# B-Tree

## Allgemein

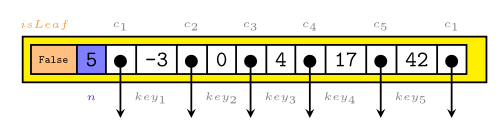
Ein **B-Baum** (englisch *B-tree*) ist in der Informatik eine Daten- oder Indexstruktur, die häufig in Datenbanken und Dateisystemen eingesetzt wird. Ein B-Baum ist ein immer vollständig balancierter Baum, der Daten nach Schlüsseln sortiert speichert. Er kann binär sein, ist aber im Allgemeinen kein Binärbaum. Das Einfügen, Suchen und Löschen von Daten in B-Bäumen ist in amortisiert logarithmischer Zeit möglich. B-Bäume wachsen und schrumpfen, anders als viele Suchbäume, von den Blättern hin zur Wurzel.

## Definition

Für einen **B-Baum der Ordnung t**, t∈N∖ {1}t∈N∖ {1}, gilt:

1. Jeder Knoten xx hat die folgenden Attribute:
   1. nn, die Anzahl der Schlüssel die im Knoten xx gespeichert wird,
   2. die nn Schlüssel, die in aufsteigender Reihenfolge gespeichert werden (also key1≤key2≤...≤keynkey1≤key2≤...≤keyn),
   3. isLeafisLeaf, ein boolscher Wert der True ist, falls xx ein Blatt ist und False ist, falls xx ein innerer Knoten ist
2. Jeder innere Knoten hat n+1n+1 Zeiger c1, c2, ...cn+1c1, c2, ...cn+1 auf seine Kinder. Blattknoten haben keine Kinder, also sind ihre cici-Attribute undefiniert.

Ein Knoten eines B-Baumes sieht also so aus:

[](https://martin-thoma.com/images/2012/07/b-tree-node.png)

Die Schlüssel keyikeyi setzen grenzen für die Werte der Schlüssel, die in den einzelnen Subbäumen gespeichert sind.  
Falls kiki ein Schlüssel im Subbaum mit der Wurzel cici ist, dann gilt: k1≤key1≤k2≤key2≤...≤keyn≤kn≤keyn+1k1≤key1≤k2≤key2≤...≤keyn≤kn≤keyn+1

1. Alle Blätter haben die gleiche Tiefe.
2. Für die Anzahl der Schlüssel eines Knotens gilt:
   1. Jeder Knoten (bis auf die Wurzel) hat mindestens t-1 Schlüssel.
   2. Jeder Knoten hat höchstens 2t−12t−1 Schlüssel

## **Stolperfallen**[**¶**](https://martin-thoma.com/b-baume/#stolperfallen)

Ein Schlüssel ist etwas anderes als ein Zeiger!

## **Folgerungen**[**¶**](https://martin-thoma.com/b-baume/#folgerungen)

* (2)⇒(2)⇒ Die Wurzel hat min. 2 Kinder, falls der Baum nicht leer ist.
* (2),(5.1)⇒(2),(5.1)⇒ Jeder innere Knoten (bis auf die Wurzel) hat min. t Kinder.
* (2),(5.2)⇒(2),(5.2)⇒ Jeder innere Knoten hat maximal 2t Kinder.
* Wenn man an den Zeigern die Kinder in die Elterknoten zieht, sodass am Ende alle Knoten in der Wurzel sind, entsteht wegen (3) eine sortierte Liste.
* h≤logtn+12h≤logt⁡n+12 (Cormen, S. 489)
* Ein B-Baum hat in der Tiefe h min. 2th−12th−1 Knoten.

## **Besondere B-Bäume**[**¶**](https://martin-thoma.com/b-baume/#besondere-b-baume)

Ein B-Baum der Ordnung t = 2 wird auch [2-3-4-Baum](http://de.wikipedia.org/wiki/2-3-4-Baum) genannt, da jeder Knoten entweder 2, 3 oder 4 Kinder hat.

Ich habe ja ausgeschlossen, dass es einen B-Baum der Ordnung t = 1 gibt. Warum eigentlich? Aus (5.1) folgt: In einem B-Baum der Ordnung t = 1 müssten einzelne Knoten keine Schlüssel haben. Das ist nicht sinnvoll. Also muss t≥2t≥2 gelten.

