Algorithmen 1 SS 2013 – Tutorium 7

8. Tutorium

Sarah Lutteropp

11. Juni 2013

Übersicht

7. Übungsblatt und Probeklausur

Die Korrekturen

Gibt es nächste Woche. Schaut euch den Stapel an und ihr könnt euch denken wieso.

Neues von der Tutorenbesprechung

Zitat von den Übungsleitern

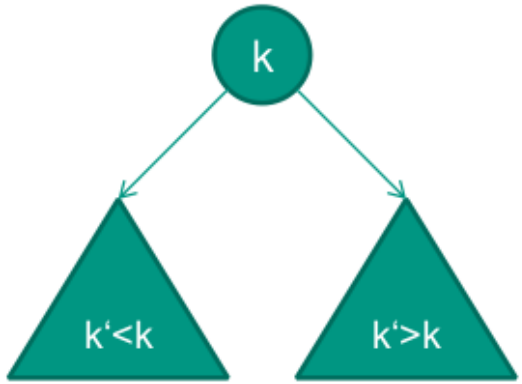
Altklausuren von Zitterbart “lohnen sich nicht”.

Wahlwerbung

Wir sollen in den Tutorien Wahlwerbung machen. Es werden gewählt: StuPa + FS-Sprecher.

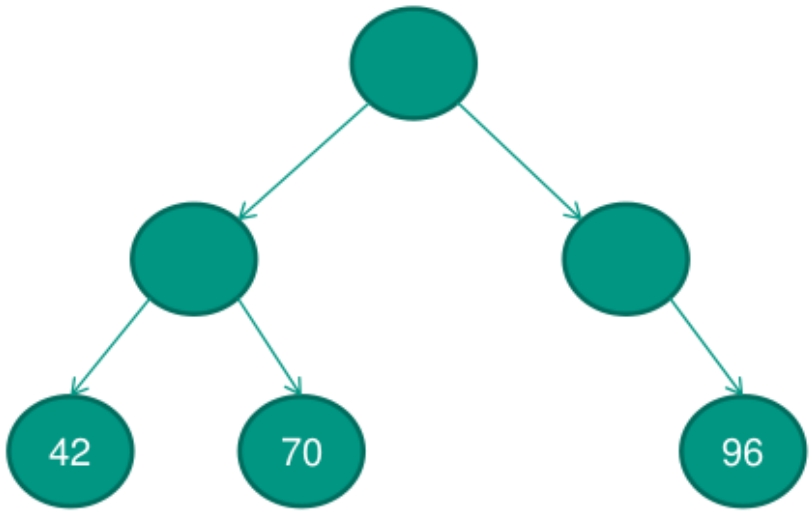
Suchbäume

- Werte im linken Teilbaum sind kleiner als betrachteter Knoten
- Werte im rechten Teilbaum sind größer als betrachteter Knoten

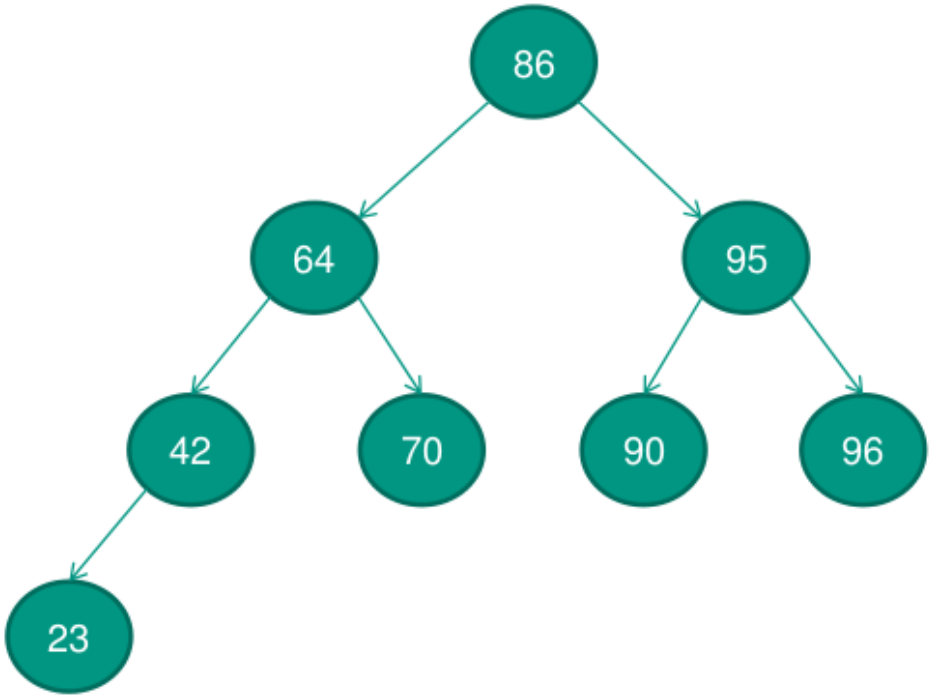


Klassifikation	Alternative 1	Alternative 2
#Kindknoten	Binärbaum: max. 2 Kinder	Vielwegbaum: Anzahl beliebig
Nutzdaten	Blattbaum: nur in Blattknoten	Natürlicher Baum: in jedem Knoten
Balance	Balancierter Baum: Höhe der Unterbäume unterscheidet sich maximal um 1	Unbalancierter Baum: keine Einschränkung

