Aufgabe 4: Nandu

Team-ID: 01005

Team: Jonathan

Bearbeiter/-innen dieser Aufgabe: Jonathan Salomo

19. November 2023

Inhaltsverzeichnis

Lösungsidee

Um die Aufgabe zu lösen, habe ich mir überlegt, dass ich erst ein Programm schreibe, welches die bereitgestellten Daten soweit vorbereitet und in die richtige Form bringt, dass ich sie danach mit zwei weiteren Programmen weiterverarbeiten kann. Hierbei übernimmt das erste die Änderung der Eingangskriterien, also welche der Eingangs-LEDs an und welche aus sind, und das zweite die Errechnung der Zustände der Ausgangs-LEDs. Als letztes verwende ich dann mein Ausgabeprogramm an, dass ich schon für die Junioraufgaben programmiert hatte, um meine Lösung nicht nur auf die Konsole, sondern auch in eine separate Datei zu schreiben.

Um die Lösung jeder mögliche Kombination zu errechnen, beginne ich damit zu errechnen wie viele Möglichkeiten es gibt. Ich rechne also 2 hoch die Anzahl der Eingangs-LEDs, da jede dieser LEDs zwei Zustände haben kann (an / aus) und es so mit jeder zusätzlichen LED doppelt so viele Kombinationen gibt wie zuvor. Nun errechne ich jeden dieser Zustände, indem ich sie durchnummeriere und jeweils die Binärzahl der Nummer des Zustandes errechne, indem ich erst den Rest einer ganzzahligen Division durch zwei von rechts zu meiner Binärzahl hinzufüge und dann den Ausgangswert durch zwei teile und dies solange wiederhole bis dieser Wert kleiner oder gleich null ist. Dann schaue ich mir eine Stelle dieser Binärzahl nach der anderen an und stelle die jeweilige Eingangs-LED mit derselben Nummer, je nachdem ob diese Stelle eine 1 oder eine null ist, auf an oder aus. Danach schaue ich mir für jede Zeile der Konstruktion jede LED (in der ersten Zeile die Eingangs LEDs) an, welche in der Zeile davor leuchtet. Für jede dieser LEDs betrachte ich mir als erstes den Baustein, der direkt darunter ist, wenn dieser blau ist, merke ich mir für die nächste Zeile, dass die LED an dieser Stelle an ist. Sollt dieser weiß sein, überprüfe ich erst in welche Richtung der Baustein weitergeht und dann ob in die andere Hälfte des Bausteins auch Licht scheint. Je nachdem was diese Überprüfungen ergeben, merke ich mir nach den Regeln des weißen

Steines, ob und welche LEDs für die nächste Zeile an sind. Das Gleiche mache ich dann noch für den Sensorteil des roten Bausteines, wobei ich hier nach den Regeln von diesem nicht beachte ob Licht in die sensorlose Hälfte des Bausteins fällt. Im Anschluss überprüfe ich noch ob irgendwo in der Zeile rote oder weiße Bausteine sind, die sich nicht unter einer leuchtenden LED befinden und prüfe jeweils in welche Richtung diese sich fortsetzen und welche der LEDs für die nächste Zeile folglich nach ihren jeweiligen Regeln an sein müssen. Danach wiederhole ich dies für jede weitere Zeile und gebe, wenn ich am Ende angekommen bin, die Lösung für die jeweilige Konstellation aus. Diesen Prozess wiederhole ich für jede weitere mögliche Konstellation, indem ich wie oben beschrieben, diese aus der Zahl des Durchlaufes mit dem Maximum an Durchläufen der vorher berechneten Anzahl der benötigten Durchläufe berechne.

Umsetzung

Hier beginne ich damit, dass ich die Größe der Konstruktion einlese. Danach erstelle ich mit Hilfe dieser Information ein Array von Strings, in das ich die Konstruktion abspeichere. Dies mache ich unter Zuhilfenahme eines Buffered Readers sowie der .split Funktion, wobei ich vor diesem Schritt noch jede Dopplung von Leerzeichen durch ein einziges ersetze, damit ich keine leeren Werte in meine Array einlese. Danach errechne ich mit einer for-Schleife sowie einem if clause, der für jeden Eintrag in der ersten Zeile überprüft, ob dieser mit einem „L“ beginnt und wenn dies der Fall, ist die Anzahl an LEDs um eins erhöht die Anzahl der LEDs. Danach erstelle ich noch ein Objekt der Klasse MapGenerator mit der vorbereiteten Matrix, der Konstruktion, wobei der MapGenerator die erste Zeile von dieser in ein Array von Booleans übersetzt, wobei an den Koordinaten der LEDs der Wert auf true gesetzt wird.

