Tutorium 2:
Korrektur, Mastertheorem, Schleifeninvarianten

Matthias Schimek | 3. Juni 2017

TUTORIUM ZUR VORLESUNG ALGORITHMEN I IM SS17

Gliederung

Schleifeninvarianten - Beispiel

```
1 Function f(n : ℕ) : ℕ
2     a = 1 : ℕ
3     i = 0 : ℕ
4     while i < n do
5         invariant a = 2^i
6         a := a + a
7         i := i + 1
8     end
9     return a − 1
```

- **IA.:** Vor 1. Schleifendurchlauf gilt $a = 2^0 = 1$ wegen Initialisierung
- **IV.:** Vor dem j. Schleifendurchlauf gelte $a = 2^i$
- **IS.:** Zeile 6: $a := a + a \overset{IV.}{=} 2^i + 2^i = 2^{i+1}$
  Zeile 7: $i := i + 1$
  $\Rightarrow$ Nach j. Schleifendurchlauf und damit vor $j+1$. Schleifendurchlauf gilt $a = 2^i$

Mastertheorem - Teile und Herrsche

Teile-und-Herrsche-Paradigma:

- Teile das Problem in kleinere Probleminstanzen auf
- Löse die kleineren Probleminstanzen
- Füge diese zu einer Gesamtlösung zusammen

Hierbei ergeben sich oft *Rekurrenzgleichungen*
$\Rightarrow$ mittels *Mastertheorem* lösen

Mastertheorem - Teile und Herrsche

Teile-und-Herrsche-Paradigma:

- Teile das Problem in kleinere Probleminstanzen auf
- Löse die kleineren Probleminstanzen
- Füge diese zu einer Gesamtlösung zusammen

Hierbei ergeben sich oft *Rekurrenzgleichungen*
$\Rightarrow$ mittels *Mastertheorem* lösen

Mastertheorem - Teile und Herrsche

Teile-und-Herrsche-Paradigma:

- Teile das Problem in kleinere Probleminstanzen auf
- Löse die kleineren Probleminstanzen
- Füge diese zu einer Gesamtlösung zusammen

Hierbei ergeben sich oft *Rekurrenzgleichungen*
$\Rightarrow$ mittels *Mastertheorem* lösen

Mastertheorem - Teile und Herrsche

Beispiel: Sortieralgorithmen
T(n) : Laufzeit bei Eingabegröße n

```
Function msort(a : array [1...n] of ℕ)
    if n = 1 then
        return a[1]
    end
    a1 = msort(a[1... (n/2)])          // T(⌈n/2⌉)
    a2 = msort(a[(n/2 + 1)... n])      // T(⌈n/2⌉)
    return merge (a1, a2)              // Θ(n)
```

$\Rightarrow$ Rekurrenzgleichung: $T(n) = 2 \cdot T\lceil n/2 \rceil + c \cdot n$

Mastertheorem - Teile und Herrsche

Beispiel: Sortieralgorithmen

T(n) : Laufzeit bei Eingabegröße n

Function `msort(`a : **array** $[1 \ldots n]$ **of** $\mathbb{N}$`)`

    if $n = 1$ **then**

        return $a[1]$

    end

    $a_1 =$ `msort(`$a[1 \ldots (n/2)]$`)` // T($\lceil n/2 \rceil$)

    $a_2 =$ `msort(`$a[(n/2+1) \ldots n]$`)` // T($\lceil n/2 \rceil$)

    return `merge` (a_1, a_2) // $\Theta(n)$

$\Rightarrow$ Rekurrenzgleichung: $T(n) = 2 \cdot T\lceil n/2 \rceil + c \cdot n$

Mastertheorem

Mittels *Mastertheorem* lassen sich folgende Rekurrenzen lösen:

- a, b, c und d positive Konstanten, $n \in \mathbb{N}$

$$\mathrm{T}(n) = \begin{cases} a & \text{für } n = 1 \\ d \cdot \mathrm{T}(n/b) + c \cdot n & \text{für } n > 1 \end{cases}$$

