3 Rekursive Funktionen

Beispiel: Fakultät

- **iterativ**

  $1! = 1$

  $2! = 1 \cdot 2 = 2$

  $3! = 1 \cdot 2 \cdot 3 = 6$

  $n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n$

```
public int fakultaet_iter(int n){
    int fak=1;
    for(int i=2; i<=n; i++){
        fak=fak*i;
    }
    return fak;
}
```

- **rekursiv**

  = zurückgehend bis zu bekannten Werten

Rekursionsvorschrift:

Funktion ruft sich selbst auf

$$fak(n) = \begin{cases} n \cdot fak(n-1) & \text{falls } n > 0 \\ 1 & \text{falls } n = 0 \end{cases}$$

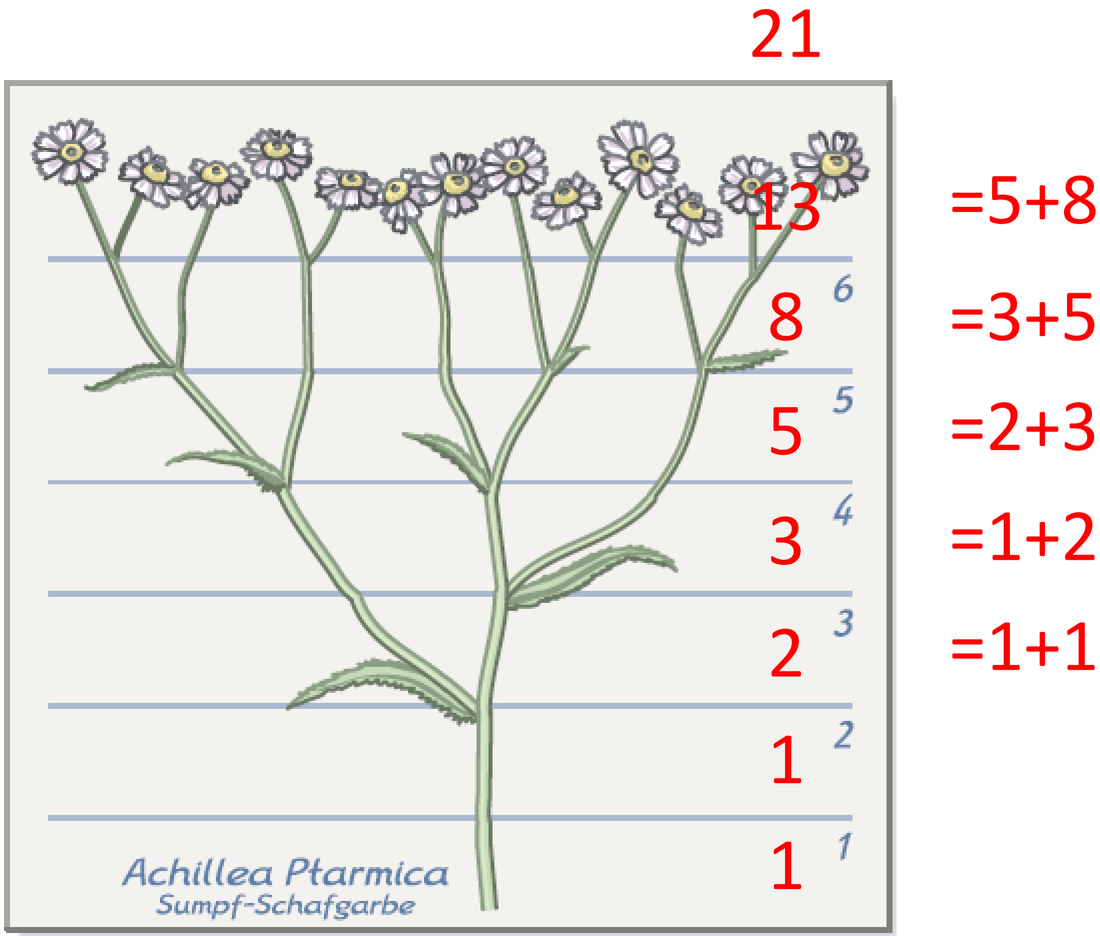
Abbruchbedingung

```java
public int fakultaet_rek(int n){
        if (n==0) {return 1;}
        else {return n*fakultaet_rek(n-1);}
}
```

Der Funktionsaufruf einer rekursiven Funktion endet nur dann, wenn die Abbruchbedingung erfüllt ist.

Lineare Rekursion:
In jedem Fall der rekursiven Definition darf höchstens ein rekursiver Aufruf vorkommen.

S. 22 Schafgarbe



Allgemein: A(n) = A(n-1)+A(n-2)

