**Speicherzugriffsstrukturen und Balancierungstechniken**

Inhaltsverzeichnis

[Bäume Allgemein 3](#_Toc442213754)

[Balancierter Baum 3](#_Toc442213755)

[Binärbaum 4](#_Toc442213756)

[Partiell geordneter Baum 5](#_Toc442213757)

[Vollständig balancierter Binärbaum 5](#_Toc442213758)

[Suchbaum 6](#_Toc442213759)

[B-Baum 6](#_Toc442213760)

[2-3-4-Baum 7](#_Toc442213761)

[Binärer Suchbaum 8](#_Toc442213762)

[Gewichteter binärer Suchbaum 9](#_Toc442213763)

[Rot-Schwarz-Baum 9](#_Toc442213764)

[AVL-Baum 14](#_Toc442213765)

[Splay-Baum 18](#_Toc442213766)

[Heap 20](#_Toc442213767)

[Binärer Heap 20](#_Toc442213768)

[Binomial-Heap 22](#_Toc442213769)

[Fibonacci-Heap 23](#_Toc442213770)

[Hashverfahren 24](#_Toc442213771)

[Hashing mit Verkettung 24](#_Toc442213772)

[Hashing mit offener Adressierung 24](#_Toc442213773)

[Verfahren zur Ermittlung der Folge von Hash-Funktionen 24](#_Toc442213774)

[Lineares Sondieren 25](#_Toc442213775)

[Quadratisches Sondieren 25](#_Toc442213776)

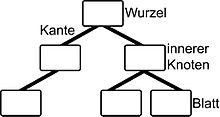
[Doppel-Hashing 25](#_Toc442213777)

[Brent-Hashing 25](#_Toc442213778)

[Kuckucks-Hashing 25](#_Toc442213779)

[Dynamisches Hashing 26](#_Toc442213780)

# Bäume Allgemein

In der Informatik ist ein Baum eine Datenstruktur und ein abstrakter Datentyp, mit dem sich hierarchische Strukturen abbilden lassen. Die durch die Hierarchie vorgegebenen Objekte nennt man Knoten. Typischerweise speichert jeder Knoten ausgehend von einem ersten Knoten, der Wurzel, eine Liste von Verweisen auf die ihm untergeordneten Knoten. Diese Verweise heißen Kanten. Es ist dann üblich, bei den untergeordneten Knoten von Kindern und bei dem verweisenden Knoten von einem Elternteil zu sprechen. Hat ein Knoten selbst keine Kinder, nennt man ihn ein Blatt.  
Typischerweise wird gefordert, dass in Bäumen bis auf die Wurzel, die keine Eltern hat, jeder Knoten nur genau ein Elternteil haben darf.

Die Tiefe eines Knotens gibt an, wie viele Kanten er von der Wurzel entfernt ist. Die Knoten mit derselben Tiefe bilden zusammen ein Niveau. Die Höhe eines Baumes ist dann die maximale Tiefe eines Knotens. In diesem Dokument wird aber auf die Definition zurückgegriffen, die besagt, dass die Höhe die maximale Tiefe + 1 ist. Somit hat ein Baum der nur aus einer Wurzel besteht bereits die Höhe 1, was viele Definitionen vereinfacht.  
Grad eines Knotens: Anzahl der Kanten, die diesen Knoten mit anderen Knoten verbinden.  
Eingangsgrad: Anzahl der eingehenden Kanten  
Ausgangsgrad: Anzahl der ausgehenden Kanten

# Balancierter Baum

Ein balancierter Baum) ist in der Informatik ein Spezialfall der Datenstruktur Baum, der eine maximale Höhe von c\*log(n) garantiert, wobei n die Anzahl der Elemente im Baum angibt und c eine von n unabhängige Konstante ist. Manche Autoren rechnen auch Datenstrukturen dazu, die Vorkehrungen enthalten, dass die mittlere Höhe oder Pfadlänge bei jedem Baum logarithmisch bleibt.

