STARKER START ins Studium

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Fachbereich Informatik und Mathematik
Institut für Informatik

Vorsemesterkurs Informatik

Sommersemester 2013

Aufgabenblatt Nr. II.1

Aufgabe 1 (Wachstum von Funktionen)

Ordnen Sie die folgenden Funktionen nach ihrer Größe, wenn n gegen ∞ geht. Unterstreichen Sie dabei in jeder Funktion den Summanden, der am größten wird.

a) $n \cdot e^n + n$

b) $n \cdot \ln(n)$

c) $n^2 + 3n + 1$

d) $n^2 + 2n - 5$

e) $\log_3(n) + 1000$

f) $n^{31} + e^n$

Aufgabe 2 (Beweis Lemma 5.9)

Sei $a \in \mathbb{N}$, und $k > 1$:

$$f_1(n) := \log_a n, \quad f_2(n) := n, \quad f_3(n) := n \cdot \log_a n, \quad f_4(n) := n^k, \quad f_5(n) := a^n$$

Zeigen Sie, dass $f_i = O(f_{i+1})$ für $1 \leq i \leq 4$.

Aufgabe 3 (ausschlaggebende Terme)

Sortieren Sie die Terme innerhalb der folgenden Funktionen nach ihrer Größe, wenn n gegen ∞ geht. Geben Sie dann für jede der Funktionen $f(n)$, die kleinstmögliche Funktion $g(n)$ an, sodass $f = O(g)$ gilt.

a) $f(n) = n \cdot (e^n)^2 + n^2 \cdot e^n$

b) $f(n) = 100 \cdot \ln(n) + n \cdot \ln(n)$

c) $f(n) = n^2 + 20 \ln(n)^2 + 4 \cdot n \cdot \ln(n)^2$

d) $f(n) = 3 \cdot n^{20} + 3n \cdot \ln(n)^{30} - 5$

e) $f(n) = n^{31} + e^{4 \cdot n} + e^n$

Aufgabe 4 (Laufzeiten)

Für ein Problem seien 5 Algorithmen $A_1, \ldots, A_5$ gegeben, die das Problem lösen. Dabei benötigt Algorithmus A_i, $b_i(n)$ Operationen zur Lösung eines Problems mit Eingabelänge n. Es gilt:

$$b_1(n) = \log_2(n), \quad b_2(n) = n, \quad b_3(n) = n^2, \quad b_4(n) = 1000 \cdot n^2, \quad b_5(n) = 2^n$$

Wie verändert sich der Berechnungsaufwand, wenn

a) die Eingabelänge verdoppelt wird?

b) die Eingabelänge um 1 erhöht wird?

Hinweis: Überlegen Sie was *logarithmisches*, *lineares*, *quadratisches* und *exponetielles* Wachstum bedeutet. Falls Sie das nicht weiter bringt, setzen Sie $2n$ bzw. $(n+1)$ in die entsprechende Funktion ein, und formen Sie diese so um, dass Sie $b_i(n)$ einsetzen können.

Aufgabe 5 (Pseudocode)

Betrachten Sie die folgenden Algorithmen in Pseudocode.

a) Versuchen Sie zu verstehen, welche Rechenvorschriften diese Algorithmen darstellen. (Auch wenn diese manchmal keinen tieferen Sinn haben)

b) Geben Sie die Laufzeit in der O-Notation an.

Algorithmus 1:

```
1  function algo_1(A[0..n]){
2    zahl=A[0];
3    for (int i = 1; i <= n; i++){
4      if(A[i]<=zahl){
5        zahl=A[i];
6      }
7    }
8    return(zahl);
9  }
```

Algorithmus 2:

```
1  function algo_2(n){
2    i = 1;
3    for (int i = 1; i <= n; i++){
4      for(int j = 3; j <= 3n; i++){
5          for (int l = -3; l <= n/2; l++){
6            print i + j - l;
7          }
8      }
9    }
10 }
```

Algorithmus 3:

```
1  function algo_3(n){
2    i = n;
3    while (1 <= i){
4      i = i / 3;
5    }
6  }
```

Algorithmus 4:

```
1  function algo_4(n){
2    for (int i = 1; i <= n*n; i = i * n){
3      print i;
4    }
5  }
```

Algorithmus 5:

```
1  function algo_5(n){
2    zahl = 5;
3    for (int i = 1; i <= 5*n; i++){
4      zahl = zahl * 5;
5    }
6    while(zahl >= 1){
7      print zahl;
8      zahl--;
9    }
10 }
```

Algorithmus 6:

```
1  function algo_6(A[0..n]){
2    for(int i=0; i<=n;i++){
3      zahl=algo_1(A[i..n]);
4      vertausche A[i] mit zahl;
5    }
6  }
```