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

München, 18.06.2019

Algorithmen und Datenstrukturen

SS 2019

Übungsblatt 8: Suchen II

Aufgabe 8-1 *Lineare und Interpolationssuche*

(a) Führen Sie eine lineare Suche auf dem Array {0,A; 6,E; 2,C; 5,D; 1,B} nach 5 durch. Wieviele Schritte benötigen Sie? Wieviele Schritte werden, abhängig von der Listenlänge n, im schlechtesten Fall sowie durchschnittlich benötigt?

Lösungsvorschlag:
Es werden 4 Schritte benötigt, im schlechtesten Fall n, im Durchschnitt n/2.

(b) Für welche Art von Listen ist lineare Suche sinnvoll einsetzbar? Wie kann eine schnellere Suche ermöglicht werden?

Lösungsvorschlag:
Lineare Suche ist für nicht sortierte Listen sinnvoll. Durch Sortieren von Listen können andere Sucharten wie z.B. Interpolationssuche oder Binäre Suche verwendet werden.

(c) Sortieren Sie nun die Liste und führen Sie erneut eine Suche nach dem Schlüssel 2 mithilfe der Interpolationssuche durch. Wieviele Schritte benötigen Sie?

Lösungsvorschlag:
Sortiertes Array:{0,A; 1,B; 2,C; 5,D; 6,E}
1. Schritt: low = 0; high = 4; a = 2;
i = 0 + (2-0)*4/(6-0) = 1
a ist größer als 1, daher low = i + 1 = 1 + 1 = 2
2.Schritt: low = 2; high = 4; a = 2
i = 2 + (2-2) * (4-2) / (6-2) = 2 + 0 = 2
a ist gleich dem Element an Position 2. Suche beendet

(d) Wann ist eine Interpolationssuche sinnvoll auf einem Array möglich? Begründen Sie Ihre Antwort.

Lösungsvorschlag:
Wenn die Schlüsselwerte regelmäßige Abstände haben. Sonst kann eine erhöhte Suchlaufzeit entstehen.

Aufgabe 8-2 *B-Bäume*

Gegeben sei ein Array mit folgenden Werten: A = [16, 42, 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) Löschen sie die folgenden Werte aus dem B-Baum aus Aufgabenteil a): 16, 49, 89

(c) 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.

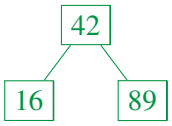
Lösungsvorschlag:

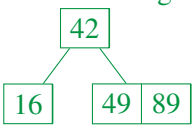
a) k= 1. Das heißt es dürfen maximal 2 Elemente in jedem Knoten sein.

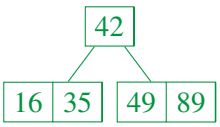
Einfügen der 16 

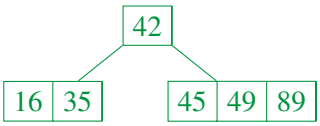
Einfügen der 42 

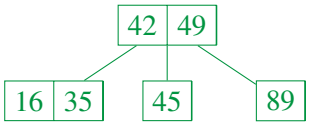
Einfügen der 89 

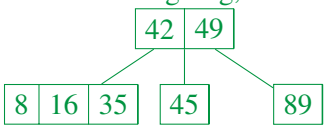
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

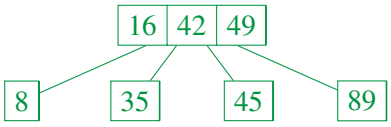
Einfügen der 49 

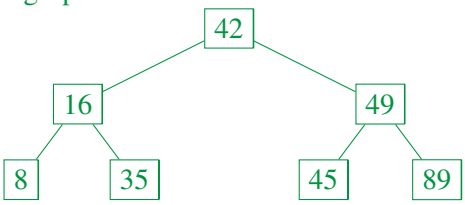
Einfügen der 35 

Einfügen der 45 

Der resultierende Baum ist ungültig, es muss gesplitted werden: 

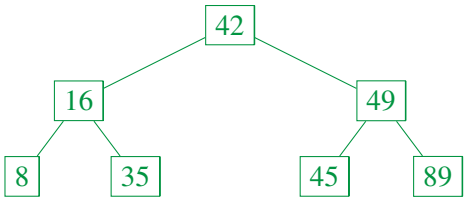
Einfügen der 8. 

Der resultierende Baum ist ungültig, es muss gesplitted werden: 

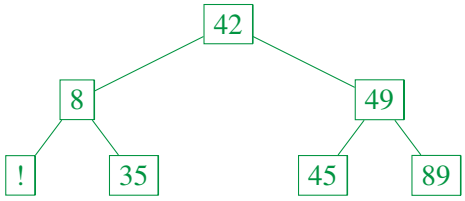
Es muss ein weiters mal gesplitted werden: 

Lösungsvorschlag:

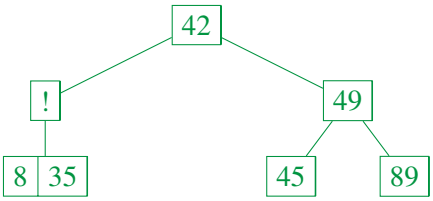
b)



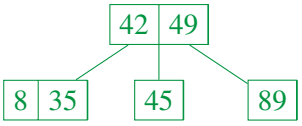
Löschen der 16. Da es sich um einen inneren Knoten handelt, wird die 16 durch den größten Wert im linken Teilbaum ersetzt



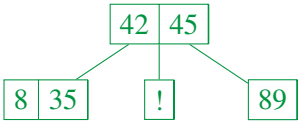
Es folgt ein Unterlauf in dem Knoten, in dem zuvor die 8 war. Da kein Nachbar mehr als k Einträge hat, wird ein neuer Knoten bestehend aus dem Nachbarknoten und dem Element, dass im Elternknoten beide Kindknoten trennt, gebildet.



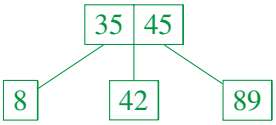
Es folgt nun ein Unterlauf in dem Elternknoten. Da auch hier kein Nachbar mehr als k Elemente hat, wird wieder unter Einbezug des Elementes aus dem Elternknoten, das beide Kindknoten trennt, ein neuer Knoten erstellt. Der neu erstellte Knoten wird dann in diesem Fall zur Wurzel.



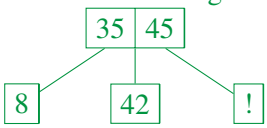
Löschen der 49. Die 49 wird ersetzt durch das größte Element im linken Teilbaum.



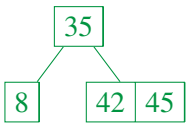
Wo bisher die 45 stand ist jetzt ein Unterlauf aufgetreten. Da der linke Nachbarknoten mehr wie k Elemente hat, findet unter Zuhilfenahme des gemeinsamen Elternknoten ein Ausgleich statt.



Löschen der 89.



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



Lösungsvorschlag:

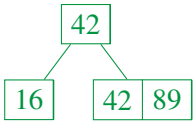
c) Möglicher Algorithmus:

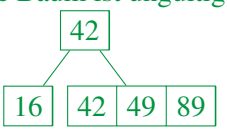
1. Einfügen in den Blättern wie bei einem B-Baum
2. Falls ein Blatt $(x_1, ..., x_{2k+1})$ aufgeteilt werden muss, teile es in zwei gleich große Knoten auf: $(x_1, ..., x_k)$ und $(x_{k+1}, ..., x_{2k+1})$. Erstelle im Elternknoten einen neuen Eintrag mit dem Wert x_k und verweise links und rechts von 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.

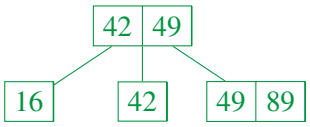
Einfügen der 16: 

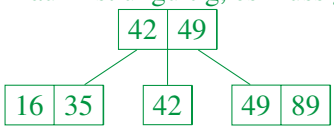
Einfügen der 42 

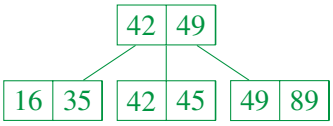
Einfügen der 89. 

