dEIn Labor

an der Fakultät IV der TU Berlin

Einführung in Rekursion

FAKULTÄT IV
ELEKTROTECHNIK UND INFORMATIK

dEIn Labor.

Rekursion

Stell dir vor, du sollst jemandem, der das Bild nicht sieht, erklären, was eine Matrjoschka-Puppe ist. Du könntest zum Beispiel sagen: »Eine *Matrjoschka-Puppe* ist eine Puppe, die eine etwas kleinere *Matrjoschka-Puppe* enthält«.

So erklärst du den Begriff Matrjoschka-Puppe, **indem du in der Erklärung den Begriff selbst wieder verwendest.** *Clever*!

Aber geht das überhaupt? Wenn du genau drüber nachdenkst, dann ist die Erklärung für alle Puppen richtig, außer für die kleinste ganz innen. Denn die ist, korrekt gesehen, keine Matrjoschka-Puppe, sondern einfach nur ein Holzpüppchen, was man nicht aufmachen kann.

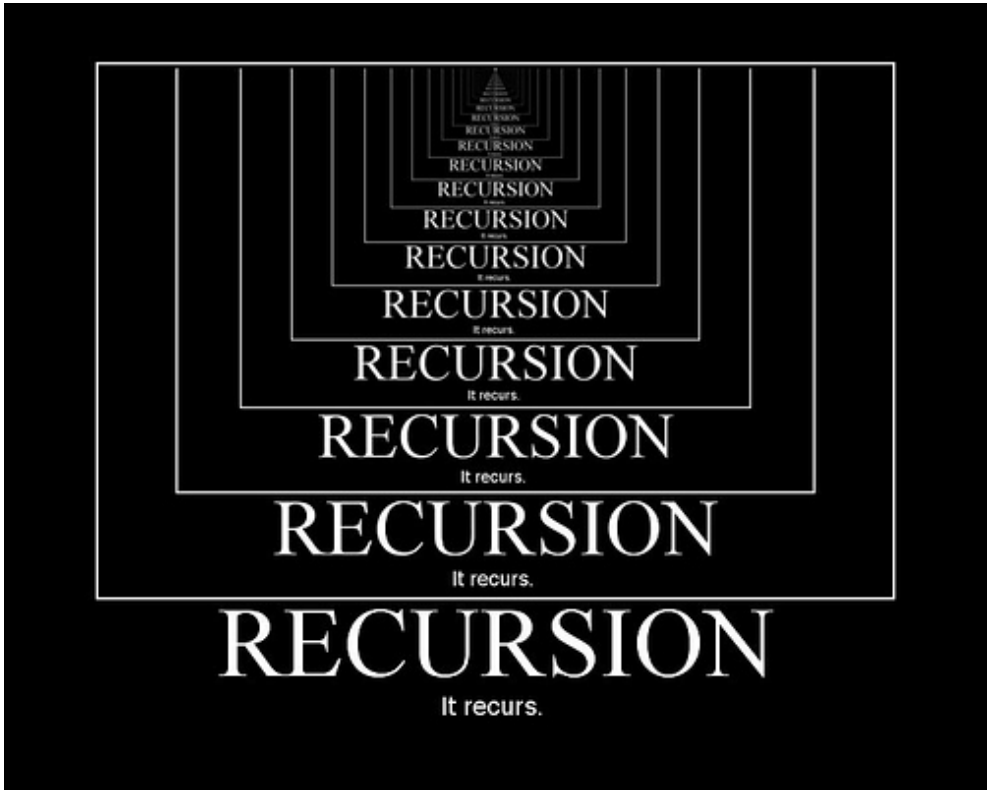
Rekursion: Was ist das?

- »Eine *Matrjoschka-Puppe* ist eine Puppe, die eine etwas kleinere *Matrjoschka-Puppe* enthält.«
- Rekursion ist eine Erklärung (Definition, Funktion) für einen Begriff, die den Begriff selbst für kleinere Varianten von sich selbst benutzt.

