Selbstkorrigierende Kodierung im Ternärsystem

Einführung

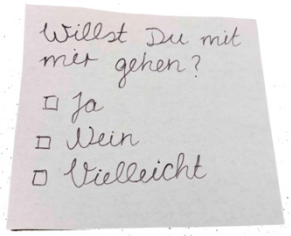
Wir sind es uns bisher gewohnt, Informationen im Binär- oder in davon abgeleiteten Stellenwertsystemen (wie bspw. das Hexadezimalsystem) darzustellen. Dabei kann es manchmal durchaus sinnvoll sein, ein Zahlensystem zu wählen, dem eine ungerade Basis zugrunde liegt!

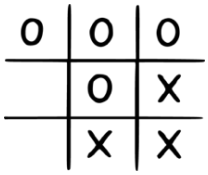
Ein Beispiel für solch ein Stellensystem ist das «**Ternärsystem**» (auch Trinär-, Dreier- oder 3-adisches System genannt), das die **Basis 3** verwendet. D.h. eine einzelne, ternäre Ziffer codiert **drei verschiedene Zustände**.

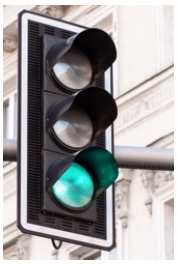
In dieser Unterrichtseinheit wird es darum gehen, wie man geschickt selbstkorrigierenden Code für das Ternärsystem gestalten kann. Bevor wir uns aber hierzu genauer Gedanken machen, werden wir zunächst das Ternärsystem selbst etwas näher kennenlernen.

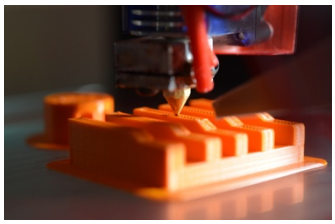
Aufgabe

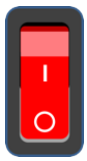
Betrachten Sie die nachfolgenden Bilder. Für welche Beispiele erachten Sie den Einsatz des Ternärsystems als geeignet?











Überlegen Sie sich ein eigenes Beispiel, für welches man Information sinnvollerweise ternär darstellen könnte:

Die Funktionsweise des Ternärsystems

Bei den Dezimalzahlen haben wir zehn Ziffern (`0` bis `9`) zur Verfügung, bei den Binärzahlen nur deren zwei (`0` und `1`). Beim Ternärsystem, das auf der Basis 3 beruht, verwenden wir **drei Ziffern**, üblicherweise:

`0, 1, 2`

Für gewisse Anwendungen bevorzugt man die Ziffern -1, 0, 1, die wir aber hier nicht weiter benutzen werden. Eine einzelne, ternäre Ziffer wird – ganz analog zum Bit – ein «**Trit**» genannt, und sechs Trits ergeben – analog zum Byte – ein «**Tryte**».

Um zu verstehen, wie das Ternärsystem genau funktioniert, schauen wir uns am besten ein Beispiel an. Wir werden feststellen, dass es sich um ein ganz gewöhnliches Stellenwertsystem handelt und deshalb völlig analog zum Dezimal- bzw. zum Binärsystem aufgebaut ist – nur, dass wir hier mit 3er-Potenzen statt mit 10-er bzw. 2er Potenzen arbeiten.

Beispiel

① Betrachten wir die ternäre Zahl **1210** und schreiben (von rechts nach links in aufsteigender Reihenfolge) die Potenzen der Basis (d.h. 3^0, 3^1, 3^2, 3^3, ...) daneben. Dann ergänzen wir die Faktoren 0 bis 2 vor den Dreierpotenzen – je nachdem, was im Ternärcode angegeben ist.

② Als nächstes rechnen wir alle Dreierpotenzen explizit aus.

③ Zu guter Letzt rechnen wir alles zusammen, unter Berücksichtigung der Faktoren. Wir erhalten als Ergebnis die Zahl in Dezimaldarstellung – in diesem Beispiel wäre das **48**.

①	1210_3 =	1 * 3^3 +	2 * 3^2 +	1 * 3^1 +	0 * 3^0
②		1 * 27 +	2 * 9 +	1 * 3 +	0 * 1
③	48_{10} =	27 +	18 +	3 +	0

Möchte man umgekehrt eine Dezimalzahl in eine Ternärzahl umwandeln, so muss man den dezimalen Wert mit Scheinen à Dreierpotenzen (1, 3, 9, 81 etc.) «bezahlen» und dabei möglichst wenig Scheine verwenden. Algorithmisch kann man das so bewerkstelligen: Man teilt die Dezimalzahl durch die Basis 3 und notiert sich jeweils den Rest. Man fährt damit so lange fort, bis der Quotient 0 ergibt, dann stoppt man. Die Reste ergeben – rückwärts gelesen – die ternäre Darstellung der Zahl.

