Grundzüge der Informatik 1

Vorlesung 17

Überblick Vorlesung

Überblick

- Wiederholung
  - Rot-Schwarz-Bäume
- Höhe von Rot-Schwarz-Bäumen
- Einfügen in Rot-Schwarz-Bäume

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

Rot-Schwarz-Bäume

- Binäre Suchbäume
- Verbundtyp Knoten enthält color, key, parent, left, right
- NIL-Zeiger werden als Zeiger auf Blätter interpretiert
(im Gegensatz zu unserer „normalen“ Betrachtungsweise)
- Informationen werden nur in den internen Knoten gespeichert
- Erfüllen die Rot-Schwarz-Eigenschaften

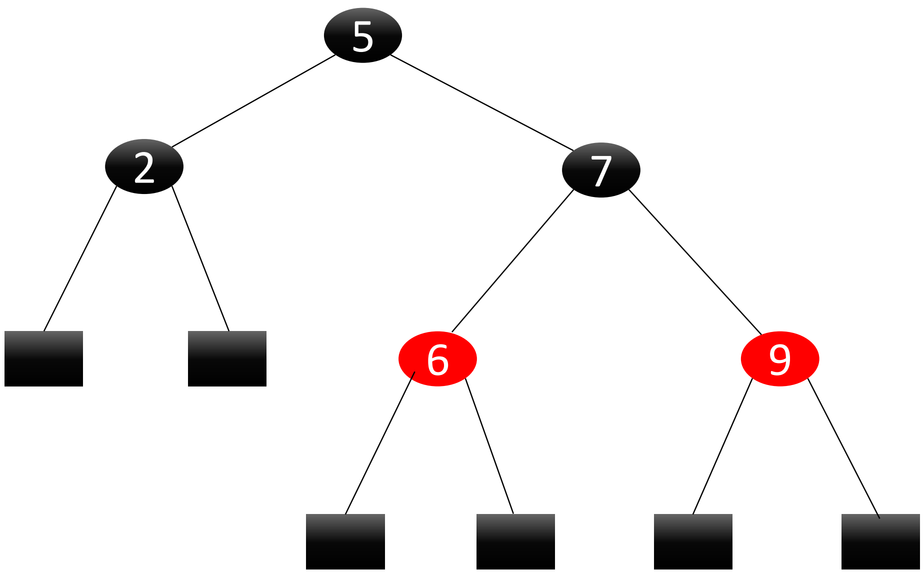
Datenstrukturen

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

Beispiel Rot-Schwarz-Baum



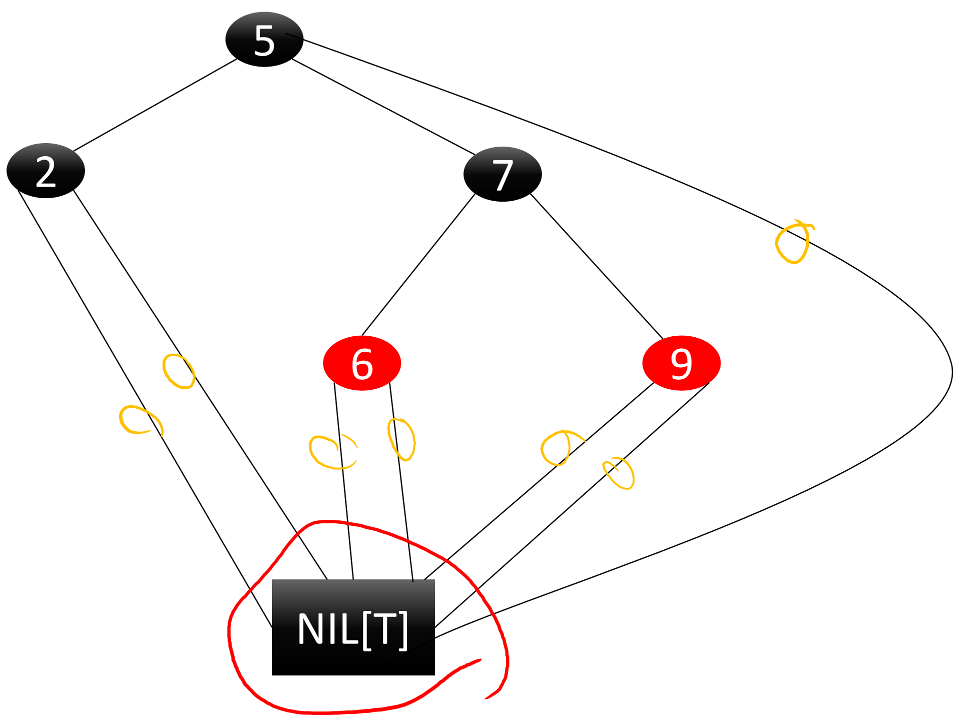
Datenstrukturen

Speichern von Rot-Schwarz-Bäumen

- Für einen Rot-Schwarz-Baum T gibt es einen speziellen Knoten NIL[T]
- NIL[T] ist ein Verbundobjekt vom Typ Knoten, das folgende Werte enthält
- color ist schwarz
- key ist 0
- left, right und parent sind NIL
- Wir werden alle NIL Zeiger in der Baumrepräsentation (also, die Blätter und den parent Zeiger der Wurzel) durch Zeiger auf NIL[T] ersetzen
- Dies vereinfacht die Implementierung

Datenstrukturen

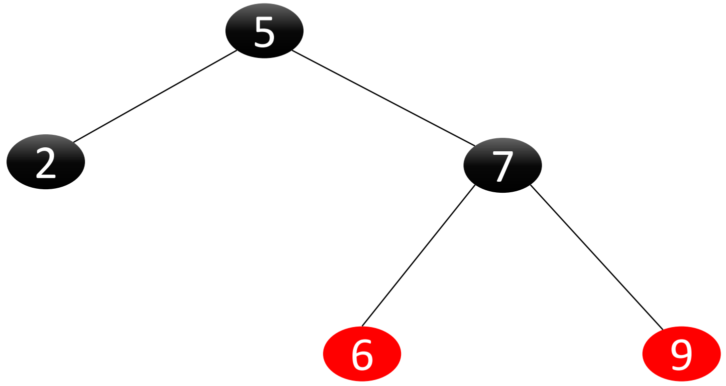
Beispiel Rot-Schwarz-Baum mit Knoten NIL[T]



Universität zu Köln

Datenstrukturen

Beispiel Rot-Schwarz-Baum (Darstellung im weiteren Verlauf)

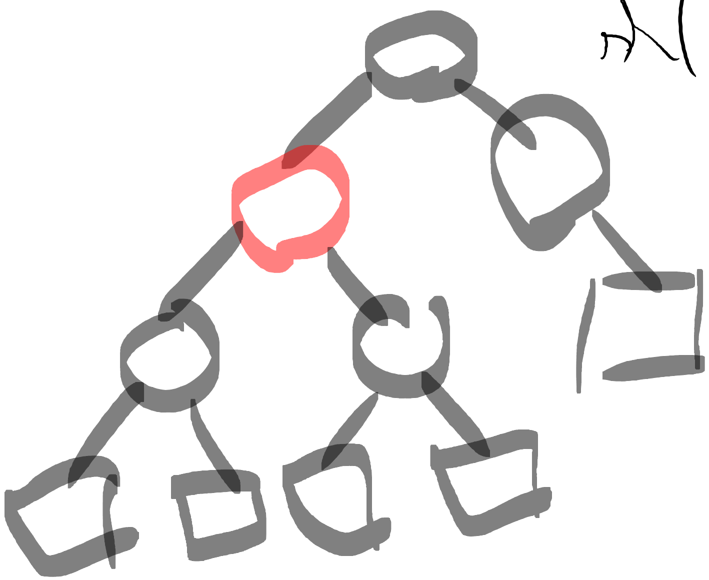


Datenstrukturen

Definition

- Die Schwarzhöhe sh(v) eines Knotens v in einem Rot-Schwarz-Baums ist die Anzahl der schwarzen Knoten ohne Knoten v auf einem Pfad von v zum NIL[T] Knoten (der NIL[T] Knoten wird mitgezählt, sofern v≠NIL[T]).

$sh(root(T)) = 2$



Datenstrukturen

Lemma 17.1

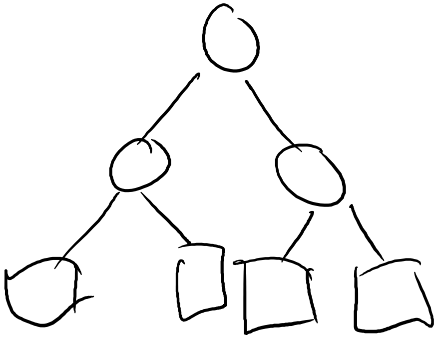
- Ein Unterbaum mit Wurzel v eines Rot-Schwarz-Baums hat mindestens $2^{sh(v)}-1$ interne Knoten.

Beweis

- Induktion über Höhe h des Unterbaums. Induktionsanfang:
- Ein Unterbaum, in dem kein Knoten abgespeichert ist besteht nur aus einem Knoten v=NIL[T] und hat sh(v)=0. Außerdem gilt $2^0-1=0$.
- Induktionsannahme: Die Aussage gelte für Unterbäume der Höhe h-1
- Induktionsschluss: Der linke und rechte Unterbaum von v hat sh(v) oder sh(v)-1
- Die Höhe der Unterbäume ist kleiner als h
- Nach Induktionsannahme hat der Unterbaum mit Wurzel v mindestens $2^{sh(v)-1}-1 + 2^{sh(v)-1}-1+1= 2^{sh(v)}-1$ interne Knoten

Universität zu Köln

Datenstrukturen



Lemma 17.2

- Ein Rot-Schwarz-Baum mit n Schlüsseln hat eine Höhe von höchstens 2 log (n+1).

Beweis

- Sei h die Höhe eines Rot-Schwarz Baums T (inkl. NIL[T] Knoten)
- Jeder Pfad von der Wurzel zu einem Blatt hat mindestens genau so viele rote wie schwarze Knoten
- Daher ist $sh(root(T)) \geq h/2$
- Nach Lemma 17.1 gilt somit $n \geq 2^{h/2} -1$
- Es folgt $n+1 \geq 2^{h/2}$ und $\log(n+1) \geq h/2$ (Monotonie des Logarithmus)
- somit folgt $2\log(n+1) \geq h$

Universität zu Köln

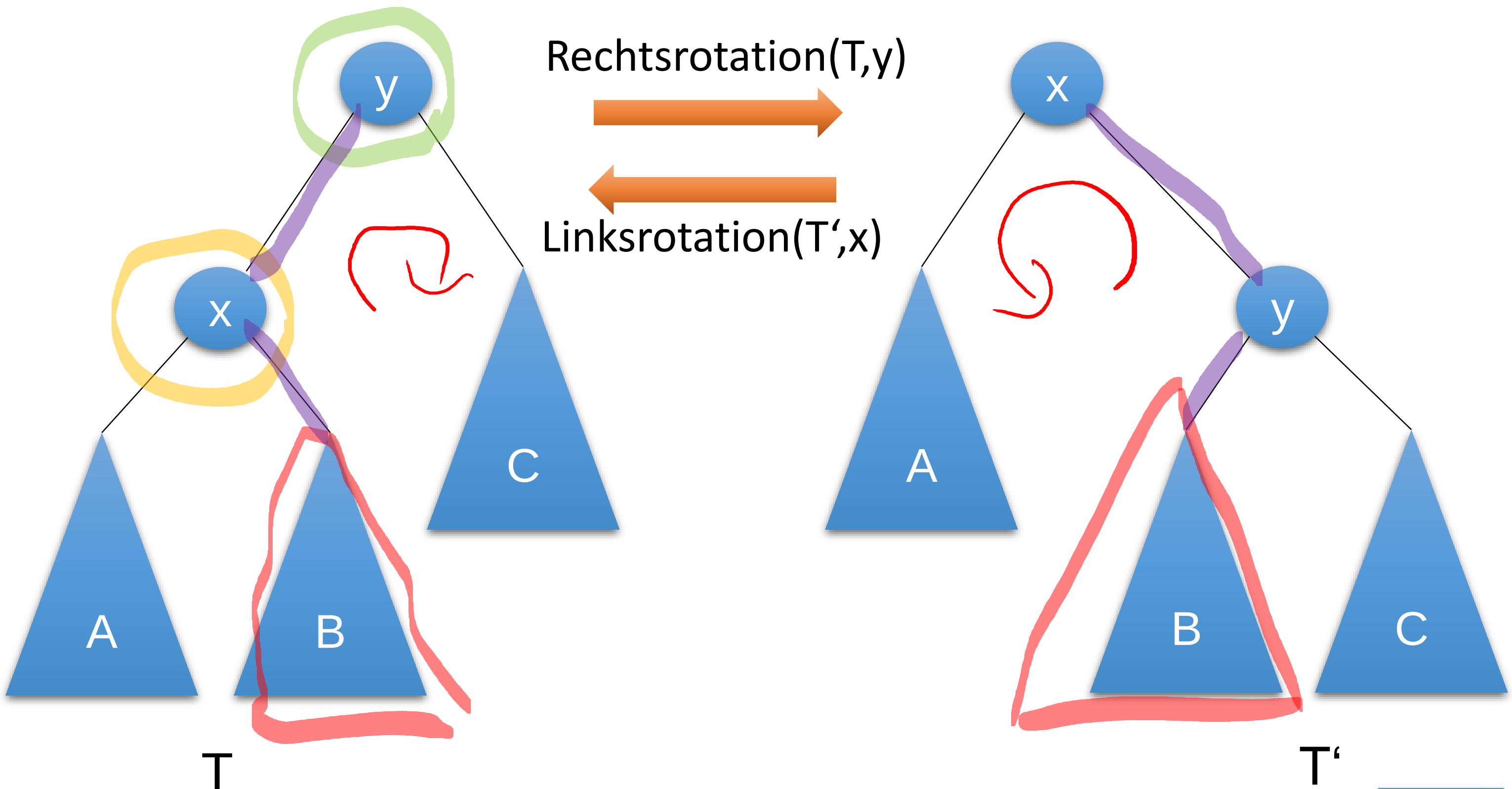
Datenstrukturen

Modifikation der Struktur von Suchbäumen

- Rotationen können die Struktur von Suchbäumen verändern und gleichzeitig die Suchbaumeigenschaft aufrecht erhalten
- Man kann Rotationen zur Balancierung von Suchbäumen benutzen

Datenstrukturen

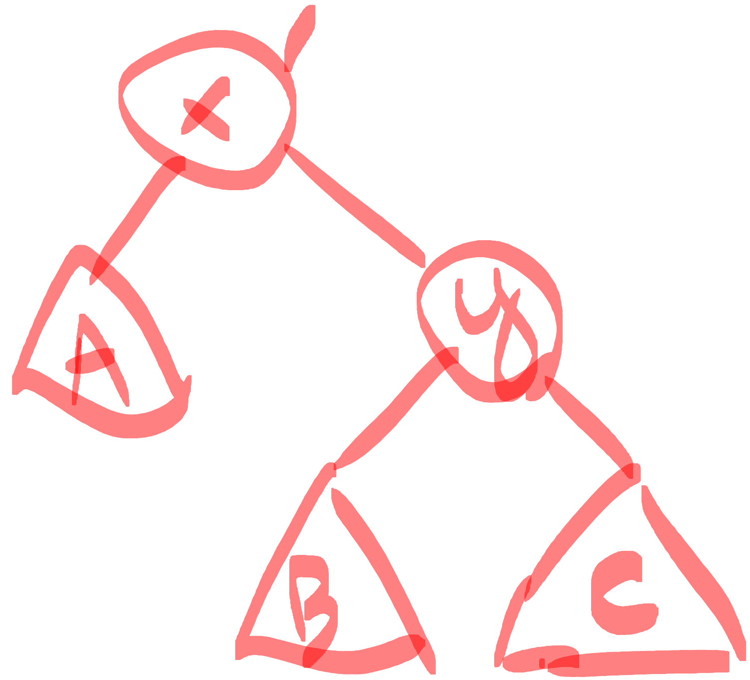
Rotationen

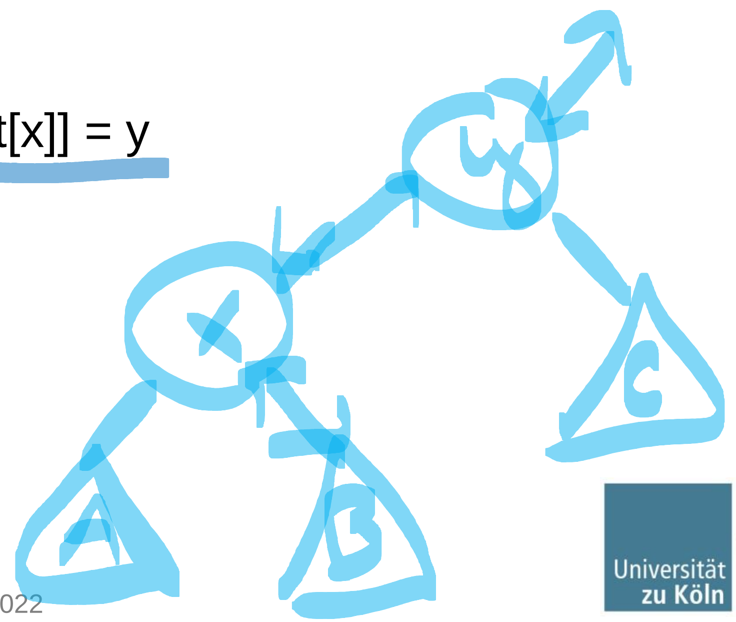


Datenstrukturen

Linksrotation(T,x)

1. y = right[x]
2. right[x] = left[y]
3. **if** left[y]≠NIL **then** parent[left[y]] = x
4. parent[y] ← parent[x]
5. **if** parent[x]=NIL **then** root[T] = y
6. **else if** x=left[parent[x]] **then** left[parent[x]] = y
7. **else** right[parent[x]] ← y
8. left[y] ← x
9. parent[x] ← y