## **Suchen eines Schlüssels**[**¶**](https://martin-thoma.com/b-baume/#suchen-eines-schlussels)

Das Suchen eines Schlüssels funktioniert so:

SEARCH-KEY(node, key):

int i = 0

while i < node.n and node.key[i].key < key:

i += 1

if node.key[i].key == key: # Schl&uuml;ssel ist gesuchter Schl&uuml;ssel

return node.key[i].satelittendaten

else if node.isLeaf: # Erfolglose Suche

return NIL

else: # Rekursiv weitersuchen

DISK-READ(node.c[i])

return SEARCH-KEY(node.c[i], key)

Es wird also zuerst der Knoten durchsucht und dann gegebenenfalls der passende Subbaum.

Laut Vorlesung (Folie 97) gilt:

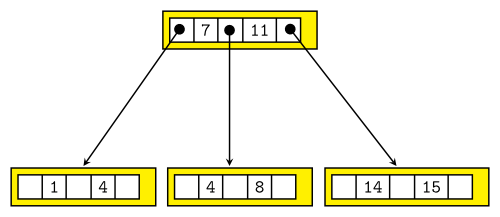
* Anzahl Zugriffe auf Hintergrundspeicher maximal: O(logt(n))O(logt⁡(n))
* Innerhalb eines Knotens: O(t)O(t)
* Insgesamt also: O(t⋅logt(n))O(t⋅logt⁡(n))

## **Einfügen von Schlüsseln**[**¶**](https://martin-thoma.com/b-baume/#einfugen-von-schlusseln)

Wenn ein Schlüssel in einen B-Baum eingefügt werden soll, dann muss man insebesondere auf Regel (5.1) und (5.2) achten: Jeder Knoten enthält n Schlüssel, mit t−1≤n≤2t−1t−1≤n≤2t−1.

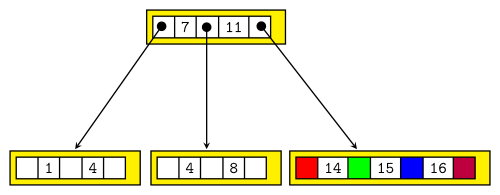
Die Idee ist, dass man das Blatt sucht, in dem der Schlüssel sein müsste. Falls noch Platz ist, kann man den Schlüssel einfach einfügen. Falls nicht, muss man das Blatt aufsplitten.

### **Beispiel zu Fall 1: Es ist noch Platz**[**¶**](https://martin-thoma.com/b-baume/#beispiel-zu-fall-1-es-ist-noch-platz)

[](https://martin-thoma.com/images/2012/07/b-tree-2-small.png)Abb. 2: B-Baum der Ordnung t = 2

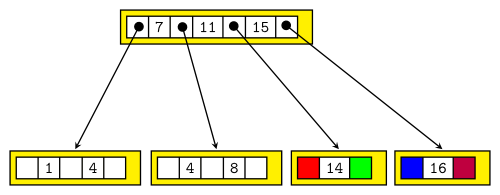
In den B-Baum aus Abb. 2 soll nun der Schlüssel 16 eingefügt werden. In einem B-Baum der Ordnung 2 hat jeder Knoten mindestens einen und höchstens 3 Schlüssel. Egal wo wir also landen würden, es würde noch in diesen Baum passen. Wir landen aber im Knoten rechts unten, da 11<1611<16 ist.

Das Ergebnis ist also:

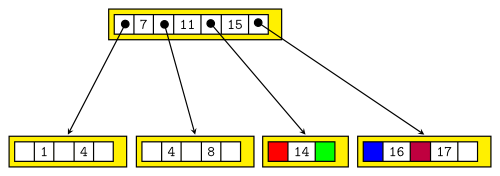
[](https://martin-thoma.com/images/2012/07/b-tree-2-small-2.png)Abb. 3: B-Baum der Ordnung t = 2