**Problem: Entartung**

Bäume können leicht und schnell entarten. Fügt man zum Beispiel einem Suchbaum eine große Menge bereits sortierter Daten ein, wächst dieser ungleichmäßig und resultiert im Extremfall in einer einfach verketteten Liste, was alle Vorteile eines Baumes zunichtemacht.

**Gegenstrategie: Balance halten**

Balancierte Bäume wurden entwickelt, um die Entartung zu verhindern und eine Höhe von c\*log(n) zu garantieren. Dazu verfolgt man unterschiedliche Konzepte aus denen folgt, dass die Höhe des Baumes in jedem Fall c\*log(n) ist und es geeignete Such-, Einfüge- und Löschoperationen gibt, die die speziellen Eigenschaften wahren.  
Man erhält eine solche Operation, indem man die Operation der allgemeinen Suchbäume verwendet und nach jeder Ausführung an der Stelle der Änderung die Balance überprüft, adjustiert und ggf. neu balanciert. Es kann vorkommen, dass sich diese Anpassungs- und Reparatur-Welle bis zur Wurzel hinauf fortsetzt.

Höhenbalance

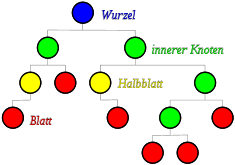
Nach der Höhe balancierte Bäume stellen für jeden Knoten sicher, dass die Höhe des linken Unterbaumes und die Höhe des rechten Unterbaumes nur um ein bestimmtes Verhältnis oder eine bestimmte Differenz voneinander abweichen.

Gewichtsbalance

Unter dem Gewicht eines Knotens wird die Wahrscheinlichkeit des Zugriffs auf ihn verstanden.  
Ist der Baum statisch, spielen also Einfüge- oder Löschoperationen keine Rolle, so bietet sich der Bellman-Algorithmus an, der einen optimalen gewichteten binären Suchbaum konstruiert. Seine Effizienz ist auch dann gegeben, wenn die Gewichte nur ungefähr bekannt sind.  
Sind Einfüge- oder Entfernoperationen wichtig, so sind im Prinzip auch die Gewichte zu pflegen. Im Grenzfall sogar beim Aufsuchen, da sich hierbei zumindest die Zugriffsstatistik ändert.

# Binärbaum

**Allgemein**

Binärbäume sind in der Informatik die am häufigsten verwendete Unterart der Bäume. Im Gegensatz zu anderen Arten von Bäumen können die Knoten eines Binärbaumes nur höchstens zwei Nachkommen haben. Meist wird verlangt, dass sich die Kindknoten eindeutig in linkes und rechtes Kind einteilen lassen.  
Ein Binärbaum ist entweder leer, oder er besteht aus einer Wurzel mit einem linken und rechten Teilbaum, die wiederum Binärbäume sind. Ist ein Teilbaum leer, bezeichnet man den entsprechenden Kindknoten als fehlend.

Ein Binärbaum ist **geordnet**, wenn jeder innere Knoten ein linkes und eventuell zusätzlich ein rechtes Kind besitzt (und nicht etwa nur ein rechtes Kind), sowie der linke Knoten „kleiner“, der rechte Knoten „größer“ als der Betrachtungsknoten ist.  
Man bezeichnet ihn als **voll**, wenn jeder Knoten entweder Blatt ist (also kein Kind besitzt), oder aber zwei (also sowohl ein linkes wie ein rechtes) Kinder besitzt.  
Man bezeichnet volle Binärbäume als **vollständig**, wenn alle Blätter die gleiche Tiefe haben.  
Er wird **entartet** genannt, wenn jeder Knoten entweder Blatt ist (Anzahl Kinder ist 0) oder Halbblatt (Anzahl Kinder ist 1). In diesem Fall stellt der Baum eine Liste dar. Besondere Formen sind die geordneten Listen, bei denen ein Baum jeweils nur aus linken oder nur aus rechten Kindern besteht.

**Traversierung**

Traversierung bezeichnet das systematische Untersuchen der Knoten des Baumes in einer bestimmten Reihenfolge. Es gibt verschiedene Möglichkeiten, die Knoten von Binärbäumen zu durchlaufen. Man unterscheidet die folgenden Varianten (dabei ist „Durchlaufen der Teilbäume“ l und r rekursiv):

* **pre-order (auch depth-first) oder Hauptreihenfolge (auch Tiefensuche) (N–l–r):**Zuerst wird die Wurzel N betrachtet und anschließend der linke l, schließlich der rechte Teilbaum r durchlaufen.
* **post-order oder Nebenreihenfolge (l–r–N):**Zuerst wird der linke l, dann der rechte Teilbaum r durchlaufen und schließlich die Wurzel N betrachtet.
* **in-order oder symmetrische Reihenfolge (l–N–r):**Zuerst wird der linke Teilbaum l durchlaufen, dann die Wurzel N betrachtet und schließlich der rechte Teilbaum r durchlaufen.
* **reverse in-order oder anti-symmetrische Reihenfolge (r–N–l):**Zuerst wird der rechte Teilbaum r durchlaufen, dann die Wurzel N betrachtet und schließlich der linke Teilbaum l durchlaufen.
* **level-order (auch breadth-first, deutsch Breitensuche):**Beginnend bei der Baumwurzel werden die Ebenen von links nach rechts durchlaufen.

**Einfügen**

Es sei angenommen, dass die Navigation zu einem Einfügepunkt bereits erfolgt ist. Zum Einfügen lässt man das Kind auf der geforderten Richtung des Knotens auf das neue Element verweisen, damit ist dieses korrekt eingefügt. Nach dem Einfügen ist das neue Element ein Blatt des Binärbaums.

**Löschen**

Beim Löschen muss man mehr Fälle unterscheiden. Wichtig ist z. B., wie viele Kinder der Knoten hat.  
**Fall A:** Zu löschender Knoten hat höchstens ein Kind.  
Ist der Knoten ein Blatt (Knoten ohne Kinder), dann wird beim Löschen einfach der Knoten entfernt. Hat der zu löschende Knoten genau ein Kind, wird dieses an die Stelle des zu löschenden Knotens gesetzt.

**Fall B:** Zu löschender Knoten hat zwei Kinder.  
In diesem Fall kann die Löschung sowohl über den linken wie über den rechten Teilbaum vollzogen werden. Um die in-order-Reihenfolge aufrechtzuerhalten, ist aber ein Abstieg bis zu einem Knoten mit einem Kind unvermeidlich.  
Auch eine Lösung: Diesen Knoten mit dem nächsthöheren Knoten (im rechten Teilbaum ganz links) oder nächstniedrigeren Knoten (im linken Teilbaum ganz rechts) austauschen.

# Suchbaum

In der Informatik ist ein Suchbaum eine abstrakte Datenstruktur, bei der die Menge von Elementen, in der gesucht werden soll, in einer Baumstruktur dargestellt wird. Diese Repräsentation unterstützt ein effizientes Suchen.  
Die charakteristische Operation ist das Suchen. Die meisten anderen Operationen, wie Einfügen, Löschen, Traversieren werden von der unterliegenden Baumstruktur geerbt.  
Die Suchoperation gibt ein Element mit übereinstimmendem Schlüssel zurück oder, falls der Schlüssel nicht vorkommt, das NULL-Element.

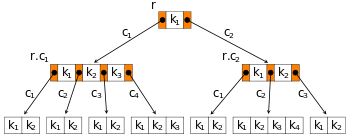
Der maximale Aufwand für das Suchen, also die Maximalzahl der erforderlichen Vergleiche, ist proportional zur Baumhöhe.

## B-Baum

Ein B-Baum ist ein immer vollständig balancierter Baum, der Daten nach Schlüsseln sortiert speichert. Er kann binär sein, ist aber im Allgemeinen kein Binärbaum. B-Bäume wachsen – und schrumpfen – anders als viele Suchbäume von den Blättern hin zur Wurzel.

