AUFGABE 1: ARUKONE

Team-ID: 00250

Team: pip install BWINF_42

Lukas Marius Hoeschen

19. November 2023

INHALTSVERZEICHNIS

1. LÖSUNGSIDEE

In der Aufgabe sollen Rätsel, sogenannte Arukone, erstellt werden. Hierfür werden in einem *n*n* großem Feld *k* Zahlenpaare eingefügt, die über Wege der Länge *L* verbunden sind.

Um dies zu programmieren, wird für alle k Zahlenpaare ein zufälliges freies Feld gesucht. Dann wird von diesem Anfangsfeld aus ein Weg der Länge *L* zufällig nach oben, unten, rechts oder links gesucht. Hierfür wird überprüft, ob das jeweilige Feld noch frei ist. Jeder Wegabschnitt wird zudem als bereits „benutzt“ markiert, sodass kein anderer Weg oder Start-/Endpunkt hier verlaufen kann. Wenn der Weg fertig ist, wird der letzte Wegabschnitt als Endpunkt markiert.

Falls ein Weg in eine Sackgasse führt, wird eine neuer zufälliger Startpunkt, Länge und Weg gesucht. Sobald für alle *k* Zahlenpaare ein Weg gefunden wurde, werden alle Wegmarkierungen mit 0 ersetzt, sodass nur noch die Zahlenpaare übrig sind.

2. Implementierung

Das Programm wurde in Python implementiert. Das Rätsel wird als zweidimensionales Array repräsentiert. Anfangs ist jeder Wert 0, um zu symbolisieren, dass das Feld noch frei ist.

```
def create(n, k):
    a = make(n)
    for i in range(1, k+1):
        if not addNum(a, i):
            failCounter += 1
            return create(n, k)
```

Hier wird versucht, für alle *k* Zahlenpaare einen Weg zu erstellen. Falls dies nicht funktioniert, wird ein komplett neues Rätsel erstellt.

```
def addNum(x, i):
    for t in range(n*n):
        a = random.randint(0, n - 1)
        b = random.randint(0, n - 1)
        if x[a][b] == 0:
            x[a][b] = i
        else:
            continue
```

Hier wird für ein Zahlenpaar *i* ein Weg gesucht. *a* und *b* beschreiben das Anfangsfeld.

```
        d = random.randint(0, 3)
        while counter < random.randint(2, n*2) and countFails < 10:
            if random.randint(0,3) == 0:
                d = random.randint(0, 3)
```

Die maximale Länge L eines Weges ist hier mit random.randint(2, n/2) festgelegt. d beschreibt die nächste Richtung, in die der Weg weiterführen soll. 0 steht für oben, 1 steht für links, usw. Damit die Richtung nicht allzu häufig gewechselt wird und der Weg im Kreis verläuft, wir der Weg nur ca. jedes dritte Mal gewechselt, oder wenn die aktuelle Richtung in einen anderen Weg verläuft.

```
            try:
                if d == 0: # up
                    if x[a -1][b] == 0 and a != 0:
                        x[a - 1][b] = -i
                        counter += 1
                        a -= 1
                        countFails = 0
                    else:
                        countFails += 1
                        d = random.randint(0, 3) # next direction
```

Hier wird überprüft, ob zum Beispiel das Feld über dem aktuellen Feld noch frei ist.

```
        if countFails > 0: # Sackgasse
            # remove all i
            sackGassenCounter += 1
            for j in range(n):
                for k in range(n):
```

```
                    if x[j][k] == i or x[j][k] == -i:
                        x[j][k] = 0
            continue
```

Falls eine Sackgasse erkannt wurde (da nach Zehn zufälligen Richtungswechseln kein freies Feld gefunden wurde), wird der aktuelle Weg gelöscht und ein neuer gesucht.

```
        try:
            if x[a+1][b] == i:
                x[a+1][b] = 0
                x[a][b] = 0
                continue
        except IndexError:
            pass
```

Zudem wird überprüft, ob das Start- und End-Feld direkt nebeneinander liegen.

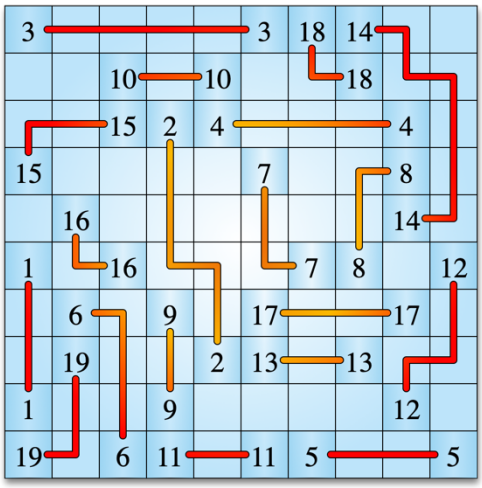
```
    for i in range(n):
        for j in range(n):
            if a[i][j]< 0:
                a[i][j] = 0
    return a
```

Wenn alle Wege gefunden wurden, werden alle Wege mit 0 ersetzt, sodass nur noch die Zahlenpaare übrig sind. Diese werden dann ausgegeben.

3. Ergebnisse

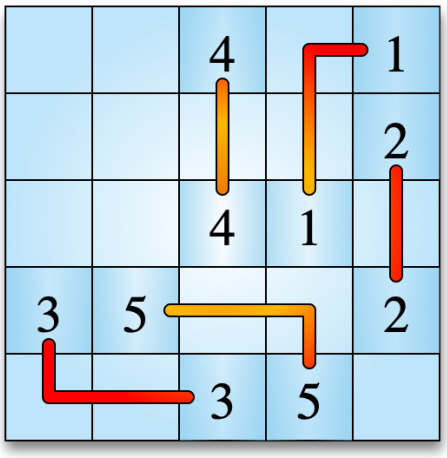
3.1 Beispiel 1

n = 10
k = 19



3.2 Beispiel 2

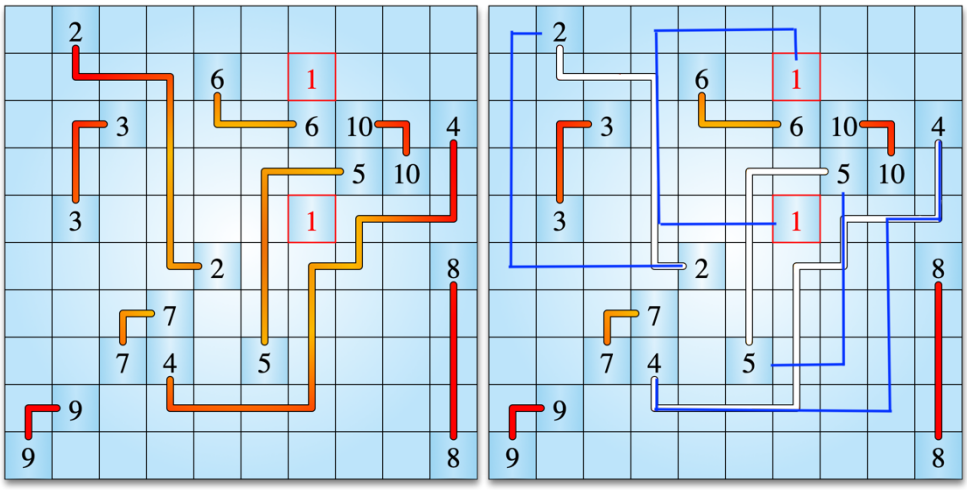
n = 5
k = 5



3.3 Beispiel 3 ohne „Online-Lösung“

n = 10
k = 10

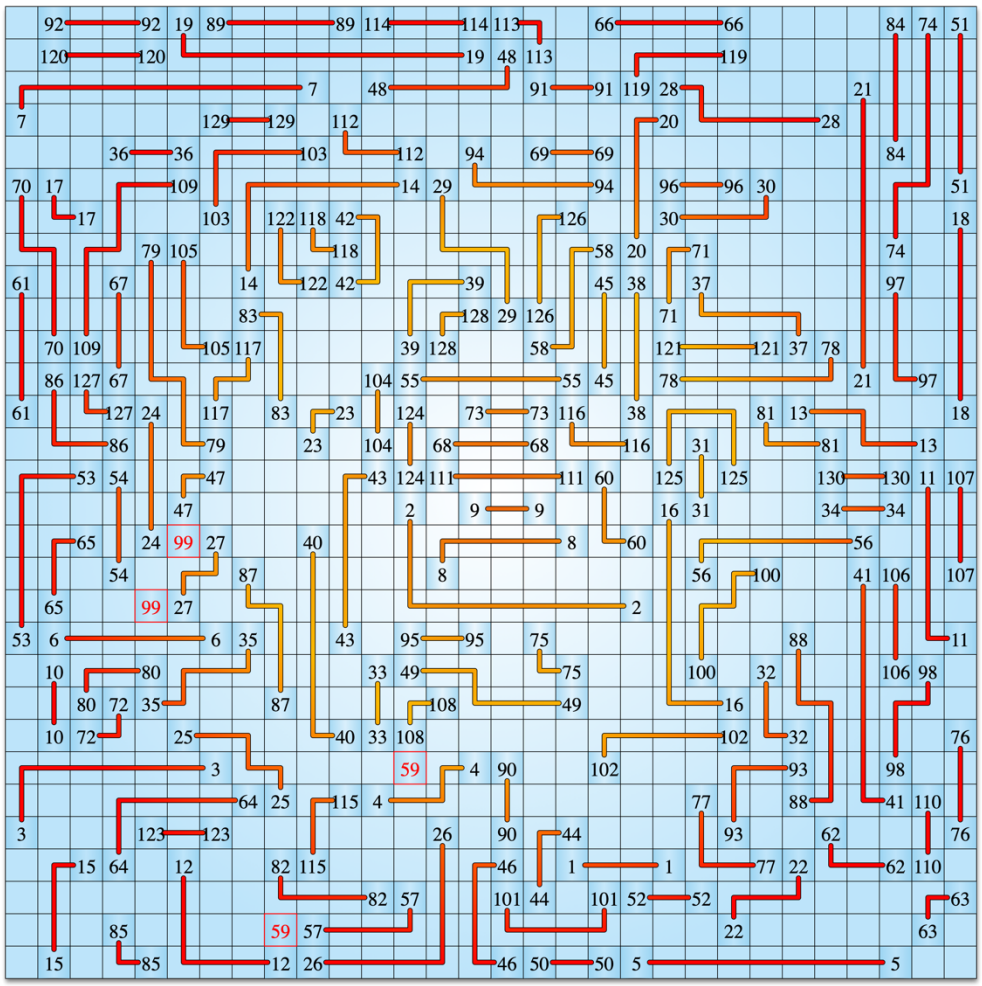
Bei diesem Rätsel wurde keine Lösung gefunden. Jedoch ist es dennoch möglich, dieses Rätsel zu lösen (blau):

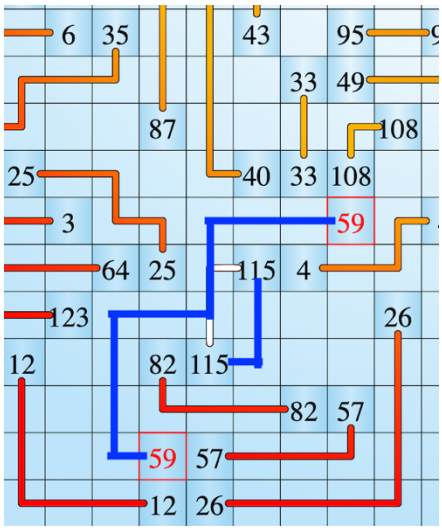


3.4 Beispiel 4 ohne „Online-Lösung“

n = 30
k = 130

Bei diesem Rätsel wurde keine Lösung gefunden. Jedoch ist es dennoch möglich, dieses Rätsel zu lösen (blau):





4. Laufzeit

Die Laufzeit beträgt: $O(k)$, da die Problemgröße hauptsächlich von der Anzahl an Zahlenpaaren abhängt.

5. QUELLCODE

```
def addNum(x, i):
    global sackGassenCounter
    for t in range(n*n):
        oldX = x
        a = random.randint(0, n - 1)
        b = random.randint(0, n - 1)
        if x[a][b] == 0:
            x[a][b] = i
        else:
            continue

        # search way
        counter = 0
        countFails = 0
        d = random.randint(0, 3) # next direction
        while counter < random.randint(2, int(n*2)) and countFails < 10:
            if random.randint(0,3) == 0:
                d = random.randint(0, 3) # next direction
            try:
                if d == 0: # up
                    if x[a -1][b] == 0 and a != 0:
                        x[a - 1][b] = -i
                        counter += 1
                        a -= 1
                        countFails = 0
                    else:
                        countFails += 1
                        d = random.randint(0, 3) # next direction
                if d == 1: # right
                    if x[a][b + 1] == 0:
                        x[a][b + 1] = -i
                        counter += 1
                        b += 1
                        countFails = 0
                    else:
                        countFails += 1
                        d = random.randint(0, 3) # next direction
                if d == 2: # down
                    if x[a + 1][b] == 0:
                        x[a + 1][b] = -i
                        counter += 1
                        a += 1
                        countFails = 0
                    else:
                        countFails += 1
                        d = random.randint(0, 3) # next direction
```

```
            if d == 3: # left
                if x[a][b - 1] == 0 and b != 0:
                    x[a][b - 1] = -i
                    counter += 1
                    b -= 1
                    countFails = 0
                else:
                    countFails += 1
                    d = random.randint(0, 3) # next direction
        except:
            countFails += 1

    if countFails > 0: # Sackgasse
        # remove all i
        sackGassenCounter += 1
        for j in range(n):
            for k in range(n):
                if x[j][k] == i or x[j][k] == -i:
                    x[j][k] = 0
        continue



    try:
        if x[a+1][b] == i:
            x[a+1][b] = 0
            x[a][b] = 0
            continue
    except IndexError:
        pass
    try:
        if x[a-1][b] == i:
            x[a-1][b] = 0
            x[a][b] = 0
            continue
    except IndexError:
        pass
    try:
        if x[a][b+1] == i:
            x[a][b+1] = 0
            x[a][b] = 0
            continue
    except IndexError:
        pass
    try:
        if x[a][b-1] == i:
            x[a][b-1] = 0
            x[a][b] = 0
            continue
```

```
        except IndexError:
            pass
        x[a][b] = i
        return True
    return False

def make(n):
    a = []
    for i in range(n):
        b = []
        for j in range(n):
            b.append(0)
        a.append(b)
    return a


def create(n, k):
    global failCounter
    a = make(n)
    for i in range(1, k+1):
        if not addNum(a, i):
            failCounter += 1
            return create(n, k)

    for i in range(n):
        for j in range(n):
            if a[i][j]< 0:
                a[i][j] = 0
    return a


n = int(input("Geben Sie eine Zahl ein: "))
k = random.randint(int(n/2)+1, int(2*n))

a = create(n, k)
```