Dokumentation Aufgabe 4

Jonte Puschmann – Städt. Kopernikus Gymnasium Rheine

Zusammenfassung der Aufgabenstellung

Nora findet eine Kiste mit drei Arten von elektronischen Bausteinen: weiß, rot und blau. Jeder Baustein hat Lichtsensoren und LEDs.

- Weiße Bausteine: Haben zwei Sensoren. Wenn beide Sensoren bestrahlt werden, sind beide LEDs aus. Andernfalls leuchten beide LEDs.
- Rote Bausteine: Haben einen Sensor. Wenn dieser Sensor beleuchtet wird, sind beide LEDs aus. Andernfalls leuchten beide LEDs.
- Blaue Bausteine: Haben zwei Sensoren. Die obere LED leuchtet, wenn der obere Sensor beleuchtet wird, und die untere LED leuchtet, wenn der untere Sensor beleuchtet wird.

Die Aufgabe besteht darin, ein Programm zu schreiben, das die Beschreibung einer Konstruktion mit diesen Bausteinen einliest und anzeigt, wie die LEDs der am weitesten rechts liegenden Bausteine auf die Lichtquellen reagieren.

Die Daten von einer URL hohlen

Die Funktion `def get_url(url):` hohlt sich die Textdateien der BwInf-Website.

Cell Klasse

Diese Klasse repräsentiert eine Zelle in der Matrix und enthält Informationen über den Zelltyp und den Zustand des benachbarten Zustands.

Matrix Klasse

Diese Klasse beinhaltet die gesamte Schaltung. Sie enthält die Breite, Höhe und Reihen der Matrix. Außerdem gibt es die Funktionen `def connect(self):` , welche bestimt, welche Zellen miteinander verbunden sind und `def process(self, inputs: list[int]) -> list[int]:`, welche den Zustand jeder Zelle berechnet.

Die Textdatei für das Programm formatieren

Die Funktion `def parse_matrix(source):` konvertiert die Datei in eine Matrix. Diese beinhaltet eine width, eine height, eine output_count und eine Liste der jeweiligen Reihen der Matrix.

Jeden möglichen Start berechnen

Mit `possible_start = bin(j)[2:].zfill(starts)` in der for Schleife für alle Startpunkte wird jede Kombination für die Zustände der Lampen berechnet.

Ergebnis berechnen

Die Funktion `def process(self, inputs: list[int]) -> list[int]:` berechnet nun Reihe für Reihe die Zustände der Blöcke und beachtet dabei immer den Zustand des darüberliegenden Blockes.

Lösungen der Beispielaufgaben:

Datei 1:

Input: [0, 0] | Output: [1, 1]

Input: [0, 1] | Output: [1, 1]

Input: [1, 0] | Output: [1, 1]

Input: [1, 1] | Output: [0, 0]

Datei 2:

Input: [0, 0] | Output: [0, 1]

Input: [0, 1] | Output: [0, 1]

Input: [1, 0] | Output: [0, 1]

Input: [1, 1] | Output: [1, 0]

Datei 3:

Input: [0, 0, 0] | Output: [1, 0, 0, 1]

Input: [0, 0, 1] | Output: [1, 0, 0, 0]

Input: [0, 1, 0] | Output: [1, 0, 1, 1]

Input: [0, 1, 1] | Output: [1, 0, 1, 0]

Input: [1, 0, 0] | Output: [0, 1, 0, 1]

Input: [1, 0, 1] | Output: [0, 1, 0, 0]

Input: [1, 1, 0] | Output: [0, 1, 1, 1]

Input: [1, 1, 1] | Output: [0, 1, 1, 0]

Datei 4:

Input: [0, 0, 0, 0] | Output: [0, 0]

Input: [0, 0, 0, 1] | Output: [0, 0]

Input: [0, 0, 1, 0] | Output: [0, 1]

Input: [0, 0, 1, 1] | Output: [0, 0]

Input: [0, 1, 0, 0] | Output: [1, 0]

Input: [0, 1, 0, 1] | Output: [1, 0]

Input: [0, 1, 1, 0] | Output: [1, 1]

Input: [0, 1, 1, 1] | Output: [1, 0]

Input: [1, 0, 0, 0] | Output: [0, 0]

Input: [1, 0, 0, 1] | Output: [0, 0]

Input: [1, 0, 1, 0] | Output: [0, 1]

Input: [1, 0, 1, 1] | Output: [0, 0]

Input: [1, 1, 0, 0] | Output: [0, 0]

Input: [1, 1, 0, 1] | Output: [0, 0]

Input: [1, 1, 1, 0] | Output: [0, 1]

Input: [1, 1, 1, 1] | Output: [0, 0]

Datei 5:

Input: [0, 0, 0, 0, 0, 0] | Output: [0, 0, 0, 1, 0]

Input: [0, 0, 0, 0, 0, 1] | Output: [0, 0, 0, 1, 0]

Input: [0, 0, 0, 0, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 0, 0, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 0, 1, 0, 0] | Output: [0, 0, 1, 0, 0]

Input: [0, 0, 0, 1, 0, 1] | Output: [0, 0, 1, 0, 0]

Input: [0, 0, 0, 1, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 0, 1, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 1, 0, 0, 0] | Output: [0, 0, 0, 1, 0]

Input: [0, 0, 1, 0, 0, 1] | Output: [0, 0, 0, 1, 0]

Input: [0, 0, 1, 0, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 1, 0, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 1, 1, 0, 0] | Output: [0, 0, 1, 0, 0]

Input: [0, 0, 1, 1, 0, 1] | Output: [0, 0, 1, 0, 0]

Input: [0, 0, 1, 1, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 0, 1, 1, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 0, 0, 0, 0] | Output: [0, 0, 0, 1, 0]

Input: [0, 1, 0, 0, 0, 1] | Output: [0, 0, 0, 1, 0]

Input: [0, 1, 0, 0, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 0, 0, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 0, 1, 0, 0] | Output: [0, 0, 1, 0, 0]

Input: [0, 1, 0, 1, 0, 1] | Output: [0, 0, 1, 0, 0]

Input: [0, 1, 0, 1, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 0, 1, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 1, 0, 0, 0] | Output: [0, 0, 0, 1, 0]

Input: [0, 1, 1, 0, 0, 1] | Output: [0, 0, 0, 1, 0]

Input: [0, 1, 1, 0, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 1, 0, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 1, 1, 0, 0] | Output: [0, 0, 1, 0, 0]

Input: [0, 1, 1, 1, 0, 1] | Output: [0, 0, 1, 0, 0]

Input: [0, 1, 1, 1, 1, 0] | Output: [0, 0, 0, 1, 1]

Input: [0, 1, 1, 1, 1, 1] | Output: [0, 0, 0, 1, 1]

Input: [1, 0, 0, 0, 0, 0] | Output: [1, 0, 0, 1, 0]

Input: [1, 0, 0, 0, 0, 1] | Output: [1, 0, 0, 1, 0]

Input: [1, 0, 0, 0, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 0, 0, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 0, 1, 0, 0] | Output: [1, 0, 1, 0, 0]

Input: [1, 0, 0, 1, 0, 1] | Output: [1, 0, 1, 0, 0]

Input: [1, 0, 0, 1, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 0, 1, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 1, 0, 0, 0] | Output: [1, 0, 0, 1, 0]

Input: [1, 0, 1, 0, 0, 1] | Output: [1, 0, 0, 1, 0]

Input: [1, 0, 1, 0, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 1, 0, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 1, 1, 0, 0] | Output: [1, 0, 1, 0, 0]

Input: [1, 0, 1, 1, 0, 1] | Output: [1, 0, 1, 0, 0]

Input: [1, 0, 1, 1, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 0, 1, 1, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 0, 0, 0, 0] | Output: [1, 0, 0, 1, 0]

Input: [1, 1, 0, 0, 0, 1] | Output: [1, 0, 0, 1, 0]

Input: [1, 1, 0, 0, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 0, 0, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 0, 1, 0, 0] | Output: [1, 0, 1, 0, 0]

Input: [1, 1, 0, 1, 0, 1] | Output: [1, 0, 1, 0, 0]

Input: [1, 1, 0, 1, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 0, 1, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 1, 0, 0, 0] | Output: [1, 0, 0, 1, 0]

Input: [1, 1, 1, 0, 0, 1] | Output: [1, 0, 0, 1, 0]

Input: [1, 1, 1, 0, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 1, 0, 1, 1] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 1, 1, 0, 0] | Output: [1, 0, 1, 0, 0]

Input: [1, 1, 1, 1, 0, 1] | Output: [1, 0, 1, 0, 0]

Input: [1, 1, 1, 1, 1, 0] | Output: [1, 0, 0, 1, 1]

Input: [1, 1, 1, 1, 1, 1] | Output: [1, 0, 0, 1, 1]