Nonogramme lösen mithilfe von Backtracking

Marco Romanutti[1,2]

[1] *Fachhochschule Nordwestschweiz FHNW, Brugg*
[2] `Effiziente Algorithmen, Klasse 5ld`

Ein Nonogramm ist ein Logikrätsel, bei dem ein Gitter so einzufärben ist, dass für jede Zeile und jede Spalte die Reihenfolge und die Länge der gefärbten Blöcke den angegebenen Zahlen entsprechen. Zwischen zwei Blöcken muss immer mindestens ein leeres Feld vorhanden sein. Im folgenden Dokument wird grob umschrieben, wir das Problem mithilfe von Backtracking gelöst werden konnte.

1 Lösungsansatz

Beim verwendeten Lösungsansatz werden, basierend auf den Blockgrössenangaben für die Zeilen aus dem Input-File `nonogramm.in`, mit der Funktion `calculatePermutations` alle möglichen Permutationen für jede Zeile berechnet. Für jede dieser Permutationen wird anschliessend geprüft, ob die Permutation eine mögliche Lösung ist. Dazu wird in der Funktion `getExpectedRow` anhand der vorigen Zeile und den Blockgrössenangaben für die Spalten geprüft, welcher Wert eine Zelle haben muss. Es werden dabei folgende Situationen unterschieden:

- **Zelle in voriger Zeile ist ausgefüllt:** Falls der aktuelle Block weniger Zellen umfasst, als in den Blockgrössenangaben angegeben, muss die aktuelle Zelle auch ausgefüllt werden. Ansonsten muss die Zelle leer bleiben, da zwischen den Blöcken immer mindestens eine leere Zelle folgt.
- **Zelle in voriger Zelle ist leer:** In dieser Situation können wir idR.[1] nichts über den Wert der Zelle aussagen. Die Zelle wird als `unknown` markiert.

Für das Lösen des Nonogramms wird Backtracking verwendet. Der Einstieg erfolgt über die Funktion `solve`. Permutationen, welche gemäss dem Kontrolle mit den Blockgrössenangaben der Spalten nicht gültig sind werden nicht weiter verfolgt und entsprechende Äste im Lösungsbaum „abgeschnitten“.

Beim Erarbeiten der Lösung wurde zu Beginn ein naiver Ansatz gewählt, welcher für jedes Feld prüfte, ob die Lösung mit weisser, resp. schwarzer Zelle noch gültig ist. Dieser Ansatz stellte sich als äusserst langsam heraus. Eine erste Optimierung bestand darin, anstatt aller Zellen nur noch alle Zeilen-Permutationen mit allen Spalten-Permutationen zu kombinieren. Doch auch diese Lösung hatte bei grösseren Nonogrammen lange Laufzeiten zur Folge. Erst der eingangs beschriebene Ansatz, bei welchem einzig die Zeilen-Permutationen (und diese anhand der vorigen Zeilen auf die weiterhin möglichen Permutationen reduziert) betrachtet werden, brachte zufriedenstellende Laufzeiten.

2 Anleitung Software

Die Blockgrössenangaben für das Nonogramm werden von der Datei `nonogramm.in` gelesen. Die Ausführung kann gestartet werden, indem die `main`-Funktion in der Klasse `NonogramSolver` aufgerufen wird. Die Resultate werden in die Datei `nonogramm.out` geschrieben.

3 Testinputs und -outputs

Für das Testen der Funktionalität und Performance des Algorithmus wurden nachfolgende Inputs verwendet. Die dazugehörigen Dateien befinden sich im `resources`-Verzeichnis.

- **Nonogramm Rabitt (a)**
  Dieses Nonogramm wurde zusammen mit der Aufgabenstellung abgegeben. Es wurde verwendet

[1] Ausnahmen, falls z.B. Ende der Zeilen erreicht wurde

um die Funktionsweise und die Performance bei grossen Nonogrammen zu testen. Das gelöste Nonogramm ist in Abbildung 1 abgebildet.

- **Nonogramm Pelican (b)**
  Das gelöste Nonogramm stellt einen Pelican dar (vgl. Abbildung 2). Es wurde eingesetzt, um die Funktionsweise des Algorithmus auf visueller Basis zu überprüfen.
- **Nonogramm Chaplin (c)**
  In diesem Nonogramm ist nach dem Lösen das Gesicht von Charlie Chaplin zu erkennen (vgl. Abbildung 4). Es wurde eingesetzt, um das Verhalten bei vielen möglichen Permutationen zu testen.
- **Nonogramm Multiple Solutions (d)**
  In diesem Nonogramm sind mehrere Lösungen möglich - damit wurde das korrekte Verhalten des Algorithmus bei mehreren möglichen Lösungen überprüft. Abbildung 5 zeigt die beiden möglichen Lösungen.
- **Nonogramm No Solution (e)**
  Dieses Nonogramm hat keine gültige Lösung. Es gibt deshalb auch keine Abbildung dazu. Es wurde eingesetzt, um den Fall von keiner gültigen Lösung zu prüfen.

Figure 1: *Rabbit*

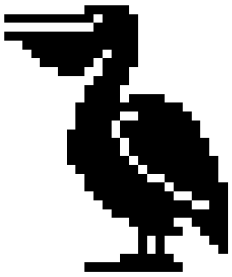

Figure 2: *Pelican*

Figure 3: *Chaplin*

Figure 4: *Mehrere Lösungen*

4 Performance

Table 1: *Gegenüberstellung der Laufzeit von verschiedenen Nonogrammen (vgl. Testinputs und -outputs)*

Nonogramm	Dimensionen	Laufzeit
Rabbit (a)	35 x 25	1.9 s
Pelican (b)	30 x 25	56 ms
Chaplin (c)	30 x 30	537 ms
Multiple Solutions (d)	7 x 8	30 ms
No Solution (e)	5 x 5	1 ms

References

[1] Wikipedia: Nonogramm, `https://de.wikipedia.org/wiki/Nonogramm`

[2] Wikipedia: Backtracking Algorithmus, `https://de.wikipedia.org/wiki/Backtracking`

[3] Nonograms Katana: Templates and solutions `https://nonogramskatana.wordpress.com/`