Rekursion

Thanh Viet Nguyen, Fourat Zai El Amri, Luis Sperling, Daniel Pessler, Mikhail Safonov

Hochschule Hannover

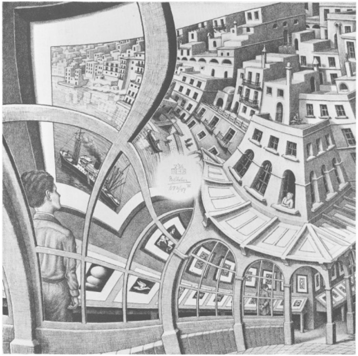
15.11.2024

Agenda

Einleitung

- Was ist Rekursion?
- Wozu ist sie da?

- Rekursion bedeutet wörtlich Zurückführen.
- Rekursion liegt dann vor, wenn eine Funktion, ein Algorithmus, eine Datenstruktur, ein Begriff, etc. durch sich selbst definiert wird.



M. C. Escher, Bildergalerie, 1956.

- Linear verkettete Listen und Bäume (später) sind rekursiv definierte Datentypen.
- Beispiel: linear verkettete Liste

```
class Node {
    Node next;
    int  data;
    // ...
}
```

Node wird durch sich selbst definiert.

- Eine rekursive Funktion ist eine Funktion, die sich selbst aufruft.
- Beispiel: Fakultätsfunktion

fak(0) = 1

fak(n) = n * (n-1) *… * 2 * 1 = n * fak(n-1)   falls n ≥ 1

```
static int fak(int n) {
    if (n == 0)
        return 1;
    else
        return n * fak(n-1) ;
}
```

fak ruft sich selbst auf.

```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```

```
main()
  ↓
fak(3):
if(3 == 0)
   return 1;
else
   return 3*fak(2);
```

https://sharelatex.gwdg.de/project/66f655b805396533edc0c4be

```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```

```
main()

fak(3):
if(3 == 0)
   return 1;
else
   return 3*fak(2);

fak(2):
if(2 == 0)
   return 1;
else
   return 2*fak(1);
```

```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```

```
main()
```

fak(3):

```
if(3 == 0)
   return 1;
else
   return 3*fak(2);
```

fak(2):

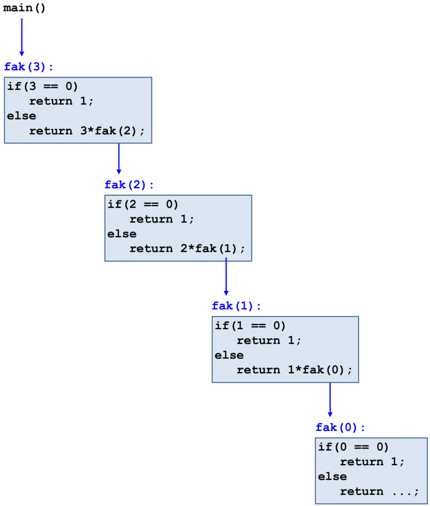
```
if(2 == 0)
   return 1;
else
   return 2*fak(1);
```

fak(1):

```
if(1 == 0)
   return 1;
else
   return 1*fak(0);
```

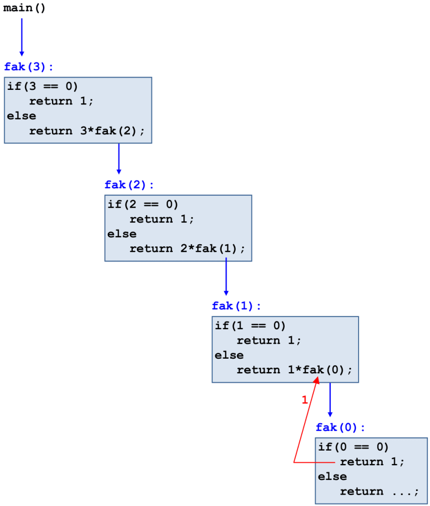
```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```



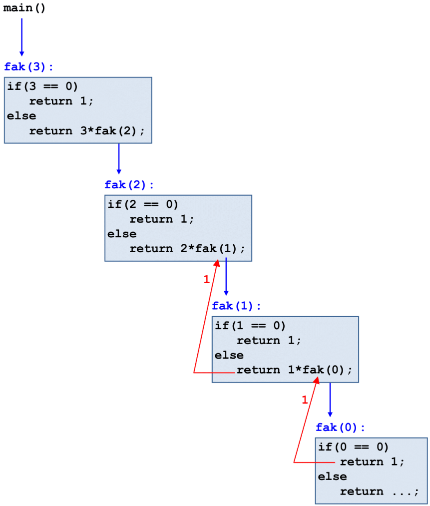
```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```



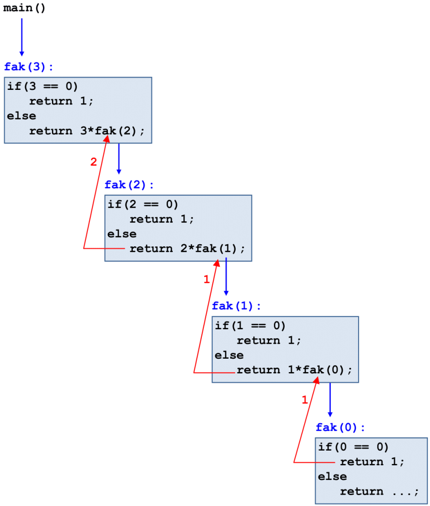
```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```



