Aufgabe 4: Würfelglück

Team-ID: 01048

Team: Lorian

Bearbeiter/-innen dieser Aufgabe:
Lorian Linnertz

17. November 2021

Inhaltsverzeichnis

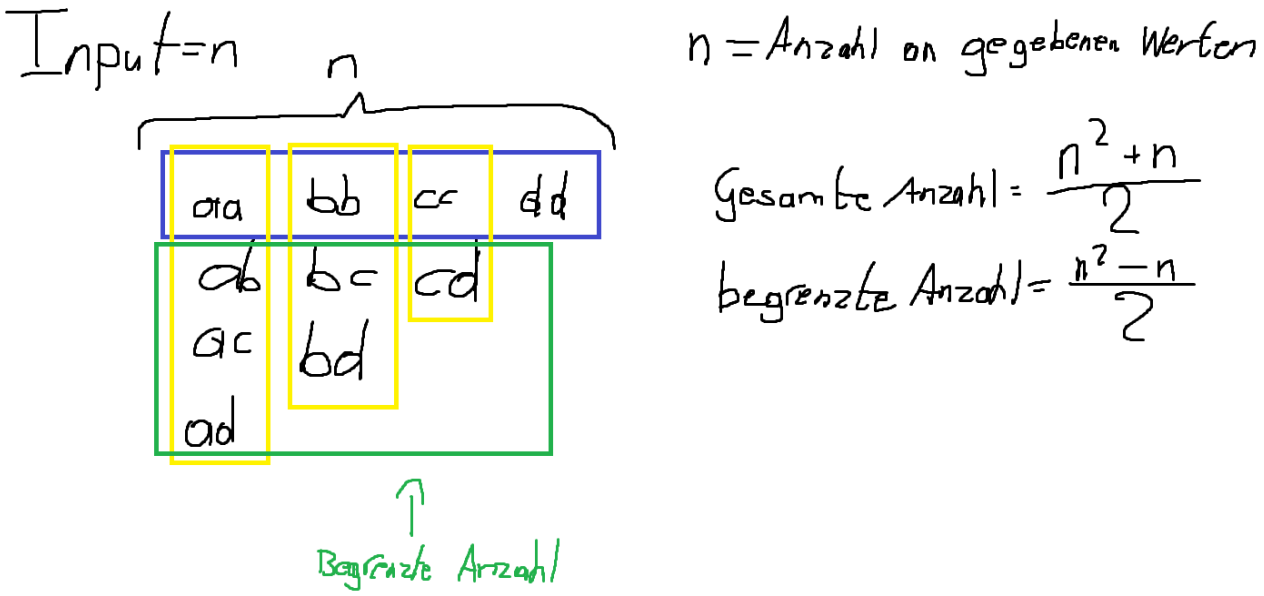
Lösungsidee

Mein Lösungsansatz um herauszufinden welcher Würfel sich am besten eignet ist eine Simulation. Beduetet, dass man für jedes Würfelpaar eine bestimmte Anzahl an Runden, ich habe als Standartwert 100 Runden gennommen (je mehr Runden man nimmt, um so kleiner wird der Zufallsfaktor auf das Resultat) simuliert und dann schaut, welcher der Würfel das Duell gewonnen hat, der Würfel mit den meisten gewonnen Duellen ist am Ende der beste Würfel.

Umsetzung

Simulation: Die Simulation habe dadurch gelöst in dem ich drei Klassen erstellt habe. Eine Klasse für die Figuren, eine für die Spieler und eine für das Spielbrett. Der Hintergedanke ist, dass die Spielfeld Klasse entscheidet wer am Zug ist und auch überprüft ob ein Spieler einen anderen geschlagen hat. Die Klasse des Spielers ist für die Kontrolle und Bewegung der Figuren verantwortlich. Um die Simulation zu vereinfachen geht man einfach von 2 unabhängigen 1. Dimensionalen Spielfeldern mit einer länge von 44 Felder so wie 4 B-Felder aus(Für jeden Spieler 1 die Klasse Gamefield beinhaltet diese beiden Spielfelder). Um jetzt die beiden Spielfelder (Welche um 20 verschoben sind) zu vergleichen muss man einfach von der Figur von dem einen Spielfeld minus 20 rechnen. Dies ergibt dann die Position der Figur auf dem anderen Spielfeld. Sollte die Zahl negativ sein so muss man zu dieser negativen Zahl 40 hinzuaddieren. Jeder Zug wird mit den Spielregeln auf die Richtigkeit abgeglichen und es wird überprüft ob es ein Sieger gibt.