Der komplette weitere Code befindet sich nun in einer for-Schleife, welche mit null beginnt ihre Durchläufe zu zählen und sich so oft wiederholt bis die Anzahl ihrer Durchläufe 2 hoch die Anzahl der LEDs übersteigt. Also macht sie einen Durchgang für jede mögliche Kombination. In dieser Schleife beginne ich damit ein Objekt der Klasse LEDberechner zu initialisieren, wobei ich die Größenangaben der Konstruktion sowie die erste Kombination von Eingangswerten in Form eines Arrays aus Booleans, übergebe, welche ich errechne, indem ich die Funktion getNextCombination, mit der Nummer des Durchlaufes der vorher beschriebenen Schleife auf dem mapgenerator aufrufe.

Der mapgenerator errechnet dieses Array, indem er die ihm übergebene Nummer des Durchlaufes zu einer Binärzahl macht. Dies wird erreicht, indem die Eingangszahl solange erst ganzzahlig mit dem modulo Operator durch 2 geteilt und der Rest jeweils einem vorher erstellten Array von hinten hinzufügt wird und dann der Eingangswert durch 2 geteilt und die Koordinate des Eintrages in das Array für die Binärzahl um eins verringert wird bis der Anfangswert kleiner oder gleich null ist. Diese Binärzahl wird dann in ein Array von Booleans übersetzt, das die passende Länge hat, indem für die Länge von dem Ausgabe Array (welches bei der Initialisierung mit der Breite der Konstruktion erstellt wurde) in das auch bei der Initialisierung erstellte Booleanarray guckt, welches die Koordinaten der LEDs enthält und für jede dieser den passenden Wert ins Ausgabearray

schreibt, je nachdem ob der Wert im Array der Binärzahl dessen Koordinate mit der Nummer der LED übereinstimmt eine 1(true) oder eine 0 (false) ist.

Dieses Array wird dem LEDberechner übergeben, welcher sich zwei Boolean Arrays, die lastonLEDs und die onLEDs, mit der Breite der Konstruktion erstellt. In die onLEDs schreibt er nur false Werte und in die lastonLEDs überträgt er die Werte des übergebenen Arrays.

Danach wird die Funktion berechneonLEDs mit der passenden Zeile der Konstruktions-Matrix für jede Zeile des Konstruktion aufgerufen.

In dieser Funktion durchläuft der LEDberechner immer eine große for-Schleife, welche einen Durchgang für jeden Eintrag in der übergebenen Zeile macht. Für jeden dieser Einträge wird nun in dieser Schleife durch if und if else Bedingungen geprüft, ob der Wert an dieser Stelle im Array lastonLEDs true ist und wenn dies der Fall ist, ob einer der Steine in dieser Zeile dafür sorgt, dass bestimmte Werte in dem Array onLEDs true sein müssen. Dann wird noch überprüft, ob einer der Steine in der Zeile auch ohne LED, die auf ihn scheint, nach seinen jeweiligen Regeln dafür sorgt, dass ein bestimmter Wert in dem Array onLEDs true sein muss. Die Vergleiche werden hier immer mit der Funktion .equals durchgefürt. Am Ende werden noch die onLEDs in die lastonLEDs kopiert und alle Werte in onLEDs wieder auf false gesetzt.

Dies wird für jede Zeile in der Konstruktion wiederholt. Danach wird abschließend die Anfangskonstellation sowie die errechnete Lösung auf der Konsole ausgegeben und jeweils auch in eine Datei geschrieben.

Beispiele

Eingabe Beispiel 1:

4 6
X Q1 Q2 X
X W W X
r R R r
X B B X
X W W X
X L1 L2 X

Ausgabe:

Ergebnis:
Q1: Aus
Q2: Aus
L1: An
L2: An

Ergebnis:
Q1: Aus
Q2: An
L1: An
L2: An
Ergebnis:
Q1: An
Q2: Aus
L1: An
L2: An
Ergebnis:
Q1: An
Q2: An
L1: Aus
L2: Aus
Eingabe Beispiel 2:

```
6 8
X  X  Q1 Q2 X  X
X  X  B  B  X  X
X  X  W  W  X  X
X  r  R  R  r  X
r  R  W  W  R  r
X  W  W  B  B  X
X  X  B  B  X  X
X  X  L1 L2 X  X
```

Ausgabe:

Ergebnis:
Q1: Aus
Q2: Aus
L1: Aus
L2: An
Ergebnis:
Q1: Aus
Q2: An
L1: Aus
L2: An
Ergebnis:
Q1: An
Q2: Aus
L1: Aus
L2: An

Ergebnis:
Q1: An
Q2: An
L1: An
L2: Aus
Eingabe Beispiel 3:

```
8 7
X  X  Q1 Q2 X  X  Q3 X
X  X  B  B  X  r  R  X
X  r  R  R  r  W  W  X
X  W  W  B  B  W  W  X
r  R  B  B  B  B  R  r
X  B  B  W  W  B  B  X
X  L1 L2 X  L3 L4 X  X
```

Ausgabe:

Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
L1: An
L2: Aus
L3: Aus
L4: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
L1: An
L2: Aus
L3: Aus
L4: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
L1: An
L2: Aus
L3: An
L4: An
Ergebnis:
Q1: Aus
Q2: An
Q3: An

L1: An
L2: Aus
L3: An
L4: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
L1: Aus
L2: An
L3: Aus
L4: An
Ergebnis:
Q1: An
Q2: Aus
Q3: An
L1: Aus
L2: An
L3: Aus
L4: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
L1: Aus
L2: An
L3: An
L4: An
Ergebnis:
Q1: An
Q2: An
Q3: An
L1: Aus
L2: An
L3: An
L4: Aus

Eingabe Beispiel 4:

```
8 8
X  X  Q1 Q2 Q3 Q4 X  X
X  r  R  B  B  R  r  X
X  X  W  W  W  W  X  X
X  r  R  B  B  R  r  X
```

```
r  R  W  W  B  B  R  r
X  W  W  W  W  W  W  X
X  X  B  B  X  X  X  X
X  X  L1 L2 X  X  X  X
```

Ausgabe:

Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: Aus
L1: Aus
L2: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: An
L1: Aus
L2: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: Aus
L1: Aus
L2: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: An
L1: Aus
L2: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: Aus
L1: An
L2: An
Ergebnis:
Q1: Aus

Q2: An
Q3: Aus
Q4: An
L1: An
L2: An
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: Aus
L1: An
L2: An
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: An
L1: An
L2: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: Aus
L1: Aus
L2: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: An
L1: Aus
L2: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: Aus
L1: Aus
L2: Aus
Ergebnis:
Q1: An

Q2: Aus
Q3: An
Q4: An
L1: Aus
L2: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: Aus
L1: Aus
L2: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: An
L1: Aus
L2: Aus
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: Aus
L1: Aus
L2: An
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: An
L1: Aus
L2: Aus
Eingabe Beispiel 5:

```
22 14
X  X  X  X  X  Q1 Q2 X  X  Q3 Q4 X  X  Q5 Q6 X  X  X  X  X  X  X
X  X  X  X  X  R  r  X  X  r  R  X  X  R  r  X  X  X  X  X  X  X
X  X  X  X  r  R  R  r  r  R  R  r  r  R  R  r  X  X  X  X  X  X
X  X  X  X  W  W  B  B  W  W  B  B  W  W  B  B  X  X  X  X  X  X
X  X  X  r  R  B  B  B  B  B  B  B  B  B  B  R  r  X  X  X  X  X
X  X  r  R  B  B  X  B  B  B  B  B  B  B  B  X  R  r  X  X  X  X
X  r  R  B  B  R  r  B  B  B  B  B  B  B  B  r  R  R  r  X  X  X
X  X  B  B  W  W  B  B  X  B  B  B  B  B  B  W  W  B  B  X  X  X
X  r  R  W  W  W  W  R  r  W  W  W  W  W  W  W  W  W  W  X  X  X
X  W  W  B  B  W  W  X  B  B  W  W  B  B  W  W  B  B  R  r  X  X
```

```
r  R  B  B  B  B  R  r  X  B  B  B  B  B  B  B  B  B  B  R  r  X
X  B  B  B  B  B  B  R  r  B  B  B  B  B  B  B  B  B  B  W  W  X
r  R  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  R  r
L1 X  X  X  X L2  X  X  X  X L3  X  L4 X  X  X  X  X  X  X  X  L5
```