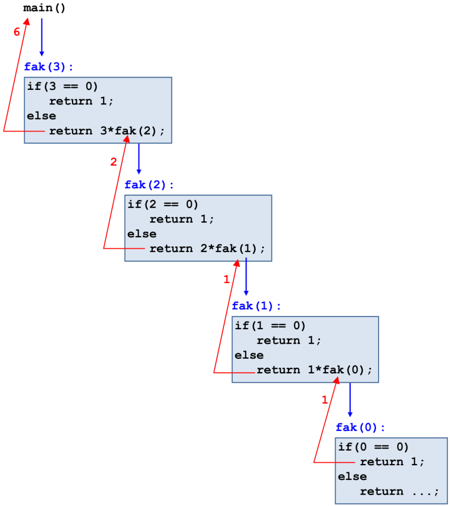
```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```



```
void main(){
    fak(3);
}

static int fak(int n){
    if (n == 0)
        return 1;
    else
        return n*fak(n-1) ;
}
```



Aufgaben

1. Die Türme von Hanoi
2. Die Fibonacci-Folge
3. Zusäzliche aufgabe

Die Türme von Hanoi

Es soll ein Algorithmus gefunden werden, der den Turm von Stab A auf den Stab C versetzt.

- man darf immer nur eine Scheibe pro Zug umlegen
- eine größere Scheibe darf nie auf einer kleineren Scheibe liegen
- eine kleinere Scheibe darf auf eine größere Scheibe liegen

```python
# hanoi.py
def hanoi(n, start, end, helper):
  if n == 1:
  print(start + "->" + end)
  else:
    hanoi(n-1, start, helper, end)
    print(start + "->" + end)
    hanoi(n-1, helper, end, start)

def user_input():
  return int(input())

