Turn All the Lights Off

15. August 2012

Max von Tettenborn

max.von.tettenborn@in.tum.de

***Zusammenfassung*—*Lights Out!* ist ein Logik-Spiel, welches in Variationen schon oft veröffentlicht wurde, in den 90ern eher von Spielwarenherstellern als eigenständige Handheld-Geräte, inzwischen in Form von Smartphone- oder Tablet-Apps. Das Spielfeld besteht aus beleuchteten Knöpfen, die es gilt, durch geschicktes Drücken auszuschalten.**

Im Folgenden wird die Problemstellung mathematisch analysiert und auf ein umfassend erforschtes Standard-Lösungsverfahren reduziert. Ein einfach verständlicher, im Hinblick auf Effizienz jedoch ausbaufähiger Algorithmus zur Lösung wird auch vorgestellt.

I. Spielregeln

Das Spielfeld von *Lights Out!* besteht aus einer quadratischen Matrix, deren Elemente hintergrundbeleuchtete Knöpfe sind. Die übliche Matrixgröße beträgt 5×5 Knöpfe. Zum Spielstart ist eine vorgegebene Menge der Knöpfe beleuchtet, die Übrigen sind ausgeschaltet. Ausgehend von dieser Startkofiguration ist es die Aufgabe des Spielers, durch geschicktes Drücken der Knöpfe den Zustand des Spielfeldes nach und nach zu verändern, bis schließlich alle Lichter aus sind.

Dabei schaltet jeder Knopf genau fünf Lichter um, nämlich das Licht, auf das gedrückt wurde, und die Lichter, die sich in *Von-Neumann-Nachbarschaft* um das betätigte Licht befinden. Dies ist der linke und rechte, sowie der obere und untere Nachbar.

Sollte sich der betätigte Knopf an einem der Ränder oder in einer der Ecken des Spielfeldes befinden, so fehlen ihm ein oder zwei Nachbarn. Hier tritt ein *Wrap-Around* Effekt auf: die Nachbarschaftsbeziehung wickelt sich einmal um das Spielfeld herum, so dass die äußersten Elemente wieder jeweils Nachbarn sind. Mathematisch ausgedrückt schaltet der Knopf mit den Koordinaten (x, y) die Lichter mit den folgenden Koordinaten: $(x, y), ((x + 1) \mod n, y), ((x - 1) \mod n, y), (x, (y + 1) \mod n), (x, (y - 1) \mod n)$, wobei die Matrixgröße $n \times n$ beträgt. *Abb. 1* verdeutlich die Auswirkungen eines Druckes auf den rot umrandeten Knopf.

Dieser *Wrap-Around* Effekt ist nicht Teil der originalen Regeln, sondern wurde zusätzlich eingeführt. Er sorgt dafür, dass ein beliebter Algorithmus zum Lösen eines gegebenen Spielfeldes nicht mehr so leicht anwendbar ist: beim *Light-Chasing* schaltet man die Zeilen von oben nach unten aus, indem man für jedes angeschaltete Licht den Knopf direkt darunter drückt. Dies kann man rein mechanisch fortsetzen, bis nur noch in der untersten Zeile Lichter an sind. Für jede der wenigen dann übrigen Möglichkeiten kann man sich eine einfache Lösung merken. Mit *Wrap-Around* wird dieses Vorgehen erschwert, da man bei der Elimination der vorletzten Zeile wieder Lichter in der ersten Zeile anschalten würde.

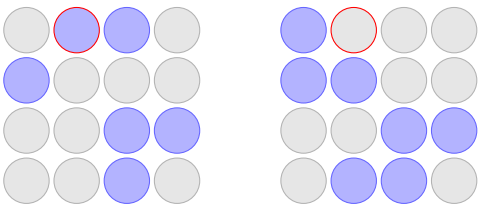

Abbildung 1: Ein Spielfeld vor und nach einem Knopfdruck

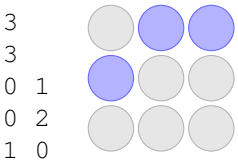

Abbildung 2: Beispiel-Input und dessen Interpretation

II. Spezifikation des Algorithmus

Der Algorithmus soll eine Eingabe akzeptieren, die ein Spielfeld in Form einer Reihe von Integers kodiert. Die Liste dieser Integers wird wiederum als String übergeben. Dabei gibt der erste Wert die Anzahl der Reihen bzw. Spalten des quadratischen Spielfeldes an. Dieser Wert wird im Folgenden mit n bezeichnet. Ein weiterer Wert, durch einem Zeilenumbruch vom ersten getrennt, gibt Auskunft über die Anzahl der eingeschalteten Lichter. Deren Koordinaten folgen als Tupel von Integer-Werten. *Abb. 2* zeigt einen beispielhaften Input und das dazugehörige Spielfeld.

Der geforderte Algorithmus soll jedoch nicht berechnen, wie ein Spieler zur Lösung gelangen kann, sondern lediglich, ob dies für die gegebene Startkonfiguration prinzipiell möglich ist. Diese Frage soll im Rückgabewert mit „Yes“ oder „No“ beantwortet werden.

III. Mathematische Modellierung

Wie bei jedem komplexeren Problem ist auch hier die Modellierung durch mathematische Strukturen außerordentlich hilfreich. So lässt sich das Spiel, dessen Regeln und deren teils überaschende Implikationen einem Betrachter nicht sofort intuitiv klar werden, auf bekannte und ausgiebig erforschte Zusammenhänge reduzieren. Einem mathematisch versiertem Verstand erschließen sich damit Erkenntnisse und Lösungsansätze, die anfangs weitaus schwerer sichtbar waren.

Da die folgende Modellierung auf dem Körper $\mathbb{Z}_2$ basiert, soll dieser hier kurz vorgestellt werden. Details sind der

$+_2$	0	1
0	0	1
1	1	0

$\cdot_2$	0	1
0	0	0
1	0	1

Abbildung 3: Operationen auf $\mathbb{Z}_2$

einschlägigen Literatur zu entnehmen. $\mathbb{Z}_2$ ist ein endlicher Körper oder Galoiskörper. Oft auch durch $\mathbb{F}_2$ oder $GF(2)$ beschrieben, bezeichnet er die zweielementige Menge $\{0, 1\}$. Die Körpereigenschaft ergibt sich durch die Primheit von 2. Dadurch kann mit $\mathbb{Z}_2$ wie mit jedem anderen Körper, beispielsweise $\mathbb{R}$ oder $\mathbb{C}$ gerechnet werden. Die Operationen Addition und Multiplikation werden *modulo 2* ausgeführt, um Abgeschlossenheit zu gewährleisten. An der Wertetabelle in *Abb. 3* lässt sich sehen, wie eine Addition mit 1 genau einem Umschalten entspricht, so wie es in den Spielregeln von Bedeutung ist. Dieser Umstand wird bei der Modellierung ausgenutzt.

Folgende Aspekte von *Lights-Out!* lassen sich nun in mathematische Ausdrücke übersetzen:

- Der Zustand des Spielfeldes bezeichnet die aktuelle Verteilung von angeschalteten und ausgeschalteten Lichtern. Er lässt sich als Vektor aus $\mathbb{Z}_2^{n^2}$ darstellen. Indem die Lichter des Spielfeldes zeilen- und spaltenweise wie in *Abb. 4* durchnummeriert werden, erhält jedes eine Zahl $i \in [0, n^2 - 1]$. Ein Vektor stellt dann einen Zustand dar, falls seine i-te Komponente eine 1 enthält, g.d.w. das i-te Licht des Spielfeldes angeschaltet ist. Das Spielfeld aus *Abb. 4* wird also durch den Vektor $(0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0)$ dargestellt. Der Zustand, in dem sich das Spielfeld zu Beginn des Spiels befindet, soll im Folgenden mit b bezeichnet werden.
- Ein Knopfdruck lässt sich ebenfalls als Vektor darstellen. Wie oben beschrieben entspricht eine Addition einer Zahl aus $\mathbb{Z}_2$ mit einer 1 einem Umschalten dieser Zahl, während eine Addition mit 0 nichts bewirkt. Dies wird bei der Formulierung eines Knopfdrucks ausgenutzt, denn ein Knopfdruck schaltet ja gerade manche Lichter um und lässt andere unbeeinflusst. Eine komponentenweise Addition des Spielfeldzustandes mit einem Knopfdruckvektor, welcher genau die umzuschaltenden Lichter mit einer 1 markiert, ergibt also den Zustand nach den Knopfdruck. Von diesen Knopfdruckvektoren gibt es n^2 viele, für jeden Knopf einen Eigenen. Im Folgenden wird der Knopfdruckvektor für den i-ten Knopf mit a_i bezeichnet.
- Das Ziel des Spiels stellt den Spielfeldzustand dar, in dem alle Lichter ausgeschaltet sind. Dieser Zustand wird natürlich durch den Nullvektor $0 \in \mathbb{Z}_2^{n^2}$ repräsentiert.

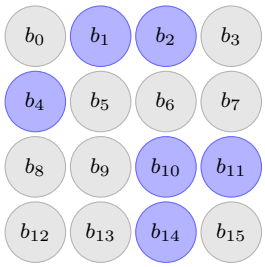

Abbildung 4: Modellierung des Spielfeldes als Vektor

IV. Analyse und Lösungsweg

Es steht nun ein mathematisches Modell zur Verfügung, welches das Problem beschreibt. Im nächsten Schritt wird es Verwendung in der Beschreibung der Lösung finden.

Zunächst besagen die Spielregeln, dass das Spielfeld im Anfangszustand b startet und durch eine Reihe von Knopfdrücken in den Endzustand 0 überführt wird. Angenommen die Sequenz $8 \to 11 \to 2 \to 0 \to 6$ würde das Spiel lösen. Dann bedeutet das nach dem Modell, dass folgende Gleichung gilt:

$$b + a_8 + a_{11} + a_2 + a_0 + a_6 = 0$$

Beim Betrachten dieser Gleichung stechen zwei Erkenntnisse hervor:

1) Die Reihenfolge, in der die Knöpfe gedrückt werden, beeinflusst das Ergebnis nicht, denn die Addition modulo 2 ist kommutativ. Daher lassen sich die Summanden beliebig umordnen.
2) Jeder Knopf muss höchstens einmal gedrückt werden. Falls es eine Lösung gibt, bei der ein Knopf mehr als einmal gedrückt wird, lässt diese sich umformen zu einer Lösung mit höchstens einer Betätigung pro Knopf. Man muss nur die Knopfdrücke so umordnen, dass der gleiche Knopf zweimal direkt hintereinander betätigt wird. Offensichtlich kann man dieses Paar dann aus der Sequenz entfernen, da der Zustand vor den zwei Betätigungen identisch mit dem Zustand danach ist.

Mit Hilfe dieser beiden Erkenntnisse wird sichtbar, dass sich jede Lösung in eine einheitliche Normalform überführen lässt, für die gilt, dass jeder Knopf höchstens einmal gedrückt wird und die Knöpfe in der Reihenfolge ihrer Nummerierung gedrückt werden. Jede Lösung wird damit vollständig beschrieben, indem für jeden Knopf gespeichert wird, ob er zur Lösung gedrückt werden muss oder nicht. Diese Information lässt sich genau in einem weiteren Vektor $x \in \mathbb{Z}_2^{n^2}$ speichern. Obige Gleichung wird nun mit Hilfe dieses Vektors in eine Linearkombination der Knopfdruckvektoren umgewandelt. Für jede Lösung x gilt dann:

$$b + a_0 x_0 + a_1 x_1 + \cdots + a_{n^2-1} b_{n^2-1} = 0$$

Eine einfache Umformung bringt b auf die rechte Seite, wobei anzumerken ist, dass in $\mathbb{Z}_2$ keine negativen Zahlen enthalten sind und somit $-b$ unter *modulo 2* zu $+b$ wird.

$$a_0 x_0 + a_1 x_1 + \cdots + a_{n^2-1} b_{n^2-1} = b$$

Durch Umschreiben wird ersichtlich, dass es sich bei der Gleichung um ein lineares Gleichungssystem handelt.

$$Ax = b \text{ mit } A \in \mathbb{Z}_2^{n^2 \times n^2} \text{ und } x, b \in \mathbb{Z}_2^{n^2}$$

Der Vektor b beschreibt dabei den Startzustand des Spielfeldes und die Matrix A beschreibt die Spielregeln, indem sie zu jedem Knopfdruck die Lichter angibt, die durch ihn umgeschaltet werden. Ein genauerer Blick auf diese Matrix lässt ein sich wiederholendes Muster erkennen, was es erlaubt, sie wie folgt aufzuschreiben.

$$A = \begin{pmatrix} D & I & 0 & \cdots & I \\ I & D & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & D & I \\ I & \dots & 0 & I & D \end{pmatrix}, D = \begin{pmatrix} 1 & 1 & 0 & \cdots & 1 \\ 1 & 1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 & 1 \\ 1 & \dots & 0 & 1 & 1 \end{pmatrix}$$

A besteht also aus $n \times n$ Bereichen, die jeweils $n \times n$ Werte enthalten. (*Abb. 5* zeigt ein Beispiel eines vollständig ausgeschriebenen A) Die Bereiche der Diagonale gleichen der Matrix D, während die Bereiche der Nebendiagonalen und der äußersten Ecken Identitätsmatrizen sind. Um das Zustandekommen dieses Musters einzusehen, ist es empfehlenswert, sich einige Knopfdruckvektoren einzeln anzusehen und die entsprechende Spalte der Matrix zu suchen. So sieht man beispielsweise, dass die 1 in den Ecken von D stets durch den horizontalen *Wrap-Around* zustandekommt. Ebenso werden die Identitätsmatrizen in den Ecken von A durch den vertikalen *Wrap-Around* erzeugt. Aufgrund des Musters lässt sich die Matrix schnell und elegant durch ein Programm erzeugen.

V. Implementierung

Nachdem das Problem auf das Lösen eines linearen Gleichungssystems (*SLE*, System of Linear Equations) reduziert wurde, hat man nun die Qual der Wahl, welchen der vielen Algorithmen, die es zu *SLE*s gibt, man verwenden möchte. Anzumerken ist hier, dass die effizienten iterativen Verfahren, wie Glätter bzw. Relaxationsverfahren, schwierig anzuwenden sein werden. In $\mathbb{Z}_2$ kann man sich nicht an eine Lösung annähern. Entweder man liegt vollständig richtig oder komplett falsch. Gesucht ist dementsprechend keine Näherung, sondern eine genaue Lösung.

Das wohl bekannteste und älteste Lösungsverfahren für *SLE*s, die Gauss-Elimination, ist ein direkter Löser und lässt sich ohne Probleme auf dieses Problem anwenden. Die entsprechenden Operationen auf $\mathbb{Z}_2$ zu übertragen, ist eine einfache Übung. Zur effizienten Darstellung im Rechner lassen sich die Elemente 0 und 1 als einzelne Bits speichern, die Matrix wird damit besonders platzsparend.

Die Gauss-Elimination bringt das Gleichungssystem auf eine Stufenform, welche es erlaubt, sofort die Existenz einer Lösung abzulesen. Dies ist ja die Frage, die vom Algorithmus beantwortet werden soll. Zusätzlich lässt sich anschließend durch Rückwärtseinsetzen eine konkrete Lösung, also ein gültiges x, welches das Gleichungssystem erfüllt, somit also eine Menge von Knöpfen, die man zum Ausschalten aller Lichter drücken muss, errechnen. Die Umformung des *SLE* geschieht durch die Anwendung von elementaren Umformungen, welche nicht die Lösung, wohl aber das Gleichungssystem

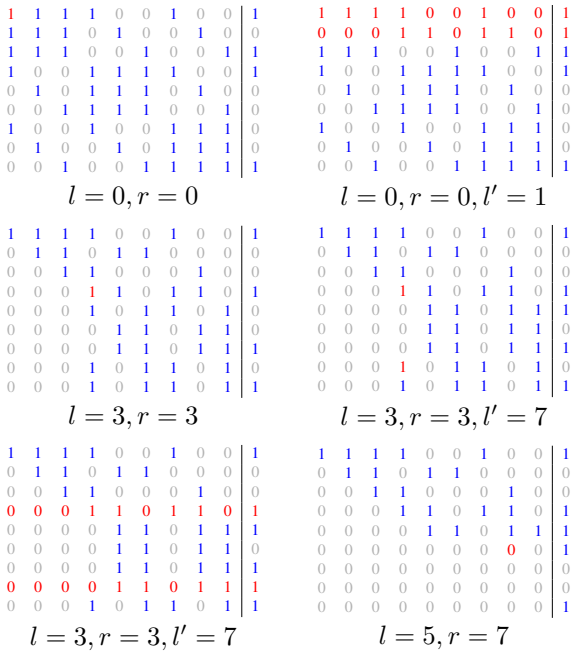

Abbildung 6: Verschiedene Zustände während der Gauss-Elimination

ändern und bei richtiger Anwendung insbesondere vereinfachen:

1) Das Vertauschen zweier Zeilen. Dies ändert ganz offensichtlich nicht die Lösung.
2) Die Addition einer Zeile auf eine andere Zeile. Über $\mathbb{R}$ würde man hier von der Addition eines Vielfachen einer Zeile auf eine andere Zeile reden, aber über $\mathbb{Z}$ gibt es ein Vielfaches nicht. Auch diese Operation ändert das Ergebnis des SLE nicht, auch wenn dies nicht ganz so offensichtlich ist wie bei der Vertauschung.

Der Ablauf des Gauss-Verfahrens ist in *Algorithmus 1* gegeben. *Abb. 6* zeigt ausgewählte Momentaufnahmen der Matrix wärend der Elimination. Diese beinhaltet, von links nach rechts die Spalten abzuarbeiten und dabei alle Elemente der jeweiligen Spalte, die sich unterhalb einer Stufe (die Anfangs entlang der Diagonalen verläuft) befinden, falls nötig auf 0 zu setzen. Dies geschieht, indem die Zeile der jeweiligen Stufe

Data : $A \in \mathbb{Z}_2^{n^2 \times n^2}$, $b \in \mathbb{Z}_2^{n^2}$
1 $l := 0$;
2 **for** $r := 0$ **to** $n^2 - 1$ **do**
3 pivotize(l,r);
4 **if** $A_{lr} = 1$ **then**
5 **for** $l' := l + 1$ **to** $n^2 - 1$ **do**
6 **if** $A_{l'r} = 1$ **then**
7 $A_{l'} := A_{l'} +_2 A_l$;
8 $b_{l'} := b_{l'} +_2 b_l$;
9 $l := l + 1$;

Algorithmus 1: Die Gauss-Elimination als Pseudo-Code

			Zeile 0				Zeile 1				Zeile 2				Zeile 3			
			Spalte 0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
Zeile 0	Spalte	0	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
		1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0
		2	0	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0
		3	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1
Zeile 1	Spalte	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0
		1	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0
		2	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0
		3	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0
Zeile 2	Spalte	0	0	0	0	0	1	0	0	0	1	1	0	1	1	0	0	0
		1	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0
		2	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0
		3	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1
Zeile 3	Spalte	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1
		1	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1	0
		2	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	1
		3	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1

Abbildung 5: A für $n = 4$

auf die Zeile mit dem zu eliminierenden Element aufaddiert wird. Vorher muss noch sichergestellt werden, dass im Stufenelement eine 1 steht, indem man bei der sog. Pivotisierung die erste in Frage kommende Zeile mit der Stufenzeile vertauscht. Eine Ausnahme tritt auf, wenn alle in Frage kommenden Elemente der Spalte, sowie das Stufenlement bereits 0 sind. Dann wird die Spalte übersprungen und die Stufe bleibt in der selben Zeile, während mit der nächsten Spalte weiter gearbeitet wird. Falls dieser Fall mindestens einmal auftritt, ist das System unterbestimmt und kann mehrere oder keine Lösung haben, aber nicht mehr eine eindeutige Lösung. Da es sich hierbei um das Standard-Lösungsverfahren für *SLE*s handelt, wird auch hier auf die Literatur verwiesen.

Nachdem das System in Stufenform gebracht wurde, lässt sich die Existenz einer Lösung direkt an den Nullzeilen, falls vorhanden, ablesen. Im Fall, dass das System keine Nullzeilen enthält, gibt es genau eine Lösung. Falls Nullzeilen vorhanden sind, werden das die untersten Zeilen des Systems sein. Indem man sich die Gleichungen, die diese Zeilen zusammen mit dem Vektor b rechts davon darstellen, ansieht, lässt sich bestimmen, ob es ein ein x gibt, welches sie erfüllen kann. Jede dieser Gleichungen kann nämlich eine der beiden folgenden Formen annehmen:

$$0 \cdot x_0 + 0 \cdot x_1 + \cdots + 0 \cdot x_{n^2-1} = 0$$

In diesem Fall ist klar, dass jedes x diese Gleichung erfüllt.

$$0 \cdot x_0 + 0 \cdot x_1 + \cdots + 0 \cdot x_{n^2-1} = 1$$

Hier sieht man sofort, dass es kein x geben kann, welches die Gleichung erfüllt.

Sollte in der Stufenform also eine 1 rechts neben einer Nullzeile stehen, so bedeutet das, dass es keine Lösung für das Gleichungssystem gibt. Andernfalls gibt es mehrere Lösungen. Diese Erkenntnis ist das gesuchte Ergebnis und wird vom Algorithmus zurückgegeben.

Eine Laufzeitbetrachtung des Algorithmus ergibt, dass seine Komplexität kubisch in der Dimension der Matrix A ist. Dies erkennt man an den drei geschachtelten Schleifen:

1) Die Schleife in Zeile 2 läuft alle Spalten der Matrix von links nach rechts ab.
2) Die Schleife in Zeile 5 bearbeitet alle Zeilen unter der jeweils betrachteten Stufe.
3) Die dritte Schleife ist implizit in Zeile 7 erkennbar. Die Summanden und das Ziel der Zuweisung sind Zeilen der Matrix, erkennbar daran, dass nur ein einzelner Wert im Index steht. Um zwei Zeilen zu addieren muss in der Praxis eine weitere Schleife über die Elemente laufen.

Da die Matrixgröße jedoch quatratisch mit der Spielfeldgröße wächst, erhält man eine Gesamtkomplexität von $\mathcal{O}(n^6)$, wobei n wie gehabt die Kantenlänge des Spielfeldes beschreibt. Asymptotisch betrachtet ist der Algorithmus also nicht besonders schnell. Hochrechnung hat ergeben, dass die naive Java-Implementierung auf einem aktuellen Desktop-Rechner mehrere Tage für die Berechnung mit $n = 40$ benötigen würde. Andererseits ist das Verfahren leicht verständlich und einfach zu implementieren. Da über $\mathbb{Z}_2$ außerdem mit einzelnen Bits gerechnet wird, lassen sich die Additionen äußerst effizient implementieren, nämlich durch die boolsche Operation *XOR* auf ganzen Bitvektoren. So ergibt sich ein großes Potenzial für Optimierungen.

Abbildungsverzeichnis

1 Ein Spielfeld vor und nach einem Knopfdruck . 1
2 Beispiel-Input und dessen Interpretation 1
3 Operationen auf $\mathbb{Z}_2$ 2
4 Modellierung des Spielfeldes als Vektor 2
6 Verschiedene Zustände während der Gauss-Elimination 3
5 A für $n = 4$ 4

Liste der Algorithmen

1 Die Gauss-Elimination als Pseudo-Code 3