7.5 AVL-Bäume

Definition 1

AVL-Bäume sind binäre Suchbäume, die die folgende Balancierungsbedingung erfüllen: Für jeden Knoten v

$$|\mathrm{height}(\text{left sub-tree}(v)) - \mathrm{height}(\text{right sub-tree}(v))| \leq 1 \ .$$

Lemma 2

Ein AVL-Baum der Höhe h enthält mindestens $F_{h+2} - 1$ und höchstens $2^h - 1$ interne Knoten, wobei F_n die n-te Fibonaccizahl ist ($F_0 = 0$, $F_1 = 1$), und h die Höhe des Baumes bezeichnet.

In einem AVL-Baum werden Schlüssel nur an internen Knoten gespeichert. Die Blattknoten sind sogenannte dummy-leafs, die einfach ein nicht vorhandenes Kind symbolisieren.

Zusätzlich hat jeder interne Knoten genau zwei Kinder.

7.5 AVL-Bäume

Definition 1

AVL-Bäume sind binäre Suchbäume, die die folgende Balancierungsbedingung erfüllen: Für jeden Knoten v

$$|\text{height}(\textsf{left sub-tree}(v)) - \text{height}(\textsf{right sub-tree}(v))| \leq 1 \ .$$

Lemma 2

Ein AVL-Baum der Höhe h enthält mindestens $F_{h+2} - 1$ und höchstens $2^h - 1$ interne Knoten, wobei F_n die n-te Fibonaccizahl ist ($F_0 = 0$, $F_1 = 1$), und h die Höhe des Baumes bezeichnet.

In einem AVL-Baum werden Schlüssel nur an internen Knoten gespeichert. Die Blattknoten sind sogenannte dummy-leafs, die einfach ein nicht vorhandenes Kind symbolisieren.

Zusätzlich hat jeder interne Knoten genau zwei Kinder.

AVL-Bäume

Beweis.

Die obere Schranke folgt, da ein Binärbaum der Höhe h nur

$$\sum_{j=0}^{h-1} 2^j = 2^h - 1$$

interne Knoten enthalten kann.

AVL trees

Beweis (cont.)

Induktionsanfang:

1. ein AVL-Baum der Höhe $h = 1$ enthält mindestens einen internen Knoten, $1 \geq F_3 - 1 = 2 - 1 = 1$.
2. ein AVL-Baum der Höhe $h = 2$ enthält mindestens zwei interne Knoten, $2 \geq F_4 - 1 = 3 - 1 = 2$

Beweis (cont.)

Induktionsanfang:

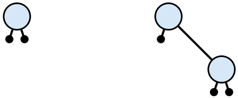
1. ein AVL-Baum der Höhe $h = 1$ enthält mindestens einen internen Knoten, $1 \geq F_3 - 1 = 2 - 1 = 1$.
2. ein AVL-Baum der Höhe $h = 2$ enthält mindestens zwei interne Knoten, $2 \geq F_4 - 1 = 3 - 1 = 2$

AVL trees

Beweis (cont.)

Induktionsanfang:

1. ein AVL-Baum der Höhe $h = 1$ enthält mindestens einen internen Knoten, $1 \geq F_3 - 1 = 2 - 1 = 1$.
2. ein AVL-Baum der Höhe $h = 2$ enthält mindestens zwei interne Knoten, $2 \geq F_4 - 1 = 3 - 1 = 2$

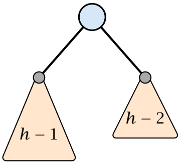


Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.

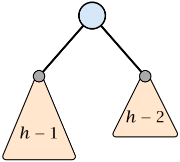
Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



Induktionsschritt ($h-1, h-2 \rightarrow n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.

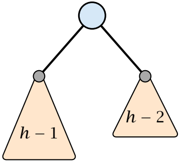


Sei

$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \; .$$

Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



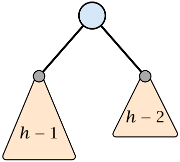
Sei

$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \; .$$

Dann

Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



Sei

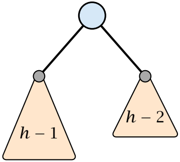
$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \ .$$

Dann

$$g_1 = 2 \qquad = F_3$$

Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



Sei

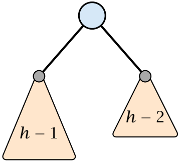
$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \;.$$

Dann

$$g_1 = 2 \qquad = F_3$$
$$g_2 = 3 \qquad = F_4$$

Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



Sei

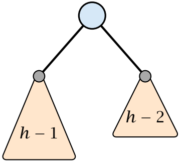
$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \;.$$

Dann

$$\begin{aligned} g_1 &= 2 && = F_3 \\ g_2 &= 3 && = F_4 \\ g_h - 1 &= 1 + g_{h-1} - 1 + g_{h-2} - 1\,, && \text{also} \end{aligned}$$

Induktionsschritt ($h-1, h-2 \to n$):

Ein minimaler AVL-Baum der Höhe $h \geq 2$ hat eine Wurzel mit zwei Teilbäumen — einer mit Höhe $h-1$ und einer mit Höhe $h-2$. Beide sind minimal.



Sei

$$g_h := 1 + \text{minimale Größe von AVL-Baum mit Höhe } h \; .$$

Dann

$$
\begin{aligned}
g_1 &= 2 && = F_3 \\
g_2 &= 3 && = F_4 \\
g_h - 1 &= 1 + g_{h-1} - 1 + g_{h-2} - 1 \,, && \text{also} \\
g_h &= g_{h-1} + g_{h-2} && = F_{h+2}
\end{aligned}
$$

7.5 AVL-Bäume

Ein AVL-Baum der Höhe h enthält mindestens $F_{h+2} - 1$ interne Knoten.

Da

$$n + 1 \geq F_{h+2} = \Omega\left(\left(\frac{1+\sqrt{5}}{2}\right)^h\right) ,$$

erhalten wir

$$n \geq \Omega\left(\left(\frac{1+\sqrt{5}}{2}\right)^h\right) ,$$

und daher $h = \mathcal{O}(\log n)$.

7.5 AVL-Bäume

Wir müssen die Balancierungsbedingung aufrechterhalten.

Dafür speichern wir an jedem internen Knoten v die Balance des Knotens. Sei v ein Baumknoten mit linkem Kind c_ℓ und rechtem Kind c_r.

$$\text{balance}[v] := \text{height}(T_{c_\ell}) - \text{height}(T_{c_r}) \quad ,$$

wobei T_{c_ℓ} und T_{c_r}, die Teilbäume mit Wurzeln c_ℓ und c_r sind.

7.5 AVL-Bäume

Wir müssen die Balancierungsbedingung aufrechterhalten.