Ebenfalls wichtig ist, dass jeder Würfel gegen jeden in einem 1 gegen 1 antritt. Dazu verwendet man die hier gezeigte Begrenzte Anzahl an Kombinationen.



Beispiele

In der Würfelliste "WUERFEL0.TXT" ist der Würfel [1, 1, 1, 6, 6, 6] der Beste mit 5 gewonnen Duellen

In der Würfelliste "WUERFEL1.TXT" ist der Würfel [1, 2, 3, 4, 5, 6] der Beste mit 5 gewonnen Duellen

In der Würfelliste "WUERFEL2.TXT" ist der Würfel [1, 6, 6, 6, 6, 6] der Beste mit 4 gewonnen Duellen

In der Würfelliste "WUERFEL3.TXT" ist der Würfel [1, 2, 5, 6] der Beste mit 5 gewonnen Duellen

Quellcode

```
class Figure:
    def __init__(self):
        self.position = -1

    def goToA(self):
        self.position = 0

    def goBackToB(self):
        self.position = -1
```

```
    def move_forward(self, distance):
        self.position += distance

#-----------------------------------------------------------------------------------------------------------------------

class Player:
    def __init__(self,dice_numbers,name):
        self.dice_numbers = dice_numbers
        self.fList = [Figure(),Figure(),Figure(),Figure()]
        self.name = name



    def random_dice(self):
        randomIndex = randint(0,len(self.dice_numbers)-1) #Züfälliger Index aus der Länge der Liste des Würfels. Da die Methode len() bei 1 anfängt zu zählen und die Indexe bei 0 anfangen muss man minus 1 rechnen um diesen Unterschied auszugleichen
        return self.dice_numbers[randomIndex]


    def control_Movment(self,randomNum): #entscheidet ob eine Figure auf das A Feld geht, oder ob eine Figur vom A Feld weiterläuft


        if Player.CheckFieldA(self)[0] and Player.CheckMovement(self,Player.CheckFieldA(self)[1],randomNum):
            self.fList[Player.CheckFieldA(self)[1]].move_forward(randomNum) #Führt die Funktion move_forward() von der Figur aus, welche auf dem Feld A steht; mit self.fList, greift man auf die Liste der Figuren zu und mit CheckFieldA(self)[1] bekommt man den Index der Figur auf der Position A wieder
            #print("hat sich von Feld A entfernt")


        elif (randomNum == 6 and Player.CheckFieldA(self)[0] != True and Player.CheckFieldB(self)[0]):
            #print(f"Player{self.name} hat eine Figur aus dem Haus")
            self.fList[Player.CheckFieldB(self)[1]].goToA()


        else:
            Player.move_Figure(self,randomNum)






    def CheckFieldA(self): #"""Note"""
        for i in range(0,len(self.fList)): #Kontolliert ob eine Figur auf dem Feld 0 steht
            if self.fList[i].position == 0:
```

```
            return [True,i] #gibt zum einen zurück ob eine Figur auf B ist und zum anderen dden Index von fList dieser Figur
        return [False, None]
    def CheckFieldB(self): #"""Note"""
        for i in range(0,len(self.fList)): #Kontolliert ob eine Figur auf dem Feld -1 (auch die Felder B genannt) steht
            if self.fList[i].position == -1:
                return [True,i] #gibt zum einen zurück ob eine Figur auf B ist und zum anderen dden Index von fList dieser Figur
        return [False, None]


    def CheckMovement(self,fListIndex,movement_range): #"""""NOTE"""#Diese Funktion soll überprüfen, ob der Spieler bei einem Zug auf ein Feld kommt auf dem bereits einer seiner Figuren steht
        if fListIndex == None:
            return False
        deltaPosition = self.fList[fListIndex].position + movement_range #Die Variable deltaPosition gibt an wo sich die Figur nach dem Zug befinden würde
        for f in self.fList:
    #        print(f.position)
            if f.position in range(0,44) and deltaPosition == f.position: #Überprüft als erstes ob f.position zwische 0 und 43 liegt, anschliessend wird überprüft, ob deltaPosition den Gleichen Wert wie f.position hat
