Lösungen zu den Übungsaufgaben im Buch *Automaten und Sprachen: Theoretische Informatik für die Praxis* von Andreas Müller, ISBN 978-3-662-70145-4 (Softcover), ISBN 978-3-662-70146-1 (eBook), https://link.springer.com/book/10.1007/978-3-662-70146-1. Website zum Buch: https://autospr.ch

Kapitel 12: Komplexität

12.1. Konstruieren Sie ein binäres Additionsprogramm für eine geeignet ausgebaute Turing-Maschine, welches Laufzeit $O(n)$ hat, wobei n die Anzahl Stellen der Summanden ist. Warum ist Ihre Maschine so viel schneller als die im auf Seite 226 verlinkten Video gezeigte Maschine?

Lösung. Wir verwenden eine Maschine mit drei Bändern. Zunächst schreiben wir den zweiten Summanden auf das zweite Band, was in $O(n)$ Schritten durchgeführt werden kann. Dann beginnt die eigentlich Addition, wir lesen den ersten Summanden vom ersten Band, beginnend beim niederwertigsten Bit, und den zweiten Summanden vom zweiten Band, ebenfalls beginnend beim niederwertigsten Bit. Die Summe wird jeweils auf Band 3 geschrieben. Nach $O(n)$ solchen Operationen steht das Resultat auf Band 3.

Nach den in Abschnitt 12.1.2 zusammengetragenen Erkenntnissen benötigt die Simulation dieser Maschine mit drei Bändern auf einer Standardmaschine die Laufzeit $O(n^2)$, was sich mit der Laufzeit der Maschine des Videos deckt. ○

12.2. Zwei Graphen G und H heißen isomorph, wenn die Knoten von G so umgeordnet werden können, dass die beiden Graphen übereinstimmen. Zeigen Sie, dass

$$ISO = \{(G, H) \mid G \text{ und } H \text{ sind isomorph.}\}$$

in NP ist.

Lösung. Als Lösungszertifikat c brauchen wir die Zuordnung der Knoten von G zu den Knoten von H. Dann müssen wir nur noch überprüfen, ob zu jeder Kante von G auch eine Kante von H gehört und umgekehrt, was in $O(n^4)$ möglich ist (n ist die Anzahl der Ecken). Also hat ISO einen polynomiellen Verifizierer, und ist damit in NP. ○

12.3. Seien A_1 und A_2 Sprachen, die von einer nichtdeterministischen Turing-Maschine in polynomieller Zeit entscheidbar sind. Zeigen Sie:

a) Die Schnittmenge $A_1 \cap A_2$ ist in NP.

b) Die Verkettung A_1A_2 ist in NP.

Lösung. Man muss nur nachweisen, dass es einen Verifizierer gibt, der polynomielle Laufzeit hat. Dabei kann man davon ausgehen, dass V_1 in polynomieller Zeit $p_1(n)$ $w_1 \in A_1$ verifiziert, und analog für V_2.

a) Um $w \in A_1 \cap A_2$ zu verifizieren, verlangt man als Zertifikat die beiden Zertifikate, die zertifizieren, dass $w \in A_1$ ist und $w \in A_2$. Dann lässt man beide Verifizierer V_1 und V_2 laufen, wozu Laufzeit $p_1(n) + p_2(n)$ notwendig ist, also polynomielle Laufzeit.

		1	4
2	4		

3	2	1	4
4	1	2	3
1	3	4	2
2	4	3	1

Abbildung 1: $2^2 \times 2^2$-Sudoku zu Aufgabe 12.4, links die Problemstellung, rechts die Lösung.

