# Aufgabe 5: Hüpfburg

Team-ID: 01103

Team: Skadi Baumgarte

Bearbeiter/-innen dieser Aufgabe:
Skadi Baumgarte

17. November 2022

## Inhaltsverzeichnis

Lösungsidee....1
Umsetzung....2
  controll....2
  main....2
Beispiele....2
  huepfburg0.txt....2
  huepfburg1.txt....2
  huepfburg2.txt....2
  huepfburg3.txt....3
  huepfburg4.txt....3
Quellcode....3

# Lösungsidee

Um diese Aufgabe zu lösen, muss man jeglichen Sprung beider Personen simulieren und herausfinden, ob sie überhaupt jemals auf dem gleichen Punkt ankommen. Für den zweiten Punkt gibt es eine relativ einfache Lösung:sie müssen sich spätestens nach $(n/2)*n+n$ Sprüngen treffen. Wenn man davon aus geht, dass alle Felder (n) in einem Kreis angeordnet sind, wodurch es definitiv nicht funktioniert, und dann eine Abkürzung einfügt, dauert es die Hälfte des Kreises (da dies der größte mögliche Abstand (also n/2) beider Spieler ist) mal den Kreis (da er jede Runde genau ein Feld aufholt (also mal n)) plus noch maximal n Sprünge bis man zum ersten mal zu der Abkürzung kommt (also +n). Dies ist der längste Weg, da jede weitere Verbindung nur noch mehr Abkürzungen ergeben können und das Löschen einer Verbindung es wieder unmöglich macht. Daraus kann man also schließen, dass man eine Simulation, die mir jede Möglichkeit von jedem Feld gibt, genauso lange laufen lassen müsste.

Außerdem ist es unwichtig, wie ein Spieler auf ein Feld kommt. Daher kann man in jeder Runde alle Wege (bis auf einen) löschen, die auf ein gleiches Feld führen. Dadurch wird die Simulation

nicht exponentiell, sondern man hat zu jedem Zeitpunkt maximal n Wege, die man vergleichen muss.

# Umsetzung

## controll

Die controll Methode überprüft, ob in zwei Listen (Spieler 1 und 2), in denen wieder Listen (die Wege der Spieler) sind, für jede dieser inneren Listen mit jeder inneren Liste der zweiten Liste der letzte Eintrag gleich ist, wie folgt im Beispiel zu sehen:

[[1,3,5],[1,4,6]] und [[2,7,6],[2,9,3]

wird jede gelbe Zahl mit jeder roten Zahl verglichen und in dem Fall wäre die 6 eine Lösung. Dannwürden die Listen, in der diese Zahlen enthalten sind, zurückgegeben also:

[1,4,6] und [2,7,6].

## main

Hier wird zuerst das Txt-Document geöffnet und dann die erste Zeile zur Berechnung von n (siehe Lösungsidee) benutzt, dann beginnt die Simulation. Nun wird aus den möglichen Feldern der letzten Runde (bei Runde 1 aus 1 bzw. 2) die neuen Felder für jeden der beiden Spieler berechnet. Darauf folgend wird überprüft, ob diese neuen Felder doppelt auftauchen und eine der beiden Möglichkeiten gelöscht. Dabei wird die ganze Zeit, wie in controll gezeigt, mit ineinander verschachtelten Listen gearbeitet, wobei die äußere Liste für einen Spieler steht und die innere Liste für einen Weg, den dieser Spieler bestreiten kann. Nun wird noch die controll Methode aufgerufen, bevor überprüft wird, ob die Simulation beendet werden muss. Wenn die Simulation beendet worden ist, entweder, weil es eine Möglichkeit gab oder weil sie lange genug gelaufen ist wird, das Ergebnis ausgegeben.

# Beispiele

Die Beispiele von der offiziellen Webseite:

### huepfburg0.txt

Spieler 1 springt [1, 18, 13, 10]

Spieler 2 springt [2, 19, 20, 10]

### huepfburg1.txt

leider gibt es keine Möglichkeit

### huepfburg2.txt

Spieler 1 springt [1, 51, 76, 59, 116, 112, 95, 8, 51]

Spieler 2 springt [2, 106, 136, 108, 100, 12, 114, 3, 51]

### huepfburg3.txt

leider gibt es keine Möglichkeit

### huepfburg4.txt

Spieler 1 springt [1, 99, 89, 79, 78, 77, 76, 66, 56, 55, 54, 44, 43, 33, 23, 13, 12]

Spieler 2 springt [2, 12, 11, 100, 2, 12, 11, 100, 2, 12, 11, 100, 2, 12, 11, 100, 12]

# Quellcode

```
player1 = [[1]]
player2 = [[2]]
data = open("huepfburg0.txt")
jumps = []
time = 0

# Methode zu Überprüfung, ob die Spieler auf dem selben Feld stehen
def controll():
    for i in player1:
        for j in player2:
            if i[-1] == j[-1]:
                return i,j

# Umwandeln von Data in eine Liste
for line in data:
    jumps = jumps+[line[0:len(line)-1]]
```

```
# Nutzen der Ersten Zeile der Liste um eine maxmile Länge der Simulation zu bestimmen
counts = jumps[0].split()
counts = int(counts[0])
often = (counts*counts)/2+counts # often = maximale Länge

status = "main"

# Starten der Simulation
while status == "main":
    time = time +1
    # für jedes Feld auf dem Spieler 1 stehen kann, werden die nächsten möglichen Felder bestimmt
    extra = []
    for i in player1:
        for j in jumps:
            j = j.split()
            first = int(j[0])
            second = int(j[1])
            if first == i[-1]:
                extra = extra+[i+[second]]
    player1 = extra

    # das gleiche für Spieler 2
    extra = []
    for i in player2:
        for j in jumps:
            j = j.split()
            first = int(j[0])
            second = int(j[1])
```

```
        if first == i[-1]:
            extra = extra+[i+[second]]
    player2 = extra

    # alle mehrfach auftretenden Zielfelder werden gelöscht
    x = 0
    for i in player1:
        y=0
        x=x+1
        for j in player1:
            y=y+1
            if i[-1] == j[-1] and x != y:
                del player1[y-1]

    # dito für Spieler 2
    x = 0
    for i in player2:
        y=0
        x=x+1
        for j in player2:
            y=y+1
            if i[-1] == j[-1] and x != y:
                del player2[y-1]

    # Überprüfung, ob beide auf dem selben Feld stehen können
    ergebniss = controll()
    if ergebniss != None:
        status = "end"
    # Test, ob die Simulation beendet werden soll
```

```
    if time == counts+1:
        status = "stop"

# Ergebnisausgabe
if status == "stop":
    print("leider gibt es keine Möglichkeit")

if status == "end":
    player1_springt = str(ergebniss[0])
    print("Spieler 1 springt " + player1_springt)
    player2_springt = str(ergebniss[1])
    print("Spieler 2 springt " + player2_springt)
```