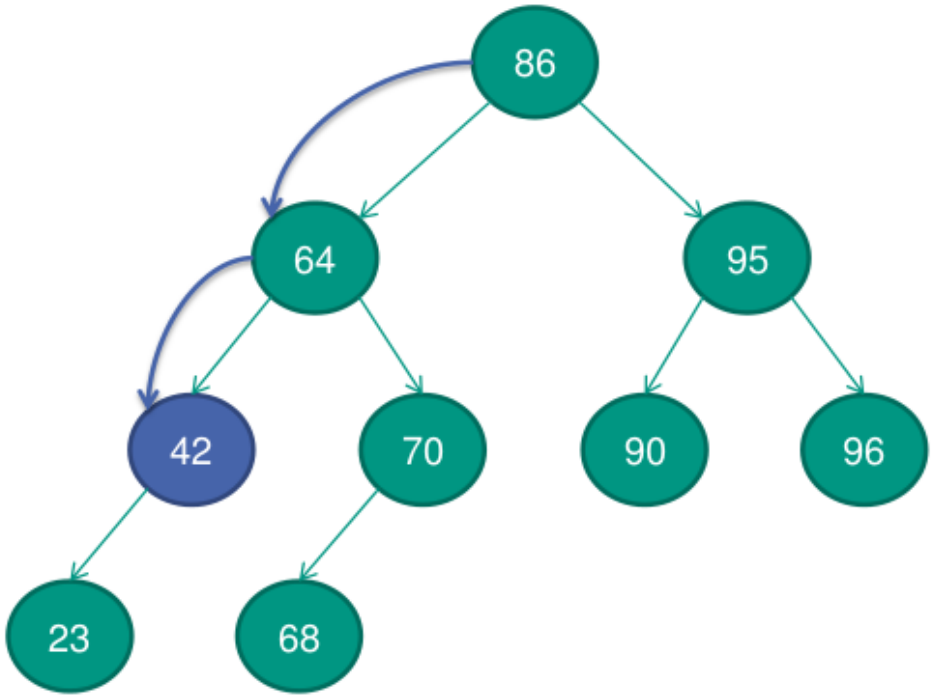
Beispiel: Blattbaum



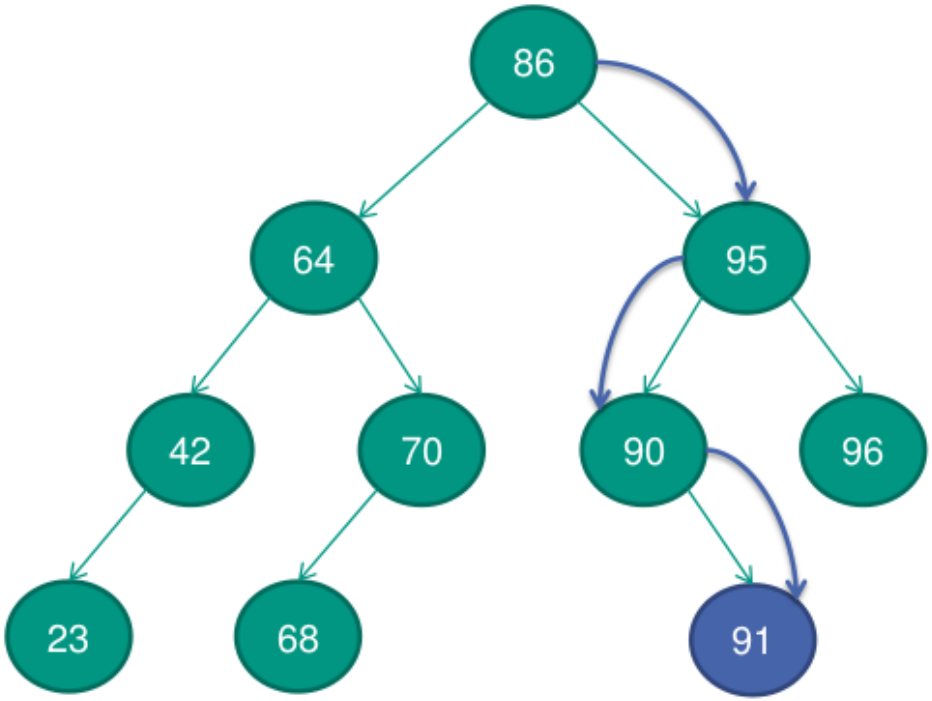
Beispiel: Balancierter Baum



Beispiel: Suche die 42



Beispiel: Füge die 91 ein



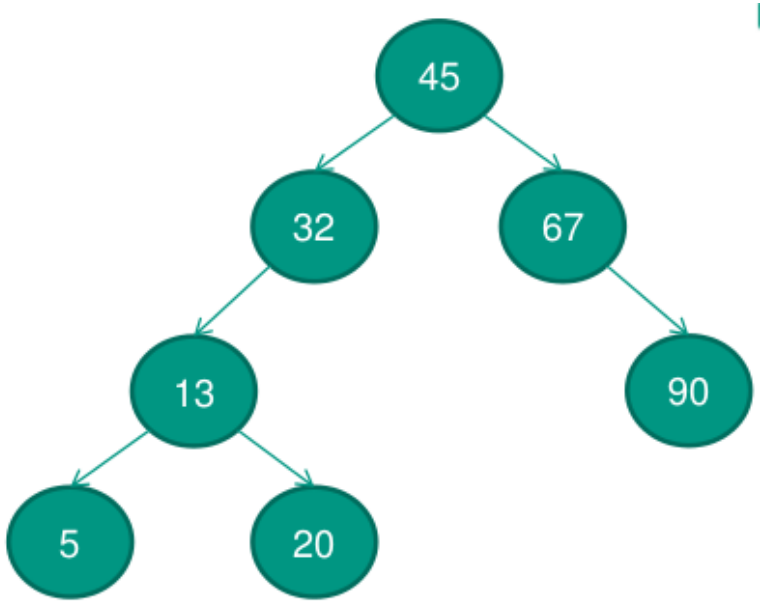
Aufgabe

Einfügen

Füge folgende Schlüssel nacheinander in einen binären Suchbaum ein: 45, 32, 13, 67, 20, 5, 90

Ist der resultierende Baum balanciert?

Lösung



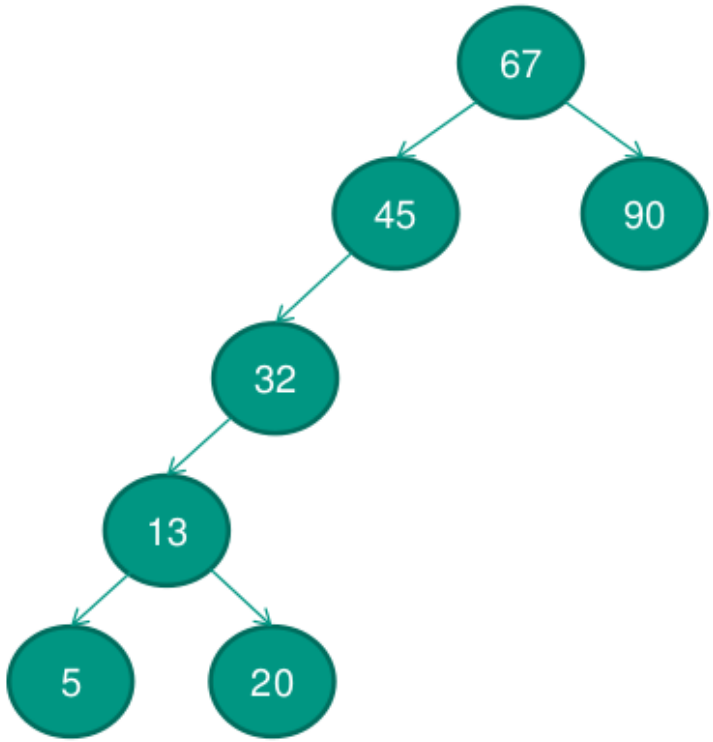
Ja.

Aufgabe

Einfügen

Füge folgende Schlüssel nun in folgender Reihenfolge in einen binären Suchbaum ein: 67, 45, 32, 13, 20, 5, 900
Ist der resultierende Baum balanciert?

Lösung



Nein.

Binäre Suchbäume: Löschen

Fall 1: Knoten ist Blattknoten

Einfach löschen.

Fall 2: Knoten hat genau ein Kind

Ersetze zu löschenden Knoten mit dem Teilbaum.

Fall 3: Knoten hat zwei Kinder

Binäre Suchbäume: Löschen

Fall 1: Knoten ist Blattknoten

Einfach löschen.

Fall 2: Knoten hat genau ein Kind

Ersetze zu löschenden Knoten mit dem Teilbaum.

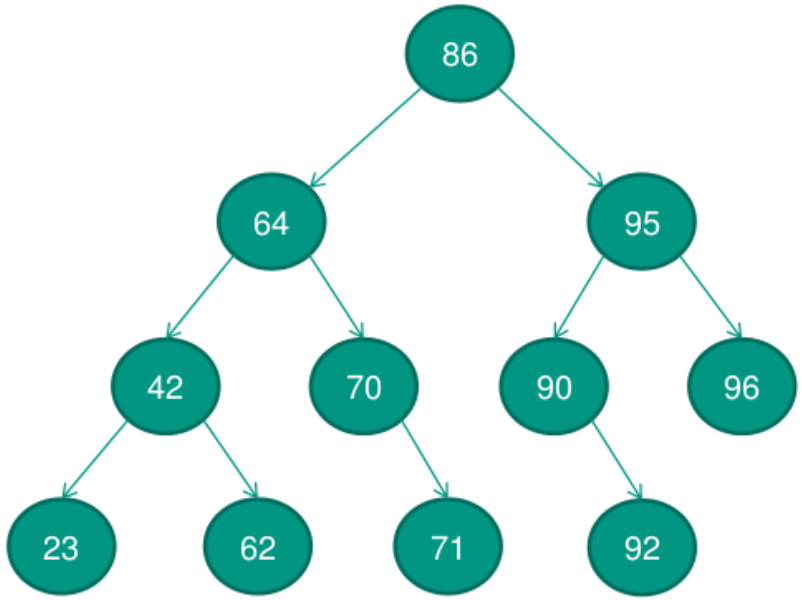
Fall 3: Knoten hat zwei Kinder

- A) Eines der Kinder hat NIL als Kind $\rightsquigarrow$ Ersetze zu löschenden Knoten mit diesem Teilbaum.
- B) Tausche zu löschenden Knoten mit größtem Knoten aus linkem Teilbaum oder kleinsten Knoten aus rechtem Teilbaum.

