STARKER START ins Studium

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Fachbereich Informatik und Mathematik
Institut für Informatik

Vorsemesterkurs Informatik

Sommersemester 2015

Aufgabenblatt Nr. 6B

Aufgabe 1 (Beweis Lemma 6.9)

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

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

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

Aufgabe 2 (O und Θ)

Seien $f\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ zwei Funktionen. Welche der folgenden Aussagen stimmen? Gib einen Beweis oder ein Gegenbeispiel an.

a) Wenn $f \leq g$, dann gilt $f = O(g)$.

b) Wenn $f \leq 3 \cdot g$, dann gilt $f = O(g)$

c) Wenn $f = \Theta(g)$, dann ist $f \leq g$ oder $g \leq f$.

d) Wenn $f = g$, dann ist $f = \Theta(g)$.

e) Wenn $f = \Theta(g)$, dann ist auch $g = \Theta(f)$.

f) Wenn $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, dann ist $f = O(g)$.

g) Wenn $f = O(g)$, dann $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Aufgabe 3 (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: Überlege was *logarithmisches*, *lineares*, *quadratisches* und *exponetielles* Wachstum bedeutet. Falls dich das nicht weiter bringt, setze $2n$ bzw. $(n+1)$ in die entsprechende Funktion ein, und formen diese so um, dass du $b_i(n)$ einsetzen kannst.

Aufgabe 4 (Pseudocode)

Betrachte die folgenden Algorithmen in Pseudocode.

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

b) Gib die Laufzeit in der O-Notation an.

Algorithmus 1:

```
1  function algo_1(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 2:

```
1   function algo_2[A[0..n], i, j]{
2     x=A[i];
3     A[i]=A[j];
4     A[j]=x;
5     return(A[0..n]);
6     }
```

Algorithmus 3:

```
1  function algo_3(A[0..n]){
2    zahl=0;
3    for (int i = 1; i <= n; i++){
4      if(A[i]<=A[zahl]){
5        zahl=i;
6      }
7    }
8    return(zahl);
9  }
```

Algorithmus 4:

```
1  function algo_4(A[0..n]){
2    for(int i=0; i<=n;i++){
3      zahl=algo_3(A[i..n]);
4      algo_2(A[0..n],i, zahl);
5    }
6    return(A[0..n])
7  }
```

Tipp: Erinnere dich, was die Funktion von `algo_3` und `algo_2` ist. Versuche die einzelnen Zeilen in Anweisungen in menschlicher Sprache zu übersetzen. Falls das nicht hilft, wende den Algorithmus auf einen kleinen Array an.

Algorithmus 5:

```
1  function algo_5(A[0..n]){
2    for(int i=0; i<=n;i++){
3      for(int j=n; j>=i; j--){
4        if(A[j]<A[j-1]){
5          x=A[j];
6          A[j]=A[j-1];
7          A[j-1]=x;
8        }
9      }
10   }
11   return(A[0..n])
12 }
```