In einem B-Baum kann ein Knoten mehr als 2 Kindknoten haben. Dies ermöglicht es, mit einer variablen Anzahl an Schlüsseln (oder Datenwerten) pro Knoten die Anzahl der bei einer Datensuche zu lesenden Knoten zu reduzieren. Die maximale erlaubte Anzahl der Schlüssel ist von dem Verzweigungsgrad t (oder Ordnung) des B-Baumes, abhängig. Die Bedeutung des Verzweigungsgrades ist je nach Definition unterschiedlich: Entweder bezeichnet er die maximale Anzahl von Kindknoten – in diesem Fall ist die maximal erlaubte Anzahl von Schlüsseln (t-1), oder die minimal erlaubte Anzahl von Kindknoten – in dem Fall wäre die maximal erlaubte Anzahl an Schlüsseln 2t-1.

Alle Blattknoten des B-Baumes befinden sich in gleicher Tiefe. Die Tiefe der Blattknoten ist gleich der Höhe h des Baumes.

Verbreitete Varianten des B-Baumes sind **B+-Bäume**, in denen die Daten nur in den Blättern gespeichert werden, und **B\*-Bäume**, die immer zu 2/3 gefüllt sind.

**Suchen:**

Die Suche nach einem Schlüssel k liefert denjenigen Knoten x, der diesen Schlüssel speichert, und die Position i innerhalb dieses Knotens. Enthält der Baum den Schlüssel k nicht, liefert die Suche das Ergebnis NULL.  
Die Suche läuft in folgenden Schritten ab:

1. **Die Suche beginnt mit dem Wurzelknoten r als aktuellem Knoten x**
2. **Ist x ein innerer Knoten,** wird die Position j des kleinsten Schlüssels bestimmt, der größer oder gleich k ist.   
   Existiert eine solche Position j, aber der Schlüssel an dieser Position nicht der gesuchte, kann der gesuchte Schlüssel nur im Unterbaum enthalten sein.   
   Ansonsten wurde der Schlüssel gefunden und (x, j) wird als Ergebnis zurückgeliefert.  
   Existiert keine solche Position, ist der Schlüssel größer als alle im aktuellen Knoten gespeicherten Schlüssel. In diesem Fall kann der gesuchte Schlüssel nur noch in dem Unterbaum enthalten sein, auf den der letzte Kindverweis zeigt.
3. **Ist x ein Blattknoten,** wird k in den Schlüsseln von x gesucht. Wenn der Schlüssel an Position j gefunden wird, ist das Ergebnis (x, j). Ansonsten NULL.

**Einfügen:**

Das Einfügen eines Schlüssels k in einen B-Baum geschieht immer in einem Blattknoten.

In einem vorbereitenden Schritt wird der Blattknoten gesucht, in den eingefügt werden muss. Dabei werden Vorkehrungen getroffen, damit die Einfügeoperation nicht die B-Baum-Bedingungen verletzt und einen Knoten erzeugt, der mehr als 2t-1 Schlüssel enthält. In einem abschließenden Schritt wird k unter Berücksichtigung der Sortierreihenfolge lokal in x eingefügt.

**Löschen:**

Das Löschen eines Schlüssels ist eine komplexere Operation als das Einfügen, da hier auch der Fall betrachtet werden muss, dass ein Schlüssel aus einem inneren Knoten gelöscht wird. Der Ablauf ist dabei wie die Suche nach einem geeigneten Platz zum Einfügen eines Schlüssels, allerdings mit dem Unterschied, dass vor dem Abstieg in einen Unterbaum überprüft wird, ob dieser genügend Schlüssel enthält, um eine eventuelle Löschoperation ohne Verletzung der B-Baum-Bedingungen durchführen zu können. Dieses Vorgehen ist analog zum Einfügen und vermeidet anschließende Reparaturmaßnahmen.   
Enthält der Unterbaum, den die Suche für den Abstieg ausgewählt hat, die minimale Anzahl von Schlüsseln (t-1), wird entweder eine Verschiebung oder eine Verschmelzung durchgeführt. Wird der gesuchte Schlüssel in einem Blattknoten gefunden, kann er dort direkt gelöscht werden. Wird er dagegen in einem inneren Knoten gefunden, passiert die Löschung wie in Löschen aus inneren Knoten beschrieben.

