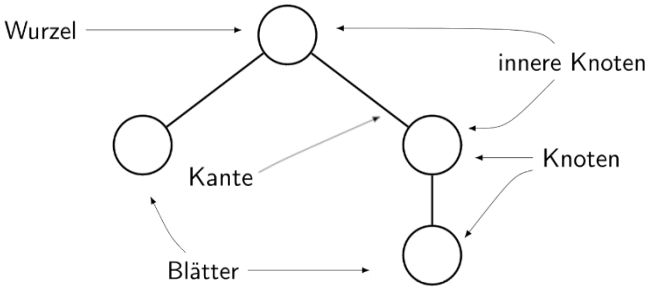
Binäre (Such-)Bäume

Binäre Bäume

Grundbegriffe



- **Knoten** = Element, dass die eigentlichen Daten und die Verweise auf die nachfolgenden Knoten trägt (vgl. Listenelement)
- **Kanten** = Symbolisiert die Verbindung zwischen zwei Knoten, wird jedoch nicht implementiert (Nur indirekt über die Verweise in den Knoten-Objekten)
- **Wurzel** = Ist der oberste Knoten im Baum und besitzt als einziger Knoten keinen Vorgänger
- **Innere Knoten** = Knoten die mindestens einen Nachfolger haben
- **Blätter** = Knoten die keine Nachfolger haben

Eigenschaften

Knoten:

- Tiefe = Anzahl der Knoten auf dem Pfad zur Wurzel (die Wurzel selbst wird mitgezählt)
- Grad = Die Anzahl der Nachfolger die ein Knoten hat

Bäume:

- Höhe = Höchste Tiefe die ein Knoten im Baum besitzt
- Ordnung = Anzahl der Nachfolger, die ein Knoten im Baum maximal haben kann
- Ausgefüllt = Jeder innere Knoten hat die maximale Anzahl an Nachfolgern
- Vollständig = jedes Niveau hat die maximale Anzahl an Knoten
    - links-vollständig = Knoten werden von links nach rechts eingefügt & Tief der Blätter unterscheiden sich höchstens um 1

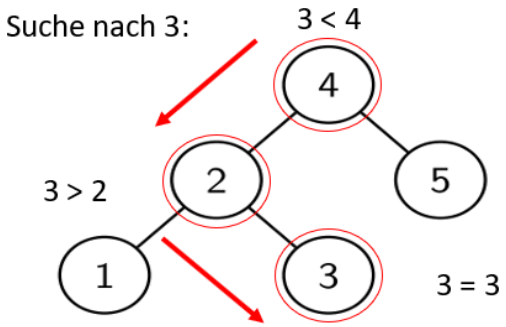
Binäre Suchbäume

Ein binärer Baum mit einem Schlüssel für jeden Knoten; Die Grundbegriffe und Eigenschaften, die es für normale binäre Bäume gibt, lassen sich übertragen.

Bedingung: Die Elemente im linken Teilbaum eines Knotens müssen immer kleiner oder gleich dem Knoten sein, die im rechten Teilbaum müssen größer sein.

Suchen

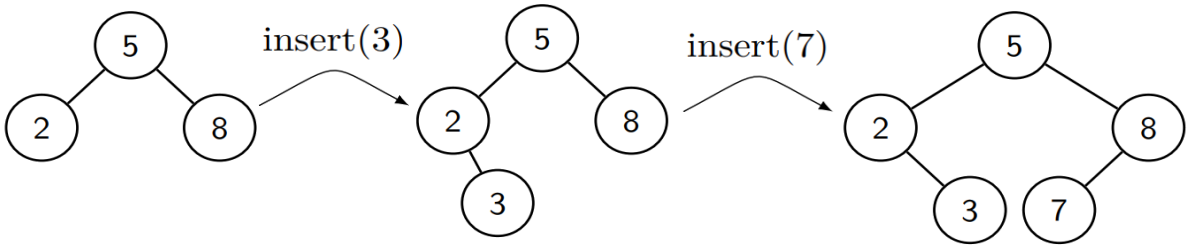
1. G (gesuchtes Element) wird mit W (Wurzel des (Teil-)Baums) verglichen
    a. G < W wiederhole Schritt 1 mit dem linken Teilbaum von W
        i. Gibt es keinen linken Teilbaum ist G nicht im Baum vorhanden
    b. G > W wiederhole Schritt 1 mit dem rechten Teilbaum von W
        i. Gibt es keinen rechten Teilbaum ist G nicht im Baum vorhanden
    c. G == W Das Element wurde gefunden



Max. Laufzeit: Höhe des Baumes +1

Einfügen

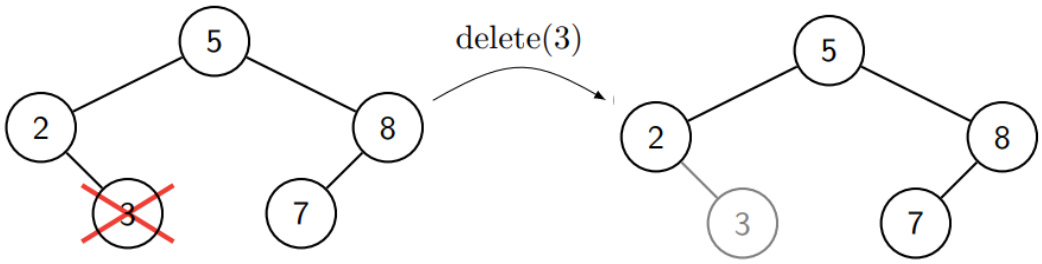
1. E (einzufügende Element) wird mit W (Wurzel des (Teil-)Baums) verglichen
    a. E <= W wiederhole Schritt 1 mit dem linken Teilbaum von W
        i. Gibt es keinen linken Teilbaum, setzte E als linken Nachfolger von W
    b. E > W wiederhole Schritt 1 mit dem rechten Teilbaum von W
        i. Gibt es keinen rechten Teilbaum, setzte E als rechten Nachfolger von W



