GOETHE UNIVERSITÄT FRANKFURT AM MAIN
Fachbereich Informatik und Mathematik
Institut für Informatik

Vorsemesterkurs Informatik

Sommersemester 2012

Aufgabenblatt Nr. II.2

Aufgabe 1

Betrachte die Funktionen:

$$f_1(n) = 5^n \quad f_2(n) = 6n + 3 \quad f_3(n) = 4^{\log_2(n)} \quad f_4(n) = 4^{(4^n)}$$

$$f_5(n) = \sum_{i=3}^{n^2} i \quad f_6(n) = n^n \quad f_7(n) = n!$$

Für welche der Funktionenpaare gilt: $f_i = O(f_j)$ mit $i, j \in \{1, 2, ..., 7\}$?

Aufgabe 2

(a) Weise nach, dass für $a, b \in \mathbb{N}$ und $a \leq b$ die folgenden Beziehungen gelten:

$$\log_b(n) = O(\log_a(n)) \quad n^a = O(n^b) \quad n \cdot \log_b(n) = O(n \cdot \log_a(n)) \quad a^n = O(b^n)$$

(b) Betrachte die Funktionen mit $a \in \mathbb{N}$:

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

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

Aufgabe 3

In der Vorlesung haben wir ein Lemma betrachtet:

Wenn der Grenzwert der Folge $\frac{f(n)}{g(n)}$ existiert und sei $c := \lim_{n\to\infty} \frac{f(n)}{g(n)}$ und $0 \leq c < \infty$
dann gilt $f = O(g)$

Gilt auch die Rückrichtung. D.h. gilt:

Wenn $f = O(g)$,
dann existiert der Grenzwert der Folge $\frac{f(n)}{g(n)}$ und für $c := \lim_{n\to\infty} \frac{f(n)}{g(n)}$ gilt $0 \leq c < \infty$

Hinweis: Gibt es zwei Funktionen f, g, s.d. der Grenzwert von $\frac{f(n)}{g(n)}$ nicht existiert?

Aufgabe 4

Betrachte die folgenden Algorithmen:
Algorithmus 1:

```
1  function algo_1(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 2:

```
1  function algo_2(n){
2    i = n;
3    while (1 <= i){
4      i = n / 3;
5    }
6  }
```

Algorithmus 3:

```
1  function algo_3(n){
2    for (int i = 1; i <= n*n; i = i * n){
3      print i;
4    }
5  }
```

Algorithmus 4:

```
1  function algo_4(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 5:

```
1  function algo_5(n){
2    k = n;
3    i = 0;
4    while (k > 0){
5      k = k - i;
6      i = i + 1;
7    }
8  }
```

Algorithmus 6:

```
1  function algo_6(n){
2    int i = 1;
3    int j = 0;
4    while(i <= n){
5      i = i+1;
6      j = i;
7      while(j <= 0){
8        j = j * i;
9      }
10   }
11 }
```

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

(b) Gebe die Laufzeit der Algorithmen in der O-Notation an.

Aufgabe 5

Überlege dir Algorithmen, die wir hier noch nicht behandelt haben, die die Laufzeit $O(n^4), O(\log_4 n), O(\ln n), O(9^n), O(n!)$ haben.

Aufgabe 6

Betrachte den Algorithmus der Binären Suche aus der Vorlesung. Betrachte das Array:

$$A = [5, 8, 9, 12, 16, 18, 19, 25, 29, 32, 36, 38, 42, 48, 49, 50]$$

(a) Sei $i = 12$. Rechne nach, wie sich der Algorithmus bei der Eingabe A und 12 verhält. Wie viele Schritte benötigt er?

(b) Sei $i = 39$. Rechne nach, wie sich der Algorithmus bei der Eingabe A und 39 verhält. Wie viele Schritte benötigt er?

(c) Betrachte das folgende Array:

$$B = [11, 15, 16, 18, 22, 26, 21]$$

Arbeitet der Algorithmus bei der Eingabe von B korrekt? Begründe deine Antwort.

Aufgabe 7

Betrachte den folgenden Algorithmus:

```
1  function bubble(int[] A){
2    int n = "die Groesse des Array's A";
3    for (int i = n; i > 1; i--){
4      for (int j = 0; j < i-1; j++){
5        if (A[j] > A[j+1]){
6          int temp = A[j];
7          A[j] = A[j+1];
8          A[j+1] = temp;
9        }
10     }
11   }
12 }
```

(a) Betrachte den Algorithmus und versuche die Rechenschritte nachzuvollziehen.

(b) Erkläre mit wenigen Worten, was der Algorithmus berechnet.

(c) Sei $A = [20, 3, 6, 88, 1]$. Rechne nach, wie dieser Algorithmus den Array verändern würde.

(d) Analysiere die Laufzeit des Algorithmus, indem du nur die Befehle zählst.

(e) Analysiere die Laufzeit des Algorithmus, indem du die Vergleiche zählst.

Aufgabe 8

Zeige, dass:

- Sei d eine feste positive reelle Zahl. Dann gilt $O(d) = O(1)$.
- Sei d eine feste, positive reelle Zahl und sei k eine natürliche Zahl. Dann gilt $O(d \cdot n^k) = O(n^k)$.
- Sei d eine feste, positive reelle Zahl und sei $f(n)$ eine Funktion, die von einer natürlichen Zahl n abhängt. Dann gilt: $O(d \cdot f(n)) = O(f(n))$.

allgemein gilt.

Aufgabe 9

Betrachte den ersten Algorithmus aus der Vorlesung:

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

(a) Gilt für diesen Algorithmus, dass er die Laufzeit $O(n!)$ besitzt?

(b) Welche Probleme werden in Aufgabenteil (a) ersichtlich?

Wir führen nun die Ω-Notation ein:

$$f = \Omega(g) \Leftrightarrow g = O(f)$$

(c) Betrachte die Beispiele zum Vergleichen von Funktionen aus dem Skript. Wir haben dort jeweils gezeigt, dass $f = O(g)$. Gilt jeweils auch $f = \Omega(g)$?

(d) Modifiziere Lemma 3 so, so dass wir aus einem Grenzwert aussagen können, dass $f = \Omega(g)$ gilt.

(e) Welche Auswirkungen hat die Ω-Notation auf die Laufzeitanalyse? Kann man diese für die Laufzeitanalyse verwenden? Wenn ja, was sagt sie über die Laufzeit eines Algorithmus aus?

(f) Gilt für den oberen Algorithmus, dass er die Laufzeit $\Omega(n^2)$ besitzt?

(g) Gilt für den oberen Algorithmus, dass er die Laufzeit $\Omega(\log_2 n)$ besitzt?

(h) Welche Probleme hat die Ω-Notation?

Wir führen nun die Θ-Notation ein:

$$\text{Wenn } f = O(g) \text{ und } g = O(f), \text{ dann schreiben wir: } f = \Theta(g).$$

(i) Welchen Vorteil könnte uns diese Notation bringen?