



Algorithmentheorie

14 – Dynamische Programmierung (1)

Prof. Dr. S. Albers

- Allgemeine Vorgehensweise, Unterschiede zur rekursiven Lösung
- Ein einfaches Beispiel: Die Berechnung der Fibonacci Zahlen

Rekursiver Ansatz: Lösen eines Problems durch Lösen mehrerer kleinerer Teilprobleme, aus denen sich die Lösung für das Ausgangsproblem zusammensetzt.

Phänomen: Mehrfachberechnungen von Lösungen

Methode: Speichern einmal berechneter Lösungen in einer Tabelle für spätere Zugriffe.

Beispiel: Fibonacci-Zahlen

$$f(0) = 0$$

$$f(1) = 1$$

$$f(n) = f(n-1) + f(n-2), \text{ falls } n \geq 2$$

Bem: Es gilt:

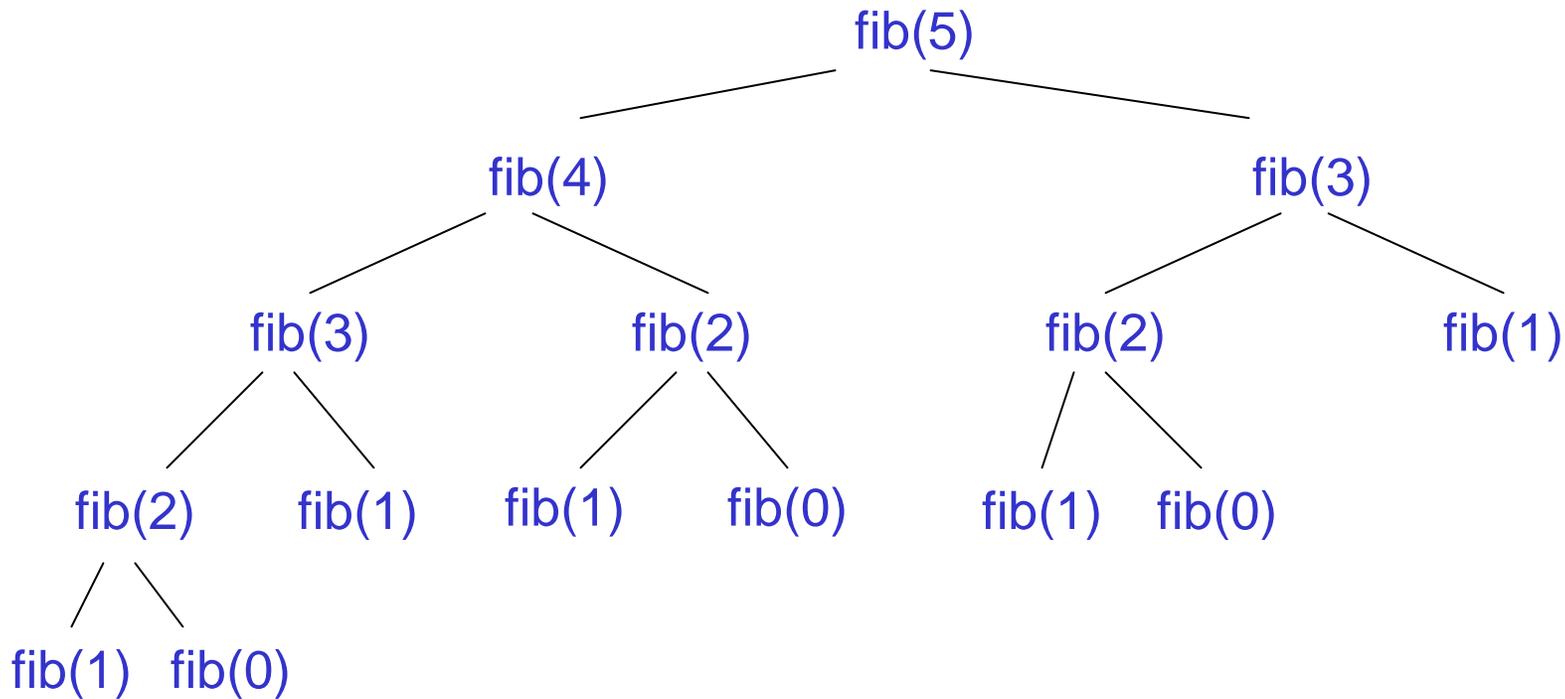
$$f(n) = \left[\frac{1}{\sqrt{5}} (1.618\dots)^n \right]$$

