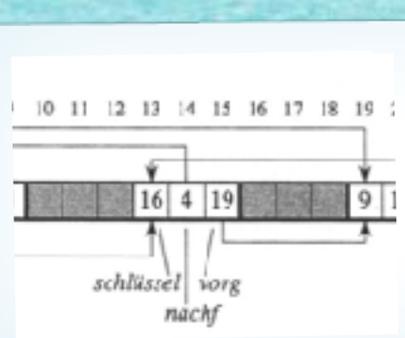



Kapitel 4.6: AVL-Bäume

Algorithmen und Datenstrukturen
WS 2021/22

Prof. Dr. Sándor Fekete

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

  Immer gleich schnell, egal wie groß S ist.

Schnell:

- ***O(log n):*** ***logarithmische Zeit***

  Wiederholtes Halbieren

4.5 Binäre Suchbäume

4.5 Binäre Suchbäume

Schnell:

4.5 Binäre Suchbäume

Schnell:

- ***O(log n): logarithmische Zeit***

4.5 Binäre Suchbäume

Schnell:

- ***O(log n): logarithmische Zeit***
- ***O(h): Tiefe des Baumes***

4.5 Binäre Suchbäume

Schnell:

- ***O(log n): logarithmische Zeit***
- ***O(h): Tiefe des Baumes***

4.5 Binäre Suchbäume

<u>Schnell:</u>

- ***O(log n): logarithmische Zeit***
- ***O(h): Tiefe des Baumes***

<u>Also: Wie können wir die Tiefe des Baumes auf O(log n) beschränken?</u>

4.6 AVL-Bäume

Definition 4.7 (Nach Adel'son-Vel'skiĭ und Landis, 1962)

Definition 4.7 **(Nach Adel'son-Vel'skiĭ und Landis, 1962)**

(1) Ein binärer Suchbaum ist *höhenbalanciert*, wenn sich für jeden inneren Konten v die Höhe der beiden Kinder von v um höchstens 1 unterscheidet.

Definition 4.7 (Nach Adel'son-Vel'skiĭ und Landis, 1962)

(1) Ein binärer Suchbaum ist ***höhenbalanciert***, wenn sich für jeden inneren Konten v die Höhe der beiden Kinder von v um höchstens 1 unterscheidet.

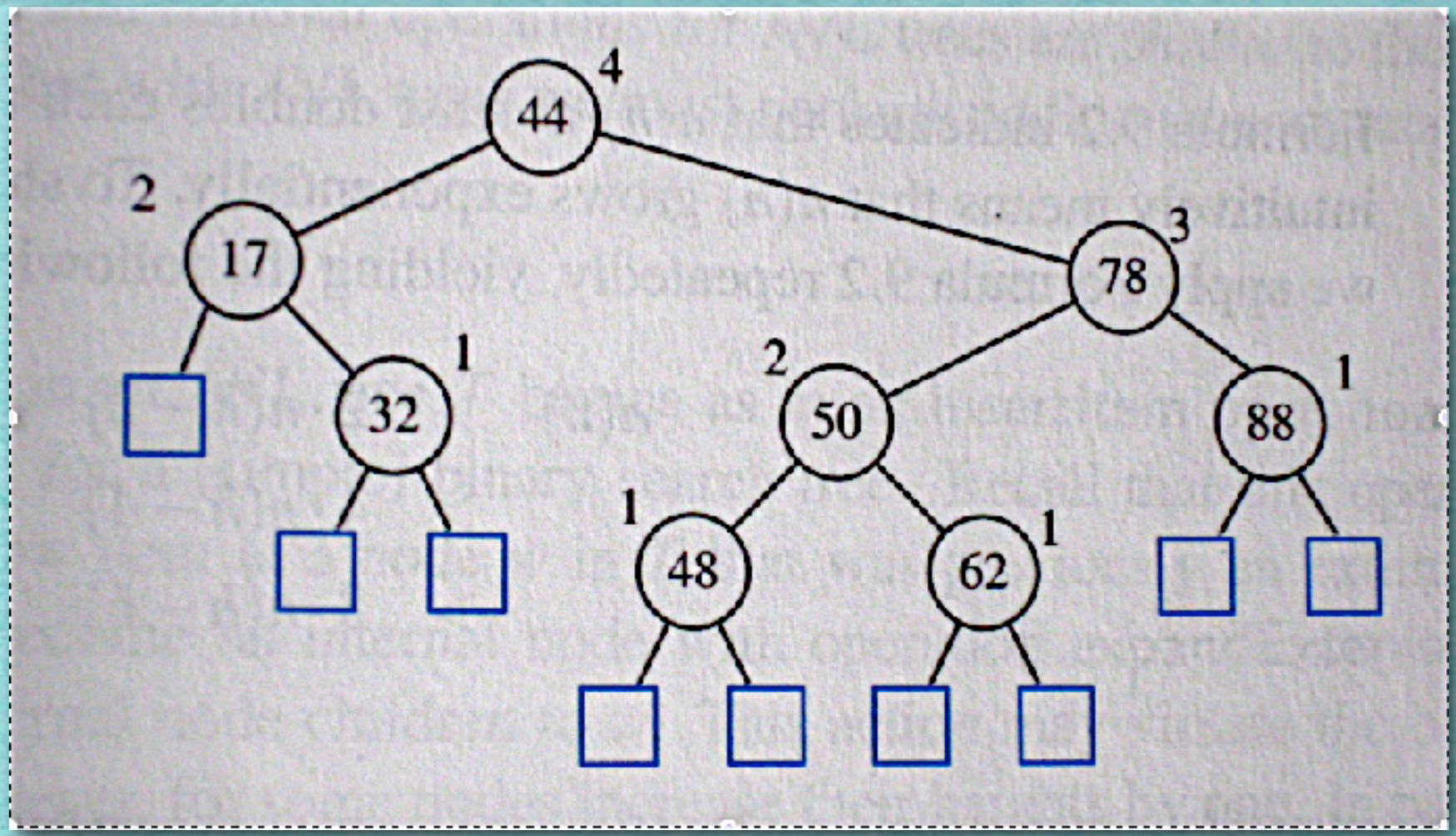
(2) Ein höhenbalancierter Suchbaum heißt auch ***AVL-Baum***.

