

Diskrete Mathematik

Univ.-Prof. Dr. Goulmara ARZHANTSEVA

SS 2018



universität
wien

Überblick: Vorlesung

Einführung in die Grundbegriffe der Diskreten Mathematik

- 1 Einfache und abzählende Kombinatorik:
Stichproben, Permutationen, Partitionen
- 2 Erzeugende Funktionen, Lösen von Rekursionen
- 3 Das Prinzip der Inklusion und Exklusion, **Suchen und Sortieren**
- 4 Graphen und Netzwerke

Algorithmus

Ein **Algorithmus** ist ein standardisierter Ablauf. Mit ihm werden verschiedene Anweisungen in einer bestimmten, festgelegten Reihenfolge nacheinander durchgeführt.

Beispiel: PageRank

Suchmaschinen benutzen Algorithmen also, um aus vielen Parametern eine Rangierung einer Website zu berechnen. Zum Beispiel ist der **PageRank** von Google ein Algorithmus.

Algorithmus: Effizienzkriterium

Wir werden uns im folgenden nur mit dem “Effizienzkriterium Geschwindigkeit” befassen. Dabei kann man untersuchen,

- wie lange der Algorithmus im **schlechtest möglichen Fall** dauert (englisch: **Worst case analysis**),
- oder wie lange der Algorithmus im **Durchschnitt** dauert (englisch: **Average case analysis**).

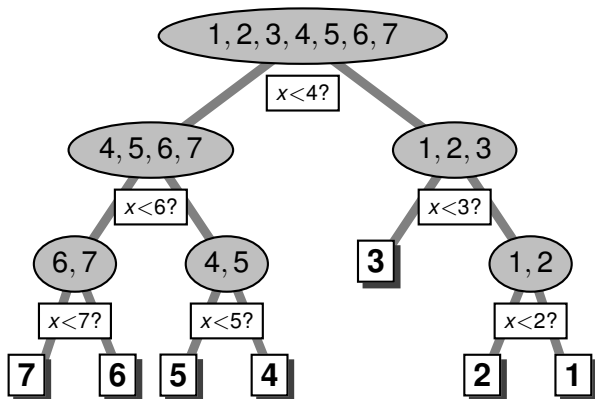
Algorithmus: Beispiel

Sei der Suchbereich $S = [7]$ gegeben.

Die Aufgabe lautet, eine unbekannte Zahl $x \in S$ durch (möglichst wenige) Fragen des Typs “ $x < i?$ ” für irgendein $i \in [7]$ zu erraten.

Einen Algorithmus zur Bestimmung der gesuchten Zahl x kann man sehr einfach durch einen **Entscheidungsbaum** beschreiben:

Beispiel: Entscheidungsbaum (aus Skriptum)



Im **schlechtesten Fall (Worst case)** führt dieser Algorithmus in 3 Schritten zum Ziel (im besten Fall braucht man 2 Schritte), und im **Durchschnitt (Average case)** braucht man $\frac{1 \cdot 2 + 6 \cdot 3}{7} = \frac{20}{7}$ Schritte.

Beispiel: Entscheidungsbaum

Unser Beispiel zeigt zweierlei:

- Ein (Such-)Algorithmus läßt sich zweckmäßig durch einen (Entscheidungs-)Baum beschreiben,
- Die Effizienz eines (Such-)Algorithmus kann man durch Analyse des entsprechenden (Entscheidungs-)Baumes ermitteln.

Wurzelbaum

Definition: Wurzelbaum

Ein **Wurzelbaum** ist ein Baum mit einem ausgezeichneten Knoten, der sogenannten **Wurzel**.

In einem Wurzelbaum gibt es eine **implizite Orientierung** der Kanten “von der Wurzel weg”, in folgendem Sinn:

Sei w die Wurzel und sei $e = \{v_1, v_2\}$ eine Kante. Wenn die Länge des Weges von w nach v_1 k ist, dann ist die Länge des Weges von w nach v_2 $k \pm 1$ — o.B.d.A. nehmen wir $k + 1$ an und orientieren die Kante dann: $\vec{e} := (v_1, v_2)$.

Wurzelbaum

Definition: Wurzelbaum

Ein **Wurzelbaum** ist ein Baum mit einem ausgezeichneten Knoten, der sogenannten **Wurzel**.

