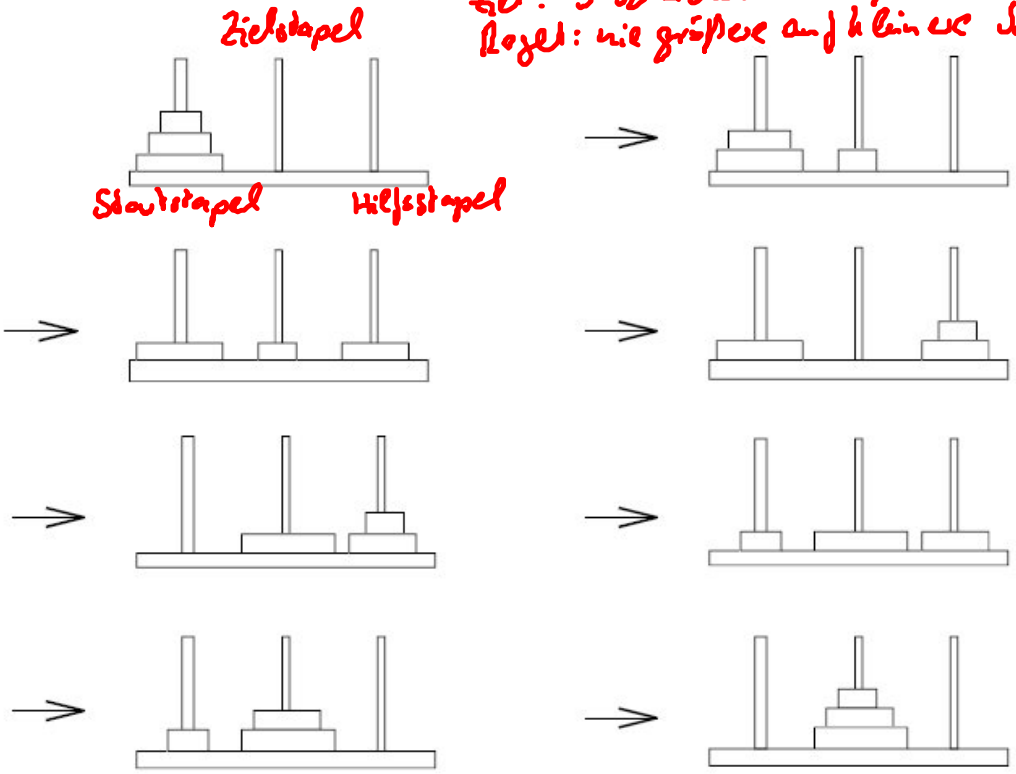
VL 09.01

→ PA 09 Aufgabenstellung ggf. Unklarheiten gelöst
in Version vom 08.01. war noch eine Assertion enthalten
diese soll nicht implementiert werden

Sorry!

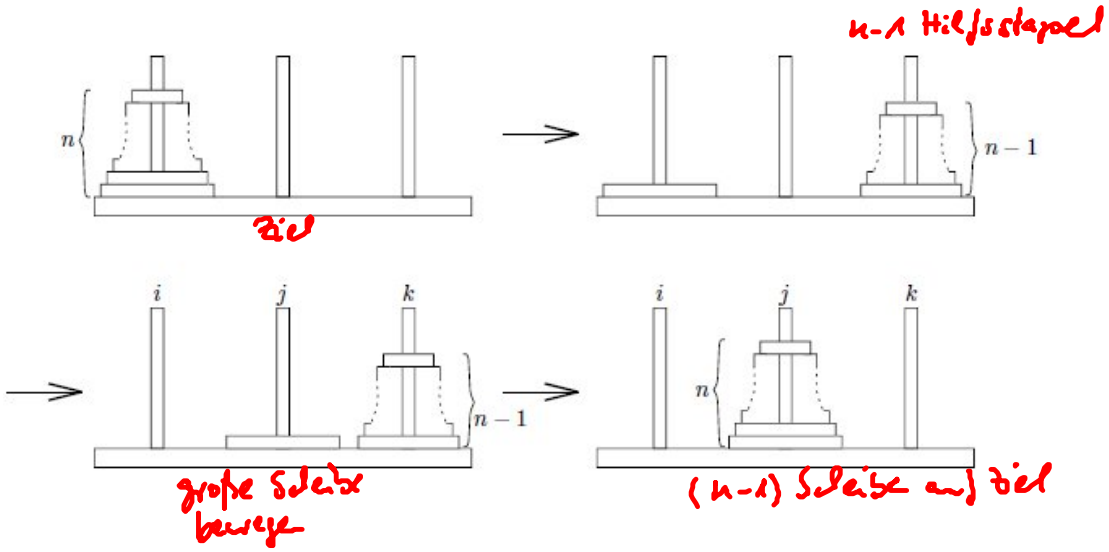
Türme von Hanoi

Ziel: 3 Scheiben umstapeln
Regel: nie größere auf kleinere Scheibe



Suche kürzeste Zugfolge zum Umstapeln von n Scheiben
iterative Lösung zu finden ist eher schwer
rekursive Lösung ist einfach