                return False #gibt den Wert False zurück, was soviel bedeutet, wie der Zug ist nicht möglich
        return True


    def move_Figure(self, randomNum): #"""Note"""
        positionList = [] #als erstes Bestimmen wir welche Figur am weitesten vorne liegt
        for figure in self.fList:
            positionList.append(figure.position)
        sortedPostition = SortIndexes_MaxToMin(positionList) #Gibt eine List mit den Sortierte Indexen zurücke, welche nach ihren Postionen absteigend sortiert sind
        for positionIndex in sortedPostition:
            if (self.fList[positionIndex].position + randomNum) in range(1,44) and self.fList[positionIndex].position in range(1,44) and Player.CheckMovement(self,positionIndex,randomNum): #überpruft ob die Figur nach dem Zug noch auf dem Spielfeld ist
                self.fList[positionIndex].move_forward(randomNum) #bewegt die Figur nach vorne
                break


#-----------------------------------------------------------------------------------------------------------------------------------------------------------


class GameField:
    def __init__(self,Player1,Player2):
        self.pList = [Player1,Player2]
        self.lastMove = []
        self.control_Bool = True
        self.Result = None
```

```
        self.TurnCounter = 0


    def Decide_FirstPlay(self): # dieses Program entscheidet wer anfängt, dabei gilt, wert die grösse Zahl hat darf anfangen
        FirstPlay_control_Bool = True
        while(FirstPlay_control_Bool): #Diese Schleife dient dazu, falls beide Würfel den gleichen Wert haben, dass so lange gewürfel wird bis unterschieliche Wert herauskommen
            p1_startnumber = self.pList[0].random_dice()
            p2_startnumber = self.pList[1].random_dice()
            if p1_startnumber > p2_startnumber:
                FirstPlay_control_Bool = False
                GameField.GameControl(self,0,1)
            elif p1_startnumber < p2_startnumber:
                FirstPlay_control_Bool = False
                GameField.GameControl(self,1,0)



    def GameControl(self,first,second):
        while(self.control_Bool):
            while True:
                Firstplayer_Dice = self.pList[first].random_dice()
                GameField.PlayerController(self,first,Firstplayer_Dice)

                if GameField.control_Winner(self)[0] or Firstplayer_Dice != 6:
                    break
            if GameField.control_Winner(self)[0]:
                break
            #-----------------------------------------------------
            while True:
                Secondplayer_Dice = self.pList[second].random_dice()
                GameField.PlayerController(self,second,Secondplayer_Dice)
                if GameField.control_Winner(self)[0] or Secondplayer_Dice != 6:
                    break
            if GameField.control_Winner(self)[0]:
                break
            #-----------------------------------------------------------
            if self.TurnCounter >= 1000:
                self.control_Bool = False
                self.Result = None #Falls es zu keinem Ergebnis kommen sollte und die Runde länger als 1000 Runden für jeden dauern sollte, so wird der Wert None zurück gegeben; Dies kommt nur vor, wenn keiner der beiden Würfel, ins Ziel kommt
```

```
            break
        #print(self.TurnCounter)
        self.TurnCounter += 1
    self.Result = GameField.control_Winner(self)[1]


def PlayerController(self,status,dice): #mit Status ist gemein ab es first oder second ist und mit dice ist die zufällige Zahl gemeint
    self.pList[status].control_Movment(dice)
    GameField.control_LastMove(self,status)



def control_LastMove(self, lastplayedPlayer): # diese Funktion überprüft ob der lastplayedPlayer auf dem Feld einer gegnerischen Figur ist #die Variable lastplayedPlayer gibt den Index des letzteb gespielten Spieler