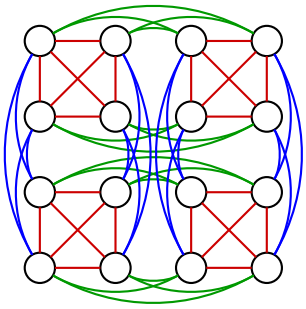
b) Um $w \in A_1A_2$ zu verifizieren verlangt man als Zertifikat die Zerlegung $w = w_1w_2$ und die beiden Zertifikate, die $w_1 \in A_1$ bzw. $w_2 \in A_2$ zertifizieren. Dann kontrolliert man, ob tatsächlich $w = w_1w_2$ ist, und lässt die Verifizierer auf den beiden Teilwörtern mit den jeweiligen Zertifikaten laufen. Dazu ist die Laufzeit $O(n + p_1(n) + p_2(n))$, also in polynomieller Zeit nötig. ○

12.4. Konstruieren Sie eine Reduktion *SUDOKU* $\leq_P$ *VERTEX-COLORING*.

Hinweis. Es reicht, die Reduktion für den Fall $n = 2$ durchzuführen, also für das $2^2 \times 2^2$-Sudoku, wenn daraus klar wird, wie die Verallgemeinerung für größere n zu handhaben ist. Sie können ganz konkret das Beispiel aus Abbildung 1 für Ihre Konstruktion verwenden.

Lösung. Zu einem Sudoku-Rätsel konstruieren wir einen Graphen wie folgt: Jedes Feld des Sudoku-Rätsel wird zu einem Vertex des Graphen. Der Graph erhält eine Kante für jedes Paar von Feldern, die nicht das gleiche Zeichen enthalten dürfen, also wenn die beiden Felder in der gleichen Zeile, der gleichen Spalte oder dem gleichen Unterfeld liegen.

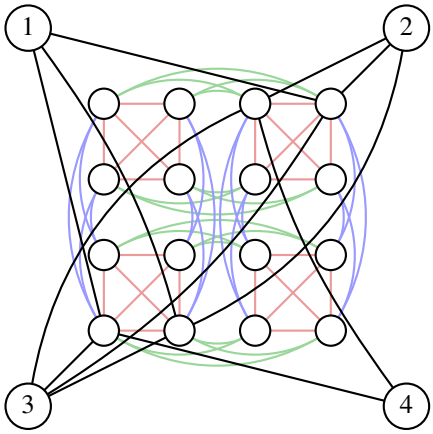
Für ein $2^2 \times 2^2$-Sudoku wie das in Abbildung 1 ergibt sich so der Graph:



(1)

Die roten Kanten stellen sicher, dass jedes Zeichen in einem Unterquadrat nur einmal vorkommt. Die grünen Kanten stellen zusätzlich sicher, dass in jeder Zeile jedes Zeichen nur einmal vorkommt (die horizontalen roten Kanten sind dazu ebenfalls nötig), während die vertikalen blauen Kanten erzwingen, dass in jeder Spalte jedes Zeichen nur einmal vorkommt.

Außerdem müssen wir die Vorgaben abbilden. Dazu konstruieren wir noch zusätzliche Vertices, die wir mit den Zeichen aus Σ anschreiben. Wir nennen diese Vertices die *Zeichenvertices*. Die Vorgabefelder werden mit all den Zeichenvertices verbunden, die verschieden sind vom Vorgabezeichen. Für das $2^2 \times 2^2$-Sudoku aus Abbildung 1 müssen wir zum Graphen (1) noch die folgenden Kanten hinzufügen:



(2)

Um sicherzustellen, dass diese neuen Knoten verschiedene Farbe erhalten, müssen diese untereinander alle verbunden werden, sie müssen also einen vollständigen Teilgraphen bilden. Diese Verbindungen sind der Übersichtlichkeit halber nicht eingezeichnet.

Die Vereinigung der Kanten in (1) und (2) liefert den Graphen, der dem gegebenen Sudoku-Rätsel zugeordnet wird. Für die Zahl der Farben setzen wir $k = |\Sigma|$, im Beispiel des $2^2 \times 2^2$-Sudokus aus Abbildung 1 ist $k = 4$.