4.6 AVL-Bäume

Definition 4.7 (Nach Adel'son-Vel'skiĭ und Landis, 1962)

(1) Ein binärer Suchbaum ist ***höhenbalanciert***, wenn sich für jeden inneren Konten v die Höhe der beiden Kinder von v um höchstens 1 unterscheidet.

(2) Ein höhenbalancierter Suchbaum heißt auch ***AVL-Baum***.



4.6 AVL-Bäume

Satz 4.8

Satz 4.8

Ein AVL-Baum mit n Knoten hat höchstens Höhe O(log n).

Satz 4.8

Ein AVL-Baum mit n Knoten hat höchstens Höhe O(log n).

Beweis:

Satz 4.8

Ein AVL-Baum mit n Knoten hat höchstens Höhe O(log n).

Beweis:

Wie gesehen!

Satz 4.8
Ein AVL-Baum mit n Knoten hat höchstens Höhe O(log n).

Beweis:
Wie gesehen!

Damit noch offen:

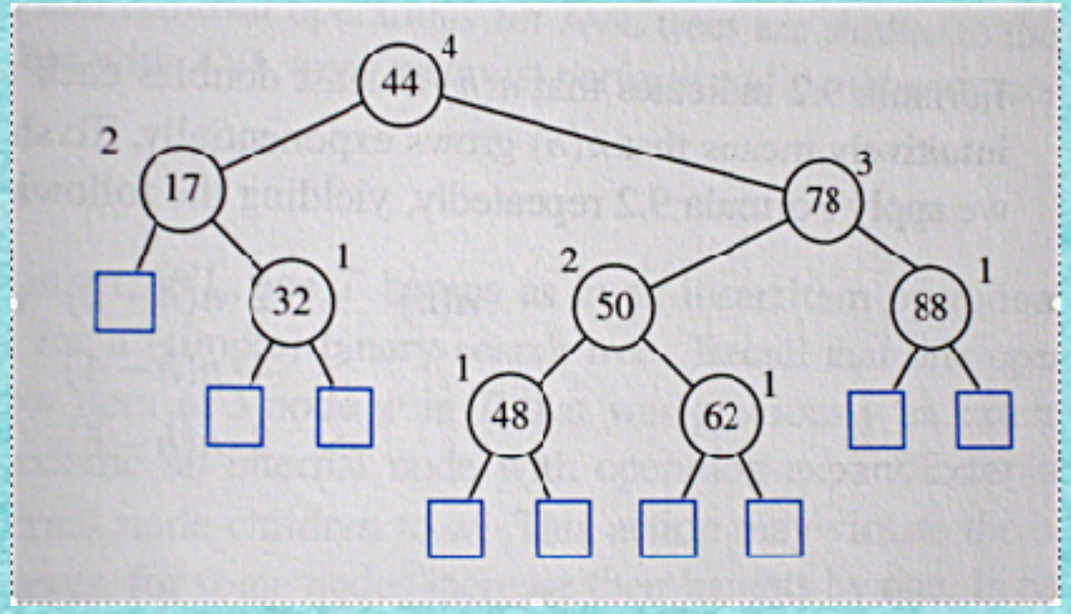
Satz 4.8
Ein AVL-Baum mit n Knoten hat höchstens Höhe O(log n).