Grundidee zur Rekursion



Die Scheiben seien $n, n-1, \ldots, 2, 1$

n: größte 1: kleinste

1. Bewege Scheiben $n-1, (n-2), \ldots, 1$ auf Hilfsstapel und gib Züge dafür an — Delegation
2. Bewege größte Scheibe (n) auf Zielstapel und gib den Zug aus
3. Bewege Scheiben $(n-1), (n-2), \ldots, 1$ auf den Zielstapel — gib die Züge dafür an — Delegation

```
String getMoves(int numberOfDisks, int origin, int destination, int auxPile){
        if (numberOfDisks <= 0) return ""; // no moves to return

        StringBuffer moves = new StringBuffer();
        // move the numberOfDisks - 1 smallest disks
        // from origin to auxPile
        // with destination as auxiliary pile
1.      moves.append(getMoves(numberOfDisks - 1, origin,
                              auxPile, destination));

        // append the move of the largest disk to StringBuffer moves
2.      moves.append(origin + " --> " + destination + "\n");

        // move the numberOfDisks - 1 smallest disks from auxPile
        // to destination with origin as auxiliary file
        // (the largest disk on destination does not interfere)
3.      moves.append(getMoves(numberOfDisks - 1, auxPile,
                              destination, origin));

        return moves.toString();
}
```

Frage: Warum funktioniert das Programm korrekt?
(obwohl die Regel „keine große auf kleine Scheibe"

irgendwo anfangen)

Induktion über die # von Scheiben

I.A. $n = 1$ ✓

I. Voraus. Programm funktioniert korrekt für $n-1 \geq 1$ Scheibe

Schluss auf n Scheiben

1) Also bewegt zunächst $(n-1)$ Scheiben zum Hilfsstapel
   → das macht es nach Ind. Voraus. korrekt
2) Also bewegt die größte Scheibe auf Zielstapel (dieser ist leer)
3) Also bewegt $(n-1)$ Scheiben vom Hilfsstapel zum Zielstapel
   → das macht es nach Ind. Voraus. korrekt

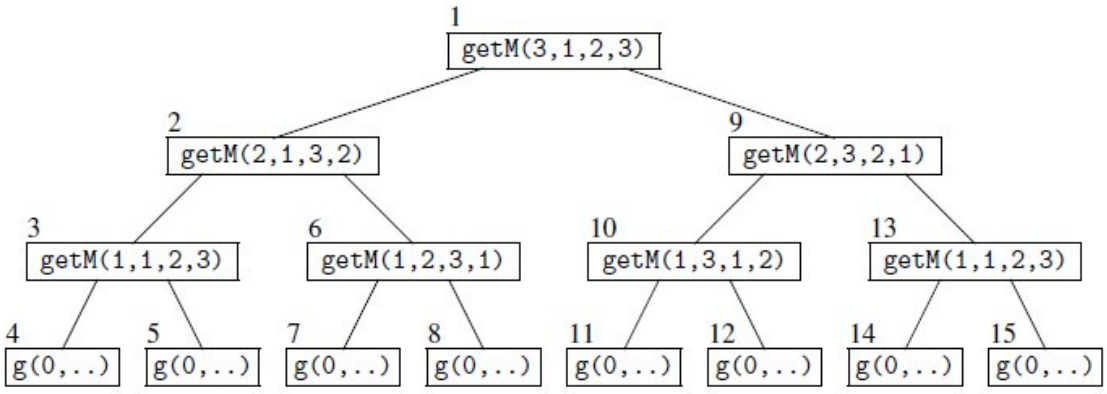


Abbildung 7.8: Rekursionsbaum zu den Türmen von Hanoi. Aus Platzgründen ist `getMoves(...)` mit `getM(...)` oder `g(...)` abgekürzt.