Direkte Implementierung:

```
procedure fib (n : integer) : integer  
if (n == 0) or (n == 1)  
  then return n  
  else return fib(n - 1) + fib(n - 2)
```

Beispiel: Fibonacci-Zahlen

Aufrufbaum:



Mehrfachberechnung!

$$T(n) \approx \left[\left(1 + \frac{1}{\sqrt{5}} \right) \left(\frac{\sqrt{5} + 1}{2} \right)^n - 1 \right] \approx [1.447 \times 1.618^n - 1]$$

Dynamisches Programmieren

Vorgehen:

1. Rekursive Beschreibung des Problems P
2. Bestimmung einer Menge T , die alle Teilprobleme von P enthält, auf die bei der Lösung von P – auch in tieferen Rekursionsstufen – zurückgegriffen wird.
3. Bestimmung einer Reihenfolge T_0, \dots, T_k der Probleme in T , so dass bei der Lösung von T_i nur auf Probleme T_j mit $j < i$ zurückgegriffen wird.
4. Sukzessive Berechnung und Speicherung von Lösungen für T_0, \dots, T_k .

Beispiel Fibonacci-Zahlen

1. Rekursive Definition der Fibonacci-Zahlen nach gegebener Gleichung.
2. $T = \{ f(0), \dots, f(n-1) \}$
3. $T_i = f(i), \quad i = 0, \dots, n - 1$
4. Berechnung von $fib(i)$ benötigt von den früheren Problemen nur die zwei letzten Teillösungen $fib(i - 1)$ und $fib(i - 2)$ für $i \geq 2$.

Beispiel Fibonacci-Zahlen

Berechnung mittels dynamischer Programmierung, Variante 1:

procedure *fib*(*n* : *integer*) : *integer*

1 $f_0 := 0; f_1 := 1$

2 **for** $k := 2$ **to** n **do**

3 $f_k := f_{k-1} + f_{k-2}$

4 **return** f_n

Beispiel Fibonacci-Zahlen

Berechnung mittels dynamischer Programmierung, Variante 2:

```
procedure fib (n : integer) : integer  
1   $f_{\text{vorletzte}} := 0; f_{\text{letzte}} := 1$   
2  for  $k := 2$  to  $n$  do  
3       $f_{\text{aktuell}} := f_{\text{letzte}} + f_{\text{vorletzte}}$   
4       $f_{\text{vorletzte}} := f_{\text{letzte}}$   
5       $f_{\text{letzte}} := f_{\text{aktuell}}$   
6  if  $n \leq 1$  then return  $n$  else return  $f_{\text{aktuell}}$  ;
```

Lineare Laufzeit, konstanter Platzbedarf!

Memoisierte Version der Berechnung der Fibonacci-Zahlen



Berechne jeden Wert genau einmal, speichere ihn in einem Array $F[0\dots n]$:

```
procedure fib (n : integer) : integer
```

```
1   $F[0] := 0$ ;  $F[1] := 1$ ;
```

```
2  for  $i := 2$  to  $n$  do
```

```
3       $F[i] := \infty$ ;
```

```
4  return lookupfib( $n$ )
```

Die Prozedur *lookupfib* ist dabei wie folgt definiert:

```
procedure lookupfib(k : integer) : integer
```

```
1  if  $F[k] < \infty$ 
```

```
2      then return  $F[k]$ 
```

```
3      else  $F[k] := \text{lookupfib}(k - 1) + \text{lookupfib}(k - 2)$ ;
```

```
4      return  $F[k]$ 
```