Algorithmen und Programmierung 1

Prof. Dr. Stefan Funke WS 2009/10

Fachberich Mathematik und Informatik
Ernst-Moritz-Arndt-Universität Greifswald

Theorie-Übungsblatt 1

Besprechung in 1 Woche

Übersicht O-Notation:

- $O(f(n)) = \{g : \mathbb{N} \to \mathbb{N}; \exists c > 0, n_0 \in \mathbb{N} \text{ sodass } \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$
- $o(f(n)) = \{g : \mathbb{N} \to \mathbb{N}; \forall c > 0\, \exists n_0 \in \mathbb{N} \text{ sodass } \forall n \geq n_0 : g(n) < c \cdot f(n)\}$
- $\Omega(f(n)) = \{g : \mathbb{N} \to \mathbb{N}; \exists c > 0, n_0 \in \mathbb{N} \text{ sodass } \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$
- $\omega(f(n)) = \{g : \mathbb{N} \to \mathbb{N}; \forall c > 0\, \exists n_0 \in \mathbb{N} \text{ sodass } \forall n \geq n_0 : g(n) > c \cdot f(n)\}$
- $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$

Informell:

- $g \in O(f(n))$ heisst: "g wächst asymptotisch höchstens so schnell wie f"
- $g \in o(f(n))$ heisst: "g wächst asymptotisch echt langsamer als f"
- $g \in \Omega(f(n))$ heisst: "g wächst asymptotisch mindestens so schnell wie wie f"
- $g \in \omega(f(n))$ heisst: "g wächst asymptotisch echt schneller als f"
- $g \in \Theta(f(n))$ heisst: "g und f verhalten sich asymptotisch gleich"

Beispiel:

$b(n) = n^2$, $c(n) = n$, $d(n) = 0.5 \cdot n^2 - 10 \cdot n$

Es gilt:

- $b(n) \in O(b(n))$, trivial mit $c = 1, n_0 = 1$
- $b(n) \in O(d(n))$, z.B. mit $c = 4, n_0 = 10$
- $d(n) \in O(b(n))$, trivial
- $c(n) \in o(b(n))$, für $n_0 = \lceil 1/c \rceil + 1$
- $d(n) \notin o(b(n))$, da für $c > 2$ es immer $\exists n_0$ sodass für $n \geq n_0$: $d(n) > c \cdot b(n)$
- $b(n) \in \Omega(c(n))$, trivial mit $c = 1, n_0 = 1$
- $b(n) \in \omega(c(n))$, trivial mit $c = 1, n_0 = 1$
- $b(n) \in \Omega(d(n))$, für $c = 1$, $n_0 = 1$
- $d(n) \in \Omega(b(n))$, für $c = 4$, $n_0 = 10$
- $d(n) \in \Theta(b(n))$ und $b(n) \in \Theta(d(n))$

Problem 1-1 Betrachten Sie folgende Funktionen und ordnen Sie sie gemäß ihrem asymptotischen Verhalten (nehmen Sie an, dass alle gebrochenen Funktionswerte nach unten abgerundet werden):

$a(n) = 3 \cdot n - \sqrt{n}$, $b(n) = n \log n$, $c(n) = \sqrt{n}$, $d(n) = n^{13}/n^{12.6}$, 2^n, $3^{n/2}$, $n!$.

Problem 1-2 Sei $f(n)$ ein Polynom in n von Grad genau k. Beweisen Sie:

1. $f(n) \in \Theta(n^k)$
2. Für $l > k$ gilt $f(n) \in o(n^l)$
3. Für $l < k$ gilt $f(n) \in \omega(n^l)$

Problem 2-1

Beweisen oder widerlegen Sie:

$$f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n)) \text{ und } g(n) \in O(f(n))$$

Problem 2-2

Beweisen oder widerlegen Sie:

$$f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in \Omega(g(n)) \text{ und } g(n) \in \Omega(f(n))$$

Problem 3-1 Geben Sie obere (O(...)) und untere (Ω(....)) Schranken für die Laufzeit der folgenden (zugegebenermaßen nicht sonderlich spannenden) Algorithmen an. Hinweis: Bedenken Sie, dass eine for-Schleife implizit einen booleschen Ausdruck auswertet (Größenvergleich der Schleifenvariable).

```
f1(n)
{
  sum=0
  for i=1 to n
    for j=1 to n
       sum=sum+1
  return sum
}
```

```
f2(n)
{
  sum=0
  for i=1 to n
    for j=i to n
       sum=sum+1
  return sum
}
```

```
f3(n)
{
  sum=0
  for i=1 to n
    for j=1 to i
       sum=sum+1
  return sum
}
```

```
f4(n)
{
  sum=0
  for i=1 to 1000*n
    for j=1 to 1000
       sum=sum+1
  return sum
}
```

```
f5(n)
{
  sum=0
  for i=1 to n-1
    sum=sum+f5(i)

  return sum
}
```

```
f6(n)
{
  sum=0
  for i=1 to n
    sum=sum+1
  sum=sum+f6(n/2)+f6(n/2)
  return sum
}
```

```
f7(n)
{
  sum=0
  for i=1 to n
    sum=sum+1
  sum=sum+f7(n/2)
  return sum
}
```

```
f8(n)
{
  sum=0
  for i=1 to n
    sum=sum+1
  sum=sum+4*f8(n/2)
  return sum
}
```

```
f9(n)
{
  sum=0
  for i=1 to n
    sum=sum+f9(n-1)
  return sum
}
```

```
f10(n)
{
  sum=0
  for i=1 to 99999
    sum=sum+1
  return sum
}
```

```
f11(n)
{
  if (n>=2)
     return f11(n/2)+1
  else
     return 1
}
```

```
f12(n)
{
  if (n>=4)
     return f12(sqrt(n))
  else
     return 1
}
```