- Es gilt:

$$\mathrm{T}(n) \in \begin{cases} \Theta(n) & \text{für } d < b \\ \Theta(n \cdot \log n) & \text{für } d = b \\ \Theta(n^{\log_b d}) & \text{für } d > b \end{cases}$$

Mastertheorem

Karlsruhe Institute of Technology

Mittels *Mastertheorem* lassen sich folgende Rekurrenzen lösen:

- a, b, c und d positive Konstanten, $n \in \mathbb{N}$

$$T(n) = \begin{cases} a & \text{für } n = 1 \\ d \cdot T(n/b) + c \cdot n & \text{für } n > 1 \end{cases}$$

- Es gilt:

$$T(n) \in \begin{cases} \Theta(n) & \text{für } d < b \\ \Theta(n \cdot \log n) & \text{für } d = b \\ \Theta(n^{\log_b d}) & \text{für } d > b \end{cases}$$

Mastertheorem - Aufgaben (direkte Anwendung)

Gib möglichst scharfe asymptotische Schranken mit Hilfe des Mastertheorems an:

i) $\mathrm{A}(1) = 1$ und für $n \in \mathbb{N}_{>1}$: $\mathrm{A}(n) = 5 \cdot \mathrm{A}(\lceil n/5 \rceil) + 12 \cdot n$

ii) $\mathrm{B}(1) = 12$ und für $n \in \mathbb{N}_{>1}$: $\mathrm{B}(n) = \mathrm{B}(\lceil n/7 \rceil) + n$

iii) $\mathrm{C}(1) = 4$ und für $n \in \mathbb{N}_{>1}$: $\mathrm{C}(n) = 25 \cdot \mathrm{C}(\lceil n/5 \rceil) + \pi \cdot n$

Mastertheorem - Aufgaben (geschachtelt)

Gib möglichst scharfe asymptotische Schranken an.

i) $A(1) = 6$ und für $n \in \mathbb{N}_{>1}$:
$A(n) = 8 \cdot A(\lceil n/2 \rceil) + 4n + 3 \cdot \sqrt{n} + 17$

ii) $B(1) = 1$ und für $n \in \mathbb{N}_{>1}$:
$B(n) = 2 \cdot B(\lceil n/4 \rceil) + n + 3 \cdot \log n + 10$

iii) $D(1) = 5$ und für $n \in \mathbb{N}_{>1}$:
$D(n) = 7 \cdot D(\lceil n/3 \rceil) + E(n) + 7n$
$E(n) \in \Theta(n)$

Mastertheorem - Aufgaben (geschachtelt)

Gib möglichst scharfe asymptotische Schranken an.

i) $A(1) = 6$ und für $n \in \mathbb{N}_{>1}$:
$A(n) = 8 \cdot A(\lceil n/2 \rceil) + 4n + 3 \cdot \sqrt{n} + 17$

ii) $B(1) = 1$ und für $n \in \mathbb{N}_{>1}$:
$B(n) = 2 \cdot B(\lceil n/4 \rceil) + n + 3 \cdot \log n + 10$

iii) $D(1) = 5$ und für $n \in \mathbb{N}_{>1}$:
$D(n) = 7 \cdot D(\lceil n/3 \rceil) + E(n) + 7n$
$E(n) \in \Theta(n)$

Mastertheorem - Aufgaben (geschachtelt)

Gib möglichst scharfe asymptotische Schranken an.

i) $\mathrm{A}(1) = 6$ und für $n \in \mathbb{N}_{>1}$:
$\mathrm{A}(n) = 8 \cdot \mathrm{A}(\lceil n/2 \rceil) + 4n + 3 \cdot \sqrt{n} + 17$

ii) $\mathrm{B}(1) = 1$ und für $n \in \mathbb{N}_{>1}$:
$\mathrm{B}(n) = 2 \cdot \mathrm{B}(\lceil n/4 \rceil) + n + 3 \cdot \log n + 10$

iii) $\mathrm{D}(1) = 5$ und für $n \in \mathbb{N}_{>1}$:
$\mathrm{D}(n) = 7 \cdot \mathrm{D}(\lceil n/3 \rceil) + \mathrm{E}(n) + 7n$
$\mathrm{E}(n) \in \Theta(n)$

