Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Anna Beer, Florian Richter

München, 05.06.2018

Algorithmen und Datenstrukturen

SS 2018

Übungsblatt 9: Suchen

Tutorien: 12.06-18.06.2018

Aufgabe 9-1 *B-Bäume*

Gegeben sei ein Array mit folgenden Werten: A = [42, 16, 89, 49, 35, 45, 8]

(a) Fügen sie die Werte des Array A in gegebener Reihenfolge in einen leeren B-Baum mit k = 1 ein.

(b) Überlegen die sich einen Algorithmus zum Einsortieren in einen B+-Baum und ordnen sie das Array A in einen leeren B+-Baum mit k = 1 ein.

(c) Löschen sie die folgenden Werte aus dem B-Baum aus Aufgabenteil a): 16, 49, 89

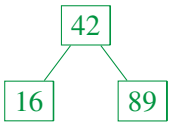
Lösungsvorschlag:

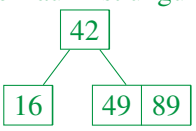
a) k= 1. Das heißt es dürfen maximal 2 Elemente in jedem Knoten sein.

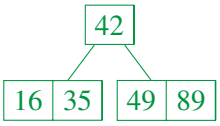
Einfügen der 16 

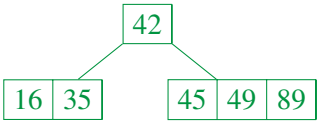
Einfügen der 42 

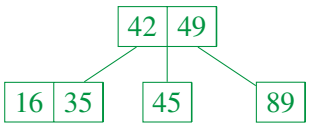
Einfügen der 89 

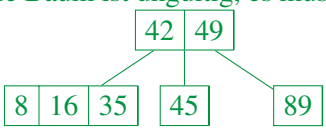
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

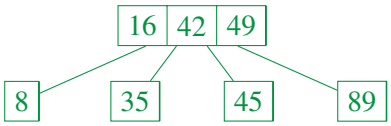
Einfügen der 49 

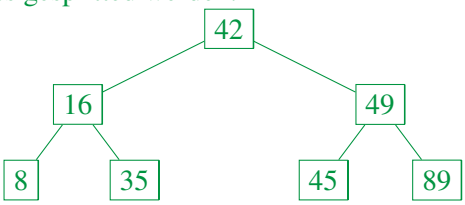
Einfügen der 35 

Einfügen der 45 

Der resultierende Baum ist ungültig, es muss gesplitted werden: 

Einfügen der 8. 

Der resultierende Baum ist ungültig, es muss gesplitted werden: 

Es muss ein weiters mal gesplitted werden: 

Lösungsvorschlag:

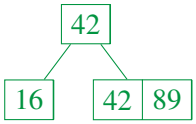
b) Möglicher Algorithmus:

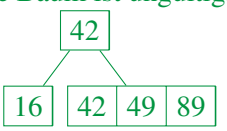
1. Einfügen in den Blättern wie bei einem B-Baum
2. Falls ein Blatt $(x_1, ..., x_{2k+1})$ aufgeteilt werden muss, teile es in zwei gleich große Knoten auf: $(x_1, ..., x_k)$ und $(x_{k+1}, ..., x_{2k+1})$. Erstelle im Elternknoten einen neuen Eintrag mit dem Wert x_k und verweise links und rechts von x_k auf die neu erstellten Blätter. Beachte, dass der im Elternknoten erstellte Schlüsselwert, eine **Kopie** des Schlüssels aus dem Kindknoten ist.
3. Muss ein innere Knoten aufgeteilt werden, verwende die gleiche Vorgehensweise wie bei einem B-Baum.

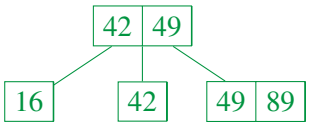
Einfügen der 16: 

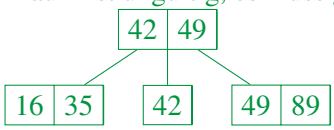
Einfügen der 42 

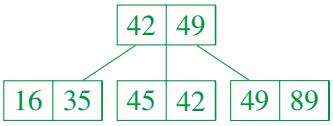
Einfügen der 89. 

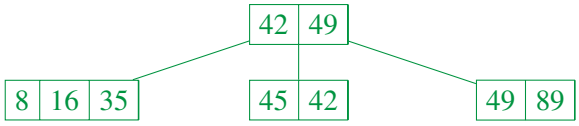
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

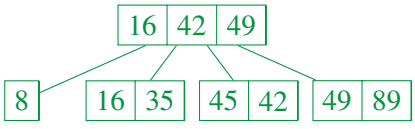
Einfügen der 49 

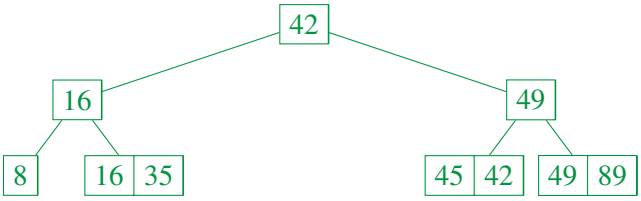
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

Einfügen der 35. 

Einfügen der 45 

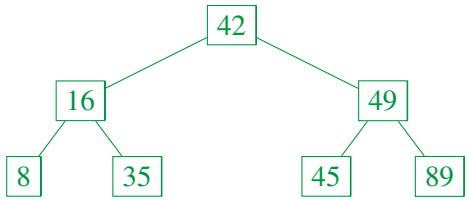
Einfügen der 8 

Der resultierende Baum ist ungültig, es muss gesplitted werden: 

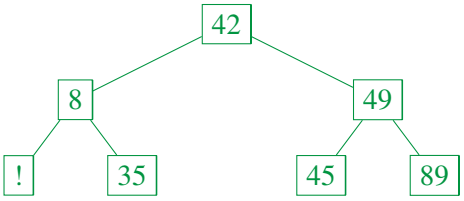
Auch hier muss ein zweites mal gesplittes werden: 

Lösungsvorschlag:

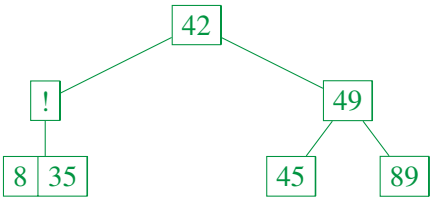
c)



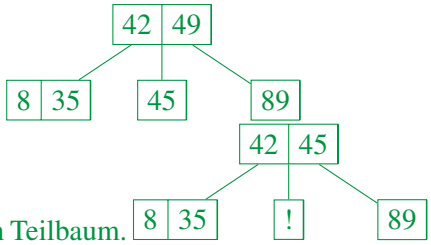
Löschen der 16. Da es sich um einen inneren Knoten handelt, wird die 16 durch den größten Wert im linken Teilbaum ersetzt



Es folgt ein Unterlauf in dem Knoten, in dem zuvor die 8 war. Da kein Nachbar mehr als k Einträge hat, wird ein neuer Knoten bestehend aus dem Nachbarknoten und dem Element, dass im Elternknoten beide Kindknoten trennt, gebildet.

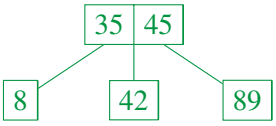


Es folgt nun ein Unterlauf in dem Elternknoten. Da auch hier kein Nachbar mehr als k Elemente hat, wird wieder unter Einbezug des Elementes aus dem Elternknoten, das beide Kindknoten trennt, ein neuer Knoten erstellt. Der neu erstellte Knoten wird dann in diesem Fall zur Wurzel.

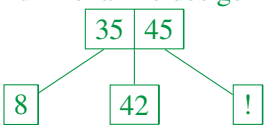


Löschen der 49. Die 49 wird ersetzt durch das größte Element im linken Teilbaum.

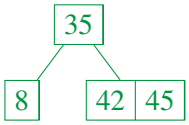
Wo bisher die 45 stand ist jetzt ein Unterlauf aufgetreten. Da der linke Nachbarknoten mehr wie k Elemente hat, findet unter Zuhilfenahme des gemeinsamen Elternknoten ein Ausgleich statt.



Löschen der 89.



Kein Nachbarknoten hat mehr wie k Elemente. Es wird also ein neuer Kindknoten gebildet.



Aufgabe 9-2 *Bitvektoren*

Gegeben ist wieder die Sammlung von Informatik-Fachbüchern:

Nummer	Titel	Autor	Jahr	Kürzel
0	Design Patterns	E. Gamma, et al.	1994	DP
1	Clean Code	Robert C. Martin	2008	CC
2	Make Your Own Neural Network	Tariq Rashid	2016	MNN
3	Agile Software Development	Robert C. Martin	2002	AgSD
4	Introduction to Algorithms	T. H. Cormen, et al.	1989	IA
5	Functional Thinking: Paradigm Over Syntax	Neal Ford	2014	FuT
6	Extreme Programming Explained: Embrace Change	Kent Beck	1999	ExPE
7	Algorithms for Reinforcement Learning	Csaba Szepesvari	2010	AlRL
8	The Software Craftsman	Robert C. Martin	2014	TheSC
9	Test Driven Development: By Example	Kent Beck	2002	TDD
10	Programming Pearls	Jon Bentley	1986	PP
11	Building Your Own Compiler with C++	Jim Holmes	1994	BYOC

Das Universum U soll im folgendem alle obigen Bücher enthalten. Ein Eintrag Bit[i] eines Bitvektors steht dafür, ob ein Buch mit Nummer i in einer gegebenen Menge enthalten ist.
Hinweis: Sie dürfen die Vektoren auch als Zeilenvektoren schreiben.

(a) Geben Sie die Größe des Universums N an.

(b) Geben Sie den Bitvektor an, der das Universum repräsentiert. Wie viele Elemente (Bücher) enthält die Menge, die er repräsentiert.

(c) Welcher Bitvektor repräsentiert folgende Menge an Büchern $M_c = \{$AgSD, TDD, BYOC, IA, TheSC$\}$?

(d) Welcher Menge repräsentiert der Bitvektor $Bit(M_d) = (1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0)^T$?

(e) Wie könnte ein effizienter Algorithmus aussehen, der mittels zweier Bitvektoren $Bit(M_1)$ und $Bit(M_2)$ die Schnittmenge und Vereinigung der Mengen, die sie repräsentieren (M_1 und M_2), berechnet und diese wieder als Bitvektor $Bit(M_{result})$ ausgibt? Geben Sie diesen Algorithmus möglichst formal korrekt in Pseudocode an. Wie groß ist die Laufzeit in O-Notation?

Lösungsvorschlag:

(a) $N = 12$

(b) $Bit(U) = (1,1,1,1,1,1,1,1,1,1,1,1)^T$. Die Menge enthält natürlich genau $N = 12$ Elemente (Bücher).

(c) $Bit(M_c) = (0,0,0,1,1,0,0,0,1,1,0,1)^T$

(d) $M_d = \{$DP, AgSD, IA, FuT, TDD$\}$

(e) Schnittmenge:

```
Algorithmus schnitt(Bit(M_1), Bit(M_2)) :
        Bit(M_result) = Initialize new Bitvektor
        for i = 0 bis N-1
                Bit(M_result)[i] = Bit(M_1)[i] AND Bit(M_2)[i]
        return Bit(M_result)
```

$\Rightarrow$ Es gilt also $Bit(M_1 \cap M_2) = schnitt(Bit(M_1), Bit(M_2))$

Vereinigung:

```
Algorithmus vereinigung(Bit(M_1), Bit(M_2)) :
        Bit(M_result) = Initialize new Bitvektor
        for i = 0 bis N-1
                Bit(M_result)[i] = Bit(M_1)[i] OR Bit(M_2)[i]
        return Bit(M_result)
```

$\Rightarrow$ Es gilt also $Bit(M_1 \cup M_2) = vereinigung(Bit(M_1), Bit(M_2))$

Die Laufzeit ermittelt sich folgendermaßen:

- Die Initialisierung benötigt $O(N)$ (siehe Vorlesung)
- Die for-Schleife wird N mal durchlaufen $\Rightarrow O(N)$
- Innerhalb der for-Schleife finden zwei Suchen nach Elementen statt, diese sind in konstanter Zeit durchführbar (siehe Vorlesung), die logischen Operationen ($\wedge$ und $\vee$) bzw. ('AND' und 'OR') sind ebenfalls in konstanter Zeit durchführbar. Eine Iteration ist also in $O(1)$ durchführbar.

Insgesamt ergibt sich für die Laufzeit also: $O(N + N * 1) = O(N)$