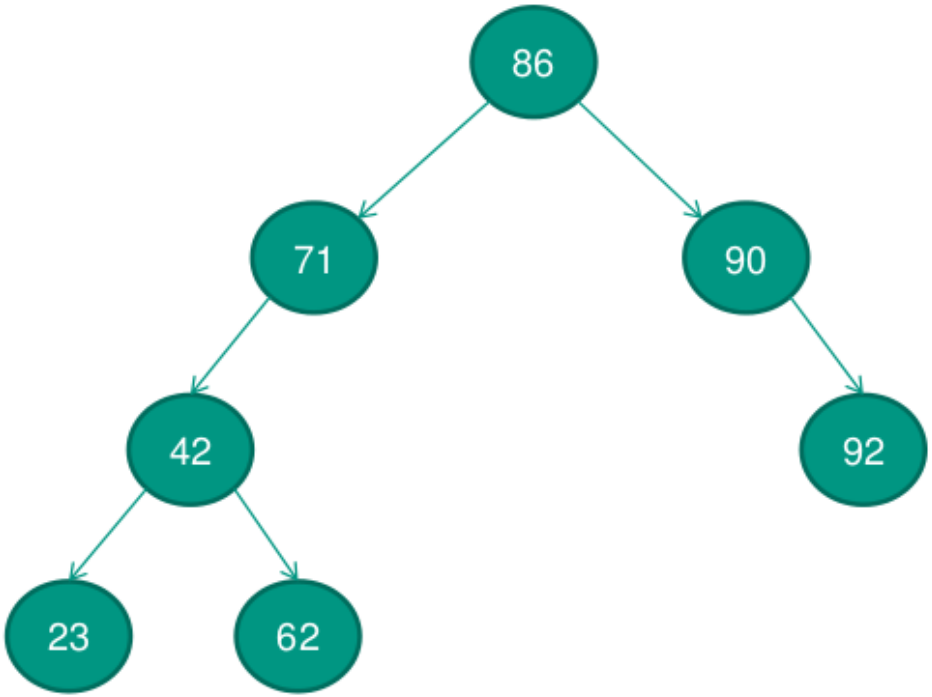
Aufgabe

Löschen

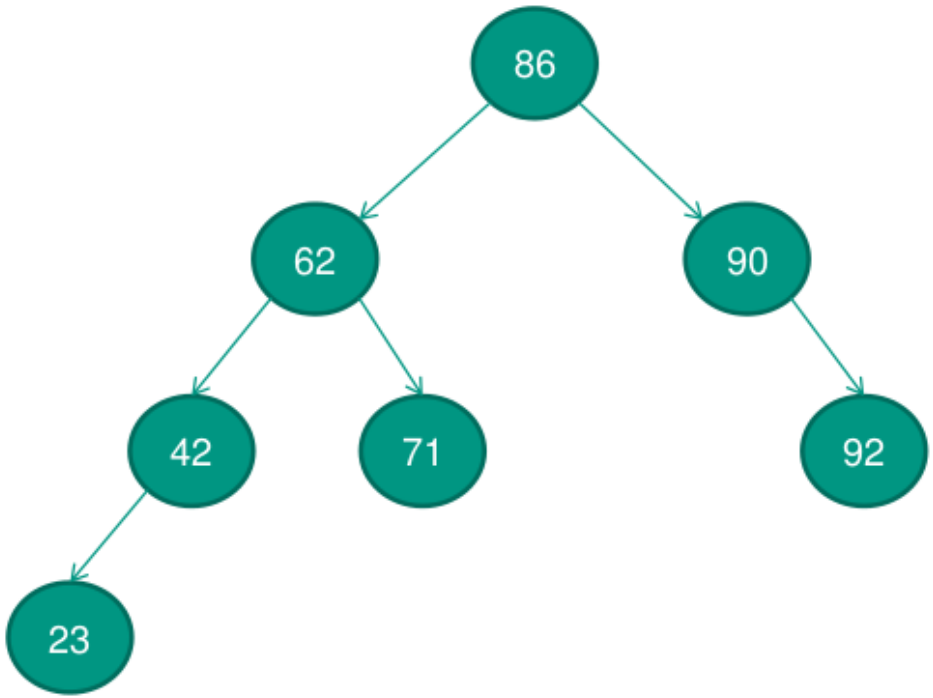
Entferne nacheinander die Knoten: 96, 95, 64, 70



Lösung Alternative 1



Lösung Alternative 2



Eigenschaften Rot-Schwarz-Baum

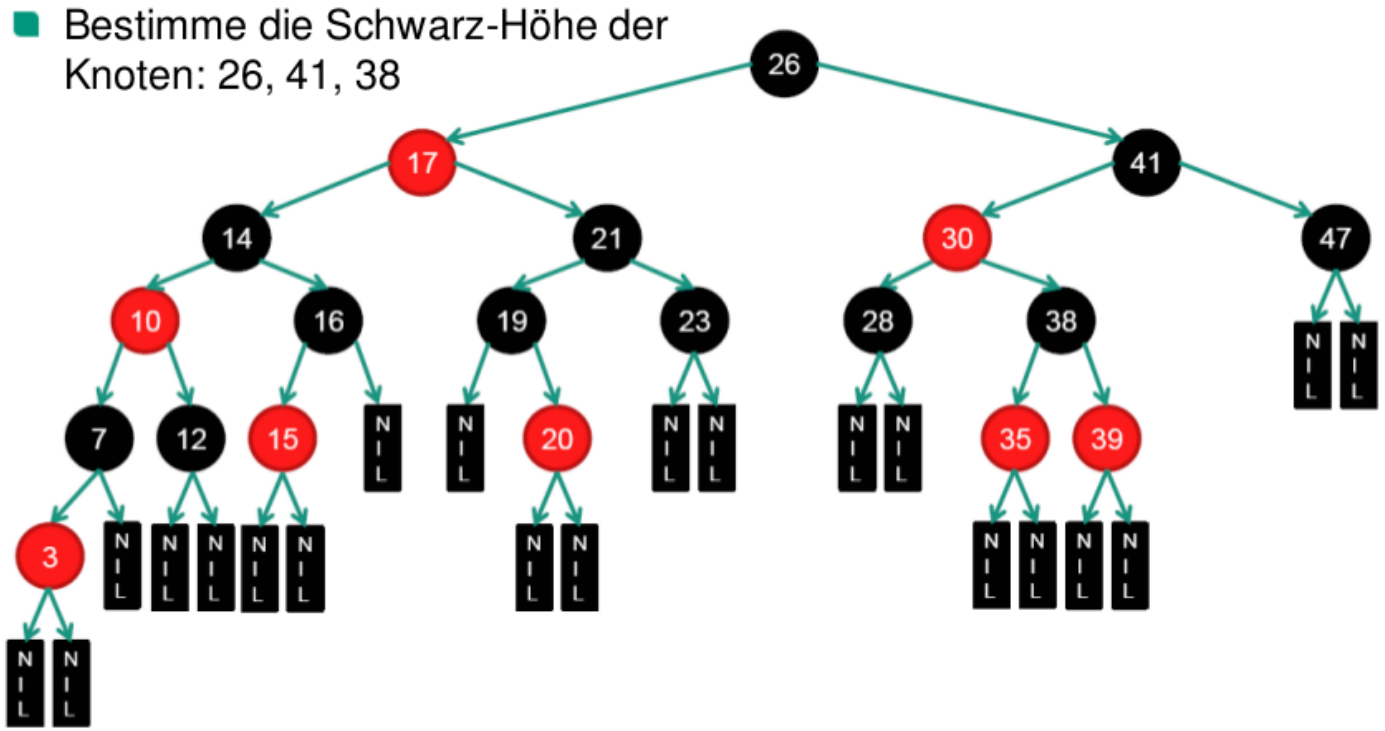
1. Jeder Knoten ist entweder rot oder schwarz.
2. Die Wurzel ist schwarz.
3. Jedes Blatt NIL ist schwarz.
4. Wenn ein Knoten rot ist, so sind beide Kinder schwarz.
5. Für jeden Knoten gilt, dass alle Pfade vom Knoten zu einem Blatt die selbe Anzahl schwarzer Knoten beinhalten.