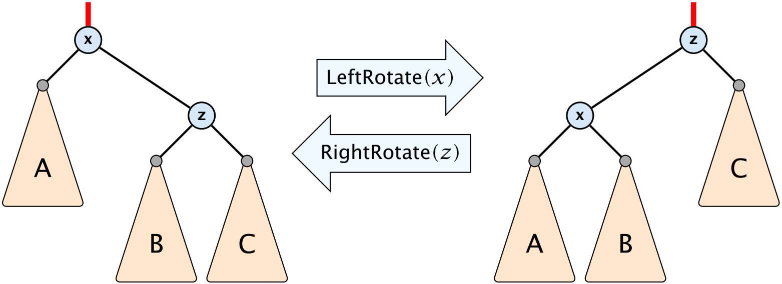
Dafür speichern wir an jedem internen Knoten v die Balance des Knotens. Sei v ein Baumknoten mit linkem Kind c_ℓ und rechtem Kind c_r.

$$\text{balance}[v] := \text{height}(T_{c_\ell}) - \text{height}(T_{c_r}) \ ,$$

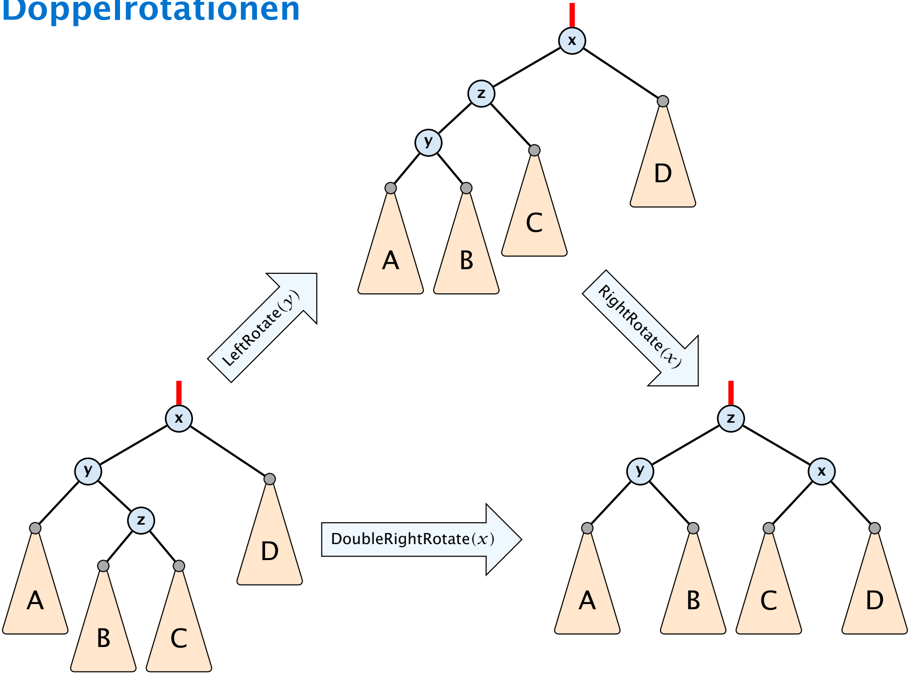
wobei T_{c_ℓ} und T_{c_r}, die Teilbäume mit Wurzeln c_ℓ und c_r sind.

Rotationen

Die Balancierungsbedingung wird durch Rotationen aufrechterhalten:



Doppelrotationen



Rotationen sind lokale Operationen.

Wenn man einen Zeiger auf einen Baumknoten gegeben hat kann man eine Rotation um diesen Knoten in konstanter Zeit durchführen (das setzt natürlich voraus, dass man `parent`-Zeiger an jedem Baumknoten hat).

AVL-Bäume: Einfügen

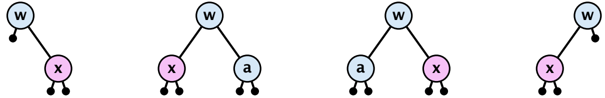
- Füge wie in einem binären Suchbaum ein.

AVL-Bäume: Einfügen

- Füge wie in einem binären Suchbaum ein.
- Sei w der Elternknoten des neuen Knotens x.

AVL-Bäume: Einfügen

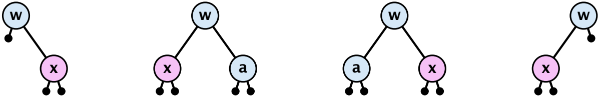
- Füge wie in einem binären Suchbaum ein.
- Sei w der Elternknoten des neuen Knotens x.
- Es gilt einer der folgenden Fälle:



$\mathrm{bal}(w) = -1$ $\quad$ $\mathrm{bal}(w) = 0$ $\quad$ $\mathrm{bal}(w) = 0$ $\quad$ $\mathrm{bal}(w) = 1$

AVL-Bäume: Einfügen

- Füge wie in einem binären Suchbaum ein.
- Sei w der Elternknoten des neuen Knotens x.
- Es gilt einer der folgenden Fälle:

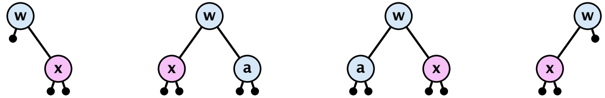


$\text{bal}(w) = -1$ $\quad$ $\text{bal}(w) = 0$ $\quad$ $\text{bal}(w) = 0$ $\quad$ $\text{bal}(w) = 1$

- Falls $\text{bal}[w] \neq 0$, hat T_w seine Höhe geändert; die Balancierungsbedingung könnte an Vorgängern von w verletzt sein.

AVL-Bäume: Einfügen

- Füge wie in einem binären Suchbaum ein.
- Sei w der Elternknoten des neuen Knotens x.
- Es gilt einer der folgenden Fälle:



$\text{bal}(w) = -1$ $\text{bal}(w) = 0$ $\text{bal}(w) = 0$ $\text{bal}(w) = 1$

- Falls $\text{bal}[w] \neq 0$, hat T_w seine Höhe geändert; die Balancierungsbedingung könnte an Vorgängern von w verletzt sein.
- Wir rufen `AVL_fix_up_insert(w->parent)` um diese wiederherzustellen.

AVL-Bäume: Einfügen

Invariante zu Beginn von AVL-fix-up-insert(v):

1. Die Balancierungsbedingung gilt für alle Nachfolger von v.
2. Ein Knoten wurde in T_c eingefügt, wobei c ein Kind von v ist.
3. T_c hat seine Höhe um 1 erhöht (sonst hätten wir die fix-up Prozedur schon beendet).
4. Die Balance am Knoten c erfüllt $\text{balance}[c] \in \{-1, 1\}$. Dies gilt, da ansonsten der Teilbaum T_c seine Höhe nicht geändert hätte.