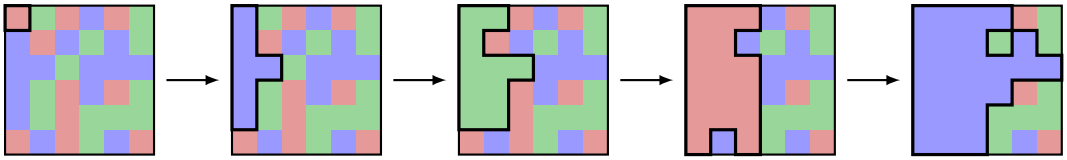
Ist das Sudoku-Rätsel lösbar, kann man eine beliebige Zuordnung von Farben zu den einzelnen Zeichen von Σ verwenden, um die Vertices des Graphen einzufärben.

Ist umgekehrt der konstruierte Graph mit k Farben einfärbbar, dann kann man die Farben der Zeichen-Vertices verwenden, um allen anderen Vertices ebenfalls ein Zeichen zuzuordnen. Diese Zeichenbelegung ist automatisch eine Lösung. Die Kanten innerhalb des Sudoku-Feldes sorgen dafür, dass die Sudoku-Regeln eingehalten werden, die Kanten zu den Zeichen-Vertices sorgen dafür, dass die Vorgaben erfüllt sind.

Wir haben damit gezeigt, dass der konstruierte Graph genau dann mit k Farben einfärbbar ist, wenn das Sudoku-Rätsel lösbar ist. Außerdem ist der Aufwand für die Konstruktion des Graphen von der Größenordnung des ursprünglichen Sudoku, also $O(n)$. Die konstruierte Reduktion ist also sogar eine polynomielle Reduktion. ○

12.5. Beim Spiel Flood-It sind die Felder eines $n \times m$-Spielfeld mit mindestens drei verschiedenen Farben eingefärbt. Man sagt, zwei Felder sind verbunden, wenn sie die gleiche Farbe haben und über eine Kante benachbart sind. Ein Gebiet besteht aus allen miteinander verbundenen Feldern. Der Spieler kann jetzt jeweils die Farbe des Feldes in der linken oberen Ecke ändern, dabei wird auch die Farbe des ganzen zugehörigen Gebietes geändert. Da an das Gebiet Felder der neuen Farbe anstoßen können, kann das Gebiet durch den Farbwechsel größer werden. Ziel ist, das Gebiet der linken oberen Ecke in möglichst wenigen Farbwechseln so anwachsen zu lassen, dass es das ganze Spielfeld überdeckt.

Im folgenden Bild sind vier Schritte des Spiels dargestellt, das Gebiet des Feldes in der linken oberen Ecke ist jeweils fett ausgezogen.



Kann eine nichtdeterministische Turing-Maschine in polynomieller Zeit herausfinden, ob zu einer vorgegeben Zahl k ein Flood-It Rätsel in höchstens k Schritten gelöst werden kann?

Lösung. Es gibt c^k Abfolgen von Farbwechseln, wenn c die Anzahl der Farben ist. Indem man alle diese Abfolgen durchprobiert, kann man entscheiden, ob das Problem eine Lösung hat. Das Problem ist also entscheidbar.

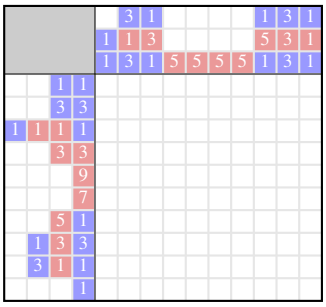
Um nachzuweisen, dass das Problem von einer nichtdeterministischen Turing-Maschine in polynomieller Zeit entschieden werden kann, suchen wir jetzt einen polynomiellen Verifizierer.

