

Lösungsidee

Das Programm soll für beliebige $n \geq 4$ verschiedene lösbare Arukone mit Gittergröße $n \times n$ erstellt. Da n nicht nach oben begrenzt ist, empfiehlt es sich, für kleine n Lösungen zu finden, die man dann für größere n durch geregeltes „Anbauen“ von weiteren Zeilen und Spalten vergrößern kann. Zu einfache Lösungen, wie zum Beispiel

1 0 0 0 1	Lösung: 1-----1
2 0 0 0 2	2-----2
3 0 0 0 3	3-----3
4 0 0 0 4	4-----4
5 0 0 0 5	5-----5

hier für $n=5$, die sich auch sehr leicht auf größere n übertragen ließen, werden aber vom Lösungsprogramm auf der BWINF-Seite gelöst. Daher müssen etwas schwerere Grundrätsel her, die vom Lösungsprogramm nicht gelöst werden. Nach etwas ausprobieren haben wir uns für folgende Schemata entschieden:

n	n
n-2	n
1 2 1 0 3 4 ... n-2	4 0 0 0 5 6 7 8 ... n
0 0 0 0 0 0 ... 0	3 0 0 0 0 0 0 0 ... 0
0 0 2 0 0 0 ... 0	0 1 0 0 0 0 0 0 ... 0
3 0 0 0 0 0 ... 0	0 2 0 0 0 0 0 0 ... 0
0 0 0 0 0 0 ... 0	...
4 0 0 0 0 0 ... 0	0 0 0 0 0 6 0 8 ... 0
...	0 0 3 4 5 7 0 1 ... 0
n-2 0 0 0 0 ... 0	0 0 0 0 0 0 2 0 ... 0
	0 0 0 0 0 0 0 0 ... n

Das erste (linke) Schema startet bei $n=5$ und hat immer $n-2$ Zahlenpaare. Das 5 mal 5 Feld ist vorgegeben, für alle größeren n werden rekursiv immer links unten und rechts oben die nächste Zahl eingefügt und der Rest der neuen Zeile und Spalte mit Nullen belegt, sodass das neue Zahlenpaar in der Lösung einfach am rechten/unteren Rand entlang verbunden wird und der Rest wie vorher geht.

Das zweite (rechte) Schema ist etwas komplizierter und startet bei $n=8$ (für $n < 8$ mussten wir uns deshalb ein paar extra-Arukone ohne Schema überlegen, dazu in der Umsetzung mehr) und hat immer n Zahlenpaare. Für $n > 8$ wird dann rekursiv immer in der Mitte eine Zeile mit nur Nullen eingefügt und rechts eine Spalte in der oben und unten das neue Zahlenpaar steht. Das 8 mal 8 Grundmuster bleibt dabei gleich und ist so konstruiert, dass durch die Mitte genau eine Linie pro Übergang verläuft, sodass die neue Zeile bei der Lösung nicht stört und die Linien alle einfach um eins länger werden.

Umsetzung

Die Lösungsidee wird in Python umgesetzt. Am Anfang wird die Variable n vom Benutzer abgefragt, die die Gittergröße der Rätsel sein soll.

Die erste Methode `erstes_arukon_printen(n)` erstellt ein Rätsel mit der gegebenen Gittergröße nach dem ersten Schema und gibt dieses aus. Dabei werden immer mithilfe von for-Schleifen immer an die ersten fünf Zeilen weitere Zeichen angefügt und unter der fünften Zeile neue Zeilen der Länge n eingefügt, sodass das Gitter am Ende die Größe $n \times n$ hat. Das Arukon wird in der Methode direkt ausgegeben

Die zweite Methode `zweites_arukon_printen(n)` erstellt ein weiteres Rätsel mit Gittergröße $n \times n$ und gibt es direkt aus nach dem zweiten Schema. Dabei wird zwischen $n \geq 8$ und $5 \leq n < 8$ unterschieden. Für $n \geq 8$ wird das Schema mit durch for-Schleifen eingefügten Zeilen in der Mitte und Spalten am rechten Rand umgesetzt. Für $n = 5, 6$ oder 7 haben wir uns spezielle Arukons ausgedacht, die das Lösungsprogramm nicht lösen kann und einfach nur die ausgegeben lassen.

