Junioraufgabe 2: Kacheln

Team-ID: 00041

Team-Name: team@sebastianbrunnert

Bearbeiter/-innen dieser Aufgabe:
Sebastian Brunnert

5. September 2019

Inhaltsverzeichnis

Lösungsidee

Zum Lösen dieser Aufgabe, betrachte ich nicht eine Kachel als solches, sondern nur die einzelnen Nachbarfelder (die in einer anderen Kachel sind) des Feldes. Ein Beispiel dafür wäre das Feld, was in einer Kachel oben links ist. Untersucht wird dabei das Feld links und über diesem (beide müssen in einer anderen Kachel sein). Sollten diese beiden Felder nicht den selben Wert haben, so ist die Karte nicht lösbar. Haben allerdings beide Felder den selben Wert, so ist dieser auch der Wert, der in das noch nicht gelöste Feld eingesetzt werden. Diese Suche nach dem einzusetzenden Wert, wird für jedes Feld getätigt. Es muss auch berücksichtigt werden, ob ein Feld überhaupt ein direktes Nachbarfeld in einer anderen Kachel hat, und ob diese überhaupt bereits einen Wert haben.

Umsetzung

Die Umsetzung des Programms geschieht in der Programmiersprache Python. Benötigt wird die Programmbiblithek sys. Diese ist dafür zuständig, das Programm stoppen zu können und für die Übergabe von Kommandozeilenargumenten.

Zu Start des Programms wird die angegebene Karte (Textdatei) geladen. Die Reihenfolge der Einsen und Nullen und auch von minus Eins, welches in meinem Speicherschema „noch nicht definiert“ - also ein noch leeres Feld - darstellt, werden in einem Array gespeichert. Für jede Reihe, gibt es also einen Array. Diese werden alle in dem Array „lines“ gespeichert. Nun wird jede Reihe und jedes Element darin in einer Schleife durchgegangen, um die fehlenden Felder zu ermitteln. Bei einem Durchgang wird ebenfalls der Index der Reihe und der Index des Feldes in der Zeile zwischengespeichert. Wenn ein Feld also nun den Wert minus Eins hat, wird erst die Position des Feldes in der simulierten Kachel gesucht, damit im späteren Verlauf der Suche auch mit Bezug auf Nachbarfelder gesucht werden kann. Ist der Index der Reihe gerade, lässt sich sagen, dass das Feld relativ gesehen in einer Kachel „oben“ ist. Ist der Index des Feldes gerade, lässt sich auch sagen, dass das Feld relativ gesehen in einer Kachel „links“ ist.

Line-Index gerade & Feld-Index gerade	Feld ist „oben links“

Line-Index gerade & Feld-Index ungerade	Feld ist „oben rechts“
Line-Index ungerade & Feld-Index gerade	Feld ist „unten links“
Line-Index ungerade & Feld-Index ungerade	Feld ist „unten rechts“

Nun müssen die beiden benachbarten Felder, die (wie in Lösungsidee beschrieben) nicht in der selben simulierten Kachel sein können, gefunden werden. Natürlich wird im Algorithmus auch beachtet, ob überhaupt benachbarte Felder vorliegen, und ob diese überhaupt schon einen Wert haben.

Feld ist „oben links“	Mögliches benachbartes Feld 1: Zeile $n_{LineIndex}$-1 und Spalte $n_{FeldIndex}$ Mögliches benachbartes Feld 2: Zeile $n_{LineIndex}$ und Spalte $n_{FeldIndex}$-1
Feld ist „oben rechts“	Mögliches benachbartes Feld 1: Zeile $n_{LineIndex}$-1 und Spalte $n_{FeldIndex}$ Mögliches benachbartes Feld 2: Zeile $n_{LineIndex}$ und Spalte $n_{FeldIndex}$+1
Feld ist „unten links“	Mögliches benachbartes Feld 1: Zeile $n_{LineIndex}$+1 und Spalte $n_{FeldIndex}$ Mögliches benachbartes Feld 2: Zeile $n_{LineIndex}$ und Spalte $n_{FeldIndex}$-1
Feld ist „oben rechts“	Mögliches benachbartes Feld 1: Zeile $n_{LineIndex}$+1 und Spalte $n_{FeldIndex}$ Mögliches benachbartes Feld 2: Zeile $n_{LineIndex}$ und Spalte $n_{FeldIndex}$+1

Sind zwei benachbarte Felder (außerhalb der simulierten Kachel) gefunden, so wird geprüft, ob deren Werte übereinstimmen. Ist dies der Fall wird dieser Wert an die aktuelle Stelle des Kartenschemas gesetzt. Ist dies nicht der Fall, so ist die Karte nicht lösbar und das Programm wird gestoppt. Ist nur ein benachbartes Feld (außerhalb der simulierte Kachel) gefunden, so wird dieser an die aktuelle Stelle des Kartenschemas gesetzt. Ist keines gefunden, ist der Wert irrelevant, weil sich das Feld am äußersten Rand der Karte befindet und keine benachbarten Felder außerhalb der simulierten Kachel hat. Die aktuelle Stelle des Kartenschemas wird dann einfach zu Eins.

