Informatik II für Verkehrsingenieure
Suchen (Kapitel 11)

Janis Voigtländer

Technische Universität Dresden

Sommersemester 2007

Überblick

Problemstellung

Lineares Suchen

Binäres Suchen

Suchbäume

AVL-Bäume

Problemstellung

Gegeben: Datenstruktur mit Einträgen eines Typs der Form:

```
typedef struct Entry
                { int key;
                  ... contents;
                } EntryTyp;
```

und ein zu suchender Schlüssel `int Value;`

Gesucht: Eintrag `E` der Datenstruktur mit `E.key=Value`

Konkret: wenn Datenstruktur ein Feld der Form `EntryTyp F[n]`, dann eine Position `i` gesucht, so dass `F[i].key=Value`

Vereinfacht:

- keine `contents` gespeichert, lediglich die Schlüsselwerte selbst
- Position nicht gefordert, lediglich Signal ob oder ob nicht gefunden

Beispiele:

12	7	9	8	4	6	+ 9 ↦ 1

12	7	9	8	4	6	+ 5 ↦ 0

Lineares Suchen

Naive Idee: Durchsuchen von links nach rechts

Beispiele:

12	7	9	8	4	6

9

12	7	9	8	4	6

5

In C:

```
int i=0;
while ((i<n) && (F[i]!=Value)) i++;
found=(i<n);
```

Komplexität: linearer Aufwand

Binäres Suchen

Voraussetzung: sortiertes Feld

Idee: „Intervallschachtelung“

Beispiele:

2	3	5	6	8	9	11	12	14	16	17	19	20	22	23

9

2	3	5	6	8	9	11	12	14	16	17	19	20	22	23

21

In C:

```
int search(int li,int re)
{ int pos;
  if (li>re) return 0;
  pos=(li+re)/2;
  if (F[pos]==Value) return 1;
  if (F[pos]<Value) return search(pos+1,re);
  return search(li,pos-1);
}
found=search(0,n-1);
```

Komplexität: Aufwand $\log(n)$

Was nun?

Problem:

- effiziente Suche nur in sortierter Datenstruktur möglich
- Sortieren vor jedem Suchvorgang keine Option ($n \cdot \log(n)$)
- einmaliges Sortieren nicht genug, falls Datenstruktur erweiterbar sein soll

Lösung:

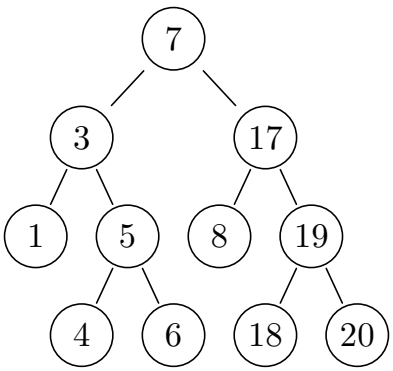
- Verwendung einer sortierten Datenstruktur, in welche effizient eingefügt werden kann, unter Beibehaltung der Sortierung
- aber: sowohl Feld als auch verkettete Liste zu unflexibel
- stattdessen: Bäume

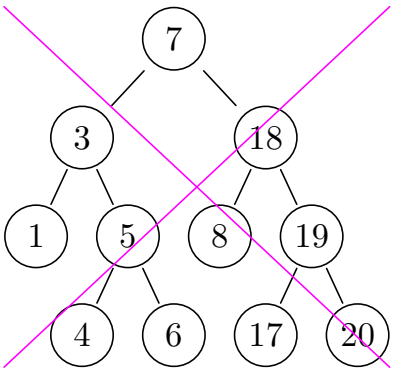
Suchbäume

Definition: für jeden Knoten:

- alle Knoten in linkem Nachfolgerbaum mit kleinerer Zahl beschriftet
- alle Knoten in rechtem Nachfolgerbaum mit größerer Zahl beschriftet

Beispiele:

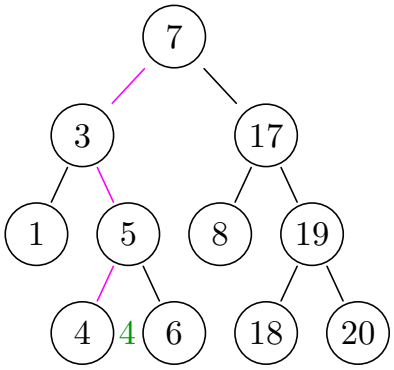


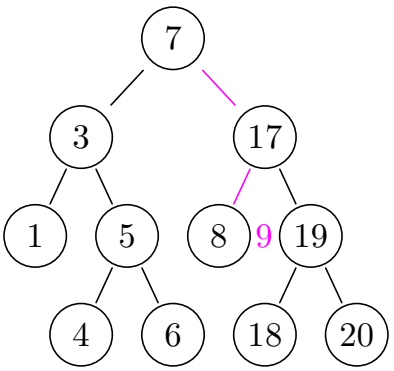


Suchen in Suchbäumen

Idee: analog zu binärem Suchen

Beispiele:





Aufwand: entsprechend Entfernung der Blätter von der Wurzel

Suchbäume in C

```
typedef struct Nodeelem *Ptr;

typedef struct Nodeelem { int key;
                          Ptr left, right;
                        } Node;

int search(Ptr t,int x)
{ if (t==NULL) return 0;
  if (t->key == x) return 1;
  if (t->key < x) return search(t->right,x);
  return search(t->left,x);
}
```


Einfügen in Suchbäume

Idee: zunächst suchen; wenn nicht vorhanden, neues Blatt erzeugen

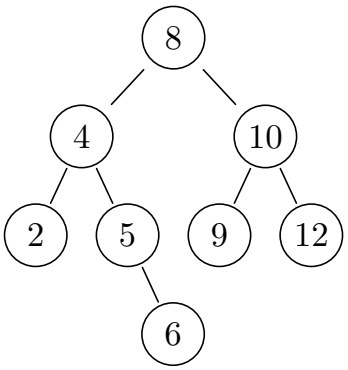
Beispiel:

8	4	5	10	2	9	6	12

Einfügen in Suchbäume

Idee: zunächst suchen; wenn nicht vorhanden, neues Blatt erzeugen

Beispiel:



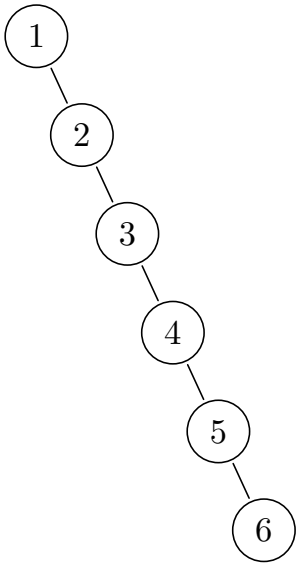
In C:

```
void insert(Ptr *t,int x)
{ if (*t==NULL)
    { *t=(Ptr) malloc(sizeof(Node));
      (*t)->key=x;
      (*t)->left=NULL;
      (*t)->right=NULL;
    }
  else
    { if ((*t)->key < x) insert(&((*t)->right),x);
      else if ((*t)->key > x) insert(&((*t)->left),x);
    }
}
```

Unbalancierte Bäume

Problem: Form der Bäume, und somit Aufwand beim Suchen und Einfügen, hängt stark von Einfügereihenfolge ab

Beispiel:



Lösung: Umbalancieren während des Einfügens

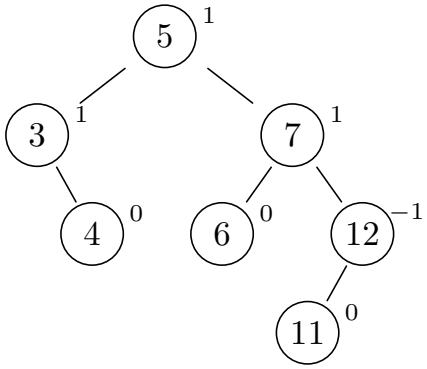
Ziel: sowohl Suchen als auch Einfügen mit Aufwand $\log(n)$

AVL-Bäume (nach Adelson-Velskij und Landis)

Definition:

- Suchbaum
- an jedem Knoten Höhenunterschied zwischen rechtem und linkem Nachfolgerbaum nicht größer als Eins

Beispiel:

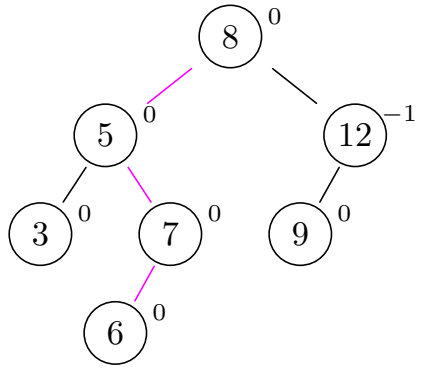


Balancefaktor: Differenz soll stets -1, 0 oder 1 sein

Einfügen in AVL-Bäume (I)

1. Einfügen: wie bei Suchbäumen als neues Blatt an (einzig) geeigneter Stelle

   Beispiel:

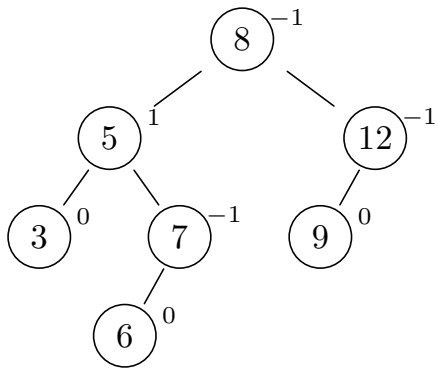


2. Aktualisieren: Balancefaktoren entlang des Suchpfades anpassen

Einfügen in AVL-Bäume (I)

1. Einfügen: wie bei Suchbäumen als neues Blatt an (einzig) geeigneter Stelle

   Beispiel:

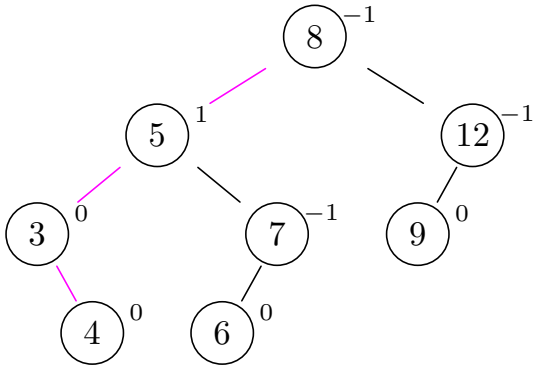


2. Aktualisieren: Balancefaktoren entlang des Suchpfades anpassen

Einfügen in AVL-Bäume (II)

Variante: Anpassung der Balancefaktoren nicht immer bis zur Wurzel nötig

Beispiel:

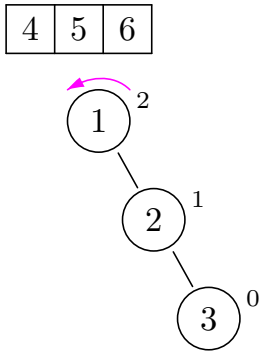


Problem: Balancefaktoren können den Bereich $-1, 0, 1$ verlassen

Lösung: Rotationen

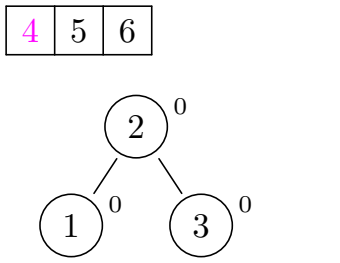
Einfügen in AVL-Bäume (III)

Beispiel:



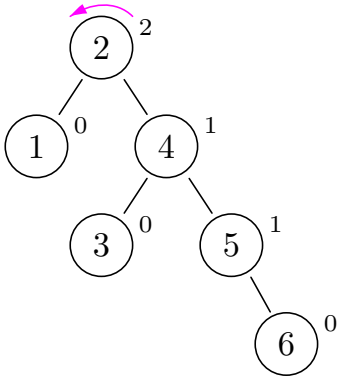
Einfügen in AVL-Bäume (III)

Beispiel:



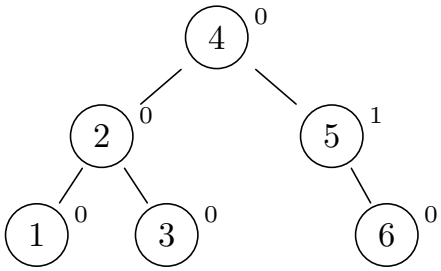
Einfügen in AVL-Bäume (III)

Beispiel:



Einfügen in AVL-Bäume (III)

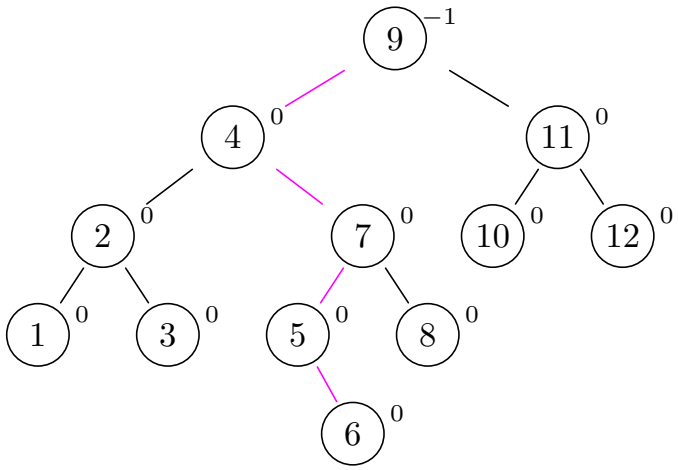
Beispiel:



Einfügen in AVL-Bäume (IV)

Problem: einfache Rotation nicht immer ausreichend

Beispiel:



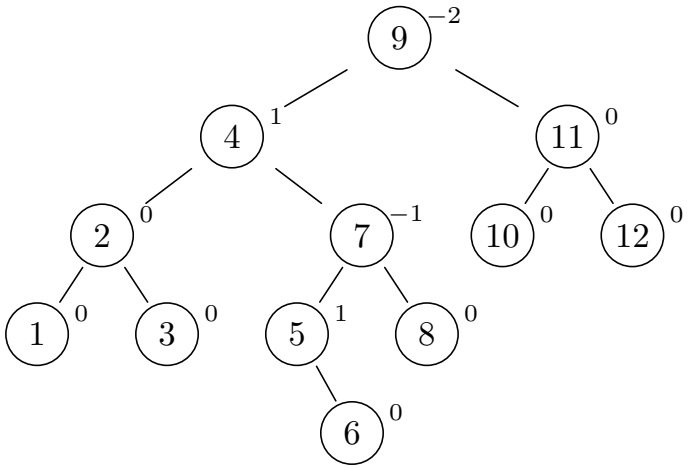
$\rightsquigarrow$ Rotation an Wurzel führt nicht zu AVL-Baum!

Lösung: in bestimmten Fällen Doppelrotation

Einfügen in AVL-Bäume (IV)

Problem: einfache Rotation nicht immer ausreichend

Beispiel:



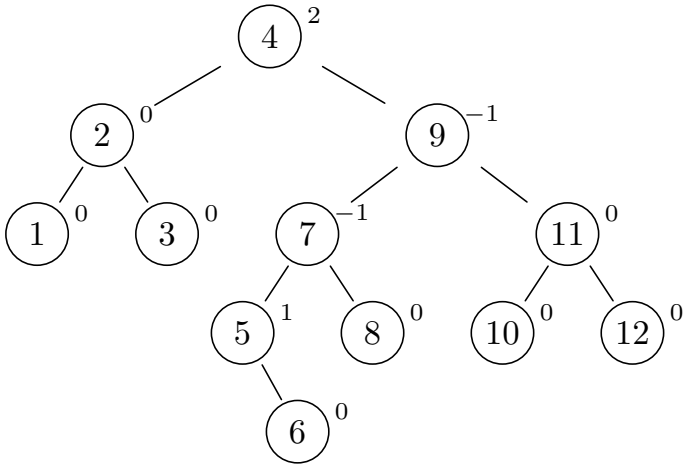
⇝ Rotation an Wurzel führt nicht zu AVL-Baum!

Lösung: in bestimmten Fällen Doppelrotation

Einfügen in AVL-Bäume (IV)

Problem: einfache Rotation nicht immer ausreichend

Beispiel:

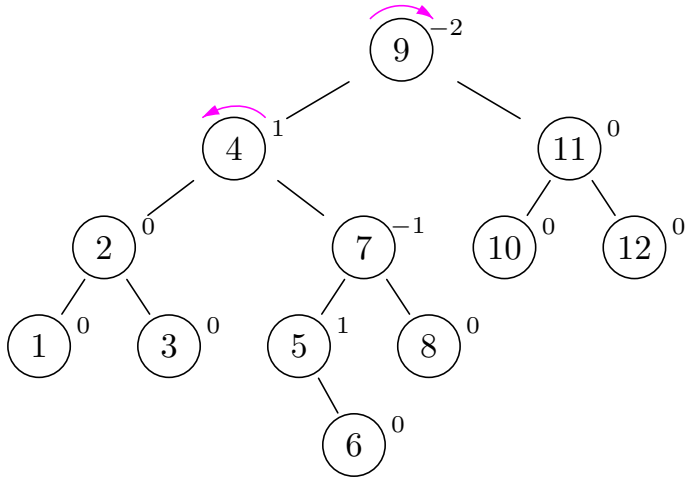


⇝ Rotation an Wurzel führt nicht zu AVL-Baum!