Im Hauptcode werden dann bei $n > 4$ einfach nur die beiden Methoden aufgerufen. Für $n = 4$ sind beide Schemata zu groß und es werden einfach zwei speziell dafür ausgedachte Arukone ausgegeben, die das Lösungsprogramm nicht lösen kann

Beispiele

Für $n = 4$ wird der Sonderfall ausgegeben:

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)4
Das ist das erste Arukon:
```

```
4
3
2 0 1 3
1 0 0 0
0 2 0 0
0 0 0 3
```

```
Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:
```

```
4
3
0 0 0 1
1 2 0 0
0 0 3 0
3 2 0 0
```

```
>>>
```

mit folgenden Lösungen:

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
```

```
Was soll n sein? (n≥4)4
```

```
Das ist das erste Arukon:
```

```
4
3
2 0 1 3
1 0 0 0
0 2 0 0
0 0 0 3
```

```
Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:
```

```
4
3
0 0 0 1
1 2 0 0
0 0 2 0
3 2 0 0
>>>
```

Für $n = 5$ ist oben das erste Grundschema und unten ein speziell ausgedachtes:

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
```

```
Was soll n sein? (n≥4)5
```

```
Das ist das erste Arukon:
```

```
5
3
1 2 1 0 3
0 0 0 0 0
0 0 2 0 0
3 0 0 0 0
0 0 0 0 0
```

```
Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:
```

```
5
5
0 0 0 0 0
0 1 2 3 0
0 0 0 0 0
0 3 1 2 0
0 0 0 0 0
>>>
```

mit den Lösungen

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)5
Das ist das erste Arukon:
```

```
5
3
1 2 1 0 3
0 0 0 0 0
0 0 2 0 0
3 0 0 0 0
0 0 0 0 0
```

Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
5
5
0 0 0 0 0
0 1 2 3 0
0 0 0 0 0
0 3 1 2 0
0 0 0 0 0
```

>>>

Bei $n = 6$ wird oben einfach das einmal erweiterte Schema und unten ein Sonderfall ausgegeben (und danach haben wir wieder die Lösungen angegeben)

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)6
Das ist das erste Arukon:
```

```
6
4
1 2 1 0 3 4
0 0 0 0 0 0
0 0 2 0 0 0
3 0 0 0 0 0
0 0 0 0 0 0
4 0 0 0 0 0
```

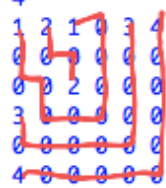
Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
6
6
1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 3 0 0
0 0 4 4 0 0
2 0 0 3 2 0
1 0 0 0 0 0
```

>>>

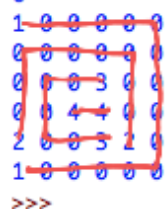
```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)6
Das ist das erste Arukon:
```

```
6
4
1 2 1 0 3 4
0 0 0 0 0 0
0 0 2 0 0 0
3 0 0 0 0 0
0 0 0 0 0 0
4 0 0 0 0 0
```



Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
6
6
1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 3 0 0
0 0 4 0 0 0
2 0 0 3 1 0
1 0 0 0 0 0
>>>
```



$n = 7$ ist das letzte Mal, dass unten ein speziell konstruierter Sonderfall ausgegeben wird