Ausgabe:

Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: Aus
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: Aus
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: Aus
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:

Q1: Aus
Q2: Aus
Q3: Aus
Q4: Aus
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: An
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: An
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: An
Q5: An
Q6: Aus

L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: Aus
Q4: An
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: Aus
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: Aus
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:

Q1: Aus
Q2: Aus
Q3: An
Q4: Aus
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: Aus
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: An
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: An
Q5: Aus
Q6: An

L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: An
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: Aus
Q3: An
Q4: An
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: Aus
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:

Q1: Aus
Q2: An
Q3: Aus
Q4: Aus
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: Aus
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: Aus
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: An
Q5: Aus
Q6: Aus

L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: An
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: An
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: Aus
Q4: An
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:

Q1: Aus
Q2: An
Q3: An
Q4: Aus
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: Aus
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: Aus
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: Aus
Q5: An
Q6: An

L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: An
Q5: Aus
Q6: Aus
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: An
Q5: Aus
Q6: An
L1: Aus
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: Aus
Q2: An
Q3: An
Q4: An
Q5: An
Q6: Aus
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:

Q1: Aus
Q2: An
Q3: An
Q4: An
Q5: An
Q6: An
L1: Aus
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: Aus
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: Aus
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: Aus
Q5: An
Q6: Aus

L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: Aus
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: An
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: An
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:

Q1: An
Q2: Aus
Q3: Aus
Q4: An
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: Aus
Q4: An
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: Aus
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: Aus
Q5: Aus
Q6: An

L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: Aus
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: Aus
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: An
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:

Q1: An
Q2: Aus
Q3: An
Q4: An
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: An
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: Aus
Q3: An
Q4: An
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: Aus
Q5: Aus
Q6: Aus

L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: Aus
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: Aus
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: Aus
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:

Q1: An
Q2: An
Q3: Aus
Q4: An
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: An
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: An
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: Aus
Q4: An
Q5: An
Q6: An

L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: Aus
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: Aus
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: Aus
Q5: An
Q6: Aus
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:

Q1: An
Q2: An
Q3: An
Q4: Aus
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: An
Q5: Aus
Q6: Aus
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: An
Q5: Aus
Q6: An
L1: An
L2: Aus
L3: An
L4: Aus
L5: Aus
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: An
Q5: An
Q6: Aus

L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Ergebnis:
Q1: An
Q2: An
Q3: An
Q4: An
Q5: An
Q6: An
L1: An
L2: Aus
L3: Aus
L4: An
L5: An
Eingabe eigenes Beispiel 1:

4 6
Q1 X Q2 X
W W W W
r R R r
X B B X
X W W X
X L1 L2 X
Ausgabe:
Ergebnis:
Q1: Aus
Q2: Aus
L1: An
L2: An
Ergebnis:
Q1: Aus
Q2: An
L1: An
L2: An
Ergebnis:
Q1: An
Q2: Aus
L1: An
L2: An
Ergebnis:

Q1: An
Q2: An
L1: An
L2: An

Kommentar:
Dieses Beispiel deckt den Sonderfall ab, dass eine Lampe und ein Stein mit zwei relevanten Hälften sich am Rand der Konstruktions-Matrix befindet und somit bei der Abfrage wo sich die zweite Hälfte des Steines befindet eine ArrayIndexOutOfBoundsException entstehen kann. Dies lässt sich jedoch, wie ich es in meinem Programm gemacht habe, einfach mit der zusätzlichen Überprüfung der Koordinate in Bezug auf die Größe des Arrays abfangen.