Beispiele

Dezimalzahl:	48_{10}	*Dezimalzahl:*	121_{10}
48 / 3 = 16	Rest: 0	121 / 3 = 40	Rest: 1
16 / 3 = 5	Rest: 1	40 / 3 = 13	Rest: 1
5 / 3 = 1	Rest: 2	13 / 3 = 4	Rest: 1
1 / 3 = 0	Rest: 1	4 / 3 = 1	Rest: 1
		1 / 3 = 0	Rest: 1
Ternärzahl:	1210_3	*Ternärzahl:*	11111_3

Ein paar kleine Übungsaufgaben

1) Für welche Dezimalzahlen stehen die folgenden ternären Codes?

a) 20_3

b) 1120_3

c) 10110_3

d) 212121_3

2) Rechnen Sie die folgenden Dezimalzahlen in Ternärzahlen um:

a) 16_{10}

b) 98_{10}

c) 111_{10}

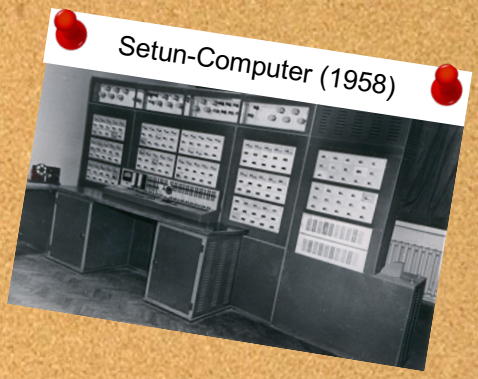
d) 2021_{10}

Kurzlösungen
1a) 6 b) 42 c) 93 d) 637 2a) 121 b) 10122 c) 11010 d) 2202212

Funfacts zum Ternärsystem

- Das ternäre Zahlensystem weist die höchste «Informationsdichte» unter den ganzzahlig basierten Systemen auf. Das bedeutet, dass man im Dreiersystem Zahlen am effizientesten (= platzsparendsten) speichern kann, weil man damit das bestmögliche Verhältnis zwischen Anzahl Stellen und Anzahl verschiedener Ziffern pro Stelle erreicht.

- Tatsächlich wurde in den 1960er-Jahren in der Sowjetunion ein Computer gebaut, der auf dem ternären System (mit den Ziffern -1, 0, 1) basierte: der Setun-Computer. Dieser wurde für Lehr- und wissenschaftliche Zwecke eingesetzt. Allerdings war bereits anfangs der 1970er-Jahre die binäre Technologie so weit fortgeschritten, dass man das ternäre System effizient auf binären Rechnern emulieren konnte. Die Entwicklung und Produktion ternärer Rechner wurde deshalb eingestellt.



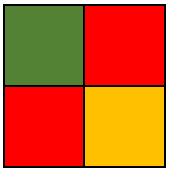
Bildquelle: Earl T. Campbell: «The Setun Computer»
https://earltcampbell.com/2014/12/29/the-setun-computer/ (abgerufen am 18.1.22)

Selbstkorrigierender Code im Ternärsystem

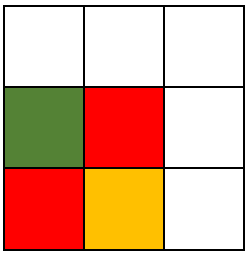
Wir haben verstanden, wie das Ternärsystem aufgebaut ist. Jetzt geht es darum, herauszufinden. mit welchen Kniffen man ternären Code erstellen kann, bei dem sich Fehler, die z.B. bei der Übertragung passieren, eruieren und ggf. korrigieren lassen.

Lernaufgabe 1: Ein Kartentrick mit drei Farben

Zwei Magier führen einen Kartentrick vor. Um den Zaubertrick vorzuführen, benötigen sie Karten mit drei verschiedenen Farben: grün, gelb und rot. Ein Magier verlässt den Raum und sein Kollege fordert einen Freiwilligen dazu auf, vier beliebige Karten auszuwählen und damit ein 2x2-Quadrat zu bilden. Der verbleibende Magier fügt anschliessend 5 weitere Karten hinzu, um das 2x2-Quadrat zu einem 3x3-Quadrat zu ergänzen.



