Aufgabe 3 - Zara Zackigs Zurueckkehr

Teilnahme-ID: 60302

Bearbeitet von
Florian Bange

13. April 2022

Inhaltsverzeichnis

1 Definierung dex XORs

XOR bzw. $\oplus$ sei zunaechst auf zwei Bits/Wahrheitswerte, wie folgt ueber die Gleichheit, definiert:

Sein a, b zwei Wahrheitswerte.

$$a\ XOR\ b \iff a \oplus b = \neg(a \iff b)$$

Die dazugehoerige Wahrheitstabelle sieht wie folgt aus:

a	b	a $\iff b$	$\neg(a \iff b)$	a XOR b
0	0	1	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

Bzw. sieht die Verknuepfungstabelle des XORs so aus:

$\oplus$	0	1
0	0	1
1	1	0

2 Darstellung durch $\mathbb{Z}_2$ mit der Addition

Die Menge $\mathbb{Z}_2 = \{\ 0, 1\ \}$ bilded zusammen mit der Addition eine abelsche Gruppe.
Die dazugehoerige Verknuepfungstabelle sieht wie folgt aus:

+	0	1
0	0	1
1	1	0

Wie zu erkennen ist, ist diese Verknuepfungstabelle identisch zu der des XORs.
Somit kann die Verknuepfung XOR mit der Addition in $\mathbb{Z}_2$ dargestellt werden.

3 Eigenschaften des XORs

Aufgrund dessen, dass das XOR mit der abelschen Gruppe ($\mathbb{Z}_2$, +) dargestellt werden kann, gelten fuer das XOR die gleichen Eigenschaften, wie fuer die abelsche Gruppe ($\mathbb{Z}_2$, +):

Sein a, b, c beliebige Wahrheitwerte (0, oder 1), bzw. $a, b, c \in \mathbb{Z}_2$.

1. Assoziativitaet: (a $\oplus$ b) $\oplus$ c = a $\oplus$ (b $\oplus$ c)
2. Neutrales Element 0: a $\oplus$ 0 = a
3. Selbstinvers: a $\oplus$ a = 0
4. Kommutativitaet: a $\oplus$ b = b $\oplus$ a

Weiter kann man aus der Darstellung des XORs durch ($\mathbb{Z}_2$, +) Folgende Erkenntniss machen:
Da $\mathbb{Z}_2$ equivalent zu den Restklassen modulo 2 ist, kann man das XOR ebenfalls in $\mathbb{Z}$ darstellen, indem man die Bits normal addiert und dann Modulo 2 rechnet:
Fuer $a, b, c \in \mathbb{Z}_2$,

$$a \oplus b \oplus c = (a + b + c)\ mod\ 2$$

4 XOR auf Bitfolgen

Wird das XOR auf mehrere aneinandergereite Bits, sprich Bitfolgen, benutzt, so wird das bereits eingefuehrte XOR elementweise wie folgt angewendet:
Fuer die Bits aller Bitfolgen an Stelle i, wird nacheinander das XOR angewendet.Dies wird fuer alle Stellen i der Bitfolgen durchgefuerht. Das jeweilige Ergebniss wird in der resultierenden Bitfolge an Stelle i notiert. Dafuer muessen alle mit dem XOR verbundenen Bitfolgen die gleiche Laenge haben.

Dies sieht allgemein wie folgt aus:
Fuer n Bitfolgen mit je m Bits, wobei $a_{i,1}$, ... , $a_{i,m}$ die Bits der i-ten Bitfolge sind.

$$a_{1,1} \oplus \cdots \oplus a_{n,1} = b_1$$

$$\dots$$

$$a_{1,m} \oplus \dots \oplus a_{n,m} = b_m$$

b_1 , ... , b_m sind die m Bits der resultierenden Bitfolge.

Beispiel:
Moechte man folgende Bitfolgen mit dem XOR verbinden, geht dies, wie anschlieszend in der Tabelle gezeigt.

$$1001, 1100, 1101$$

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

Die zuvor beschriebene Vorgehensweise sieht fuer das Beispiel wie folgt aus:

$$1 \oplus 1 \oplus 1 = 1$$

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

$$0 \oplus 0 \oplus 0 = 0$$

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

Weiter sei angemerkt, dass fuer das XOR mit Bitfolgen die gleichen Eigenschaften gelten, wie bei einzelnen Bits, da das XOR fuer einzelene Bits elementweise angewendet wird.

