Vorlesung Theoretische Informatik

Prof. Dr. Martin Schmollinger

Fakultät für Informatik
Masterstudiengang Wirtschaftsinformatik
überarbeitet von F. Laux (Stand: 22.03.2010)

Sommersemester 2010

Hochschule Reutlingen
Reutlingen University

Foliensatz - Inhaltsverzeichnis

Zur Vorlesung

Allgemeine Informationen

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS
- Modul Theoretische Grundlagen (WIM04)

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS
- Modul Theoretische Grundlagen (WIM04)
- Prüfung: Klausur (2-stündig)

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS
- Modul Theoretische Grundlagen (WIM04)
- Prüfung: Klausur (2-stündig)

Themengebiete

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS
- Modul Theoretische Grundlagen (WIM04)
- Prüfung: Klausur (2-stündig)

Themengebiete

- Grundlagen

Zur Vorlesung

Allgemeine Informationen

- Umfang 2 SWS / 3 ECTS
- Modul Theoretische Grundlagen (WIM04)
- Prüfung: Klausur (2-stündig)

Themengebiete

- Grundlagen
- Automaten und Formale Sprachen

Literatur

Boris Hollas

Grundkurs Theoretische Informatik, Elsvier Spektrum Akademischer Verlag, 1. Auflage, 2007

Uwe Schöning

Theoretische Informatik - kurzgefasst, Elsevier Spektrum Akademischer Verlag, 4. Auflage, 2001

John E. Hopcraft, Jeffrey D. Ullmann

Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Pearson Studium, 2. Auflage, 2003

Indirekter Beweis

Grundidee

- $A \longrightarrow B$ ist äquivalent zu $\neg B \longrightarrow \neg A$
- Wird verwendet, wenn $\neg B$ leichter formulierbar ist wie A

Beispiel

Aus a^2 gerade folgt a gerade.

Beweis durch Widerspruch

Grundidee

- Logischer Spezialfall des indirekten Beweises
- Aussage A wird bewiesen, indem gezeigt wird, dass die Annahme $\neg A$ zu einem Widerspruch führt
- Nützlich, wenn die Aussage $\neg A$ eine einfachere Form hat, als die Aussage A

Beispiel

Die Zahl $\sqrt{2}$ ist irrational!

Vollständige Induktion

Grundidee der Vollständigen Induktion (kurz Induktion)

- Beweis von Aussagen wie „für jedes Element $n \in \mathbb{N}$ gilt ...“ lassen sich oft durch Induktion beweisen
- Man zeigt, dass die Behauptung für $n = 1$ gilt
- Dann wird gezeigt: Wenn die Behauptung für n gilt, dann auch für $n + 1$

Vollständige Induktion

Ein Induktionsbeweis besteht demnach aus 2 Teilen

1. Induktionsanfang: Beweis für $n = 1$ (oder ein anderer für n geeigneter Wert)
2. Induktionsschritt $n \longrightarrow n + 1$

Die Prämisse des Beweises heißt **Induktionsvoraussetzung** oder -**hypothese** (Wenn die Behauptung für $n \in \mathbb{N}$ gilt ...)

Vollständige Induktion

Beispiel: Beweis durch Vollständige Induktion

Für alle $n \in \mathbb{N}$ gilt

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Vollständige Induktion

Formale Struktur

Sei $A(n)$ eine Aussage über die natürlichen Zahlen mit den Eigenschaften:

1. $A(1)$ gilt.
2. Wenn $A(n)$ gilt, dann gilt auch $A(n+1)$.

Dann gilt $A(n)$ für alle $n \in \mathbb{N}$.

Hinweise

- Man nimmt an, dass $A(n)$ wahr ist, um damit $A(n+1)$ zu zeigen.
- Achtung nicht verwechseln mit „Sei $A(n)$ wahr für alle n, dann ...“.

Vollständige Induktion

Strukturelle Induktion

- Verallgemeinerung der Vollständigen Induktion.
- Beweis von Behauptungen über induktiv definierte Mengen.

Induktive Definition einer Menge M

1. Definition von Elementen, die von keinen anderen Elementen von M abhängig sind.
2. Definition, wie mit Elementen von M weitere Elemente erzeugt werden können.

Beispiele: Menge der natürlichen Zahlen, Formeln der Aussagenlogik

Mengen und Mächtigkeit

Mächtigkeit (Kardinalität)

- Definition: Zwei Mengen A und B heißen gleich mächtig, wenn es eine Bijektion $f : A \longrightarrow B$ gibt
  Schreibweise: $|A| = |B|$.
- Die Mächtigkeit von Mengen ist eine Äquivalenzrelation. Die Äquivalenzklasse einer Menge wird auch als Kardinalität bezeichnet.

Beispiel

Sei $A := \{1, 2, 3\}$ und $B := \{a, b, c\}$.
Sei 3 die Äquivalenzklasse von A, also alle 3-elementigen Mengen, dann ist $3 = |A| = |B| = ...$

Beispiele für Mengen und ihre Mächtigkeit

Beispiele

1. $|\emptyset| = 0$
2. $|\{1, 2, 3\}| = 3$
3. $|\{x \in \mathbb{N}_{100} | \exists n \in N : x = n^2\}| = 10$
4. $|\{x \in \mathbb{N} | x \text{ gerade}\}| = \aleph_0$
5. $|\{\mathbb{Q}\}| = |\{x \in \mathbb{Q} | x < 10\}| = \aleph_0$

Diagonalisierung

Grundidee

Prinzip zum Beweis von Aussagen über die Abzählbarkeit von Mengen.

Definition **abzählbar** (unendlich)

Eine Menge M heißt *abzählbar*, wenn es eine Bijektion $f : \mathbb{N} \to M$ gibt. Das bedeutet, M läßt sich in der Form $M = \{f(1), f(2), f(3), \ldots\}$ darstellen.
$|M| = |\mathbb{N}| = \aleph_0$

Diagonalisierung

Definition **höchstens abzählbar**

Eine Menge N heißt *höchstens abzählbar*, wenn sie endlich oder abzählbar ist. In diesem Fall ist $N = \{g(1), g(2), g(3), \ldots\}$ für eine beliebige Funktion $g : \mathbb{N} \to N$

Definition **überabzählbar**

Eine Menge heißt *überabzählbar*, wenn sie nicht höchstens abzählbar ist.

Häufige Fehler

- Verwendung unbewiesener Annahmen
  - Es dürfen nur logische Folgerungen aus bereits Bekanntem oder Bewiesenem verwendet werden.
- Verwechseln von $\Rightarrow, \Leftarrow, \Leftrightarrow$
  - Bei logischen Äquivalenzen müssen beide Richtungen bewiesen werden.
  - Ähnliche Fehler: Beweis der Gleichheit von Mengen $(\subseteq, \supseteq, =)$.
- Zirkelschluss
  - Benutzung der zu beweisenden Behauptung im Beweis selbst.

Wie findet man einen Beweis?

- Es gibt kein Patentrezept!
- Auch Profis grübeln u.U. Wochen über Beweisen!
- Bei Aussagen über abzählbare Mengen hilft oft die Induktion.
- Beweisideen erhält man durch Spezialfälle oder die Vereinfachung der Annahme.
- Widersprüche können durch Angabe eines Gegenbeispiels gezeigt werden.
- Beweise müssen nicht konstruktiv sein.

Übung macht den Meister!

Beweisen oder Widerlegen Sie die folgenden Aussagen.

Beispiel 1:

Das Quadrat einer ungeraden natürlichen Zahl n ist ungerade.

Beispiel 2:

Ist die Wurzel aus einer geraden natürlichen Zahl n eine natürliche Zahl, so ist diese gerade.

Beispiel 3:

$\sum_{i=1}^{n}(2i-1) = n^2$

Aussagenlogik

Die Formeln der Aussagenlogik sind zweiwertig, d.h. jede Formel ist entweder wahr oder falsch.

Induktive Definition

Die Formeln der Aussagenlogik sind induktiv definiert:

- Die Konstanten 0 (falsch), 1 (wahr) und jede Aussagenvariable sind Formeln der Aussagenlogik (*Atomformeln*).
- Wenn F, G Formeln der Aussagenlogik sind, dann auch $F \wedge G$, $F \vee G$, $\neg F$.

Aussagenlogik

Eine Aussagenvariable kann man als eine Tatsache interpretieren, die entweder wahr oder falsch ist.

Beispiele

- Der SSV Reutlingen 05 ist Deutscher Meister!
- Wenn der SSV Deutscher Meister ist, spielt er in der Champions League!

Wie ergibt sich aus den Wahrheitswerten der Variablen der Wahrheitswert der Formeln?

Aussagenlogik

Definition Belegung

Eine Belegung einer Formel F der Aussagenlogik ist eine Zuordnung von Wahrheitswerten (wahr/falsch) zu den Variablen in F. Der Wahrheitswert der Formel ergibt sich aus den Wahrheitswerten ihrer Variablen und ist induktiv definiert.

Aussagenlogik

Definition Belegung

Eine Belegung einer Formel F der Aussagenlogik ist eine Zuordnung von Wahrheitswerten (wahr/falsch) zu den Variablen in F. Der Wahrheitswert der Formel ergibt sich aus den Wahrheitswerten ihrer Variablen und ist induktiv definiert.

Wertermittlung einer Formel

Aussagenlogik

Definition Belegung

Eine Belegung einer Formel F der Aussagenlogik ist eine Zuordnung von Wahrheitswerten (wahr/falsch) zu den Variablen in F. Der Wahrheitswert der Formel ergibt sich aus den Wahrheitswerten ihrer Variablen und ist induktiv definiert.

Wertermittlung einer Formel

- Eine Variable ist wahr, wenn sie mit wahr belegt ist. Die Konstante 1 ist wahr, 0 ist falsch.

Aussagenlogik

Definition Belegung

Eine Belegung einer Formel F der Aussagenlogik ist eine Zuordnung von Wahrheitswerten (wahr/falsch) zu den Variablen in F. Der Wahrheitswert der Formel ergibt sich aus den Wahrheitswerten ihrer Variablen und ist induktiv definiert.

Wertermittlung einer Formel

- Eine Variable ist wahr, wenn sie mit wahr belegt ist. Die Konstante 1 ist wahr, 0 ist falsch.
- Die Formel $F \wedge G$ ist wahr, wenn F wahr ist und G wahr ist. $F \vee G$ ist wahr, wenn F wahr ist oder G wahr ist. $\neg F$ ist wahr, wenn F falsch ist.

Aussagenlogik

erfüllbar

Eine Formel ist erfüllbar, wenn es eine Belegung gibt, so dass die Formel wahr wird.

Aussagenlogik

erfüllbar

Eine Formel ist erfüllbar, wenn es eine Belegung gibt, so dass die Formel wahr wird.

falsch, unerfüllbar

Eine Formel ist falsch oder unerfüllbar, wenn es keine Belegung gibt, so dass die Formel wahr wird.

Aussagenlogik

erfüllbar

Eine Formel ist erfüllbar, wenn es eine Belegung gibt, so dass die Formel wahr wird.

falsch, unerfüllbar

Eine Formel ist falsch oder unerfüllbar, wenn es keine Belegung gibt, so dass die Formel wahr wird.

Tautologie, gültig

Eine Formel heißt Tautologie oder gültig, wenn die Formel für alle Belegungen wahr wird.

Aussagenlogik

Beispiel 1: erfüllbar, falsch oder gültig?

$F_1 = \neg(x \wedge y)$

Aussagenlogik

Beispiel 1: erfüllbar, falsch oder gültig?

$F_1 = \neg(x \wedge y)$

Beispiel 2: erfüllbar, falsch oder gültig?

$F_2 = ((\neg x \wedge y) \vee (x \wedge \neg y)) \wedge \neg(x \vee y)$

Aussagenlogik

Beispiel 1: erfüllbar, falsch oder gültig?

$F_1 = \neg(x \wedge y)$

Beispiel 2: erfüllbar, falsch oder gültig?

$F_2 = ((\neg x \wedge y) \vee (x \wedge \neg y)) \wedge \neg(x \vee y)$

Beispiel 3: erfüllbar, falsch oder gültig?

$\neg F_2$

Aussagenlogik

Semantische Äquivalenz

Wir schreiben $F \equiv G$, wenn für jede Belegung gilt: F ist wahr, genau dann wenn G wahr ist.

Aussagenlogik

Semantische Äquivalenz

Wir schreiben $F \equiv G$, wenn für jede Belegung gilt: F ist wahr, genau dann wenn G wahr ist.

Logische Äquivalenz und Folgerung

Aussagenlogik

Semantische Äquivalenz

Wir schreiben $F \equiv G$, wenn für jede Belegung gilt: F ist wahr, genau dann wenn G wahr ist.

Logische Äquivalenz und Folgerung

- $F \rightarrow G \equiv \neg F \vee G$

Aussagenlogik

Semantische Äquivalenz

Wir schreiben $F \equiv G$, wenn für jede Belegung gilt: F ist wahr, genau dann wenn G wahr ist.

Logische Äquivalenz und Folgerung

- $F \rightarrow G \equiv \neg F \vee G$
- $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$

Aussagenlogik

Semantische Äquivalenz

Wir schreiben $F \equiv G$, wenn für jede Belegung gilt: F ist wahr, genau dann wenn G wahr ist.

Logische Äquivalenz und Folgerung

- $F \rightarrow G \equiv \neg F \vee G$
- $F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$

Semantische und Logische Äquivalenz

Es gilt $F \equiv G$ genau dann, wenn $F \leftrightarrow G$ eine Tautologie ist.

Aussagenlogik

Bezeichnung	Äquivalenz
De Morgan	$\neg(a \vee b) \equiv \neg a \wedge \neg b$ $\neg(a \wedge b) \equiv \neg a \vee \neg b$
Distributivität	$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$ $a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$
Kommutativität	$a \vee b \equiv b \vee a$ $a \wedge b \equiv b \wedge a$
Assoziativität	$(a \vee b) \vee c \equiv a \vee (b \vee c)$ $(a \wedge b) \wedge c \equiv a \wedge (b \wedge c)$

Aussagenlogik

Bezeichnung	Äquivalenz
Idempotenz	$a \vee a \equiv a$ $a \wedge a \equiv a$
Absorption	$a \vee (a \wedge b) \equiv a$ $a \wedge (a \vee b) \equiv a$
Doppelte Negation	$\neg\neg a \equiv a$
	$a \wedge b \equiv a$ falls b Tautologie $a \wedge b \equiv F$ falls b falsch $a \vee b \equiv W$ falls b Tautologie $a \vee b \equiv a$ falls b falsch

Aussagenlogik

Beispiel 1: Zeigen Sie die Äquivalenz der beiden Formeln

$(a \leftrightarrow b) \rightarrow c$ und $(a \wedge \neg b) \vee (b \wedge \neg a) \vee c$

Aussagenlogik

Beispiel 1: Zeigen Sie die Äquivalenz der beiden Formeln

$(a \leftrightarrow b) \rightarrow c$ und $(a \wedge \neg b) \vee (b \wedge \neg a) \vee c$

Beispiel 2: Zeigen Sie die Äquivalenz der beiden Formeln

$((a \rightarrow b) \leftrightarrow (a \wedge \neg b)) \vee (b \rightarrow c)$ und $c \vee \neg b$

O-Notation und Landau-Symbole

1. Kennzahlen für die Leistung eines Programmes sind die Laufzeit oder der maximal belegte Speicher.

O-Notation und Landau-Symbole

1. Kennzahlen für die Leistung eines Programmes sind die Laufzeit oder der maximal belegte Speicher.
2. Laufzeit und Speicherbedarf hängen aber von der verwendeten Hardware und Codierung ab

O-Notation und Landau-Symbole

1. Kennzahlen für die Leistung eines Programmes sind die Laufzeit oder der maximal belegte Speicher.
2. Laufzeit und Speicherbedarf hängen aber von der verwendeten Hardware und Codierung ab
3. Algorithmen sind abstrakt, weshalb diese Kennzahlen nicht sinnvoll angewendet werden können.

O-Notation und Landau-Symbole

1. Kennzahlen für die Leistung eines Programmes sind die Laufzeit oder der maximal belegte Speicher.
2. Laufzeit und Speicherbedarf hängen aber von der verwendeten Hardware und Codierung ab
3. Algorithmen sind abstrakt, weshalb diese Kennzahlen nicht sinnvoll angewendet werden können.
4. Wir brauchen ein Komplexitätsmaß für Algorithmen, das unabhängig von technischen Details Aussagen über die Leistungsfähigkeit macht.

O-Notation und Landau-Symbole

1. Kennzahlen für die Leistung eines Programmes sind die Laufzeit oder der maximal belegte Speicher.
2. Laufzeit und Speicherbedarf hängen aber von der verwendeten Hardware und Codierung ab
3. Algorithmen sind abstrakt, weshalb diese Kennzahlen nicht sinnvoll angewendet werden können.
4. Wir brauchen ein Komplexitätsmaß für Algorithmen, das unabhängig von technischen Details Aussagen über die Leistungsfähigkeit macht.
5. Mit der O-Notation lassen sich obere Schranken für das Laufzeitverhalten eines Algorithmus angeben.

O-Notation und Landau-Symbole

Definition O-Notation

Für eine Funktion $f : \mathbb{N} \longrightarrow [0, \infty)$ ist $O(f)$ die Menge aller Funktionen $g : \mathbb{N} \longrightarrow [0, \infty)$ mit $g(n) \leq cf(n)$ für eine Konstante $c > 0$ und alle hinreichend großen $n \in \mathbb{N}$.

Die Formulierung "hinreichend groß“ ermöglicht, dass endlich viele Werte, für die $g(n) \leq cf(n)$ **nicht** gilt, ausgeschlossen werden können.
Es handelt sich mathematisch um eine asymptotische Abschätzung

O-Notation und Landau-Symbole

Beispiel 1: $\frac{n(n+1)}{2} \in O(n^2)$

$\frac{n(n+1)}{2} \leq \frac{n^2+n^2}{2} = n^2$ für alle n.

Ebenso gilt $\frac{n(n+1)}{2} \in O(n^3)$ oder $\frac{n(n+1)}{2} \in O(\frac{1}{2}n^2)$, wir wollen aber immer eine möglichst **gute und einfache Abschätzung**!

Beispiel 2: $\frac{n+1}{n-1} \in O(1)$ für $n > 1$

$\frac{n+1}{n-1} \leq \frac{2+1}{2-1} = 3$ für alle $n > 2$.

O-Notation und Landau-Symbole

Beispiel 2: Suche in einer linearen Liste

Ein Algorithmus durchsuche eine Liste der Länge n nach der ersten Null. Dazu muss er höchstens n-mal auf die Liste zugreifen. Die Laufzeit liegt deshalb in $O(n)$

Mit der O-Notation wird immer der schlechteste Fall des Algorithmus abgeschätzt!

O-Notation und Landau-Symbole

Beispiel 2: Suche in einer linearen Liste

Ein Algorithmus durchsuche eine Liste der Länge n nach der ersten Null. Dazu muss er höchstens n-mal auf die Liste zugreifen. Die Laufzeit liegt deshalb in $O(n)$

Mit der O-Notation wird immer der schlechteste Fall des Algorithmus abgeschätzt!

Notationsvariationen

Oft wird die Menge $O(f)$ identifiziert mit einem Element dieser Menge. Deshalb kann man z.B. auch schreiben: $\frac{n(n+1)}{2} = O(n^2)$ und $\frac{n(n+1)}{2} = \frac{1}{2}n^2 + O(n)$.

O-Notation und Landau-Symbole

Rechenregeln der O-Notation

- $O(f) + O(g) = O(f + g)$
- $cO(f) = O(f)$
- $O(f)O(g) = O(fg)$

O-Notation und Landau-Symbole

Untere Schranken für Algorithmen können mit Hilfe der Ω-Notation angegeben werden.

Definition Ω-Notation

Für eine Funktion $f : \mathbb{N} \longrightarrow [0, \infty)$ ist $\Omega(f)$ die Menge aller Funktionen $g : \mathbb{N} \longrightarrow [0, \infty)$ mit $g(n) \geq cf(n)$ für eine Konstante $c > 0$ und alle hinreichend großen $n \in \mathbb{N}$.

O-Notation und Landau-Symbole

Die Θ(Theta)-Notation dient dazu, gleichzeitig eine obere und eine untere Schranke anzugeben. Damit läßt sich bis auf konstante Faktoren das genaue Wachstum einer Funktion angeben.

Definition Θ-Notation

$$\Theta(f) = O(f) \bigcap \Omega(f)$$

Beispiele

$1 + \sin^2(n) \in \Theta(1)$, $n + \sin(n) \in \Theta(n)$, und $n \sin(1/n) \in \Theta(1)$

O-Notation und Landau-Symbole

Schätzen Sie die folgenden Funktionen in $n \in \mathbb{N}$ mit der O-Notation ab!

1. $(n+1)^k$
2. $\log(n(n+1))$
3. $n^3|\sin(n)|$
4. $O(f)^2$

Grundlagen der Graphentheorie

Definition

Formal ist ein (*ungerichteter*) Graph G=(V,E) definiert durch

- die endliche Menge der Knoten V,
- der Menge der Kanten E, die aus ungeordneten Paaren {u,v} von Knoten bestehen mit $u \neq v$.

Graphen finden Verwendung in vielen Bereichen der Informatik (Z.B. Modellierung, Datenstrukturen).

Beispiel

Graph mit den Knoten 1 bis 6, bei dem alle Paare von teilerfremden Knoten durch eine Kante verbunden sind.

Grundlagen der Graphentheorie

Wie viele Kanten besitzt ein Graph mit n Knoten höchstens?

- Es gibt $\binom{n}{2}$ Möglichkeiten zwei Knoten durch eine Kante zu verbinden.
- $\binom{n}{2} = \frac{n(n-1)}{2}$

Probe

Graph mit $n = 4$ Knoten hat maximal $\binom{4}{2} = \frac{4(4-1)}{2} = 6$ Kanten!

Grundlagen der Graphentheorie

Gerichtete Graphen

Ist E eine Menge von (geordneten) Paaren (u, v), dann heißt der Graph *gerichtet*. Für gerichtete Graphen sind auch *Schlingen* (u, u) zugelassen.

- Automaten werden durch gerichtete Graphen beschrieben.
- Im Folgenden ist ein Graph immer ein ungerichteter Graph.

Grundlagen der Graphentheorie

Geometrische Interpretation der Knoten

Knoten werden auch Ecken (Vertices) genannt, weil man sie geometrisch interpretieren kann.

Beispiel: Graph K_4

Der Graph K_4 besitzt 4 Knoten. Jeder Knoten ist mit den drei anderen durch eine Kante verbunden.

Grundlagen der Graphentheorie

Weitere Definitionen:

Vollständigkeit

Ein Graph heißt vollständig, wenn alle Knoten paarweise verbunden sind.

Subgraph

Ein Subgraph eines Graphen $G_1 = (V_1, E_1)$ ist ein Graph $G2 = (V_2, E_2)$, der aus einer Teilmenge der Knoten $V_2 \subseteq V_1$ besteht zusammen mit den Kanten $E_2 = \{\{u, v\} \in E_1 | u, v \in V_2\}$.

Grundlagen der Graphentheorie

Weitere Definitionen:

Clique

Eine k-Clique ist ein vollständiger Subgraph, der k Knoten enthält.

Anticlique

Eine k-Anticlique ist eine Menge von k Knoten, zwischen denen es keine Kanten gibt.

Grundlagen der Graphentheorie

Wir betrachten nun Verbindungen zwischen Knoten durch Wege und Kreise.

Weg

Ein Weg (von v_1 nach v_k) ist eine Folge von Knoten $v_1, v_2, \ldots, v_k$ mit $\{v_1, v_2\} \in E$, $\{v_2, v_3\} \in E, \ldots, \{v_{k-1}, v_k\} \in E$.

Zusammenhängigkeit

Ein Graph heißt zusammenhängend, wenn es für **alle Paare** von Knoten u, v einen Weg von u nach v gibt.

Grundlagen der Graphentheorie

Kreis, Zyklus

Ein Weg $v_1, v_2, \ldots, v_k$ heißt **Kreis** oder **Zyklus**, wenn $v_1 = v_k$.

Hamilton-Pfad bzw. -Kreis

Ein Weg, der jeden Knoten genau einmal enthält, heißt **Hamilton-Pfad**. Ein Kreis, der jeden Knoten des Graphen genau einmal enthält, heißt **Hamilton-Kreis**.

Grundlagen der Graphentheorie

Euler-Kreis

Ein Euler-Kreis ist ein Kreis, der jede Kante des Graphen genau einmal enthält.

Ein hinreichendes und notwendiges Kriterium für die Existenz lässt sich mit Hilfe des Grades eines Knotens angeben.

Grundlagen der Graphentheorie

Grad eines Knotens

Ein Knoten v hat den Grad k, wenn v mit genau k anderen Knoten verbunden ist. Wir schreiben dafür $deg(v) = k$.

Satz von Euler

Ein zusammenhängener Graph enthält einen Euler-Kreis, wenn der Grad aller Knoten gerade ist.

Repräsentation von Graphen

Um Graphen in Programmen effizient einsetzen zu können, müssen geeignete Datenstrukturen entworfen werden

Definition Adjazenzmatrix

Eine Adjazenzmatrix ist eine Matrix (a_{uv}) mit

$$a_{uv} = \begin{cases} 1 \text{ für } \{u,v\} \in E \\ 0 \text{ sonst} \end{cases}$$

Repräsentation von Graphen

Adjazenzmatrix bei gerichteten Graphen

Gerichtete Graphen können auch durch eine Adjazenzmatrix repräsentiert werden.

Die Matrix ist dann durch die folgenden Eigenschaften charakterisiert:

- Im Allgemeinen ist die Matrix unsymmetrisch.
- Matrix kann Einträge auf der Hauptdiagonalen enthalten.

Repräsentation von Graphen

Adjazenzliste

Eine Adjazenzliste ist ein Feld, das an der Position $u \in V$ eine Liste aller Knoten $v \in V$ mit $\{u, v\} \in E$ enthält.

Bemerkung

Konkrete Implementierungen können an Stelle eines Feldes auch andere Datenstrukturen verwenden. Wichtig ist nur, dass der Zugriff auf eine Stelle des Feldes in $O(1)$ möglich ist.

Repräsentation von Graphen

Kennzahlen der Adjazenzmatrix und der Adjazenzliste

	A-Matrix	A-Liste
Speicheraufwand	$O(\|V\|^2)$	$O(\|V\|+\|E\|)$
Sind u und v verbunden?	$O(1)$	$O(\|V\|)$
Bestimme die Nachbarn von u	$O(\|V\|)$	$O(\deg(u))$

Wie können diese Zahlen interpretiert werden?

Durchlaufen von Graphen

Breitensuche

Die Suche beginnt bei einem Startknoten v_0. Zuerst werden die Nachbarn von v_0 besucht. Anschließend wird die Nachbarschaft dieser Knoten durchsucht, bis zuletzt alle Knoten besucht wurden.

Zur Verwaltung der auf den Besuch wartenden Knoten kann eine Warteschlange (Queue) verwendet werden. Die Anzahl der Knoten sei im Folgenden n.

Durchlaufen von Graphen

Pseudo-Code der Breitensuche

```
void bfs(int u0) {
   bool besucht[n];
   Queue<int> Q;
   int u,v;

   foreach(v aus der Menge V) besucht[v]=false;
   Q.Enter(u0); //Element stellt sich an der Schlange an.
   while(Q.empty()==false) {
      u=Q.Exit(); //nächstes Element wird entfernt
      foreach(v|{u,v} ist Kante)
         if(besucht(v)==false) {
            besucht[v]=true;
            Q.Enter(v); //Nächstes Element wird hinzugefügt
         }
    }
 }
```

Durchlaufen von Graphen

Übung

Wie kann der Algorithmus für Breitensuche erweitert werden, dass für jeden Knoten die kürzeste Distanz zum Startknoten berechnet wird? (Unter der Annahme, dass die Distanz für eine Kante mit 1 in die Rechnung ein geht.)

Durchlaufen von Graphen

Tiefensuche

Tiefensuche verfolgt Wege maximaler Länge. In jedem Schritt wird ein noch nicht besuchter Nachbar bestimmt. Als nächstes werden seine nicht besuchten Nachbarn besucht, noch bevor die anderen nicht besuchten Nachbarn auf gleicher Ebene besucht werden. Das wird wiederholt bis es keine nicht besuchten Nachbarn mehr gibt.

Wie kann der Algorithmus realisiert werden?

Durchlaufen von Graphen

Pseudo-Code der Tiefensuche (Rekursiv)

```
void tiefensuche(int u0) {
   int v;
   foreach(v aus der Menge V) besucht[v]=false;
   dfs(u0);
}
void dfs(int u) {
   int v;
   foreach(v|{u,v} ist Kante)
      if(besucht(v)==false) {
         besucht[v]=true;
         dfs(v); //Suche auf neues Element anwenden.
      }
 }
```

Durchlaufen von Graphen

Pseudo-Code der Tiefensuch (Iterativ)

```
void dfs(int u0) {
   bool besucht[n];
   Stack<int> S;
   int u,v;

   foreach(v aus der Menge V) besucht[v]=false;
   S.Push(u0); //Element wird auf den Stapel gelegt.
   while(S.empty()==false) {
      u=S.Pop();
      foreach(v|{u,v} ist Kante)
         if(besucht(v)==false) {
            besucht[v]=true;
            S.Push(v); //Neues Element auf Stapel legen
         }
    }
 }
```

Durchlaufen von Graphen

Übung

Welche Laufzeit haben die Algorithmen für Breiten- und Tiefensuche?

Bäume

Bäume haben viele besondere Eigenschaften, die sie zu den wichtigsten Graphen in der Informatik machen.

Definition Baum

Ein zusammenhängender Graph, der keine Kreise enthält, heißt Baum. In einem Baum heißen Knoten vom Grad 1 Blätter.

Definition DAG

Ein gerichteter Graph, der keine gerichteten Kreise enthält, heißt *gerichteter azyklischer Graph* (DAG).

Bäume

Verhältnis von $|V|$und $|E|$

Sei $B = (V, E)$ ein Baum. Dann gilt $|E| = |V| - 1$.

$\Rightarrow$ Beweis durch Induktion.

Bäume

Definition Wurzelbäume

Ein Baum mit einem als Wurzel besonders ausgezeichnetem Knoten heißt *Wurzelbaum*.

- Wurzelbäume wachsen von oben nach unten (Die Wurzel ist oben).
- Die Kanten sind von der Wurzel zu den Blättern gerichtet (Wird jedoch weggelassen).
- Vorgänger heißen *Vater*, Nachfolger *Söhne*.

Bäume

Weglänge in Binärbäumen

Sei B ein binärer Wurzelbaum mit mindestens 2^k Blättern. Dann enthält B einen Weg der Länge k.

$\Rightarrow$ Beweis durch Induktion.
Binäre Wurzelbäume werden oft verwendet um das Verhalten eines Algorithmus zu beschreiben, der in jedem Schritt aus zwei Alternativen eine Wahl trifft.

Bäume

Entscheidungsbaum

Bei Entscheidungsbäumen startet der Algorithmus an der Wurzel. An jedem weiteren inneren Knoten eine Entscheidung getroffen. Jedes Blatt repräsentiert eine Lösung.

Beispiel: Untere Schranke für Sortierverfahren

Sortierverfahren, die ausschließlich paarweise Vergleiche verwenden, können mit binären Wurzelbäumen ganz allgemein untersucht werden. Es kann eine untere Schranke von $\Omega(n \log n)$ ermittelt werden.