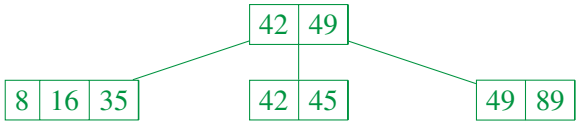
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

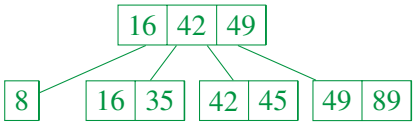
Einfügen der 49 

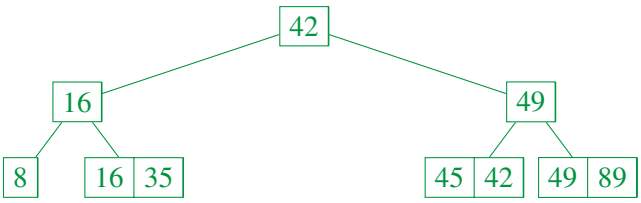
Der resultierende Baum ist ungültig, es muss gesplitted werden: 

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: 

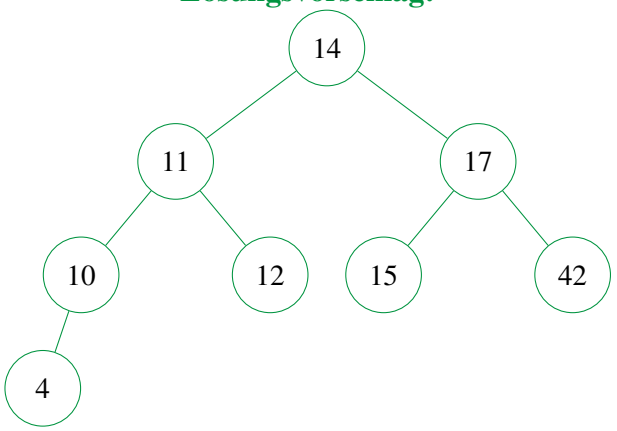
Aufgabe 8-3 *Binärer Suchbaum*

Für die Repräsentation von n Datensätzen haben Sie einen Binärbaum gewählt.

(a) Wie sieht ein binärer Suchbaum mit geringstmöglicher Tiefe für die folgenden Zahlen aus?

$$42, 17, 12, 15, 4, 11, 10, 14$$

Lösungsvorschlag:



(b) Sie wollen ein neues Element in den Baum einordnen. Geben Sie eine knappe Beschreibung des Verfahrens in Pseudocode an. Begründen Sie knapp, in welcher Komplexitätsklasse Ihr Verfahren für Insert liegt.

Lösungsvorschlag:

Folgende rekursive Methode sucht das Blatt, das am nächsten an dem einzufügenden Wert liegt und fügt den Wert als Nachfolger dieses Blattes ein.
Insert (a, B) if(a $<$ key(B)) if(B.left) Insert (a, B.left()) else B.left = new Node(a) else if(B.right) Insert (a, B.right()) else B.right = new Node(a)
Komplexität: O(n).

(c) Was muss bei der Insert-Methode beachtet werden, wenn die Suche in dem Baum nach dem Einfügen immer noch in $O(logn)$ sein soll?

Lösungsvorschlag:

Einfügen in $O(logn)$ ist nur dann möglich, wenn der Baum balanciert ist. Deshalb muss nach dem Einfügen rebalanciert werden. Dasselbe gilt auch für das Löschen von Einträgen.

(d) Beschreiben Sie kurz die Vorteile der Baumdarstellung gegenüber der Darstellung als sortierte Liste oder sortiertem Array. Beachten Sie dabei insbesondere das Einfügen und Löschen von Elementen.

Lösungsvorschlag:

Die Baumdarstellung ermöglicht das sortierte einfügen neuer Elemente in $O(logn)$. Einfügen in das Array und der Liste benötigt hingegen O(n). Dasselbe gilt auch für das Löschen von Elementen. Der Zugriff auf die Elemente kann im Array in Konstanter Zeit realisiert werden. Bei Listen werden O(n) Operationen benötigt. Hierbei werden beim Baum $O(logn)$ Operationen benötigt. Somit ist der Baum beim Zugriff nicht so effizient wie Arrays, jedoch besser als die Liste.