5 Umformung der Aufgabe

Das Ziel der viertel Aufgabe des Bundeswettbewerbs Informatik 2022 laesst sich wie folgt formal definieren:
Fuer n Bitfolgen der Laenge m sind gesucht k Bitfolgen s_1, ... , s_k, fuer welche eine weitere Bitfolge x exestiert, mit

$$s_1 \oplus \dots \oplus s_k = x.$$

Formt man die Gleichung um, indem man zu beiden Seiten $\oplus\, x$ hinzufuegt, erhaelt man

$$(s_1 \oplus \dots \oplus s_k) \oplus x = x \oplus x.$$

Durch die Eigenschaft des Selbstinversen, erhaelt man

$$(s_1 \oplus \dots \oplus s_k) \oplus x = 0,$$

wobei 0 fuer die Bitfolge bestehend aus m Nullen steht.
Weiter erhaelt man mit der Assoziativitaet

$$s_1 \oplus \dots \oplus s_k \oplus x = 0.$$

Nun ist das Ziel also, die $k+1$ der n Bitfolgen s_1, ... s_k , s_{k+1} zu finden, bei welchen gilt

$$s_1 \oplus \dots \oplus s_k \oplus s_{k+1} = 0.$$

Diese $k+1$ Bitfolgen stellen eine valide Loesung dar.

6 Loesung durch ein Gleichungssystem

Fuer das zuvor beschriebene Problem werden nun n Entscheidungsvariablen eingefuehrt: x_1 , ... , x_n.
Diese koennen entweder 0 oder 1 annehmen. Ist die Entscheidungsvariable x_i mit $1 \leq i \leq n$ 1, so ist die i-te Bitfolge Teil der Loesung, andernfalls nicht.

Fuer eine gueltige Loesung muss folgendes Gleichungssystem mit m Gleichungen in $\mathbb{Z}_2$ geloest werden:

$$a_{1,1} * x_1 + \dots + a_{n,1} * x_n = 0$$

$$\dots$$

$$a_{1,m} * x_1 + \dots + a_{n,m} * x_n = 0$$

Dabei stellen

$$a_{i,1} \ , \ \dots \ , \ a_{i,m}$$

die m Bits der i-ten Bitfolge dar.
Die Bits der gegebenen Bitfolgen werden also vertikal untereinander geschrieben und die Bitfolgen horizontal nebeneinander. Dabei erhaelt jede Spalte, also jede Bitfolge, eine Entscheidungsvariable.
Wodurch m - Anzahl der Bits - Reihen und n - Anzahl an Bitfolgen - Spalten entstehen.

Weiter muss

$$x_1 + \dots + x_n = k + 1$$

in $\mathbb{Z}$ (nicht in $\mathbb{Z}_2$!) gelten.
Dadurch ist gegeben, dass exakt die benoetigten Anzahl an $k+1$ Bitfolgen gewaehlt werden.

Dass eine Loesung fuer die zuvor beschriebenen Gleichungen ebenfalls eine Loesung fuer das Grundlegene Problem ist, ist einfach zu zeigen:

Sein ohne Einschraenkung der Allgemeinheit $x_1 \ , \ \dots \ , \ x_{k+1}$ die $k+1$ Entscheidungsvariablen, welche 1 annehmen und die Gleichungen erfuellen.
Nun lassen sich die zuvor beschriebenen Gleichungen je auf $k+1$ Summanden reduzieren, welche zusammen 0 in $\mathbb{Z}_2$ ergeben.
Diese Gleichungen sehen nun wie folgt aus:

$$a_{1,1} + \dots + a_{k+1,1} = 0$$

$$\dots$$

$$a_{1,m} + \dots + a_{k+1,m} = 0$$

Jede Gleichung ist (wie in Punkt 2 beschrieben) equivalent zum XOR angewandt auf mehrere Bits:

$$a_{1,1} \oplus \dots \oplus a_{k+1,1} = 0$$

$$\dots$$

$$a_{1,m} \oplus \dots \oplus a_{k+1,m} = 0$$

Wie in Punkt 4 beschrieben wurde ist dies wiederrum gleichwertig zum XOR auf Bitfolgen. Hier bei den Bitfolgen 1 bis $k+1$, welche zusammen die Bitfolge bestehend aus m Nullen ergeben.
Somit wurden $k+1$ Bitfolgen gefunden, welche, verknuepft durch das XOR, die Bitfolge bestehend aus m Nullen ergeben.

7 Loesen des Gleichungssystems

Nun ist die Aufgabe folgende Gleichungen in $\mathbb{Z}_2$:

$$a_{1,1} * x_1 + \dots + a_{n,1} * x_n = 0$$

$$\dots$$

$$a_{1,m} * x_1 + \dots + a_{n,m} * x_n = 0$$

und diese Gleichung in $\mathbb{Z}$:

$$x_1 + \dots + x_n = k + 1$$

zu loesen.

Mein Ansatz besteht daraus, zu naechst das Gleichungssystem in $\mathbb{Z}_2$ zu loesen und anschlieszend nach einer Loesung der Loesungsmenge zu suchen fuer welche die letzte Gleichung gilt.

7.1 Loesen des Gleichungssystems in $\mathbb{Z}_2$

Die zu loesende Gleichungen lassen sich wie folgt mit Matrix und Vektoren darstellen:

$$A * \vec{x} = \vec{0}$$

mit

$$A = \begin{bmatrix} a_{1,1} & \dots & a_{n,1} \\ \vdots & & \vdots \\ a_{1,m} & \dots & a_{n,m} \end{bmatrix}$$

und

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

Es ist also das homogene System zu A in $\mathbb{Z}_2$ zu loesen.

