GBI Tutorium Nr. 2^5

Tutorium 10

Dominik Muth - dominik.muth@student.kit.edu | 9. Januar 2013

INSTITUT FÜR INFORMATIK

Outline/Gliederung

Überblick

Wiederholung - Quiz

- In Adjazenzmatrizen sind immer schneller als Adjazenzlisten, benötigen aber mehr Speicher.
- Hat ein Knoten x keine Ausgehenden Kanten, so steht in der Wegematrix in der x. Zeile nur 0en.
- Über Matrizenmultiplikation und addition lässt sich eine Wegematrix konstruieren.
- Die 2-Erreichbarkeitsrelation sagt uns nichts darüber aus, ob es einen Pfad der Länge 1 von einem Knoten zu einem andern gibt.

Wiederholung - Quiz

- In Adjazenzmatrizen sind immer schneller als Adjazenzlisten, benötigen aber mehr Speicher. *X*
- Hat ein Knoten x keine Ausgehenden Kanten, so steht in der Wegematrix in der x. Zeile nur 0en.
- Über Matrizenmultiplikation und addition lässt sich eine Wegematrix konstruieren.
- Die 2-Erreichbarkeitsrelation sagt uns nichts darüber aus, ob es einen Pfad der Länge 1 von einem Knoten zu einem andern gibt.

Wiederholung - Quiz

- In Adjazenzmatrizen sind immer schneller als Adjazenzlisten, benötigen aber mehr Speicher. *X*
- Hat ein Knoten x keine Ausgehenden Kanten, so steht in der Wegematrix in der x. Zeile nur 0en. *X*
- Über Matrizenmultiplikation und addition lässt sich eine Wegematrix konstruieren.
- Die 2-Erreichbarkeitsrelation sagt uns nichts darüber aus, ob es einen Pfad der Länge 1 von einem Knoten zu einem andern gibt.

Wiederholung - Quiz

- In Adjazenzmatrizen sind immer schneller als Adjazenzlisten, benötigen aber mehr Speicher. *X*
- Hat ein Knoten x keine Ausgehenden Kanten, so steht in der Wegematrix in der x. Zeile nur 0en. *X*
- Über Matrizenmultiplikation und addition lässt sich eine Wegematrix konstruieren. $\surd$
- Die 2-Erreichbarkeitsrelation sagt uns nichts darüber aus, ob es einen Pfad der Länge 1 von einem Knoten zu einem andern gibt. $\surd$

Wiederholung - Aufgaben

A^2

Gegeben sei A mit: $A = \begin{array}{c|c} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$

- Zeichnen Sie den Graphen
- Berechnen Sie A^2

Wiederholung - Aufgaben

Karlsruher Institut für Technologie

A^2

Gegeben sei A mit: $A = \begin{array}{c c} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$

- Zeichnen Sie den Graphen
- Berechnen Sie A^2

Wiederholung - Fragen

- Was ist eine Äquivalenzrelation?

Wiederholung - Fragen

- Was ist eine Äquivalenzrelation?
- Was ist eine symmetrische Relation?

Wiederholung - Fragen

- Was ist eine Äquivalenzrelation?
- Was ist eine symmetrische Relation?
- Was ist eine reflexive Relation?

Wiederholung - Fragen

- Was ist eine Äquivalenzrelation?
- Was ist eine symmetrische Relation?
- Was ist eine reflexive Relation?
- Was ist eine transitive Relation?

Überblick

Master Theorem

Wozu?

Laufzeitabschätzung von rekursiv definierten Funktionen

Grundaufbau

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

Erläuterung

- a = Anzahl der Unterprobleme in der Rekursion
- $\frac{1}{b}$ = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird
- $f(n)$ = Aufwand, welcher durch die Rekursion der Teilprobleme und Kombination der Teillösungen auftritt.

Wozu?

Laufzeitabschätzung von rekursiv definierten Funktionen

Grundaufbau

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

Erläuterung

- a = Anzahl der Unterprobleme in der Rekursion
- $\frac{1}{b}$ = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird
- $f(n)$ = Aufwand, welcher durch die Rekursion der Teilprobleme und Kombination der Teillösungen auftritt.

Master Theorem

Karlsruher Institut für Technologie

Wozu?

Laufzeitabschätzung von rekursiv definierten Funktionen

Grundaufbau

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

Erläuterung

- a = Anzahl der Unterprobleme in der Rekursion
- $\frac{1}{b}$ = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird
- $f(n)$ = Aufwand, welcher durch die Rekursion der Teilprobleme und Kombination der Teillösungen auftritt.

Master Theorem

Karlsruher Institut für Technologie

Wozu?

Laufzeitabschätzung von rekursiv definierten Funktionen

Grundaufbau

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

Erläuterung

- a = Anzahl der Unterprobleme in der Rekursion
- $\frac{1}{b}$ = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird
- $f(n)$ = Aufwand, welcher durch die Rekursion der Teilprobleme und Kombination der Teillösungen auftritt.

Master Theorem

Wozu?

Laufzeitabschätzung von rekursiv definierten Funktionen

Grundaufbau

$$T(n) = a \cdot T(\frac{n}{b}) + f(n)$$

Erläuterung

- a = Anzahl der Unterprobleme in der Rekursion
- $\frac{1}{b}$ = Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird
- $f(n)$ = Aufwand, welcher durch die Rekursion der Teilprobleme und Kombination der Teillösungen auftritt.

Master Theorem

Karlsruher Institut für Technologie

Wie funktionierts?

Man unterscheidet zwischen 3 Fällen:

1. Wenn $f(n) \in \mathcal{O}(n^{log_b a - \epsilon})$ mit $\epsilon > 0$, $\Rightarrow T(n) \in \Theta(n^{log_b a})$
2. Wenn $f(n) \in \Theta(n^{log_b a}$, $\Rightarrow T(n) \in \Theta(n^{log_b a} log n)$
3. Wenn $f(n) \in \Omega(n^{log_b a + \epsilon}$ mit $\epsilon > 0$,
   und wenn es ein c gibt, mit $0 < c < 1$,
   sodass für alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$,
   $\Rightarrow T(n) \in \Theta(f(n))$

Wie funktionierts?

Man unterscheidet zwischen 3 Fällen:

1. Wenn $f(n) \in \mathcal{O}(n^{log_b a - \epsilon})$ mit $\epsilon > 0$, $\Rightarrow T(n) \in \Theta(n^{log_b a})$
2. Wenn $f(n) \in \Theta(n^{log_b a}$, $\Rightarrow T(n) \in \Theta(n^{log_b a} logn)$
3. Wenn $f(n) \in \Omega(n^{log_b a + \epsilon}$ mit $\epsilon > 0$,
   und wenn es ein c gibt, mit $0 < c < 1$,
   sodass für alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$,
   $\Rightarrow T(n) \in \Theta(f(n))$

Master Theorem

Wie funktionierts?

Man unterscheidet zwischen 3 Fällen:

1. Wenn $f(n) \in \mathcal{O}(n^{log_b a - \epsilon})$ mit $\epsilon > 0$, $\Rightarrow T(n) \in \Theta(n^{log_b a})$
2. Wenn $f(n) \in \Theta(n^{log_b a}$, $\Rightarrow T(n) \in \Theta(n^{log_b a} logn)$
3. Wenn $f(n) \in \Omega(n^{log_b a + \epsilon}$ mit $\epsilon > 0$,
   und wenn es ein c gibt, mit $0 < c < 1$,
   sodass für alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$,
   $\Rightarrow T(n) \in \Theta(f(n))$

Master Theorem

Karlsruher Institut für Technologie

Wie funktionierts?

Man unterscheidet zwischen 3 Fällen:

1. Wenn $f(n) \in \mathcal{O}(n^{log_b a - \epsilon})$ mit $\epsilon > 0$, $\Rightarrow T(n) \in \Theta(n^{log_b a})$
2. Wenn $f(n) \in \Theta(n^{log_b a}$, $\Rightarrow T(n) \in \Theta(n^{log_b a} logn)$
3. Wenn $f(n) \in \Omega(n^{log_b a + \epsilon}$ mit $\epsilon > 0$,
   und wenn es ein c gibt, mit $0 < c < 1$,
   sodass für alle hinreichend großen n gilt: $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$,
   $\Rightarrow T(n) \in \Theta(f(n))$

Master Theorem

Karlsruher Institut für Technologie

Beispiele

$$49 \cdot T(\frac{n}{7}) + 3n + 5$$

$$49 \cdot T(\frac{n}{7}) + 3n^3 + 5$$

Überblick

Mealy-Automaten

Definition: Mealy-Automat

Der Mealy-Automat $A = (Z, z_0, X, f, Y, g)$ besteht aus

- der endlichen Zustandsmenge $\mathbf{Z}$,
- dem Startzustand $\mathbf{z_0}$,
- dem Eingabealphabet $\mathbf{X}$,
- der Zustandsübergangsfunktion $\mathbf{f} : \mathbf{Z} \times \mathbf{X} \to \mathbf{Z}$,
- einem Ausgabealphabet $\mathbf{Y}$ und
- der Ausgabefunktion $\mathbf{g} : \mathbf{Z} \times \mathbf{X} \to \mathbf{Y}^*$.

Mealy-Automaten

Interaktive Aufgabe

Einen Getränke**automaten** modellieren:

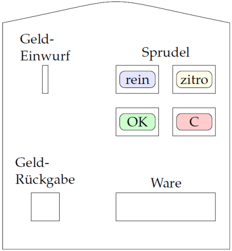
saurer Sprudel = *rein* (Eingabe/Ausgabe: R)

süßer Sprudel = *zitro* (Eingabe/Ausgabe: Z)

Abbrechen = *C* (Eingabe: C)

Bestätigen = *OK* (Eingabe: O)

Ein Getränk kostet 1 Euro (Eingabe: 1)



Mealy-Automaten

Karlsruher Institut für Technologie

f^* und f^{**}

f*: $f^*(z, w)$ kann im Gegensatz zu f ein ganzes Wort w, als zweites Funktionsargument nehmen, und gibt somit an, in welchem Zustand sich man sich befindet, nachdem man das Wort w abgearbeitet hat.

f**: $f^{**}(z, w)$ gibt die Durchlaufenen Zustände bei der Eingabe w an.

g^* und g^{**}

Simultan zu f^* und f^{**} geben die Funktionen $g^*(z, w)$ und $g^{**}(z, w)$ die Ausgabe nach dem eingegebenen Wort w an.

Mealy-Automaten

KIT Karlsruher Institut für Technologie

f^* und f^{**}

f*: $f^*(z, w)$ kann im Gegensatz zu f ein ganzes Wort w, als zweites Funktionsargument nehmen, und gibt somit an, in welchem Zustand sich man sich befindet, nachdem man das Wort w abgearbeitet hat.

f**: $f^{**}(z, w)$ gibt die Durchlaufenen Zustände bei der Eingabe w an.

g^* und g^{**}

Simultan zu f^* und f^{**} geben die Funktionen $g^*(z, w)$ und $g^{**}(z, w)$ die Ausgabe nach dem eingegebenen Wort w an.

Mealy-Automaten - Aufgaben

- Berechnen Sie $f^{**}((0, -), RZR11C)$ für den an der Tafel stehenden Automaten
- Berechnen Sie $g^{**}((0, -), RZR11O)$ für den an der Tafel stehenden Automaten

Überblick

Moore-Automaten - Aufgaben

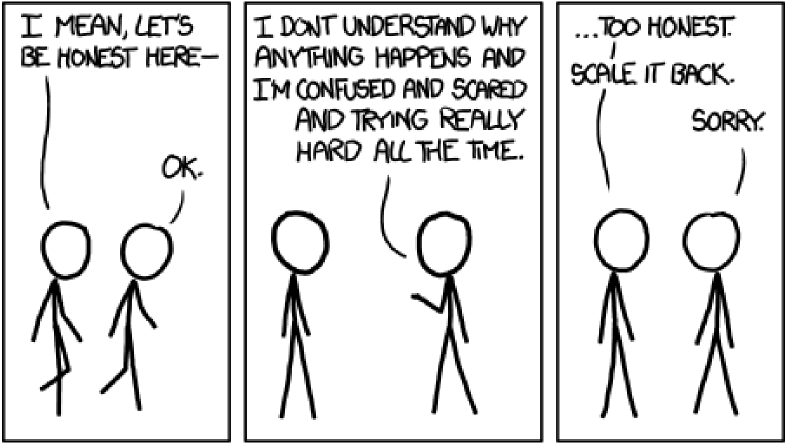
- Modellieren Sie einen Automaten, welcher Wörter mit gerader Länge akzeptiert, welche mindestens einmal aa und einmal bb enthalten.

Überblick

Fragen

- Fragen zum Stoff?
- Fragen zum nächsten Übungsblatt?
- Generelle Fragen?
- Feedback?

Frohe Weihnachten



source : http : //imgs.xkcd.com/comics/honest.png