Löschen

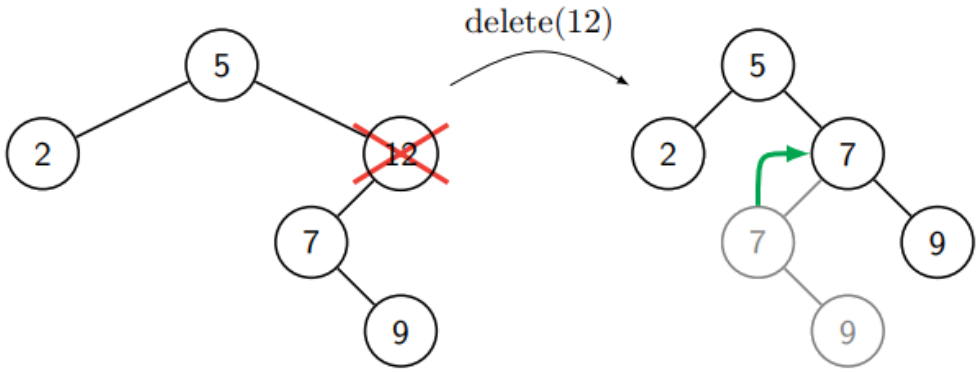
Fall 1: Das zu löschende Element hat keinen Nachfolger

Der Verweis auf das Element wird im Vater-Knoten entfernt.



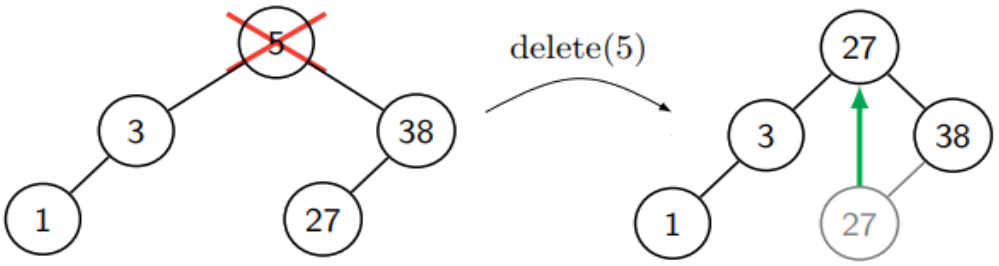
Fall 2: Das zu löschende Element hat einen Nachfolger

Der Verweis, im Vater Knoten, auf das zu löschende Element wird auf den Nachfolger des zu löschenden Elements geändert.



Fall 3: Das zu löschende Element hat zwei Nachfolger

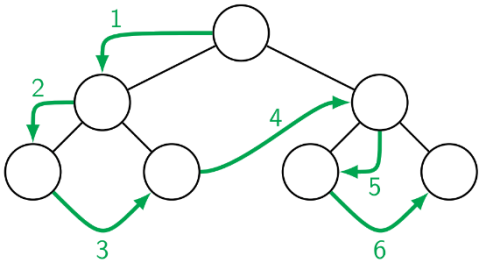
Das kleinste Element im rechten Teilbaum des zu löschenden Elements wird gesucht und mit diesem das zu löschende Element komplett ersetzt.



Traversierung

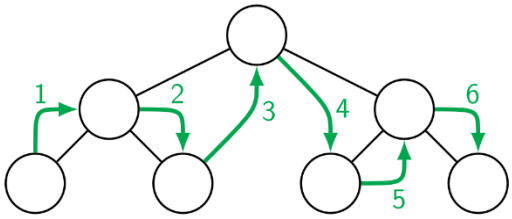
Preorder (Hauptreihenfolge)

1. Gib den Inhalt der Wurzel aus
2. Durchlaufe den linken Teilbaum in der Preorder-Reihenfolge
3. Durchlaufe den rechten Teilbaum in der Preorder-Reihenfolge



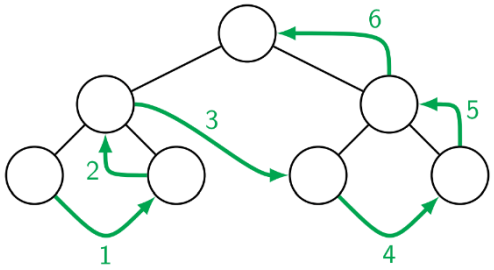
Inorder (symmetrische Reihenfolge)

1. Durchlaufe den linken Teilbaum in der Inorder-Reihenfolge
2. Gib den Inhalt der Wurzel aus
3. Durchlaufe den rechten Teilbaum in der Inorder-Reihenfolge



Postorder (Nebenreihenfolge)

1. Durchlaufe den linken Teilbaum in der Postorder-Reihenfolge
2. Durchlaufe den rechten Teilbaum in der Postorder-Reihenfolge
3. Gib den Inhalt der Wurzel aus



Quellen

ETH-Zürich (+Übungsaufgaben): https://bit.ly/2HUIOrB

Unsere Unterlagen: https://github.com/Bennet303/BinarySearchTree