- *Wichtig*: Die Erklärung braucht ein Ende (eine **Abbruchbedingung**).
- »Die achte (kleinste) *Matrjoschka-Puppe* ist eine einfache Puppe.«
- Ohne Abbruchbedingung gibt es unendlich viele Wiederholungen, das können Computer nicht!
- Die Anzahl der ineinander geschachtelten Varianten von sich selbst (z.B. acht bei den Puppen) heißt »**Rekursionstiefe**«.

Rekursion in Bildern

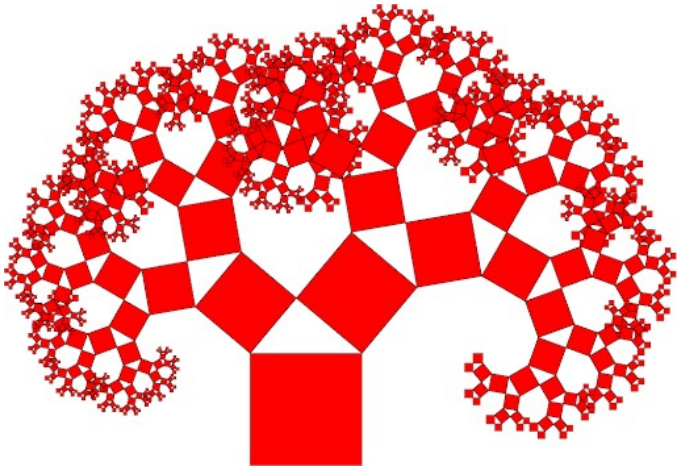
lat. „recurrere“ = zurücklaufen, wiederkehren



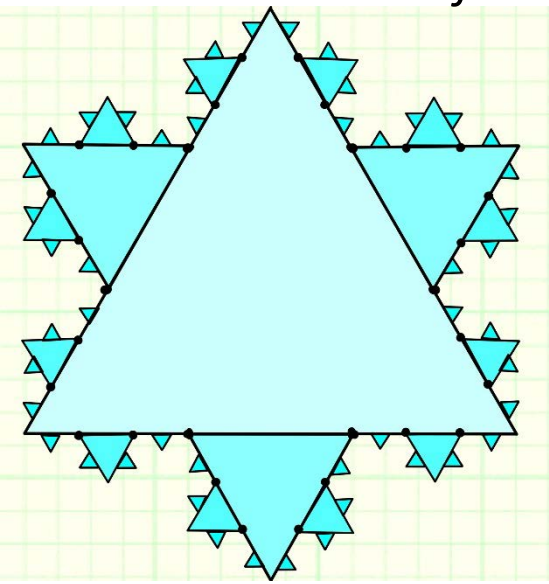
Rekursion in der Natur

Rekursion in Mustern

Pythagoras-Baum: ein Quadrat, auf dessen Oberseite zwei *Pythagoras-Bäume* schräg stehen.



Kochsche Schneeflocke: ein Dreieck, wo auf jedem Linienstück im mittleren Drittel eine *Kochsche Schneeflocke* sitzt.



Sierpinsky-Dreieck: ein Dreieck, das aus drei *Sierpinsky-Dreiecken* besteht

Rekursion in der Mathematik

Virusausbreitung als mathematische Folge:

- Am Anfang (Zeitpunkt 0) ist ein Mensch infiziert, z. B. mit Corona
- Jeder Infizierte steckt zwei Menschen an (R = 2)
- Folge der Angesteckten *a(t)* zu den Zeitpunkten *t* :