Als Lösungszertifikat für den Verifizierer verlangen wir die Abfolge von Farben, die für die Lösung des Flood-It-Rätsels nötig ist. Dann simulieren wir das Spiel, wobei wir in jedem Schritt die nm Felder überprüfen müssen und feststellen müssen, welche Felder jetzt das Gebiet ausmachen, also welche Felder mit der neuen Farbe eingefärbt werden müssen. Dazu sind maximal nm Prüfungen nötig. Nach maximal k Schritten müssen wir überprüfen, ob das ganze Feld mit der gleichen Farbe eingefärbt ist. Dazu sind nochmals nm Prüfungen nötig. Der Rechenaufwand für Verifikation ist also $O((k+1)nm)$. Wir wissen außerdem, dass $k \leq nm - 1$, also ist der Aufwand für die Verifikation $O((nm)^2)$ und damit sicher polynomiell. Somit ist nachgewiesen, dass das Problem in polynomieller Zeit von einer nichtdeterministischen Turing-Maschine entscheidbar ist. ○

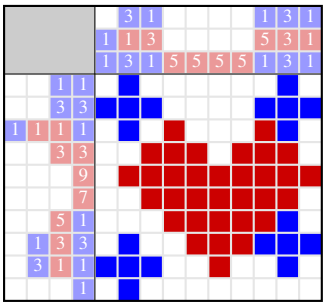
Diskussion. Clifford, Jalsenius, Montanaro und Sach haben in https://arxiv.org/pdf/1001.4420v3.pdf bewiesen, dass die oben beschriebene Version von Flood-It NP-vollständig ist. Sie haben auch weitere Version diskutiert, und approximative Lösungen mit polynomieller Laufzeit angegeben.

12.6. Die japanische Designerin Non Ishida hat im Jahre 1986 eine Art von Logikrätsel erfunden, die ihr zu Ehren Nonogramme genannt werden. In einem $n \times m$ Spielfeld sind die einzelnen Felder mit einer von mehreren möglichen Farben zu füllen. Am Rand des Feldes wird angegeben, wie lange Folgen benachbarter gleichfarbiger Felder in einer Zeile oder Spalte jeweils zu finden sind:

Aufgabe



Lösung



Die Vorgabe blau 1, rot 3, blau 3 in der drittuntersten Zeile bedeutet, dass in dieser Zeile zunächst ein einzelnes blaues Feld vorkommen muss, dann drei benachbarte rote Felder gefolgt von drei benachbarten blauen Feldern.

Das Problem *NONOGRAMM* ist also die Aufgabe, zu entscheiden, ob ein Nonogramm-Rätsel überhaupt lösbar ist.

a) Ist *NONOGRAMM* entscheidbar?

b) Kann eine nichtdeterministische Turing-Maschine in polynomieller Zeit entscheiden, ob ein Nonogramm gelöst werden kann?

Lösung. a) Wenn c verschiedene Farben gewählt werden können, dann gibt es für jedes der nm Felder $c+1$ Wahlmöglichkeiten. Das Feld kann daher auf $(c+1)^{nm}$ verschiedene Arten ausgfüllt werden. Indem man alle diese Möglichkeiten durchprobiert kann man immer entscheiden, ob die Aufgabe lösbar ist. Das Problem ist also entscheidbar.

b) Wir brauchen einen polynomiellen Entscheider. Als Lösungszertifikat verlangen wir die Belegung der Felder mit Farben. folgendes muss überprüft werden:

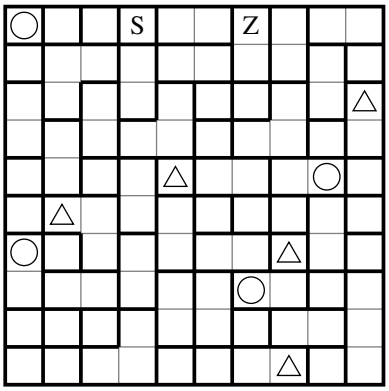
Was	Anzahl	Aufwand	Total
Jede Zeile durchlesen und überrüfen, ob die vorgeschriebenen Lauflängen jeder Farbe vorliegen.	n	$O(m)$	$O(mn)$
Jede Spalte durchlesen und überrüfen, ob die vorgeschriebenen Lauflängen jeder Farbe vorliegen.	m	$O(n)$	$O(mn)$
Total			$O(mn)$