if __name__ == "__main__":
  n = user_input()
  hanoi(n, 'A', 'C', 'B')
```

Die Fibonacci-Folge

Die Fibonacci-Zahlen sind eine berühmte Folge natürlicher Zahlen, die in vielen Bereichen der Mathematik und Naturwissenschaften eine besondere Bedeutung haben. Sie wurden vom italienischen Mathematiker Leonardo von Pisa, auch bekannt als *Fibonacci*, im Jahr 1202

Die Fibonacci-Folge

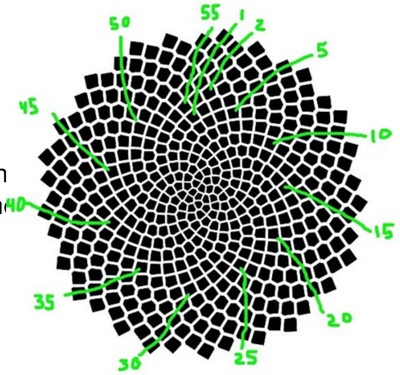
Die Fibonacci-Folge wird *rekursiv* definiert:

$$a_n = \begin{cases} 1, & \text{für } n = 1 \\ 1, & \text{für } n = 2 \\ a_{n-1} + a_{n-2}, & \text{für } n > 2 \end{cases}$$

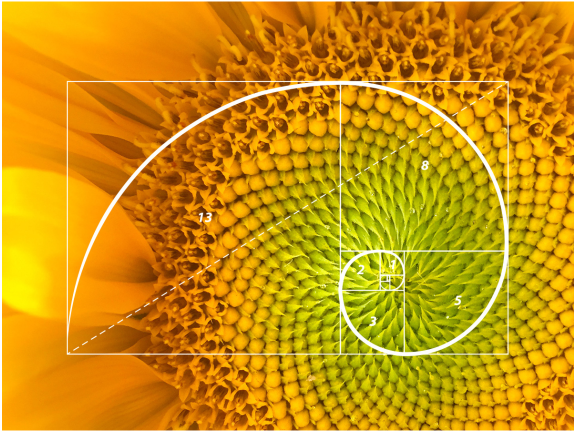
Diese einfache rekursive Struktur führt zu einer faszinierenden Folge, die oft in der Natur und in mathematischen Zusammenhängen auftaucht, wie in der Verteilung von Blättern, in Spiralmustern, und in Wachstumsprozessen.

Fibonacci-Zahl in der Natur

Viele Samenköpfe, Tannenzapfen und Früchte zeigen spiralförmige Muster, die beim Zählen Fibonacci-Zahlen ausdrücken. Schaut man auf die Spiralen der Same im Zentrum einer Sonnenblume erkennt man Muster, die sich nach links und rechts winden.
Wenn man diese Spiralen zählt, ergibt die Summe eine Fibonacci-Zahl.



Quelle: https://momath.org/home/fibonacci-numbers-of-sunflower-seed-spirals/



Quelle: https://clevelanddesign.com/insights/the-nature-of-design-the-fibonacci-sequence-and-the-golden-ratio/

1. **Nachteile der Rekursiven Form:**
   - Um einen Term zu berechnen, müssen alle vorherigen Terme berechnet werden.
   - Hoher Rechenaufwand bei großen n, was zu ineffizientem Ressourcenverbrauch führt.
   - Besonders in der Programmierung kann dies zu langen Berechnungszeiten führen.
2. **Vorteile der Geschlossenen Form:**
   - Direkte Berechnung von a_n, ohne vorherige Terme berechnen zu müssen.
   - Spart Rechenzeit und Ressourcen, besonders bei großen n.
   - Sehr effizient für schnelle und ressourcenschonende Berechnungen.

Eine geschlossene Darstellung liefert die Formel von Moivre-Binet:

$$a_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right]$$

Eine geschlossene Darstellung liefert die Formel von Moivre-Binet:

$$a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

Um die Berechnung zu vereinfachen, setzen wir $\alpha = \frac{1+\sqrt{5}}{2}$ und $\beta = \frac{1-\sqrt{5}}{2}$.

a) Beweisen durch vollständige Induktion:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} \quad \text{für alle } n \in \mathbb{N}$$

a) Beweisen durch vollständige Induktion:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} \quad \text{für alle } n \in \mathbb{N}$$

b) Rechnen Sie nach:

i

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix} = \frac{1+\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix}$$

a) Beweisen durch vollständige Induktion:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} \quad \text{für alle } n \in \mathbb{N}$$

b) Rechnen Sie nach:

i

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix} = \frac{1+\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix}$$

ii

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \frac{1-\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1 + \beta \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1+\beta \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{1}{\beta}+1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1+\beta \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{1}{\beta}+1 \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{2}{1-\sqrt{5}}+1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1+\beta \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{1}{\beta}+1 \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{2}{1-\sqrt{5}}+1 \end{pmatrix}$$

$$= \beta \begin{pmatrix} 1 \\ \frac{1-2\sqrt{5}+5}{2(1-\sqrt{5})} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1+\beta \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{1}{\beta}+1 \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{2}{1-\sqrt{5}}+1 \end{pmatrix}$$

$$= \beta \begin{pmatrix} 1 \\ \frac{1-2\sqrt{5}+5}{2(1-\sqrt{5})} \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{(1-\sqrt{5})^2}{2(1-\sqrt{5})} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \beta \\ 1+\beta \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{1}{\beta} + 1 \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{2}{1-\sqrt{5}} + 1 \end{pmatrix}$$

$$= \beta \begin{pmatrix} 1 \\ \frac{1-2\sqrt{5}+5}{2(1-\sqrt{5})} \end{pmatrix} = \beta \begin{pmatrix} 1 \\ \frac{(1-\sqrt{5})^2}{2(1-\sqrt{5})} \end{pmatrix} = \frac{1-\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix}$$

$\square$

1 a) Beweisen Sie durch vollständige Induktion:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} \quad \text{für alle } n \in \mathbb{N}$$

b) Rechnen Sie nach:

i

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix} = \frac{1+\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix}$$

ii

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} = \frac{1-\sqrt{5}}{2} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix}$$

iii

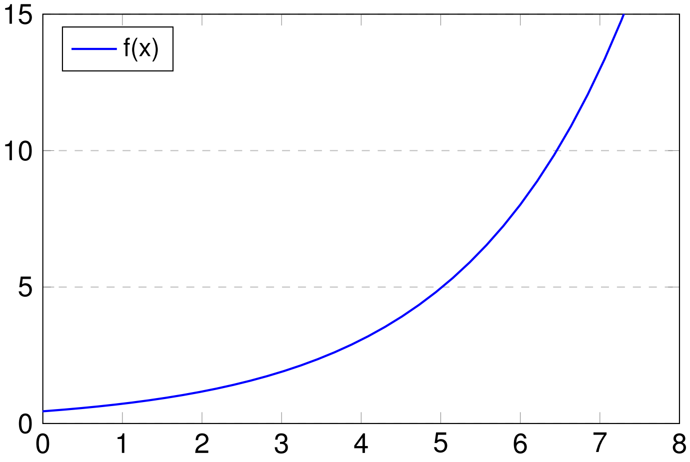
$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{5}} \cdot \left[\begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix} - \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix} \right]$$

Wir zeigen dass:

$$\begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} = \frac{1}{\sqrt{5}} \cdot \begin{pmatrix} \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \\ \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} \end{pmatrix}$$

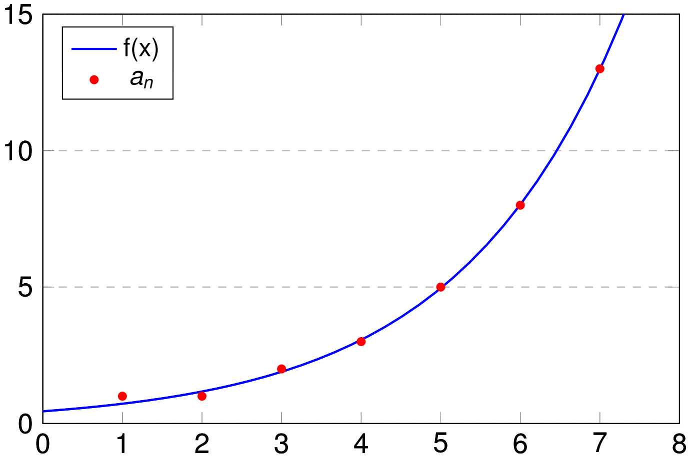
zunächst sei die Funktion $f(x) = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^x$

Graph von :$f(x) = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^x$



Desmos

Graph von :$f(x) = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^x$



Desmos

Aufgabe 3.4.i):

Zu zeigen ist, dass der Abstand zwischen $f(n)$ und a_n immer kleiner wird, wobei $n \in \mathbb{N}$ ist.
bzw.

$$\lim_{n \to \infty} |f(n) - a_n| = 0$$

Aufgabe 3.4.ii):

$$a_n = \lfloor f(n) + 0.5 \rfloor \quad \text{für alle } n \in \mathbb{N}$$

Dabei ist $\lfloor \cdot \rfloor$ die Abrundungsfunktion. $\lfloor \cdot \rfloor$ wird auch als Gauß-Klammer bezeichnet und die Formel $\lfloor \cdot + 0.5 \rfloor$ beschreibt das übliche Runden auf die nächste ganze Zahl.

Aufgabe 3.5):

