Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Sommersemester 2021
25. 04. 2021

Prof. Dr. Bernhard Seeger
Nikolaus Glombiewski M.Sc.

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Abgabe: 02. 05. 2021,
bis **spätestens** 23:59 Uhr
über die ILIAS Plattform

Übung 2

Aufgabe 2.1: $\mathcal{O}$-Notation Zuordnung (1+1) (2 Punkte)

Ordnen Sie die Methoden f1 und f2 den niedrigsten Laufzeiten in $\mathcal{O}$-Notation zu:
1 , $\log n$, $\sqrt{n}$, n , $n \log n$, n^2 , n^3 , 2^n , 3^n , $n!$.
Begründen Sie Ihre Antworten.

```java
public static int f1(int n) {
    int x = 0;
    for ( int i = 0; i < n; i++ )
        for ( int j = 0; j <= i; j++ )
            for ( int k = 0; k < 5; k++ )
                x = x + j;
    return x;
}

public static int f2(int n) {
    if ( n <= 1 )
        return 2;
    else
        return f2(n-1) * f2(n-2);
}
```

Aufgabe 2.2: $\mathcal{O}$-Notation Berechnung (1+1+1+1) (4 Punkte)

Beweisen Sie die folgenden Aussagen:

a) Gegeben $f(n) = n^3 + n$, zeigen Sie: $f(n) = O(n^3)$

b) Gegeben $f(n) = \log(n^2)$, zeigen Sie: $f(n) = O(\log(n))$

c) Gegeben $f(n) = \sqrt(n)$, zeigen Sie: $f(n) = \Omega(\log(n))$

d) Gegeben $f(n) = 4n^4 + n^3$, zeigen Sie: $f(n) = \Theta(2n^4 + 15n^3)$

Aufgabe 2.3: $\mathcal{O}$-Notation Aussagen (1+1+1) **(3 Punkte)**

Zeigen oder widerlegen Sie folgende Aussagen:

a) $f(n) = \mathcal{O}(f(n)^2)$

b) $\forall \epsilon > 0 : n \log(n) = \mathcal{O}(n^{1+\epsilon})$

c) Wenn $f(n) = \mathcal{O}(g(n))$, dann gilt $2^{f(n)} = \mathcal{O}(2^{g(n)})$

Aufgabe 2.4: Rekursive Algorithmen (4+2) **(6 Punkte)**

In der Vorlesung wurde folgende Rekursionsgleichung für die Laufzeit der binären Suche hergeleitet:

$$T(n) = \begin{cases} b + T\left(\frac{n-1}{2}\right) & \text{, falls } n > 0 \\ a & \text{, sonst} \end{cases}$$

und folgende Vermutung für die geschlossen Form aufgestellt: $T(n) = \log_2(n+1) \cdot b + a$ (vgl. Skript S. 86).

a) Beweisen Sie mittels vollständiger Induktion, dass die geschlossene Form gilt. Bedenken Sie, dass wir vereinfachend annehmen, dass $n = 2^k - 1$ ist und daher die Induktion auch über k vollzogen werden kann.

b) Wie kann die Laufzeit für n abgeschätzt werden, die sich nicht als $2^k - 1$ darstellen lassen? **Hinweis: Die Kostenfunktion ist monoton.**

Aufgabe 2.5: Laufzeit von binärer Suche* (2+2+1) **(5 Punkte)**

Die binäre Suche prüft stets das mittlere Element eines Suchraums und schränkt diesen dann entsprechend ein. In der Praxis ist das mittlere Element meist nicht der optimale Einstieg. Setzen wir eine ungefähre Gleichverteilung der Daten voraus, können wir aus Minimal-, Maximalwert und dem gesuchtem Element schätzen, wo es im Array zu finden ist. Zur Lösung der folgenden Aufgaben sollen Sie die Vorlage `BinSearchPerformance.java` verwenden.

a) Implementieren Sie analog zu der `contains2`-Methode aus der Vorlesung (siehe Codevorlage) eine Methode `contains3`, die nicht das mittlere Element prüft, sondern jeweils interpoliert wo das gesuchte Element zu finden sein müsste.

b) Lassen Sie die Methoden `contains2` und `contains3` gegeneinander antreten. Generieren Sie dazu 10 „große“ Arrays mit sortierten positiven Zufallszahlen und suchen Sie nach einem zufälligen Element. Messen Sie die Rekursionstiefe der beiden Methoden mittels einer statischen Zählervariablen. Geben Sie die Laufzeiten der Durchgänge (= Rekursionstiefe) auf der Konsole aus und geben Sie diese mit ab.

c) Welcher Algorithmus ist im Best-/Worst-Case besser? Wie verhalten sich ihre empirischen Beobachtungen dazu?

*** Aufgabe 2.5 ist für Lehramtsstudierende optional.**