

Algorithmen und Datenstrukturen 1

Felix Ichtters, Lukas Dzielski*

Sommersemester 2023

Begleitmaterial zur Vorlesung 'Algorithmen und Datenstrukturen'.

*Universität Heidelberg

Inhaltsverzeichnis

1	Beweise	3
1.1	Statements	3
1.2	Formal mathematical proofs	3
1.3	Set definition rule: Beispiel	4
1.4	Macro-steps in proofs	4
1.5	Einfache Beweistechniken	5
1.6	\forall Aussagen beweisen	5
2	Einführung	7
2.1	Algorithmenanalyse	7
2.1.1	Asymptotische Algorithmenanalyse	7
3	Fogen, Felder und Listen	8
3.1	8
4	Hashing	9
4.1	9
5	Sortieren	10
5.1	10
6	Prioritätslisten	11
6.1	11
7	Sortierte Listen	12
7.1	12
8	Graphenrepräsentation	13
8.1	13
9	Graphtravesierung	14
9.1	14
10	Kürteste Wege	15
10.1	15
11	Minimale Spannbäume	16
11.1	16
12	Optimierung	17
12.1	17

1 Beweise

1.1 Statements

Ein **Statement/ Aussage** ist ein Mathematischer Ausdruck der entweder wahr oder Falsch ist.

Beispiel:

- $2 \in \{x \in \mathbb{R} | x < 5\}$ (wahr)
- $3^2 + 5^2 = 8^2$ (falsch)

Dabei werden Ausdrücke wie $0 < x < 1$ verwendet um Mengen zu definieren.

$$A = \{x \in \mathbb{R} | 0 < x < 1\}$$

Wichtig ist hierbei der **Wahrheitswert** eines offenen Ausdrucks $0 < x < 1$ hängt von gewähltem x ab. Also ist

- $x = 1/2$ (wahr)
- $x = 5$ (falsch)

Die **Domäne** ist hierfür wichtig zu beachten. Für \mathbb{N} , gibt es kein x s.t. $0 < x < 1$, aber es gibt welche für \mathbb{R}

1.2 Formal mathematical proofs

Ein **formaler mathematischer Beweis** besteht aus einer nummerierten Sequent von **wahren Aussagen**. Jede Aussage in einem Beweis ist ein **Annahme** oder **folgt** aus vorherigen Aussagen durch Ableitungsregel/ Inferenzregel (rule of inference). Die Letzte Aussage ist die die wir bewiesen haben.

\Rightarrow Offene Aussagen können in Beweisen nicht auftreten!

Beispiel einer Inferenzregel: set definition rule

Wenn ein Element in einer Menge ist, dann können wir definierende Eigenschaften ableiten. Andererseits, wenn es die definierende Eigenschaften erfüllt, dann können wir ableiten das das Element in der Menge ist.

1.3 Set definition rule: Beispiel

Definiere $C = \{x \in \mathbb{R} \mid x < 2\}$

($x < 2 \wedge x \in \mathbb{R}$ ist die definierende Eigenschaft). Dabei gibt es zwei Möglichkeiten für die Ableitung

- Möglichkeit 1
1. $a \in C$
 2. $a < 2 \wedge a \in \mathbb{R}(1; def C)$

- Möglichkeit 2
1. $b < 2 \wedge b \in \mathbb{R}$
 2. $b \in C(1; def C)$

Jede Aussage in dem Beweis hat eine Nummer. Wir begründen wie wir eine Aussage ableiten, zb $(1; def C)$ bedeutet wir leiten die aktuelle Aussage aus Aussage 1 mit der Definition von C und der set definition rule ab.

Bemerkung: $\wedge b \in R$

wird oft ausgelassen, wenn Kontext es zulässt.

1.4 Macro-steps in proofs

Problem:

Schauen wir uns folgenden Beweis an:

(ass = Annahme (assumption) und prop = Eigenschaft(property))

Beweis.

Annahme:

$$1. X = \{x \in \mathbb{R} \mid x < 1\}$$

$$2. a \in X$$

Zeige: $a < 2$

- | | | |
|----|-----------|-------------------|
| 1. | $a \in X$ | (ass 2) |
| 2. | $a < 1$ | (1, ass 1; def X) |
| 3. | $1 < 2$ | (prop R) |
| 4. | $a < 2$ | (2, 3; prop R) |

□

Ist das ein akzeptabler Beweis? Akzeptanz von Makro-Schritten wie "**prop** R" hängt von der Zielgruppe ab!

Welche Eigenschaft von \mathbb{R} wurde benutzt?

1.5 Einfache Beweistechniken

Beweis durch Beispiel

Beispiel: Zeigen Sie es gibt eine Primzahl zwischen 80 und 90.

Idee: Zeugen angeben für Primzahl (p) für die die Aussage gilt.

Beweis. Wähle $p = 83$

□

Ist das ausreichend?

Eigentlich, **NEIN**. Wie müssen noch **zeigen** das 83 tatsächlich eine Primzahl ist.

Das können wir tun indem wir alle Teiler ausprobieren.

Widerlegen von Behauptungen

Behauptung: Nimm an n ist eine Primzahl größer als 1. Dann ist $2^n - 1$ ebenfalls eine Primzahl.

Können Sie die Behauptung beweisen? Try hard ...

Wenn Sie es nicht können, dann sollen Sie darüber nachdenken die Behauptung zu widerlegen.

Eine Primzahl n für die $2^n - 1$ nicht prim ist, ist genug!

Das Gegenbeispiel ist $n = 11$ da 11 prim ist aber

$$2^{11} - 1 = 2047 = 23 \cdot 89$$

keine Primzahl ist!

1.6 \forall Aussagen beweisen

Inferenzregel für definierte Beziehungen

The definition rule

Angenommen, es wurde eine Beziehung definiert. Wenn die Beziehung gilt (in irgendeinem Beweisschritt oder Annahme), dann kann die definierende Eigenschaft abgeleitet werden. Andererseits, wenn die definierende Eigenschaft gilt, dann kann die Beziehung abgeleitet werden.

Beispiel: Für Mengen A und B , definiere A ist **Teilmenge** von B , $A \subseteq B$, wenn **für alle** x mit $x \in A : x \in B$. Mit anderen Worten:

$A \subseteq B$ if and only if (iff) $\forall x((x \in A) \rightarrow (x \in B))$ ist wahr.

Möglichkeit 1:

1. $A \subseteq B$ (ass 2)
2. für alle x s.t. $x \in A : x \in B$ (1, def \subseteq)

Möglichkeit 2:

1. für alle x s.t. $x \in A : x \in B$ (1, def \subseteq)
2. $A \subseteq B$ (ass 2)

Inferenzregel für \forall

Sei \mathfrak{P} eine Formel. Beispielsweise steht $\mathfrak{P}(x)$ für $x \in A$ und $\mathfrak{Q}(x)$ steht für $x \in B$. Dann kann "für alle x s.t. $x \in A : x \in B$ " als "für alle x s.t. $\mathfrak{P}(x) : \mathfrak{Q}(x)$ " geschrieben werden.

Regeln um \forall Aussagen zu beweisen (pr \forall)

Um Aussagen der Form "für alle x in s.t. $\mathfrak{P}(x) : \mathfrak{Q}(x)$ ", zu beweisen nimmt man an x sei **beliebig gewähltes Element (eigenvariable)** s.t $\mathfrak{P}(x)$ wahr ist. Dann zeige man $\mathfrak{Q}(x)$ ist wahr.

Generaliesierungen z.B. "für alle x, y s.t. $\mathfrak{P}(x, y) : \mathfrak{Q}(x, y)$ " möglich

Inferenzregel für \forall : Ein Beispiel

Sei $C = \{x \in \mathbb{R} | x < 1\}$ und $D = \{x \in \mathbb{R} | x < 2\}$. Zeige $C \subseteq D$!

Beweis.

Annahme:

1. $C = \{X \in \mathbb{R} | x < 1\}$
2. $D = \{x \in \mathbb{R} | x < 2\}$

Zeige: $C \subseteq D$

1. Sei $x \in C$ beliebig
2. $x < 1$ (1, ass 1; def C)
3. $x < 2$ (2, prop \mathbb{R})
4. $x \in D$ (3, ass 2; def D)
5. für alle $x \in X : x \in D$ (1 – 4; pr \forall)
6. $C \subseteq D$ (5, def \subseteq)

□

Wie können wir "für alle x s.t. $\mathfrak{P}(x) : \mathfrak{Q}(x)$ " wiederlegen?

2 Einführung

2.1 Algorithmenanalyse

2.1.1 Asymptotische Algorithmenanalyse

Zeitkomplexität Unter Zeitkomplexität eines Problems wird die Anzahl der Rechenschritte die ein Algorithmus zur Lösung des Problems benötigt, in Abhängigkeit der Länge der Eingabe.
...

3 Fogen, Felder und Listen

3.1 ...

...

4 Hashing

4.1 ...

...

5 Sortieren

5.1 ...

...

6 Prioritätslisten

6.1 ...

...

7 Sortierte Listen

7.1 ...

...

8 Graphenrepräsentation

8.1 ...

...

9 Graphtravesierung

9.1 ...

...

10 Kürteste Wege

10.1 ...

...

11 Minimale Spannbäume

11.1 ...

...

12 Optimierung

12.1 ...

...