Aufgabe 5: Marktwaage

Team-ID: 01048

Team: Lorian

Bearbeiter/-innen dieser Aufgabe:
Lorian Linnertz

17. November 2021

Inhaltsverzeichnis

Lösungsidee

Meine Lösungsidee war, dass ich den Betrag aus der Differenz von Zielgewicht und dem aktuellen Gewicht nehme und dann das Gewicht hinzufüge oder abziehe, welches am nächsten an diesem Wert dran ist. Wenn das aktuelle Gewicht grösser ist als das Ziel Gewicht, so wird das neue Gedicht abgezogen und wenn das aktuelle Gewicht kleiner als das Zielgewicht ist, so wird das neue Gewicht dazu addiert. Jedes Gewicht welches benutzt wurde wird entfernt, wenn am Ende kein Gewicht mehr übrig ist und wir nicht beim Zielgewicht sind, so ist es nicht möglich das Zielgewicht mit den gegebenen Gewichten zu erreichen.

Umsetzung

Als erstes habe ich eine Schleife Gemacht welche von 0 bis 10000 in 10er Schritten durch iteriert.

In dieser Schleife läuft eine weitere While Schleife, welche solange läuft bis keine Gewichte mehr da sind oder das Zielgewicht erreicht worden ist. Im ersten Schritt wird mit dem in der Lösungsidee bereits beschriebenem Verfahren das beste Gewicht rausgesucht und ebenfalls wie in der Lösungsidee beschrieben dem aktuellen Gewicht angefügt. Am Ende eines jeden durchlaufs durch die While-Schleife wird überprüft ob wir das Zielgewciht erreicht haben und es wir das verwendete Gewicht mit der pop() Funktion aus der Liste der gegebenen Gewichte entfernt. Diese Liste wird bei jedem Durchgang der for – Schleife erneuert

Beispiele

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE0.TXT", kann man 993 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 99.3%

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE1.TXT", kann man 25 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 2.5%

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE2.TXT", kann man 1000 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 100.0%

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE3.TXT", kann man 336 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 33.6%

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE4.TXT", kann man 106 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 10.6%

Mit Hilfe von den Gewichten aus "GEWICHTSSTUECKE5.TXT", kann man 0 der 1000 möglich Gewichte nachstellen, das entspricht einer Quote von 0.0%

Quellcode

```
def Find_Min(list):
    minIndex=0              #Definiert die Variable x als int oder float
    minNum = list[minIndex]    #Die Variable minNum,bekommt als Startwert die erste Zahl aus einer gegebenen Liste, diese wird dann immer durch die nächst kleiner Zahl ersetz, solange bis keine kleinere Zahl mehr in der Liste ist.


    for i in range(0,len(list)): #iteriert durch alle Indexe der gegebenen Liste durch
        if (list[i] < minNum):   #Vergleicht ob das Element mit dem Index [i] aus einer gegebenen Liste kliner ist als die bislang kleinste gefundene Zahl
            minNum = list[i]    #Falls eine kleinere Zahl als minNum gefunden wird, so wird minNum durch diese Zahl ersetzt
            minIndex = i        #ebenfalls wird der Index dieser neuen kliensten Zahl aktualisiert


    return [minNum, minIndex]   #gibt eine Liste zurück mit der Form [minNum, minIndex]


def NearestWeight(mass, weightsList): #diese Funktion ermittelt den das am nächsten gelegene Gewicht
    mass = (mass**2)**0.5
    valueList = []
    for i in range(0,len(weightsList)):
        valueList.append(((mass- weightsList[i])**2)**0.5)
    return Find_Min(valueList)



def CheckIfItFit(weightsList):
    ResultList_Bool = []
```

```
for mass in range(10,10010,10):
    weights = []
    weights.extend(tuple(weightsList))
    controlBool = True
    deltaMass = 0
    while True: #versucht sich solange wie möglich dem Zielgewicht anzunähern
        difference = mass - deltaMass
        index_deltaWeights = NearestWeight(difference, weights)[1]

        if deltaMass < mass:
            deltaMass += weights[index_deltaWeights]
        elif deltaMass > mass:
            deltaMass -= weights[index_deltaWeights]
        if deltaMass == mass:
            ResultList_Bool.append(True)
            break
        elif len(weights) == 1:
            ResultList_Bool.append(False)
            break

        weights.pop(index_deltaWeights)
return ResultList_Bool
```