Aufgabe 3 - Zara Zackigs Zurueckkehr

Teilnahme-ID: 60302

Bearbeitet von
Florian Bange

23. 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:

4 Bitfolgen

Bitfolgen bestehen aus mehren aneinandergereihten Bits.
Eine Bitfolge b aus m Bits besteht aus den Bits $b_1, \ \dots\ , \ b_m$ und laesst sich entweder darstellen als aneinandergereihten Bits:

$$b_1\ \dots\ b_m$$

Oder als Vektor:

$$b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Wird das XOR auf mehrere Bitfolgen angewandt, wird auf die Bits aller Bitfolgen an Stelle i nacheinander das XOR angewendet. Dies wird fuer alle Stellen i der Bitfolgen durchgefuerht. Das jeweilige Ergebnis wird in der resultierenden Bitfolge an Stelle i notiert. Dafuer muessen alle mit dem XOR verbundenen Bitfolgen die gleiche Laenge haben.

Am sinnvollsten ist dazu die Darstellung der Bitfolgen als Vektoren zusammen mit der Vektoraddition. Dies sieht allgemein wie folgt aus:
Fuer n Bitfolgen $b_1, , \dots , b_n$ mit je m Bits, wobei

$$b_i = \begin{bmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \end{bmatrix}$$

ist, ist die entstehende Bitfolge u mit

$$u = b_1 + \cdots + b_n,$$

wobei + hier die Vektoraddition ist.

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:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

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 Reduzierung der Aufgabe

Das Problem der Bonusaufgabe „Zara Zackigs Zurueckkehr“ (oder kurz das ZZZ-Problem) laesst sich wie folgt formal definieren:
Gegeben sind n Bitfolgen der Laenge m, und eine Ganzzahl $k > 0$.
Fuer die n Bitfolgen ist gesucht die Menge an k Bitfolgen $\{s_1, \dots , s_k\}$, fuer welche eine weitere Bitfolge x exestiert, mit

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

Existiert eine solche Loesung nicht, soll die leere Menge das Ergebnis sein.

Sei folgendes Problem „Null-Bitfolge“ wie folgt definiert:
Gegeben sind n Bitfolgen der Laenge m und eine Ganzzahl $k > 0$.
Das Problem lautet, k der n gegebenen Bitfolgen zu finden, bei welchen gilt

$$s_1 \oplus \cdots \oplus s_k = 0,$$

wobei 0 fuer die Bitfolge bestehend aus m Nullen steht.
Existiert eine solche Loesung nicht, soll die leere Menge das Ergebnis sein.

Das Null-Bitfolge Problem laesst sich auf das ZZZ-Problem reduzieren, indem man die n gegebenen Bitfolgen und $k - 1$ als Eingaben fuer das ZZZ-Problem benutzt. Dadurch erhaelt man k der n Bitfolgen fuer welche gilt

$$s_1 \oplus \cdots \oplus s_{k-1} = x.$$

Das Null-Bitfolge Problem ist durch das Ergebnis des ZZZ-Problems ebenfalls geloest, da man die Gleichung

$$s_1 \oplus \cdots \oplus s_{k-1} = x$$

wie folgt umstellen kann:
Fuegt man zu beiden Seiten $\oplus\, x$ hinzu, erhaelt man

$$(s_1 \oplus \cdots \oplus s_{k-1}) \oplus x = x \oplus x.$$

Durch die Eigenschaft des Selbstinversen, erhaelt man

$$(s_1 \oplus \cdots \oplus s_{k-1}) \oplus x = 0,$$

Weiter erhaelt man mit der Assoziativitaet

$$s_1 \oplus \cdots \oplus s_{k-1} \oplus x = 0.$$

Somit hat man k Bitfolgen gefunden, welche die Null-Bitfolge ergeben.
Fuer den Fall, dass die leere Menge das Ergebnis des ZZZ-Problems ist, gibt es ebenfalls keine Loesung fuer das Null-Bitfolge Problem, so das die leere Menge ebenfalls das Ergebnis ist.

Dass sich das neue Problem auf das ZZZ-Problem reduzieren laesst bedeutet, dass wenn das Null-Bitfolge Problem NP-schwer ist, auch das ZZZ-Problem NP-schwer ist.