AVL-Bäume: Einfügen

Invariante zu Beginn von AVL-fix-up-insert(v):

1. Die Balancierungsbedingung gilt für alle Nachfolger von v.
2. Ein Knoten wurde in T_c eingefügt, wobei c ein Kind von v ist.
3. T_c hat seine Höhe um 1 erhöht (sonst hätten wir die fix-up Prozedur schon beendet).
4. Die Balance am Knoten c erfüllt $\text{balance}[c] \in \{-1, 1\}$. Dies gilt, da ansonsten der Teilbaum T_c seine Höhe nicht geändert hätte.

AVL-Bäume: Einfügen

Invariante zu Beginn von AVL-fix-up-insert(v):

1. Die Balancierungsbedingung gilt für alle Nachfolger von v.
2. Ein Knoten wurde in T_c eingefügt, wobei c ein Kind von v ist.
3. T_c hat seine Höhe um 1 erhöht (sonst hätten wir die fix-up Prozedur schon beendet).
4. Die Balance am Knoten c erfüllt $\text{balance}[c] \in \{-1, 1\}$. Dies gilt, da ansonsten der Teilbaum T_c seine Höhe nicht geändert hätte.

AVL-Bäume: Einfügen

Invariante zu Beginn von AVL-fix-up-insert(v):

1. Die Balancierungsbedingung gilt für alle Nachfolger von v.
2. Ein Knoten wurde in T_c eingefügt, wobei c ein Kind von v ist.
3. T_c hat seine Höhe um 1 erhöht (sonst hätten wir die fix-up Prozedur schon beendet).
4. Die Balance am Knoten c erfüllt $\text{balance}[c] \in \{-1, 1\}$. Dies gilt, da ansonsten der Teilbaum T_c seine Höhe nicht geändert hätte.

AVL-Bäume: Einfügen

Invariante zu Beginn von AVL-fix-up-insert(v):

1. Die Balancierungsbedingung gilt für alle Nachfolger von v.
2. Ein Knoten wurde in T_c eingefügt, wobei c ein Kind von v ist.
3. T_c hat seine Höhe um 1 erhöht (sonst hätten wir die fix-up Prozedur schon beendet).
4. Die Balance am Knoten c erfüllt $\text{balance}[c] \in \{-1, 1\}$. Dies gilt, da ansonsten der Teilbaum T_c seine Höhe nicht geändert hätte.

AVL-Bäume: Einfügen

```
AVL_fix_up_insert(v)
    if (v->balance ∈ {-2,2})
        v = DoRotationInsert(v);
    if (v->balance == 0)
        return;
    AVL_fix_up_insert(v->parent);
```

Wir zeigen, dass dieses Verfahren korrekt ist, und dass es höchstens eine Rotation ausführt.

AVL-Bäume: Einfügen

```
DoRotationInsert(v)
    if (v->balance == -2) // insert in right sub-tree
        if (v->right->balance ∈ {0,-1})
            v = LeftRotate(v);
        else
            v = DoubleLeftRotate(v);
    else // insert in left sub-tree
        if (v->left->balance ∈ {0,1})
            v = RightRotate(v);
        else
            v = DoubleRightRotate(v);
    return v;
```

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Die Invariante für die fix-up Routine gilt solange wie keine Rotationen durchgeführt werden.

Wir zeigen, dass nach einer Rotation **all** Balancebedingungen erfüllt sind.

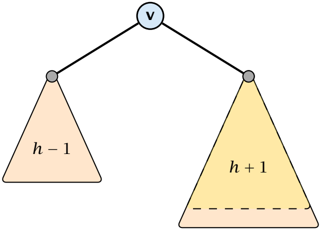
Wir zeigen dass nach einer Rotation an v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v immer noch ihre Bedingung erfüllen.
- Die Höhe des Teilbaums T_v die gleiche ist wie vor der Einfügeoperation.

Wir betrachten nur den Fall, dass in den rechten Teilbaum von v eingefügt wurde. Der andere Fall ist symmetrisch.

AVL-Bäume: Einfügen

Wir haben die folgende Situation:

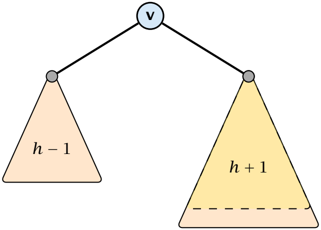


Der rechte Teilbaum von v hat seine Höhe erhöht. Dadurch ensteht die Balance von -2 am Knoten v.

Vor der Einfügeoperation war die Höhe von T_v gleich $h+1$.

AVL-Bäume: Einfügen

Wir haben die folgende Situation:



Der rechte Teilbaum von v hat seine Höhe erhöht. Dadurch ensteht die Balance von -2 am Knoten v.

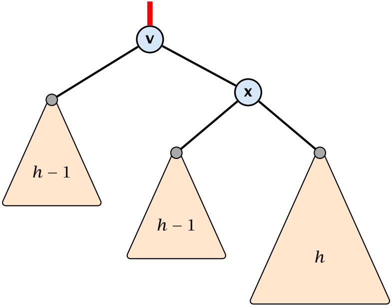
Vor der Einfügeoperation war die Höhe von T_v gleich $h+1$.

