Algorithmen 1 SS 2013 – Tutorium 7

1. Tutorium

Sarah Lutteropp

23. April 2013

Übersicht

Eure Tutorin

Über mich

- Sarah Lutteropp
- 20 Jahre
- 6. Semester Informatik, 5. Semester Mathematik
- Zum zweiten Mal Algorithmen-Tutorin

Kontakt

- Mail: sarah.lutteropp@student.kit.edu
- Jabber: quebbeck@jabber.ccc.de

Vorstellungsrunde

Jetzt seid ihr dran!

- Name?
- Studiengang?
- Semester?
- Bisheriger Eindruck von der Vorlesung?

Übungsblätter

Übungsblätter

- Erscheinen Mittwochs
- Abgabe Freitags bis 12:45 Uhr unten im Infobau
- Abgabe zu zweit möglich
- Rückgabe in den Tutorien

Beschriftung des Übungsblattes

- Name, Matrikelnummer, Nummer des Übungsblatts, Name des Tutors
- Nummer des Tutoriums oben rechts (groß!)

Übungspunkte

Zusammensetzung

- Punkte auf den Übungsblättern
- Vorrechnen einer Übungsblattaufgabe im Tutorium: 6 Übungspunkte
- Mittsemesterklausur am 03.06.2013: Ca. so viel Punkte wie zwei Übungsblätter

Zusatzpunkte

- Zusatzaufgaben auf Übungsblättern
- Programmierwettbewerb (?)
- Zählen nicht zu den 100 %

Klausur

Bonuspunkte für die Klausur

- $\geq$ 20 % der Übungspunkte: 1 Bonuspunkt
- $\geq$ 40 % der Übungspunkte: 2 Bonuspunkte
- $\geq$ 60 % der Übungspunkte: 3 Bonuspunkte
- $\geq$ 80 % der Übungspunkte: 4 Bonuspunkte

Klausurtermin

- Hauptklausur: 30.7.2013

Sinn und Zweck des Tutoriums

Das Tutorium ...

- ... bereitet auf die Übungsblätter vor
- ... bietet Möglichkeit zum Vorrechnen der Übungsblattaufgaben
- ... klärt Fragen
- ... bearbeitet Kreativaufgabe
- ... ist **KEINE** Ersatzvorlesung!

Bei Fragen und Unklarheiten ...

- Sofort "STOPP" rufen (oder melden) und nachfragen

Wiederholung $\mathcal{O}$-Kalkül

Klassische Definition

- $\mathcal{O}(f(n)) = \{g(n) \colon \exists_{c>0} \exists_{n_0 \in \mathbb{N}_+} \forall_{n \geq n_0} : g(n) \leq c \cdot f(n)\}$
- $o(f(n)) = \{g(n) \colon \forall_{c>0} \exists_{n_0 \in \mathbb{N}_+} \forall_{n \geq n_0} : g(n) \leq c \cdot f(n)\}$
- $\Omega(f(n)) = \{g(n) \colon \exists_{c>0} \exists_{n_0 \in \mathbb{N}_+} \forall_{n \geq n_0} : g(n) \geq c \cdot f(n)\}$
- $\omega(f(n)) = \{g(n) \colon \forall_{c>0} \exists_{n_0 \in \mathbb{N}_+} \forall_{n \geq n_0} : g(n) \geq c \cdot f(n)\}$
- $\Theta(f(n)) = \mathcal{O}(f(n)) \cap \Omega(f(n))$

Definition über Grenzwerte

- $o(f(n)) = \{g(n) \colon \limsup_{n \to \infty} \frac{g(n)}{f(n)} = 0\}$
- $\omega(f(n)) = \{g(n) \colon \limsup_{n \to \infty} \frac{g(n)}{f(n)} = \infty\}$

Aufgaben zur Laufzeitanalyse

Rechenaufgaben

- Gilt $4 \cdot n^2 + 5 = \mathcal{O}(n^2)$?
- Gilt $2^{n+1} = \Theta(2^n)$?
- Gilt $4n = \Omega(n)$?
- Gilt $2^{2n} = \mathcal{O}(2^n)$?
- Gilt $n! = \omega(2^n)$?
- Gilt $2 \cdot n! + 3n + 4 = o(n^n)$?

Aufgaben zur Laufzeitanalyse

Welchen asymptotischen Aufwand hat Bubblesort?

Function BubbleSort(integer array a[1 . . . n]) : int[]

```
1: for i := 0; i < n; i++ do
2:     for j := 0; j < n − i; j++ do
3:         if a[j] > a[j + 1] then
4:             temp := a[j]
5:             a[j] := a[j + 1]
6:             a[j + 1] := a[j]
```

- Worst-Case: $\Theta(2 \cdot n)$

Aufgaben zur Laufzeitanalyse

Welchen asymptotischen Aufwand hat Bubblesort?