Quellcode

Der folgende Code ist gekürzt und nicht funktionsfähig:

```
public class LEDs {
     public static void main(String[] args) throws Exception {

          int[] größe = fileReaderx.readLineToArray(dateipfad,1); //Liest die
Breite und Höhe aus.

          String[][] matrix = fileReaderx.redarray(dateipfad, 1, größe[1],
größe[0], größe[1]); //Liest die kommplette Konstruktion in ein
zweidimensonales Array aus.

          int LEDs = 0;
          for(int i = 0; i < größe[0]; i++){
               if (matrix[0][i].startsWith("Q")){
                    LEDs ++;
               }
          }

          for (int q = 0; q < (int) Math.pow(2,LEDs);q++){

               LEDberechner leDberechner = new LEDberechner(größe[0],
größe[1], mapgenarator.getNextCombination(q)); //Erstellt mit den
übergebenen Werten die benötigten Arrays in der passenden Größe.

               for(int i = 1; i < größe[1]; i++){
                    leDberechner.berechneonLEDs(matrix[i]); //Berechnent die
leuchtenden LEDs Zeile für Zeile.
               }
      // Hier wird im echten Programm die Lösung ausgegeben
          }
```

```
     }
}


public class MapGenerator {
     boolean[] inputArray;
     boolean[] currentCombination;

 // hier ist im echten Programm der Konstrucktor welcher die Arrays usw.
Inizialisiert


     public boolean[] getNextCombination(int dez) {
          int i = LEDnummer-1;

          while (dez > 0) {
               int rest = dez % 2;
               bin[i] = rest;
               dez = dez/2;
               i--;
          }

          int g = 0;
          for (int r = 0; r < currentCombination.length; r++){
               if (inputArray[r]){
                    if (bin[g] == 1){
                         currentCombination[r] = true;
                    } else{
                         currentCombination[r] = false;
                    }
                    g++;
               }
          }

          return currentCombination.clone();
     }
}

public class LEDberechner {
     Boolean[] lastonLEDs;
     Boolean[] onLEDs;

 // hier ist im echten Programm der Konstrucktor welcher die Arrays usw.
Inizialisiert

     public void berechneonLEDs(String[] zeile){
          for (int i = 0; i < zeile.length; i++) {
```

```
            if (lastonLEDs[i]) {
                if (zeile[i].equals("B")) {
                    onLEDs[i] = true;
                } else if (zeile[i].equals("W")) {
                    if (i > 0 && zeile[i - 1].equals("W")) {
                        if (i - 1 >= 0 && lastonLEDs[i - 1]) {

                        } else if (i - 1 >= 0 && !lastonLEDs[i - 1]) {
                            onLEDs[i - 1] = true;
                            onLEDs[i] = true;
                        }
                    } else if (i < zeile.length - 1 && zeile[i +
1].equals("W")) {
                        if (i + 1 < zeile.length && lastonLEDs[i + 1]) {

                        } else if (i + 1 < zeile.length && !lastonLEDs[i
+ 1]) {
                            onLEDs[i + 1] = true;
                            onLEDs[i] = true;
                        }
                    }
                } else if (zeile[i].equals("r")) {
                    if (i > 0 && zeile[i - 1].equals("R")) {
                        if (i - 1 >= 0 && lastonLEDs[i - 1]) {

                        } else if (i - 1 >= 0 && !lastonLEDs[i - 1]) {
                            onLEDs[i - 1] = true;
                            onLEDs[i] = true;
                        }
                    } else if (i < zeile.length - 1 && zeile[i +
1].equals("R")) {
                        if (i + 1 < zeile.length && lastonLEDs[i + 1]) {

                        } else if (i + 1 < zeile.length && !lastonLEDs[i
+ 1]) {
                            onLEDs[i + 1] = true;
                            onLEDs[i] = true;
                        }
                    }
                } else if (zeile[i].startsWith("L")) {
                    onLEDs[i] = true;
                }
            }

            if (i < zeile.length && zeile[i].equals("R") && !
lastonLEDs[i]) {
                if (i > 0 && zeile[i - 1].equals("r")) {
```

```
                    onLEDs[i - 1] = true;
                    onLEDs[i] = true;
                } else if (i < zeile.length - 1 && zeile[i +
1].equals("r")) {
                    onLEDs[i + 1] = true;
                    onLEDs[i] = true;
                }
            } else if (i < zeile.length && zeile[i].equals("W") && !
lastonLEDs[i]) {
                if (i > 0 && zeile[i - 1].equals("W") && !lastonLEDs[i -
1]) {
                    onLEDs[i - 1] = true;
                    onLEDs[i] = true;
                } else if (i < zeile.length - 1 && zeile[i +
1].equals("W") && !lastonLEDs[i + 1]) {
                    onLEDs[i + 1] = true;
                    onLEDs[i] = true;
                }
            }
        }
        lastonLEDs = onLEDs.clone();
        }

    }
```