Algorithmen und Komplexität
Vorlesung 2

Wolfgang Globke

universität wien

DHBW Duale Hochschule Baden-Württemberg

4. April 2019

Zwischenspiel: Vollständige Induktion

Es sein $P(n)$ eine Aussage (Prädikat), die von einer natürlichen Zahl $n \in \mathbb{N}_0$ abhängt.

Angenommen, wir wollen beweisen, dass für alle $n \in \mathbb{N}$ gilt:

$$P(n): \quad 1 + 2 + \ldots + n = \frac{n(n+1)}{2}.$$

Vollständige Induktion erlaubt uns, diese und ähnliche Aussagen zu beweisen.

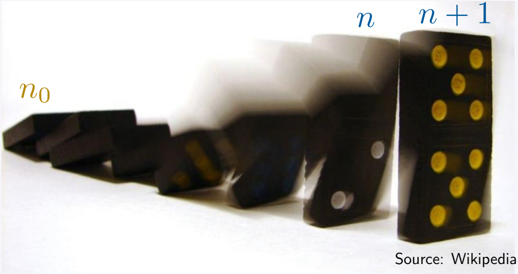
Das Induktionsprinzip

Es sein $P(n)$ eine Aussage (Prädikat), die von einer natürlichen Zahl $n \in \mathbb{N}_0$ abhängt.

Falls folgendes gilt,

1. $P(n_0)$ ist wahr für ein gewisses $n_0 \in \mathbb{N}_0$,
2. $P(n)$ impliziert $P(n+1)$ für alle $n \geq n_0$,

so gilt $P(n)$ für alle $n \geq n_0$.



In drei Schritten beweisen wir $P(n)$ für alle $n \geq n_0$:

Induktionsanfang (I.A.):
Beweise die Aussage $P(n_0)$ für den Anfangswert n_0.

Induktionsvoraussetzung (I.V.):
Für ein gewisses $n \geq n_0$ gilt $P(k)$ für alle $n \geq k \geq n_0$.

Induktionsschluss (I.S.):
Zeige, unter der Induktionsvoraussetzung, dass $P(n+1)$ gilt.

Dann folgt aus dem Induktionsprinzip, dass $P(n)$ für alle $n \geq n_0$ gilt.

Beispiel: Gaußsche Summenformel

Für alle $n \in \mathbb{N}$ gilt:

$$1 + 2 + \ldots + n = \frac{n(n+1)}{2}.$$

Beweis durch Induktion:

- Induktionsanfang $n_0 = 1$:
  $1 = \frac{1 \cdot (1+1)}{2} = \frac{2}{2}$.
- Induktionsvoraussetzung:
  Für ein gewisses $n \geq 1$ gilt $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.
- Induktionsschluss $n \rightsquigarrow n + 1$:
  Betrachte $1 + 2 + \ldots + n + (n + 1)$. Mit der Induktionsvoraussetzung gilt:

$$\begin{aligned} 1 + 2 + \ldots + n + (n+1) &\overset{\text{I.V.}}{=} \frac{n(n+1)}{2} + (n+1) \\ &= \frac{n(n+1)}{2} + \frac{2(n+1)}{2} \\ &= \frac{(n+2)(n+1)}{2}. \end{aligned}$$

  Also gilt die Formel auch für $n + 1$.
- Nach dem Induktionsprinzip gilt die Formel für alle $n \geq n_0 = 1$. $\square$

Satz

Jede ganze Zahl $n > 1$ ist ein Produkt von Primzahlen.

Beweis durch Induktion:

- Induktionsanfang $n_0 = 2$:
  2 ist selbst eine Primzahl.
- Induktionsvoraussetzung:
  Für ein gewisses n sind alle Zahlen $2 \leq k \leq n$ Produkte von Primzahlen.
- Induktionsschritt $n \rightsquigarrow n + 1$:
  Falls $n + 1$ selbst eine Primzahl ist, so sind wir fertig.
  Andernfalls gilt $n + 1 = ab$ für ganze Zahlen $a, b \geq 2$.
  Außerdem sind a, b beide kleiner als $n + 1$.
  Nach Induktionsvoraussetzung sind $a = p_1 \cdots p_m$ und $b = q_1 \cdots q_k$ beide Produkte von Primzahlen.
  Also ist auch $n + 1 = ab = p_1 \cdots p_m \cdot q_1 \cdots q_k$ ein Produkt von Primzahlen. $\square$

(Zusatz: Die Zerlegung ist sogar eindeutig (Satz von Euklid).)

Achtung!

Der Induktionsanfang und der Induktionsschritt müssen immer bewiesen werden!
Andernfalls können schreckliche Dinge passieren!

Wir wollen beweisen, dass $5^n > 5^{n+1}$ gilt für alle $n \geq 1$.