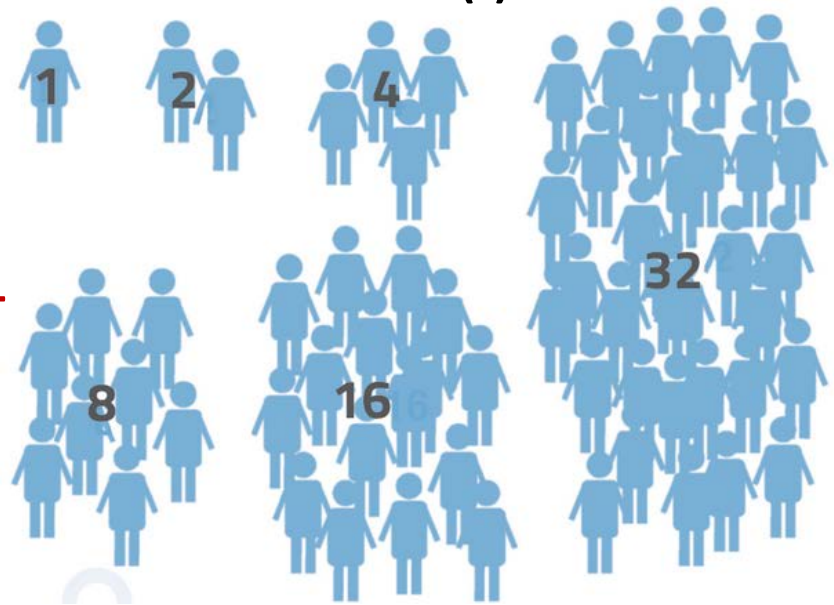
t	0	1	2	3	4	5	6	7
a(t)	1	2	4	8	16	32	64	128

- Definition als normale Funktion ist hier noch einfach: **$a(t) = 2^t$**

- Das geht aber auch rekursiv:

  a(t) = 2 * a(t-1)

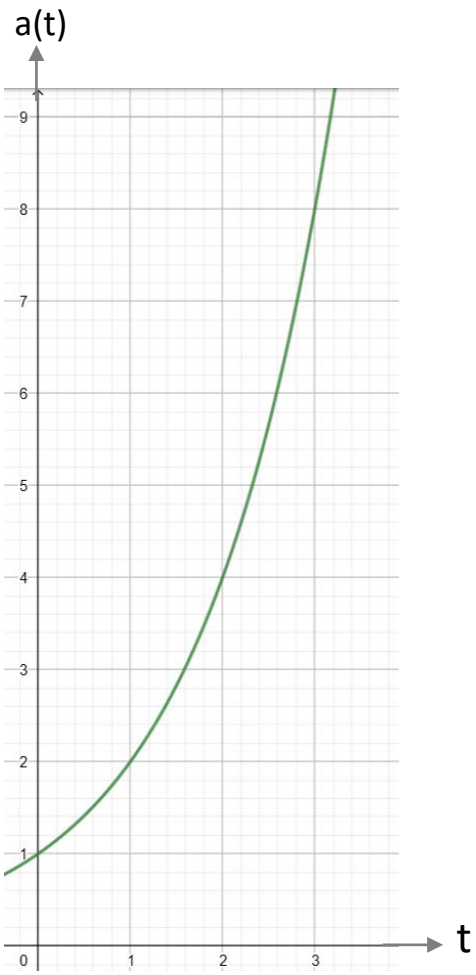
  Abbruchbedingung: a(0) = 1



Rekursion in der Mathematik

Virusausbreitung als mathematische Folge:

t	0	1	2	3	4	5	6	7
a(t)	1	2	4	8	16	32	64	128



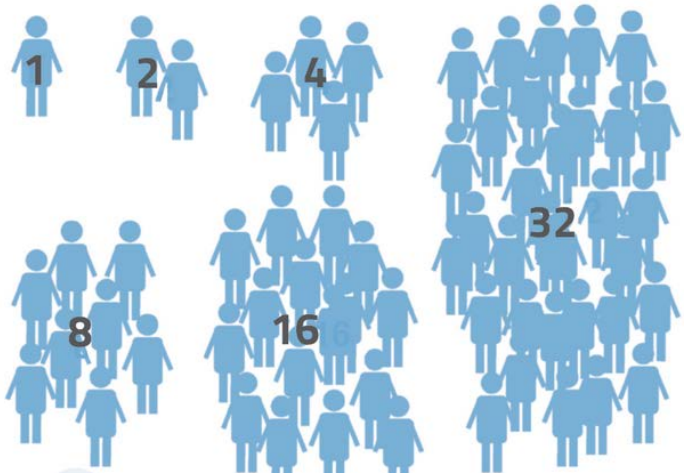
- Definition als normale Funktion **$a(t) = 2^t$**
- Das geht aber auch rekursiv:

  $a(t) = 2 * a(t-1)$

  Abbruchbedingung: $a(0) = 1$

Diese Funktion wächst *exponentiell*,

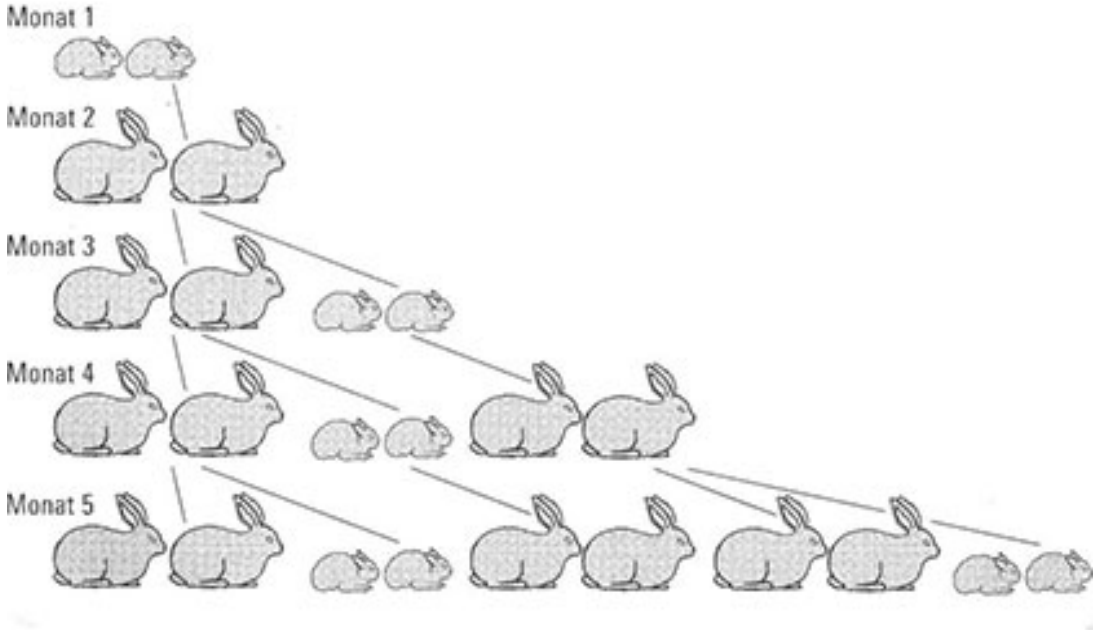
weil die Zeit *t* bei **$a(t) = 2^t$** im Exponenten steht



Rekursion in der Mathematik

Fibonacci-Folge:

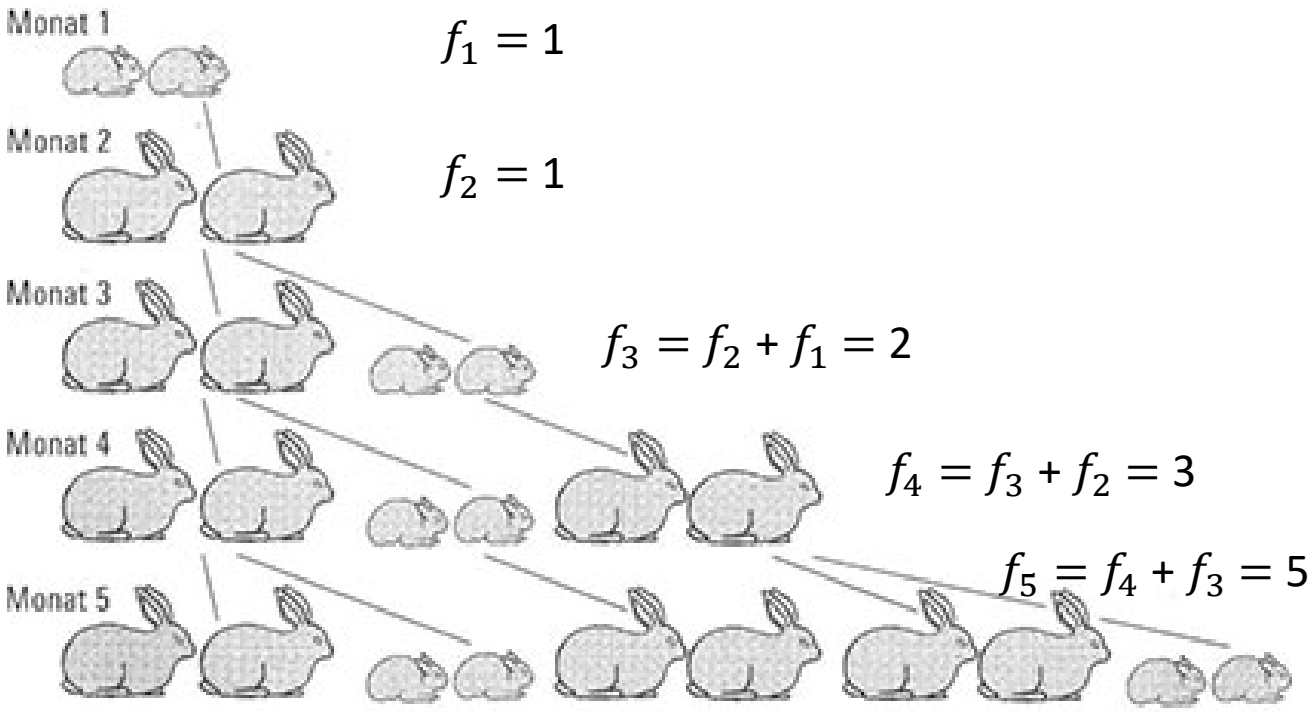
- Benannt nach Leonardo Fibonacci, der 1202 das Wachstum einer **Kaninchenpopulation** beschrieb
- Wie viele Kaninchenpaare gibt es nach n Monaten?
- Neugeborene Kaninchen brauchen 2 Monate, um geschlechtsreif zu werden und bekommen dann pro Paar pro Monat zwei Kinder



Rekursion in der Mathematik

Fibonacci-Folge:

- f_n ist die Anzahl der Paare nach n Monaten
- $f_n = f_{n-1} + f_{n-2}$
- mit Abbruchbedingung $f_1 = f_2 = 1$.