```
n=5
def Fibo (n):
    if n==0:
        return 1
    if n==1:
        return 1
    else:
        n=Fibo(n-1)+Fibo(n-2)
        return n
print("a_",n,"=", Fibo(n))
```

```
n=8
def aufruf(n):
    if n==0 or n==1:
        return 1
    else:
        return aufruf(n-2)+aufruf(n-1)+1

def Fibo (n):
    if n==0:
        return 1
    if n==1:
        return 1
    else:
        n=Fibo(n-1)+Fibo(n-2)
        return n
print("a_"+str(n)+" =", Fibo(n))
print("benoetigte aufrufe= ",aufruf(n))
```

```python
counter=0
n=5
def Fibo (n):
    global counter
    counter+=1
    if n==0:
        return 1
    if n==1:
        return 1
    else:
        n=Fibo(n-1)+Fibo(n-2)
        return n
print("a_",n,"=", Fibo(n))
print("Benoetigte Aufrufe=", counter)
```

```python
from functools import cache
counter=0
n=5
@cache
def Fibo (n):
    global counter
    counter+=1
    if n==0:
        return 1
    if n==1:
        return 1
    else:
        n=Fibo(n-1)+Fibo(n-2)
        return n
print("a_",n,"=", Fibo(n))
print("Benoetigte Aufrufe=", counter)
```

Aufgabe 3.6
Die Zahl

$$\frac{1+\sqrt{5}}{2} \approx 1.61803$$

wird als Goldener Schnitt bezeichnet.
Zeigen Sie mit Hilfe der Formel von Moivre-Binet, dass

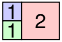
$$\lim_{n\to\infty} \frac{a_{n+1}}{a_n} = \frac{1+\sqrt{5}}{2}.$$

Goldene Schnitt

