Algorithmus und Programm

Informatik

Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, insbesondere der automatischen Verarbeitung mit Hilfe von Rechenanlagen (Wikipedia).

Wir beschäftigen uns zu Beginn mit Algorithmen und deren Programmierung in Python.

Ein *Algorithmus* ist eine endlich lange Vorschrift, bestehend aus Einzelanweisungen.

Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, insbesondere der automatischen Verarbeitung mit Hilfe von Rechenanlagen (Wikipedia).

Wir beschäftigen uns zu Beginn mit Algorithmen und deren Programmierung in Python.

Ein *Algorithmus* ist eine endlich lange Vorschrift, bestehend aus Einzelanweisungen.

Ein in einer Computersprache formulierter Algorithmus heißt *Programm*.

Eine umgangssprachliche Formulierung, die die Struktur des Algorithmus deutlich macht, nennen wir *Pseudocode*.

Der Collatz-Algorithmus in Pseudocode

```
lies x ein
setze z auf 0
solange x ≠ 1 tue
    wenn x gerade
          dann halbiere x
    sonst
          verdreifache x und erhoehe um 1
    erhoehe z um 1
gib z aus
```

Der Collatz-Algorithmus in Pseudocode

```
lies x ein
setze z auf 0
solange x ≠ 1 tue
    wenn x gerade
          dann halbiere x
    sonst
          verdreifache x und erhoehe um 1
    erhoehe z um 1
gib z aus
```

Um zu prüfen, was ein Algorithmus macht, ist es manchmal hilfreich, ein Ablaufprotokoll zu erstellen. Dabei werden die Werte der (wichtigsten) beteiligten Variablen schrittweise mitverfolgt.

Eingabe $= 3 \rightarrow$ Collatz-Algorithmus $\rightarrow$ Ausgabe $=$

Der Collatz-Algorithmus in Pseudocode

```
lies x ein
setze z auf 0
solange x ≠ 1 tue
    wenn x gerade
          dann halbiere x
    sonst
          verdreifache x und erhoehe um 1
    erhoehe z um 1
gib z aus
```

Um zu prüfen, was ein Algorithmus macht, ist es manchmal hilfreich, ein Ablaufprotokoll zu erstellen. Dabei werden die Werte der (wichtigsten) beteiligten Variablen schrittweise mitverfolgt.

Eingabe $= 3 \rightarrow$ Collatz-Algorithmus $\rightarrow$ Ausgabe $= 7$

Beim Collatz-Algorithmus reicht es, den Wert der eingegebenen Zahl zu verfolgen und anschließend die Anzahl der Durchgänge zu zählen:

14 -

Beim Collatz-Algorithmus reicht es, den Wert der eingegebenen Zahl zu verfolgen und anschließend die Anzahl der Durchgänge zu zählen:

14 - 7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1

Eingabe $= 14 \rightarrow$ Collatz-Algorithmus $\rightarrow$ Ausgabe $=$

Beim Collatz-Algorithmus reicht es, den Wert der eingegebenen Zahl zu verfolgen und anschließend die Anzahl der Durchgänge zu zählen:

14 - 7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1

Eingabe $=$ 14 $\rightarrow$ Collatz-Algorithmus $\rightarrow$ Ausgabe $=$ 17

Beim Collatz-Algorithmus reicht es, den Wert der eingegebenen Zahl zu verfolgen und anschließend die Anzahl der Durchgänge zu zählen:

14 - 7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1

Eingabe $=$ 14 $\rightarrow$ Collatz-Algorithmus $\rightarrow$ Ausgabe $=$ 17

Der Algorithmus wurde 1937 von Lothar Collatz formuliert. Es ist ein bis heute ungelöstes mathematisches Problem, ob dieser Algorithmus für jede Eingabe zu einem Ende kommt.

Der Collatz-Algorithmus in Python

Der Collatz-Algorithmus in Python

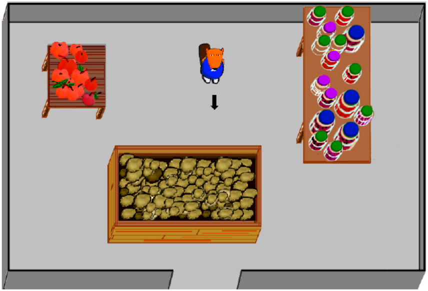
```
x = int(input("Bitte eine Zahl eingeben: "))
z = 0
while x != 1:
    if x % 2 == 0:
        x = x // 2
    else:
        x = 3 * x + 1
    z = z + 1
print(z)
```

Der Pledge-Algorithmus:

Ausweg im Dunkeln

Mitten in der Nacht muss der Biber den Weg aus einem unbekannten Keller finden.
Er weiß nur, dass die Wände und alle anderen Hindernisse
in rechten Winkeln angeordnet sind.
Der Biber hat folgende Regeln gelernt, wie man einen Ausweg findet.
Die Regeln arbeiten mit einem Zähler, der zu Beginn Null ist:

- Drehst du dich 90 Grad nach rechts, dann erhöhe den Zähler um eins.
- Drehst du dich 90 Grad nach links, dann erniedrige den Zähler um eins.
- Ist der Zähler Null, dann gehe solange geradeaus, bis du auf ein Hindernis stößt.
- Stößt du auf ein Hindernis, dann drehe dich 90 Grad nach rechts und gehe solange an dem Hindernis entlang (auch um Ecken herum), bis der Zähler Null ist.



Welches sind die Werte des Zählers auf dem Weg des Bibers nach draußen?

A) 0, 1, 0, 1, 2, 3, 4, 3, 2, 3, 4, 5, 4
B) 0, -1, 0, 1, 0
C) 0, 1, 0, 1, 2, 3, 4, 3, 4, 5, 4
D) 0, 1, 0, -1, 0

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 = 2 \cdot 2 \cdot 13$
$\text{ggT}(28, 52) =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 = 2 \cdot 2 \cdot 13$
$\text{ggT}(28, 52) = 4$

Bestimmung des ggT von 60 und 90 durch Primfaktorzerlegung:
$60 =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 = 2 \cdot 2 \cdot 13$
$\text{ggT}(28, 52) = 4$

Bestimmung des ggT von 60 und 90 durch Primfaktorzerlegung:
$60 = 2 \cdot 2 \cdot 3 \cdot 5$
$90 =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 = 2 \cdot 2 \cdot 13$
$\mathrm{ggT}(28, 52) = 4$

Bestimmung des ggT von 60 und 90 durch Primfaktorzerlegung:
$60 = 2 \cdot 2 \cdot 3 \cdot 5$
$90 = 2 \cdot 3 \cdot 3 \cdot 5$
$\mathrm{ggT}(60, 90) =$

Algorithmen zur Bestimmung des ggT (größter gemeinsamen Teiler)

Bestimmung des ggT von 28 und 52 durch Primfaktorzerlegung:
$28 = 2 \cdot 2 \cdot 7$
$52 = 2 \cdot 2 \cdot 13$
$\text{ggT}(28, 52) = 4$

Bestimmung des ggT von 60 und 90 durch Primfaktorzerlegung:
$60 = 2 \cdot 2 \cdot 3 \cdot 5$
$90 = 2 \cdot 3 \cdot 3 \cdot 5$
$\text{ggT}(60, 90) = 30$

```
# Algorithmus ggtDumm
Lies zwei positive ganze Zahlen ein
Setze ggt = erste Zahl
Solange nicht beide Zahlen durch ggt teilbar:
    erniedrige ggt um 1
Gib ggt aus
```

Implementation in Python:

```
# Algorithmus ggtDumm
Lies zwei positive ganze Zahlen ein
Setze ggt = erste Zahl
Solange nicht beide Zahlen durch ggt teilbar:
    erniedrige ggt um 1
Gib ggt aus
```

Implementation in Python:

```python
a = int(input())
b = int(input())
ggt = a
while (a % ggt != 0 or b % ggt != 0):
    ggt -=1
print(ggt)
```

Implementation als Funktion:

```
# Algorithmus ggtDumm
Lies zwei positive ganze Zahlen ein
Setze ggt = erste Zahl
Solange nicht beide Zahlen durch ggt teilbar:
    erniedrige ggt um 1
Gib ggt aus
```

Implementation in Python:

```python
a = int(input())
b = int(input())
ggt = a
while (a % ggt != 0 or b % ggt != 0):
    ggt -=1
print(ggt)
```

Implementation als Funktion:

```python
def ggtDumm(a, b):
    ggt = a
    while (a % ggt != 0 or b % ggt != 0):
        ggt -=1
    return ggt
```

Beobachtung von Euklid:
Wenn t Teiler von a und b ist, dann ist t auch Teiler von $a - b$ (falls $a > b$).

52 28

Beobachtung von Euklid:
Wenn t Teiler von a und b ist, dann ist t auch Teiler von $a - b$ (falls $a > b$).

Beobachtung von Euklid:
Wenn t Teiler von a und b ist, dann ist t auch Teiler von $a - b$ (falls $a > b$).

52	28
24	28
24	4
20	4
16	4
12	4
8	4
4	4

```
# Euklidscher Algorithmus
Ziehe von der größeren Zahl
die kleinere ab, solange bis beide gleich sind.
```

52	28
24	28
24	4
20	4
16	4
12	4
8	4
4	4

Beobachtung von Euklid:
Wenn t Teiler von a und b ist, dann ist t auch Teiler von $a - b$ (falls $a > b$).

```
# Euklidscher Algorithmus
Ziehe von der größeren Zahl
die kleinere ab, solange bis beide gleich sind.

def ggt(a,b):
    while a != b:
        if (a > b):
            a = a - b
        else:
            b = b - a
    return a
```

Beobachtung: Immer wenn die größere Zahl die Seiten wechselt, können wir die neue Zahl aus den beiden oberen berechnen.

128	34
94	34
60	34
26	34
26	8
18	8
10	8
2	8
2	6
2	4
2	2

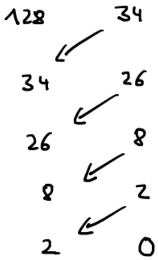
Beobachtung: Immer wenn die größere Zahl die Seiten wechselt, können wir die neue Zahl aus den beiden oberen berechnen.

128	34
94	34
60	34
26	34
26	8
18	8
10	8
2	8
2	6
2	4
2	2

128	34
34	26
26	8
8	2
2	0

128	34
94	34
60	34
26	34
26	8
18	8
10	8
2	8
2	6
2	4
2	2

Beobachtung: Immer wenn die größere Zahl die Seiten wechselt, können wir die neue Zahl aus den beiden oberen berechnen.




```
# moderner euklidscher Algorithmus
def ggtTurbo(a,b):
    while b != 0:
        a, b = b, a % b
    return a
```