### **Beispiel zu Fall 2: Knotenüberlauf**[**¶**](https://martin-thoma.com/b-baume/#beispiel-zu-fall-2-knotenuberlauf)

Will man nun in den B-Baum der Ordnung 2 aus Abb. 3 den Schlüssel 17 hinzufügen, so gibt es einen "Knotenüberlauf". Der Schlüssel müsste in den Knoten rechts unten. Damit hätte dieser 4 Schlüssel, er darf aber nur 3 haben. Also splitten wir zuerst den Knoten. Schlüssel 15 wandert zu dem Elternknoten hoch, die beiden einzelnen Schlüssel bilden eigene Knoten. Damit man sieht, was mit den Schlüsseln geschehen würde, wenn der Baum größer wäre, habe ich diese mal eingefärbt:

[](https://martin-thoma.com/images/2012/07/b-tree-2-small-3.png)Abb. 4: B-Baum der Ordnung t = 2

Nun ist man beim Einfügen von 17 wieder in Fall 1. Das Ergebnis sieht so aus:

[](https://martin-thoma.com/images/2012/07/b-tree-2-small-4.png)Abb. 5: B-Baum der Ordnung t = 2

Natürlich kann es auch passieren, dass beim hochwandern des mittleren Knoten (15 von Abb. 3 nach Abb. 4) der Elternknoten überläuft. Dann muss halt auch dieser gesplittet werden. Wenn die Wurzel überläuft, muss eine neue Wurzel erstellt werden. Dann kann die alte Wurzel gesplittet werden.

Die Laufzeit des Einfügens ist in O(t⋅logt(n))O(t⋅logt⁡(n)).

## **Löschen eines Schlüssels**[**¶**](https://martin-thoma.com/b-baume/#loschen-eines-schlussels_1)

Falls sich der Schlüssel in einem Blatt befindet, kann man ihn einfach löschen. Allerdings muss man darauf achten, dass mindestens t-1 Schlüssel im Knoten verbleiben.

Ist der Schlüssel in einem inneren Knoten ist das ganze schwerer.

### **Fall 1: Schlüssel in Blatt**[**¶**](https://martin-thoma.com/b-baume/#fall-1-schlussel-in-blatt)

Der Einfachste Fall ist der [1. Fall des Einfügens](https://martin-thoma.com/b-baume/#Beispiel_zu_Fall_1_Es_ist_noch_Platz), nur umgekehrt. Also aus dem B-Baum aus Abb. 3 die 16 entfernen. Dann entsteht der B-Baum aus Abb. 2.

### **Fall 2 - 3**[**¶**](https://martin-thoma.com/b-baume/#fall-2-3)

Für die anderen Fälle habe ich leider kein kleines Beispiel und will deshalb auf die Erklärung verzichten.

Falls ihr da Hilfe braucht: Cormen, dritte Ausgabe, S. 499ff war sehr hilfreich. ([Link](http://shafaetsplanet.com/uploads/pdf/Introduction%20to%20Algorithms.pdf) - Warum auch immer der eine PDF-Datei vom Buch hat. Die KIT-Bibliothek hat leider keine Online-Version.)

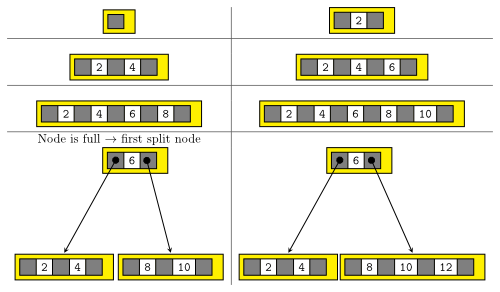
## **Trivia**[**¶**](https://martin-thoma.com/b-baume/#trivia_1)

[...] standard B-trees had numerous characteristics that were at odds with the ext2 design philosophy of simplicity and robustness. For example, XFS's B-tree implementation was larger than all of ext2 or ext3's source files combined.

Source: [ext2.sourceforge.net](http://ext2.sourceforge.net/2005-ols/paper-html/node3.html)

## **Beispiel**[**¶**](https://martin-thoma.com/b-baume/#beispiel)

Fügt man die Schlüssel 2, 4, 6, 8, 10, 12 in einen anfangs leeren B-Baum ein, entwickelt sich dieser wie folgt:

[](https://martin-thoma.com/images/2012/07/b-tree-3-evolution.png)Entwicklung eines B-Baumes der Ordnung t = 3