Schlussendlich wird das Kartenschema ausgesendet. (Array in Array)

Beispiele

Wir rufen das Programm nun auf und geben die Werte auf der BWINF-Webseite an:

1	$ python3 kacheln.py map_spacy.txt Ausgabe: [[1, 0, 0, 1], [0, 1, 1, 0], [0, 1, 1, 0], [1, 1, 1]]
2	$ python3 kacheln.py map1_spacy.txt Ausgabe: [[1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1], [0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1], [0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1], [1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1], [1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1], [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, -1], [0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, -1], [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0], [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0], [1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1], [1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1], [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]]
3	$ python3 kacheln.py map2_spacy.txt Ausgabe: [[1, 1, 1, 1, 1, 1], [1, 0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 1], [1, 1, 1, 0, 0, 1], [1, 1, 1, 0, 0, 1], [1, 0, 0, 0, 0, 1], [1, 1, 1, 1, 1, 1]]
4	$ python3 kacheln.py map3_spacy.txt Ausgabe:

	[[1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1], [1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0], [1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0], [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1], [1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1], [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1]]
5	$ python3 kacheln.py map4_spacy.txt Ausgabe: Diese Karte ist nicht lösbar.

Quellcode

Der vollständige Quellcode, ist in der Datei kacheln.py zu finden.

```
# Deklariere Schema der Karte (siehe Dokumentation)
lines = []

# Gehe jede Zeile der Textdatei durch
for line in file.readlines():
    # Überspring erst zwei Zeilen
    if(len(line) > 3):
        # Lösche Leerzeichen
        line = line.replace(" ", "")
        # Liste Ziffern auf
        fields = list(line)
        # Lösche \n an Zeilenende
        if(fields[-1] != "1" or fields[-1] != "0"):
            fields = fields[:-1]
        # Strings zu Zahlen
        for n, i in enumerate(fields):
            if i == "1":
                fields[n] = 1
            elif i == "0":
                fields[n] = 0
            else:
                fields[n] = -1
        # Diese Zeile zu Resultat hinzufügen
        lines.append(fields)

# Gehe alle Felder durch (sichere auch Index der Zeile (y) und des Feldes (x))
for line_count in range(0,len(lines)):
    line = lines[line_count]
    for column_count in range(0,len(line)):
```

```
    column = line[column_count]


    # Feld ist nicht vorhanden
    if(column == -1):


        possible_solution = 1


        if(line_count % 2 == 0):
            if(column_count  % 2 == 0):
                # Feld ist oben links in Kachel
                if(line_count-1 < 0 and column_count-1 < 0):
                    # Feld ist auf gesamten Feld oben links, aufgrunddessen ist es irrelevant welche Zahl zugewiesen
wird
                    possible_solution = 1
                elif(column_count-1 < 0):
                    # Feld hat auf x-Achse keinen linken Nachbarn, deswegen wird Wert des Feldes über diesem
übernommen
                    possible_solution = lines[line_count-1][column_count]
                elif(line_count-1 < 0):
                    # Feld hat auf y-Achse keinen oberen Nachbarn, deswegen wir Wert des Feldes links neben
diesen übernommen
                    possible_solution = line[column_count-1]
                else:
                    # Feld liegt nicht am Rand, deshalb muss Feld über und Feld links neben diesem, den selben Wert
haben. Ist dies nicht
                    # der Fall, so ist diese Karte nicht lösbar
                    if(lines[line_count-1][column_count] == -1 and line[column_count-1] == -1):
                        # Beide benachbarten Felder sind noch nicht definiert
                        possible_solution = 1
                    elif(lines[line_count-1][column_count] == -1):
                        # Einer der beiden Werte ist noch nicht definiert
                        possible_solution = line[column_count-1]
                    elif(line[column_count-1] == -1):
                        possible_solution = lines[line_count-1][column_count]
                    elif(lines[line_count-1][column_count] == line[column_count-1]):
```

```
                possible_solution = line[column_count-1]
            else:
                print("Diese Karte ist nicht lösbar")
                sys.exit(0)
        else:
            # Feld ist oben rechts in Kachel
            if(line_count-1 < 0 and column_count+1 == len(line)):
                # Feld ist auf dem gesammten Feld oben rehts, aufgrunddessen ist es irrelevant welche Zahl zugewiesen wird
                possible_solution = 1
            elif(column_count+1 == len(line)):
                # Feld hat auf x-Achse keinen rechten Nachbarn, deswegen wird Wert des Feldes über diesem übernommen
                possible_solution = lines[line_count-1][column_count]