Fall 1: balance[right[v]] $= -1$

Linksrotation um v

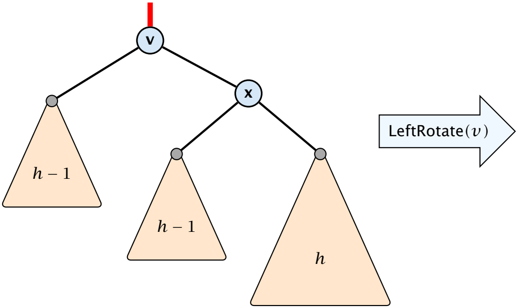
Fall 1: balance[right[v]] $= -1$

Linksrotation um v



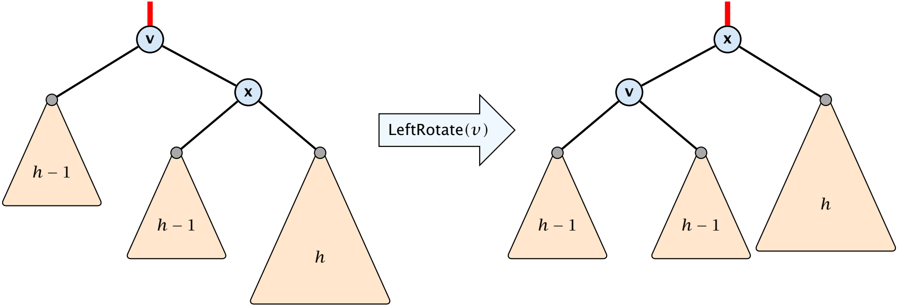
Fall 1: balance[right[v]] $= -1$

Linksrotation um v



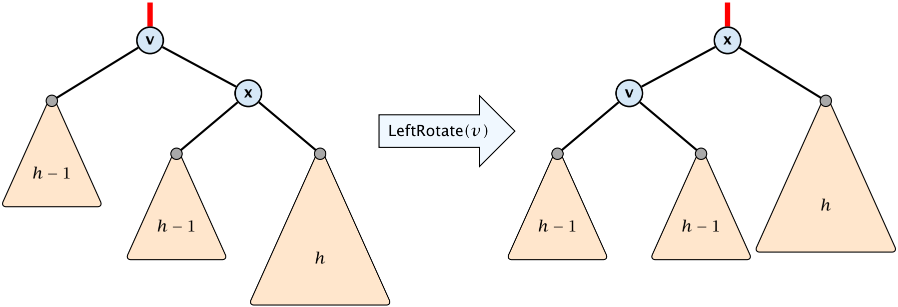
Fall 1: balance[right[v]] $= -1$

Linksrotation um v



Fall 1: balance[right[v]] $= -1$

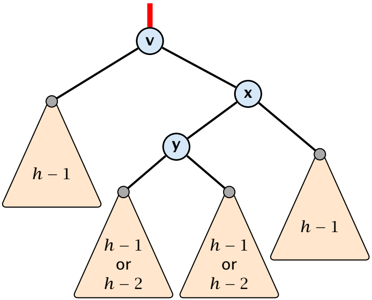
Linksrotation um v



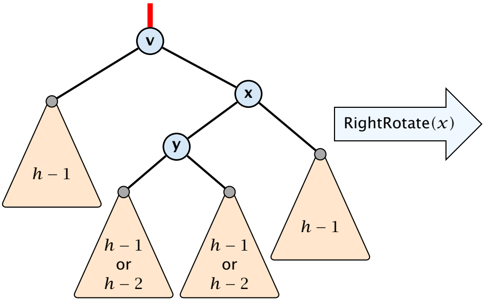
Der Teilbaum T_v hat jetzt Höhe $h+1$ wie vor dem Einfügen.

Fall 2: balance[right[v]] $= 1$

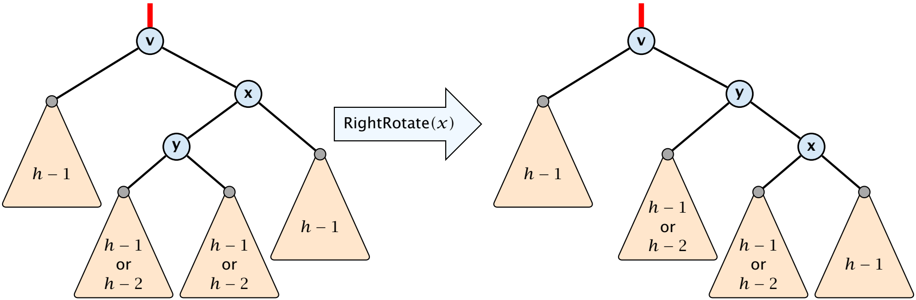
Fall 2: balance[right[v]] = 1



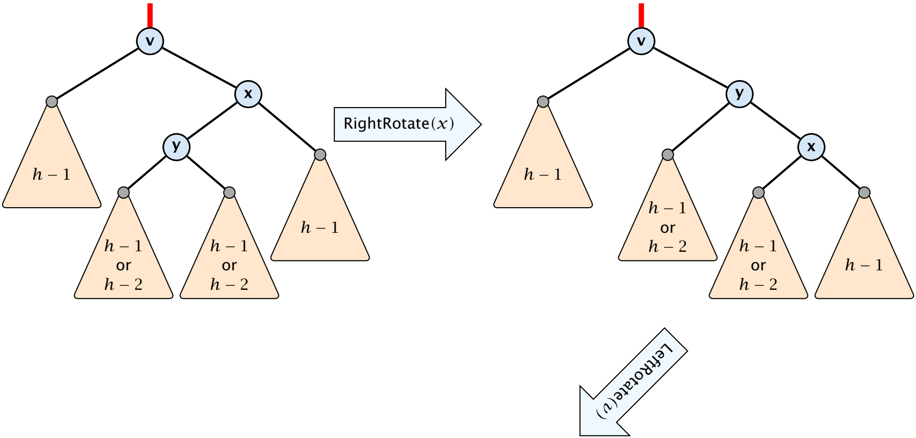
Fall 2: balance[right[v]] = 1



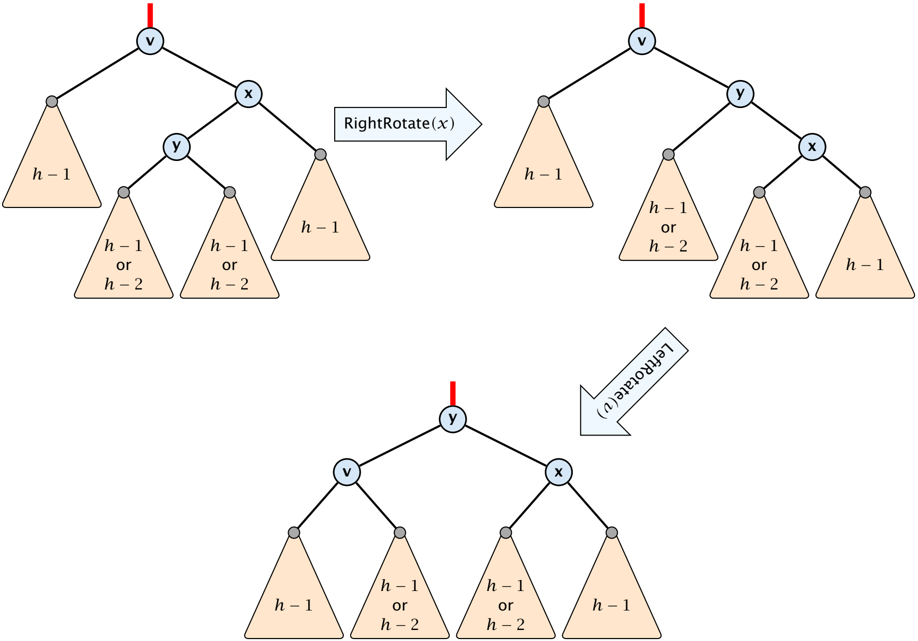
Fall 2: balance[right[v]] = 1



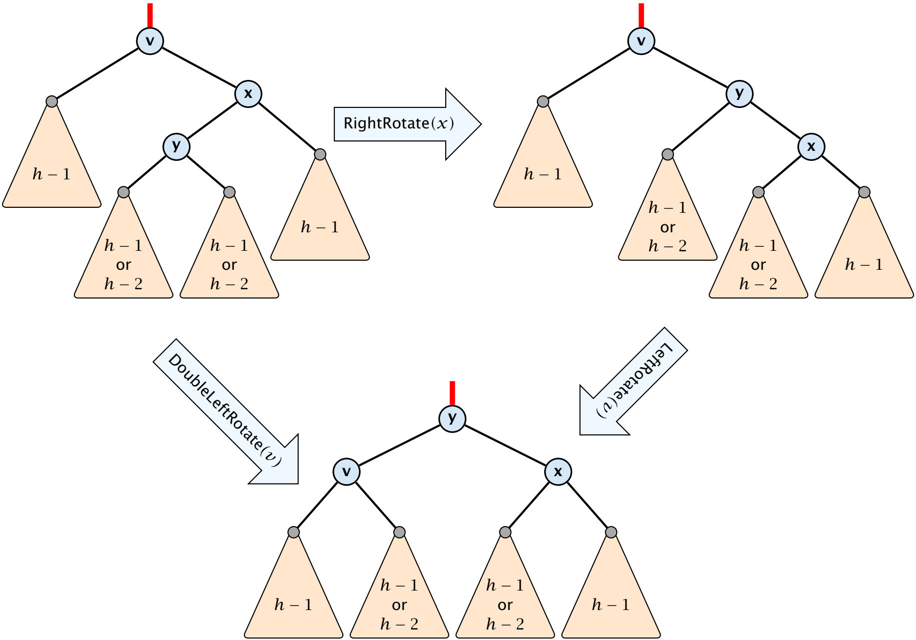
Fall 2: balance[right[v]] = 1



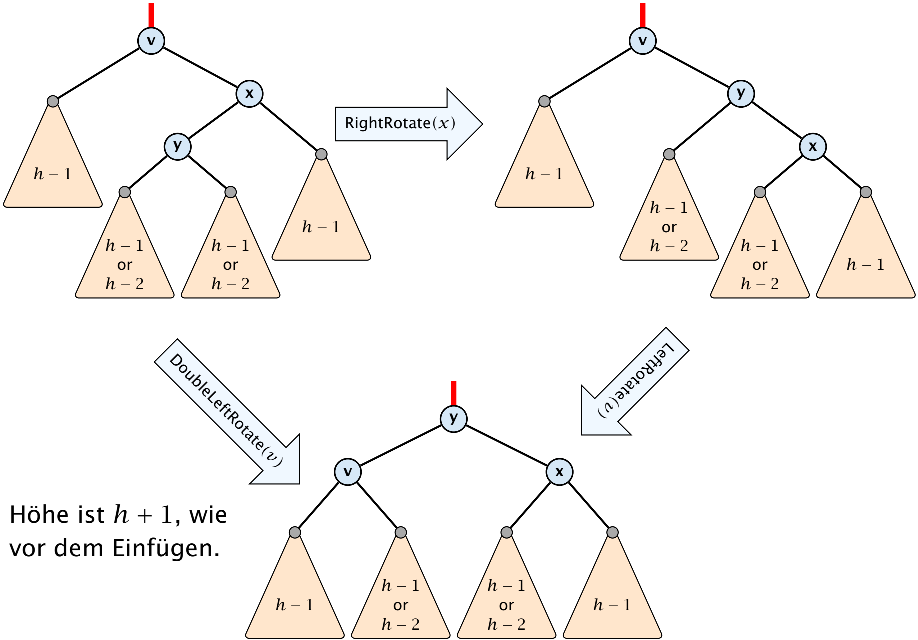
Fall 2: balance[right[v]] = 1



Fall 2: balance[right[v]] = 1



Fall 2: balance[right[v]] = 1



Höhe ist $h+1$, wie vor dem Einfügen.

AVL-Bäume: Löschen

- Löschen wie im normalen Suchbaum.
- Sei v der Elternknoten des überbrückten Knotens.
- Die Balancierungsbedingung kann an v, oder an Vorgängern von v verletzt sein, da einer der Teilbäume der Kinder von v seine Höhe verändert hat.
- Initial, ist der Knoten c—die neue Wurzel des Teilbaums der sich geändert hat—entweder ein Dummyblatt oder ein Knoten mit zwei Dummyblättern als Kindern.

Case 1 Case 2

In beiden Fällen gilt $\mathrm{bal}[c] = 0$.

- Wir nutzen AVL-fix-up-delete(v) um die Balancebedingungen wiederherzustellen.

AVL-Bäume: Löschen

- Löschen wie im normalen Suchbaum.
- Sei v der Elternknoten des überbrückten Knotens.
- Die Balancierungsbedingung kann an v, oder an Vorgängern von v verletzt sein, da einer der Teilbäume der Kinder von v seine Höhe verändert hat.
- Initial, ist der Knoten c—die neue Wurzel des Teilbaums der sich geändert hat—entweder ein Dummyblatt oder ein Knoten mit zwei Dummyblättern als Kindern.

  In beiden Fällen gilt $\mathrm{bal}[c] = 0$.

- Wir nutzen AVL-fix-up-delete(v) um die Balancebedingungen wiederherzustellen.

AVL-Bäume: Löschen

- Löschen wie im normalen Suchbaum.
- Sei v der Elternknoten des überbrückten Knotens.
- Die Balancierungsbedingung kann an v, oder an Vorgängern von v verletzt sein, da einer der Teilbäume der Kinder von v seine Höhe verändert hat.
- Initial, ist der Knoten c—die neue Wurzel des Teilbaums der sich geändert hat—entweder ein Dummyblatt oder ein Knoten mit zwei Dummyblättern als Kindern.

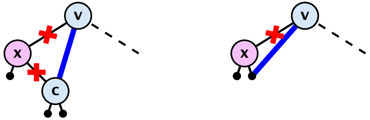
Case 1 Case 2

In beiden Fällen gilt $\text{bal}[c] = 0$.

- Wir nutzen AVL-fix-up-delete(v) um die Balancebedingungen wiederherzustellen.

AVL-Bäume: Löschen

- Löschen wie im normalen Suchbaum.
- Sei v der Elternknoten des überbrückten Knotens.
- Die Balancierungsbedingung kann an v, oder an Vorgängern von v verletzt sein, da einer der Teilbäume der Kinder von v seine Höhe verändert hat.
- Initial, ist der Knoten c—die neue Wurzel des Teilbaums der sich geändert hat—entweder ein Dummyblatt oder ein Knoten mit zwei Dummyblättern als Kindern.



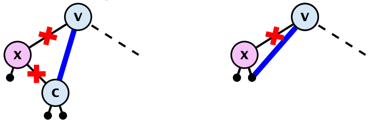
Case 1 Case 2

In beiden Fällen gilt $\mathrm{bal}[c] = 0$.

- Wir nutzen AVL-fix-up-delete(v) um die Balancebedingungen wiederherzustellen.

AVL-Bäume: Löschen

- Löschen wie im normalen Suchbaum.
- Sei v der Elternknoten des überbrückten Knotens.
- Die Balancierungsbedingung kann an v, oder an Vorgängern von v verletzt sein, da einer der Teilbäume der Kinder von v seine Höhe verändert hat.
- Initial, ist der Knoten c—die neue Wurzel des Teilbaums der sich geändert hat—entweder ein Dummyblatt oder ein Knoten mit zwei Dummyblättern als Kindern.