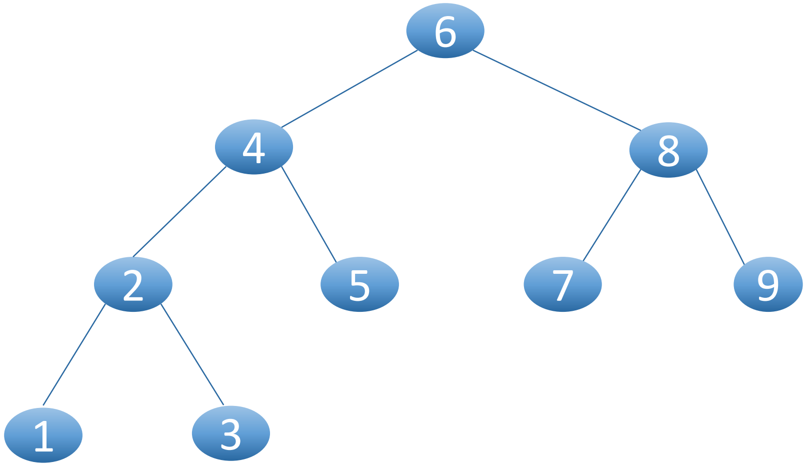




Datenstrukturen

Aufgabe

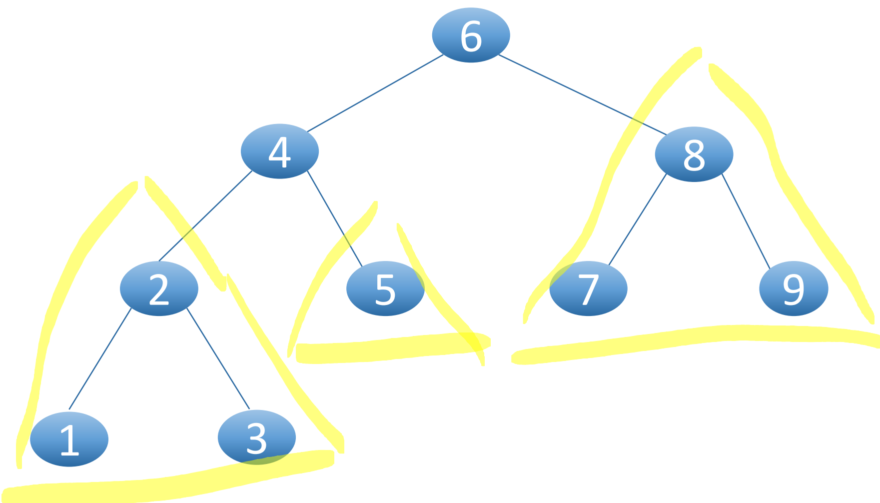
- Führen Sie eine Rechtsrotation um Knoten 6 in folgendem Baum durch
- Führen Sie eine Linksrotation um Knoten 8 im selben Baum durch

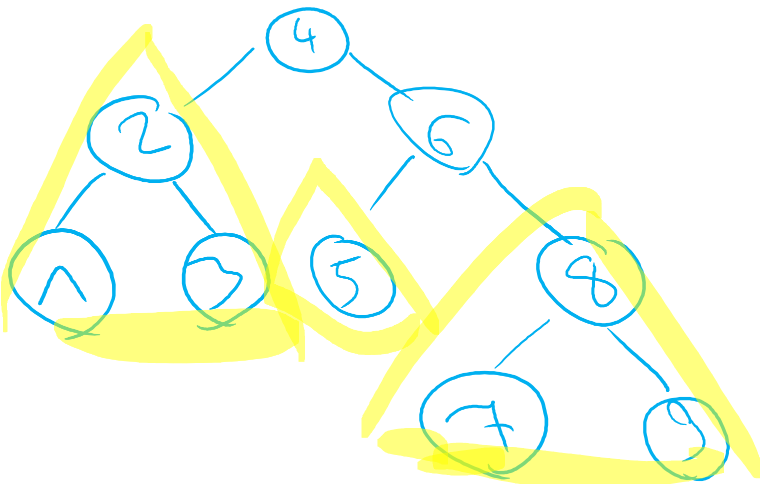


Datenstrukturen

Aufgabe

- Führen Sie eine Rechtsrotation um Knoten 6 in folgendem Baum durch
- Führen Sie eine Linksrotation um Knoten 8 im selben Baum durch

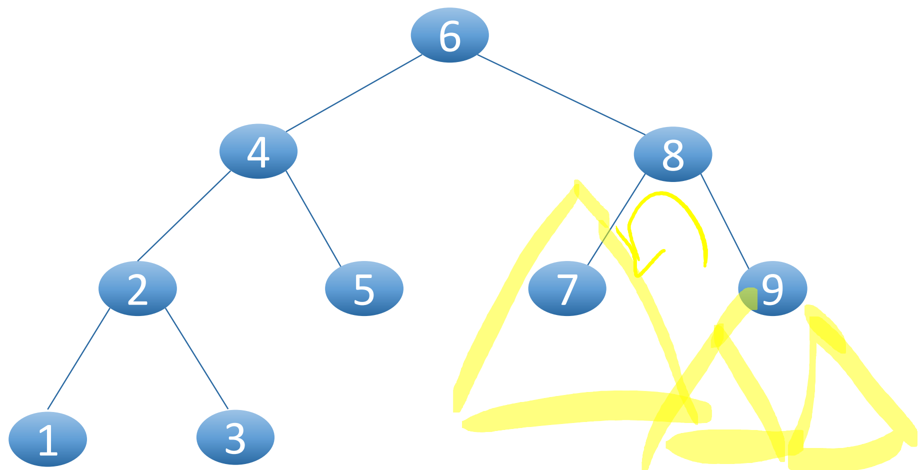


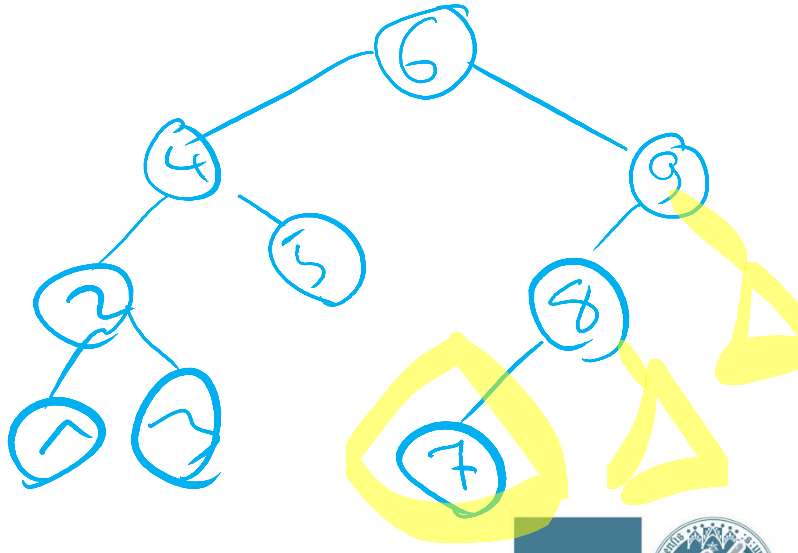


Datenstrukturen

Aufgabe

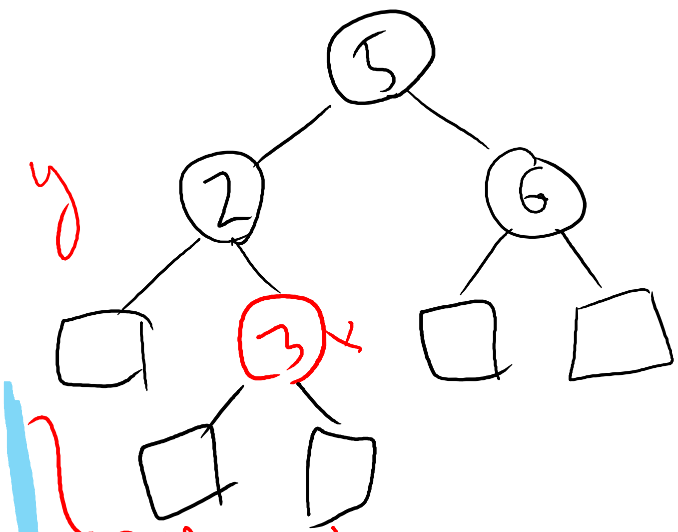
- Führen Sie eine Rechtsrotation um Knoten 6 in folgendem Baum durch
- Führen Sie eine Linksrotation um Knoten 8 im selben Baum durch





Datenstrukturen

3



RS-Einfügen(T,k)

1. z = **new** Knoten
2. y=NIL[T]; x = Root[T]
3. **while** x≠NIL[T] **do**
4.   y=x
5.   **if** k<key[x] **then** x=left[x] **else** x=right[x]
6. parent[z] = y
7. **if** y=NIL[T] **then** root[T] = z
8. **else if** k<key[y] **then** left[y] = z **else** right[y]=z
9. key[z] = k; left[z] = NIL[T]; right[z] = NIL[T]; color[z] = rot
10. RS-Einfügen-Fix(T,z)

$O(1)$

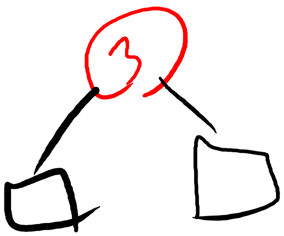
$O(\log n)$

$O(1)$

? $O(\log n)$

$O(\log n)$

Datenstrukturen



Zustand nach dem Einfügen

- Der neue Knoten z ist rot

Die Rot-Schwarz-Eigenschaft?

(1) Jeder Knoten ist rot oder schwarz

(2) Die Wurzel ist schwarz ?

(3) Jedes Blatt ist schwarz

(4) Wenn ein Knoten rot ist, dann sind seine Kinder schwarz ?

(5) Für jeden Knoten v haben alle Pfade vom Knoten v zu den Blättern im Unterbaum mit Wurzel v dieselbe Anzahl schwarzer Knoten

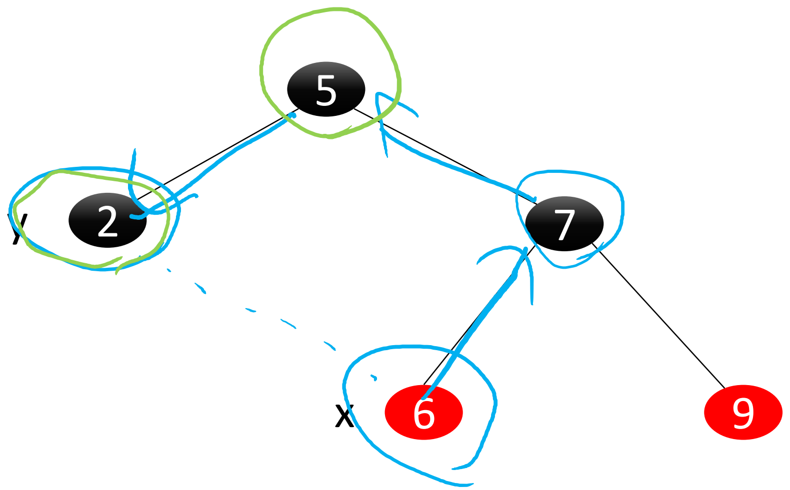
Datenstrukturen

Definition

- Die Tiefe eines Knotens v ist die Länge eines Weges von der Wurzel zu v
- Der Onkelknoten eines Knotens v mit Tiefe mindestens 2 ist das Kind von parent[parent[v]], was nicht parent[v] ist

Beispiel

- y ist Onkelknoten von x



Datenstrukturen

Überblick: Wiederherstellen der Rot-Schwarz-Eigenschaft

- 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

Invariante (Inv) beim Wiederherstellen

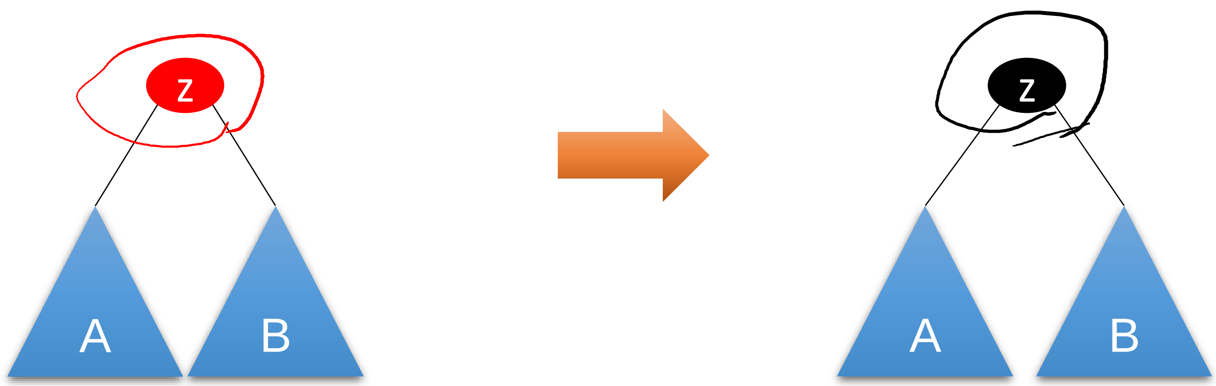
- Knoten z ist rot
- Wenn parent[z] die Wurzel ist, dann ist parent[z] schwarz
- Es gibt max. eine Verletzung der Rot-Schwarz-Eigenschaften (entweder Eigenschaft 2 oder 4)
- Ist Eigenschaft 2 verletzt, dann ist z die Wurzel
- Ist Eigenschaft 4 verletzt, dann ist parent[z] rot

Insbesondere

- Die Anzahl der schwarzen Knoten auf allen Pfaden von der Wurzel zu jedem Blatt ist gleich

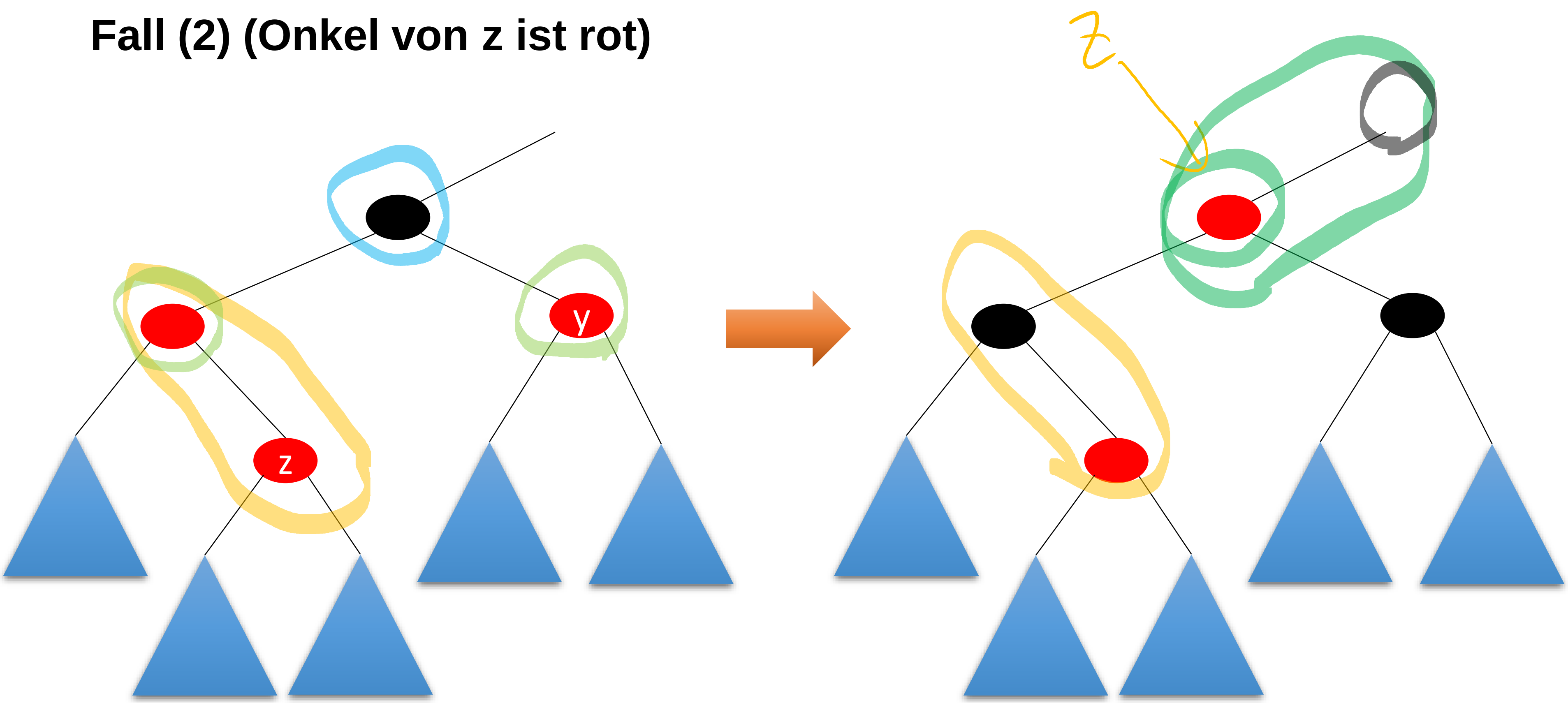
Datenstrukturen

Fall (1) (z ist die Wurzel)



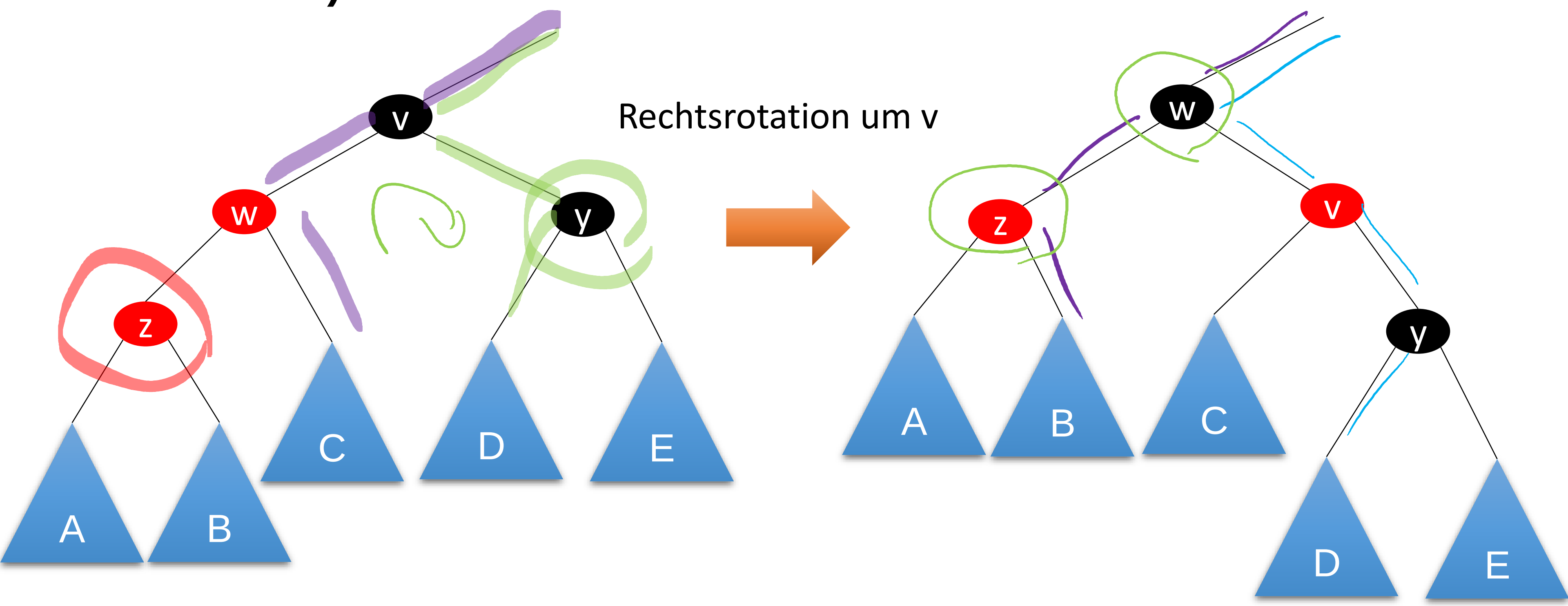
Datenstrukturen

Fall (2) (Onkel von z ist rot)



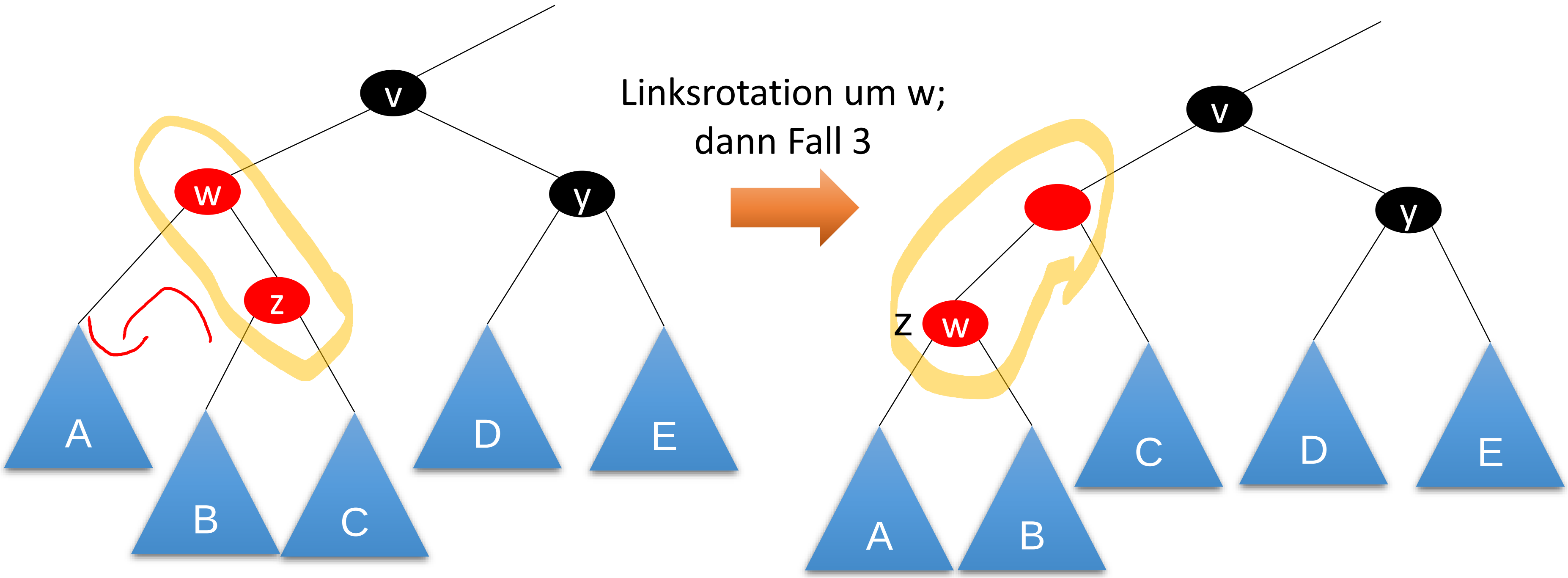
Datenstrukturen

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

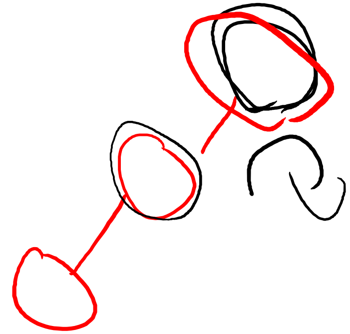


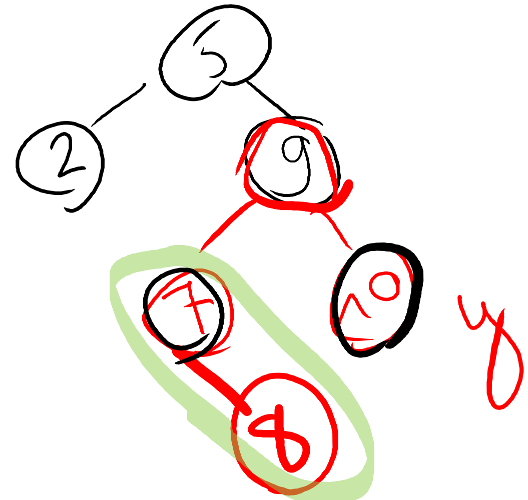
Datenstrukturen

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



Datenstrukturen





RS-Einfügen-Fix(T,z)

1. **while** color[parent[z]]= rot **do**
2. **if** parent[z] = left[parent[parent[z]]] **then**
3. y=right[parent[parent[z]]]
4. **if** color[y] = rot **then**
5. color[parent[z]]=schwarz; color[y] = schwarz * Fall 2
6. color[parent[parent[z]]] = rot; z= parent[parent[z]] * Fall 2
7. **else if** z = right[parent[z]] **then** z = parent[z]; Linksrotation(T,z) * Fall 4
8. color[parent[z]]=schwarz; color[parent[parent[z]]]=rot *Fall 3
9. Rechtsrotation(T, parent[parent[z]]) * Fall 3
10. **else** *** symmetrischer Fall (Übung)***
11. color[root[T]] = schwarz * Fall 1

O(log n)

O(1)

O(log n)

Datenstrukturen

Lemma 17.3

- Die Laufzeit von RS-Einfügen ist O(log n), wobei n die Anzahl Schlüssel im Baum ist.

Beweis

- Wir haben bereits die Laufzeit analysiert.

Datenstrukturen

Lemma 17.4

- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- Wir zeigen zunächst, dass (Inv) vor dem ersten Schleifendurchlauf gilt:
- Knoten z ist zu Beginn rot
- Wenn parent[z] die Wurzel ist, dann ist die Wurzel schwarz
- Es gibt maximal eine Verletzung der Rot-Schwarz-Eigenschaften
- Eigenschaft 2 ist nur verletzt, wenn z als erster Knoten in den Baum eingefügt wird
- Eigenschaft 4 kann nur an einer Stelle verletzt sein, da der Baum vor dem Einfügen ein Rot-Schwarz-Baum war

Datenstrukturen

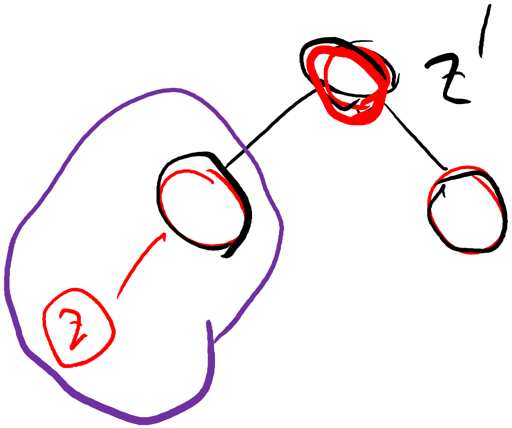
Lemma 17.4

- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- Aufrechterhaltung der Invariante:
- Wir zeigen, dass nach jedem Durchlauf der Schleife, die Invariante erfüllt ist
- Es gibt insgesamt 6 unterschiedliche Fälle
- Diese entsprechen unseren Fällen 2-4 und deren symmetrischen Fällen
- Wir gehen die Fälle 2-4 durch
  (die Argumentation für die symmetrischen Fälle ist analog)
- Wir beobachten als erstes, dass parent[parent[z]] immer existiert,
  wenn die Schleife ausgeführt wird

Datenstrukturen



Lemma 17.4

- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- Fall 2 (der Onkel von z ist rot)
- Wir benutzen z'=parent[parent[z]] für z nach dem Durchlauf der Schleife
- z' ist nach dem Durchlauf der Schleife rot
- Ist parent[z'] die Wurzel, so gilt: Die Farbe von parent[z'] wurde nicht verändert und ist damit immer noch schwarz
- Da parent[parent[z]] schwarz ist und parent[z] und der Onkel von z rot sind, können wir parent[parent[z]] rot färben und parent[z] und den Onkel von z schwarz, ohne Eigenschaft 5 zu verletzen
- Eigenschaft 4 wird durch das Umfärben für Knoten z und parent[z] wiederhergestellt

Datenstrukturen

Lemma 17.4

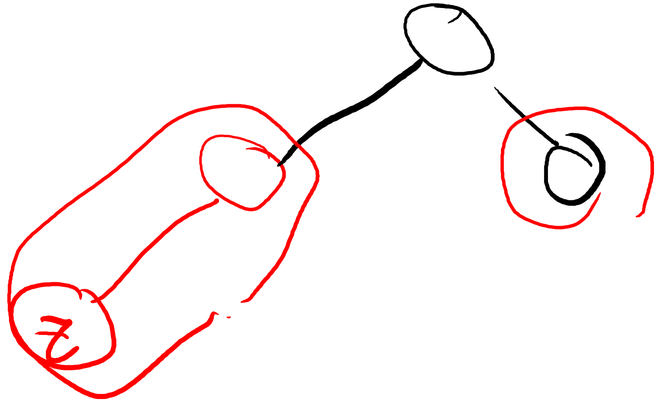
- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- z‘ kann nun entweder die Wurzel sein (dann verletzt z‘ Eigenschaft 2 aber es gibt keine Verletzung von Eigenschaft 4) oder z‘ ist nicht die Wurzel. Dann wird Eigenschaft 2 nicht verletzt und Eigenschaft 4 kann nur von z‘ und parent[z‘] verletzt werden

Universität zu Köln

Datenstrukturen



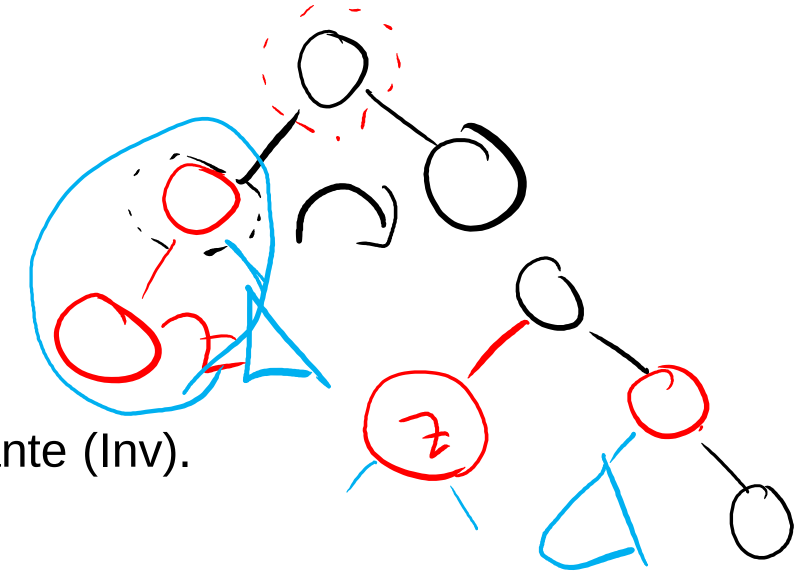
Lemma 17.4

- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- Fall 3 und 4 (Onkel von z ist schwarz)
- Fall 4 führt den Fall, dass z rechtes Kind von parent[z] ist, direkt in Fall 3 über
- Die Linksrotation beeinflusst nicht die Anzahl schwarzer Knoten auf Pfaden zu den Blättern
- Da wir vor der Linksrotation z auf parent[z] setzen ist nach der Linksrotation der Onkel von z derselbe Knoten wie vor der Rotation und damit schwarz
- Die Farbe von z ändert sich dadurch nicht
- Eigenschaft 4 wird weiterhin von z und parent[z] verletzt und ansonsten gibt es keine Verletzung der Rot-Schwarz-Eigenschaften

Datenstrukturen



Lemma 17.4

- Die while-Schleife erfüllt die Schleifeninvariante (Inv).

Beweis

- In Fall 3 wird parent[z] schwarz gefärbt, so dass der entsprechende Teil der Invariante erfüllt ist, wenn parent[z] die Wurzel ist
- Außerdem bleibt Knoten z rot
- Wir beobachten, dass nach Rotation und Umfärben die Verletzung von Eigenschaft 4 korrigiert wurde und keine neue Verletzung einer Rot-Schwarz-Eigenschaft hinzukommt
- Damit erfüllt die while-Schleife unsere Invariante

Datenstrukturen

Lemma 17.5

- Nach Durchführen des Aufrufs von RS-Einfügen-Fix(T,z) in der Prozedur RS-Einfügen erfüllt der Baum T die Rot-Schwarz-Eigenschaft.

Beweis

- Wir wissen aus Lemma 17.4, dass die Schleife unsere Invariante (Inv) aufrecht erhält
- Die Schleife terminiert, da die Laufzeit nach Lemma 17.3 beschränkt ist
- Wenn die Schleife terminiert, ist p[z] schwarz
- Dann folgt aus der Invariante, dass nur Eigenschaft (2) verletzt sein kann
- Am Ende wird aber die Wurzel schwarz gefärbt und somit gilt auch Eigenschaft (2)
- Damit folgt das Lemma

Überblick Vorlesung

Zusammenfassung

- Höhe von Rot-Schwarz-Bäumen
- Einfügen in Rot-Schwarz-Bäume

Referenzen

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