Algorithmen und Datenstrukturen – Serie 4

1 (6 Punkte) Laufzeitanalyse

Geben Sie die Worst Case Laufzeitfunktion für die folgenden Algorithmen an. Fassen Sie die Konstanten c soweit wie möglich zu einer zusammen. Achten Sie genau darauf, welche Konstanten sich zusammenfassen lassen. Verwenden Sie c_1,c_2,... als Bezeichnung verschiedener Konstanten. Sind mehrere Algorithmen angegeben, ist die Laufzeitanalyse ausgehend von *start()* durchzuführen. Gehen Sie davon aus, dass die Laufzeit zur Initialisierung eines Arrays konstant O(1) ist. (4 Punkte)

a)
```
swap (a)
    tmp = a[1]
    a[1] = a[2]
    a[2] = tmp
```

b)
```
swapAndAdd (a)
    FOR i = 1 TO a.length DO
        tmp = a[1]
        a[1] = a[2]
        a[2] = tmp
    ENDDO
    a[1] = a[1]+1
    a[2]++
```

c)
```
start (n)
    a = array(n)
    a[1] = 5
    a[2] = 1
    kaese (a)
    wurst (a)

kaese (a)
    FOR i = 1 TO a.length DO
        a[i] = "kaese"
    ENDDO

wurst (a)
    FOR i = 1 TO a.length/2 DO
        a[i] = "wurst"
        kaese (a)
    ENDDO
```

d)
```
start (n)
     A = array(n)
     salat (A)

salat (A)
     IF A.length < 2 THEN
           RETURN (NIL)
     ENDIF
     B = array(A.length/2)
     salat (B)
```

e) Geben Sie die Laufzeitfunktion für die in der Vorlesung vorgestellten Algorithmen *SequentialSearch* und *BinarySearch* an. (2 Punkte)

2 (10 Punkte) Wachstumsraten

Gegeben sind die folgenden 8 Funktionen, bezeichnet als $T_{\mathrm{a}}, T_{\mathrm{b}}, \ldots, T_{\mathrm{h}}$:

a) $n \mapsto n * c_1 + c_2$ b) $n \mapsto 15n^2 + n \sin(42 * n^n + n!)$ c) $n \mapsto 2^{n+1}$

d) $n \mapsto 6\sqrt{n^5}$ e) $n \mapsto \frac{n^2 * \sqrt{100}}{n^2 + \sqrt{100}}$ f) $n \mapsto \frac{n^{13}}{n-1} + 1987 + 16279n^5$

g) $n \mapsto 123 + \frac{2n}{n+n}$ h) $n \mapsto \log(3n^{(n^2)})$

Berechnen Sie für jede Funktion T_x ($x = \mathrm{a}, \ldots, \mathrm{h}$) mittels Additions-, Produkt- und Koeffizientenregel sowie ggfalls weiterer passender Methoden die **jeweils minimal wachsende** asymptotische Wachstumsrate $O(f)$, so dass $T_x \in O(f)$, für eine Funktion f von n. Verwenden Sie die in der Vorlesung vorgestellten typischen Wachstumsraten, wobei sie gegebenenfalls andere Konstanten verwenden müssen und drücken Sie sie möglichst einfach aus.
Ordnen sie die Funktionen T_x aufsteigend, d.h. es stehe T_x vor T_y, genau dann wenn $T_x \in O(T_y)$.
c_* bezeichnet Konstanten im Sinne der Vorlesung.

3 (3 Punkte) Obere und Untere Schranken

Geben Sie in einer Tabelle jeweils für zwei gegebene Funktionen f und g mit `ja` oder `nein` an, ob $f \in O(g)$ ist, ob $f \in \Omega(g)$ ist und ob $f \in \Theta(g)$.

a) $f(n) = \frac{n^4 - n + 1}{n-2}$
$g(n) = n^2 + 2n^3 + 1$

b) $f(n) = (n^2 - \frac{1}{n}) * (n - \frac{1}{n}) * n^2$
$g(n) = n^4 - 4n + 4$

c) $f(n) = n^6$
$g(n) = 6^n$

$f(n)$	$g(n)$	$f \in O(g)$	$f \in \Omega(g)$	$f \in \Theta(g)$

4 (3 Punkte) Master-Theorem

Gegeben sei folgender Pseudocode:

```
b (n, x)
    erg = x
    IF (n > 1) THEN
        FOR i = 1 TO n DO
            erg = erg + (-1)^i * i
        ENDDO
        erg = erg + b(n/3,erg/2)
        erg = erg + b(n/3,erg/4)
        erg = erg + b(n/3,erg/8)
        erg = erg + b(n/3,erg/8)
    ENDIF
    return erg;
```

a) Geben Sie die Laufzeitfunktion `T(n)` für den gegebenen Pseudocode an, sodass dieser mittels Master-Theorem analysiert werden kann.

b) Berechnen Sie mittels Master-Theorem mit der auf den Bemerkungsfolien vermerkten vereinfachten Fallunterscheidung das `Wachstum von T(n)`. Geben Sie den Rechenweg mit an.

c) Berechnen Sie mittels Master-Theorem ohne der auf den Bemerkungsfolien vermerkten vereinfachten Fallunterscheidung das `Wachstum von T(n)`, nutzen Sie also nur die auf Folien 262-267 vorgestellte ursprüngliche Fallunterscheidung des Master-Theorems. Geben Sie den Rechenweg mit an.