„Beweis":

- Induktionsvoraussetzung:
  Die Behauptung gilt für ein gewisses $\boldsymbol{n}$.
- Induktionsschritt $\boldsymbol{n \rightsquigarrow n+1}$:
  Benutze die Induktionsvoraussetzung:

$$5^{n+1} = 5 \cdot 5^n > 5 \cdot 5^{n+1} = 5^{n+2}.$$

  Also folgt die Behauptung für $\boldsymbol{n+1}$ aus der Behauptung für $\boldsymbol{n}$.

Der Beweis ist Unsinn, da wir den Induktionsanfang nicht bewiesen haben!

Die Behauptung ist

$$n^2 + n + 41 \text{ ist eine Primzahl für alle natürlichen Zahlen } n.$$

Prüfe Anfangswerte:

n	$n^2 + n + 41$
0	41
1	43
2	47
3	53
4	61
5	71
$\vdots$	
39	1601

alles Primzahlen

Allerdings, für $n = 40$:

$$40^2 + 40 + 41 = 41^2,$$

was natürlich keine Primzahl ist.

Zwischenspiel: Rekursion

Eine Folge ist eine Funktion f, die auf den natürlichen Zahlen $\mathbb{N}_0$ (oder einer Teilmenge davon) definiert ist.

Oft schreiben wir f_n anstelle von $f(n)$ für den Funktionswert an der Stelle n.

Eine Folge f ist rekursiv definiert, falls

1. sie für gewisse Anfangswerte definiert ist,
2. der Wert f_n durch (einige) vorhergehende Werte $f_{n-1}, f_{n-2}, \ldots$ definiert ist.

Eine Rekurrenz-Relation ist eine Gleichung, die f_n durch $f_{n-1}, f_{n-2}, \ldots$ ausdrückt.

Beispiel: Fibonacci-Zahlen

Die berühmten Fibonacci-Zahlen sind rekursiv definiert durch

1. zwei Anfangswerte

$$f_0 = 0, \quad f_1 = 1$$

2. und die Rekurrenz-Relation

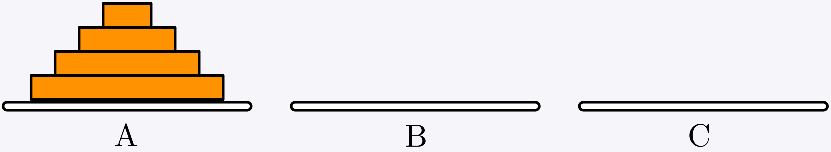
$$f_n = f_{n-1} + f_{n-2} \quad \text{für } n \geq 2.$$

int FIBO(n)
  if $n = 0$ **then return** 0
  if $n = 1$ **then return** 1
  else return FIBO($n-1$) $+$ FIBO($n-2$)

Rekursive Programmierung folgt dem Prinzip rekursiv definierter Folgen.

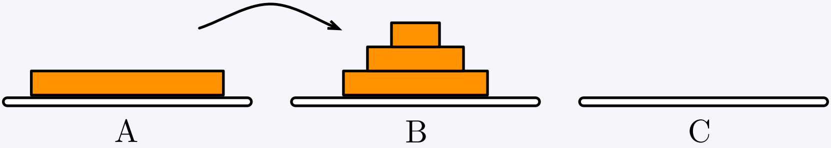
- Durch die Verwandtschaft zum Induktionsprinzip sind rekursive Algorithmen der mathematischen Analyse ihrer Korrektheit bzw. Terminierung besonders gut zugänglich.
- Aufwandsanalyse ist in der Regel komplizierter.
- Nachteil: Wiederholte Funktionsaufrufe verbrauchen Prozessorzyklen und Speicher.
- Lisp lernen!

Wir haben drei Stellplätze A, B, C, auf denen wir Scheiben stapeln können.
Auf Platz A liegt ein Stapel von n Scheiben, wobei jede Scheibe kleiner als die jeweils darunterliegende ist.



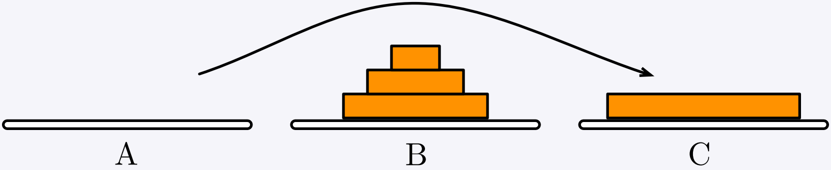
Aufgabe:
Bewege all Scheiben von A nach C (und benutze B als Puffer) wobei immer nur eine Scheibe auf eine größere Scheibe oder auf einen leeren Platz bewegt werden darf.

Lösung:
Angenommen wir wissen bereits, wie wir einen Stapel von $n-1$ Scheiben von einem Platz auf einen andere bewegen können.

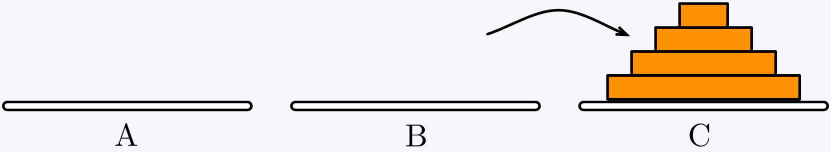


Dann:
Bewege die obere $n-1$ Scheiben auf Platz B (wobei wir C als Puffer benutzen), nur Scheibe n verbleibt auf Platz A.