Weiter sei gesagt, dass man das ZZZ-Problem ebenfalls auf das Null-Bitfolge Problem reduzieren kann.
Dazu benutzt man die n gegebenen Bitfolgen und $k+1$ als Eingaben fuer das Null-Bitfolge Problem. Dadurch erhaelt man

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

Anschlieszend kann man irgendeine der $k+1$ gefundenen Bitfolgen auf beiden Seiten addieren und erhaelt

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

$$\Leftrightarrow s_1 \oplus \cdots \oplus s_k + 0 = \oplus s_{k+1}$$

$$\Leftrightarrow s_1 \oplus \cdots \oplus s_k = \oplus s_{k+1}.$$

Dadurch hat man k Bitfolgen erhalten fuer welche eine weitere Bitfolge exestiert, bei welcher die vorausgesetzte Bedingung gilt.
Fuer den Fall, dass die leere Menge das Ergebnis des Null-Bitfolge Problems ist, gibt es ebenfalls keine Loesung fuer das Null-Bitfolge Problem, so das die leere Menge ebenfalls das Ergebnis ist.

Dass wiederum bedeutet, dass man das ZZZ-Problem loesen kann, indem man das Null-Bitfolge Problem loest.

6 Komplexitaet

Das ZZZ-Problem ist NP-schwer.
Diese Aussage wird nun bewiesen, indem gezeigt wird, dass das Null-Bitfolge Problem NP-schwer ist. Dies ist moeglich, da das Null-Bitfolge Problem bereits auf das ZZZ-Problem reduziert wurde.

Um zu zeigen, dass ein Problem L NP-hard ist muss jedes Problem in NP durch eine Polynomialzeitreduktion auf L reduziert werden koennen.
Um dies zu zeigen, kann man ein als NP-vollstaendiges Problem auf L reduzieren.

Das Problem von welchem reduziert wird, soll hier das „Weight Distribution“ ([3]) oder auch „Subspace Weights“ ([2]) genannte Problem sein, welches in [2] als NP-vollstaendig bewiesen wurde:
Gegeben sind eine binaere $m \times n$ Matrix A und eine Ganzzahl $k > 0$.
Die Entscheidungsfrage ist, ob eine Menge an k Spalten aus A gibt, welche sich zum Null-Vektor aufaddieren.

Dieses Entscheidungsproblem laesst sich wie folgt durch Polynomialzeitreduktion zum ZZZ-Problem reduzieren:
Sein die Spalten der Matrix die Vektoren

$$\vec{v_1},\ \ldots\ ,\ \vec{v_n}.$$

Diese Vektoren kann man zu Bitfolgen umschreiben:

$$\vec{v_i} = \begin{bmatrix} v_{i,1} \\ \vdots \\ x_{i,m} \end{bmatrix}$$

wird zur Bitfolge

$$s_i = v_{i,1}\ \ldots\ v_{i,m}.$$

Somit erhaelt man die Bitfolgen

$$s_1,\ \ldots\ ,\ s_n.$$

Diese Bitfolgen und k kann man nun als Eingabe fuer das Null-Bitfolge Problem benutzten.
Dadurch erhaelt man k Bitfolgen, welche aufaddiert die Bitfolge bestehend aus m Nullen ergeben oder die leere Menge.
Erhaelt man k Bitfolgen, so handelt es sich um eine Ja-Instanz, erhaelt man die leere Mengen, handelt es sich um eine

Nein-Instanz.
Dass es sich um eine Ja-Instanz handelt, wenn k Bitfolgen das Ergebnis des Null-Bitfolge Problem sind, liegt daran, dass jede der k Bitfolgen einer Spalte der Matrix entspricht. Somit gibt es k Spalten der Matrix, welche den Null-Vektor ergeben.
Dass es sich um eine Nein-Instanz handelt, wenn die leere Menge zurueck gegeben wurde, ergibt ebenfalls Sinn. Denn in dem Fall konnten keine k Bitfolge, sprich keine k Spalte, gefunden werden, welche die Bitfolge bestehend aus m Nullen, bzw. den Null-Vektor, bilden.

Aufgrund dessen, dass das ZZZ-Problem NP-schwer ist, kann man davon ausgehen, dass kein Algorithmus existiert, welche das Problem effizient, bzw. in Polynomialzeit laeuft.