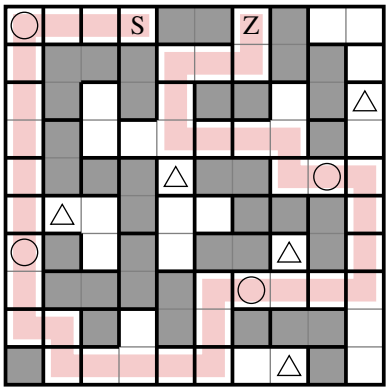
Dies zeigt, dass die Lösung in polynomieller Zeit in der Problemgröße $N = mn$ verifiziert werden kann. Es folgt, dass das Problem *NOMOGRAMM* in NP ist. ○

Diskussion. Nobuhisa Ueda und Tadaaki Nagao haben 1996 bewiesen, dass *NONOGRAMM* NP-vollständig ist: *NP-completeness Results for NONOGRAM via Parsimonious Reductions*, Technical Report, Departement of Computer Science, Tokyo Institue of Technology, doi:10.1.1.57.5277

12.7. *Nurimeizu* (塗り メイズ, von 塗り Malerei, und メイズ Labyrinth) ist ein japanisches Logikrätsel, welches auf einem n×m-Feld gespielt wird. Das Feld wird von fetten Linien in kleine Gebiete unterteilt. Der Spieler muss nun Gebiete grau einfärben, die keines der Zeichen S, Z, Dreieck oder Kreis enthalten. Die verbleibenden weißen Felder bilden ein Labyrinth. Weder die weißen noch die grauen Felder dürfen irgendwo einen Bereich der Größe 2×2 enthalten. Die weißen Felder müssen einen zusammenhängenden Bereich bilden. Sie bilden ein Wegnetz im Labyrinth, welches keinen Rundweg enthalten darf. Es muss einen Weg von S nach Z geben, der alle mit Kreis markierten Felder enthält, aber keines der mit Dreieck markierten Felder.

Das folgende Beispiel zeigt links das Rätsel und rechts die Lösung. Der gesuchte Weg ist rosa eingezeichnet.





Kann eine nichtdeterministische Turing-Maschine in polynomieller Zeit entscheiden, ob ein Nurimeizu-Rätsel lösbar ist?

Lösung. Das Problem ist sicher entscheidbar, indem man jede beliebige Einfärbung von Gebieten durchprobiert.

Es ist von einer nichtdeterministischen Turingmaschine in polynomieller Zeit genau dann entscheidbar, wenn es einen polynomiellen Verifizierer gibt. Als Lösungszertifikat c verlangen wir die grau einzufärbenden Felder und den rosa eingezeichneten Weg von S nach Z. Dann müssen folgende Verifikationen vorgenommen werden:

	Verifikation	Aufwand
1	Kein 2×2-Bereich im weißen Gebiet	$O(mn)$
2	Kein 2×2-Bereich im grauen Gebiet	$O(mn)$
3	Alle Kreise auf dem rosa Weg	$O(mn)$
4	Keine Dreiecke auf dem rosa Weg	$O(mn)$
5	Keine Kreise oder Dreiecke im grauen Gebiet	$O(mn)$
6	S und Z auf dem rosa Weg	$O(mn)$
7	Mit einem Markierungsalgorithmus, der ausgehend von S im weißen Bereich iterativ alle benachbarten Felder bereits markierter Felder markiert, feststellen, ob das weiße Gebiet zusammenhängend ist.	$O(m^2n^2)$
8	Überprüfen, ob das weiße Gebiet keine Rundwege enthält. Dies kann dadurch geschehen, indem man von jedem schwarzen Teilgebiet prüft, ob es von einem Weg von weißen Feldern umgeben ist.	$O(m^2n^2)$
	Total	$O(m^2n^2)$

Der Verifizierer hat also polynomielle Laufzeit. ○