Übungsserie 2
Datenstrukturen & Algorithmen

Universität Bern

Frühling 2018

Übungsserie 2

- 7 theoretische Aufgaben
  - Laufzeitkomplexitäten
- Keine praktische Aufgabe
  - Keine Poolstunde

Aufgabe 1

> Spielen mit den Definitionen von $\mathcal{O}, \Omega, \Theta$
> Siehe auch Kapitel 3.1 im Buch
> Aufgabe 1(a)
  — Slides 6, 7 der zweiten Vorlesung
> Aufgabe 1(b)
  — Begründung mit Hilfe der Definition von $O(g(n))$ & $\Omega(g(n))$

Aufgabe 2

- Rechenregeln von Logarithmen auffrischen
- Siehe auch Kapitel 3.2 im Buch (s. 58)
- Aufgabe 2(a)
  - Auf beiden Seiten log_a berechnen
  - Rechenregeln des Logarithmus anwenden
- Aufgabe 2(b)
  - $\Theta(g(n)) = \Theta(h(n))$ wenn es eine Konstante c gibt so dass $g(n) = c*h(n)$

Aufgabe 3

- Rekursionsgleichung lösen
  - Induktion/Substitutionsmethode
  - Wie in der Vorlesung / Buch Kapitel 4.3

Aufgabe 3

UNIVERSITÄT
BERN

- Aufgabenstellung

3. Zeigen Sie mittels Induktion/der Substitutionsmethode, dass die Rekursionsgleichung $T(n) = 2T(\lceil n/4 \rceil + 12) + 3n$ die Lösung $O(n)$ hat. (**1 Punkt**)

- Vorgehen:
  - Erraten der allgemeinen Form der Lösung (hier bereits gemacht, `O(n)`)
  - Verifizieren durch Induktion
  - Bestimmen der Konstanten
- Schwierigkeit:
  - Geschickte Wahl von `c, d`
  - Sonst klappt Induktion nicht!
- Zu beweisen: $T(n) \leq c*n$ für eine geeignete Konstante `c`
  - **Problem** Diese Annahme ist nicht stark genug!

Substitutionsmethode

3. Zeigen Sie mittels Induktion/der Substitutionsmethode, dass die Rekursionsgleichung $T(n) = 2T(\lceil n/4 \rceil + 12) + 3n$ die Lösung $O(n)$ hat. (**1 Punkt**)

> Ersetze ursprüngliche Annahme durch stärkere Variante: $T(n) \leq cn - d$, wobei $d > 0$ eine weitere Konstante ist

> Zeige also:

$$\exists c, d, n_0 : \; T(n) \leq cn - d \;\; \forall \, n > n_0$$

Substitutionsmethode

UNIVERSITÄT BERN

3. Zeigen Sie mittels Induktion/der Substitutionsmethode, dass die Rekursionsgleichung $T(n) = 2T(\lceil n/4 \rceil + 12) + 3n$ die Lösung $O(n)$ hat. (**1 Punkt**)

$$\exists c, d, n_0 : \; T(n) \leq cn - d \;\; \forall \, n > n_0$$

> Technische Details I

— «Klassische» Induktion:

- $T(n+1) \leq c(n+1) - d$ zeigen unter der Annahme, dass $T(n) \leq cn - d$
- $T(n+1)$ nicht abhängig von $T(n)$, sondern von $T(n/4 + 12)$
- Klappt nicht!

— Besser:

- Annehmen, dass $T(k) \leq ck - d$ gilt für alle $k < n$
- Annahme verwenden, um $T(n)$ zu beweisen

Substitutionsmethode

3. Zeigen Sie mittels Induktion/der Substitutionsmethode, dass die Rekursionsgleichung $T(n) = 2T(\lceil n/4 \rceil + 12) + 3n$ die Lösung $O(n)$ hat. (**1 Punkt**)

$$\exists c, d, n_0 : \; T(n) \leq cn - d \;\; \forall \, n > n_0$$

- Technische Details II
  - Induktionsverankerung
    - Angenommen c, d, n_0 gut gewählt
    - Normalerweise: ein n einsetzen.
      - Aber was ist z.B. $T(13)$? Was ist $T(1)$?
- Vollständig formulierte Rekursionsgleichung beinhaltet Rekursion und Randbedingung!

Substitutionsmethode

> Oft wird die Randbedingung weggelassen

$$T(n) = 2T(\lceil n/4 \rceil + 12) + 3n$$

— $T(13) = ?$

— Rekursive Funktion eigentlich undefiniert.

> Bedeutung: Man kann n_0 für eine Randbedingung nach Gutdünken festlegen, Induktion muss für das festgelegte n_0 aber für beliebige C funktionieren.

$$T(n) = \begin{cases} C & wenn\ n < n_0 \\ 2T(\lceil n/4 \rceil + 12) + 3n & \text{sonst} \end{cases}$$

Randbedingungen

> Die Konvention mit Randbedingung in Ordnung **in unserem Kontext**
  - Siehe s. 69 im Buch
  - Zeitaufwand für einzelne Operation sowieso unbekannt, Algorithmen unter einer beliebigen festen Inputgrösse brauchen aber **konstante** Zeit
  - Konstante in der Regel nicht wesentlich für Komplexitätsklasse

Zusammenfassung Aufgabe 3

- z.z.: $T(n) = O(n)$, d.h. es existieren c, d so dass $T(n) \leq cn - d$
- **Zuerst** Induktionsschritt, **dann** Verankerung

1. **Induktionsschritt**
   - Annahme: $T(k) \leq ck - d$ für $k < n+1$
     Zeige: $T(n+1) \leq c(n+1) - d$
   - Schränke dabei c, d, n so ein, dass Induktion klappt, aber nur so stark wie nötig, z.B:

$$\frac{3c+4}{4}(n+1) + 13 \leq c(n+1)$$

Falls $c > 4$, n *gross genug. Z.B. beliebige* $\underline{c > 8}$ *und* $\underline{n \geq 13}$

2. **Induktionsverankerung**
   - Wähle Randbedingung passend $T(n) = C$ für $\underline{n \leq 13}$
   - Prüfe Verankerung: $T(13) = C \leq \underline{c} \cdot 13$
     - Passt, da c immer gross genug gewählt werden kann

Aufgaben 4, 5

UNIVERSITÄT BERN

- Aufgabe 4
  - Rekursionsbaum zeichnen
  - Wie in Vorlesung / Buch Kapitel 4.4
- Aufgabe 5
  - Mastermethode
  - Wie in Vorlesung / Buch Kapitel 4.5

Aufgabe 6

> $T(n)$ gegeben (in hundertstel Sekunden)

- Finde n so dass $T(n) = 10$
- Finde n so dass $T(n) = 1000$
- $T(n) * 0.01 = 10$ / $T(n) * 0.01 = 1000$ nach n auflösen

Aufgabe 7

> Asymptotische Laufzeit von Pseudocode abschätzen

```
i ← 1
while i < n
        do
            j ← 0
            while j ≤ i
                    do
                        j ← j + 1
            i ← 3i
```

Schlaufe wird nicht
n mal ausgeführt!

Fragen?

?