Die **Schwarz-Höhe** eines Knotens ist die Anzahl schwarzer Knoten auf einem Pfad vom Knoten selbst bis zu einem Blatt.

Aufgabe

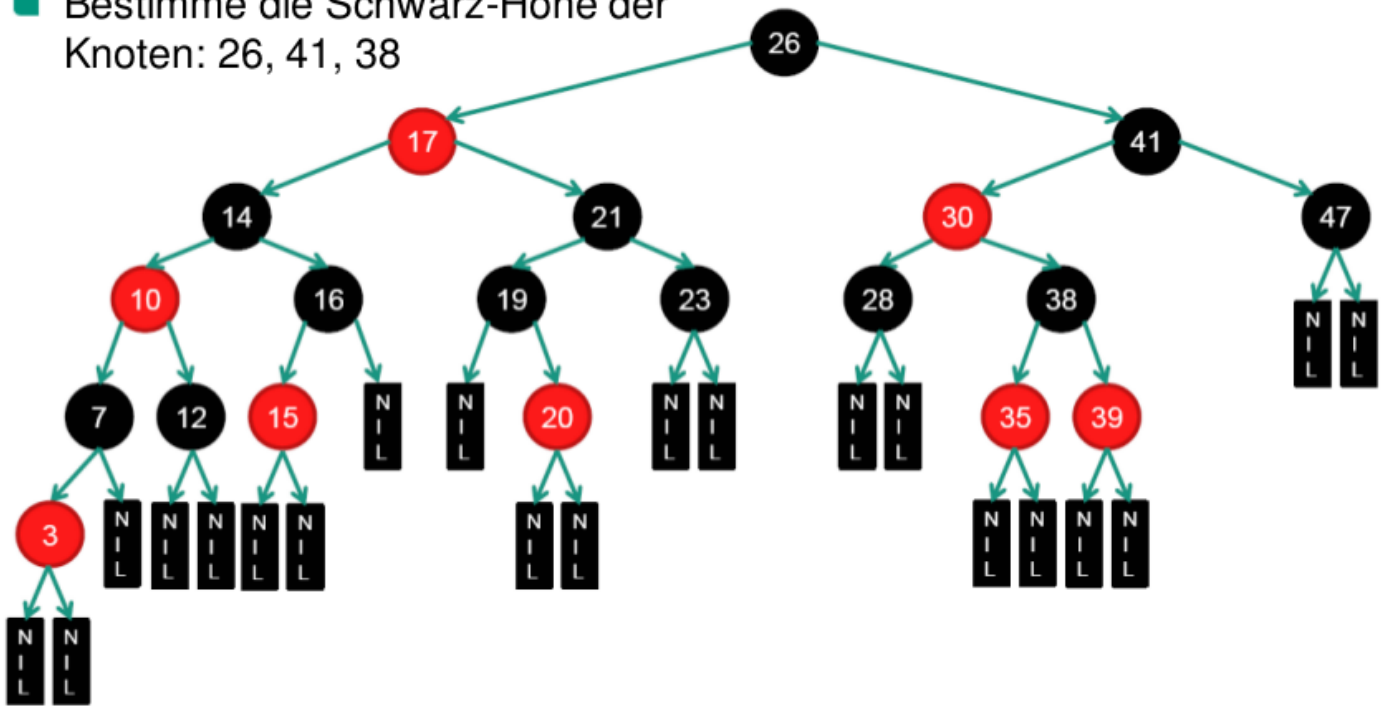
Schwarz-Höhe

Bestimme die Schwarz-Höhe der Knoten: 26, 41, 38.