Case 1 Case 2

In beiden Fällen gilt $\text{bal}[c] = 0$.

- Wir nutzen AVL-fix-up-delete(v) um die Balancebedingungen wiederherzustellen.

AVL-Bäume: Löschen

Invariante zu Beginn von AVL-fix-up-delete(v):

1. Die Balancebedingungen gelten für alle Nachfolger von v.
2. Ein Knoten ist im Teilbaum T_c entfernt worden, wobei c entweder linkes oder rechtes Kind von v ist
3. T_c hat seine Höhe um eins reduziert.
4. Die Balance am Knoten c erfüllt $\mathrm{balance}[c] = 0$. Dies gilt, da wir zeigen werden, dass im Fall $\mathrm{balance}[c] \in \{-1, 1\}$ der Baum T_c seine Höhe nicht geändert hat und das deshalb die Prozedur schon abgebrochen worden wäre.

AVL-Bäume: Löschen

Invariante zu Beginn von AVL-fix-up-delete(v):

1. Die Balancebedingungen gelten für alle Nachfolger von v.
2. Ein Knoten ist im Teilbaum T_c entfernt worden, wobei c entweder linkes oder rechtes Kind von v ist
3. T_c hat seine Höhe um eins reduziert.
4. Die Balance am Knoten c erfüllt $\mathrm{balance}[c] = 0$. Dies gilt, da wir zeigen werden, dass im Fall $\mathrm{balance}[c] \in \{-1, 1\}$ der Baum T_c seine Höhe nicht geändert hat und das deshalb die Prozedur schon abgebrochen worden wäre.

AVL-Bäume: Löschen

Invariante zu Beginn von AVL-fix-up-delete(v):

1. Die Balancebedingungen gelten für alle Nachfolger von v.
2. Ein Knoten ist im Teilbaum T_c entfernt worden, wobei c entweder linkes oder rechtes Kind von v ist
3. T_c hat seine Höhe um eins reduziert.
4. Die Balance am Knoten c erfüllt $\mathrm{balance}[c] = 0$. Dies gilt, da wir zeigen werden, dass im Fall $\mathrm{balance}[c] \in \{-1, 1\}$ der Baum T_c seine Höhe nicht geändert hat und das deshalb die Prozedur schon abgebrochen worden wäre.

AVL-Bäume: Löschen

Invariante zu Beginn von AVL-fix-up-delete(v):

1. Die Balancebedingungen gelten für alle Nachfolger von v.
2. Ein Knoten ist im Teilbaum T_c entfernt worden, wobei c entweder linkes oder rechtes Kind von v ist
3. T_c hat seine Höhe um eins reduziert.
4. Die Balance am Knoten c erfüllt $\mathrm{balance}[c] = 0$. Dies gilt, da wir zeigen werden, dass im Fall $\mathrm{balance}[c] \in \{-1, 1\}$ der Baum T_c seine Höhe nicht geändert hat und das deshalb die Prozedur schon abgebrochen worden wäre.

AVL-Bäume: Löschen

```
AVL_fix_up_delete(v)
    if (v->balance ∈ {-2,2})
        v = DoRotationDelete(v);
    if (v->balance ∈ {-1,1})
        return;
    AVL_fix_up_delete(v->parent);
```

Wir zeigen, dass dies korrekt ist. Eventuell benötigen wir aber eine logarithmische Anzahl an Rotationen.

AVL-Bäume: Löschen

```
AVL_fix_up_delete(v)
    if (v->balance ∈ {-2,2})
        v = DoRotationDelete(v);
    if (v->balance ∈ {-1,1})
        return;
    AVL_fix_up_delete(v->parent);
```

Wir zeigen, dass dies korrekt ist. Eventuell benötigen wir aber eine logarithmische Anzahl an Rotationen.

AVL-Bäume: Löschen

```
DoRotationDelete(v)
    if (v->balance == -2) // deletion in left sub-tree
        if (v->right->balance ∈ {0,-1})
            v = LeftRotate(v);
        else
            v = DoubleLeftRotate(v);
    else // deletion in right sub-tree
        if (v->left->balance ∈ {0,1})
            v = RightRotate(v);
        else
            v = DoubleRightRotate(v);
    return v;
```

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\text{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\mathrm{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\mathrm{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\mathrm{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\mathrm{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Die Invariante der fix-up Routine gilt solange keine Rotationen erfolgt sind.

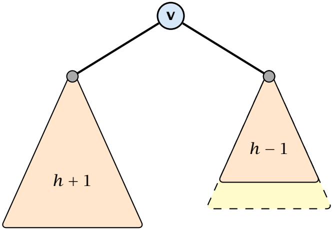
Wir zeigen, dass nach Rotation um v:

- v seine Balancebedingung erfüllt.
- Alle Kinder von v ihre Bedingung immer noch erfüllen
- Falls jetzt $\mathrm{balance}[v] \in \{-1, 1\}$ können wir aufhören, da der Teilbaum T_v die gleiche Höhe hat wie vor der Löschoperation.

Wir betrachten nur den Fall dass der entfernte Knoten im rechten Teilbaum von v war. Der andere Fall ist symmetrisch.

AVL-Bäume: Löschen

Folgende Situation:

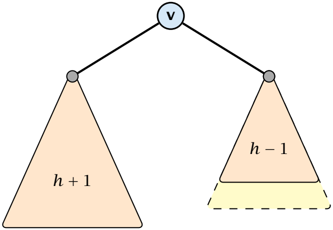


Der rechte Teilbaum von v hat seine Höhe reduziert. Dies ergibt eine Balance von 2 für v.

Vor der Löschoperation war die Höhe von T_v gleich $h+2$.

AVL-Bäume: Löschen

Folgende Situation:

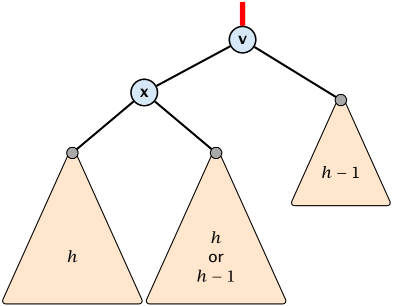


Der rechte Teilbaum von v hat seine Höhe reduziert. Dies ergibt eine Balance von 2 für v.

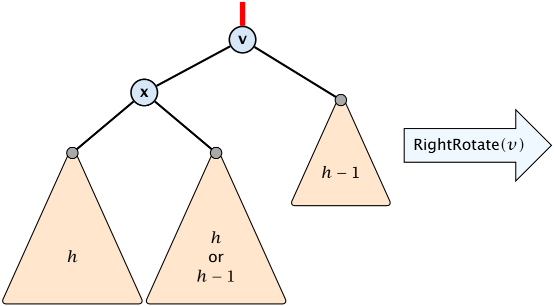
Vor der Löschoperation war die Höhe von T_v gleich $h+2$.

