Wörterbücher

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Dynamische Menge an Elementen, die über Schlüssel definiert sind:

Paare(Schlüssel, Wert)

Operationen:
element search*(schlüssel* x*)*
void insert*(element* e*)*
void delete(*schlüssel* x)
element max/min()
void traverse()

Realisirung:

- Verketette Liste
- Feld
- BitVector mit Kapazität m

	Search	**Insert**	**Delete**	**Max/Min**	**Traverse**
Liste	O(n)	O(n)	O(n)	O(n)	$O(n^2)$
Feld	O(log n)	O(n)	O(n)	O(1)	O(n)
BitVector	O(1)	O(1)	O(1)	O(m)	O(m)

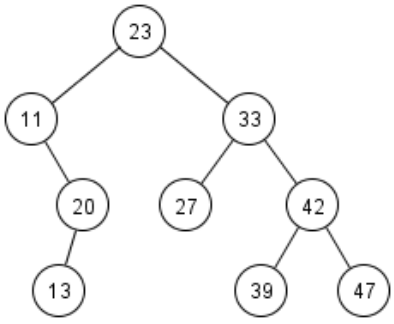
Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Ein binärer Suchbaum ist ein binärer Baum mit der Folgender Eigenschaft:
- **z.key > x.key ∀z im rechten Teilbaum von x**
- **z.key < x.kez ∀z im linken Teilbaum von x**

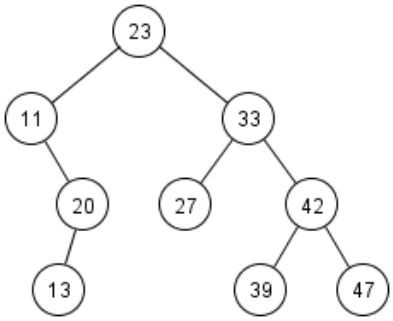
Knoten *x*:
knoten x.left
knoten x.right
knoten x.parent
schlüssel x.key
wert x.value

A={ 23 , 33 , 27 , 11 , 42 , 20 , 13 , 39 , 47 }



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE




```
void traverse (Knoten  x)
if x != NULL then
    traverse(x.left)
    print(x)
    traverse(x.right)
```

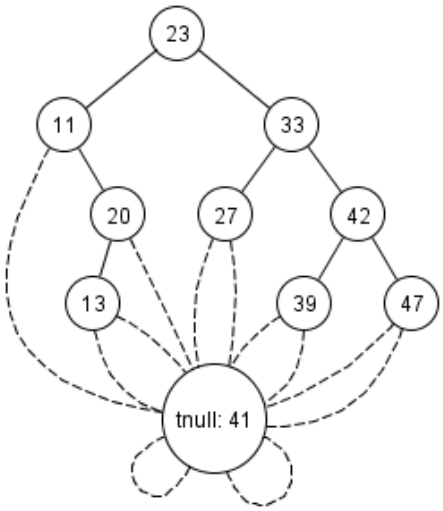
```
Knoten max (Knoten x)
if x.right != NULL
    then return Max(x.right)
    else return x
```


Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Suchen nach 41

Wächter (Sentinel) TNULL
- Speziller Knoten, der kein Element permanent abspeichert
- Alle Null Zeiger zeigen jetzt zum Sentinel
- Sentinel ist dereferenzierbar und erlaubt, Speziallfälle zu vermeiden
- TNULL.left = TNULL
- TNULL.right = TNULL



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
Knoten search (Knoten  x, Schlüssel k)
TNULL.key = k
z = sentinel_search(x, k)
if z != NULL
    then return z
    else return NULL

Knoten sentinel_search (Knoten  x, Schlüssel k)
if x.key > k  then return sentinel_search(x.left, k)
else if x.key < k then return sentinel_search(x.right, k)
else return x
```

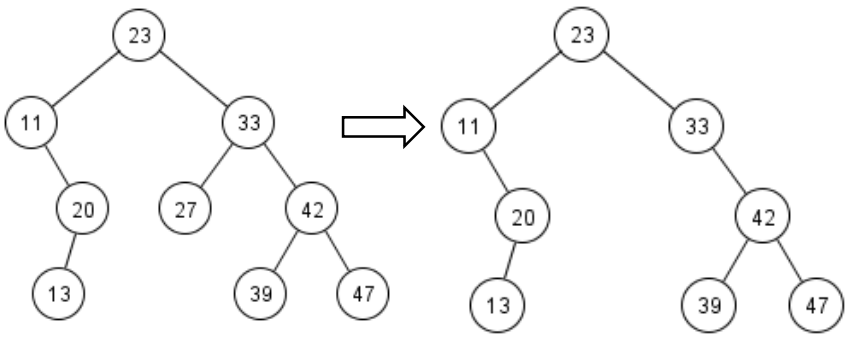
```
void insert (Knoten  x, Schlüssel k)
if x=TNULL then x = makenode( k )
else if x.key > k  then insert(x.left, k)
else if x.key < k then insert(x.right, k)
else key k already in tree
```

Hängt neues Blatt mit Schlüssel *k* an die richtige Stelle im Baum.

Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

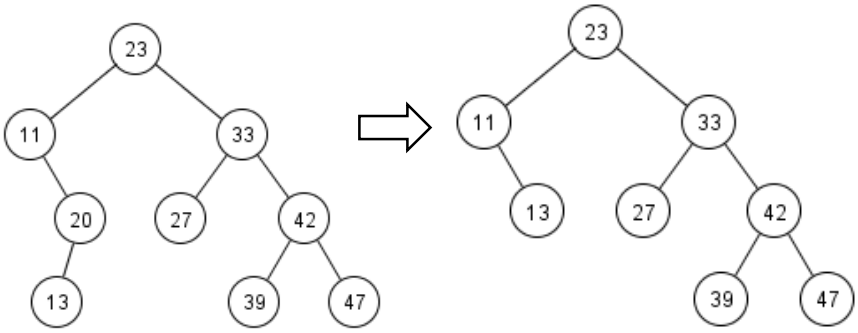
27 löschen:



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

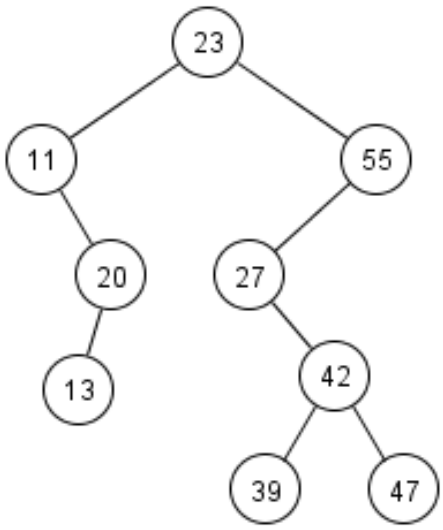
20 löschen:



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

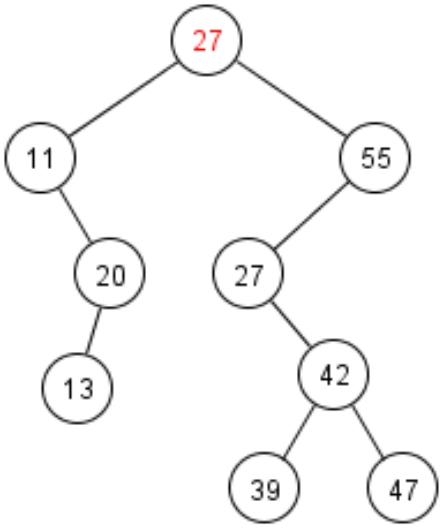
23 löschen:



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

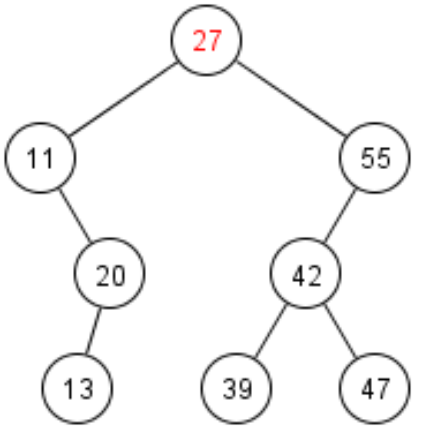
23 löschen:



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

23 löschen:



Binäre Suchbäume

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Drei Fälle beim Löschen von *x*:
• x ist Blattknoten – einfach entfernen
• x hat genau ein Kind z – x entfernen und Vater von x mit z verbinden
• x hat zwei Kinder
  - Nachfolger z von x hat nur ein Kind
  - ersetze x durch z
  - lösche z

```
void delete(Knoten  x, Schlüssel k)
if x=TNULL then x not in tree
else if x.key > z.key  then delete(x.left, z)
else if x.key < z.key then delete(x.right, z)
else if x.left = TNULL then x = x.right
else if x.right = TNULL then x = x.left
else z = min(x.right)
      copy data from z to x
      delete(x.right, z.key)
```

Laufzeit search, insert, delete: O(h), h = Höhe des Baumes
Algorithmen durchläuft maximal einen Pfad von der Wurzel zu einem (leeren) Blatt

Wie können wir einen balancierten binärer Suchbaum konstruieren?