Aufgabe 4: Nadu

Team-ID: 01036

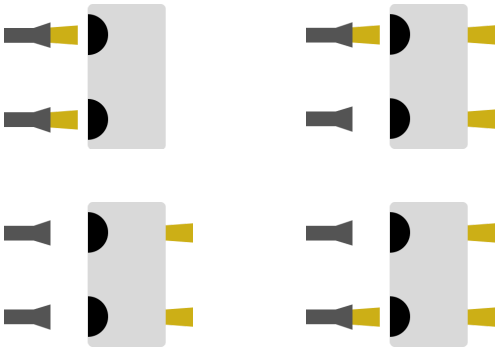
Team: Error260

Von: Weismeier Florian

Lösungsidee

Die Aufgabe besteht darin, den Endstand des Lichtes zu bestimmen, der durch die Konfiguration der Steine und Startlichter festgelegt wird. Die Aufgabe für potenziell hunderte Steine kann man vereinfachen, indem man sie auf die wiederholte Interaktion zwischen einzelnen Blöcken und den Block hinter ihnen reduziert. So betrachtet man z.B.:

Q1	Q2	Weißer Block ausgabe:
F	F	T
F	T	T
T	F	T
T	T	F

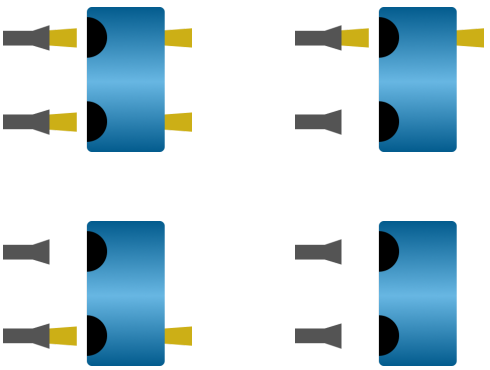


Q1	Roter Block ausgabe:
F	T
T	F

Quelle 2 spielt bei dem Beispiel keine Rolle. Der Rote Block hat nur einen Sensor, kann aber gedreht werden

Blauer Block

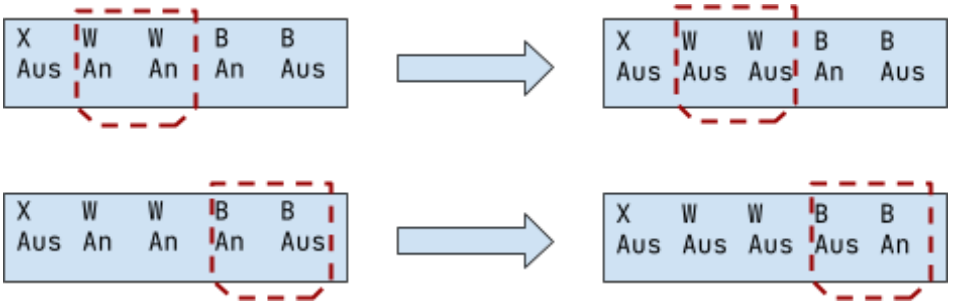
Q1	Q2	Ausgabe L1	Ausgabe L2
F	F	F	F
F	T	F	T
T	F	T	F
T	T	T	T



Wenn wir Bausteine als eine Art Logik Gatter sehen, deren Eingänge eine Referenz zu den Ausgängen anderer Gatter sind, erhalten wir eine Modulare Struktur, die für beliebig viele Steine anwendbar ist.

Umsetzung

Als erstes Kopieren wir die Anordnung in jeder Zeile in einen Array, und speichern uns dann die Position der Anfangslampen. Von da an “scannen” wir die Blöcke von oben nach unten, Zeile nach Zeile, indem wir die aktivierten Lichter der Letzten Zeile mit dem der neuen ersetzen.



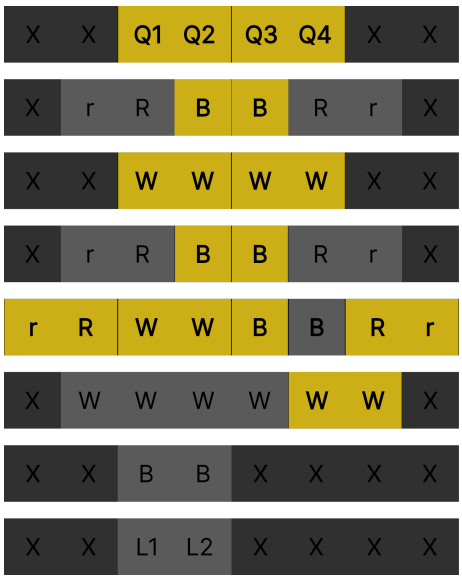
Wenn wir dies nun für jede Zeile wiederholen, erhalten wir immer den korrekten Status des Lichts an jeder Stelle. Nun müssen wir lediglich in der letzten Zeile diesen Wert an den Ausgängen ablesen und erhalten unsere Antwort.

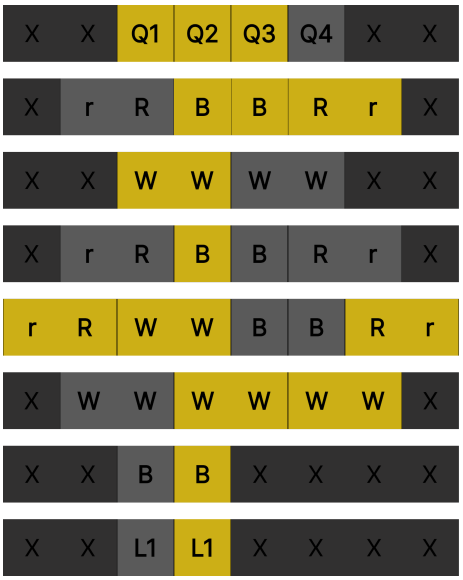
Laufzeit

Die Laufzeit hängt von der Anzahl der Blöcke, und damit ungefähr von dem Quadrat der Seitenlänge ab, als $O(n^2)$. In dem größten Beispiel hat es eine Laufzeit von $240ms$.

Beispiele

Dieses beispiel Zeigt farblich die Handlung des Algorithmus Zeile für Zeile





Quellcode

```
// Specify the path to your text file
const filePath = './test.txt';

// Read the file asynchronously line by line
const readStream = fs.createReadStream(filePath, 'utf8');

// Variables //
let width = 0
let height = 0
let countOfSources = 0
let count = 0
const start = Date.now(); // runtime measure

let currentLine // Array of the letters in the current line
let originalValues = [] // the start light (just the array of the Q's)
let defaultLights = [] // light state of every round
let currentLight = [] // the light after every row
let finalResult = [] // here is an array in an array with all the solutions

// Function to do the Light sources //
function generateCombinations(values) {
	hier werden alle Kombinationen der Q’s mit true oder falsch durchgerechnet und dann auch gespeichert Formel= 2^n
}
const firstLine = () ⇒ {
	if (count ⩵ 0) {
	for (let j = 0; j < width; j++) {
		if (currentLine[j].includes("Q")) {
		countOfSources++
		originalValues[j] = true
		} else {
		originalValues[j] = null
		}
	}
	generateCombinations(originalValues)
	} else {
	}
	count++
```

```
}
const betweenTheLines = () ⇒ {
	// Hier wird die Aktuelle Zeile durchgegangen, und geschaut mit der if welcher Block sich gerade an der Stelle befindet. Und im dem If wird dann die Logik angewendet die dieser Block genötigt
	for (let j = 0; j < width; j++) {

	if (currentLine[j] ≡≡ "W") {
		if (currentLight[j] == null && currentLight[j + 1] == null) {
			// Wenn der Lichtstand in der Zeile darüber nicht vorhanden ist, weil da kein Block war mach nichts
		} else if (currentLight[j] && currentLight[j + 1]) {
			// Wenn die Lichter davor an waren, mache sie jetzt aus
			currentLight[j] = false
			currentLight[j + 1] = false
		} else {
			// Ansonsten mach die Lichter an
			currentLight[j] = true
			currentLight[j + 1] = true
		}

		// Und weil der Blöcke immer als Doppelte vorkommen überspringe einfach den Nächsten Buchstaben, weil da der gleiche ist
		j++
	} else if (currentLine[j] ≡≡ "B") {
		// gleiche Schema wie bei WW
	} else if (currentLine[j] ≡≡ "r") {
		// gleiche Schema wie bei WW
	} else if (currentLine[j] ≡≡ "R") {
		// gleiche Schema wie bei WW
	} else if (currentLine[j] ≡≡ "X") {
		currentLight[j] = null
	}
	}
}
const lastLine = () ⇒ {
	// Wenn diese Funktion ausgeführt wird, schau sie nach den Positionen an l's in der Zeile und ist an einer Position ein L wird der Letzte Lichtstand genommen und ausgegeben
}
```

```
readStream.on('data', (chunk) ⇒ {
	// Hier wird die Datei gelesen
	// Variables //
	width = 0
	height = 0
	countOfSources = 0
	count = 0
	runTimeStart = new Date().toString()

	defaultLights = []
	currentLight = []
	finalResult = []

	// Split the chunk into lines
	const lines = chunk.split('\n'); // array of every single line

	// Grabbing the layout
	const layout = lines[0].split(" ");
	width = layout[0]
	height = layout[1]

	// Remove the first line with the width and height
	lines.shift();

	const main = () ⇒ {
	for (let i = 0; i < height; i++) {
		Wenn i ⩵ 0 ist führen wir die firstline() aus, und wenn i ⩵ height-1 ist,
führe lastline() aus, befindet sich das Programm bei keinem der Beiden, führt es immer
betweenTheLines() aus
	}
});
```