n	f_n
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55

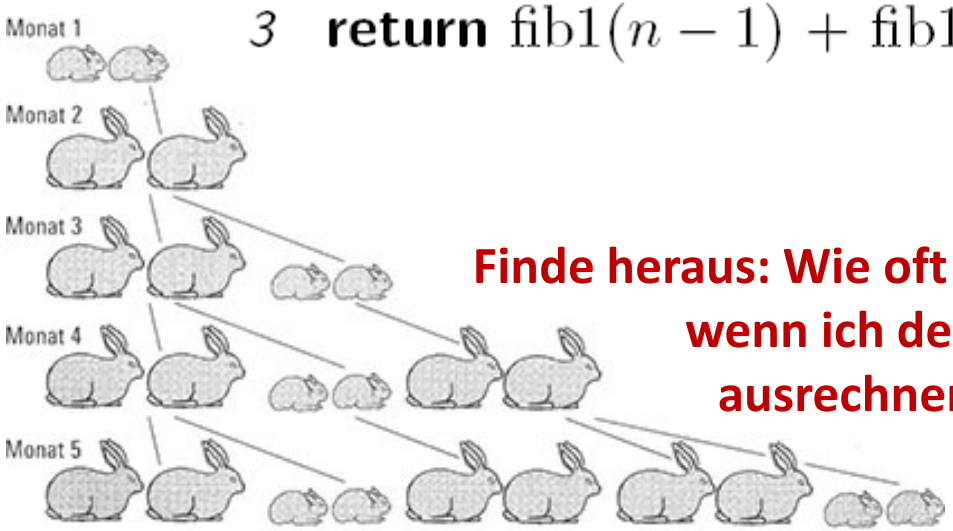
Rekursion in der Informatik

Fibonacci-Folge:

- f_n ist die Anzahl der Paare nach n Monaten
- $f_n = f_{n-1} + f_{n-2}$
- mit Abbruchbedingung $f_1 = f_2 = 1$.

n	fn	Calls
1	1	1
2	1	3
3	2	5
4	3	9
5	5	15
6	8	25
7	13	41
8	21	67
9	34	109
10	55	

```
function fib1(n)
    if n = 0 return 0
    if n = 1 return 1
    return fib1(n − 1) + fib1(n − 2)
```



Finde heraus: Wie oft wird fib1 aufgerufen, wenn ich den Wert f_{10} ausrechnen möchte?

Aufgabe: Rekursion in Javascript

Pseudocode der Fibonacci Folge:

```
function fib1(n)
  if n = 0 return 0
  if n = 1 return 1
  return fib1(n − 1) + fib1(n − 2)
```

- Entpacke *Code_Rekursion.zip* auf dem Desktop
- Öffne im Ordner *scripts* die Javascript-Datei *fibonacci.js* mit einem Texteditor wie Notepad+
- Programmiere die rekursive Fibonacci-Funktion *fibonacci_rec(n).*
- Erhöhe eine Variable *calls* bei jedem Aufruf der Funktion *fibonacci_rec* um 1 und zähle auf diese Weise die Aufrufe.
- Speichere dein Programm als *fibonacci.js* ab
- Teste, ob dein Programm wie vorgesehen funktioniert, indem du die Webseite *fibonacci.html* im Browser öffnest und die berechneten Werte überprüfst. Der Code dieser Webseite verwendet deine Funktion!
- Stelle durch die Plot -Funktion auf der *fibonacci.html* – Seite im Browser den Verlauf der rekursiven Funktion graphisch dar, um zu sehen wie schnell die Werte (Zeit und Calls) wachsen.

n	fn	Calls
1	1	1
2	1	3
3	2	5
4	3	9
5	5	15
6	8	25
7	13	41
8	21	67
9	34	109
10	55	

- Beantworte die Frage: **Wie oft wird fibonacci_rec aufgerufen, wenn ich den Wert f_{20} berechne?**

Bildnachweis

- https://uconn-today-universityofconn.netdna-ssl.com/wp-content/uploads/2018/03/Draw-the-Koch-Snowflake.jpg
- https://orderinchoas.files.wordpress.com/2013/05/sierpinskytrianglewikipedia.gif
- http://www.sfu.ca/~shaw/iat265/MultimediaProgramming/img/recursion/recursion1.jpg
- https://i.ebayimg.com/images/i/131375400699-0-1/s-l1000.jpg
- http://static.nautil.us/483_e1e32e235eee1f970470a3a6658dfdd5.jpg
- https://plantcaretoday.com/echeveria-plant-care.html
- https://skullsinthestars.files.wordpress.com/2011/07/nickieatmirror.jpg?w=640
- http://www.rendtel.de/Unterricht/Chaos/Pythagoras-Baum/pbaum.html