Lösung

- Bestimme die Schwarz-Höhe der Knoten: 26, 41, 38



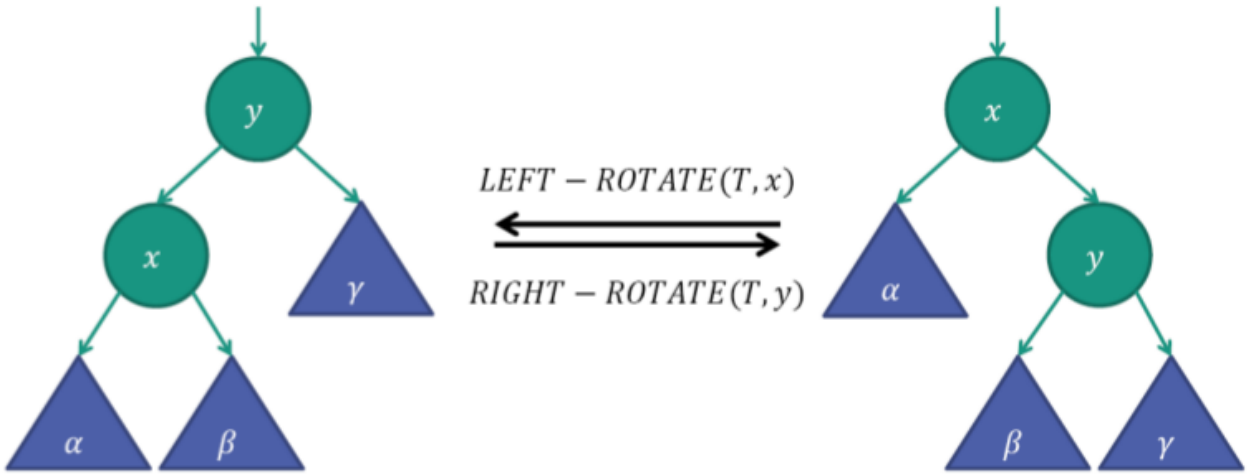
$$bh(26) = 3, bh(41) = 2, bh(38) = 1$$

Rot-Schwarz-Baum

Eigenschaften müssen nach Operationen häufig wieder hergestellt werden durch

- Umfärben von Knoten
- Links-/Rechtsrotation

Rotation



Rot-Schwarz-Baum: Einfügen

- Grundprinzip
  - Einfügen wie bei binärem Suchbaum
  - Eingefügten Knoten rot färben (Bei Wurzel schwarz)
  - Einen von sechs Fällen identifizieren
  - Eigenschaften wiederherstellen
- Fälle
  - k.vater ist linkes Kind von k.vater.vater
    - (E1) Der Onkel von k ist rot
    - (E2) Der Onkel von k ist schwarz und k ist rechtes Kind
    - (E3) Der Onkel von k ist schwarz und k ist linkes Kind
  - k.vater ist rechtes Kind von k.vater.vater
    - (E4) Der Onkel von k ist rot
    - (E5) Der Onkel von k ist schwarz und k ist linkes Kind
    - (E6) Der Onkel von k ist schwarz und k ist rechtes Kind

Rot-Schwarz-Bäume: Einfügen

- (E1) Der Onkel von k ist rot
  - k.vater und Onkel schwarz färben
  - k.vater.vater rot färben
  - Erneut bei k.vater.vater Eigenschaften prüfen und ggf. wiederherstellen
- (E2) Der Onkel von k ist schwarz und k ist rechtes Kind
  - Linksrotation
  - Bei ehemaligen k.vater Fall 3 prüfen und Eigenschaften wiederherstellen
- (E3) Der Onkel von k ist schwarz und k ist linkes Kind
  - k.vater.vater rot färben
  - k.vater schwarz färben
  - Rechtsrotation um k.vater.vater
- Fälle (E4)-(E6) analog (alles spiegelverkehrt)

Aufgabe

Einfügen in Rot-Schwarz-Bäumen

Füge folgende Schlüssel nacheinander in einen Rot-Schwarz Baum ein: 45, 32, 13, 67, 20, 5, 90

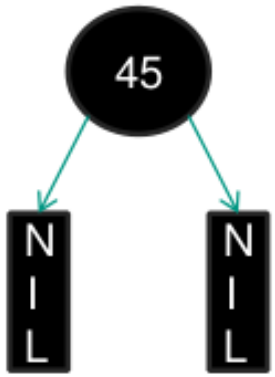
Lösung

45, 32, 13, 67, 20, 5, 90

- Eigenschaft 2 und 3 verletzt
- → *schwarz* färben
- → NIL-Blattknoten einfügen

Lösung

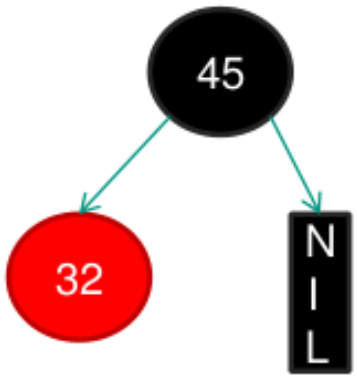
45, 32, 13, 67, 20, 5, 90



- Eigenschaften erfüllt

Lösung

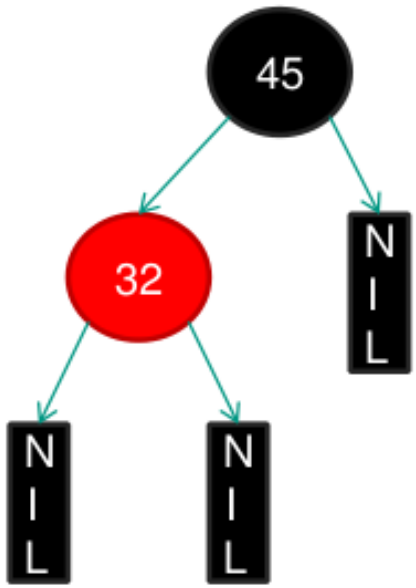
45, 32, 13, 67, 20, 5, 90



- Eigenschaft 3 verletzt
- → NIL-Blattknoten einfügen

Lösung

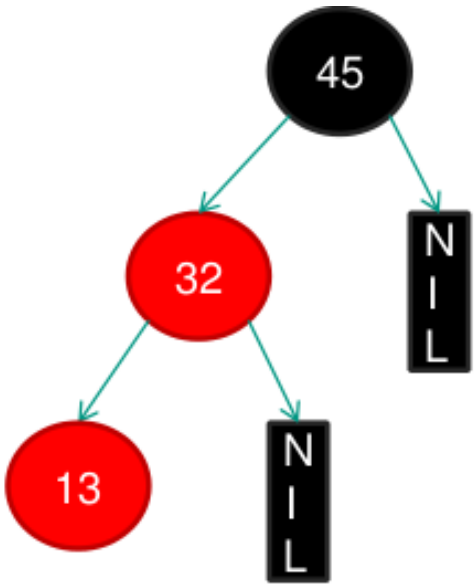
45, 32, 13, 67, 20, 5, 90



- Eigenschaften erfüllt

Lösung

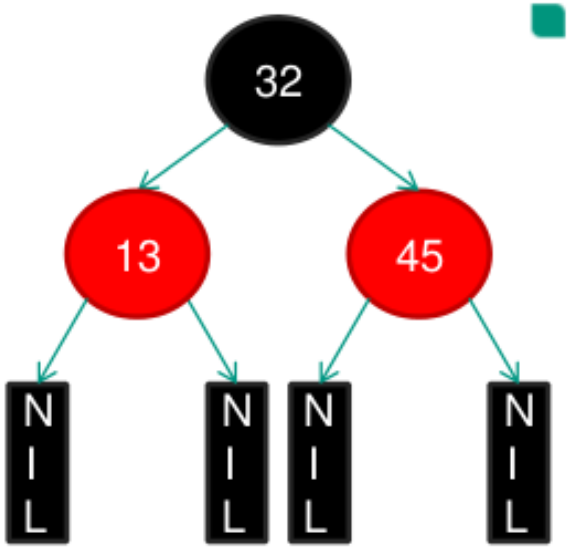
45, 32, 13, 67, 20, 5, 90



- Eigenschaften 3 und 4 verletzt
- → (E3)
- → NIL-Blattknoten einfügen

Lösung

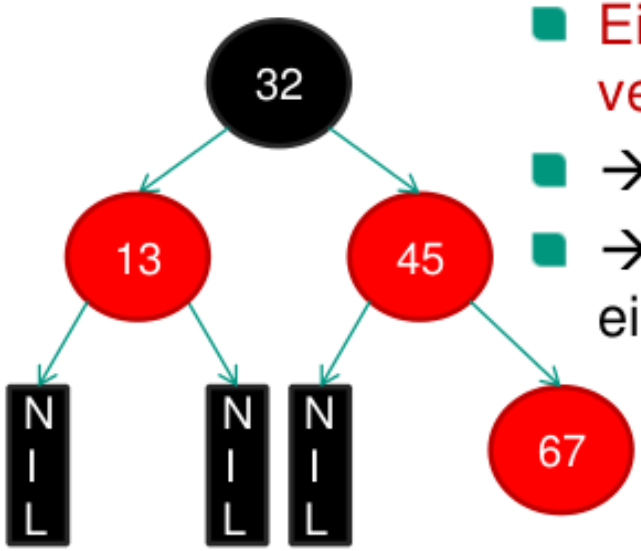
45, 32, 13, 67, 20, 5, 90



- Eigenschaften erfüllt

Lösung

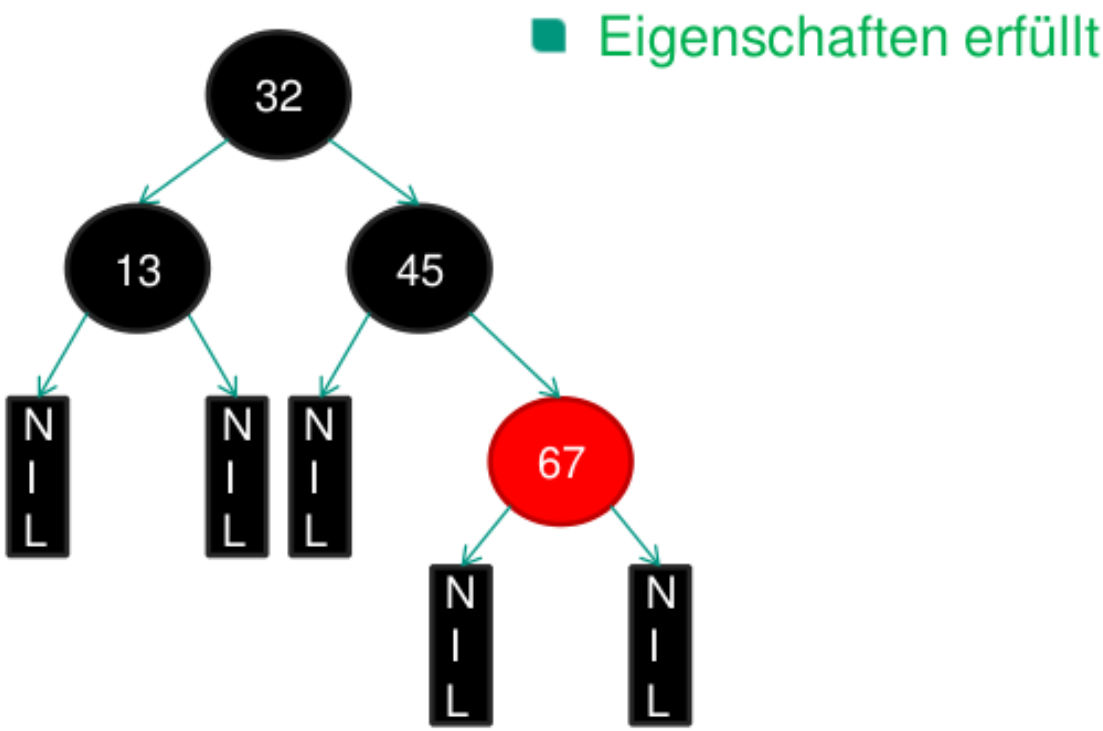
45, 32, 13, 67, 20, 5, 90



- Eigenschaften 3 und 4 verletzt
- → (E4)
- → NIL-Blattknoten einfügen

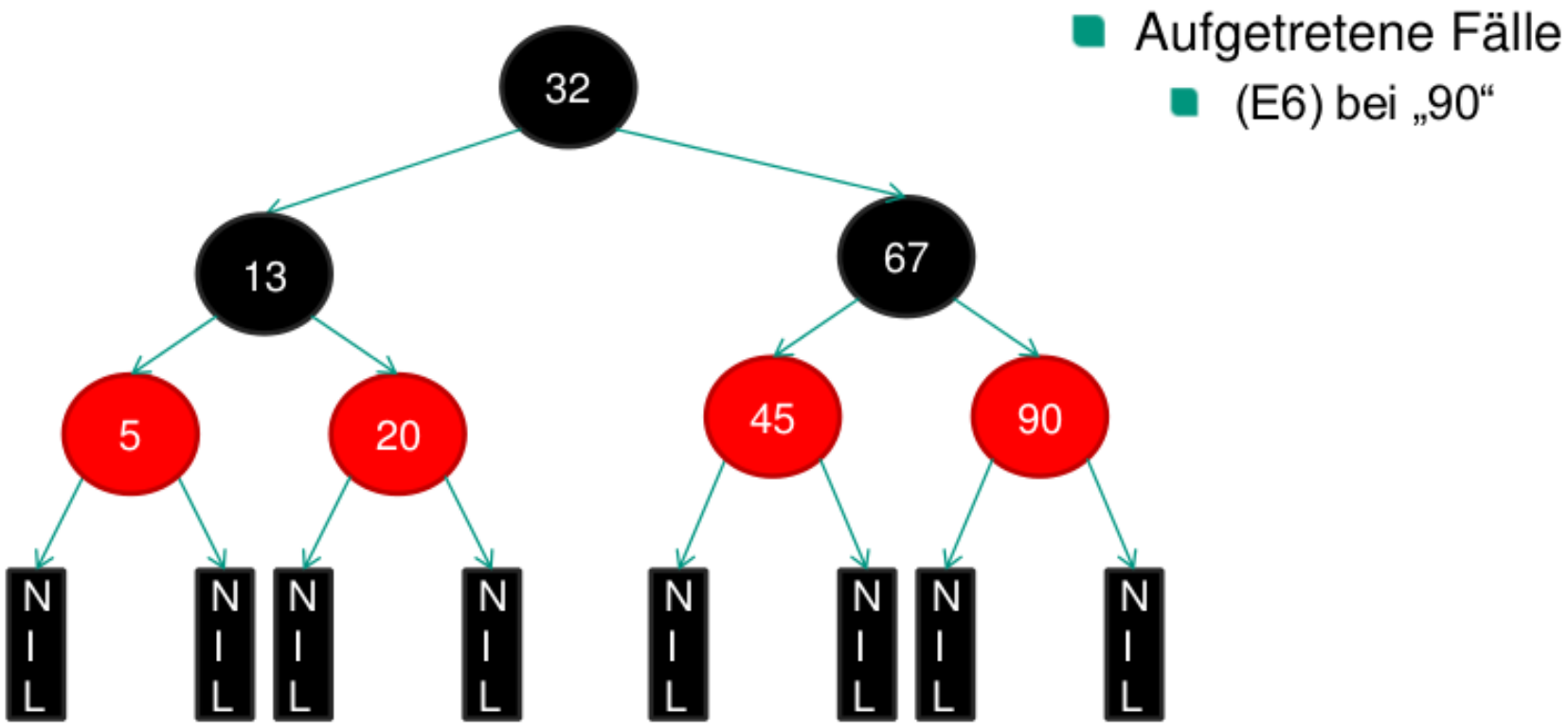
Lösung

45, 32, 13, 67, 20, 5, 90



Lösung

45, 32, 13, 67, 20, 5, 90



Löschen in Rot-Schwarz-Bäumen

- Analog zu binären Suchbäumen
  - Wenn zu löschender Knoten zwei Kinder hat
    - Kopiere Minimum m aus rechtem Teilbaum in den zu löschenden Knoten
    - Wähle m als zu löschenden Knoten k
  - → Vereinfachung des Problems auf Löschung eines Knotens mit nur einem Kind x^*
- Falls k rot → keine Verletzung der Eigenschaften
- Falls k schwarz und x rot → färbe x schwarz
- Falls k und x schwarz → doppelte Schwarzfärbung von x
  - Führt zu Verletzung von Eigenschaft 1
  - → einen von acht Fällen identifizieren

Löschen in Rot-Schwarz-Bäumen

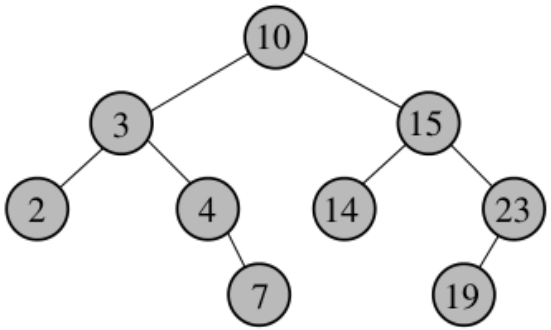
- x ist linkes Kind von $x.vater$
  - (L1) Bruder w ist rot
  - (L2) Bruder w und beide Kinder von w sind schwarz
  - (L3) Bruder w und dessen rechtes Kind ist schwarz, linkes Kind ist rot
  - (L4) Bruder w ist schwarz und rechtes Kind ist rot
- x ist rechtes Kind von $x.vater$
  - (L5) Bruder w ist rot
  - (L6) Bruder w und beide Kinder von w sind schwarz
  - (L7) Bruder w und dessen linkes Kind ist schwarz, rechtes Kind ist rot
  - (L8) Bruder w ist schwarz und linkes Kind ist rot

- Fälle (L5)-(L8) sind wieder, wie bei den Einfügeoperationen, analog zu (L1)-(L4) in Spiegelung zu behandeln

Kreativaufgabe

Baumentwurf

Gegeben sei ein binärer Suchbaum, der auch in den inneren Knoten Elemente speichert. Jeder Knoten habe drei Zeiger, zwei Kindzeiger und einen Zeiger auf den Elterknoten (u.U. sind einige davon Nullzeiger). Die Elemente sind jedoch nicht zusätzlich in einer Liste enthalten, d.h. das nächstgrößere Element kann nicht direkt gefunden werden. Es sei ein ∞-Dummy enthalten.



Kreativaufgabe 

1. Schwierigkeitsstufe

Implementiere find(x), also eine Funktion die einen Schlüssel x nimmt und entweder das passende Element aus der Datenstruktur zurückgibt, oder $\perp$ falls kein passendes Element enthalten ist. Die Laufzeit sollte in $\mathcal{O}$(Baumtiefe) liegen.

Kreativaufgabe 

2. Schwierigkeitsstufe

Implementiere locate(x) wie aus der Vorlesung. Die Laufzeit sollte in $\mathcal{O}$(Baumtiefe) liegen.

Kreativaufgabe 

3. Schwierigkeitsstufe

Implementiere locate(x) wie aus der Vorlesung. Die Baumknoten haben in diesem Fall aber keine Zeiger mehr auf ihre Elterknoten, und es soll nur $\mathcal{O}(1)$ zusätzlicher Speicher verwendet werden. Die Laufzeit sollte in $\mathcal{O}$(Baumtiefe) liegen.

Bis zum nächsten Mal.