Fall 1: balance[left[v]] $\in \{0, 1\}$

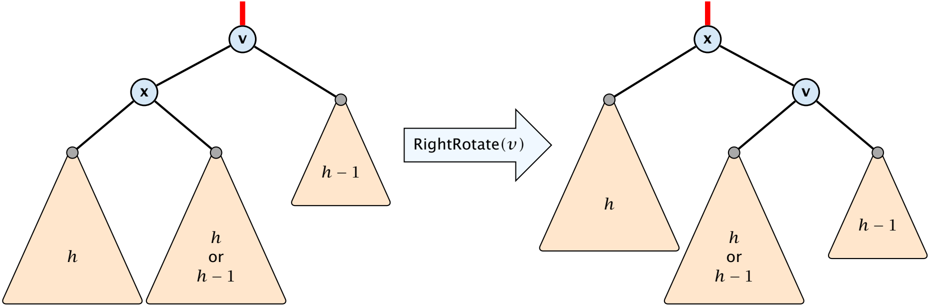
Fall 1: balance[left[v]] $\in \{0, 1\}$



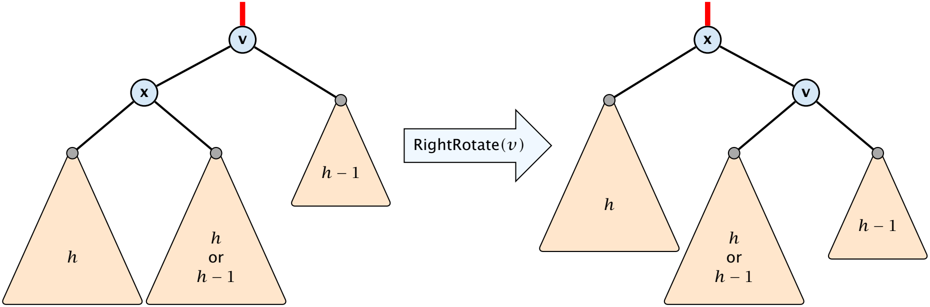
Fall 1: balance[left[v]] $\in$ {0, 1}



Fall 1: balance[left[v]] $\in \{0, 1\}$

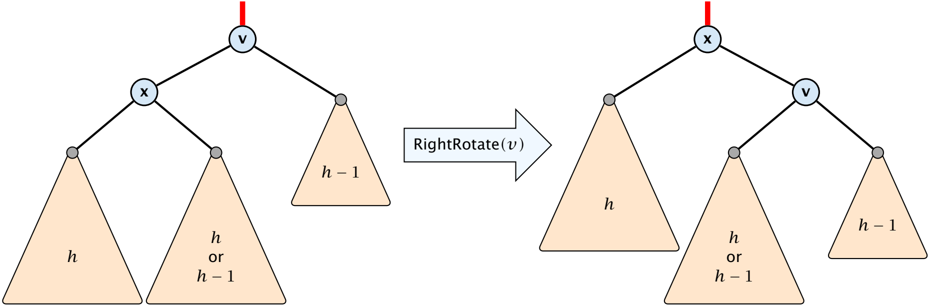


Fall 1: balance[left[v]] $\in \{0, 1\}$



Falls der mittlere Teilbaum Höhe h hat, hat der Gesamtaum Höhe $h + 2$ wie vor der Operation. Die Iteration bricht ab, da die Balance an der Wurzel nicht Null ist.

Fall 1: balance[left[v]] $\in \{0, 1\}$

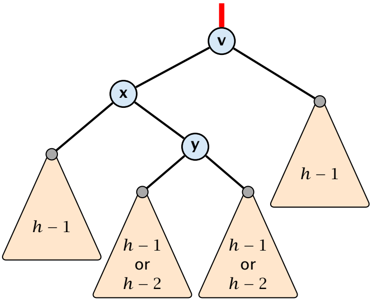


Falls der mittlere Teilbaum Höhe h hat, hat der Gesamtaum Höhe $h+2$ wie vor der Operation. Die Iteration bricht ab, da die Balance an der Wurzel nicht Null ist.

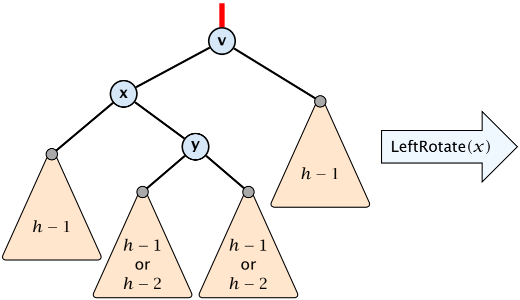
Falls der mittlere Teilbaum Höhe $h-1$ hat, hat der Gesamtbaum die Höhe von $h+2$ auf $h+1$ reduziert. Wir machen mit der fix-up Routine weiter.

Fall 2: $\mathrm{balance}[\mathrm{left}[v]] = -1$

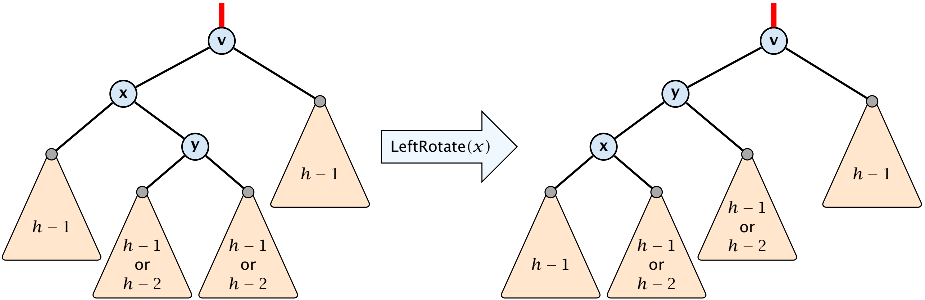
Fall 2: balance[left[v]] $= -1$



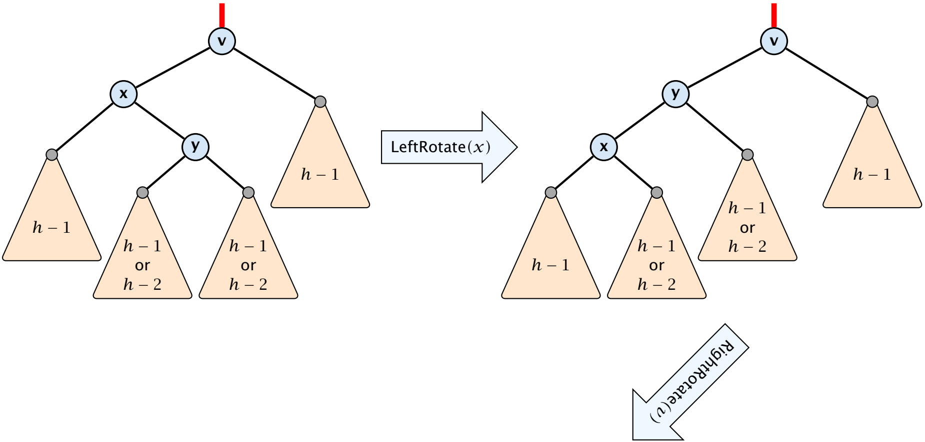
Fall 2: balance[left[v]] $= -1$



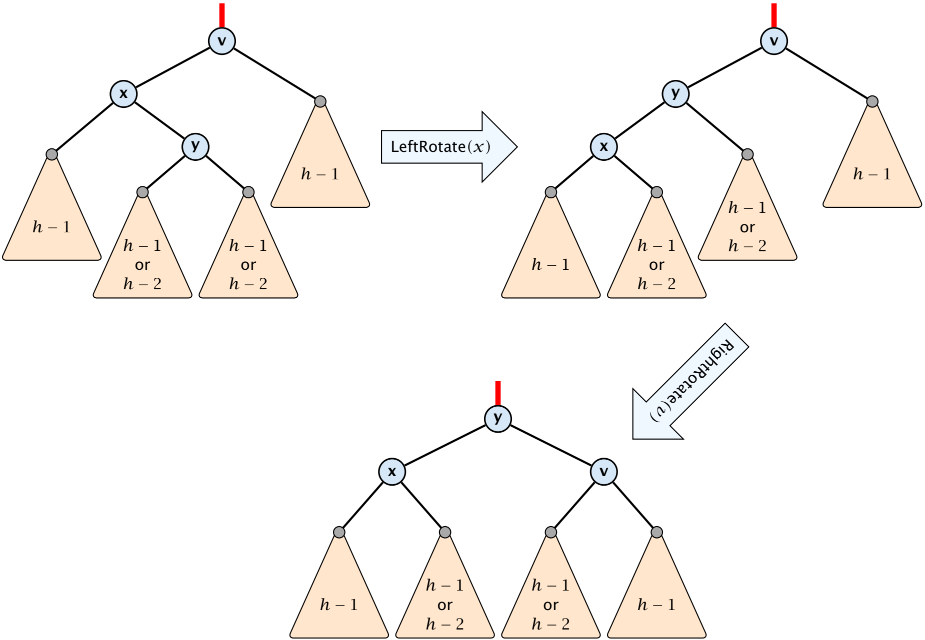
Fall 2: balance[left[v]] $= -1$



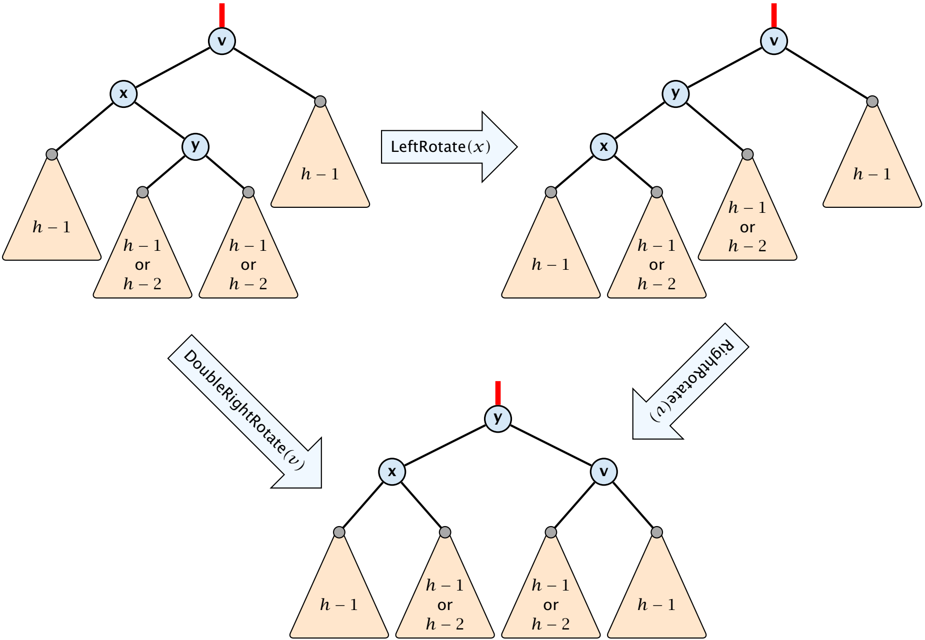
Fall 2: balance[left[v]] $= -1$



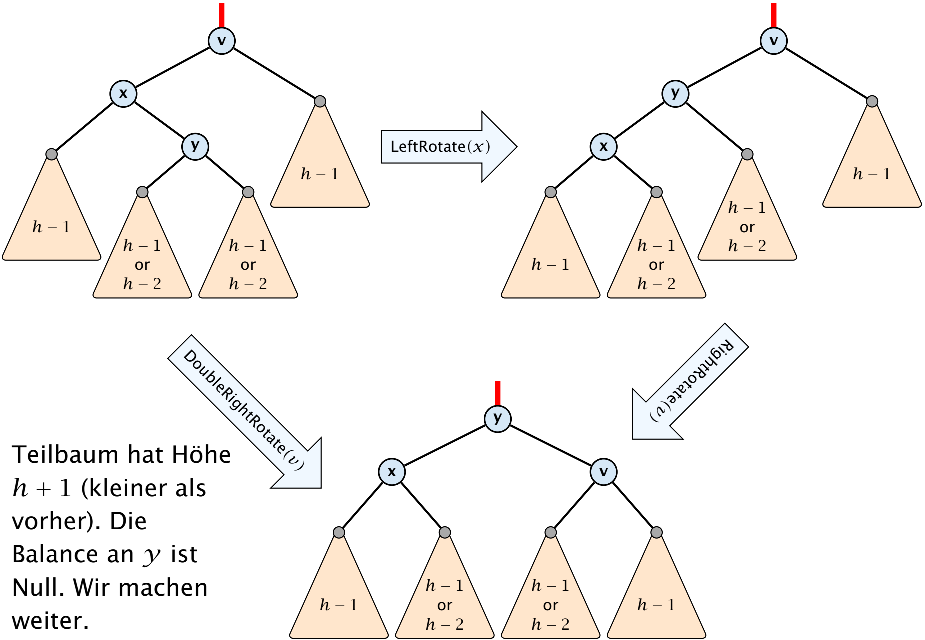
Fall 2: balance[left[v]] $= -1$



Fall 2: balance[left[v]] $= -1$



Fall 2: balance[left[v]] $= -1$



Teilbaum hat Höhe $h+1$ (kleiner als vorher). Die Balance an y ist Null. Wir machen weiter.