Übung zur VL „Grundlagen der Programmierung“

2. Übung

Dr. Zubow

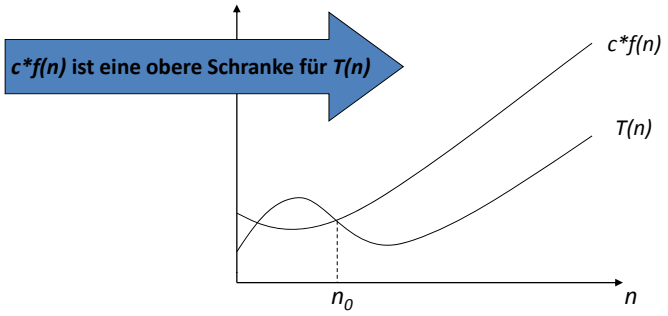
Analyse von Algorithmen

- Big-O Notation:
  - Asymptotisch Analyse
  - Ignoriere Konstanten in T(n)
  - Analysiere T(n) für „große“ *n*
  - Bsp.: $T(n) = 13n^3 + 42n^2 + 2n\log n + 4n$
  - Für große *n* dominiert n^3 den Term T(*n*)
  - Die Laufzeit ist in der Größenordnung von n^3
  - Notation: $T(n) = O(n^3)$

Big-O (*O*) Notation

Big-O Notation

- *T(n) = O(f(n))* wenn Konstanten *c* und *n0* existieren sodass gilt *T(n) < c*f(n)* wenn $n \geq n0$



Rekursion

- Beispiel:
  - Fakultät

    n! = (n-1)! * n,
    wobei 0! = 1! = 1
  - Fibonacci-Zahlen

    fib(n) = fib(n-1) + fib(n-2),
    wobei fib(0) = fib(1) = 1
  - ggT zweier Zahlen n und m

    ggt(m,n) = m, falls n = 0
    = ggt(n, m mod n), sonst

Fibonacci Folge

- Naive Implementation basierend auf der direkten mathematischen Definition:

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

- Bei Aufruf von fib(5) wird folgender Aufrufbaum erzeugt:
  1. fib(5)
  2. fib(4) + fib(3)
  3. (fib(3) + fib(2)) + (fib(2) + fib(1))
  4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
  5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
- Probleme? Lösung?

Dynamische Programmierung - A

- Gegeben sei eine Datenstruktur *map*, welche jede bereits berechnete fib-Zahl sich merkt.
- Die resultierende Funktion hat lediglich eine (Zeit-)Komplexität von O(n):

```
var m := map(0 → 0, 1 → 1)
function fib(n)
    if map m does not contain key n
        m[n] := fib(n − 1) + fib(n − 2)
    return m[n]
```

Dynamische Programmierung - B

- **Bottom-up** Ansatz
  - berechnet zuerst die kleinen fib-Werte um anschließend daraus die großen zu konstruieren
  - Zeitkomplexität (time): O(n)
  - Raumkomplexität (space): O(1), bei Lösung A: O(n)

```
function fib(n)
    var previousFib := 0, currentFib := 1
    if n = 0
        return 0
    else if n = 1
        return 1
    repeat n − 1 times
        var newFib := previousFib + currentFib
        previousFib := currentFib
        currentFib := newFib
    return currentFib
```