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)7
Das ist das erste Arukon:
```

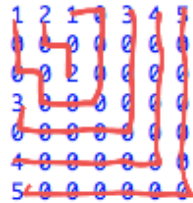
```
7
5
1 2 1 0 3 4 5
0 0 0 0 0 0 0
0 0 2 0 0 0 0
3 0 0 0 0 0 0
0 0 0 0 0 0 0
4 0 0 0 0 0 0
5 0 0 0 0 0 0
```

Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
7
7
1 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 3 0 0
0 0 4 0 0 0 0
0 0 4 0 0 0 0
2 0 0 0 3 2 0
1 0 0 0 0 0 0
>>>
```

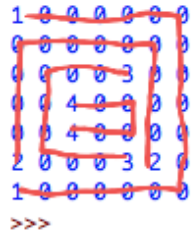
```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)7
Das ist das erste Arukon:
```

```
7
5
1 2 1 3 4 5
0 0 0 0 0 0
0 0 2 0 0 0
3 0 0 0 0 0
0 0 0 0 0 0
4 0 0 0 0 0
5 0 0 0 0 0
```



Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
7
7
1 0 0 0 0 0
0 0 0 0 0 0
0 0 0 3 0 0
0 0 4 0 0 0
0 0 4 0 0 0
2 0 0 3 2 0
1 0 0 0 0 0
>>>
```



Bei $n = 8$ wird dann zum ersten Mal unten das Grundschema ausgegeben und oben weiterhin das gleiche für 8 Zeilen und Spalten erweitert:

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
Was soll n sein? (n≥4)8
Das ist das erste Arukon:
```

```
8
6
1 2 1 0 3 4 5 6
0 0 0 0 0 0 0 0
0 0 2 0 0 0 0 0
3 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0
```

Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

```
8
8
4 0 0 0 5 6 7 8
3 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 2 0 0 0 0 0 0
0 0 0 0 0 6 0 8
0 0 3 4 5 7 0 1
0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0
>>>
```

```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
```

```
Was soll n sein? (n≥4)8
```

```
Das ist das erste Arukon:
```

```
8
6
1 2 1 0 3 4 5 6
0 0 0 0 0 0 0 0
0 0 2 0 0 0 0 0
3 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0
```

```
Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:
```

```
8
8
4 0 0 0 5 6 7 8
3 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 2 0 0 0 0 0 0
0 0 0 0 6 0 0 0
0 0 3 4 5 7 0 1
0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0
```

```
>>>
```

Bei allen größeren n werden bei beiden Rätseln einfach nur noch neue Zeilen und Spalten eingefügt, als Beispiel hier noch n = 13:

===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====

Was soll n sein? (n≥4)13

Das ist das erste Arukon:

13

11

```
1 2 1 0 3 4 5 6 7 8 9 10 11
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 2 0 0 0 0 0 0 0 0 0 0
3 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 0 0 0 0
5 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 0 0 0 0 0
8 0 0 0 0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0 0 0
11 0 0 0 0 0 0 0 0 0 0 0 0
```

Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

13

13

```
4 0 0 0 5 6 7 8 9 10 11 12 13
3 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 2 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 6 0 8 0 0 0 0 0
0 0 3 4 5 7 0 1 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 0 0 9 10 11 12 13
```

>>>


```
===== RESTART: /Users/anton/Desktop/231014BWINF_aufgabe_1.py =====
```

Was soll n sein? ($n \geq 4$)¹³

Das ist das erste Arukon:

13

11

A 12x12 grid of blue numbers 0-9. A red line traces a path through the grid, starting at the top-left (0,0) and ending at the top-right (0,11). The path is a single continuous line that visits every cell in the grid exactly once, forming a Hamiltonian path. The path starts at (0,0), moves right to (0,11), then down to (11,11), then left to (11,0), then up to (0,0).

Hier ein weiteres Arukon-Rätsel mit der gleichen Gittergröße:

13

13

A 15x15 grid of numbers from 0 to 13. Red lines trace a path through the grid, starting at (0,0) and ending at (13,13). The path is composed of several connected segments, some of which are highlighted in red.

➤➤➤