Dann:
Bewege Scheibe n auf Platz C.



Da wir (nach Annahme) bereits wissen, wie wir $n-1$ Scheiben bewegen können, bewegen wir die $n-1$ Scheiben von Platz B auf Platz C (benutze A als Puffer).



Schummeln? Hexerei?
Nein, nur die Macht der Rekursion!

Code: Türme von Hanoi

Annahme:
Eine (korrekte und terminierende) Prozedur MOVE(A,B) ist gegeben, welche die oberste Scheibe von Platz A auf Platz C bewegt.

HANOI(n, A, B, C)
    if $n = 1$ **then** MOVE(A, C)
    else
        HANOI($n - 1$, A, C, B)
        MOVE(A, C)
        HANOI($n - 1$, B, A, C)

Wir wollen Korrektheit und Terminierung diese Algorithmus beweisen.

Terminierung: Türme von Hanoi

```
HANOI(n, A, B, C)
    if n = 1 then MOVE(A, C)
    else
        HANOI(n − 1, A, C, B)
        MOVE(A, C)
        HANOI(n − 1, B, A, C)
```

Behauptung

Der Algorithmus HANOI *terminiert für jede Eingabe* n*, A, B, C nach endlich vielen Schritten.*

Beweis durch Induktion über n:

- Induktionsanfang $n = 1$:

  HANOI terminiert sofort, da (nach Annahme) MOVE(A, C) terminiert.
  Induktionsvoraussetzung:

  HANOI terminiert für ein gewisses $n \geq 1$ und beliebige A, B, C.

- Induktionsschritt $n \rightsquigarrow n + 1$:

  Beim Aufruf von HANOI($n + 1$, A, B, C) gelangen wir in den **else** Block (wobei $n - 1$ durch n ersetzt wird).

  - HANOI(n, A, C, B) terminiert nach Induktionsvoraussetzung.
  - MOVE(A, C) terminiert, da wir dies über MOVE vorausgesetzt haben.
  - HANOI(n, B, A, C) terminiert auch nach Induktionsvoraussetzung.

  Danach folgen keine weiteren Anweisung, womit der Algorithmus terminiert. $\square$

Korrektheit: Türme von Hanoi

```
HANOI(n, A, B, C)
    if n = 1 then MOVE(A, C)
    else
        HANOI(n − 1, A, C, B)
        MOVE(A, C)
        HANOI(n − 1, B, A, C)
```

Behauptung

Der Algorithmus HANOI *ist für jede Eingabe* n, *A, B, C korrekt (das bedeutet, er bewegt alle Scheiben von A nach C).*

Beweis durch Induktion über n:

- Induktionsanfang $n = 1$:
  HANOI endet mit dem Aufruf von MOVE(A, C), das als korrekt vorausgesetzt wurde.
- Induktionsvoraussetzung:
  Für ein gewisses $n \geq 1$ und beliebige A, B, C ist HANOI korrekt im obigen Sinne.
- Induktionsschritt $n \rightsquigarrow n + 1$:
  Beim Aufruf von HANOI($n + 1$, A, B, C) gelangen wir in den **else** Block (wobei $n - 1$ durch n ersetzt wird).
  - HANOI(n, A, C, B) bewegt die oberen n Scheiben von A nach B, nach Induktionsvoraussetzung.
  - MOVE(A, C) bewegt die verbliebene Scheibe von A nach C, nach Annahme über MOVE.
  - HANOI(n, B, A, C) bewegt die oberen n Scheiben von B nach C, nach Induktionsvoraussetzung.

  Somit liegen nun alle Scheiben auf Platz C. □

Alle Pferde haben die gleiche Farbe.

Beweis durch Induktion:

- Induktionsanfang $n = 1$:
  Ein einzelnes Pferd hat offensichtliche die gleiche Farbe wie es selbst.
- Induktionsvoraussetzung:
  Für ein $n \geq 1$ gelte, dass in jeder n-elementigen Mengen von Pferden alle Pferde die gleiche Farbe haben.
- Induktionsschritt $n \rightsquigarrow n + 1$:
  Betrachte eine Menge $\{P_1, \ldots, P_{n+1}\}$ von $n + 1$ Pferden.
- Dann sind $\{P_1, \ldots, P_n\}$ und $\{P_2, \ldots, P_{n+1}\}$ zwei n-elementige Mengen von Pferden, so dass in jeder dieser beiden Mengen alle Pferde die gleiche Farbe haben.
- Wegen des Überlapps $\{P_2, \ldots, P_n\}$ dieser zwei Mengen müssen aber alle Pferde in der Vereinigung dieser beiden n-elementigen Teilmengen die gleiche Farbe haben.
- Dies sind aber alle $n + 1$ Pferde $P_1, \ldots, P_{n+1}$. $\square$