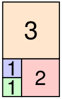
1

Goldene Schnitt

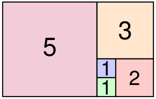
Goldene Schnitt



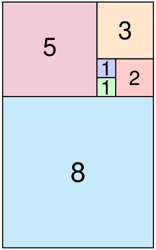
Goldene Schnitt



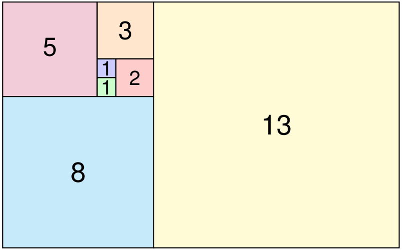
Goldene Schnitt



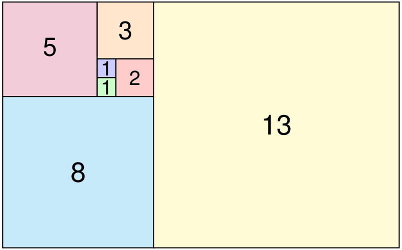
Goldene Schnitt



Goldene Schnitt

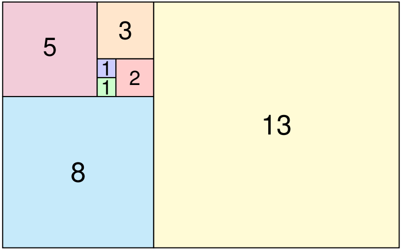


Goldene Schnitt



Die fläche des Rechteckes
$1^2 + 1^2 + 2^2 + 3^2 + 5^2 + 8^2 + 13^2$

Goldene Schnitt



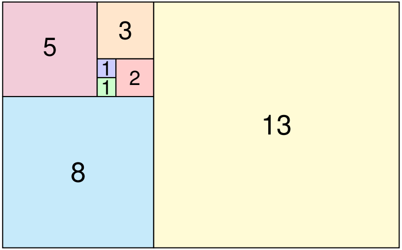
Die fläche des Rechteckes

$1^2 + 1^2 + 2^2 + 3^2 + 5^2 + 8^2 + 13^2$

Aber auch :

$13 \times (8 + 13)$

Goldene Schnitt



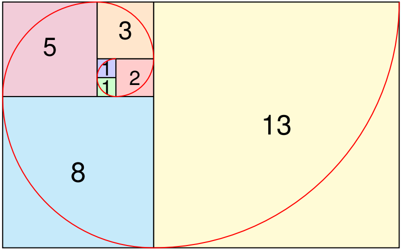
Die fläche des Rechteckes

$1^2 + 1^2 + 2^2 + 3^2 + 5^2 + 8^2 + 13^2$

Aber auch :

$13 \times (8 + 13) \quad = 13 \times 21$

Goldene Schnitt



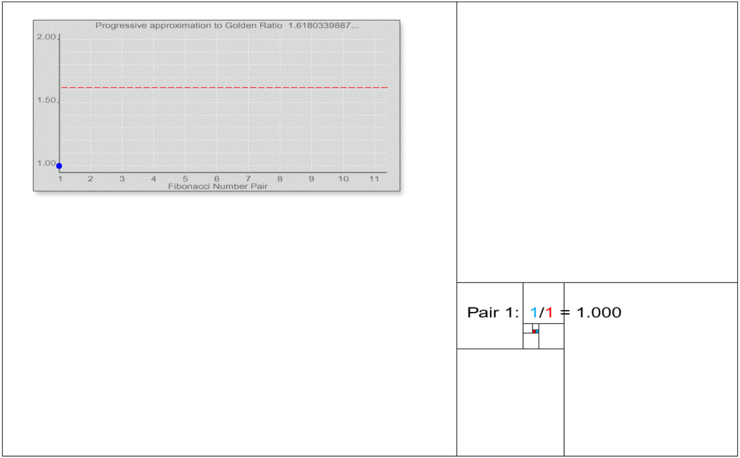
Die fläche des Rechteckes

$1^2 + 1^2 + 2^2 + 3^2 + 5^2 + 8^2 + 13^2$

Aber auch :

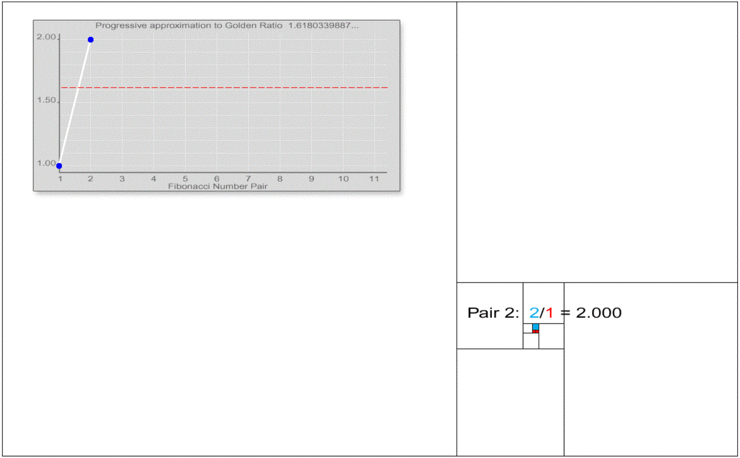
$13 \times (8 + 13) \quad = 13 \times 21$

Der Goldene Schnitt



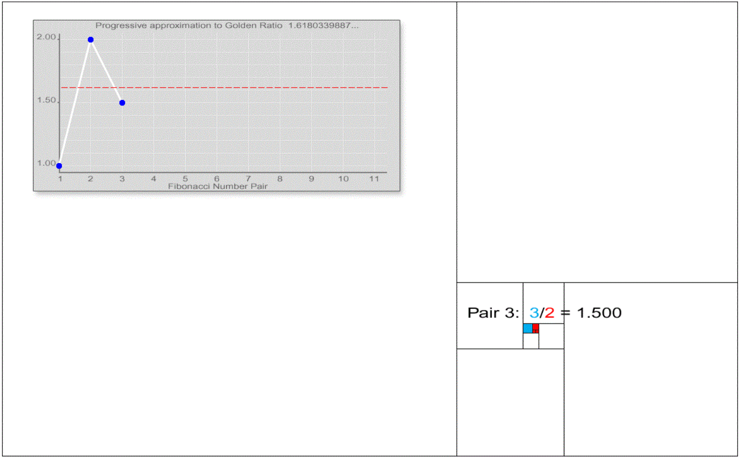
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



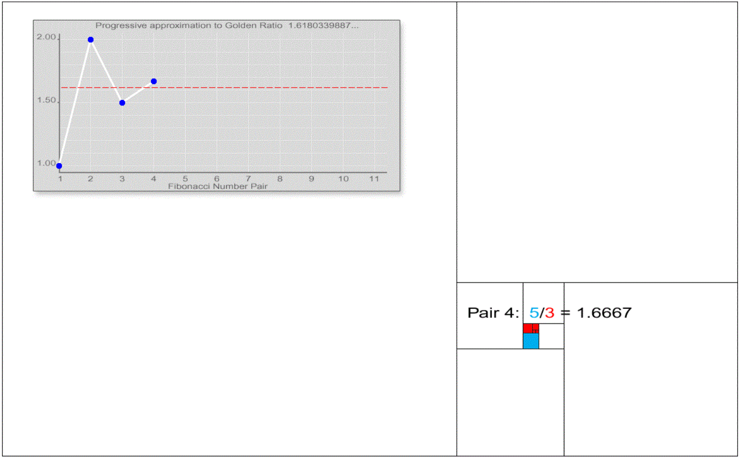
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



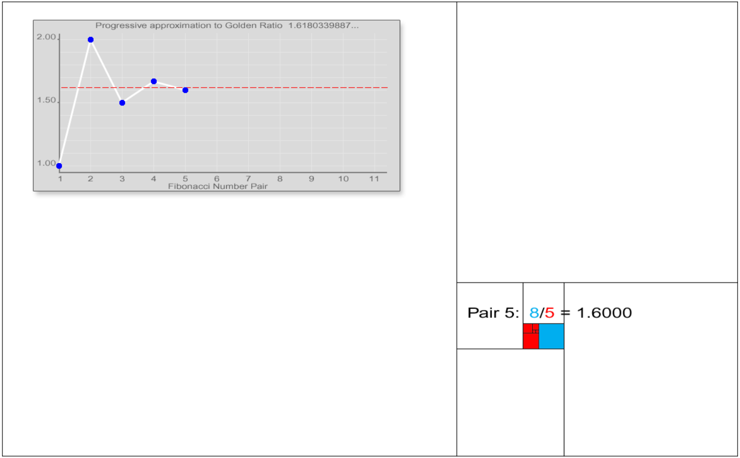
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



