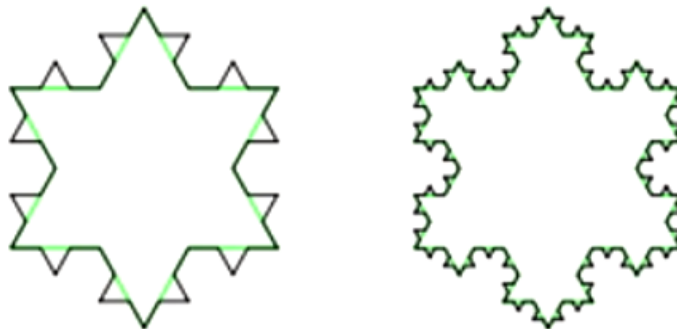
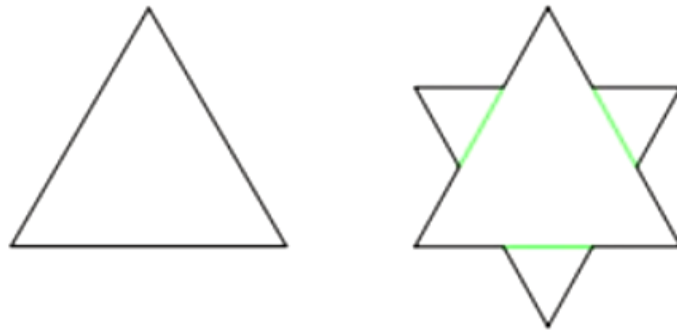


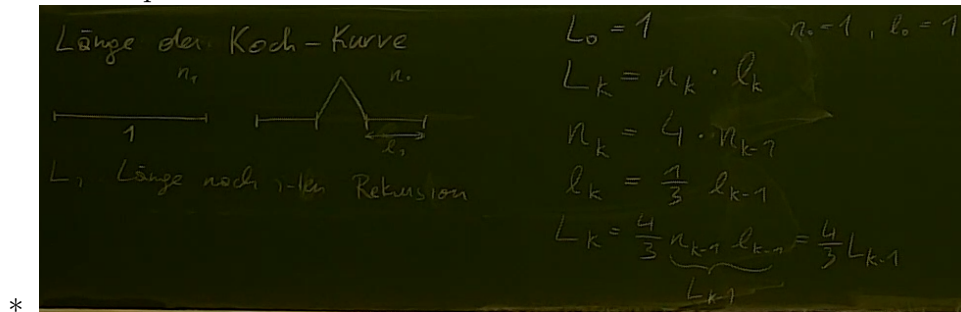
## Eigenschaften Algorithmen

- Algorithmen, welche sich selbst mit veränderten Parametern aufrufen
- benötigt Abbruchbedingung
- oft höhere Komplexität (Time/Space) als [[Iterative Algorithmen]]
  - Optimierung durch Zwischenspeichern von Ergebnissen möglich
- Koch-Flocke
  - Eigenschaften
    - \* unendlicher Umfang
    - \* begrenzte Fläche



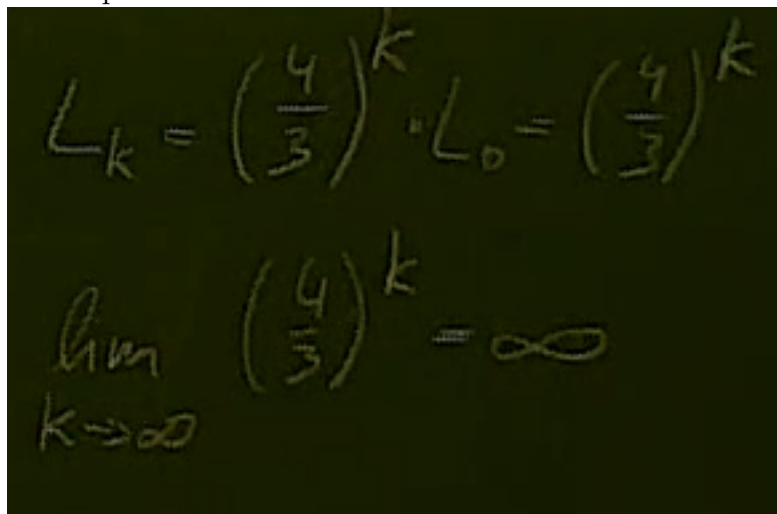
## Vergleich zwischen rekursiver und iterativer Implementation

- Koch Flocke
  - rekursive Implementation



\*

- \* iterative Implementation



◆

- Fibonacci-Zahlen
  - unterschiedliche Komplexität
  - **Beispiel:** Fibonacci-Zahlen

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 3; \quad f_1 = f_2 = 1$$

$$\Rightarrow 1, 1, 2, 3, 5, 8, 13, 21, \dots$$

```

FIBONACCI(n)
1: fib ← 1
2: fib_prev ← 1
3: FOR i ← 3 TO n
4:   fib_pprev ← fib_prev
5:   fib_prev ← fib
6:   fib ← fib_prev + fib_pprev
7: RETURN fib
  
```

```

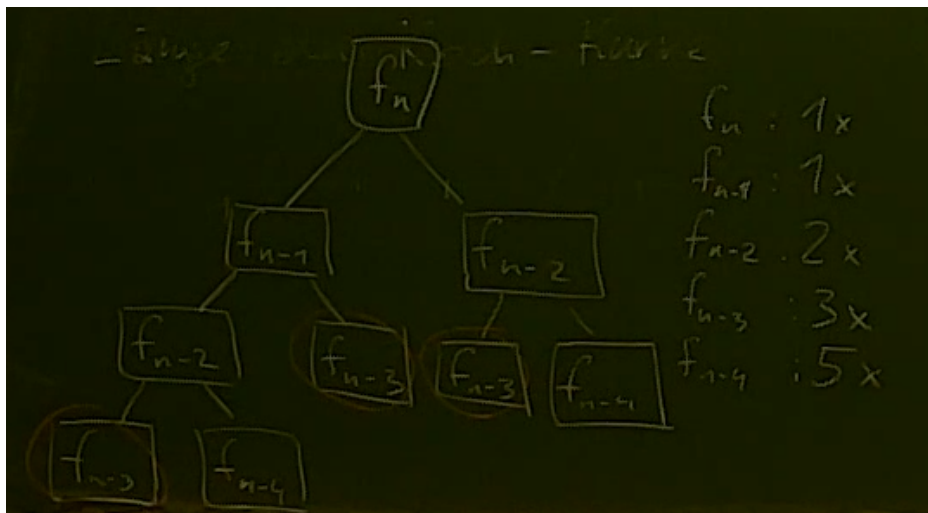
FIB_R(n)
1: IF n < 2 THEN
2:   RETURN 1
3: ELSE
4:   RETURN FIB_R(n-1) + FIB_R(n-2)
  
```

$$T(n) = O(n)$$

$$S(n) = O(1)$$

$$T(n) = \Omega\left((\sqrt{2})^n\right) \quad T(n) = O\left((\sqrt{3})^n\right)$$

$$S(n) = O(n) \text{ (entspricht Rekursionstiefe)}$$



- Optimierung durch Speichern von Zwischenergebnissen

Memorization

$A[i] \leftarrow \emptyset \quad \forall i \geq 2$   
 $A[0] \leftarrow A[1] \leftarrow 1$

In  $A[i]$  speichern wir  $f_{i+1}$

FIB\_M(n)

IF  $A[n-1] = \emptyset$

$A[n-1] \leftarrow \text{FIB\_M}(n-1) + \text{FIB\_M}(n-2)$

RETURN  $A[n-1]$

\*