Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Janina Sontheim, Florian Richter

München, 07.05.2019

Algorithmen und Datenstrukturen

SS 2019

Übungsblatt 2: Grundlagen

Tutorien: 07.05.-14.05.2019

Aufgabe 2-1 *Laufzeitanalyse*

Finden eines Wertes value in binärem Baum t mit n Elementen (Pseudocode)

```
find(value, t):
  t_current = t.root
  while(t_current Is Not Leaf):
    if value == t_current.value:
      return True
    elseif value > t_current.value:
      t_current = t_current.right
    else:
      t_current = t_current.left
  return value == t_current.value
```

Suchen gleicher Elemente in zwei Listen (Java)

```
int compare(char[] a, char[] b){
  int count = 0;
  for(char c: a){
    for(char d: b){
      if(c == d) count++;
    }
  }
  return count;
}
```

(a) Betrachten Sie die Programme a - e auf dem ersten Übungsblatt sowie die beiden obigen Algorithmen und bearbeiten Sie für alle Programme die folgenden Punkte:

- Welche Komponenten haben konstante Laufzeit?
- Welche Komponenten haben eine Laufzeit von O(n)?
- Welche Komponenten haben andere Laufzeiten, und welche Laufzeiten sind das?

sollten Funktionen nur unter Umständen terminieren, formulieren Sie die Vorraussetzungen.

(b) Erstellen Sie zu jeder Aufgabe eine ausformulierte Laufzeitformel, welche sämtliche vorhandenen Komponenten beinhaltet.

(c) Formulieren Sie nun aus den erstellten Formeln gültige O-Notationen.

Aufgabe 2-2 *O-Notation*

Zeigen oder widerlegen Sie:

(a) $O(n) \subseteq O(n^2)$

(b) $log_{10}(x) \notin O(log_2(x))$

(c) $O(n^2) \subseteq O(n * log_2(n))$

(d) $20000 * g(n) \in O(g(n))$

(e) $g(n) * n^2 \in O(n^2)$

Aufgabe 2-3 *Rekursionsgleichungen und Mastertheorem*

a) Der Algorithmus, um die Türme von Hanoi zu lösen, hat als Java-Code folgende Form:

```
public static void Hanoi(int Scheibenindex, int start, int ziel, int temp){

  if(Scheibenindex == 1) {
    VersetzeObersteScheibe(start, ziel);
  }

  else {
    Hanoi(Scheibenindex - 1, start, temp, ziel);
    VersetzeObersteScheibe(start, ziel);
    Hanoi(Scheibenindex - 1, temp, ziel, start);
  }
}
```

Die Zeitkomplexität der Funktion VersetzeObersteScheibe(...) sei $\mathcal{O}(1)$. Stelle die Rekursionsgleichung für diesen Algorithmus auf und verwende das Mastertheorem, um die Zeitkomplexität in Abhängigkeit des Parameters $Scheibenindex$ zu ermitteln.

b) Löse die folgenden Rekursionsgleichungen mithilfe des Mastertheorems.

(i) $T(n) = 2T(\frac{n}{2}) + n^3$

(ii) $T(n) = T(\sqrt{n}) + 1$

(iii) $T(n) = a \cdot T(\sqrt[x]{n}) + log_y(n)$ mit x, y >1 und $log_x(a)$ <1.

Hinweis für (iii): Substituiere $m = log_y(n)$.