Algorithmen & Datenstrukturen

Blatt 5

Dr. Matthias Thimm
Tina Walber, Leon Kastler, Martin Leinberger und Maximilian Strauch
Fachbereich Informatik, Universität Koblenz-Landau

7. Dezember 2013

1 Rekursionsgleichungen aufstellen (5 Punkte)

Ermitteln Sie die Rekursionsgleichung $T(n) = \ldots$ für die angegebenen Algorithmen. Begründen Sie in Ihrer Antwort, wie Sie auf die einzelnen Summanden in der Rekursiongleichung gekommen sind.

a) (1 Punkt) Rekursionsgleichung für fac

```
int fac(int n) {
   if (n == 0)
      return 1;

   return n * fac(n-1);
}
```

b) (2 Punkte) Rekursionsgleichungen für $func$ und $mult$.

```
int func(int x, int y) {
   if (y == 1)
      return x;

   return func(mult(x, x), y-1);
}

int mult(int x, int y) {
   if (y == 0)
      return 0;

   return x + mult(x, y-1);
}
```

c) (2 Punkte) Rekursionsgleichung für *removeDuplicates*

```
List<Integer> removeDuplicates(List<Integer> l) {
   if (l.isEmpty())
      return new LinkedList<Integer>();

   element = l.pop();
   List<Integer> result = removeDuplicates(l);

   if (result.contains(element))
      return result;
   else {
      result.addFirst(element);
      return result;
   }

}
```

Hinweis Die folgende Tabelle erklärt die Bedeutung der jeweilige Funktion und gibt deren Aufwand an.

Operation	**Beschreibung**	**Aufwand**
list.isEmpty()	Testet, ob die Liste leer ist	$\Theta(1)$
list.pop()	Entfernt das erste Element aus der liste und gibt es zurück.	$\Theta(1)$
new LinkedList<Integer>()	Konstruiert eine neue, leere Liste.	$\Theta(1)$
list.contains()	Testet ob ein Wert in einer Liste enthalten ist.	$\Theta(l.size())$
list.addFirst(x)	Fügt das Element x an den Beginn der Liste *list* an.	$\Theta(1)$

2 Rekursionsgleichungen lösen (6 Punkte)

Lösen Sie die folgenden Rekursionsgleichungen mit Hilfe der vollständigen Induktion.

a) (2 Punkte)

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq 1 \\ T(n-1) + \Theta(n) & \text{sonst} \end{cases}$$

Vermutung: $T(n) \in O(n^2)$

b) (2 Punkte)

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq 2 \\ 4T(\frac{n}{2}) + \Theta(n^2) & \text{sonst} \end{cases}$$

Vermutung: $T(n) \in O(n^2 logn)$

c) (2 Punkte)

$$T(n) = \begin{cases} \Theta(1) & \text{für } n \leq 1 \\ 9T\left(\frac{n}{3}\right) + \Theta(n) & \text{sonst} \end{cases}$$

Vermutung: $T(n) \in O(n^2)$

3 Rekursionsbäume erstellen (4 Punkte)

Gegeben ist folgende Rekursionsgleichung:

$$T_1(n) = \begin{cases} \Theta(1) & \text{für } n \leq 0 \\ a \cdot T_1(n-c) + d & \text{sonst} \end{cases}$$

a) **Rekursionsbaum skizzieren (1 Punkt)** Skizzieren sie für $T_1(n)$ die ersten drei Ebenen des Rekursionsbaums. Bestimmen Sie mit Hilfe des Rekursionsbaums folgende Informationen: (i) Wieviele Knoten hat der Rekursionsbaum auf der Ebene i? (ii) Welche Tiefe hat der Rekursionsbaum in Abhängigkeit von n? (iii) Wieviele Blattknoten hat der Rekursionsbaum in Abhängigkeit von n?

b) **Summenformel aufstellen (1 Punkt)** Benutzen Sie die Informationen aus der vorherigen Teilaufgabe um die Rekursionsgleichung $T_1(n)$ mit Hilfe einer Summenformel auszudrücken.

c) **Summenformel für $a = 1$ lösen (1 Punkt)** Lösen Sie die Summenformeln von $T_1(n)$ aus der vorherigen Teilaufgabe für $a = 1$. Welche Aufwandsklassen ergeben sich?

d) **Summenformel für $a > 1$ lösen (1 Punkt)** Lösen Sie die Summenformel von $T_1(n)$ für $a > 1$. Welche Aufwandsklasse ergibt sich?

4 Mastertheorem (5 Punkte)

Zeigen Sie mit dem Mastertheorem, welcher Aufwandsklasse die folgenden Rekursionsgleichungen angehören:

a) $T(n) = 8 \cdot T(n/2) + n + \log n$

b) $T(n) = 4 \cdot T(n/2) + n^3$

c) $T(n) = 25 \cdot (n/5) + 25n$

d) $T(n) = 9 \cdot T(n/3) + n^2 \log n$

e) $T(n) = 4 \cdot T(n/4) + n^2 \log n$