2x2-Quadrat des Freiwilligen



Der Magier ergänzt die 5 leeren Felder

Der Freiwillige wird dann aufgefordert, eine der neun Karten durch eine Karte anderer Farbe zu ersetzen. Der Magier, der den Raum zuvor verlassen hat, wird wieder zurückgerufen. Er wird jetzt in jedem Fall genau bestimmen können, welche Karte verändert wurde und welche Farbe die ursprüngliche Karte hatte.

a) Überlegen Sie sich eine Strategie, wie der im Raum verbliebene Magier die 5 Karten legen muss, so dass sein Komplize in jedem Fall die vertauschte Karte finden und korrigieren kann.

b) Der Zaubertrick kann auch durchgeführt werden, wenn der Freiwillige ein 5x5-Quadrat legt, dass anschliessend vom Magier zu einem 6x6-Quadrat ergänzt wird. Wie muss der Magier im Raum in diesem Fall die Karten legen, damit sein Komplize in jedem Fall den Fehler findet?

c) Der Trick ist so aufgebaut, dass der Magier in jedem Fall **einen** Fehler erkennen **und** korrigieren kann. Vielleicht ist aber noch mehr möglich! Finden Sie die jeweils grösste Anzahl an Karten, die verändert werden können, damit der Magier...

   i) ... erkennen kann, dass so viele Karten vertauscht wurden.

   ii) ... zusätzlich erkennen kann, wo die vertauschen Karten liegen und welche Farben sie hatten.

d) Beweisen Sie, dass der Magier immer eine passende Karte in der Ecke des 3x3- bzw. 6x6-Quadrats anfügen kann, ohne die in a) und b) definierten Regeln zu verletzen.

e) Die Regeln in a) und b) gelten nur für Rechtecke mit speziellen Seitenlängen. Können Sie eine Regel finden, wie der Magier die Karten für ein beliebiges Quadrat mit Seitenlängen a und b legen muss, damit sein Komplize den Fehler erkennt?

Lernaufgabe 2: Effiziente 1-korrigierende Codes für Trinärnachrichten

Für ein Binärwort der Länge 4 haben kennen wir bereits eine effiziente Methode, durch Hinzufügen von 4 Prüfbits den Code 1-Fehlerkorrigierend zu machen. Diese Methode kann mit einer Hamming-Matrix dargestellt werden.

c1	c2	
a1	a2	c3
a3	a4	c4

Tabellendarstellung

	a1	a2	a3	a4	c1	c2	c3	c4
c1	x		x		x			
c2		x		x		x		
c3	x	x					x	
c4			x	x				x

Hamming-Matrix

Die Spalten der Matrix beschreiben, wie sich die Prüfbits verändern, wenn eines der Nachrichtenbits verändert wird. Bei Trinärwörtern reicht es aber nicht aus, das Bit zu finden, das verändert wurde. Mit einem 1-Fehlerkorrigierenden Code wollen wir auch erkennen können, welchen Wert das fehlerhafte Bit ursprünglich hatte.

a) Beschreiben Sie eine Methode, mit der Sie anhand der veränderten Prüfbits erkennen können, welches Nachrichtenbit eines Trinärworts verändert wurden **und** welchen Wert es vor der Veränderung hatte.

b) Statt ein neues System für Trinärzahlen zu entwickeln könnte man auch alle Trinärzahlen in zweistellige Binärzahlen umwandeln und die bekannten Codierungsverfahren darauf anwenden. Begründen Sie, warum die in a) entwickelte Hamming-Matrix für Trinärcodes weniger Speicherplatz einnimmt als eine entsprechende Hamming-Matrix für zweistellige Binärzahlen.

c) Ist es möglich mit weniger als vier Prüfbits auszukommen? Begründen Sie Ihre Antwort und geben Sie gegebenenfalls die Hamming-Matrix an.

Lernaufgabe 3: Visualisierung der Abstände mit drei Zuständen (Distanzmatrix)

a) Zeichnen Sie einen möglichst übersichtlichen eindimensionalen Hyperwürfel mit drei Zuständen (0 , 1 , 2).
(D.h. listen Sie alle möglichen Folgen von (0 , 1 , 2) der Länge 1 auf und verbinden Sie alle, welche Abstand 1 haben mit einer Kante.)

b) Erweitern Sie Ihren eindimensionalen Hyperwürfel von Aufgabe 2a) zu einem möglichst übersichtlichen zweidimensionalen Hyperwürfel.
(D.h. listen Sie alle möglichen Folgen von (0 , 1 , 2) der Länge 2 auf und verbinden Sie alle, welche Abstand 1 haben mit einer Kante.)
Tipp: Ergänzen Sie die Folgen von a) mit jeweils 0 , 1 oder 2 . Sie können dazu Ihren eindimensionalen Hyperwürfel also zuerst dreimal zeichnen und dann ergänzen. So erhalten Sie alle möglichen Folgen von der Länge 2. Verbinden Sie nun alle Folgen, welche Abstand 1 haben.

c) Aus wie vielen Bitfolgen und Kanten besteht der dreidimensionale Hyperwürfel mit drei Zuständen?
Zusatz: Wie viele Kanten hat ein hat der n-dimensionale Hyperwürfel mit drei Zuständen?

d) *Knobelaufgabe: Wie könnte der dreidimensionale Hyperwürfel übersichtlich dargestellt werden?