AuD Zusammenfassung WS20/21

Leon Vatthauer

4. März 2021

Inhaltsverzeichnis

1 Multiple Choice / Wissen

- Java-specifics (Bsp. Testfall Deklaration mit @Test)
- Laufzeiten
- Generelles aus allen Bereichen

2 Bäume

2.1 Traversierung

2.2 Binärer Suchbaum

Einfügen

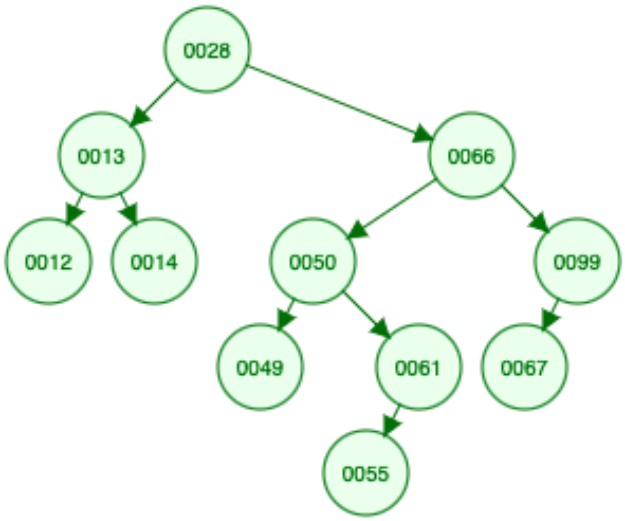
- Kleinere Zahl links einfügen
- Größere Zahl rechts einfügen

Löschen

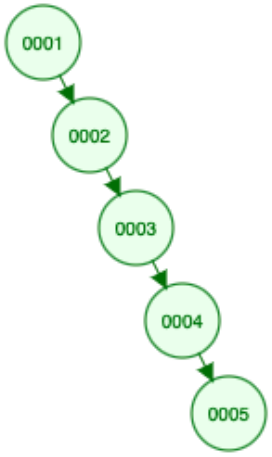
Ersetze gelöschten Knoten durch:
- kleinster Knoten des rechten Teilbaums
oder
- größter Knoten des linken Teilbaums

Weiteres

Bäume können entarten durch sortiertes Einfügen



(a) Binärer Suchbaum



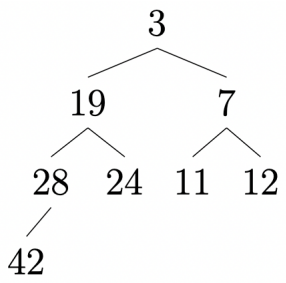
(b) Entarteter Suchbaum

Abbildung 1: Suchbäume

2.3 Halden

Aufbau

Alle Nachfolger eines Knoten sind größter / kleiner (Min/Max Halde)
'Ebenen' des Baumes werden von links nach rechts in Array eingebettet



(a) Min-Halde

3	19	7	28	24	11	12	42

(b) Halde als Array

Abbildung 2: Halden

Einfügen

Neuen Wert hinten ins Array und dann Halden-Eigenschaft wiederherstellen:
→ Wert wird mit Elternknoten verglichen, bis Halden-Eigenschaft wieder gilt

Löschen

Knoten löschen und letzten Wert im Array an Stelle bewegen
dann Halden-Eigenschaft durch tauschen wiederherstellen

2.4 AVL-Bäume

Aufbau

Balancierte binäre Suchbäume

Einfügen und Löschen

Analog zum Suchbaum
mit darauffolgender Balancierphase

2.4.1 Balancieren

Höhe
Anzahl Knoten unterhalb
Balancefaktor
$B = H_{links} - H_{rechts}$
Vorgehen bei der Klausur:

1. Höhe und Balancefaktor bei allen Knoten ermitteln
2. Knoten mit $|B| \geq 2$ betrachten
3. Vorzeichenwechsel? → 2 Rotationen
   sonst: eine Rotation

3 Graphen

3.1 Begriffe

- **Pfad**
  Weg zwischen zwei Knoten, Länge ist Anzahl von Kanten
- **Grad**
  Bei ungerichteten: Anzahl Verbindungen
  Bei gerichteten: Unterscheidung Eingangsgrad — Ausgangsgrad
- **Spannbaum**
  Zyklenfreier Teilgraph der alle Knoten enthält
- **Wurzel**
  Knoten mit Eingangsgrad 0

3.2 Darstellungen

3.2.1 TODO BILDER

3.3 Spannbaum-Algorithmen

3.3.1 Prim

Greedy-Algorithmus
Vorgehen:

- Beginne bei Startknoten
- Füge 'günstigste' anliegende Kante hinzu, die einen neuen Knoten verbindet
- Wiederhole, solange Spannbaum unvollständig

3.3.2 Kruskal

Ebenfalls Greedy-Algorithmus

- Erstelle Graph mit allen Knoten, aber ohne Kanten
- Sortiere Kanten des original aufsteigend
- Füge billigste Kante zum Spannbaum hinzu, wenn beide Knoten noch nicht verbunden sind

3.4 Traversierung

3.4.1 Tiefensuche

Graph wird zu erst in die Tiefe durchsucht
Würde ein bereits besuchter Knoten als nächstes kommen gehe einen Schritt zurück

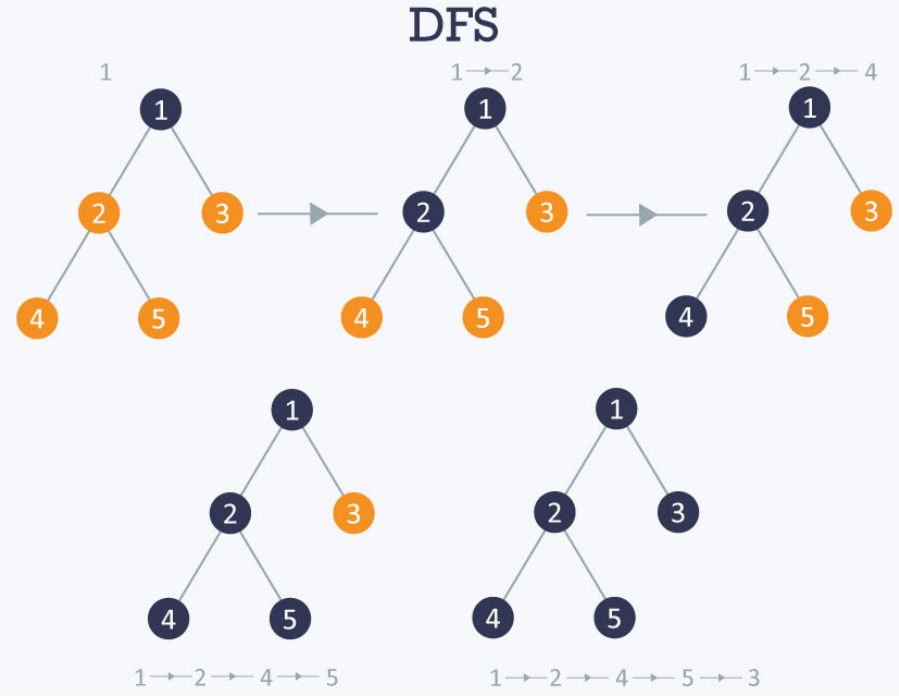


Abbildung 3: Tiefensuche am Beispiel

Als Algorithmus:
Implementierung mit Stack

- Besuche Knoten
- Füge alle Kinder zu Stack hinzu
- Während Stack nicht empty:
- betrachte erstes Element von Stack, wenn noch nicht markiert: Punkt 1

3.4.2 Breitensuche

Graph wird 'ebenenweise' durchsucht
→ Besuche erst alle Children, dann fange mit ihren Children an

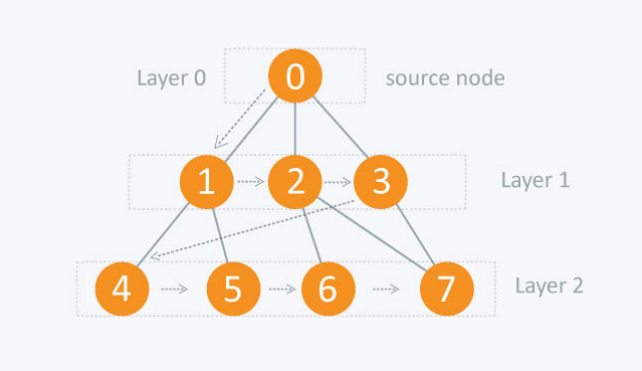


Abbildung 4: Breitensuche am Beispiel

Als Algorithmus:
Implementierung mit Queue

- Besuche Knoten
- Füge alle Children, die noch nicht besucht wurden, in Queue hinzu
- Besuche nächsten Knoten in Queue

3.5 Kürzeste Wege

4 Rekursion

5 Dyn. Prog.

6 Sortieren

7 HashMaps

8 ADTs

9 Listen

10 Backtracking

11 Suche

12 WP-Kalk, Induktion

13 Schreibtischlauf

14 UML/OOP

15 Vererbung