Lösung: in bestimmten Fällen Doppelrotation

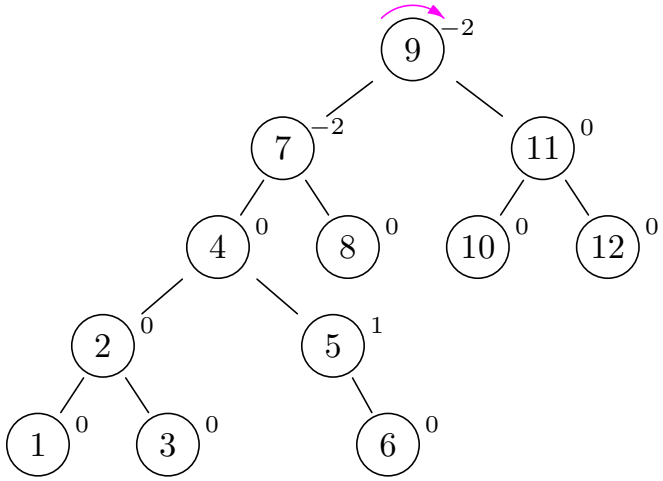
Einfügen in AVL-Bäume (V)

Beispiel:



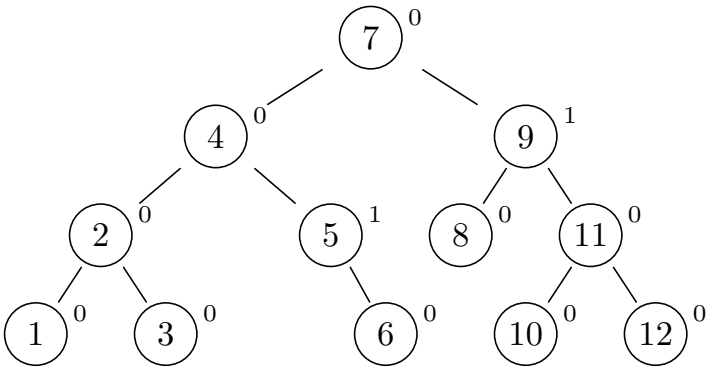
Einfügen in AVL-Bäume (V)

Beispiel:



Einfügen in AVL-Bäume (V)

Beispiel:



Einfügen in AVL-Bäume (VI)

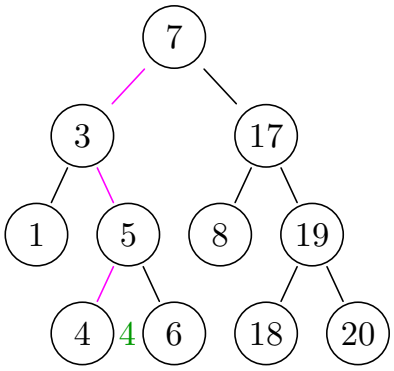
1. Sofern noch nicht vorhanden, füge das neue Element als Blatt so ein, dass die Suchbaumeigenschaft erfüllt ist, und setze dessen Balancefaktor auf 0.
2. Falls noch nicht an der Wurzel des Baumes, gehe zum Vorgängerknoten. Dort, falls
   - 2.1 aus linkem Nachfolgerbaum kommend:
     - 2.1.1 wenn Balancefaktor gleich 1, dann auf 0 setzen und Abbruch
     - 2.1.2 wenn Balancefaktor gleich 0, dann auf -1 setzen und zu 2.
     - 2.1.3 wenn Balancefaktor gleich -1, dann Rotation(en) gemäß Fallunterscheidung bezüglich Balancefaktor am linken Nachfolgerknoten. . . , und Abbruch
   - 2.2 aus rechtem Nachfolgerbaum kommend:
     . . . entsprechend „umgekehrt“

Wiederholung — Suchbäume

Definition: für jeden Knoten:

- alle Knoten in linkem Nachfolgerbaum mit kleinerer Zahl beschriftet
- alle Knoten in rechtem Nachfolgerbaum mit größerer Zahl beschriftet

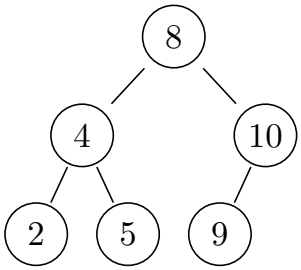
Beispiel:



Wiederholung — Einfügen in Suchbäume

Beispiel:

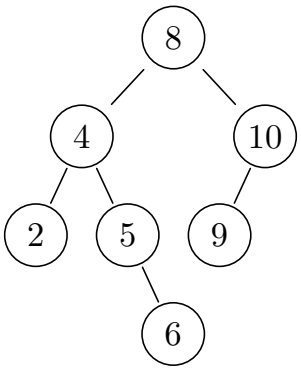
6	12



Wiederholung — Einfügen in Suchbäume

Beispiel:

12

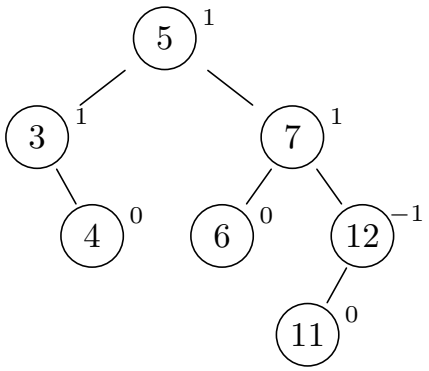


Wiederholung — AVL-Bäume

Definition:

- Suchbaum
- an jedem Knoten Höhenunterschied zwischen rechtem und linkem Nachfolgerbaum nicht größer als Eins

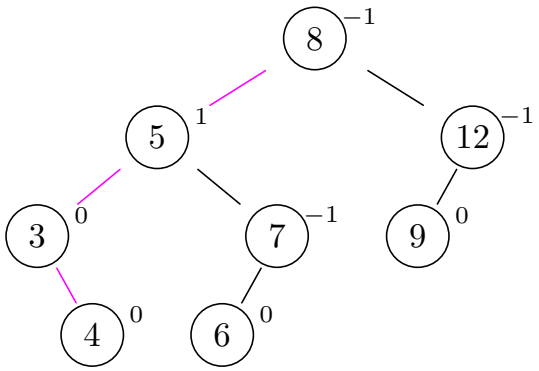
Beispiel:



Balancefaktor: Differenz soll stets -1, 0 oder 1 sein

Wiederholung — Einfügen in AVL-Bäume

Beispiel:

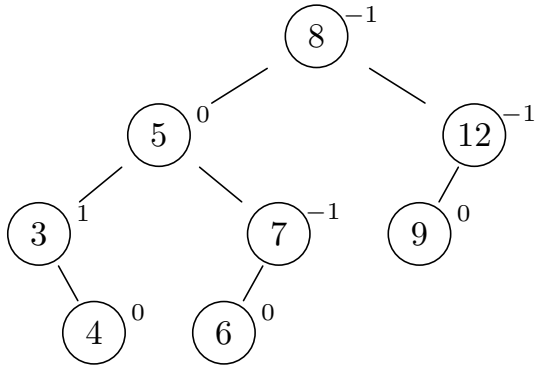


Problem: Balancefaktoren können den Bereich $-1, 0, 1$ verlassen

Lösung: Rotationen

Wiederholung — Einfügen in AVL-Bäume

Beispiel:

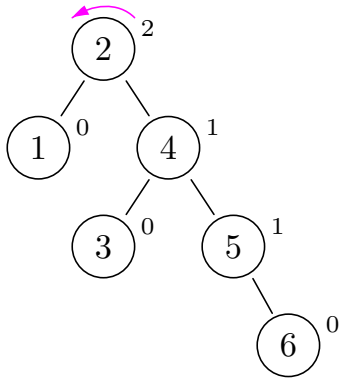


Problem: Balancefaktoren können den Bereich $-1, 0, 1$ verlassen

Lösung: Rotationen

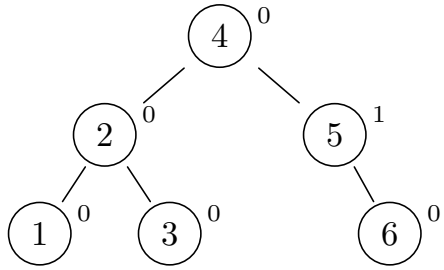
Wiederholung — Einfache Rotation

Beispiel:



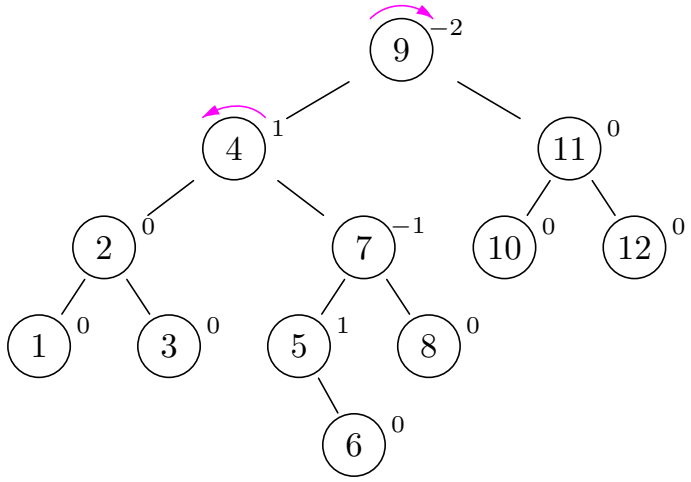
Wiederholung — Einfache Rotation

Beispiel:



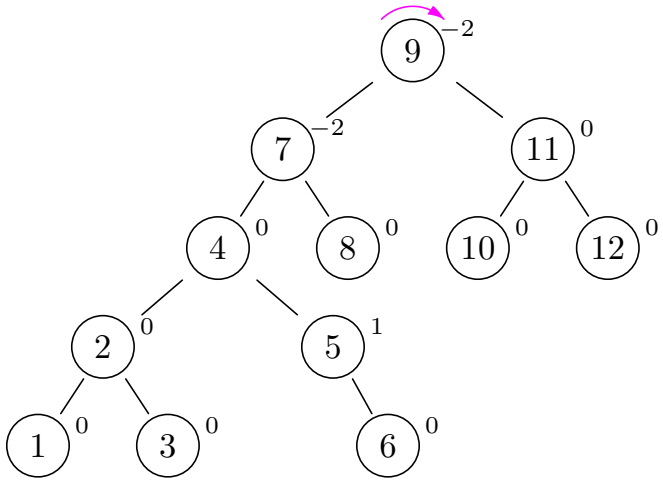
Wiederholung — Doppelrotation

Beispiel:



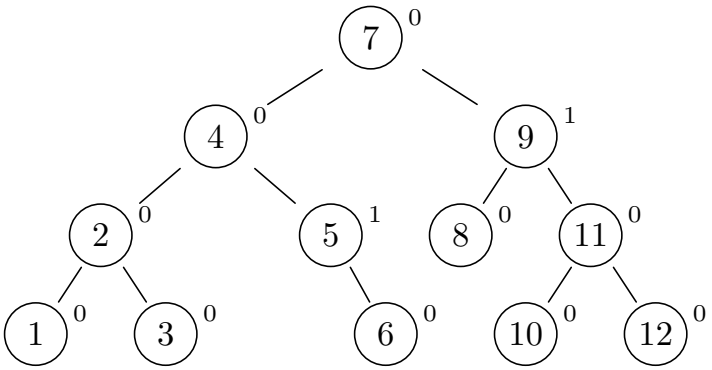
Wiederholung — Doppelrotation

Beispiel:



Wiederholung — Doppelrotation

Beispiel:



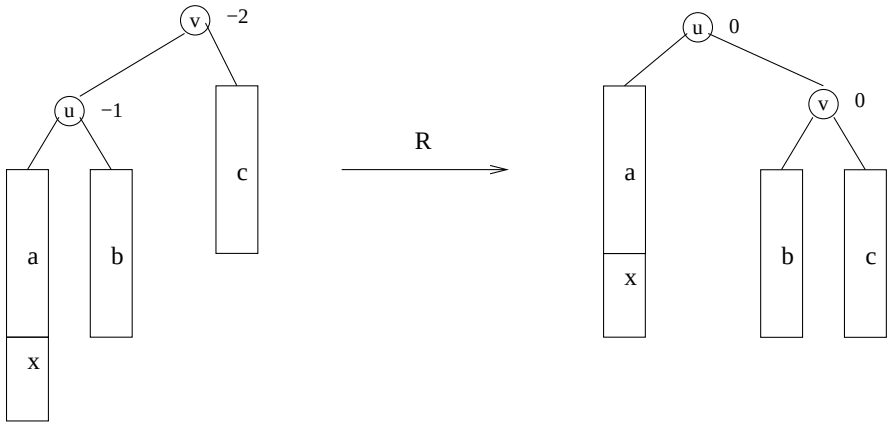
Wiederholung — Einfügen in AVL-Bäume (VI)

1. Sofern noch nicht vorhanden, füge das neue Element als Blatt so ein, dass die Suchbaumeigenschaft erfüllt ist, und setze dessen Balancefaktor auf 0.
2. Falls noch nicht an der Wurzel des Baumes, gehe zum Vorgängerknoten. Dort, falls
   - 2.1 aus linkem Nachfolgerbaum kommend:
     - 2.1.1 wenn Balancefaktor gleich 1, dann auf 0 setzen und Abbruch
     - 2.1.2 wenn Balancefaktor gleich 0, dann auf -1 setzen und zu 2.
     - 2.1.3 wenn Balancefaktor gleich -1, dann Rotation(en) gemäß Fallunterscheidung bezüglich Balancefaktor am linken Nachfolgerknoten. . . , und Abbruch
   - 2.2 aus rechtem Nachfolgerbaum kommend:
     . . . entsprechend „umgekehrt“

Einfügen in AVL-Bäume (VII)

Fallunterscheidung an einem Knoten, in dessen linkem Nachfolgerbaum das neue Element eingefügt wurde, und dessen vorheriger Balancefaktor gleich -1 war:

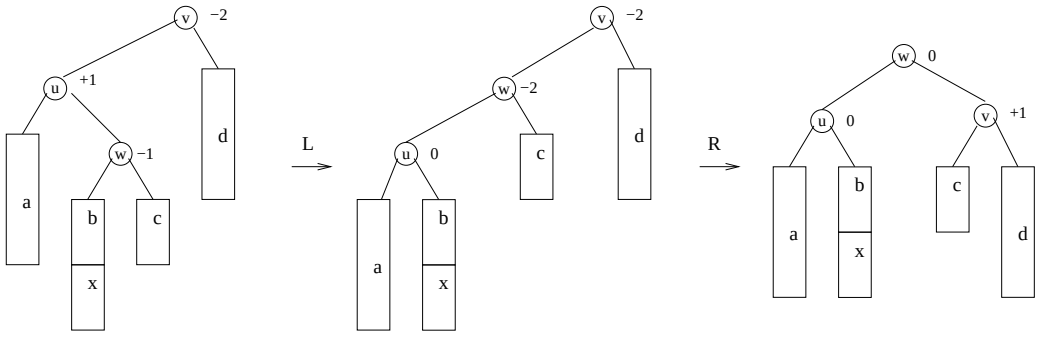
- wenn Balancefaktor am linken Nachfolgerknoten gleich -1, dann Rechtsrotation am aktuellen Knoten:



Einfügen in AVL-Bäume (VII)

Fallunterscheidung an einem Knoten, in dessen linkem Nachfolgerbaum das neue Element eingefügt wurde, und dessen vorheriger Balancefaktor gleich -1 war:

- wenn Balancefaktor am linken Nachfolgerknoten gleich 1, dann Linksrotation am linken Nachfolgerknoten und anschließend Rechtsrotation am aktuellen Knoten:

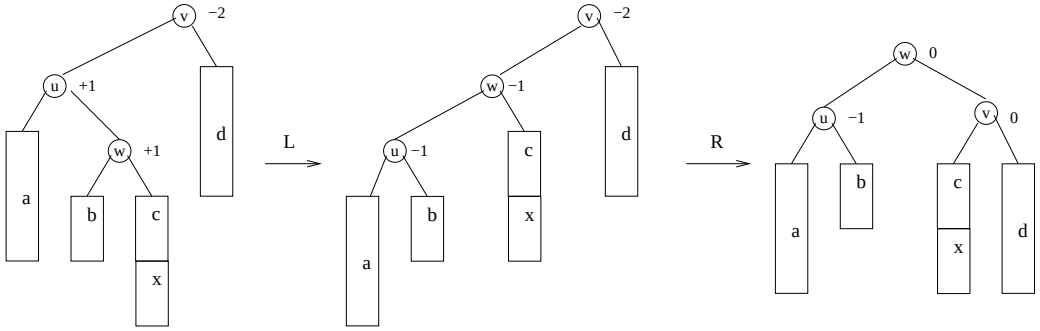


beziehungsweise...

Einfügen in AVL-Bäume (VII)

Fallunterscheidung an einem Knoten, in dessen linkem Nachfolgerbaum das neue Element eingefügt wurde, und dessen vorheriger Balancefaktor gleich -1 war:

- wenn Balancefaktor am linken Nachfolgerknoten gleich 1, dann Linksrotation am linken Nachfolgerknoten und anschließend Rechtsrotation am aktuellen Knoten:

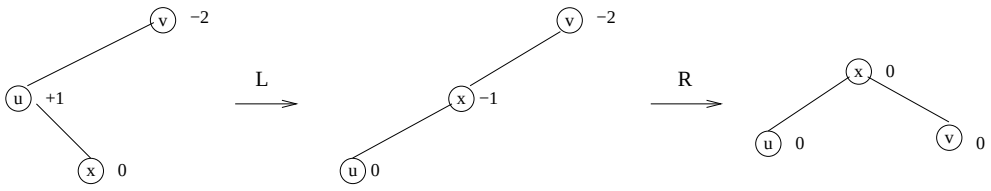


oder...

Einfügen in AVL-Bäume (VII)

Fallunterscheidung an einem Knoten, in dessen linkem Nachfolgerbaum das neue Element eingefügt wurde, und dessen vorheriger Balancefaktor gleich -1 war:

- wenn Balancefaktor am linken Nachfolgerknoten gleich 1, dann Linksrotation am linken Nachfolgerknoten und anschließend Rechtsrotation am aktuellen Knoten:



Einfügen in AVL-Bäume (VIII)

Beispiel:

7	12	3	8	5	6	4	11

Einfügen in AVL-Bäume (VIII)

Beispiel: