Grundzüge der Informatik 1

Vorlesung 18

Überblick Vorlesung

Überblick

- Wiederholung
  - Einfügen in Rot-Schwarz-Bäume
- Löschen in Rot-Schwarz-Bäumen

Datenstrukturen

Rot-Schwarz-Bäume

- Balancierter Suchbaum
- Nach Einfügen/Löschen wird die Struktur des Suchbaums so modifiziert, dass eine Höhe von O(log n) garantiert wird
- Rebalancierung nach Einfügen/Löschen wird in O(log n) Zeit möglich sein
- Damit sind Operationen Suchen, Einfügen und Löschen in O(log n) Zeit möglich

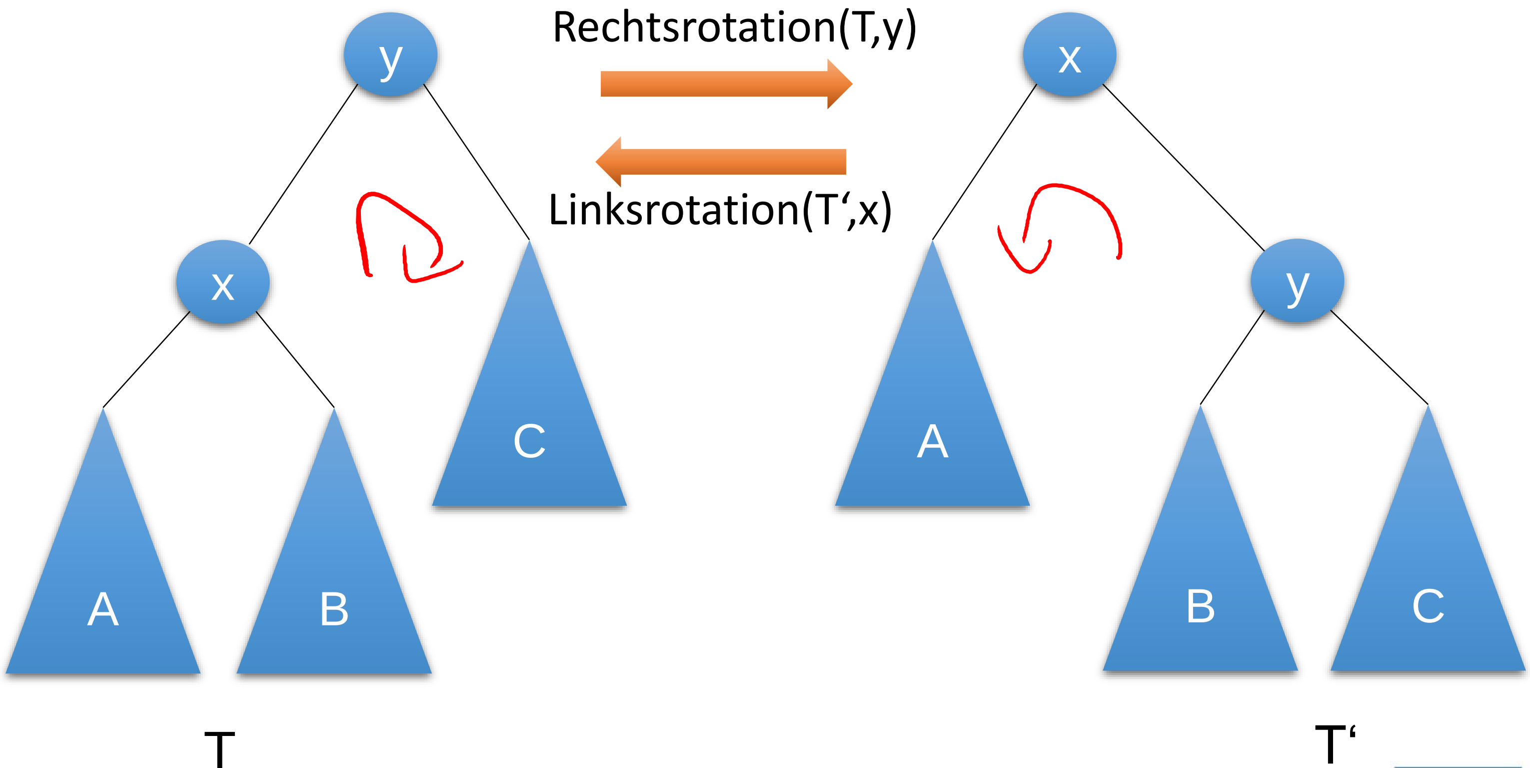
Datenstrukturen

Die Rot-Schwarz-Eigenschaften

- Jeder Knoten ist rot oder schwarz
- Die Wurzel ist schwarz
- Jedes Blatt ist schwarz
- Wenn ein Knoten rot ist, dann sind seine Kinder schwarz
- Für jeden Knoten v haben alle Pfade vom Knoten zu den Blättern im Unterbaum mit Wurzel v dieselbe Anzahl schwarzer Knoten

Datenstrukturen

Rotationen



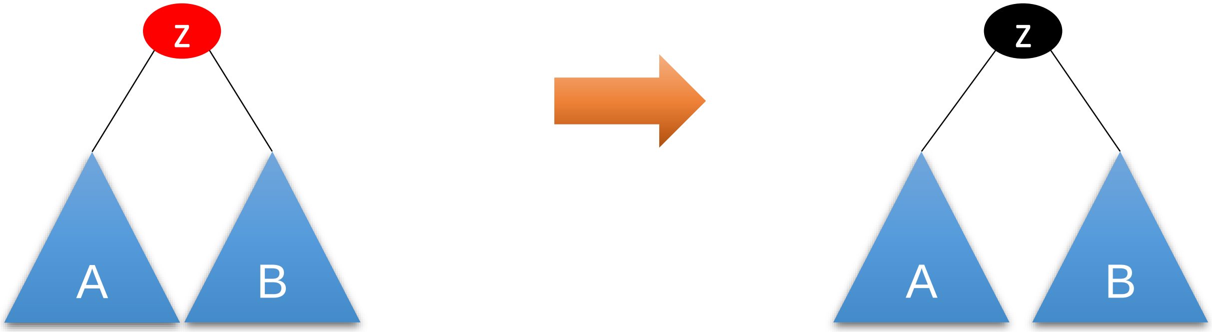
Datenstrukturen

Überblick: Wiederherstellen der Rot-Schwarz-Eigenschaften

- Starte mit eingefügtem Knoten z
- Stelle die Eigenschaft lokal wieder her, so dass sie nur von einem Knoten verletzt werden kann, der näher an der Wurzel ist
- Bei der Wurzel angekommen wird diese schwarz gefärbt

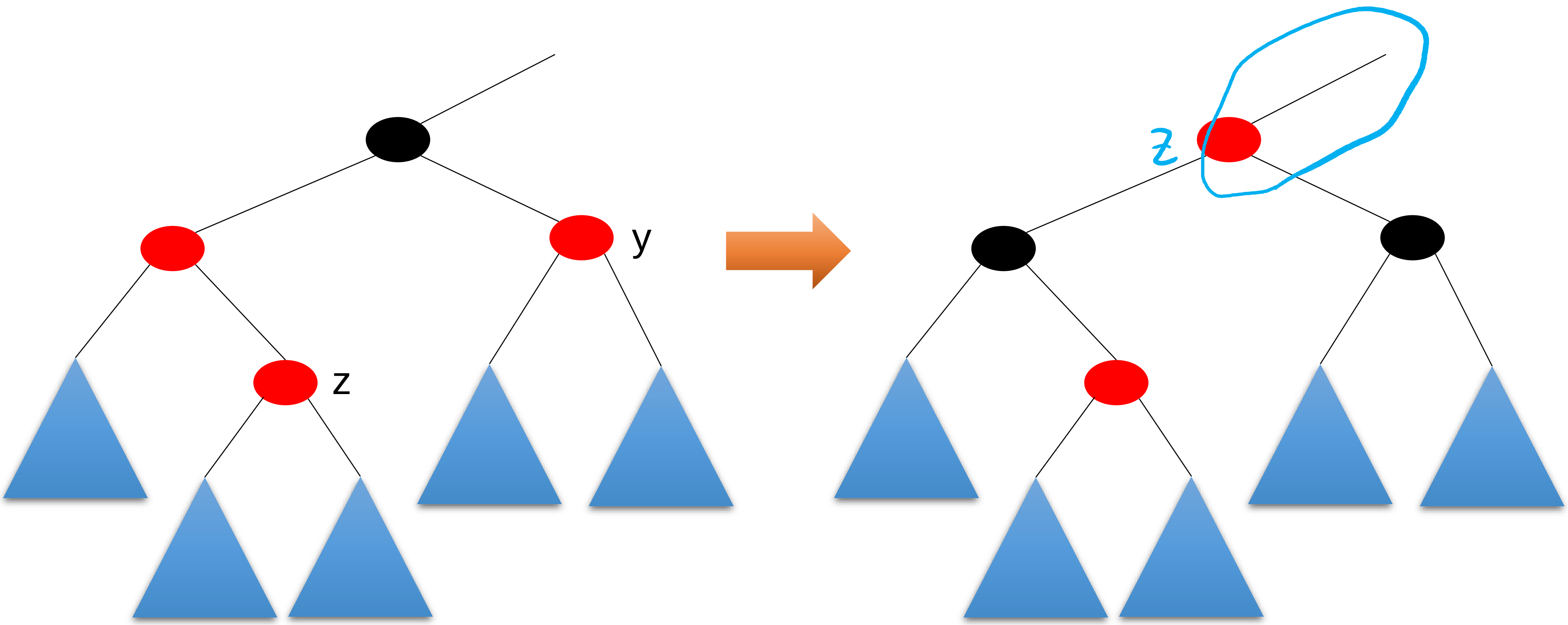
Datenstrukturen

Fall (1) (z ist die Wurzel)



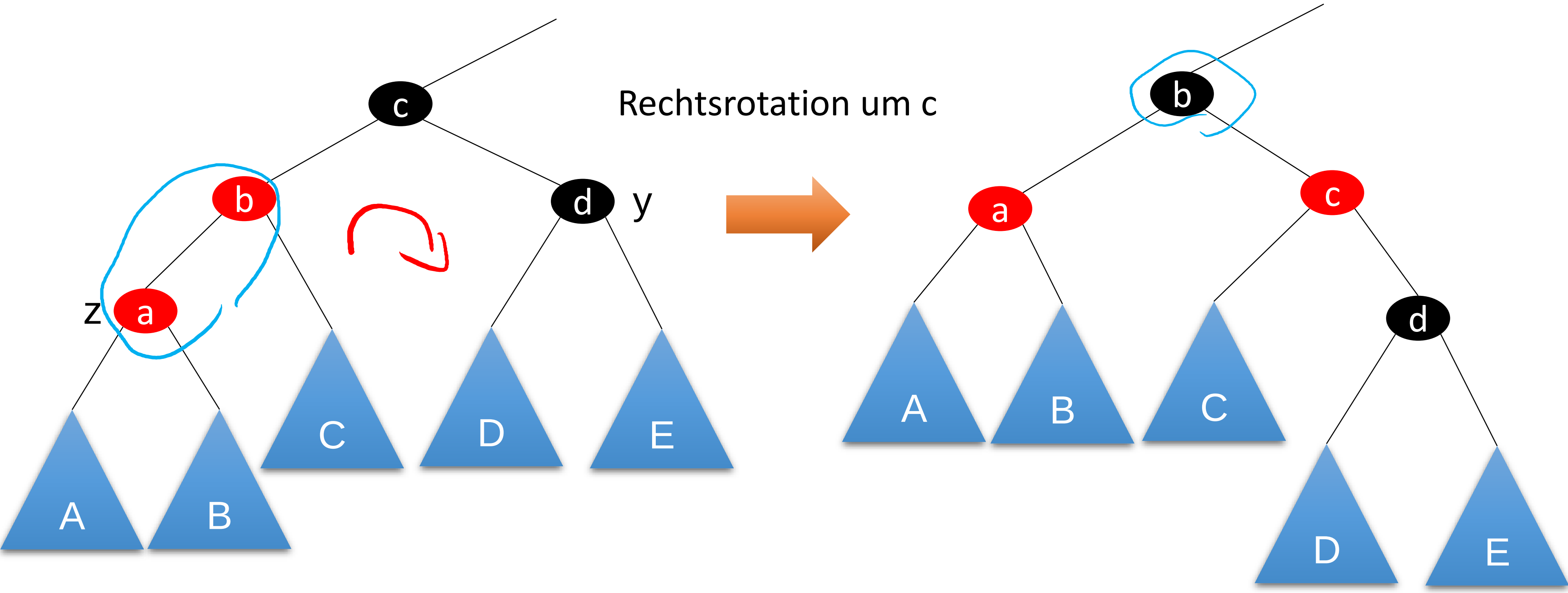
Datenstrukturen

Fall (2) (Onkel von z ist rot)



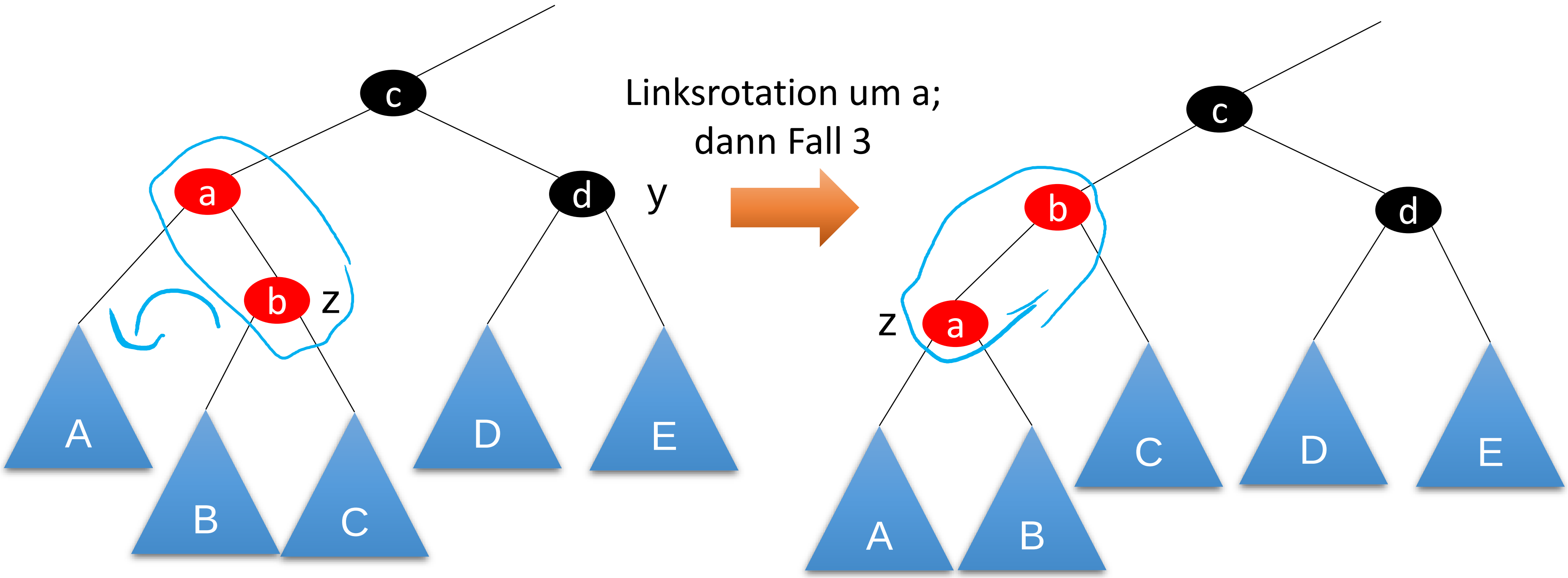
Datenstrukturen

Fall (3) (Onkel von z ist schwarz, z ist linkes Kind und parent(z) ist linkes Kind)



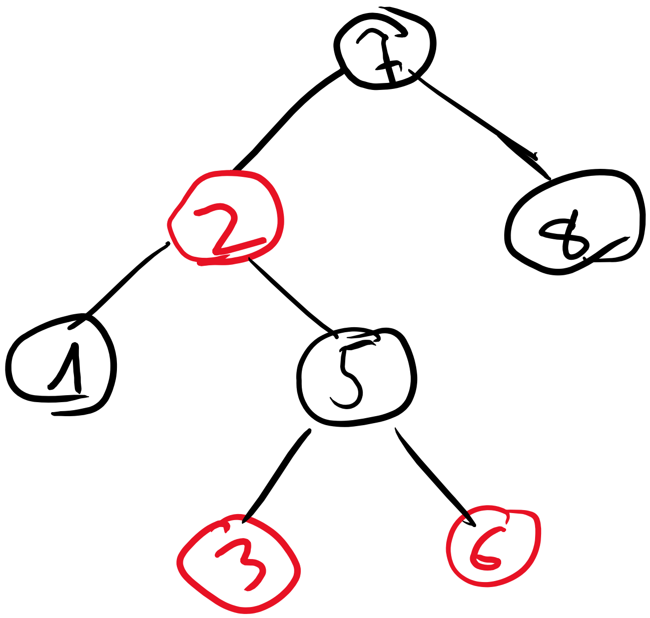
Datenstrukturen

Fall (4) (Onkel von z ist schwarz, z ist rechtes Kind und parent(z) ist linkes Kind)



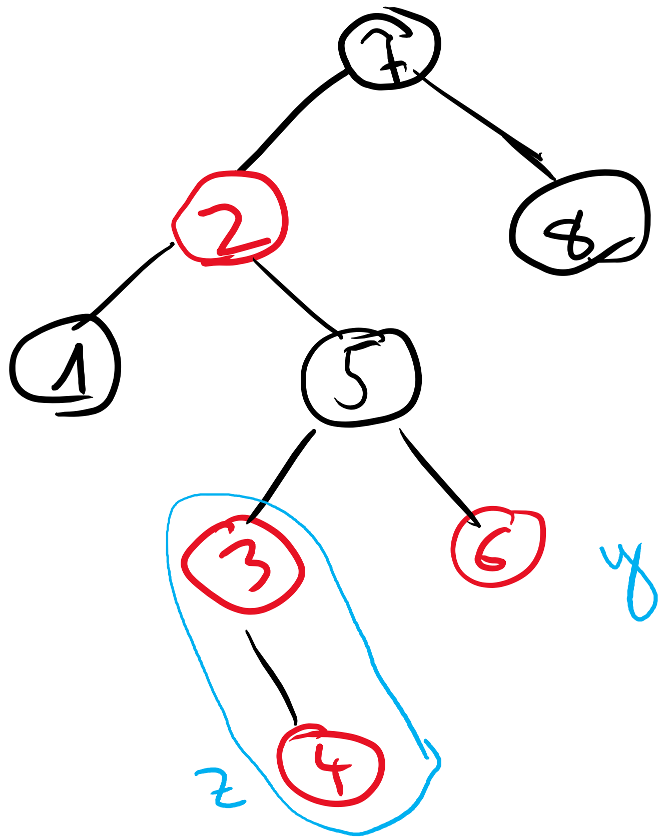
Datenstrukturen

Beispiel: Einfügen in Rot-Schwarz-Baum



Datenstrukturen

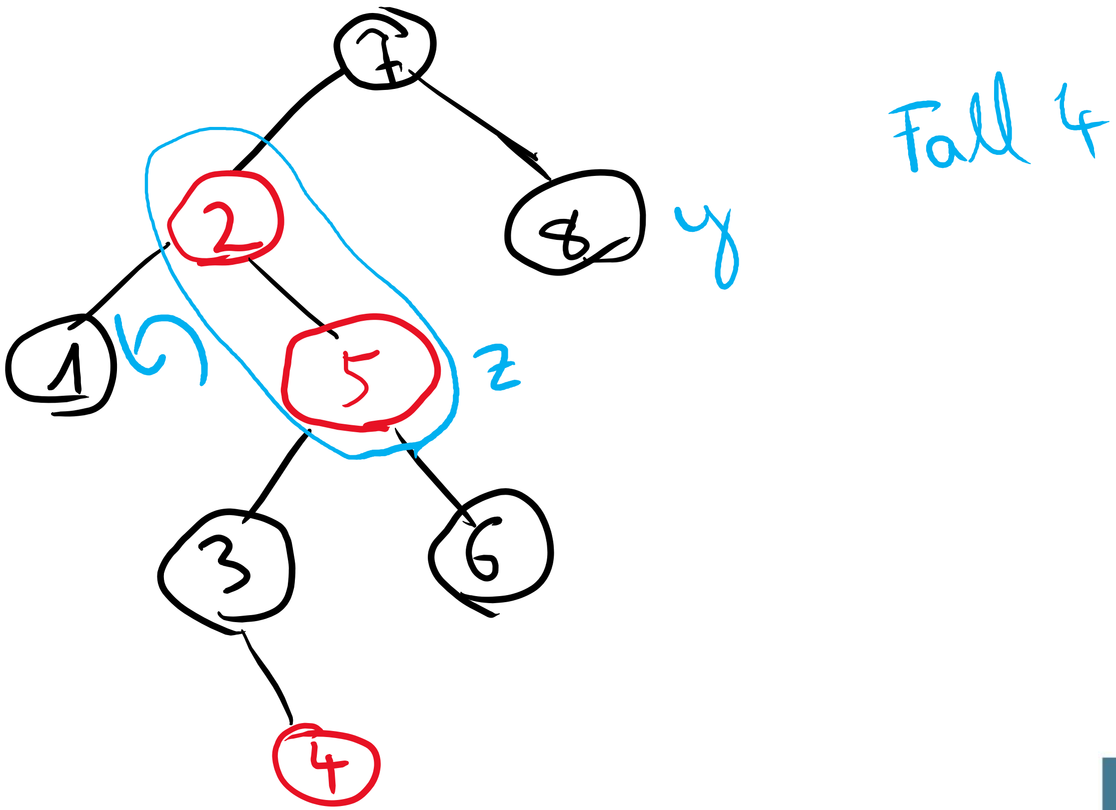
Beispiel: Einfügen in Rot-Schwarz-Baum



Fall 2

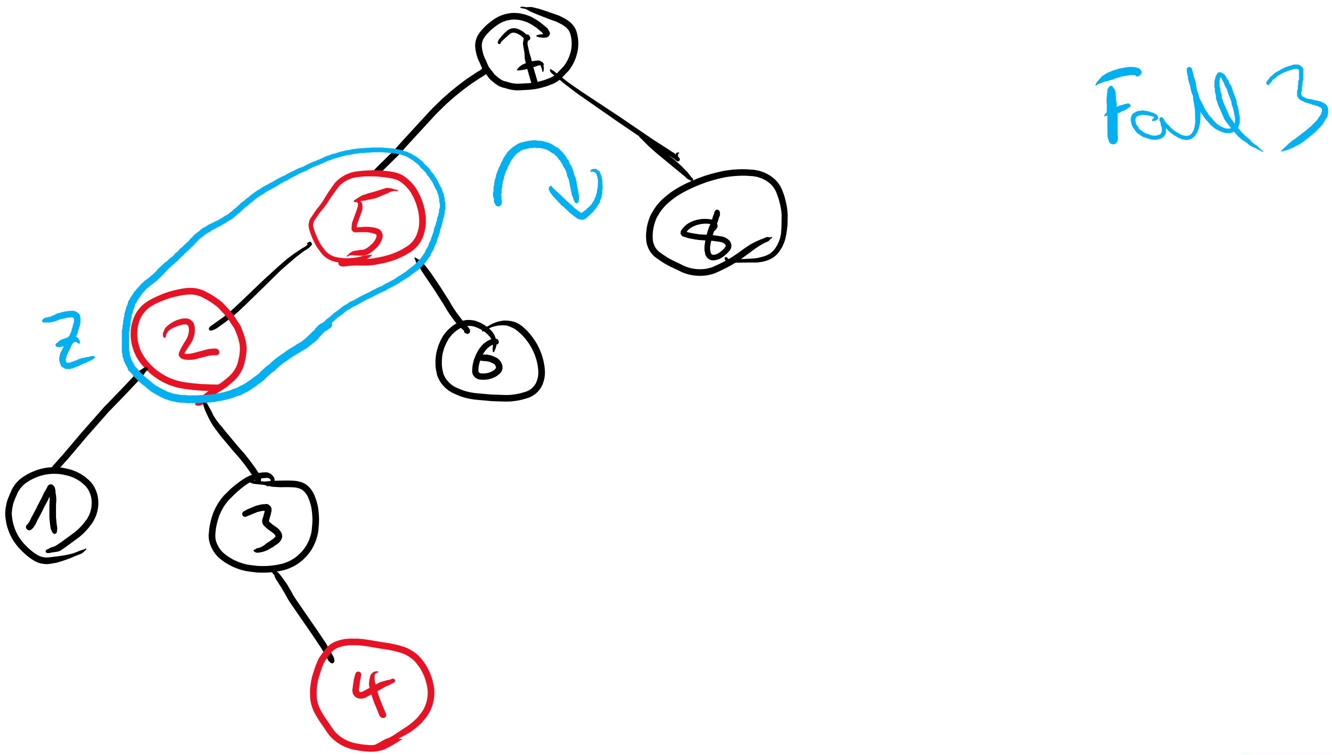
Datenstrukturen

Beispiel: Einfügen in Rot-Schwarz-Baum



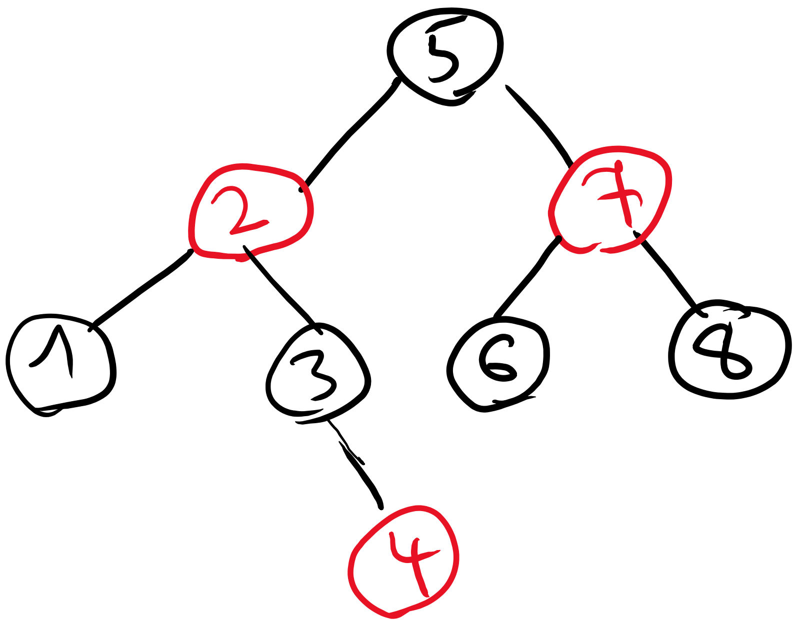
Datenstrukturen

Beispiel: Einfügen in Rot-Schwarz-Baum



Datenstrukturen

Beispiel: Einfügen in Rot-Schwarz-Baum

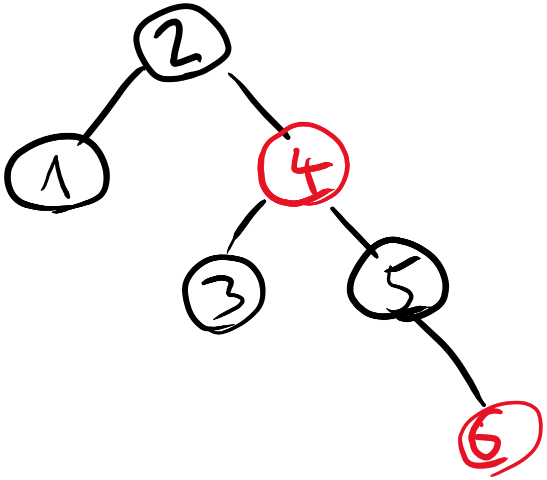


Universität zu Köln

Datenstruturen

Aufgabe

- Fügen Sie den Schlüssel 7 in den unten stehenden Rot-Schwarz-Baum ein

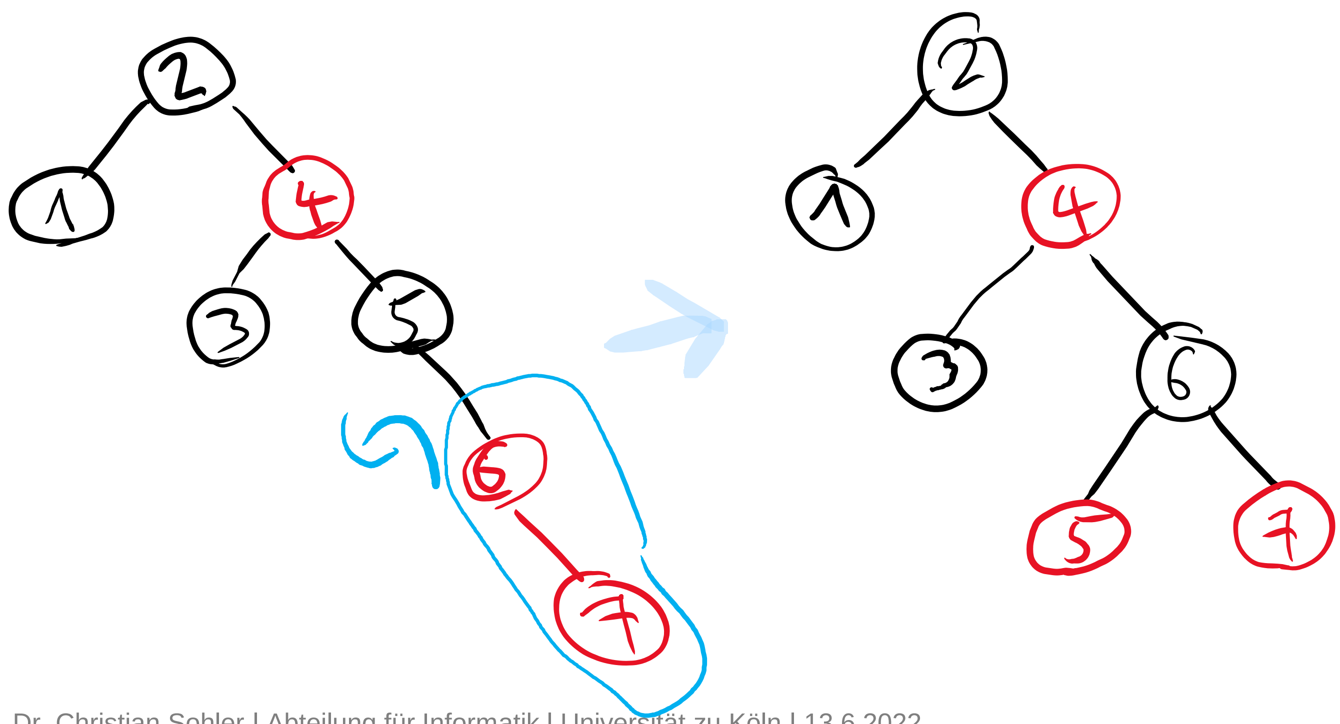


Universität zu Köln

Datenstruturen

Aufgabe

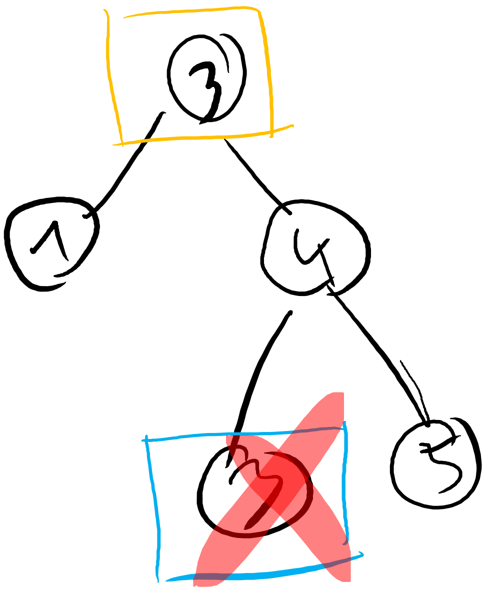
- Fügen Sie den Schlüssel 7 in den unten stehenden Rot-Schwarz-Baum ein



Datenstrukturen

Löschen in Rot-Schwarz-Bäumen

- Zunächst Löschen wie in binären Suchbäumen
- Dann Wiederherstellen der Rot-Schwarz-Eigenschaften



Datenstrukturen

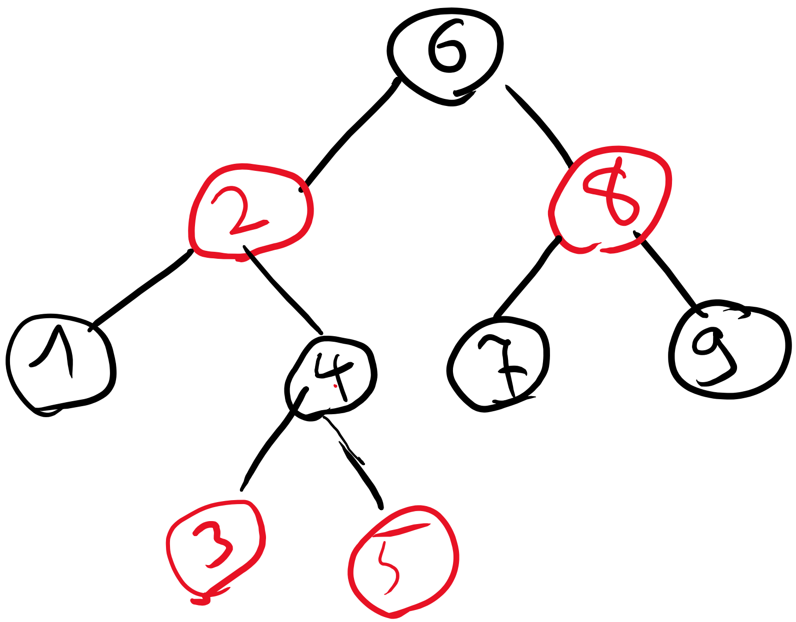
RS-Löschen(T,z)   * Zu löschender Knoten wird übergeben

1. **if** left[z]=NIL[T] or right[z]=NIL[T] **then** y=z
2. **else** y=NachfolgerSuche(z)
3. **if** left[y]≠NIL[T] **then** x=left[y] **else** x=right[y]
4. parent[x]=parent[y]
5. **if** parent[y]=NIL[T] **then** root[T]=x
6. **else if** y=left[parent[y]] **then** left[parent[y]]=x **else** right[parent[y]]=x
7. key[z]=key[y]
8. **if** color[y]=schwarz **then** RS-Löschen-Fix(T,x)
9. parent[NIL[T]]=NIL
10. **delete** y

Datenstrukturen

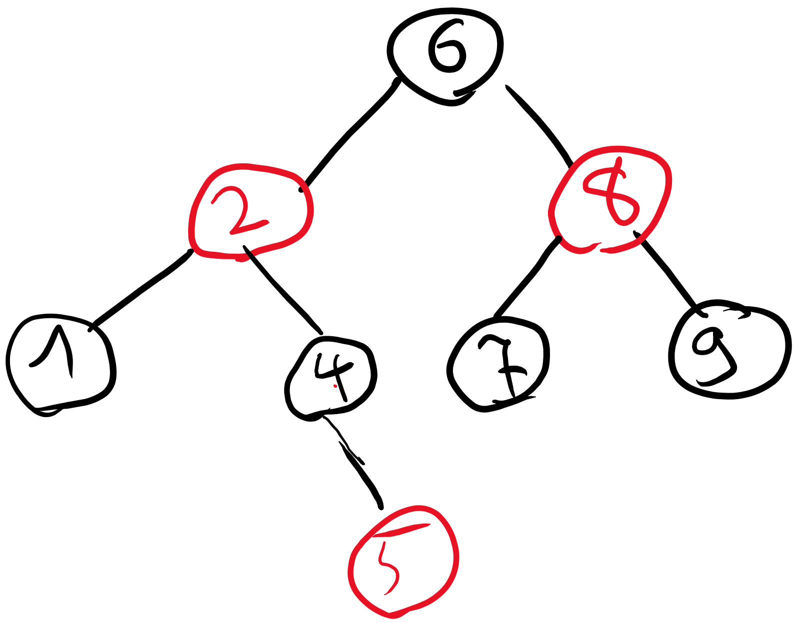
Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix

Löschen von 3



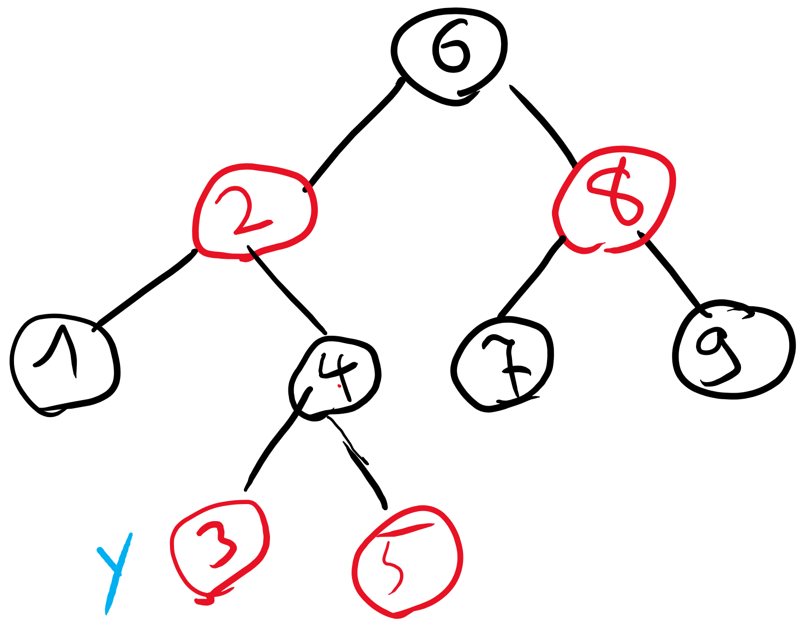
Datenstrukturen

Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix



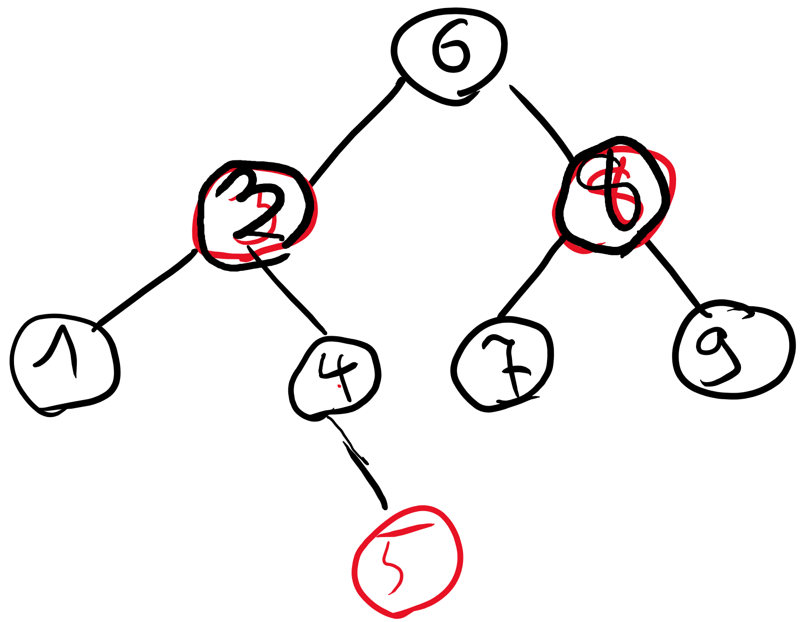
Datenstrukturen

Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix



Datenstrukturen

Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix



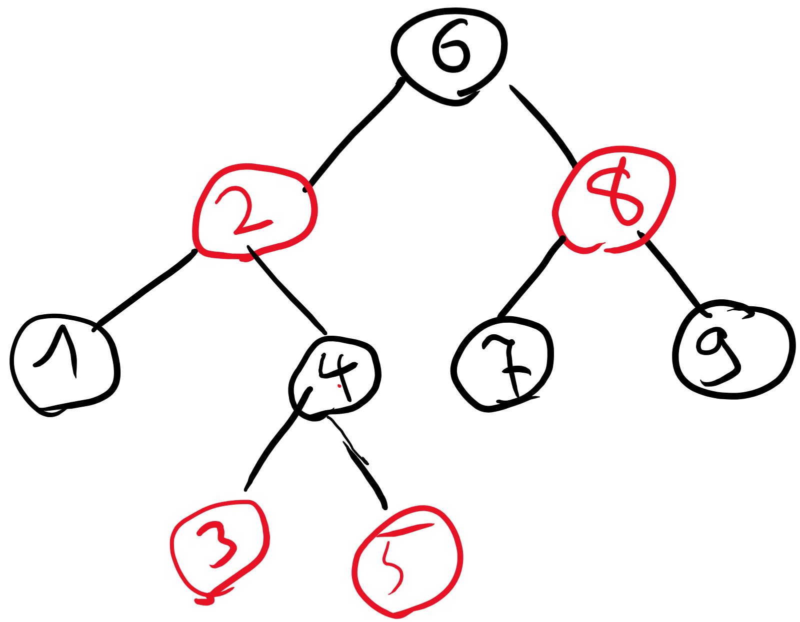
Datenstrukturen

Beobachtung

- Wenn der aus der Baumstruktur entfernte Knoten y rot ist, dann ist der resultierende Baum ein Rot-Schwarz-Baum

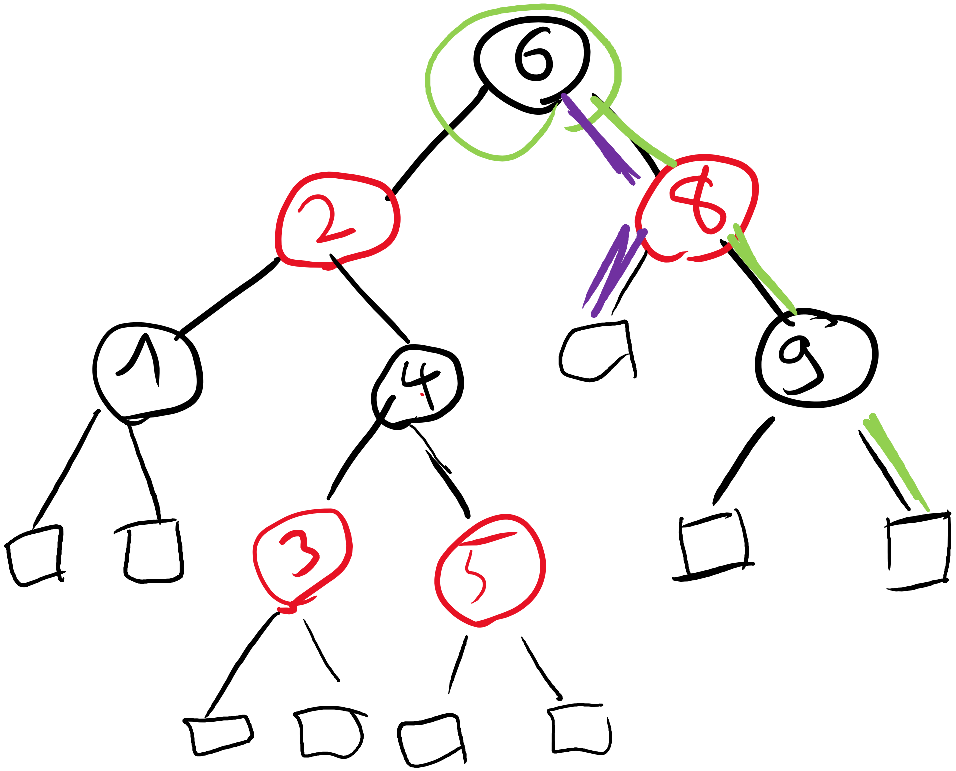
Datenstrukturen

Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix



Datenstrukturen

Beispiel: Löschen in Rot-Schwarz-Baum ohne RS-Löschen-Fix



Datenstrukturen

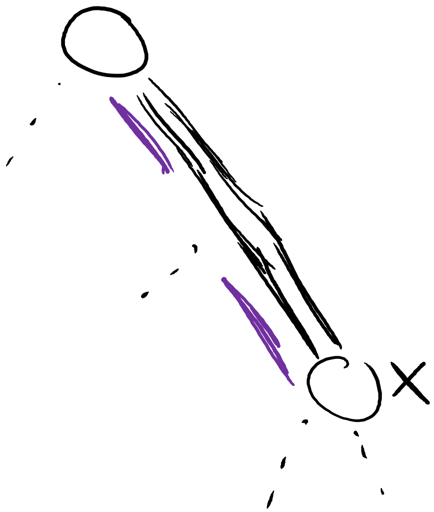
Einfacher Fall

- Wenn der Knoten x rot ist, können wir diesen schwarz färben und der resultierende Baum ist ein Rot-Schwarz-Baum
- Im Folgenden gehen wir davon aus, dass x schwarz ist

Datenstrukturen

Betrachtungsweise

- Der Knoten x ist zu Beginn das einzige Kind von y oder NIL[T]
- Der Knoten x zählt wie zwei schwarze Knoten



Datenstrukturen

Betrachtungsweise

- Der Knoten x ist zu Beginn das einzige Kind von y oder NIL[T]
- Der Knoten x zählt wie zwei schwarze Knoten

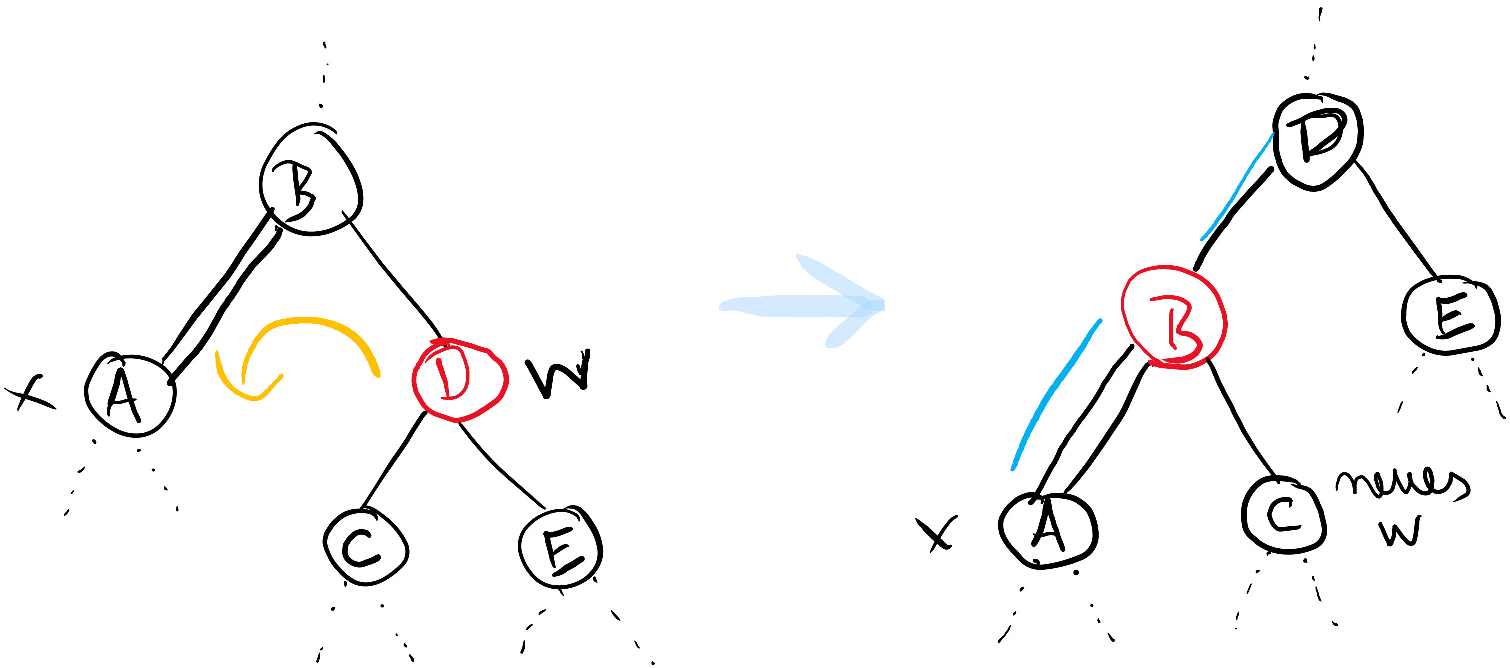
Einfacher Fall

- parent[x] ist rot, das andere Kind von parent[x] ist schwarz und seine Kinder ebenfalls
- Färbe parent[x] schwarz und das andere Kind von parent[x] rot

Universität zu Köln

Datenstrukturen

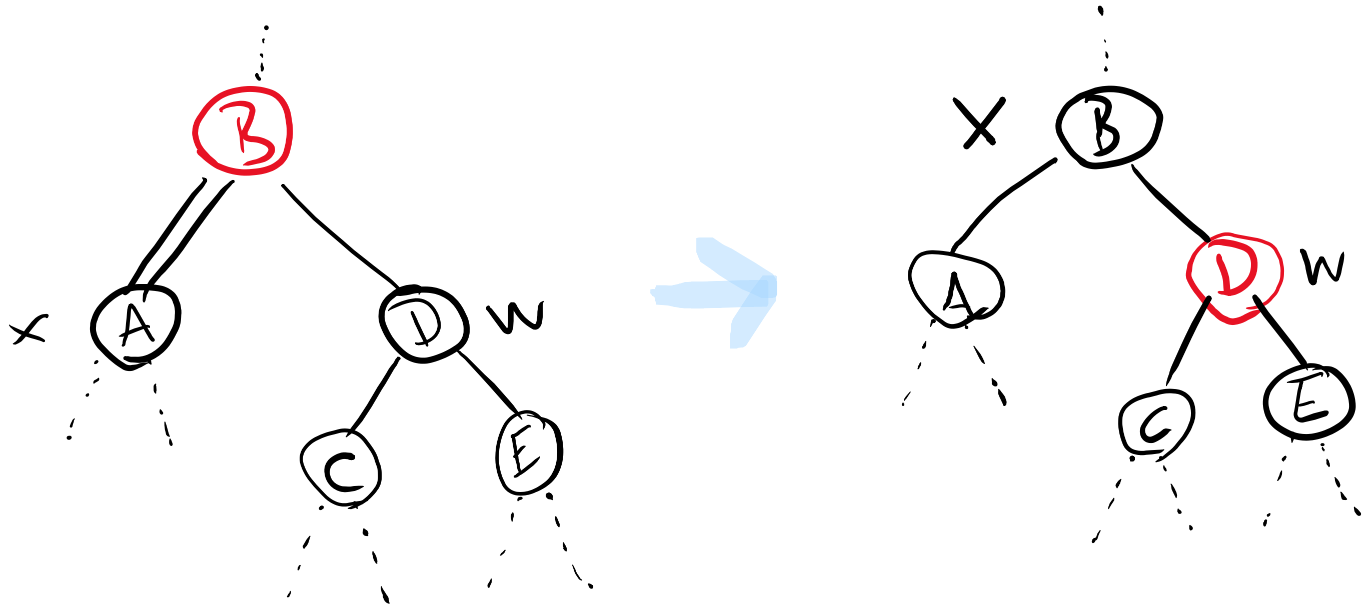
Fall 1: Geschwisterknoten von x ist rot



Universität zu Köln

Datenstrukturen

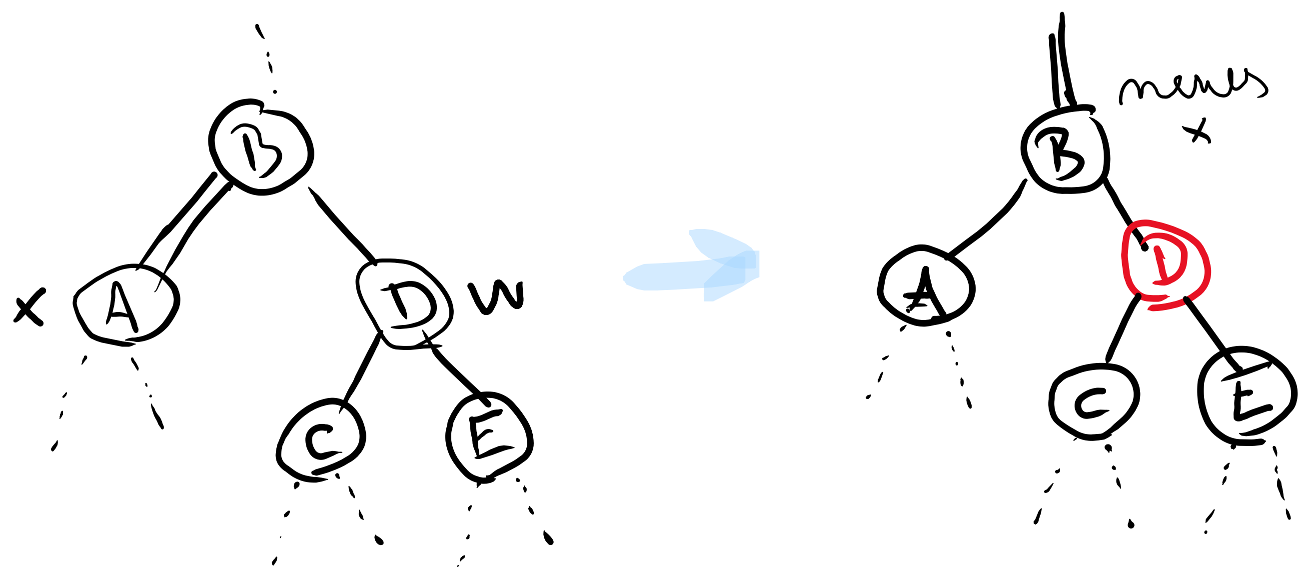
Fall 2a: Geschwisterknoten w von x ist schwarz; parent[x] ist rot;
Kinder von w sind schwarz



Universität zu Köln

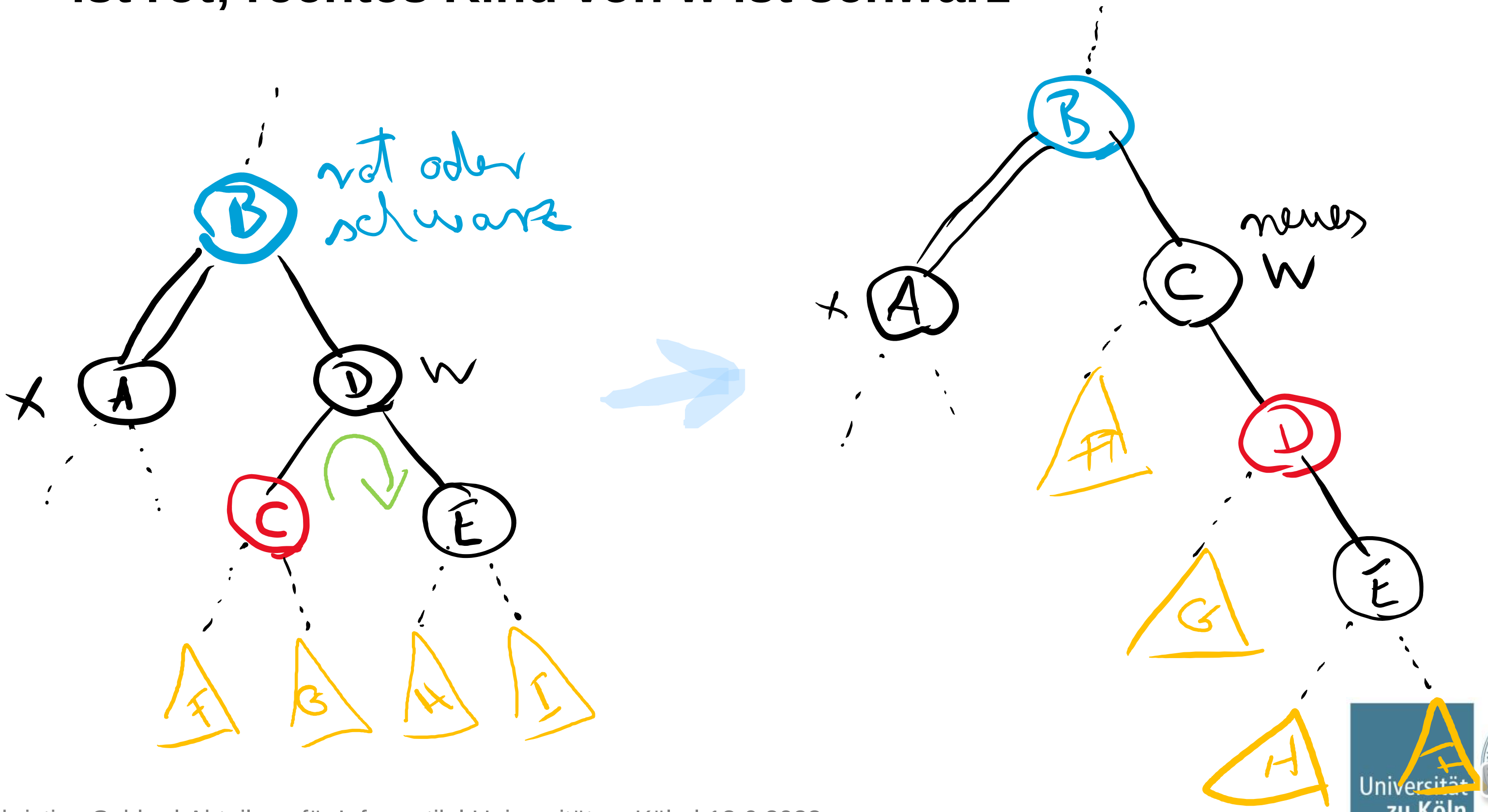
Datenstrukturen

Fall 2b: Geschwisterknoten w von x ist schwarz; parent[x] ist schwarz; Kinder von w sind schwarz



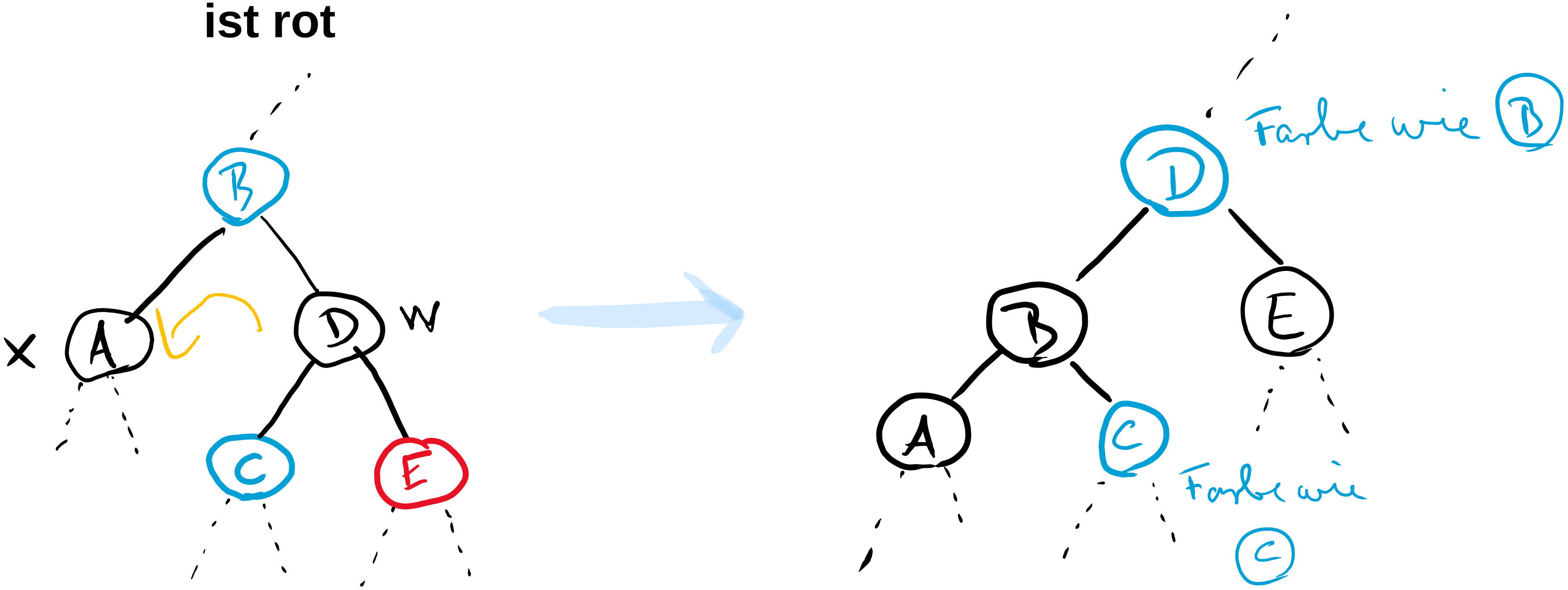
Datenstrukturen

Fall 3: Geschwisterknoten w von x ist schwarz; linkes Kind von w ist rot; rechtes Kind von w ist schwarz



Datenstrukturen

Fall 4: Geschwisterknoten w von x ist schwarz; rechtes Kind von w ist rot



Datenstrukturen

Symmetrische Fälle

- Genau wie beim Einfügen gibt es zu den Fällen 1-4 wieder spiegelverkehrte Fälle, die wir hier nicht noch einmal auflisten

Fall 1:

1. color[w]=schwarz
2. color[parent[x]]=rot
3. Linksrotation(T,parent(x))
4. w=right[parent[x]]

Fall 2:

1. color[w]=rot
2. x=parent[x]

Fall 3:

1. color[left[w]]=schwarz
2. color[w]=rot
3. Rechtsrotation(T,w)
4. w=right[parent[x]]

RS-Löschen-Fix(T,x)

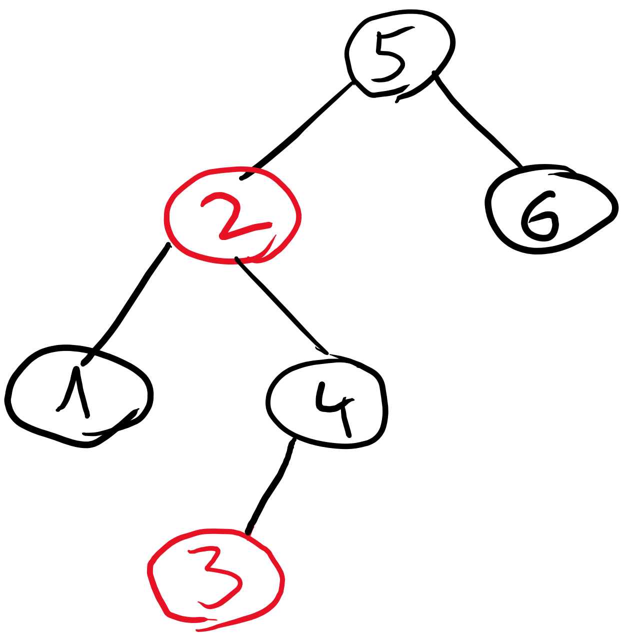
```
while x≠root[T] and color[x]=schwarz do
    if x=left[parent[x]] then
        w = right[parent[x]]
        if color[w]=rot then  ***Fall 1 ***
        if color[left[w]]=schwarz and color[right[w]]=schwarz then *** Fall 2 ***
        else
            if color[right[w]]=schwarz then *** Fall 3 ***
            *** Fall 4 ***
    else (symmetrischer Fall)
color[x] = schwarz
```

Fall 4:

1. color[w]=color[parent[x]]
2. color[parent[x]]=schwarz
3. color[right[w]]=schwarz
4. Linksrotation(T,parent[x])
5. x=root[T]

