1 gehören, haben alle eine Nummer, deren hinterstes Bit gesetzt ist, genau wie beim zugehörigen Paritätsbit, und bilden hier die 2. und 4. Spalte. Bei allen Zellennummern im Block, der zum zweiten Paritätsbit gehört, ist das zweitletzte Bit gesetzt. Das gilt genauso für Bit 3 und 4 in den entsprechenden Zeilen.

Dies lässt sich leicht erweitern. Wenn wir eine Matrix von 8×8 Bits erstellen, können wir die Zellen von 0 bis 63 durchnummerieren. Die Zellen mit den Nummern 1, 2, 4, 8, 16 und 32 nutzen wir als Paritätsbit für die entsprechenden Spalten und Blöcke. Zusammen mit dem Bit 0 benötigen wir so 7 Bits zur Fehlerkorrektur von 57 Bits. Bei 16×16 Bits benötigt man 9 Paritätsbits zur Korrektur von 247 Bits.

0	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	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

0	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	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Aufgabe 11: Zeigen Sie: Es werden $\log_2(n) + 1$ Kontrollbits benötigt, wenn man ein Code-Wort der Länge $n = 2^k$ Bits erstellt.

Da die Kontrollbits an den Stellen stehen, an denen nur ein Bit in der Nummer der Zelle gesetzt ist (Zweierpotenzen), benötigen wir für 2^k Bits genau k solche Kontrollbits und zusätzlich 1 Bit für die Gesamtparität. Also sind es $k+1 = \log_2(2^k)+1 = \log_2(n)+1$.

3.5. Der Clou der Wahl der Kontrollbits

Einerseits benötigen wir deutlich weniger Kontrollbits als bei der zeilen- und spaltenweisen Kodierung des Zaubertricks. Die falsche Zelle ist aber auch sofort identifiziert, wenn man die Nummern der Kontrollbits betrachtet, die den Fehler anzeigen. Beispiel:

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

Zellnummern:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

oder

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

Zellnummern:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Aufgabe 12: Wie hängt die Zellennummer der falschen Zelle mit den Zellennummern der fehlerhaften Kontrollbits zusammen? Können Sie den Zusammenhang erklären?

Die Summe der Zellennummern ergibt die Position des Fehlers. Das liegt daran, dass die Fehler jeweils angeben, welches Bit (bei der Angabe der Zellennummer mit Basis 2) in der fehlerhaften Zelle gesetzt ist. Wenn man alle Bits zusammenfügt, erhält man die Nummer der Zelle. Dies ist das Gleiche in Basis 10, wenn man die Zellennummern der Kontrollbits addiert.

4. Arbeitsblätter für die Klasse mit verständlichen Aufträgen (Rätsel, Aufgaben)

Die Arbeitsblätter liegen als separate Datei vor, damit die Nummerierung und Seitenzahlen unabhängig von diesem Dokument bleiben. Es werden die Aufgaben aus dem vorherigen Teil aufgegriffen.

5. Lösungen/Erklärungen

Auch die Lösungen liegen als separate Datei vor. Aufgaben und Lösungen können den Schülerinnen und Schülern für das Selbststudium abgegeben werden.

6. Fazit

Wenn man sich das Thema Hamming-Codes anschaut, sieht es zuerst sehr unübersichtlich aus. Diese Unterrichtseinheit soll den Schülerinnen und Schülern mittels einer geeigneten Darstellung dieses Thema anschaulich näherbringen. Es werden kurz Aspekte berührt wie die Skalierung der Anzahl der Kontrollbits mit dem Logarithmus. Ein weiterer Schritt wäre das Programmieren eines geeigneten Algorithmus zur Kodierung und Dekodierung von Nachrichten sowie der Fehlerkorrektur bzw. -erkennung von 1 und 2 Bit-Fehlern.

A. Automatisch erzeugte Aufgaben

a)

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

b)

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

c)

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

d)

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

e)

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

f)

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

g)

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

h)

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

B. Lösungen zu den Aufgaben

a)

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

b)

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

c)

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

d)

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

e)

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

f)

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

g)

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

h)

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