Dennoch werden nun zwei Loesungsansaetze vorgestellt, welche beide auf den Kapiteln „Loesung durch ein Gleichungssystem“ und „Loesung des Gleichungssystems mit bestimmem Hamming-Gewicht“ aufbauen.

7 Loesung durch ein Gleichungssystem

Fuer das zuvor beschriebene Null-Bitfolge Problem werden nun n Entscheidungsvariablen eingefuehrt: $x_1 \ , \ \dots \ , \ x_n$.
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 ueber $\mathbb{Z}_2$ geloest werden:

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

$$\dots$$

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

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

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

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 Null-Bitfolge 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:

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

$$\dots$$

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

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

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

$$\dots$$

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

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.

8 Loesung des Gleichungssystems mit bestimmem Hamming-Gewicht

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

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

$$\cdots$$

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

zu loesen.
Dieses Gleichungssystem ist aequivalent zu diesem:

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

mit

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

und

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

Weiter muss die Gleichung

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

ueber $\mathbb{Z}$ gelten.

Die letzte Beschraenkung ($x_1 + \cdots + x_n = k + 1$) bedeutet, dass die Loesung $\vec{x}$ des ersten Gleichungssystems genau $k + 1$ Einsen beinhalten muss.
Da in $\mathbb{Z}_2$ Eins das einzige Element ungleich Null ist, ist die Beschraenkung aequivalent dazu, dass

$$wt(\vec{x}) = k + 1$$

sein muss.
Wobei $wt(\vec{u})$ das Hamming-Gewicht von $\vec{u}$ ist, welches definiert ist, als die Anzahl an Elementen eines Vektors, welche nicht 0 sind, bzw.

$$wt(\vec{u}) = |\{i \in \{1, \ , \ \dots \ , \ n\} : e_i \neq 0\}|$$

fuer einen Vektor $\vec{u}$ mit den Elementen $e_1, \ , \ \dots \ , \ e_n$.

Es gilt also eine Loesung $\vec{x}$ des linearen Gleichungssystems

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

zu finden mit

$$wt(\vec{x}) = k + 1,$$

wobei das Gleichungssystem ueber $\mathbb{Z}_2$ ist.

9 Loesung durch Gausz-Verfahren und Bruteforce

In diesem Kapitel wird das zuvor beschriebene Problem angegangen, indem zunaechst das Gleichungssystem in $\mathbb{Z}_2$ geloest und anschlieszend nach einer Loesung der Loesungsmenge gesucht wird, welche die Bedingung des Hamming-Gewichts erfuellt.

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

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

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

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 $\vec{v_1} = \vec{0}$.

9.2 Loesen der letzten Gleichung

Zu letzt ist die Loesung der Loesungsmenge zu finden, welche ein Hamming-Gewicht von $k+1$ hat.

Dazu kann man bei Loesungsmengen mit kleiner Anzahl an freien Variablen alle Moeglichen Kombinationen ausprobieren.
Dabei sollte man beachten, dass man niemals mehr als $k+1$ Variablen auswaehlen muss, da bei jedem Vektor, welcher gewaehlt wird mindestens eine Variable aktiviert wird, welche nicht durch einen anderen Vektor wieder deaktiviert werden kann. Diese Variablen sind die freien Variablen.

Durch dieses Verfahren muss man

$$\binom{f(A)}{k+1,}$$

also $f(A)$ ueber $k+1$ moegliche Belegungen ausprobieren.
Dabei stellt $f(A)$ die Anzahl an freien Variablen der Loesungsmenge des homogenen Gleichungssystems zu A dar.

Fuer jede Belegung $\vec{u}$ muss nun ausprobiert werden, ob

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

gilt. Dies ist moegliche, indem alle durch $\vec{u}$ ausgewaehlten Spalten der Matrix addiert werden und anschlieszend ueberprueft wird, ob der Nullvektor entanden ist.

9.3 Laufzeit

Dieses Verfahren hat eine Laufzeit von

$$\mathcal{O}(p(A)) + \mathcal{O}(\binom{f(A)}{k+1} * m * (k+1)).$$

Dabei stellt p ein Polynom dar, welches fuer das Gausz-Verfahrens benoetigt wird.
$\binom{f(A)}{k+1}$ ist erneut die Anzahl an moeglichen Kombinationen. Fuer jede dieser Kombinationen muss fuer $k+1$ Zeilen der Matrix m mal ueberprueft werden, ob die ausgewaehlten Elemente der Zeile 0 ergeben, wenn sie addiert werden, um zu ueberprufen, ob die Kombination eine gueltige Loesung darstellt. Dadurch entsteht eine Laufzeit von $m * (k+1)$ pro Ueberpruefung einer Kombination.