Datenstrukturen

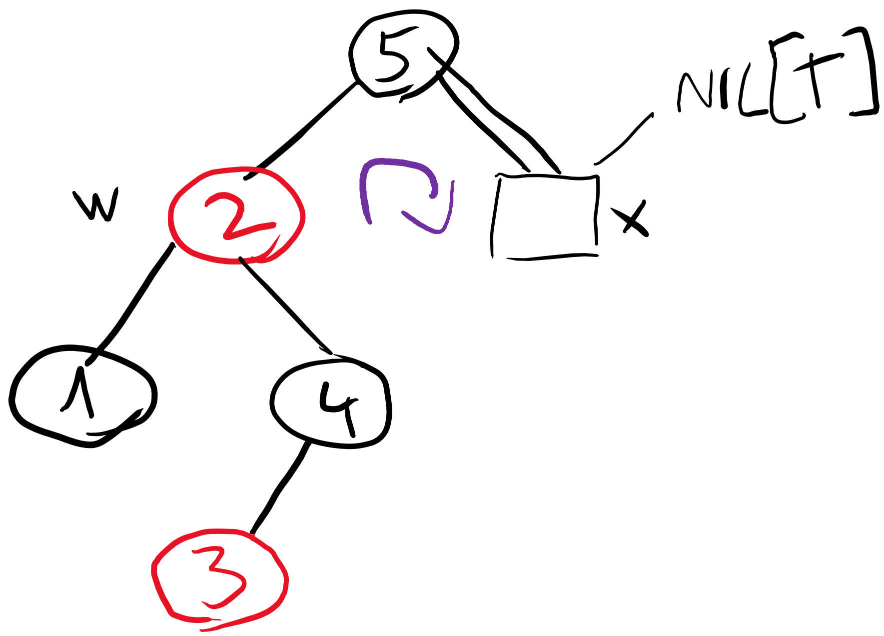
Beispiel Löschen in Rot-Schwarz-Bäumen



Datenstrukturen

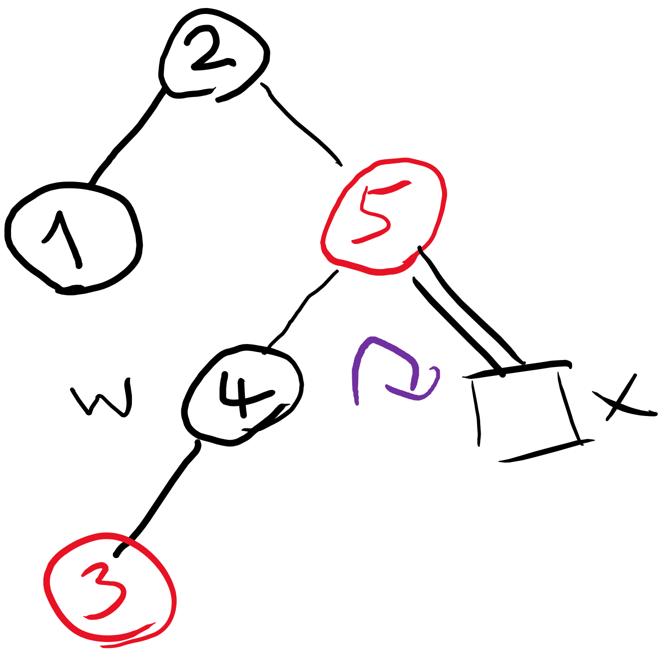
Beispiel Löschen in Rot-Schwarz-Bäumen

Fall 1



Datenstrukturen

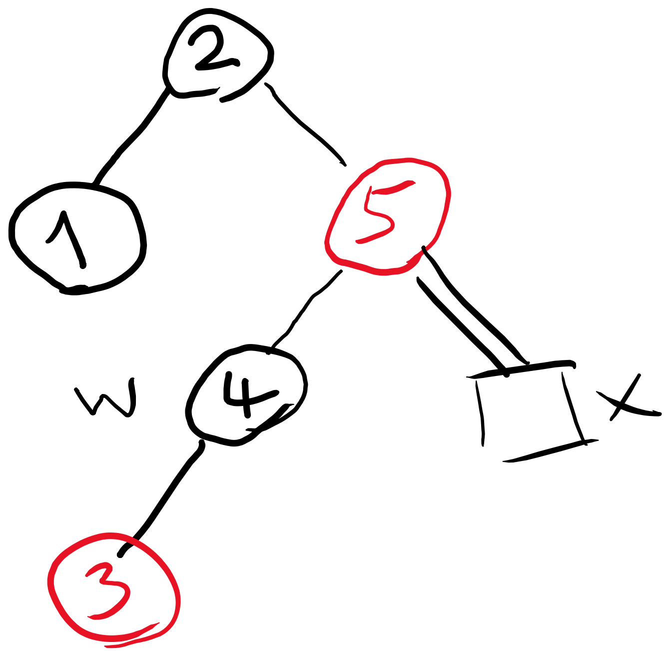
Beispiel Löschen in Rot-Schwarz-Bäumen



Fall 4

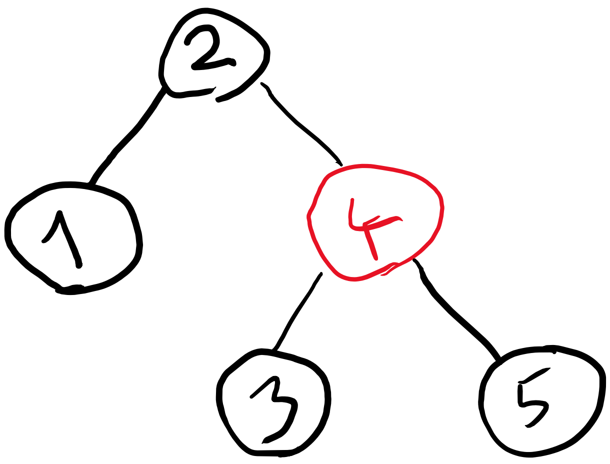
Datenstrukturen

Beispiel Löschen in Rot-Schwarz-Bäumen



Datenstrukturen

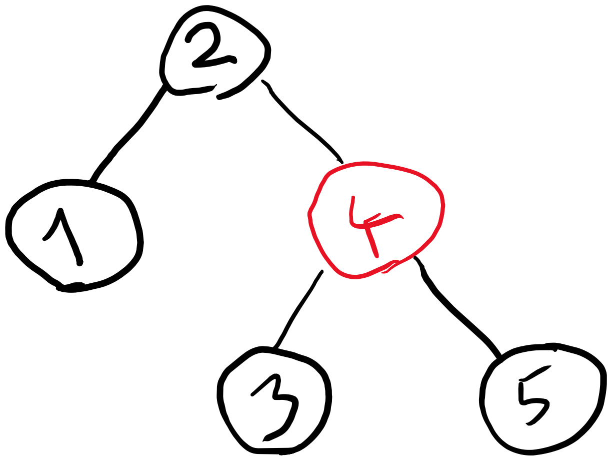
Beispiel Löschen in Rot-Schwarz-Bäumen



Datenstrukturen

Aufgabe

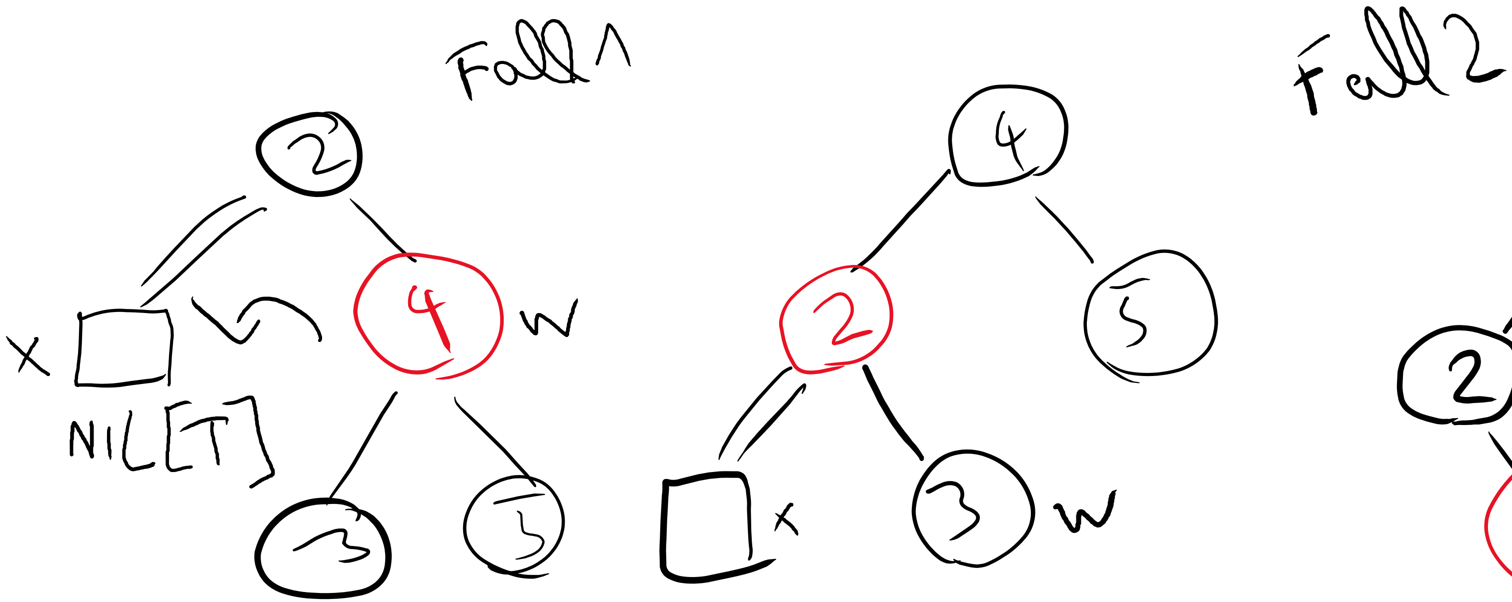
- Löschen von Schlüssel 1 in folgendem Baum



Datenstrukturen

Aufgabe

- Löschen von Schlüssel 1 in folgendem Baum



Universität zu Köln

Datenstrukturen

RS-Löschen(T,z)   * Zu löschender Knoten wird übergeben

1. **if** left[z]=NIL[T] or right[z]=NIL[T] **then** y=z
2. **else** y=NachfolgerSuche(z)
3. **if** left[y]≠NIL[T] **then** x=left[y] **else** x=right[y]
4. parent[x]=parent[y]
5. **if** parent[y]=NIL[T] **then** root[T]=x
6. **else if** y=left[parent[y]] **then** left[parent[y]]=x **else** right[parent[y]]=x
7. key[z]=key[y]
8. **if** color[y]=schwarz **then** RS-Löschen-Fix(T,x)
9. parent[NIL[T]]=NIL
10. **delete** y

$O(1)$

$O(\log n)$

$O(1)$

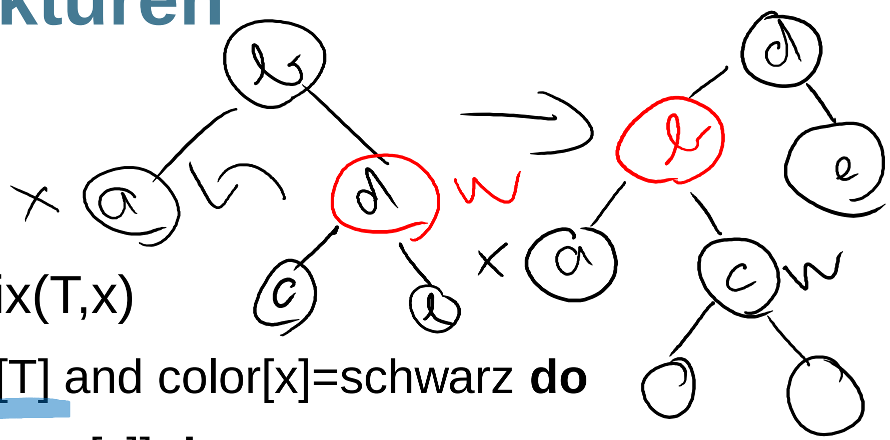
$O(\log n)$

$O(1)$

$O(\log n)$

Universität zu Köln

Datenstrukturen



RS-Löschen-Fix(T,x)

```
while x≠root[T] and color[x]=schwarz do
    if x=left[parent[x]] then
        w = right[parent[x]]
        if color[w]=rot then  ***Fall 1 ***
        if color[left[w]]=schwarz and color[right[w]]=schwarz then *** Fall 2 ***
        else
            if color[right[w]]=schwarz then *** Fall 3 ***
            *** Fall 4 ***
    else (symmetrischer Fall)
color[x] = schwarz
```

$O(\log n)$

$O(1)$

$O(\log n)$

Fall 2:

1. color[w]=rot
2. x=parent[x]

Fall 4:

1. color[w]=color[parent[x]]
2. color[parent[x]]=schwarz
3. color[right[w]]=schwarz
4. Linksrotation(T,parent[x])
5. x=root[T]

Datenstrukturen

Satz 18.1

- Die Laufzeit der Operationen Suchen, Einfügen und Löschen in Rot-Schwarz-Bäumen ist O(log n), wobei n die aktuelle Anzahl Schlüssel im Baum ist.

Beweis

- Suchen ist identisch wie in binären Suchbäumen und hat Laufzeit O(log n), da die Höhe von Rot-Schwarz-Bäumen O(log n) ist
- Die Laufzeiten von Einfügen und Löschen haben wir bereits analysiert

Datenstrukturen

Zusammenfassung

- Rot-Schwarz-Bäume sind balancierte Suchbäume
- Suchen, Einfügen, Löschen, Minimum, Maximum, Nachfolgersuche in O(log n) Zeit
- Kann man noch schneller werden?

Referenzen

- T. Cormen, C. Leisserson, R. Rivest, C. Stein. Introduction to Algorithms. The MIT press. Second edition, 2001.