Blätter Abbruch beim Zug

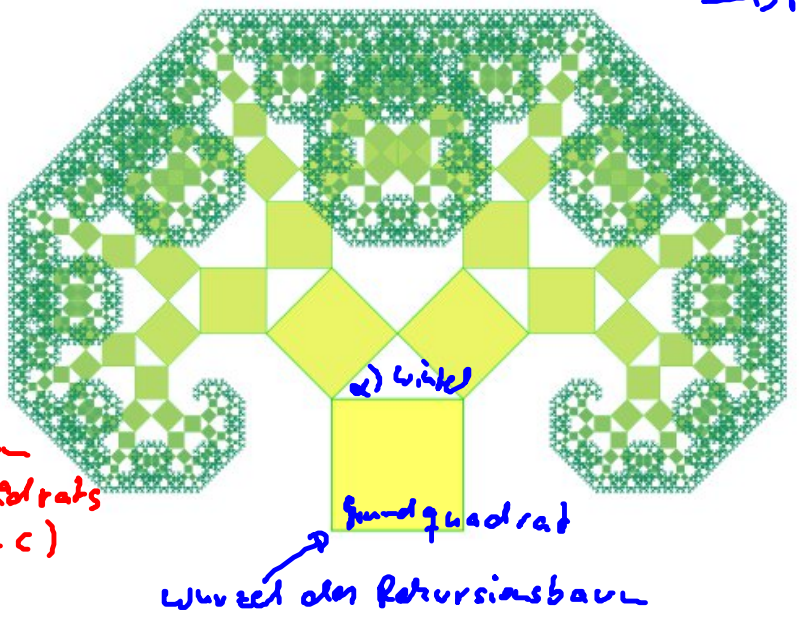
L R W Durchlauf
postorder traversal

Pythagoras-Bäume

(Aufbau)

Bsp. für rek. Programm, das seinen eigenen Rekursionsbaum zeichnet

→ Programmvorführung



Abbruch Rekursion bei Seitenlänge Quadrats $< c$ (Konstante c)

Wurzel des Rekursionsbaum

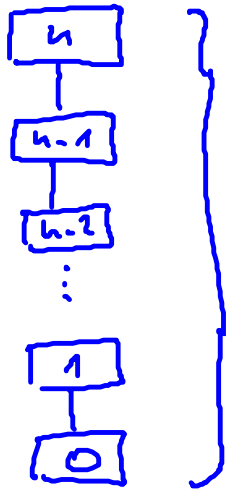
Analyse der Rekursion, Nutzen und Grenze

↑ bestimmt durch Rekursionstiefe
Anzahl rekursive Aufrufe

1. Fakultät

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

Rekursionsbaum ist Liste
in solchen Fällen existiert eine
iterative Lösung, die zu bevorzugen
ist



Rekursionstiefe
$n+1$
großes ok

2. ggT (x, y)

$x \geq y$

$$ggT(x,y) = \begin{cases} y & \text{falls } x \bmod y = 0 \\ ggT(y, x \bmod y) & \text{falls } y > 0 \end{cases}$$

Rekursionsbaum ist Liste (siehe VL gestern)
⇒ Iterative Lösung (siehe VL von Anfang Nov.)
aber Rekursionstiefe ist klein, Rekursion gut verwendbar

Satz: Für die Rekursionstiefe $R_{ggT}(x,y)$ von ggT gilt

$$R_{ggT}(a,b) \leq 1 + \log(\max\{x,y\})$$

(logarithmisch in die größere der beiden Zahlen)

Beweis: Sei $x > y$, $t := x \operatorname{div} y$, $r := x \bmod y$

$$\Rightarrow x = t \cdot y + r \underset{\uparrow}{\geq} y + r \underset{\uparrow}{\geq} 2r \Rightarrow r < \frac{x}{2}$$

$t \geq 1$ $\quad$ $y > r$

Aufrufe $(x,y) \to (y,r) \to (r, <r)$

⇒ nach 2 Aufrufen hat sich die größere der beiden Zahlen mehr als halbiert

⇒ nach $2 \cdot \log(\max\{x,y\}) + 1$ vielen Schritten ist man bei fortgesetzter Halbierung bei ≤ 1

⇒ vorher Abbruch mit der kleinen Zahl ist Teiler der größeren Zahl □

Türme von Hanoi

Satz: Rekursionstiefe $R(n) = n+1$
rekursive Aufrufe $T(n) = 2^{n+1} - 1$
Züge $z(n) = 2^n - 1$
} bei n Scheiben

Beweis durch Induktion nach # Scheiben

für $z(n)$

Ind. Anfang: $n=1$ $\quad z(1) = 1 \quad 2^1 - 1 = 1$ ✓

Ind. Vorauss: Bel sei richtig für $k < n$ Stellen ($k = 1, 2, \ldots, n-1$)

Schluss auf n Stellen:

$$\begin{aligned} z(n) &= z(n-1) + 1 + z(n-1) \\ &= 2 \cdot z(n-1) + 1 \\ &\overset{I.V.}{=} 2 \cdot (2^{n-1} - 1) + 1 \\ &= 2^n - 1 \end{aligned}$$

□

andere Beweise analog

Rekursionstiefe okay

rek. Aufrufe exponentiell, umsetzbar, aber Anzahl Züge exponentiell

außerdem gibt es keine naheliegende iterative Lösung

⇒ Rekursion insgesamt in Ordnung

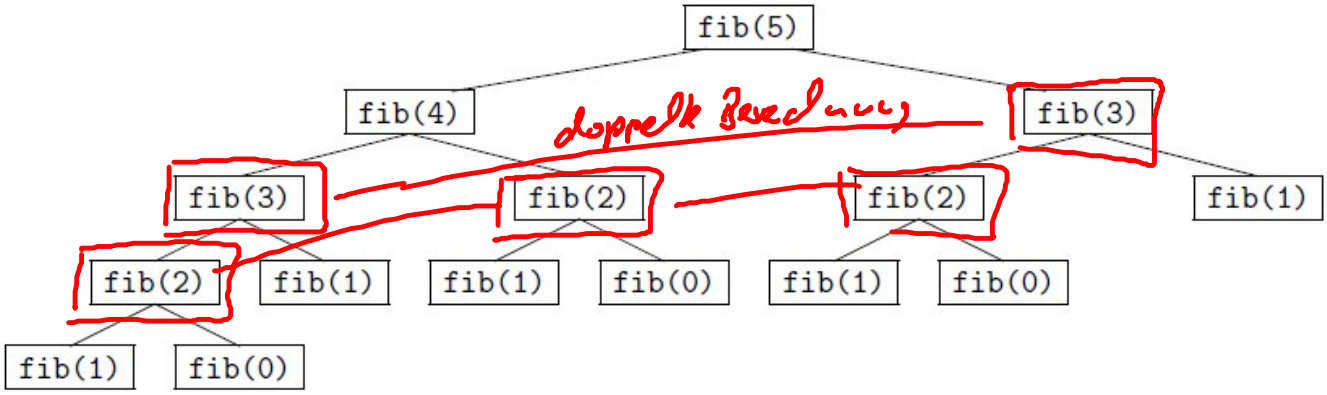
Fibonacci Folge

$f_0 = 0 \quad f_1 = 1$ $\qquad$ 0 1 1 2 3 5 8 ...

$f_{n+1} = f_n + f_{n-1}$

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

Rekursionsbaum



⇒ viele Mehrfachberechnungen

⇒ Rekursion braucht lange

```
int fib(int n) {
        // assume n >= 0
        if (n == 0) return 0;
        if (n == 1) return 1;
        int currentFib = 1, prevFib = 0;
        for ( int i = 1; i < n; i++ ){
                currentFib = currentFib + prevFib;
                prevFib    = currentFib - prevFib;
        }
        return currentFib;
}
```

→ Programmverfahren zu den Laufzeitunterschieden von rekursiver und iterativer Implementierung

Satz: Für die # der rekursiven Aufrufe $T_{Fib}(n)$ gilt $T_{Fib}(n) \geq 2^{\lfloor n/2 \rfloor}$

Exponentielles Wachstum

Beweis: durch Induktion über n

$n=0$ $\quad T(0) = 1 \quad 2^{\lfloor 0/2 \rfloor} = 1$ ✓

$n=1$ $\quad T(1) = 1 \quad 2^{\lfloor 1/2 \rfloor} = 1$ ✓

Sei Beh. richtig für $0 \leq k \leq n$

Schluss auf $n+1$

$$T(n+1) = 1 + T(n) + T(n-1) \geq 2 \cdot T(n-1)$$

$$\overset{IV}{\geq} 2 \cdot 2^{\lfloor \frac{n-1}{2} \rfloor} = 2^{\lfloor \frac{n+1}{2} \rfloor}$$

□

Extrembeispiele bei Speicherplatz und Terminierung

Ackermann Funktion

$$a(m,n) = \begin{cases} n+1 & , m=0 \\ a(m-1,1) & , m>0, n=0 \\ a(m-1, a(m,n-1)) & , m>0, n>0 \end{cases} \qquad m,n \in \mathbb{N} \cup \{0\}$$

Folge immer tiefer verschachtelter Funktionen

$a(0, n) = n + 1 > n$

$a(1, n) > n + 1$

$a(2, n) > 2n$

$a(3, n) > 2^n$

$a(4, n) > \left.2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\right] n$

$a(5, 4) > 10^{10\,000}$

nächste Woche: Fällt Seminar immer