In einem Wurzelbaum gibt es eine **implizite Orientierung** der Kanten “von der Wurzel weg”, in folgendem Sinn:

Sei w die Wurzel und sei $e = \{v_1, v_2\}$ eine Kante. Wenn die Länge des Weges von w nach v_1 k ist, dann ist die Länge des Weges von w nach v_2 $k \pm 1$ — o.B.d.A. nehmen wir $k + 1$ an und orientieren die Kante dann: $\vec{e} := (v_1, v_2)$.

In einem Entscheidungsbaum ist die Wurzel der Knoten “ganz oben”, von dem aus sich alles verzweigt.

Wurzelbaum

Definition: inneren Knoten und Blätter

In einem Wurzelbaum (mit der impliziten Orientierung) unterscheiden wir dann die sogenannten

- **inneren Knoten**, die Ausgangsgrad > 0 haben,
- und die **Blätter** (manchmal auch **Endknoten** oder **äußere Knoten** genannt), die Ausgangsgrad $= 0$ haben.

Wurzelbaum

Definition: inneren Knoten und Blätter

In einem Wurzelbaum (mit der impliziten Orientierung) unterscheiden wir dann die sogenannten

- **inneren Knoten**, die Ausgangsgrad > 0 haben,
- und die **Blätter** (manchmal auch **Endknoten** oder **äußere Knoten** genannt), die Ausgangsgrad $= 0$ haben.

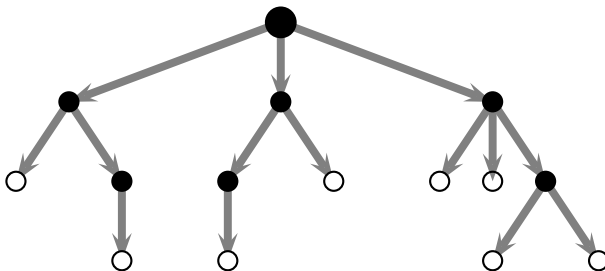
Die Wurzel kann selbst ein Blatt sein — wenn der Wurzelbaum nur aus einem einzigen Knoten besteht.

Beispiel: Wurzelbaum und implizite Orientierung (aus Skriptum)

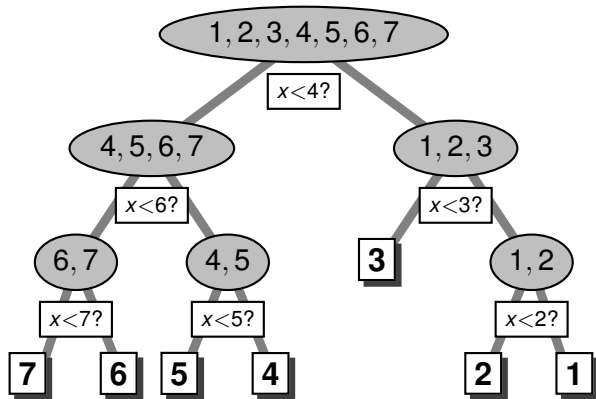
inneren Knoten = schwarze Punkte

die Blätter = weiße Kreis

die Wurzel = dick Punkt



Beispiel: Entscheidungsbaum (aus Skriptum)



Im Entscheidungsbaum eines Suchalgorithmus bedeutet

- ein **Blatt**, da der Algorithmus zu einem Ergebnis gekommen ist,
- und der maximale Ausgangsgrad eines inneren Knotens die größte Anzahl der Teile, in die der Suchraum nach einer Frage zerfallen kann.

Wurzelbaum: Länge, Niveau, q -Baum

Definition: Länge, Niveau, q -Baum

Sei \mathbf{W} ein Wurzelbaum mit Wurzel w , und sei v ein Knoten in \mathbf{W} .

Die Länge des (eindeutigen!) Weges von der Wurzel w zum Knoten v heißt **Länge** des Knotens und wird mit $\ell(v)$ bezeichnet. Die Menge der Knoten der Länge k nennen wir das **Niveau** k in \mathbf{W} .

Wenn der maximale Ausgangsgrad eines inneren Knotens von \mathbf{W} kleinergleich q ist, dann nennen wir \mathbf{W} einen **q -Baum**. \mathbf{W} heißt ein **vollständiger q -Baum**, wenn **jeder** innere Knoten von \mathbf{W} Ausgangsgrad q hat.

