

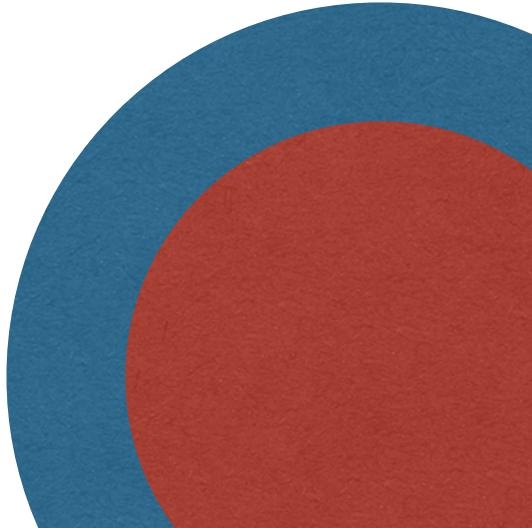
42. Bundeswettbewerb Informatik



Team Deutscher Qualitätscode

Max Eckstein, Nina Hochecker

Team-ID: 00538



Aufgabe 1: Arukone

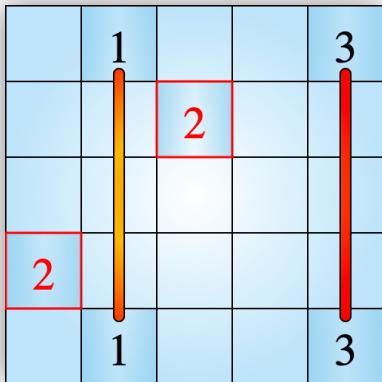
Inhaltsverzeichnis

Lösungsidee	2
Umsetzung	2
Beispiele	3
Quellcode	5

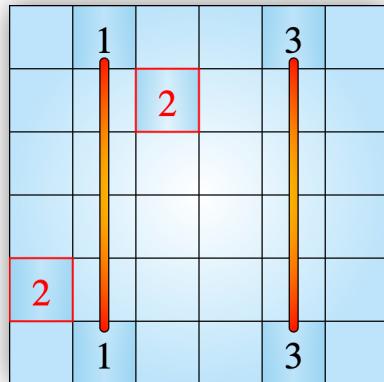
Lösungsidee

Der Algorithmus auf der BwInf-Website weist bestimmte Schwächen beim Lösen von Arukone-Rätseln auf. Unsere Idee ist es, eine dieser Schwächen zu finden, und diese auszunutzen, um die Rätsel für den Algorithmus unlösbar zu machen. Eine Schwäche, die wir ausfindig machen konnten, war die Folgende:

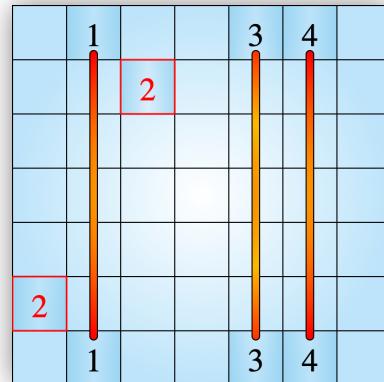
In den beiden abgebildeten Fällen kann man sehen, dass der Algorithmus die Einser auf direktem Weg verbindet, und anschließend daran scheitert, die Zweier zu verbinden. Diese Fehlfunktion kann ausgenutzt werden, um größere Arukone-Rätsel zu generieren, die für den Algorithmus nicht lösbar sind.



5x5 Feld



6x6 Feld



7x7 Feld

Umsetzung

Zuerst haben wir die Größe des Spielfeldes **n** und die Anzahl der zu verbindenden Zahlenpaare **k** (wobei $k = n/2$ aufgerundet ist) definiert.

n kann über die Kommandozeile als Parameter angefügt werden wie folgt, ist sonst standardmäßig allerdings 5: [./main 7](#) oder [./main 25](#).

Die Variable **Spielfeld** ist ein zweidimensionales Integer-Array der Größe $n \times n$, welches anfangs mit Nullen aufgefüllt wird. Die beiden Zahlenpaare, die das Arukone-Rätsel für den Algorithmus unlösbar machen (die Einser und Zweier) werden nach einem bestimmten Muster in das Spielfeld geschrieben:

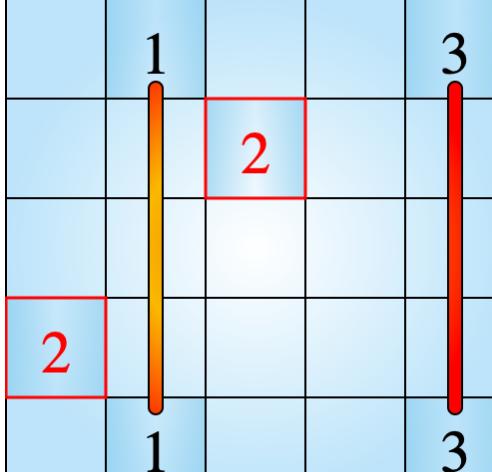
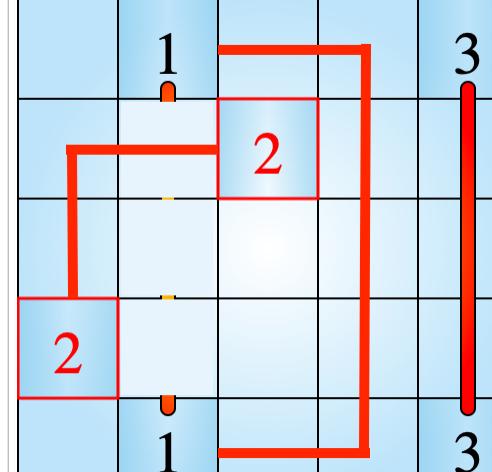
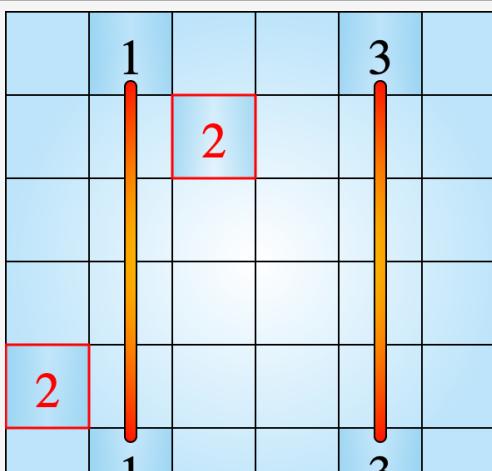
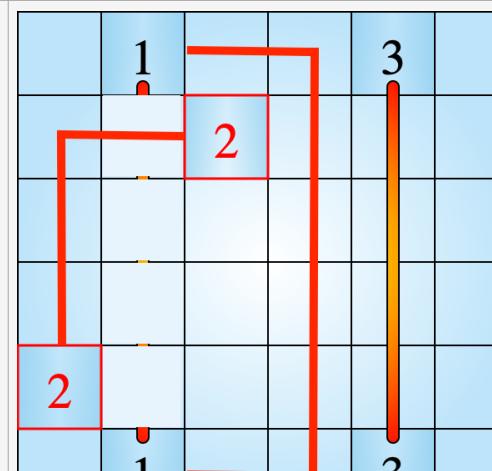
Die Zahl?	Position
Erster Einser	Ganz oben, 1 Feld entfernt vom linken Rand
Zweiter Einser	Ganz unten, 1 Feld entfernt vom linken Rand
Erster Zweier	1 Feld entfernt vom oberen Rand, 2 Felder entfernt vom Linken Rand
Zweiter Zweier	1 Feld entfernt vom unteren Rand, am linken Rand

Damit das Arukone-Rätsel wie vorgegeben k Zahlenpaare hat, müssen allerdings noch $k-2$ Zahlenpaare eingefügt werden. Dazu werden ab der fünften Spalte die Zahlenpaare hinzugefügt, wobei die eine Zahl in der ersten Zeile und die andere Zahl in der letzten Zeile der Spalte eingefügt wird, siehe Abbildungen oben.

Für ein bisschen Variation und nicht nur identische Rätsel wird die Reihenfolge der Zahlenpaare innerhalb des Feldes vertauscht. Außerdem wird mithilfe einer Matrix-Rotation das Spielfeld auch noch um eine zufällige Anzahl von Malen um 90° gedreht, wodurch die Diversität der Rätsel vergrößert wird.

Abschließend wird das zweidimensionale Array Spielfeld noch zu einem String formatiert, welcher die gewünschte Struktur eines Arukone-Feldes widerspiegelt. Dieser wird in eine Datei geschrieben, welche sich im gleichen Verzeichnis befindet wie das Executable, und den Namen arukonen n .txt hat.

Beispiele

N	Bwlnf Algorithmus	Lösung
5		
6		

Quellcode

```
import Foundation

// Seitenlänge des Spielfeldes
let dateiNummer: String = CommandLine.arguments.last ?? "-"
let n: Int = Int(dateiNummer) ?? 5

guard n >= 4 else {
    print("N ist zu klein")
    exit(EXIT_FAILURE)
}

// Anzahl der Zahlen auf dem Feld
let k: Int = Int(ceil(Double(n)/2.0))

// Generiertes Spiel
var generiertesSpiel: String = "\n\n"

// Spielfeld, bei dem anfangs alle Werte auf 0 sind (es ist noch leer)
var spielfeld = [[Int]](repeating: [Int](repeating: 0, count: n), count: n)

// Es für den Algorithmus unlösbar machen
spielfeld[0][1] = 1
spielfeld[1][2] = 2
spielfeld[n-1][1] = 1
spielfeld[n-2][0] = 2

if k != 2 {
    // Die restlichen Zahlen in graden Linien von oben nach unten einarbeiten, damit die Mindestanzahl der Zahlen gegeben ist
    var restlicheZahlen = Array(3...k)
    restlicheZahlen.shuffle()
    for i in restlicheZahlen.enumerated() {
        spielfeld[0][i.offset+4] = i.element
        spielfeld[n-1][i.offset+4] = i.element
    }
}

// Generiertes Spiel drehen
var gedrehtesSpiel = spielfeld
for _ in 0...Int.random(in: 1..<4) {
    spielfeld = gedrehtesSpiel
    for i in 0...(n-1) {
        for j in 0...(n-1) {
            gedrehtesSpiel[i][j] = spielfeld[n - j - 1][i];
        }
    }
}

// Output ergänzen
for i in gedrehtesSpiel {
    var zeile = "\n"
    for ii in i.enumerated() {
        zeile.append((ii.offset != 0 ? " " : "") + String(ii.element))
    }
    generiertesSpiel.append(zeile)
}

// Output in Datei schreiben
if (FileManager.default.createFile(atPath: "./arukone\n.txt", contents:
generiertesSpiel.data(using: .utf8), attributes: nil)) {
    print("Erfolg")
} else {
    print("Datei konnte nicht beschrieben werden... :(")
}
```