Aufgrund dessen das $\binom{f(A)}{k+1}$ sich wie folgt Umformen laesst:

$$\begin{aligned} &\binom{f(A)}{k+1} \\ &= \frac{f(A)}{(k+1)! * (f(A) - (k+1))!} \\ &= \frac{f(A) * (f(A)-1) * \ \ldots \ * (f(A)-k)}{(k+1)!} \\ &< \frac{1}{(k+1)!} * f(A)^{k+1} \\ &< f(A)^{k+1} \end{aligned}$$

Laesst sie (worst case) Laufzeit reduzieren zu

$$\mathcal{O}(p(A)) + \mathcal{O}(f(A)^{k+1} * m * (k+1)).$$

Da der zweite Summand deutlich schneller waechst als der erste, ist die Laufzeit nun

$$\mathcal{O}(f(A)^{k+1} * m * (k+1)).$$

Da $m * (k+1)$ hier ein pseudopolynomieller Faktor der exponentiellen Laufzeit ist, kann er ebenfalls ignoriert werden, so dass die finale Laufzeit

$$\mathcal{O}(f(A)^{k+1})$$

ist.

10 Loesung des Systems durch minimale Loesungen

Bei diesem Problem werden ich mich auf [1] beziehen.

Das Problem ist es $\vec{x}$ zu finden mit

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

und

$$wt(\vec{x}) = k + 1.$$

In [1] wird bewiesen, dass Jede Loesung eines homogenen Systems

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

besteht aus der Summe disjunkter minimaler Loesungen. Also Loesungen, des Gleichungssystems, welche unterschiedliche Variablen auswaehlen, so dass keine Variable von mehr als einer Loesung ausgewaehlt wird. Ausgewaehlt ist eine Variable dabei dann, wenn eine Loesung die Variable mit dem Wert eins besetzt.

Eine minimale Loesung fuer das System $A * \vec{x} = \vec{b}$ ist dabei eine Loesung mit dem geringsten Hamming-Gewicht aller Loesungen und definiert als $\vec{u} \neq \vec{0}$ mit

$$A * \vec{u} = b$$

und dass fuer jede andere Loesung $\vec{u'} \neq \vec{0}$ gilt, dass

$$u'_i \leq u_i$$

fuer alle i, bedeutet, dass $\vec{u'} = \vec{u}$.
Bzw., dass keine weitere Loesung $\vec{u'} \neq \vec{0} \neq \vec{u}$ existiert, bei welcher fuer alle Elemente $u'_i \leq u_i$ gilt.

Mit diesem Wissen, kann man nun differenzieren zwischen zwei Faellen.

1. $k + 1$ ist eine Primzahl
2. $k + 1$ ist keine Primzahl

Diese zwei Moeglichkeiten werden nun erlaeutert.

11 $k + 1$ ist eine Primzahl

Dieser Punkt ist nicht trivial, da bei allen Beispielen diese Einschraenkung zutrifft.

Geht man davon aus, dass es eine Loesung fuer das Problem gibt, gibt es nun zwei Moeglichkeiten fuer die Zusammensetzung der Loesung.
Aufgrund dessen, dass jede Loesung des Gleichungssystems aus der Summe disjunkter minimaler Loesungen besteht, muss das Hamming Gewicht der minimalen Loesungen entweder 1 oder $k + 1$ sein. Denn nur dann kann die Gleichung

$$l * u = k + 1$$

erfuellt werden, wobei $l \in \mathbb{Z}$ das minimale Hamming-Gewicht und $u \in \mathbb{Z}$ die Anzahl an minimalen disjunkten Loesungen ist, welche benoetigt werden, um das Hamming-Gewicht $k + 1$ zu erreichen.
Waere l naemlich nicht $k + 1$ oder 1, haette $k + 1$ mehr als zwei Teiler wodurch ein Wiederspruch dazu entsteht, dass $k + 1$ eine Primzahl ist (Beweis durch Widerspruch).

Der Fall, dass das minimale Gewicht 1 ist, kann sehr einfach in pseudopolynomieller Zeit ueberprueft werden.
Denn fuer ein homogenes lineares System kann das minimale Hamming-Gewicht nur dann 1 sein, wenn mindestens eine Spalte der Matrix A existiert, welche nur aus Nullen besteht.
Der Pseudocode fafuer sieht wie folgt aus:

Fuer den Fall, dass das minimale Hamming-Gewicht nicht Eins ist, wird das Problem zum Minimal Weight Problem. Dieses Problem ist als NP-Schwer bekannt und wie folgt definiert.

12 $k + 1$ ist keine Primzahl

13 Implementierung

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

14 Laufzeitanalyse

Die laufzeit des Programm besteht aus zwei Teilen.

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

15 Aufgabenteil c - Beispiele

16 Aufgabenteil b

In Aufgabenteil b 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.
Diese Aufgabenstellung wird nun mit dem allgemeineren Problem angegangen, dass es n Haeuser gibt, so dass $n+1$ Karten gefunden werden.

Sein zunaechst die Folge

$$W = (w_1, \ \dots \ , w_n)$$

die Karten der in der Aufgabenstellung erwaehnten (aufsteigend sortierten) Codeworte und x die Sicherungskarte.

Weiter seien die gefundenen Karten

$$K = (k_1, \ \dots \ , \ k_{n+1}).$$

Nun seien die Karten

$$S = (s_1, \ \dots \ , \ s_{n+1})$$

K aufsteigen sortiert.
W, K und S sollen hier Folgen sein.

Da davon ausgegangen wird, dass die $n+1$ richtigen Karten gefunden wurden, besteht S aus W und x.
Wichtig ist, dabei, dass W innerhalb von S immer noch die selbe Reihenfolge hat, da die Karten in beiden Folgen aufsteigend sortiert sind.
Somit wurde ausschlieszlich die zusaetzliche Karte x in die Folge W hinzugefuegt, wodurch S entsteht.

Sei der Index an welchem x zu W hinzugefuegt wurde j, wobei $1 \leq j \leq n+1$ gilt, so dass x innerhalb der Folge oder rechts an die Folge hinzugefuegt wird.
Dadurch verschieben sich alle Karten mit einem Index $i \geq j$ um eins nach rechts, so dass eine Karte w_i nun an Index $i+1$ ist (so fern $i \geq j$). Die restlichen Karten (mit $i < j$) bleiben an ihrer Position.
Die Folge saehe dann wie folgt aus:

$$(w_1, \ \dots \ , \ w_{j-1}, \ x, \ w_j, \ \dots \ , \ w_n)$$

In dieser Folge ist zu erkennen, dass x am Index j liegt. Rechts von q sind die Elemente $e_j, \ \dots \ , \ e_n$, welche nun alle eine Position nach rechts gesetzt wurden (der Index in der Folge hat sich je um 1 erhoeht). Somit hat die Liste nun eine Laenge von $n+1$.

Somit gibt es fuer jeden Index i eines Codewortes w_i $(1 \leq j \leq n)$ zwei Moeglichkeiten, entweder w_i ist in S an Position i oder an Position $i+1$. Je nachdem, ob $i < j$ bzw. $i \geq j$ ist.
Das bedeutet, dass man fuer das Haus k ebenfalls zwei Moeglichkeiten fuer das Codewort gibt. Entweder s_k oder s_{k+1}.

Zusammengefasst muss man also die $n+1$ gefundenen Karten sortieren und kann anschlieszend das Haus k mit dem Schluessel an Index k oder $k+1$ der sortierten Karten suchen. Dies haelt die Bedingung ein, dass maximal 3 Versuche benoetigt werden duerfen.

Auf die gegebene Aufgabe laesst sich dieses Verfahren einfach uebertragen, indem die 11 gefundenen Karten sortiert werden und fuer jedes Haus k ebenfalls die Karten k und $k+1$ probiert werden.

17 Literatur

1. V. Arvind, Johannes Köbler, ebastian Kuhnert, und Jacobo Torán, Solving Linear Equations Parameterized by Hamming Weight
2. Elwyn R. Berlekamp, Robert J. McEliece and Henk C.A. van Tilborg, On the inherent intractability of certain coding problems

3. Rod G. Downey, Michael R. Fellows, Alexander Vardy and Geoff Whittle, The parametrized complexity of some fundamental problems in coding theory