Ein innerer Knoten in einem q -Baum \mathbf{W} heißt **gesättigt**, wenn sein Ausgangsgrad gleich q ist.

Wurzelbaum: Länge, Niveau, q -Baum

Definition: Länge, Niveau, q -Baum

Sei \mathbf{W} ein Wurzelbaum mit Wurzel w , und sei v ein Knoten in \mathbf{W} .

Die Länge des (eindeutigen!) Weges von der Wurzel w zum Knoten v heißt **Länge** des Knotens und wird mit $\ell(v)$ bezeichnet. Die Menge der Knoten der Länge k nennen wir das **Niveau** k in \mathbf{W} .

Wenn der maximale Ausgangsgrad eines inneren Knotens von \mathbf{W} kleinergleich q ist, dann nennen wir \mathbf{W} einen **q -Baum**. \mathbf{W} heißt ein **vollständiger q -Baum**, wenn **jeder** innere Knoten von \mathbf{W} Ausgangsgrad q hat.

Ein innerer Knoten in einem q -Baum \mathbf{W} heißt **gesättigt**, wenn sein Ausgangsgrad gleich q ist.

Ein vollständiger q -Baum hat also nur gesättigte innere Knoten.

Vollständiger q -Baum

Lemma: vollständiger q -Baum

Sei $q \geq 2$, und sei \mathbf{W} ein vollständiger q -Baum mit genau n Blättern. Dann gilt die folgende Teilbarkeitsrelation:

$$(q - 1) \mid (n - 1) \tag{1}$$

Vollständiger q -Baum

Beweis: Induktion nach der Blattanzahl n : Für den (einzigen) q -Baum mit nur **einem** Blatt (das dann zugleich die Wurzel ist!) ist die Behauptung richtig.

¹ v kann als ein innerer Knoten maximaler Länge gewählt werden.

Vollständiger q -Baum

Beweis: Induktion nach der Blattanzahl n : Für den (einzigsten) q -Baum mit nur **einem** Blatt (das dann zugleich die Wurzel ist!) ist die Behauptung richtig.

In einem beliebigen vollständigen q -Baum mit $n > 1$ Blättern gibt es also einen inneren Knoten v , an dem q Blätter b_1, \dots, b_q "hängen"¹.

¹ v kann als ein innerer Knoten maximaler Länge gewählt werden.

Vollständiger q -Baum

Beweis: Induktion nach der Blattanzahl n : Für den (einzigen) q -Baum mit nur **einem** Blatt (das dann zugleich die Wurzel ist!) ist die Behauptung richtig.

In einem beliebigen vollständigen q -Baum mit $n > 1$ Blättern gibt es also einen inneren Knoten v , an dem q Blätter b_1, \dots, b_q “hängen”¹.

Wenn wir diese Blätter entfernen (also den Teilgraphen betrachten, der durch $V(\mathbf{W}) \setminus \{b_1, \dots, b_q\}$ induziert wird), dann hat der entstehende Wurzelbaum genau $n - q + 1$ Blätter (v ist ja nun zu einem Blatt geworden), und er ist wieder ein vollständiger q -Baum:

Nach Induktionsvoraussetzung gilt $(q - 1) \mid (n - q)$;
daraus folgt $(q - 1) \mid (n - 1)$. ■

¹ v kann als ein innerer Knoten maximaler Länge gewählt werden.

Allgemeine Suchproblem

Gegeben sei eine gewisse **Suchraum**, also eine Menge S von möglichen Ereignissen (im Beispiel oben war das $[7]$).

Weiters seien gewisse “Tests” gegeben, mit denen der Suchraum in Teilmengen partitioniert wird (im Beispiel waren das die Fragen “ $x < i?$ ”): Das allgemeine **Suchproblem** besteht darin, ein bestimmtes (aber zunächst unbekanntes) Element in $x \in S$ durch eine Kombination der verfügbaren Tests zu identifizieren.

Unter einem **Suchalgorithmus** verstehen wir eine derartige “Kombination der verfügbaren Tests”, die **in jedem Fall** (d.h., unabhängig vom Element x) zum Ziel führt.

