DSA Blatt 02

Leonard Oertelt 1276156
Julian Opitz 1302082

Aufage 4:

a)

Fibonacci:

Rekursiv mit 46 als Eingabe: 1836311903 (Berechnung benötigte ca. 4.795s)

Iterativ mit 46 als Eingabe: 1836311903 (Berechnung benötigte ca. $2.1 * 10^{-8}$s)

b)

Fakultät:

Rekursiv mit 10 als Eingabe: 3628800 (Berechnung benötigte ca. $2.6 * 10^{-8}$s)

Iterativ mit 10 als Eingabe: 3628800 (Berechnung benötigte ca. $1.628 * 10^{-9}$s)

c)

Pseudocode Logarithmusfunktion:

Algorithm logarithmusNIterationen(x,n)

sum ← 0;
for k ← 0 **to** n **do**
sum ← sum + $\frac{(x-1)^{2k+1}}{(2k+1)(x+1)^{2k+1}}$;
return 2 * sum;

d)

Algorithm logarithmusNIterationenOptimized(x,n)

sum ← 0;
bruch ← $\frac{x-1}{x+1}$;
for k ← 0 **to** n **do**
exponent ← 2k + 1;
sum ← sum + $\frac{bruch^{exponent}}{exponent}$;
return 2 * sum;

e)

Eingabe	Alg c 5 Schritte	Alg c 10 Schritte	Alg c 20 Schritte	Alg d 5 Schritte	Alg d 10 Schritte	Alg d 20 Schritte	Taschenrechner	Alg mit Eps = 1E-9
0	-3,574	-4,267	-4,267	-3,574	-4,267	-4,267	Error	-21,994
1	0	0	0	0	0	0	0	0
2	0,693	0,693	0,693	0,693	0,693	0,693	0,693	0,693
4	1,385	1,386	1,386	1,385	1,386	1,386	1,386	1,386
8	2,055	2,078	2,079	2,055	2,078	2,079	2,079	2,079
10	2,255	2,299	2,303	2,255	2,299	2,303	2,303	2,303

Aufgabe 5:

a)

$2^{n+1} \in O(2^n)$ mit $f(n) = 2^n$ und $g(n) = 2^{n+1}$

daraus folgt:

1. $2^{n+1} \leq c * 2^n$
2. nach n umstellen:

   $2^{n+1} \leq c * 2^n$ | $\log_2$

   $n+1 \leq \log_2 c + n$ | - n

   $1 \leq \log_2 c$ | $2^{()}$

   $2^1 \leq c$

   n ist weggefallen:

   die Ungleichung ist erfüllt, wenn $c \geq 2$.
3. Schluss:

   Also gilt für $g(n) = 2^{n+1}$ und $f(n) = 2^n$ und $c \geq 2$ die Aussage $g \in O(f)$

   $2^{n+1} = 2 * 2^n$

– q.e.d.

b)

$2^{n+1} \in \Omega(2^n)$ mit $f(n) = 2^n$ und $g(n) = 2^{n+1}$

daraus folgt:

1. $c * f(n) \leq g(n)$
2. umstellen:

   $c * 2^n \leq 2^{n+1}$ | $\log_2$

   $\log_2 c + n \leq n + 1$ | -n

   $\log_2 c \leq 1$ | $2^{()}$

   $c \leq 2$

3. Schluss:

die Ungleichung ist erfüllt, wenn $c \leq 2$.

Also gilt für $g(n) = 2^{n+1}$ und $f(n) = 2^n$ und $c \leq 2$ die Aussage $g \in \Omega(f)$

$2^{n+1} = 2 * 2^n$

- q.e.d.

c) Zu beweisen:

$\frac{n(n-2)}{3} \in O(n^2)$

$\frac{n(n-2)}{3} = \frac{1}{3}n^2 - \frac{2}{3}n$

Rechenregel der Addition:

$f = \frac{1}{3}n^2$

$g = \frac{-2}{3}n$

$f + g \in O(\max\{f,g\}) = O(f)$ da $g \in O(f) = \frac{-2}{3}n \in O(n^2)$

daraus folgt Terme niedrigerer Ordnung als n^2 fallen weg (Folie 33):

$\frac{1}{3}n^2 \in O(n^2)$

Rechenregel der Multiplikation:

$a = \frac{1}{3}$

$b = n^2$

$a \in O(1) \wedge b \in O(n^2) \rightarrow a * b \in O(1 * n^2)$

daraus folgt, mit Bezug auf Folie 33, konstante Faktoren fallen weg:

$n^2 \in O(n^2)$

- q.e.d.

d)

$10000 \cdot n \cdot (n+100) \cdot (n+1000)$

$= 10000n^3 + 11000000n^2 + 1000000000n$

Terme Niedrigerer Ordnung als das Polynom fallen weg:

$O(10000n^3 + 11000000n^2 + 1000000000n)$

$= O(10000n^3)$

Konstante Faktoren fallen weg:

$O(10000n^3)$

$= O(n^3)$

– q.e.d.

e)

$$\frac{n^2 + 19n - 2}{\sqrt{n}}$$

$= (n^2 + 19n - 2) \cdot n^{-\frac{1}{2}}$

$= n^{\frac{3}{2}} + 19n^{\frac{1}{2}} - 2n^{-\frac{1}{2}}$

Terme Niedrigerer Ordnung als das Polynom fallen weg:

$O(n^{\frac{3}{2}} + 19n^{\frac{1}{2}} - 2n^{-\frac{1}{2}}) = O(n^{\frac{3}{2}}) = O(n \cdot \sqrt{n})$

– q.e.d.

f)

$(n+1)! \in O(n!)$

$(n+1)! = n! \cdot (n+1)$

$n! \cdot (n+1) \leq c \cdot n!$ | :n!

$n + 1 \leq c$ | -1

$n \leq c - 1$

Da (c-1) konstant ist und n unendlich groß werden kann ist die Ungleichung nicht erfüllt:

$(n+1)! \notin O(n!)$