    if lastplayedPlayer == 0: #überprüft ob der Player1 gespielt hat
        FiguresPositions_OfPlayer1 = [] #in diese Variable werden die Positionen der Figuren von Player1 gespeichert
        for i in self.pList[0].fList: #iteriert durch die Positionen aller Figuren von Player1 um sie der Liste beizufügen
            if i.position in range(0,40):
                enemyposition = i.position -20 #diese Variable beschreibt die Position aus der sicht des Gegners, da das Spielfeld 40 Felder Lang ist und die Spieler sich gegenüber stehen. für genauere Info Erklärungsdokument auf Goodnotes
                if isNegatif(enemyposition): #falls die enemyposition negatif ist, so wirdd 40 drauf addiert damit die Zahl wieder positif ist
                    enemyposition = enemyposition +40
                FiguresPositions_OfPlayer1.append(enemyposition) #es werden nur Positionen hinzugefügt, welche auf dem Spielfeld stehen, bedeutet dass die Figuren sich zwischen 0 und 39 befinden
        for figure in self.pList[1].fList: #iteriert durch alle gegnerischen Figuren durch
            if figure.position in FiguresPositions_OfPlayer1: #falls die Position der Figur mit einer der Positionen der des Player1 übereinstimmt, so muss die Figur wieder zurück auf -1
                figure.goBackToB()
    elif lastplayedPlayer == 1: #überprüft ob der Player2 gespielt hat
        FiguresPositions_OfPlayer2 = [] #in diese Variable werden die Positionen der Figuren von Player2 gespeichert
        for i in self.pList[1].fList: #iteriert durch die Positionen aller Figuren von Player2 um sie der Liste beizufügen
            if i.position in range(0,40):
                enemyposition = i.position -20 #diese Variable beschreibt die Position aus der sicht des Gegners, da das Spielfeld 40 Felder Lang ist und die Spieler sich gegenüber stehen. für genauere Info Erklärungsdokument auf Goodnotes
                if isNegatif(enemyposition): #falls die enemyposition negatif ist, so wirdd 40 drauf addiert damit die Zahl wieder positif ist
                    enemyposition = enemyposition +40
                FiguresPositions_OfPlayer2.append(enemyposition) #es werden nur Positionen hinzugefügt, welche auf dem Spielfeld stehen, bedeutet dass die Figuren sich zwischen 0 und 39 befinden
        for figure in self.pList[0].fList: #iteriert durch alle gegnerischen Figuren durch
            if figure.position in FiguresPositions_OfPlayer2: #falls die Position der Figur mit einer der Positionen der des Player2 übereinstimmt, so muss die Figur wieder zurück auf -1
                figure.goBackToB()
```

```
    def control_Winner(self):
        if (self.pList[0].fList[0].position in range(40,44) and self.pList[0].fList[1].position in range(40,44) and
self.pList[0].fList[2].position in range(40,44) and self.pList[0].fList[3].position in range(40,44)):
            self.control_Bool = False #Kontolliert ob ein Spieler gewonnen hat, in dem geschaut wird op jede Figur im Ziel ist
            #print(f"Player {self.pList[0].name} hat gewonnen mit dem Würfel", self.pList[0].dice_numbers)
            return [True, self.pList[0].name]
        elif self.pList[1].fList[0].position in range(40,44) and self.pList[1].fList[1].position in range(40,44) and
self.pList[1].fList[2].position in range(40,44) and self.pList[1].fList[3].position in range(40,44):
            self.control_Bool = False
            #print(f"Player {self.pList[1].name} hat gewonnen mit dem Würfel", self.pList[1].dice_numbers)
            return [True, self.pList[1].name] #bestätigt zuerst, dass es einen Sieger gibt und sagt anschliessend wer es ist
        return [False, None] #gibt zurück, dass es noch keinen Sieger gibt
```