Allgemeine Suchproblem

Es ist klar, daß wir einen Suchalgorithmus durch einen **Entscheidungsbaum** beschreiben können:

Das ist ein q -Baum, wobei q die größte Anzahl von Blöcken ist, in die ein Test den Suchraum partitioniert.

Worst–Case Analyse: Informationstheoretische Schranke

Die **Worst–Case Analyse** eines Algorithmus stellt fest, wie lange der Algorithmus im **schlechtesten Fall** braucht.

Für Suchalgorithmen ist das also die **maximale** Länge eines Blattes im entsprechenden Entscheidungsbaum \mathbf{W} ; wir nennen dies die **Länge des Wurzelbaumes \mathbf{W}** und bezeichnen sie mit $L(\mathbf{W})$:

$$L(\mathbf{W}) := \max_{b \text{ Blatt in } \mathbf{W}} \ell(b).$$

Worst-Case Analyse

Satz: Die Länge des Wurzelbaumes

Sei $q \geq 2$ und \mathbf{W} ein q -Baum mit n Blättern. Dann gilt

$$L(\mathbf{W}) \geq \lceil \log_q n \rceil,$$

wo $\log_q n$ der Logarithmus von n zur Basis q ist.

Worst-Case Analyse

Satz: Die Länge des Wurzelbaumes

Sei $q \geq 2$ und \mathbf{W} ein q -Baum mit n Blättern. Dann gilt

$$L(\mathbf{W}) \geq \lceil \log_q n \rceil,$$

wo $\log_q n$ der Logarithmus von n zur Basis q ist.

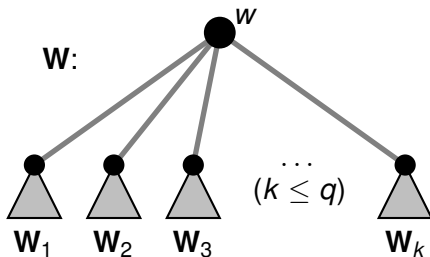
Beweis: Die Behauptung ist klarerweise äquivalent zu $L := L(\mathbf{W}) \geq \log_q n$ oder $q^L = q^{L(\mathbf{W})} \geq n$. Letzteres zeigen wir durch **Induktion nach L** .

Ist $L = 0$, dann besteht der Baum nur aus der Wurzel, die dann gleichzeitig das einzige Blatt ist, und wir haben in diesem Fall $q^0 \geq 1$.

Typischer q -Baum (aus Skriptum)

Für den **Induktionsschritt** bemerken wir, daß ein typischer q -Baum \mathbf{W} folgendermaßen aussieht:

Von der Wurzel w weg führen $k \leq q$ Kanten; an den anderen Enden dieser Kanten hängt ein Teilgraph \mathbf{W}_i , der wieder die Struktur eines q -Baumes hat (symbolisiert durch die Dreiecke im unten Bild).



Die Länge des Wurzelbaumes: Beweis

Wegen $L > 0$ befinden sich alle n Blätter in diesen Teilbäumen, einer dieser Teilbäume muß daher mindestens $n/k \geq n/q$ Blätter enthalten.

Bezeichnen wir diesen Teilbaum mit \mathbf{W}' . Klarerweise gilt $L(\mathbf{W}') \leq L(\mathbf{W}) - 1$, somit können wir auf \mathbf{W}' die Induktionsvoraussetzung anwenden:

$$q^{L(\mathbf{W}')} \geq \text{Anzahl der Blätter in } \mathbf{W}' \geq \frac{n}{q}.$$

Somit folgt:

$$q^{L(\mathbf{W})} \geq q^{L(\mathbf{W}')+1} = q^{L(\mathbf{W}')} \cdot q \geq \frac{n}{q} q = n.$$



Informationstheoretische Schranke

Der Wert $\lceil \log_q n \rceil$ heißt **informationstheoretische Schranke**.

Es ist klar, daß wir für unser zuvor beschriebenes “allgemeines Suchproblem” keinen Algorithmus angeben können, der **immer** (also auch im **worst case**) weniger Tests benötigt als $\lceil \log_q n \rceil$.

Informationstheoretische Schranke: Bemerkung

Als “Faustregel” für die Konstruktion eines Suchalgorithmus kann man offenbar ansehen: “Zerlege den Suchraum mit jedem Test in möglichst gleich große Teile”.

Es ist aber **keineswegs immer** möglich, einen Algorithmus zu konstruieren, der mit der theoretisch möglichen unteren Schranke für die Anzahl der Tests auskommt:

Typischerweise gelingt das nicht, wenn die Tests die Suchräume nicht “hinreichend gleichförmig” partitionieren können.

Beispiel: Binary search

Der klassische Algorithmus für das Einordnen eines neuen Elements x in eine bereits geordnete **Liste** $a_1 \leq a_2 \leq \dots \leq a_n$ ist **Binary Search**:

Der Suchraum ist hier die Menge der möglichen Stellen, wo x eingeordnet werden könnte, er umfaßt also $n + 1$ Elemente.

Der Algorithmus funktioniert so:

Binary search: Algorithmus

Sei $L = (a_1, a_2, \dots, a_n)$ eine der Größe nach geordnete Liste von n reellen Zahlen (entspricht einem Suchraum von $n + 1$ möglichen Stellen), bei der wir möglicherweise bereits die Relation $x \leq a_n$ kennen (entspricht einem Suchraum von nur mehr n möglichen Stellen).

Wenn der der Liste L entsprechende Suchraum nur mehr ein Element enthält, sind wir fertig; d.h.: Wenn $n = 0$ oder wenn $n = 1$ ist und wir die Relation $a_2 = a_{n+1} \leq a_n = a_1$ bereits kennen, dann schreiben wir a_{n+1} an die erste Stelle.

Binary search: Algorithmus

Ansonsten vergleichen wir a_{n+1} mit jenem Element x der Liste, das den Suchraum S möglichst gleichmäßig zerteilt (für $m = |S|$ ist $x = a_{\lceil m/2 \rceil}$).

Gilt $a_{n+1} > x$, dann setzen wir $L = (a_{\lceil m/2 \rceil + 1}, \dots, a_n)$; gilt $a_{n+1} \leq x$, dann setzen wir $L = (a_1, \dots, a_{\lceil m/2 \rceil})$ (hier kennen wir nun die Relation $x \leq a_{\lceil m/2 \rceil}$); in jedem Fall beginnen wir wieder von vorne.

Binary search: Informationstheoretische Schranke

Die informationstheoretische Schranke besagt, daß wir im worst case mindestens $\lceil \log_2(n+1) \rceil$ Tests “ $a_{n+1} > x?$ ” brauchen. Und mit Induktion sehen wir, daß der obige Algorithmus diese Schranke tatsächlich erreicht:

Für $n = 0$ brauchen wir $0 = \log_2(1)$ Tests.

Falls $n > 0$, brauchen wir einen ersten Test und haben dann eine Liste vor uns, deren entsprechender Suchraum maximal $\lceil (n+1)/2 \rceil$ Elemente enthält (denn der ursprüngliche Suchraum ist in zwei Blöcke $n+1 = \lceil (n+1)/2 \rceil + \lfloor (n+1)/2 \rfloor$ partitioniert worden).

Nach Induktion brauchen wir dafür $\lceil \log_2(\lceil (n+1)/2 \rceil) \rceil$ Tests.

Binary search: Informationstheoretische Schranke

Die Anzahl der benötigten Tests ist also tatsächlich

$$\begin{aligned}1 + \lceil \log_2 \underbrace{\lceil (n+1)/2 \rceil}_{= \frac{n+1}{2} + \frac{\lceil n \equiv 0 (2) \rceil}{2}} \rceil &= \left\lceil 1 + \log_2 \frac{n+1}{2} + \log_2 \left(1 + \frac{\lceil n \equiv 0 (2) \rceil}{n+1} \right) \right\rceil \\ &= \left\lceil \log_2 (n+1) + \log_2 \left(1 + \frac{\lceil n \equiv 0 (2) \rceil}{n+1} \right) \right\rceil \\ &= \lceil \log_2 (n+1 + \lceil n \equiv 0 (2) \rceil) \rceil \\ &= \lceil \log_2 (n+1) \rceil.\end{aligned}\tag{2}$$

Die letzte Gleichung folgt aus einer Betrachtung der Sprungstellen von $\lceil \log_2(x) \rceil$ bei 2^m , $m = 0, 1, \dots$.

$$n = 2^r - 2 \mapsto r, \quad n = 2^r - 1 \mapsto r, \quad n = 2^r \mapsto r + 1.$$