Function BubbleSort(integer array a[1 . . . n]) : int[]

```
1: for i := 0; i < n; i++ do
2:     for j := 0; j < n − i; j++ do
3:         if a[j] > a[j + 1] then
4:             temp := a[j]
5:             a[j] := a[j + 1]
6:             a[j + 1] := a[j]
```

- Worst-Case: $\Theta(2 \cdot n)$

BULLSHIT!

Aufgaben zur Laufzeitanalyse

Welchen asymptotischen Aufwand hat Bubblesort?

Function BubbleSort(integer array a[1 . . . n]) : int[]

```
1: for i := 0; i < n; i++ do
2:     for j := 0; j < n − i; j++ do
3:         if a[j] > a[j + 1] then
4:             temp := a[j]
5:             a[j] := a[j + 1]
6:             a[j + 1] := a[j]
```

- Worst-Case: $\Theta(n^2)$

Aufgaben zur Laufzeitanalyse

Verständnisfrage

Warum ist die Aussage "Die Laufzeit des Algorithmus A ist mindestens in $\mathcal{O}(n^2)$" unbrauchbar?

Schleifeninvariante

Frage

Was ist eine Schleifeninvariante? Wie beweist man die Korrektheit von Schleifen mittels Schleifeninvarianten?

Schleifeninvariante

Frage

Was ist eine Schleifeninvariante? Wie beweist man die Korrektheit von Schleifen mittels Schleifeninvarianten?

Stichpunktartige Antwort

- Aussage, die vor und nach jedem Schleifendurchlauf gilt
- Beweis durch vollständige Induktion:
  - Invariante gilt nach der Initialisierung
  - Invariante bleibt bei einem Übergang zum nächsten Schleifendurchlauf erhalten
- Beweise zusätzlich, dass die Schleife terminiert

Kreativaufgabe 

Function ?????(sortedArray a[1..n], int x)

```
1: left := 1
2: right := n
3: while left <= right do
4:     middle := ⌊(right + left)/2⌋
5:     if x = a[middle] then return middle
6:     else if x < a[middle] then
7:         right := middle
8:     else left := middle
9: return −1
```

- Was tut dieser Algorithmus?

Kreativaufgabe 

Function BinarySearch(sortedArray a[1..n], int x)

```
1: left := 1
2: right := n
3: while left <= right do
4:     middle := ⌊(right + left)/2⌋
5:     if x = a[middle] then return middle
6:     else if x < a[middle] then
7:         right := middle
8:     else left := middle
9: return −1
```

- Was könnte eine Schleifeninvariante für die While-Schleife sein?

Kreativaufgabe 

Function BinarySearch(sortedArray a[1..n], int x)

```
1: left := 1
2: right := n
3: while left <= right do
4:     middle := ⌊(right + left)/2⌋
5:     if x = a[middle] then return middle
6:     else if x < a[middle] then
7:         right := middle
8:     else left := middle
9: return −1
```

- Welche Laufzeit hat BinarySearch im Θ-Kalkül?

Kreativaufgabe 

Heute etwas militärisch . . .

Sergeant Jackson ist Dirigent des berühmtesten Marschier-Musik-Corps der Welt. Mit seiner ausgeklügelten Marsch-Choreographie sorgt er jedes Jahr für großes Aufsehen und Bewunderung. Leider geht gerade bei einer großen Aufführung etwas schief. Bei einem Taktschlag sind sich einige Soldaten unsicher, ob sie sich um 90 Grad nach links oder rechts drehen sollen. Sie erinnern sich an den Leitsatz ihres Dirigenten: „Wenn ihr euch dreht und dem Nachbar ins Gesicht schaut habt ihr Mist gebaut! Aber keine Sorge! Wenn das passiert, dreht ihr euch beim nächsten Schlag in die andere Richtung um 180 Grad! Wenn ihr dann immer noch eurem Nachbarn ins Gesicht schaut, dann macht ihr das nochmal! Und so weiter, bis wieder Ruhe einkehrt!“ Die Soldaten folgen also der Anweisung ihres Sergeant und drehen sich und drehen sich wieder. Als die Soldaten nach einigen Drehungen ins Schwitzen geraten, fragen sie sich ob sie jemals zum Stillstand kommen.

Kommen die Musiker jemals zur Ruhe? Wenn ja, wie lange dauert es? War der Leitsatz des Sergeants sinnvoll?

Bis zum nächsten Mal! ☺