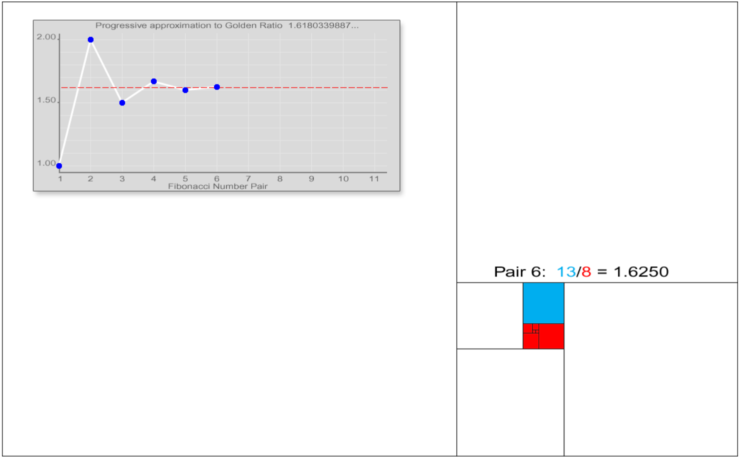
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



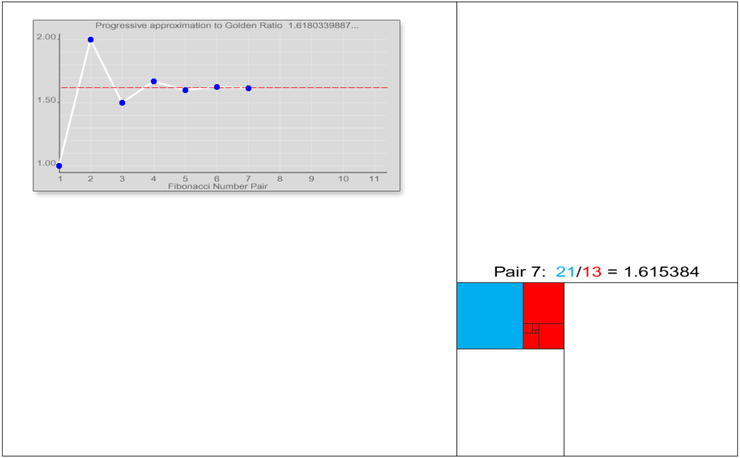
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



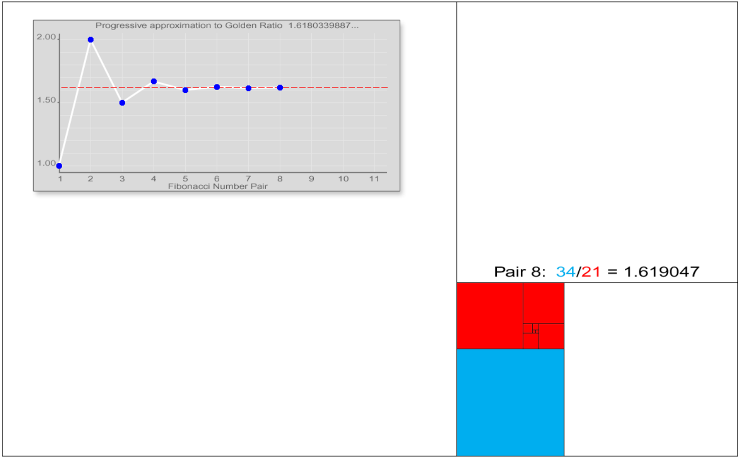
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



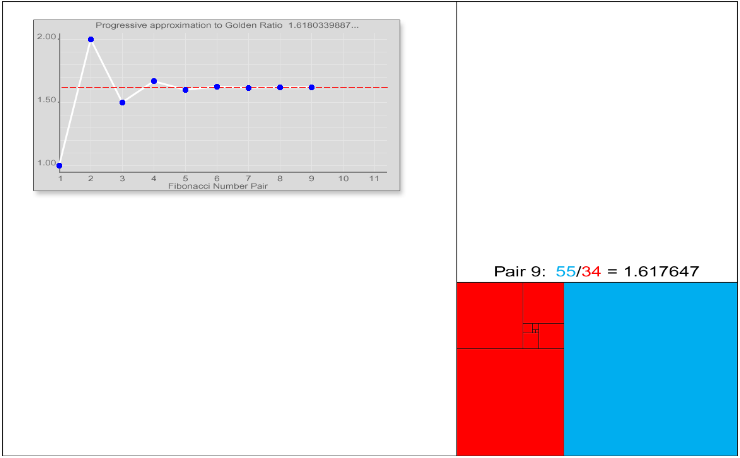
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



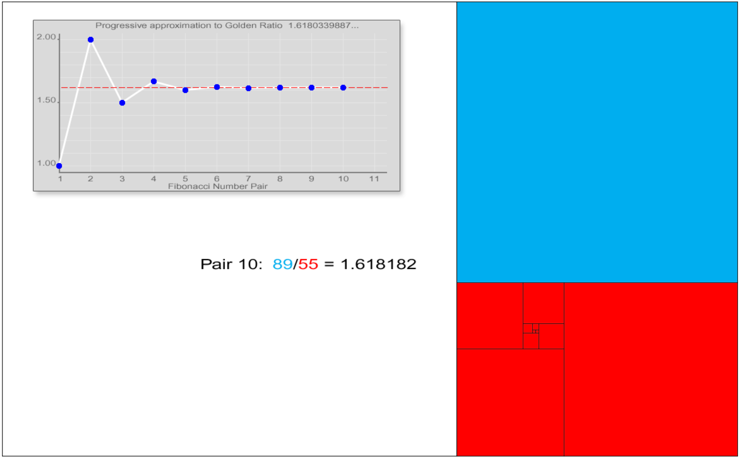
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



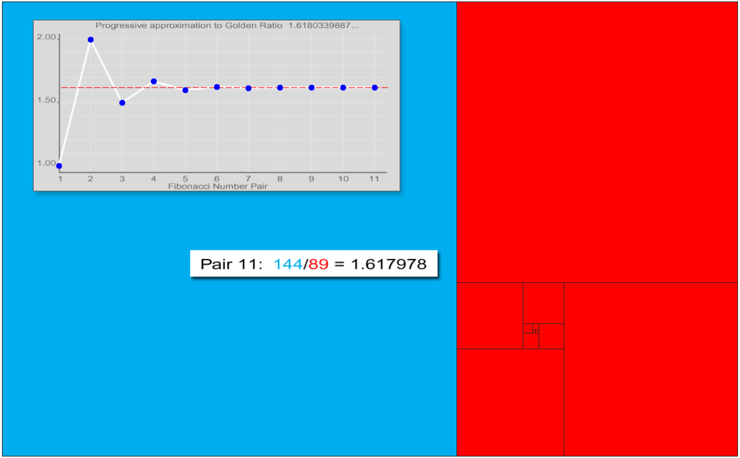
Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



Quelle: Wikipedia: FibonacciSequence

Der Goldene Schnitt



Quelle: Wikipedia: FibonacciSequence

Zusatz Aufgaben

Finden eine rekursive Formel für die Wahrscheinlichkeit, dass n Personen alle an verschiedenen Tagen im Jahr Geburtstag haben. Dabei können wir vereinfachend davon ausgehen, dass das Jahr 365 Tage hat und die sich die Geburtstage rein zufällig auf die diese 365 Tage verteilen.

Ansatz: $P_n = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \ldots \frac{365-n+1}{365}$

Allgemein gilt: $P(n) = P(n-1) \cdot \frac{364-(n-1)}{365}$

3.9 a) Rekursive Formel

Rekursive Formel für die Fakultät $n!$:

$$n! = \begin{cases} 1 & \text{,Wenn } n = 0 \\ n \times (n-1)! & \text{,Wenn } n > 0 \end{cases}$$

3.9 a) Code zur Formel

```
def fakultaet(n):
    if n == 0:
        return 1
    else:
        return n * fakultaet(n - 1)

# Nutzereingabe
try:
    zahl = int(input("Bitte geben Sie eine natürliche Zahl ein: "))
    if zahl < 0:
        print("Bitte geben Sie eine nicht-negative Zahl ein.")
    else:
        print(f"Die Fakultät von {zahl} ist {fakultaet(zahl)}")
except ValueError:
    print("Ungültige Eingabe. Bitte geben Sie eine ganze Zahl ein.")
```

✓ 0.0s

Rekursive Formel für den Binomialkoeffizienten $\binom{n}{k}$:

$$\binom{n}{k} = \begin{cases} 1 & \text{,Wenn } k = 0 \text{ oder } k = n \\ 0 & \text{,Wenn } k > n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{, Andern falls} \end{cases}$$

3.9 b) Code Rekursive Formel

```python
# 3.9b in Python
def binomial_coefficient(n, k):
    # Basisfälle
    if k == 0 or k == n:
        return 1
    if k > n:
        return 0
    # Rekursive Berechnung
    return binomial_coefficient(n - 1, k - 1) + binomial_coefficient(n - 1, k)

# Hardcode input
n = 1
k = 2

binomial_coefficient(n, k)

print(f"Der Binomialkoeffizient von {n} über {k} ist: {binomial_coefficient(n, k)}")
```

Aufgabe 3.10

betrachtet werden die folgende rekursiv definierte Folge:

$$b_n = \begin{cases} 1, \text{für } n = 1 \\ \sqrt{1 + b_{n-1}}, \text{für } n > 2 \end{cases}$$

Beweisen dass diese Folge gegen den goldenen Schnitt konvergiert:

$$\lim_{n\to\infty} b_n = \frac{1+\sqrt{5}}{2}$$

Um zu zeigen, dass die rekursiv definierte Folge b_n gegen den goldenen Schnitt $\phi = \frac{1+\sqrt{5}}{2}$ konvergiert, betrachten wir die Definition der Folge:

$$b_n = \begin{cases} 1, & \text{für } n = 1 \\ \sqrt{1 + b_{n-1}}, & \text{für } n > 1 \end{cases}$$

Zu Zeigen, dass die Folge beschränkt und monoton ist:
Zu zeigen, dass die Folge b_n beschränkt ist.
Vermutung: $b_n < \phi$ für alle n.
Beweis durch Vollständiger Induktion. **Induktionsanfang:** Für $n = 1$: $b_1 = 1 < \phi$.
Induktionsannahme: Angenommen, $b_k < \phi$ für ein $k \geq 1$.
Induktionsschritt: Zeigen, dass $_{k+1} < \phi$:

$$b_{k+1} = \sqrt{1 + b_k} < \sqrt{1 + \phi}$$

Da $\phi = \frac{1+\sqrt{5}}{2}$, gilt:
$\sqrt{1+\phi} = \sqrt{1 + \frac{1+\sqrt{5}}{2}} = \sqrt{\frac{3+\sqrt{5}}{2}}$. Zu zeigen, dass $\sqrt{\frac{3+\sqrt{5}}{2}} < \phi$:

$$\phi^2 = \left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{1+2\sqrt{5}+5}{4} = \frac{6+2\sqrt{5}}{4} = \frac{3+\sqrt{5}}{2}.$$

Somit ist $\sqrt{1+\phi} < \phi \Rightarrow b_{k+1} < \phi$.
Durch Induktion folgt, dass $b_n < \phi$ für alle n.

Zu Zeigen, dass die Folge b_n monoton wachsend ist, d.h. $b_n < b_{n+1} \forall n$.
Zeigen vollständige Induktion: Induktionsanfang: Für $n = 1$ haben wir $b_1 = 1 < b_2 = \sqrt{1+1} = \sqrt{2}$.
Induktionsannahme: Angenommen, $b_k < b_{k+1}$
Induktionsschritt: Wir zeigen, dass $b_{k+1} < b_{k+2}$:

$$b_{k+2} = \sqrt{1 + b_{k+1}}$$

Da $b_k < b_{k+1}$, gilt $1 + b_k < 1 + b_{k+1}$.
Da die Funktion $(x) = \sqrt{1+x}$ monoton wachsend ist, folgt:

$$b_{k+1} = \sqrt{1+b_k} < \sqrt{1+b_{k+1}} = b_{k+2}$$

Somit ist b_n monoton wachsend.

Da b_n eine monoton wachsende und beschränkte Folge ist, konvergiert sie.
Sei $L = \lim_{n \to \infty} b_n$ dann gilt:

$$L = \sqrt{1 + L} => L^2 = 1 + L \implies L^2 - L - 1 = 0$$

$$=> L = \frac{1 \pm \sqrt{5}}{2}$$

Da $b_n >= 0$

$$=> L = \frac{1+\sqrt{5}}{2} = \phi$$

$$=> \lim_{n\to\infty} b_n = \frac{1+\sqrt{5}}{2}$$

Aufgabe 3.11

Die folgende rekursiv definierte Folge wird betrachtet:

$$c_n = \begin{cases} 1, & \text{für } n = 1 \\ 1 + \frac{1}{c_{n-1}}, & \text{für } n > 2 \end{cases}$$

Beweisen, dass diese Folge gegen den goldenen Schnitt

konvergiert:

$$\lim_{n \to \infty} c_n = \frac{1 + \sqrt{5}}{2}$$

Fazit

- Angewandte Mathematik angenähert
- Anwendung von Rekursion in der Programmierung
- Annäherung an der Praxis
- Natur vorkommend
- Rekursion fördert das Verständnis komplexer Probleme durch Zerlegung in einfachere Teilprobleme
- Effiziente Problemlösungsstrategien, insbesondere bei Datenstrukturen wie Bäumen und Graphen
- Wichtige Rolle in der Algorithmik, z.B. bei Sortier- und Suchalgorithmen
- Ermutigung zur kreativen Problemlösung und zur Entwicklung eleganter Lösungen

Danke für das Zuhören

Quellen: `https://github.com/SpiXFamily/Rekursion-MLL/tree/main/quellen`