Aufgrund dessen, dass in $\mathbb{Z}_2$ ein Koerper ist, laesst sich zum loesen des Gleichungssystems in $\mathbb{Z}_2$ das Gausz-Verfahren verwenden.
Dadurch erhaelt man eine Loesungsmenge, welche wie folgt aussieht:

$$\{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \vec{v_1} * x_1 + \cdots + \vec{v_n} * x_n\}$$

Falls x_i mit $1 \leq i \leq n$ keine freie Variable ist, ist $\vec{v_1} = \vec{0}$.

7.2 Loesen der letzten Gleichung

Zu letzt ist die Loesung der Loesungsmenge zu finden, welche die Gleichung

$$x_1 + \cdots + x_n = k + 1$$

in $\mathbb{Z}_2$ erfuellt.

Diese Gleichung laesst sich mit der zuvor beschriebenen Loesungsmenge wie folgt umformen:

$$\begin{aligned}
& x_1 + \cdots + x_n = k + 1 \\
\Leftrightarrow\ & [(v_{1,1} * x_1 + \cdots + v_{n,1} * x_n)\ mod\ 2] + \cdots + [(v_{1,n} * x_1 + \cdots + v_{n,n} * x_n)\ mod\ 2] = k + 1 \\
\Leftrightarrow\ & ([(v_{1,1} * x_1 + \cdots + v_{n,1} * x_n)\ mod\ 2] + \cdots + [(v_{1,n} * x_1 + \cdots + v_{n,n} * x_n)\ mod\ 2])\ mod\ 2 = (k+1)\ mod\ 2 \\
\Leftrightarrow\ & [(v_{1,1} * x_1 + \cdots + v_{n,1} * x_n) + \cdots + (v_{1,n} * x_1 + \cdots + v_{n,n} * x_n)]\ mod\ 2 = (k+1)\ mod\ 2 \\
\Leftrightarrow\ & [x_1 * (v_{1,1} + \cdots + v_{1,n}) + \cdots + x_n * (v_{n,1} + \cdots + v_{n,n})]\ mod\ 2 = (k+1)\ mod\ 2 \\
\Leftrightarrow\ & ([x_1 * (v_{1,1} + \cdots + v_{1,n})]\ mod\ 2 + \cdots + [x_n * (v_{n,1} + \cdots + v_{n,n})]\ mod\ 2)\ mod\ 2 = (k+1)\ mod\ 2 \\
\Leftrightarrow\ & ([x_1 * ((v_{1,1} + \cdots + v_{1,n})\ mod\ 2)] + \cdots + [x_n * ((v_{n,1} + \cdots + v_{n,n})\ mod\ 2)])\ mod\ 2 = (k+1)\ mod\ 2 \\
\Leftrightarrow\ & [x_1 * ((v_{1,1} + \cdots + v_{1,n})\ mod\ 2)] + \cdots + [x_n * ((v_{n,1} + \cdots + v_{n,n})\ mod\ 2)] \equiv_2 k + 1
\end{aligned}$$

Dabei ist

$$\vec{v_i} = \begin{bmatrix} v_{i,1} \\ \vdots \\ v_{i,n} \end{bmatrix}.$$

Nun kann man fuer alle $\vec{v_i}$ (mit $1 \leq i \leq n$)

$$m_i = (v_{i,1} + \cdots + v_{i,n})\ mod\ 2$$

berechnen.

Nun muss die Gleichung

$$\Leftrightarrow x_1 * m_1 + \cdots + x_n * m_n \equiv_2 k + 1$$

geloest werden, bzw.

$$\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = k + 1$$

in $\mathbb{Z}_2$.

8 Implementierung

Zum loesen des Gleichungssystem in $\mathbb{Z}_2$ wird XXXX benutzt. Zur finden der korrekten Loesung in der Loesungmenge wird XXXX verwendet.

9 Laufzeitanalyse

Die laufzeit des Programm besteht aus zwei Teilen.

1. Loesen des Gleichungssystem
2. Finden der korrekten Loesung in der Loesungsmenge

10 Aufgabenteil c - Beispiele

11 Aufgabenteil b

In Aufgabenteil c ist gefragt, wie man mithilfe der 11 gefundenen Karten am naechsten Wochenende das naechste Haus aufsperren kann, ohne dafuer mehr als zwei fehlversuche zu benoetigen.

Sein zunaechst $w_1, \ldots, w_10$ die Karten der in der Aufgabenstellung erwaehnten Codeworte und x das aus ihnen resultierende XOR.
Weiter sein die gefundenen Karten $k_1, \ldots, k_11$. Diese Karten muessen offensichtlich aus den Karten $w_1, \ldots, w_10$ sowie x bestehen.

Leider ist es nicht moeglich zu wissen, welche der 11 gefundenen Karten das xor ist.

Allerdings kann man die