Rekurrenzen mittels vollst. Induktion

Gegeben sei folgende Rekurrenz für alle $n = 4^k$, wobei $k \in \mathbb{N}_0$:

$$\mathrm{T}(n) = \begin{cases} 2 & \text{falls } n = 1 \\ 2 \cdot \mathrm{T}(n/4) & \text{falls } n > 1 \end{cases}$$

Finden Sie eine Funktion $f : \mathbb{N} \to \mathbb{R}_+$ und Konstanten $c_1, c_2 \in \mathbb{R}_+$, sodass $c_1 \cdot f(n) \leq \mathrm{T}(n) \leq c_2 \cdot f(n)$ für alle $n = 4^k$ mit $k \in \mathbb{N}_0$, gilt und beweisen Sie die Behauptung.

Korrektheit - Merging

Das Merging-Problem ist folgendermaßen definiert:

Gegeben: aufsteigend sortierte Arrays $A[1...n_1]$, $B[1...n_2]$ von natürlichen Zahlen

Gesucht: aufsteigend sortiertes Array $C[1...(n_1 + n_2) =: n]$ von natürlichen Zahlen, das genau die Zahlen von A und B enthält

Beispiel: $A = [1, 3, 5, 7]$, $B = [2, 4, 6, 8]$
$\Rightarrow C = [1, 2, 3, 4, 5, 6, 7, 8]$

Korrektheit - Merging

Das Merging-Problem ist folgendermaßen definiert:

Gegeben: aufsteigend sortierte Arrays $A[1...n_1]$, $B[1...n_2]$ von natürlichen Zahlen

Gesucht: aufsteigend sortiertes Array $C[1...(n_1 + n_2) =: n]$ von natürlichen Zahlen, das genau die Zahlen von A und B enthält

Beispiel: $A = [1, 3, 5, 7]$, $B = [2, 4, 6, 8]$
$\Rightarrow C = [1, 2, 3, 4, 5, 6, 7, 8]$

Function merge(A : **array** $[1...n_1]$ **of** $\mathbb{N}$, B : **array** $[1...n_2]$ **of** $\mathbb{N}$) : **array** $[1...n]$ **of** $\mathbb{N}$

- **assert** $A[i] \leq A[j] \quad \forall i \leq j$ mit $i, j \in \{1, ...n_1\}$
- **assert** $B[i] \leq B[j] \quad \forall i \leq j$ mit $i, j \in \{1, ...n_2\}$
- $A[n_1 + 1] := \infty$, $B[n_1 + 1] := \infty$
- $n := n_1 + n_2, \;\; j_A := 1, j_B := 1$
- **for** $i := 1$ **to** n **do**
  - $C[i] = \min(A[j_A], B[j_B])$
  - **if** $A[j_A] < B[j_B]$ **then**
    - $j_A := j_A + 1$
  - **else**
    - $j_B := j_B + 1$
  - **end**
  - **invariant** $C[1 \,...\, i]$ enthält genau $A[1 \,...\, j_A - 1], B[1 \,...\, j_B - 1]$
  - **invariant** $B[k] \leq A[j_A] \;\; \forall k \in \{1, ..., j_B - 1\}$, $A[k] \leq B[j_B] \;\; \forall k \in \{1, ..., j_A - 1\}$
  - **invariant** $C[1 \,...\, i]$ ist sortiert
- **end**
- **assert** $j_A = n_1 + 1, j_B = n_2 + 1$
- **assert** $C[i] \leq C[j] \;\; \forall i \leq j, \;\; i, j \in \{1, ..., n\}$
- **assert** $C[1 \,...\, n]$ enthält genau $A[1 \,...\, n_1], B[1 \,...\, n_2]$
- **return** C

Zusammenfassung

- **Mastertheorem:** Nützlich bei Rekurrenzen
- Tipps fürs Übungsblatt: