hhu.

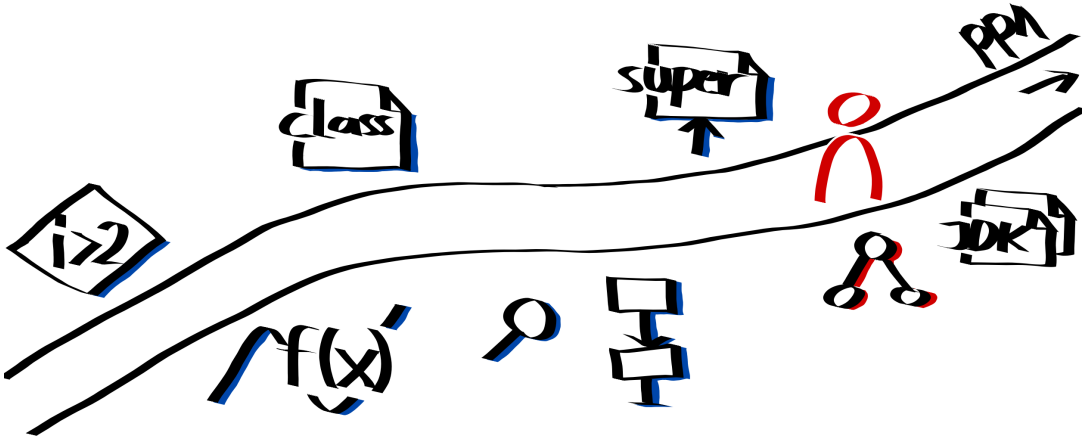
Kapitel 7: Datenstrukturen für effiziente Suche

Löschen aus Bäumen & generische Bäume

Wintersemester 2023/24

Stand 3. August 2023, 15:37 Uhr

Wo stehen wir gerade?



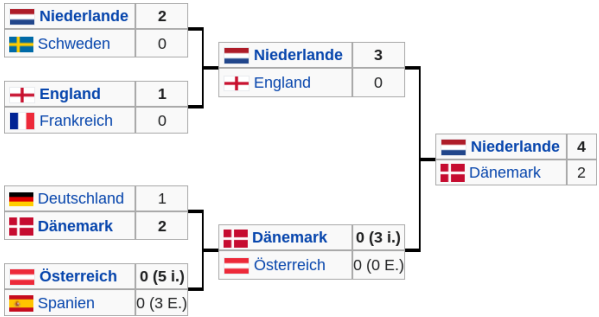
Vorlesungsfeedback

Wo werden Bäume noch benutzt?

Wo werden Bäume noch benutzt?

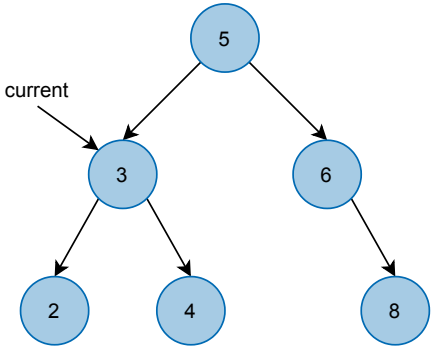
- Stammbaum
- Turnierbaum
- Dateisystembaum
- Modellierung von Argumentationen
- Verallgemeinerung: Graphen
  - Backpropagation (künstliche neuronale Netze)
  - Kürzester Weg zwischen 2 Knoten
  - Routing von Netzwerkpaketen

⇒ Informatiker:innen haben oft mit Baum-/Graph-Strukturen zu tun



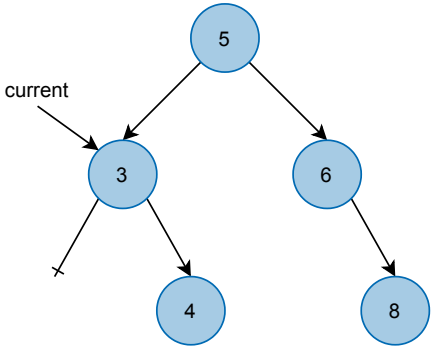
Entfernen des Minimums: Erster Versuch

1. Gehe zu Knoten *c* vorm Minimum
2. Setze linken Teilbaum von *c* auf `null`



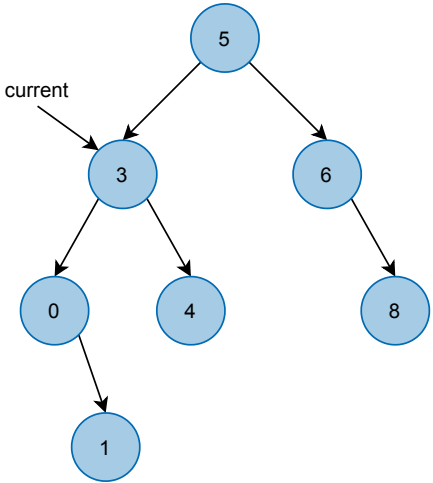
Entfernen des Minimums: Erster Versuch

1. Gehe zu Knoten *c* vorm Minimum
2. Setze linken Teilbaum von *c* auf `null`



Entfernen des Minimums: Erster Versuch

1. Gehe zu Knoten *c* vorm Minimum
2. Setze linken Teilbaum von *c* auf `null`
   - 2. Beispiel

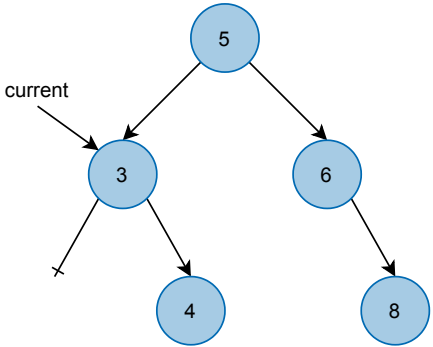


Entfernen des Minimums: Erster Versuch

1. Gehe zu Knoten *c* vorm Minimum
2. Setze linken Teilbaum von *c* auf `null`

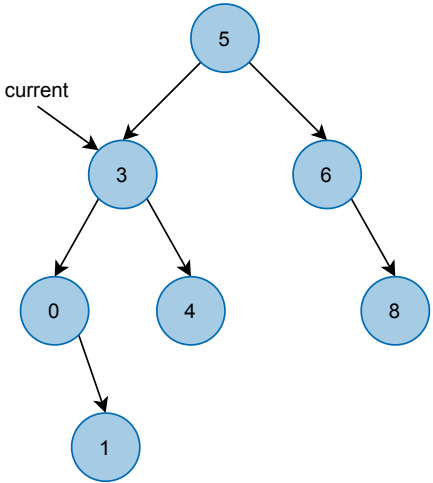
- 2. Beispiel

! falsch, wenn an Minimum ein rechter Teilbaum hängt



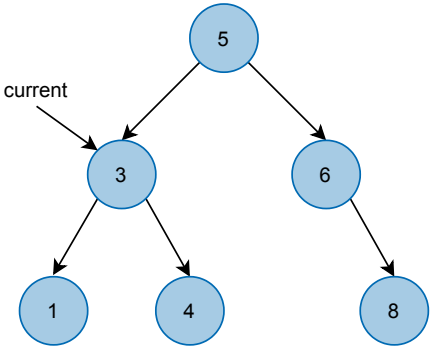
Entfernen des Minimums (jetzt richtig)

1. Gehe zu Knoten *c* vorm Minimum *m*
2. Setze linken Teilbaum von *c* gleich rechten Teilbaum von *m*



Entfernen des Minimums (jetzt richtig)

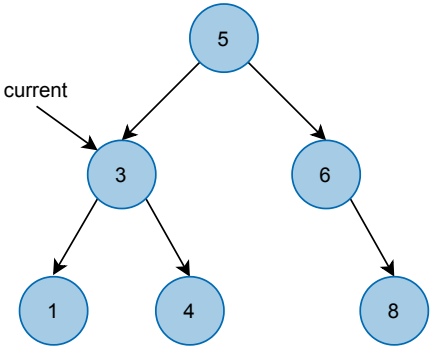
1. Gehe zu Knoten *c* vorm Minimum *m*
2. Setze linken Teilbaum von *c* gleich rechten Teilbaum von *m*



Entfernen des Minimums (jetzt richtig)

1. Gehe zu Knoten *c* vorm Minimum *m*
2. Setze linken Teilbaum von *c* gleich rechten Teilbaum von *m*

Haben wir noch einen besonderen Fall übersehen?

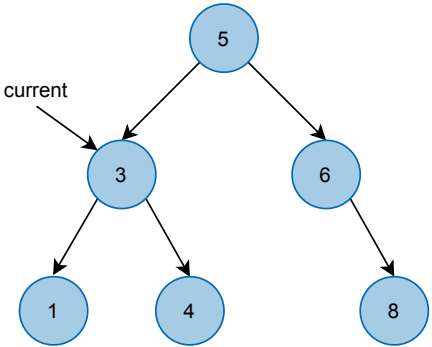


Entfernen des Minimums (jetzt richtig)

1. Gehe zu Knoten *c* vorm Minimum *m*
2. Setze linken Teilbaum von *c* gleich rechten Teilbaum von *m*

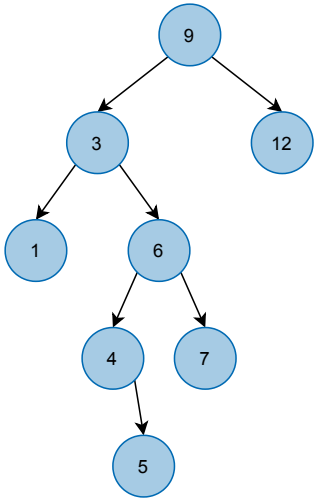
Haben wir noch einen besonderen Fall übersehen?

- Minimum = Wurzel: Setze Wurzel auf rechten Nachfolger der Wurzel
- leerer Baum: Mache nichts



Entfernen eines beliebigen Elements

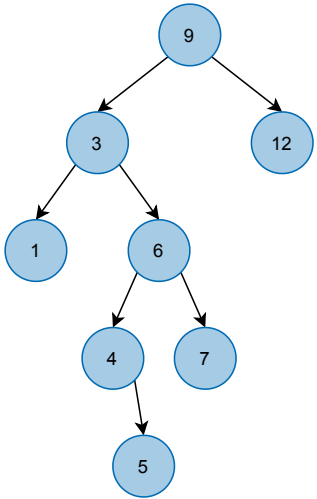
Verschiedene Fälle:



Entfernen eines beliebigen Elements

Verschiedene Fälle:

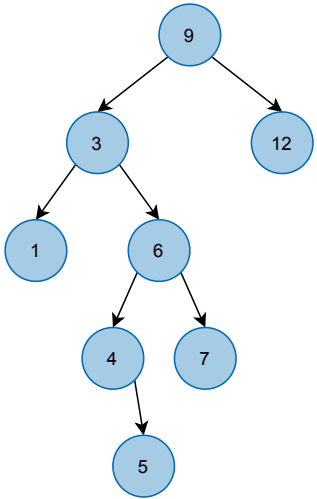
- zu löschendes Element ist Blatt
  - → trivial



Entfernen eines beliebigen Elements

Verschiedene Fälle:

- zu löschendes Element ist Blatt
  - → trivial
- zu löschendes Element *e* hat 1 Kind mit Wurzel *k*
  - ersetze *e* durch *k*

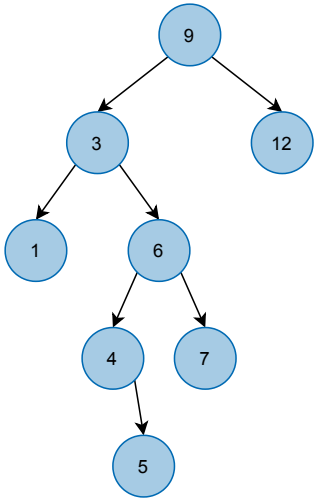


Entfernen eines beliebigen Elements

Heinrich Heine Universität Düsseldorf

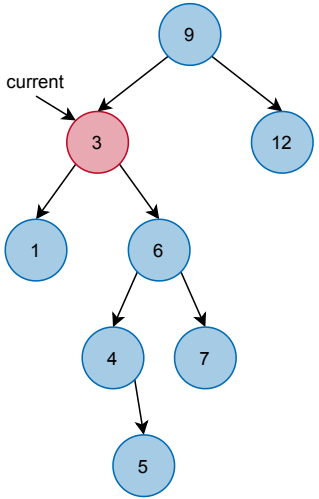
Verschiedene Fälle:

- zu löschendes Element ist Blatt
  - $\rightarrow$ trivial
- zu löschendes Element e hat 1 Kind mit Wurzel k
  - ersetze e durch k
- zu löschendes Element e hat 2 Kinder k_l, k_r
  1. suche Minimum m im Teilbaum mit Wurzel k_r
  2. lösche m
  3. ersetze e durch m



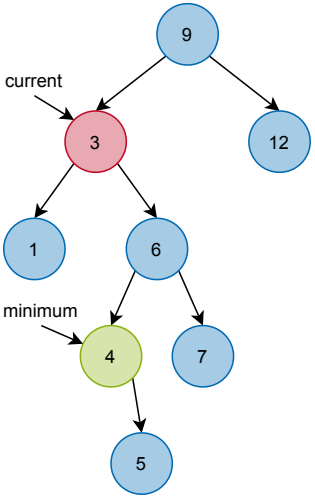
Beispiel: Entfernen der 3

0. Finde Knoten mit 3



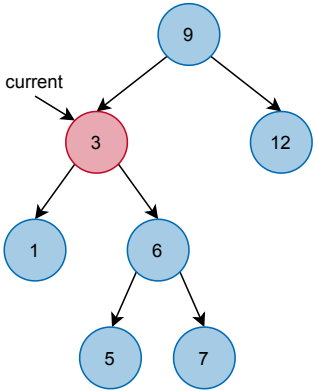
Beispiel: Entfernen der 3

0. Finde Knoten mit 3
1. Finde Minimum im rechten Teilbaum



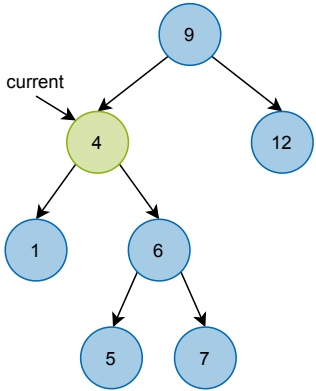
Beispiel: Entfernen der 3

0. Finde Knoten mit 3
1. Finde Minimum im rechten Teilbaum
2. Entferne das Minimum des rechten Teilbaums



Beispiel: Entfernen der 3

0. Finde Knoten mit 3
1. Finde Minimum im rechten Teilbaum
2. Entferne das Minimum des rechten Teilbaums
3. Ersetze 3 durch das gefundene Minimum



Geht Löschen nicht auch iterativ?

Geht Löschen nicht auch iterativ?

Heinrich Heine
Universität Düsseldorf

Ja, ist aber etwas unübersichtlicher und das wollte ich Ihnen (jetzt) nicht antun . . .
Probieren Sie es ruhig einmal aus.

Listen vs. Binäre Suchbäume

Listen vs. Binäre Suchbäume

Einfach verkettete Listen:

+ dynamische Datenstruktur
+ beliebige Reihenfolge speicherbar
+ auch Elemente ohne sinnvolle Sortierreihenfolge speicherbar
+ doppelte Elemente möglich
− langsamere Suche

Binäre Suchbäume:

+ dynamische Datenstruktur
− Reihenfolge durch Sortierschlüssel gegeben
− Sortierschlüssel erforderlich
− typischerweise keine doppelten Elemente
+ (potentiell) schnelle Suche
• ein bisschen mehr Speicherplatz

Listen vs. Binäre Suchbäume

Einfach verkettete Listen:

+ dynamische Datenstruktur
+ beliebige Reihenfolge speicherbar
+ auch Elemente ohne sinnvolle Sortierreihenfolge speicherbar
+ doppelte Elemente möglich
− langsamere Suche

Binäre Suchbäume:

+ dynamische Datenstruktur
− Reihenfolge durch Sortierschlüssel gegeben
− Sortierschlüssel erforderlich
− typischerweise keine doppelten Elemente
+ (potentiell) schnelle Suche
• ein bisschen mehr Speicherplatz

Wann benutze ich was?

Listen vs. Binäre Suchbäume

Heinrich Heine Universität Düsseldorf

Einfach verkettete Listen:

+ dynamische Datenstruktur
+ beliebige Reihenfolge speicherbar
+ auch Elemente ohne sinnvolle Sortierreihenfolge speicherbar
+ doppelte Elemente möglich
− langsamere Suche

Binäre Suchbäume:

+ dynamische Datenstruktur
− Reihenfolge durch Sortierschlüssel gegeben
− Sortierschlüssel erforderlich
− typischerweise keine doppelten Elemente
+ (potentiell) schnelle Suche
• ein bisschen mehr Speicherplatz

Wann benutze ich was?

- Reihenfolge soll erhalten bleiben, doppelte Elemente möglich → Liste
- Schnelle Suche erforderlich, Mengen-Eigenschaften erwünscht → Suchbaum

Anmerkung: lernen Datenstruktur kennen, die noch schnellere Suche ermöglicht

Etwas anderes speichern

Was können wir tun, um in unserem Baum beliebige Elemente zu speichern?

Etwas anderes speichern

Was können wir tun, um in unserem Baum beliebige Elemente zu speichern?

→ Generics!

- Problem: Einschränkung auf Datentypen, die Ordnung definieren

Generics & Typhierarchien

Heinrich Heine Universität Düsseldorf

- Bei Deklaration der Typvariable kann Obertyp (Klasse oder Interface) angegeben werden
- Bei Instanziierung prüft Compiler, ob der konkrete Typ den gewünschten Obertypen hat

```
1  public class BinarySearchTree<T extends Sortable> {
2
3      public boolean contains(T needle) {
4          BinaryNode current = root;
5          while(true) {
6              if(current == null) {
7                  return false;
8              } else if(current.element.getSortKey() == needle.getSortKey()) {
9                  return true;
10             } else if(current.element.getSortKey() < needle.getSortKey()) {
11                 current = current.right;
12             } else {
13                 assert current.element.getSortKey() > needle.getSortKey();
14                 current = current.left;
15             }
16         }
17     }
```

Benutzen fertige Bäume im JDK `T extends Bla`?

Heinrich Heine
Universität Düsseldorf

Benutzen fertige Bäume im JDK `T extends Bla`?

Heinrich Heine Universität Düsseldorf

Nein (ausnahmsweise)

- Grund: benutzerdefinierte Compare-Methode an Baum übergebbar
- Vorteile:
  - dieselbe Klasse je nach Anwendungsfall nach verschiedenen Eigenschaften sortierbar (und damit schnell durchsuchbar)
  - Objekte von Klassen speicherbar, die selbst keine Sortierreihenfolge definieren

Übungsblatt

Heinrich Heine
Universität Düsseldorf

- Ein kleiner Taschenrechner
- Binärer Suchbaum
- Nachrichtensystem

Klausuranmeldung

Heinrich Heine
Universität Düsseldorf

Denken Sie daran, sich unter `studierende.hhu.de` für die Klausur anzumelden.

- späteste Anmeldung: 1 Woche vor der Klausur
  - Anmeldung auch möglich, wenn Zulassungsstand jetzt noch ungewiss (automatische Abmeldung, falls Zulassung nicht erreicht wird)
- Abmeldung bis 1 Woche vor Klausur möglich
- bei Krankheit: Prüfungsamt informieren, Attest erforderlich (s. Informationen auf Uni-Seite)
- wer sich nicht übers Portal anmelden kann (Schüler:innen, BWL):
  Mail mit Betreff *Klausuranmeldung* an progra@cs.uni-duesseldorf.de