            elif(line_count-1 < 0):
                # Feld hat auf y-Achse keinen oberen Nachbarn, deswegen wir Wert des Feldes links neben diesen übernommen
                possible_solution = line[column_count-1]
            else:
                # Feld liegt nicht am Rand, deshalb muss Feld über und Feld rechts neben diesen, den selben Wert haben. Ist dies nicht
                # der Fall, so ist Karte nicht lösbar
                if(lines[line_count-1][column_count] == -1 and line[column_count+1] == -1):
                    # Beide benachbarten Felder sind noch nicht definiert
                    possible_solution = 1
                elif(lines[line_count-1][column_count] == -1):
                    # Einer der beiden Werte ist noch nicht definiert
                    possible_solution = line[column_count+1]
                elif(line[column_count+1] == -1):
                    # Einer der beiden Werte ist noch nicht definiert
                    possible_solution = lines[line_count-1][column_count]
                elif(lines[line_count-1][column_count] == line[column_count+1]):
                    possible_solution = line[column_count+1]
                else:
                    print("Diese Karte ist nicht lösbar")
```

```
                sys.exit(0)


        else:
            if(column_count  % 2 == 0):
                # Feld ist unten links in Kachel
                if(line_count+1 == len(lines) and column_count-1 < 0):
                    # Feld ist auf gesamten Feld unten links, aufgrunddessen ist es irrelevant welche Zahl zugewiesen
wird
                    possible_solution = 1
                elif(column_count-1 < 0):
                    # Feld hat auf x-Achse keinen linken Nachbarn, deswegen wird Wert des Feldes über diesem
übernommen
                    possible_solution = lines[line_count+1][column_count]
                elif(line_count+1 == len(lines)):
                    # Feld hat auf y-Achse keinen unteren Nachbarn, deswegen wir Wert des Feldes links neben
diesen übernommen
                    possible_solution = line[column_count-1]
                else:
                    # Feld liegt nicht am Rand, deshalb muss Feld unter und Feld links neben diesem, den selben
Wert haben. Ist dies nicht
                    # der Fall, so ist diese Karte nicht lösbar
                    if(lines[line_count+1][column_count] == -1 and line[column_count-1] == -1):
                        # Beide benachbarten Felder sind noch nicht definiert
                        possible_solution = 1
                    elif(lines[line_count+1][column_count] == -1):
                        # Einer der beiden Werte ist noch nicht definiert
                        possible_solution = line[column_count-1]
                    elif(line[column_count-1] == -1):
                        # Einer der beiden Werte ist noch nicht definiert
                        possible_solution = lines[line_count+1][column_count]
                    elif(lines[line_count+1][column_count] == line[column_count-1]):
                        possible_solution = line[column_count-1]
                    else:
                        print("Diese Karte ist nicht lösbar")
                        sys.exit(0)
```

```
        else:
            # Feld ist oben rechts in Kachel
            if(line_count+1 == len(lines) < 0 and column_count+1 == len(line)):
                # Feld ist auf dem gesammten Feld unten rehts, aufgrunddessen ist es irrelevant welche Zahl zugewiesen wird
                possible_solution = 1
            elif(column_count+1 == len(line)):
                # Feld hat auf x-Achse keinen rechten Nachbarn, deswegen wird Wert des Feldes über diesem übernommen
                possible_solution = lines[line_count+1][column_count]
            elif(line_count+1 == len(lines)):
                # Feld hat auf y-Achse keinen unteren Nachbarn, deswegen wir Wert des Feldes links neben diesen übernommen
                possible_solution = line[column_count-1]
            else:
                # Feld liegt nicht am Rand, deshalb muss Feld unter und Feld rechts neben diesen, den selben Wert haben. Ist dies nicht
                # der Fall, so ist Karte nicht lösbar
                if(lines[line_count+1][column_count] == -1 and line[column_count+1] == -1):
                    # Beide benachbarten Felder sind noch nicht definiert
                    possible_solution = 1
                elif(lines[line_count+1][column_count] == -1):
                    # Einer der beiden Werte ist noch nicht definiert
                    possible_solution = line[column_count+1]
                elif(line[column_count+1] == -1):
                    # Einer der beiden Werte ist noch nicht definiert
                    possible_solution = lines[line_count+1][column_count]
                elif(lines[line_count+1][column_count] == line[column_count+1]):
                    # Einer der beiden Werte ist noch nicht definiert
                    possible_solution = line[column_count+1]
                else:
                    print("Diese Karte ist nicht lösbar")
                    sys.exit(0)
```

lines[line_count][column_count] = possible_solution