### 2-3-4-Baum

Ein 2-3-4-Baum ist ein B-Baum des Verzweigungsgrades 2, das heißt, er ist ein Baum, in dem jeder Knoten zwei, drei oder maximal vier Kinder besitzt und entsprechend ein, zwei oder maximal drei Datenelemente speichert, die nach dem gewählten Ordnungskriterium aufsteigend sortiert sind. Er stellt damit zugleich einen speziellen balancierten Suchbaum dar.

**Suchen:**

Um in einem 2-3-4-Baum zu suchen, wird ein einfacher Algorithmus angewendet. Beginnend beim kleinsten (linkesten) Element des Wurzelknotens:

1. **Vergleiche, ob der gesuchte Schlüssel gleich dem aktiven Element ist.**Wenn ja, Suche beendet.  
   Wenn nein, gehe zu 2.
2. **Vergleiche, ob der gesuchte Schlüssel kleiner ist als das aktive Element im aktiven Knoten.**Wenn ja, verzweige zum Kindknoten, der links vom gerade überprüften Element angehängt ist, setze dessen kleinstes Element als aktives Element und gehe zu 1. zurück.  
   Wenn nein, markiere das nächstgrößere Element im aktiven Knoten als aktives Element und gehe zu 1. zurück. Gibt es kein größeres Element mehr im aktiven Knoten, verzweige zum Kindknoten rechts des aktiven Element und setze dessen kleinstes Element als aktives Element und gehe zurück zu 1.

**Einfügen:**

* Ein Knoten wird mit Elementen aufgefüllt, bis er drei Elemente enthält.
* Wenn ein viertes Element aufgenommen werden soll, wird der Knoten gespalten in einen Knoten mit zwei Elementen, einen Knoten mit einem Element und ein mittleres Element, das in den Elternknoten aufgenommen wird.
* Ist der Elternknoten voll besetzt, wird das Element im Baum weiter nach oben gereicht. Erreicht das Element die Wurzel des Baumes und ist dieser schon mit drei Elementen besetzt, wird eine neue Wurzel nach gleicher Aufteilungsregel erzeugt.

**Löschen:**

Das Löschen eines beliebigen Elements kann immer auf das Löschen eines Elements in einem Blatt zurückgeführt werden. Dazu merkt man sich die Position des Elements innerhalb des Knotens. Dann wird im Unterbaum mit der Position i des Knotens das Blatt gesucht, das sich am weitesten rechts befindet, dort vertauscht man das größte Element mit dem zu löschenden Element. Nun braucht nur noch das Element aus dem Blatt gelöscht zu werden, wobei drei Fälle unterschieden werden müssen:

* **Das Blatt besitzt mehr als ein Element.** In diesem Fall kann das Element einfach entfernt werden.
* **Das Blatt enthält nur ein Element. In einem Nachbarknoten (Knoten mit gleichem Vorgänger) gibt es aber mindestens zwei Elemente.** Es kann ein Element vom Nachbarknoten ausgeliehen werden. Der Schlüssel wird in den Vorgängerknoten verschoben, wobei das Vorgängerelement des zu löschenden Elements an dessen Position verschoben wird und dieses ersetzt.
* **Das Blatt hat nur ein Element. Auch die Nachbarknoten haben nur ein Element.** Das Element wird entfernt und sein Vorgängerelement wird mit einem Nachbarelement zusammengelegt. Falls der Vorgängerknoten selbst nur ein einziges Element besitzt, wird dieselbe Operation auf höherer Ebene durchgeführt.