Java Beispiel: Berechnung der Fibonacci Zahlen

Informatik I, WS 05/06

Daniel Huson

WSI, Uni Tübingen

Fibonacci Zahlen

- Die Fibonacci Sequenz sieht so aus:

  0,1,1,2,3,5,8,13,21,...

- Hier ist jede Zahl (ab der zweiten Position) gleich der Summe der beiden Vorgängerzahlen
- Genauer, wie definieren:

  Fib(0)=0, Fib(1)=1 und

  Fib(i)=Fib(i-2)+Fib(i-1), für i>1

Rekursive Berechnung in Java, 1

FibonacciRekursiv.java:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
/**
 * Recursive Berechnung der Fibonaccizahlen
 * Daniel Huson
 */
public class FibonacciRekursiv {
     // Rekursive Berechnung der Fibonaccizahl
    static int compute (int i)
    {
        if(i<=0) // fuer negative Zahl auch 0!
           return 0;
        else if(i==1)
            return 1;
        else
            return compute(i-2)+compute(i-1);
    }
...
```

Rekursive Berechnung in Java, 2

FibonacciRekursiv.java, Fortsetung:

```
 // Hauptprogramm: Lese Zahl n ein und gebe Fibonaccizahl aus
    public static void main (String[] args) throws Exception
    {
        // Vorbereitung:
        // Aufforderung zur Eingabe:
        System.out.println("Enter number:");
        // Öffene Eingabestream
        BufferedReader r=new BufferedReader(new InputStreamReader(System.in));
        // Lese erste Zeile der Eingabe
        String aLine=r.readLine();
        // Parse Eingabe als Integer
        int n=Integer.parseInt(aLine);
        // Arbeit:
        int result=compute(n);
        // Ausgabe:
        System.out.println("Fib("+n+")="+result);
    }
}
```

Programmausgabe

```
Enter number:
10
Fib(10)=55
```

Iterative Berechnung,1

FibonacciIterativ.java:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
 // Iterative Berechnung der Fibonaccizahlen
 // Daniel Huson
public class FibonacciIterativ {
     // Iterative Berechnung der Fibonaccizahl
    static int compute (int n)
    {
        if(n<=0) // fuer negative Zahl auch 0!
         return 0;
        else if(n==1)
            return 1;
        else
        {
            int a=0;  // hat am Anfang der Schleife den Wert Fib(i-2)
            int b=1;  // hat am Anfang der Schleife den Wert Fib(i-1)
            int i=2;
            while(i<=n) // Schleife fuer alle Werte von 2 bis n
            {
                int aa=b; // Wert von Fib(i-1)
                int bb=a+b; // Wert von Fib(i)
                a=aa;  // Vorbereitung fuer den naechsten Durchgang
                b=bb;  // Vorbereitung fuer den naechsten Durchgang
                i++;
            }
            return b;
        }
    }
 ...
```

Iterative Berechnung,2

FibonacciIterativ.java, Fortsetung:

```
    // Hauptprogramm: Lese Zahl n ein und gebe Fibonaccizahl aus
       public static void main (String[] args) throws Exception
       {
           // Vorbereitung:
           // Aufforderung zur Eingabe:
           System.out.println("Enter number:");
           // Öffene Eingabestream
           BufferedReader r=new BufferedReader(new InputStreamReader(System.in));
           // Lese erste Zeile der Eingabe
           String aLine=r.readLine();
           // Parse Eingabe als Integer
           int n=Integer.parseInt(aLine);
           // Arbeit:
           int result=compute(n);
           // Ausgabe:
           System.out.println("Fib("+n+")="+result);
       }
   }
```

Programmausgabe

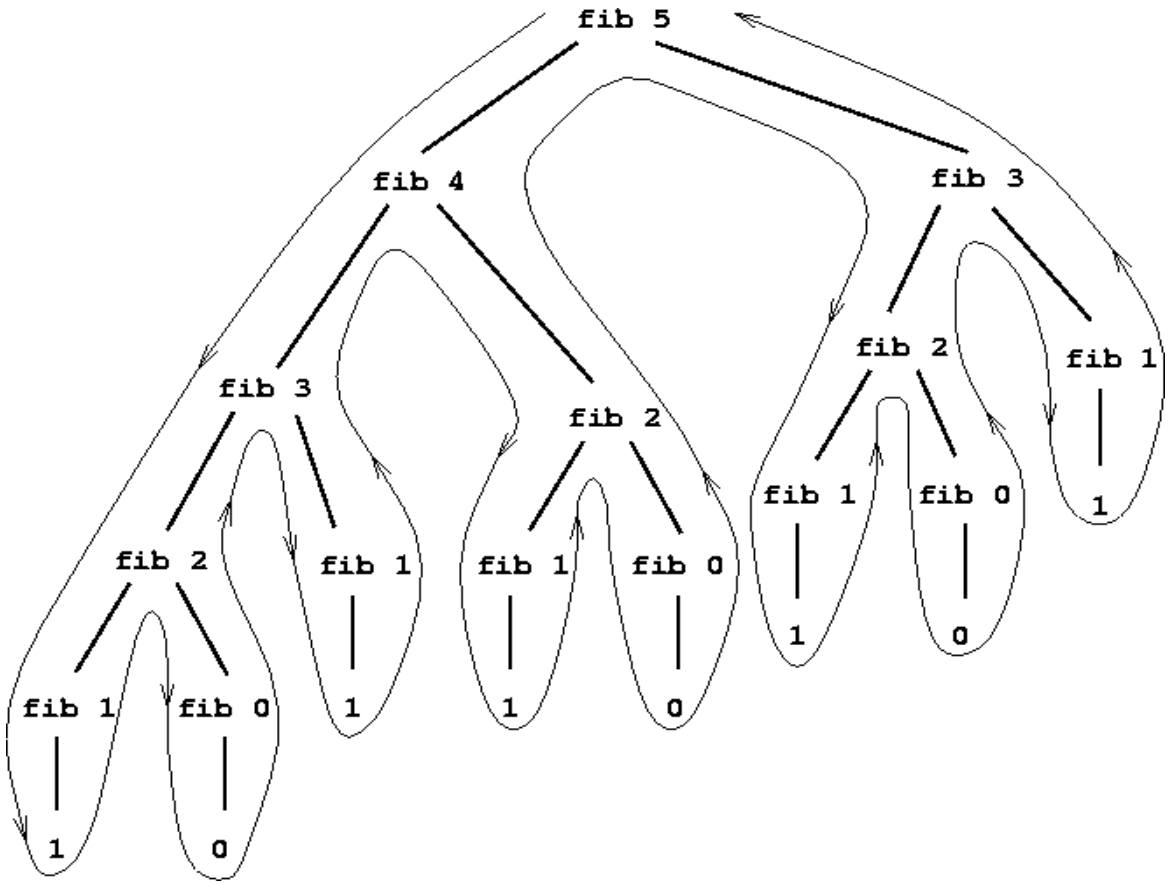
```
Enter number:
10
Fib(10)=55
```

Vergleich

- Das rekursive Programm ist kürzer und übersichtlicher als das iterative Programme
- Gibt es einen Unterschied in der Geschwindigkeit?

  Ja!!!

Anzahl der Aufrufe im rekursiven Programm



Bildquelle: http://mitpress.mit.edu/sicp/full-text/sicp/book/node16.html

Anzahl der Aufrufe von Fib() wächst exponentiell in n.

Anzahl der Aufrufe im iterativen Programm

- Die Funktion Fib() wird nur einmal aufgerufen. Die while-Schleife wird n mal durchlaufen
- D.h., die Laufzeit hängt nur linear von n ab.