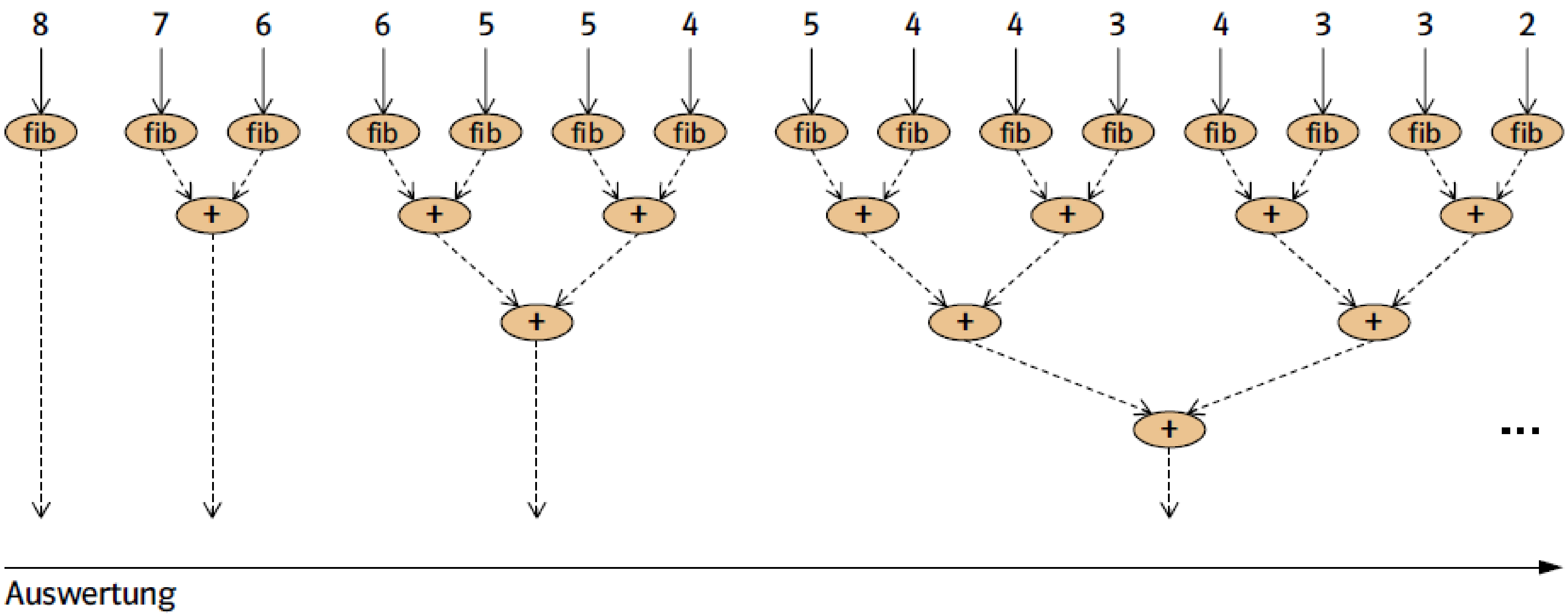
Alle Knoten der letzten Generation wachsen weiter

Alle Knoten der vorletzten Generation bilden einen neuen Trieb

Rekursionsvorschrift der Fibonacci-Zahlen:

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2) & \text{falls } n > 1 \\ 1 & \text{falls } n = 1 \\ 0 & \text{falls } n = 0 \end{cases}$$

Kaskadenartige Rekursion kann einen immensen Rechenaufwand bedeuten (vgl. Bild S. 23).



```java
public int fib(int n){
        if (n == 0) {return 0;}
        else {
            if (n==1) {return 1;}
            else {
                return fib(n-1)+fib(n-2);
            }
        }
    }
```

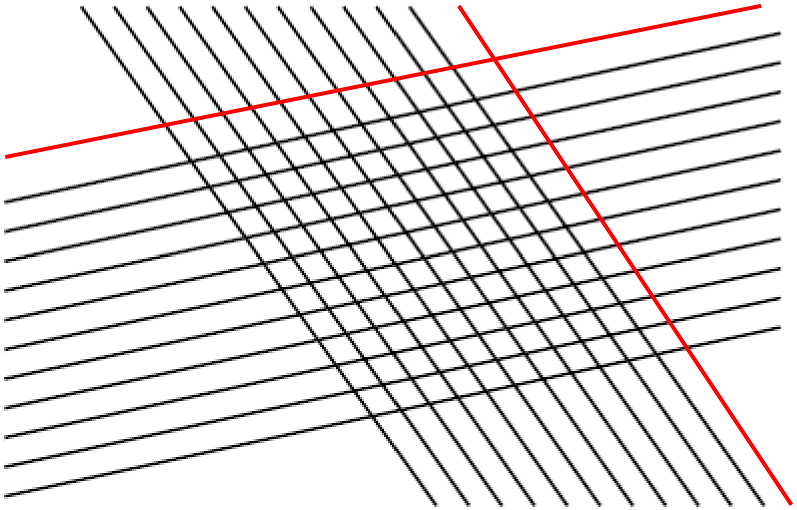
Aufrufsequenz:

fib(4) = fib(3)+fib(2)=fib(2)+fib(1)+fib(1)+fib(0)=
= fib(1)+fib(0)+1+1+0=1+1+1=3

Aufrufsequenz:

$$
\begin{aligned}
fib(4) &= fib(3)+fib(2)\\
&=fib(2)+fib(1)+fib(1)+fib(0)=\\
&= fib(1)+fib(0)+1+1+0\\
&=1+1+1=3
\end{aligned}
$$

Zur Rekursion:
S. 24/2 Gitterpunkte



Rekursive Definition:

$$A(n) = \begin{cases} A(n-1) + (n-1) + n & \text{falls } n>1 \\ 1 & \text{falls } n=1 \end{cases}$$

```
public class Gitterpunkte {
  public int anzahlSchnittpunkte(int n){
     if (n==1) return 1;
     else return anzahlSchnittpunkte(n-1)+2*n-1;
   }
}
```