Beweis:
Wie gesehen!

Damit noch offen:
Wie erhält man Höhenbalanciertheit in dynamischen Situationen?

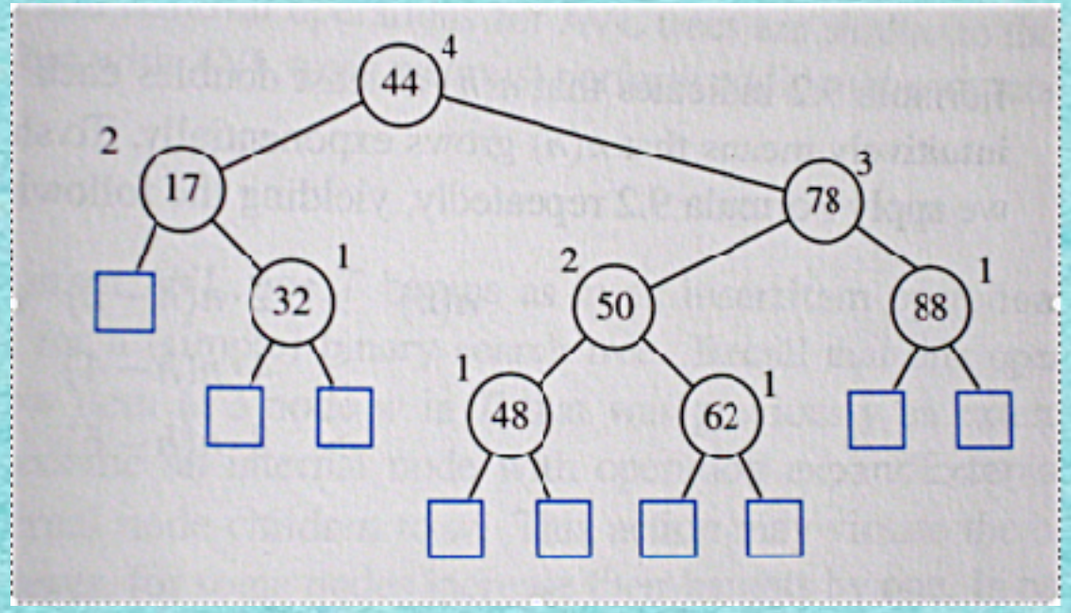
Einfügen („INSERT“)

Einfügen („INSERT“)



Einfügen („INSERT“)

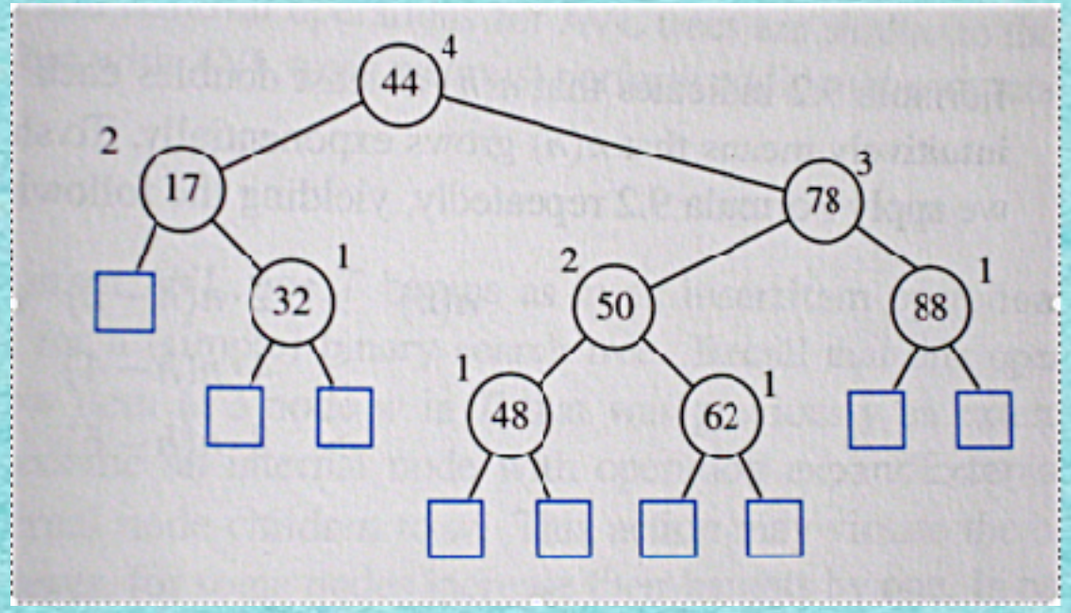
Aufgabe:



Einfügen („INSERT“)

<u>Aufgabe:</u>

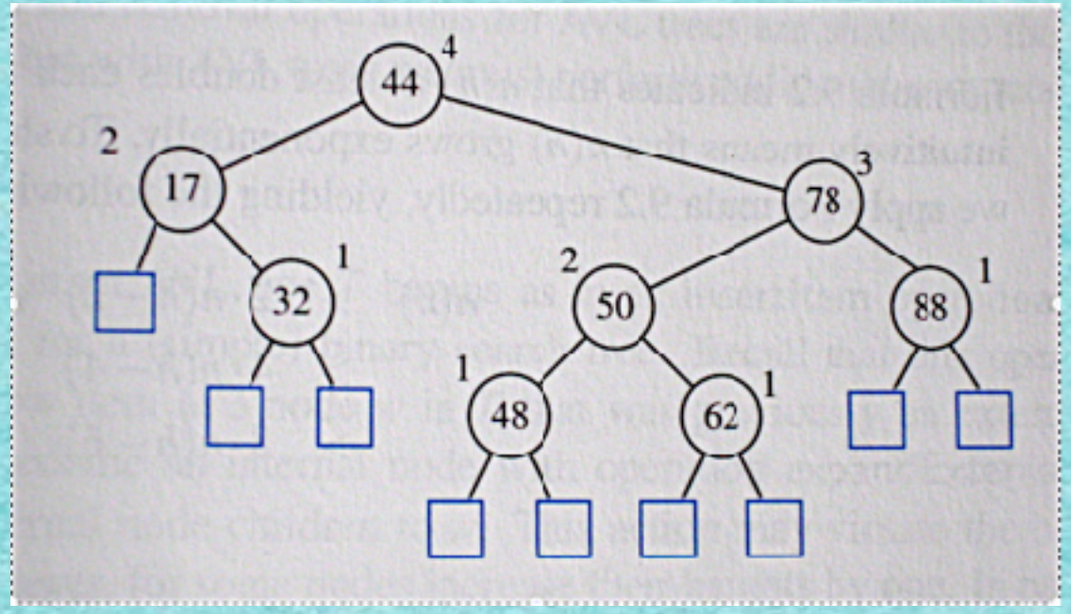
- ***Füge 54 ein!***

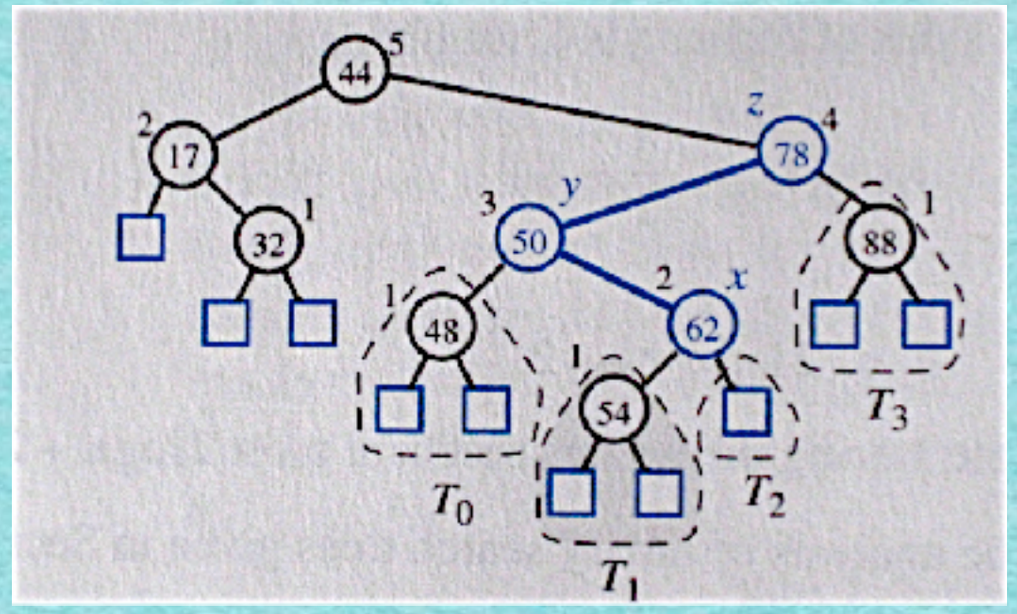


Einfügen („INSERT“)

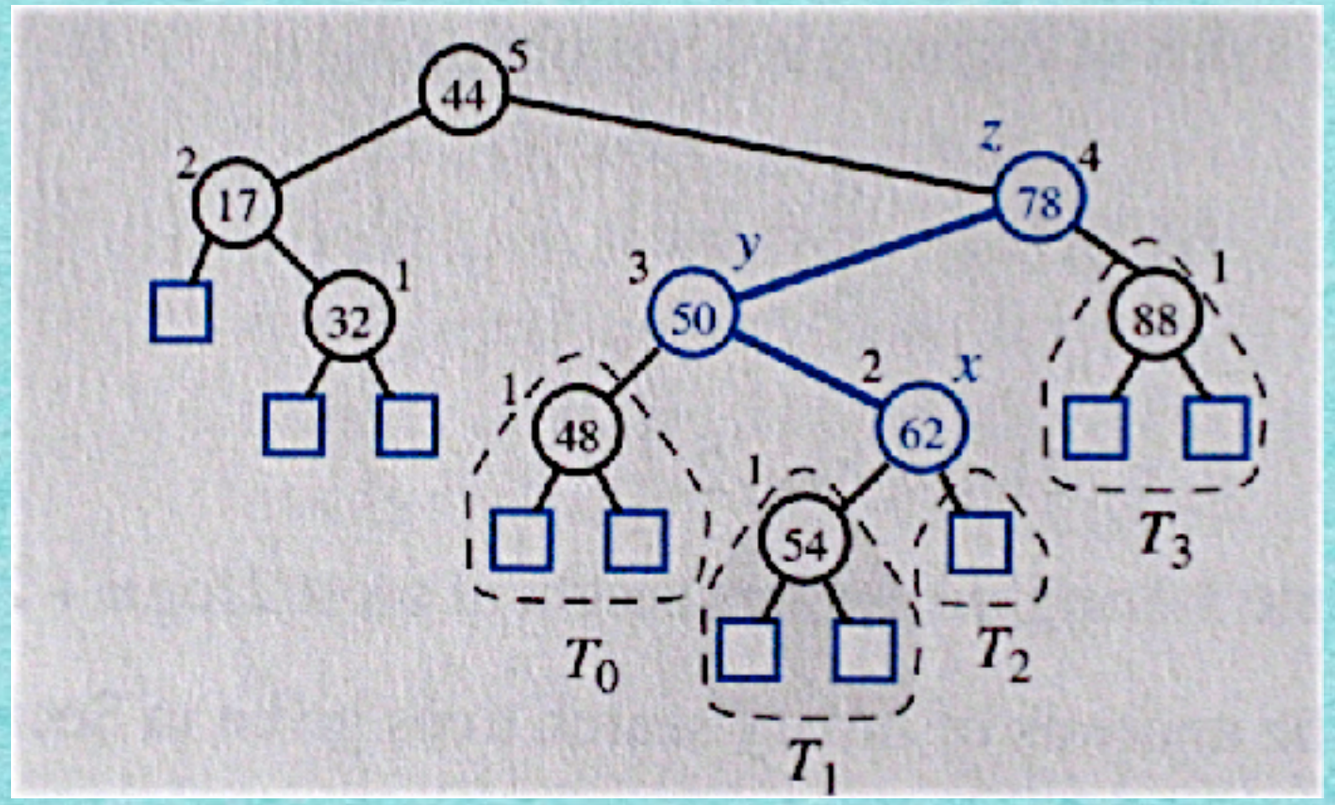
Aufgabe:

- ***Füge 54 ein!***



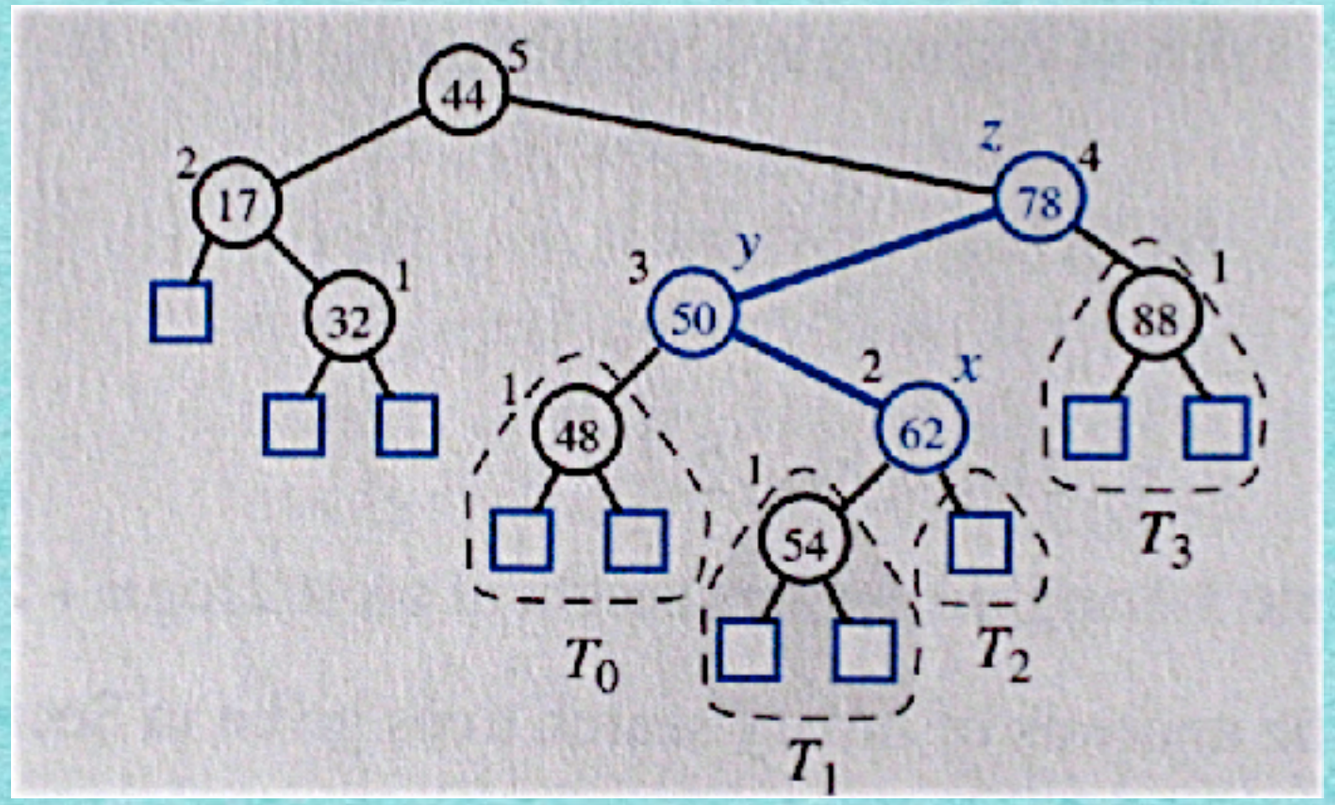


Einfügen



Einfügen

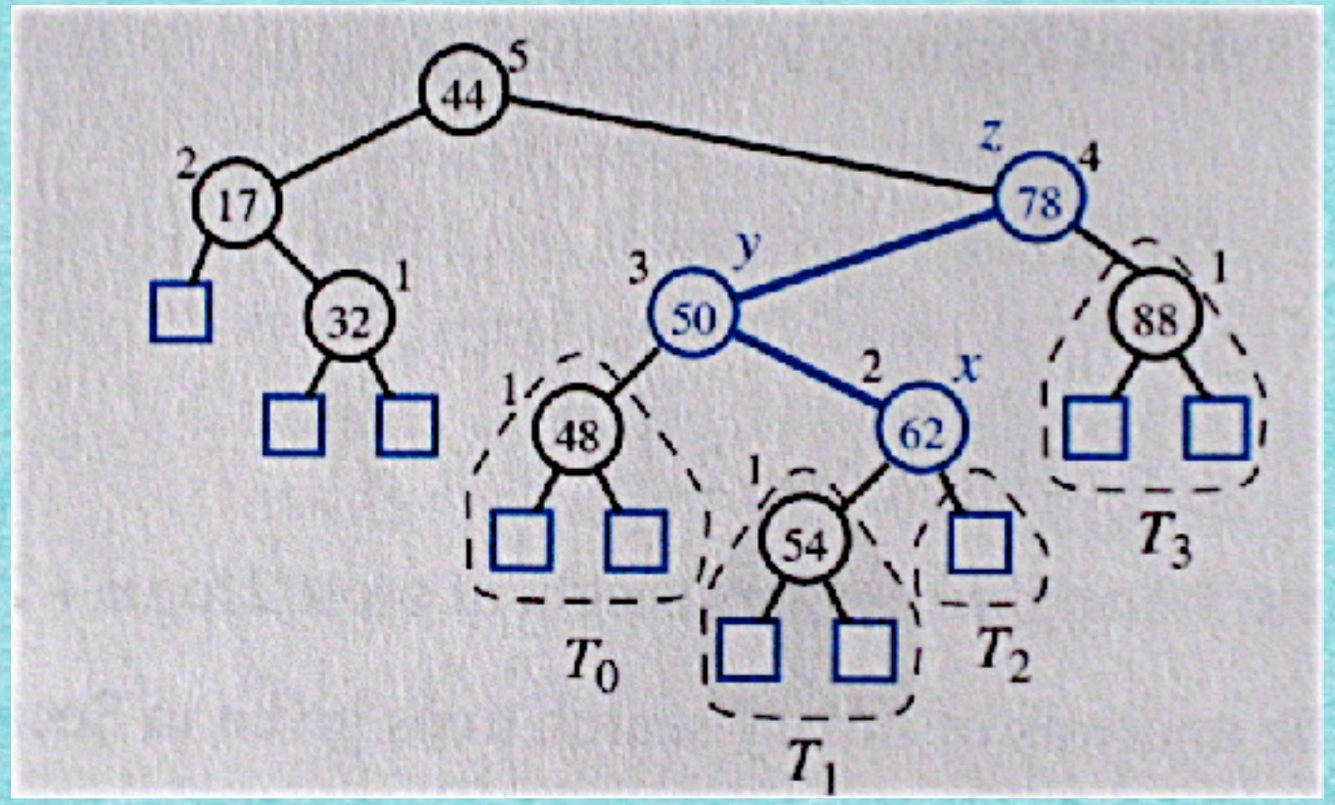
Idee:



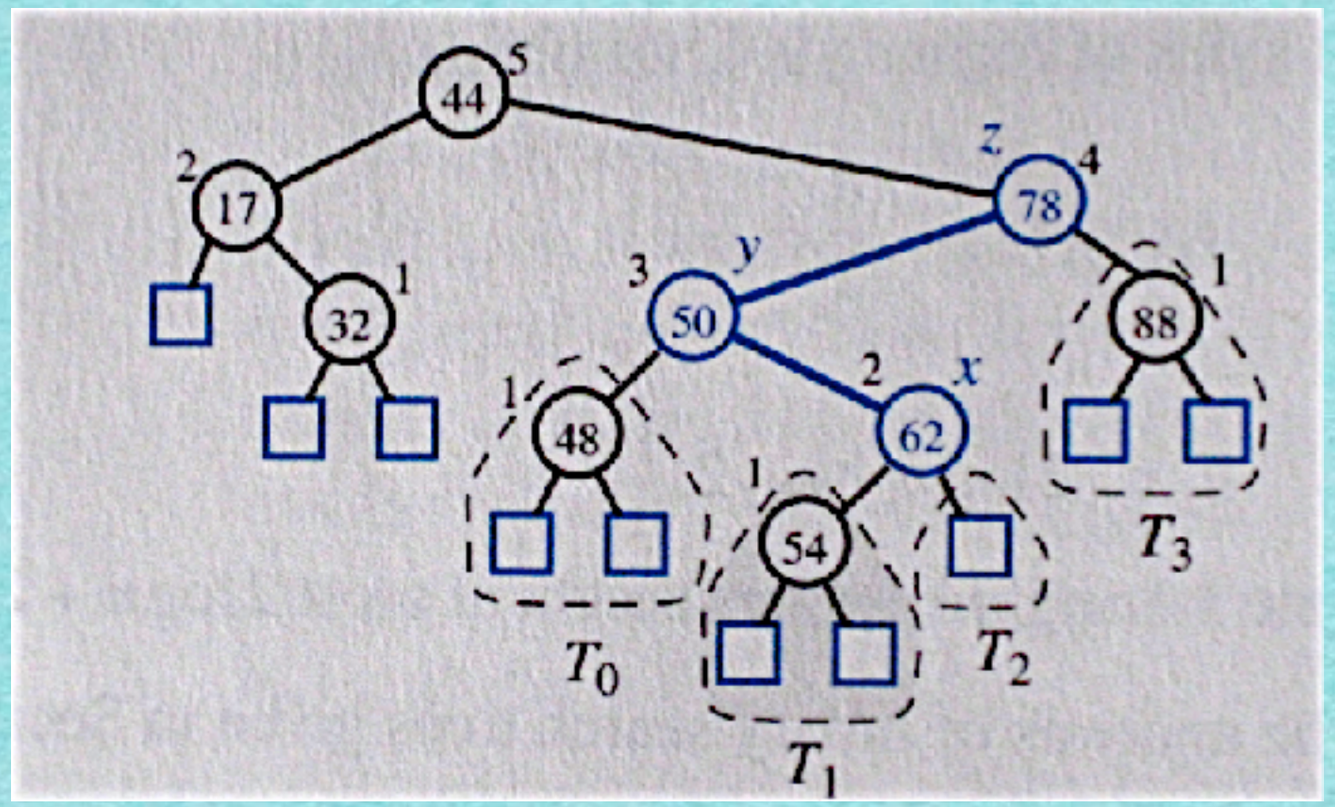
Einfügen

Idee:

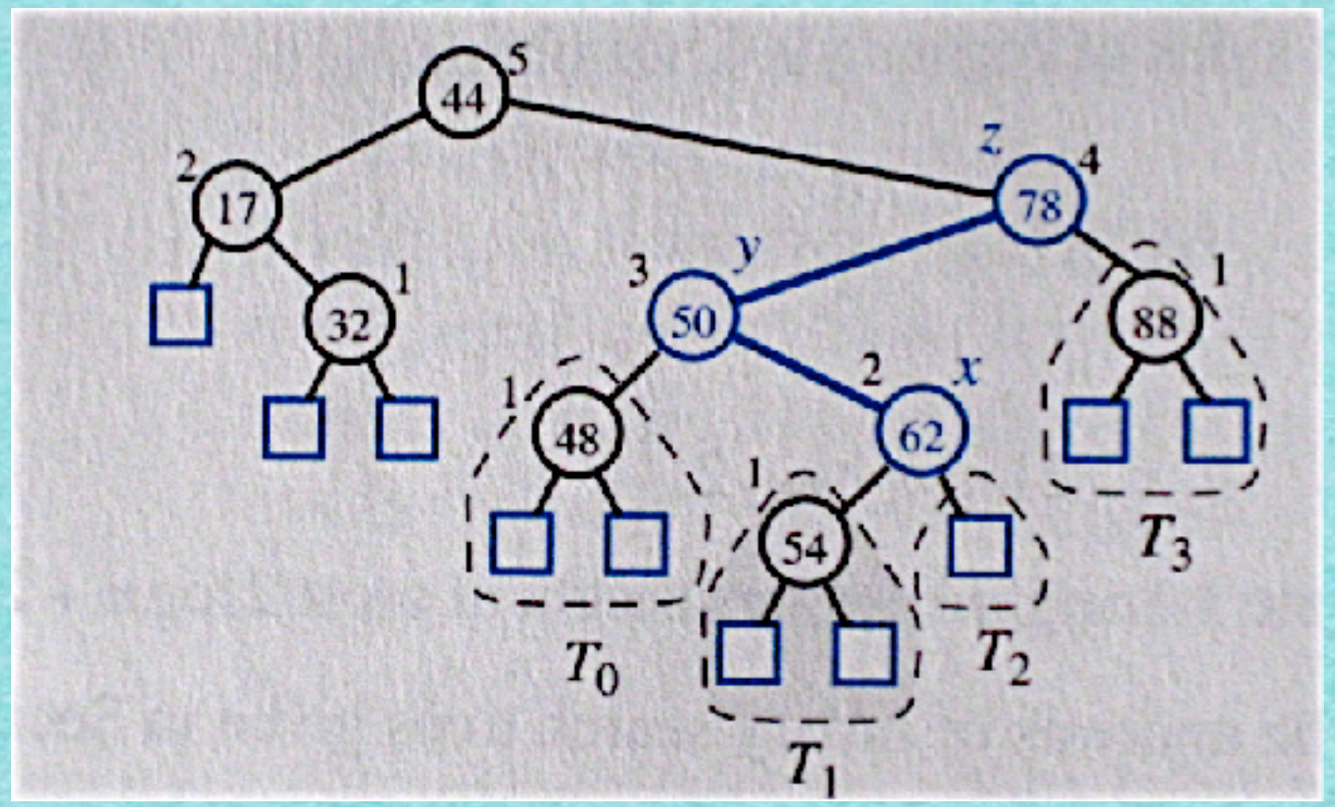
- ***Höhenbalanciertheit ändert sich beim Einfügen einzelner Elemente nur wenig - und lokal!***



Einfügen

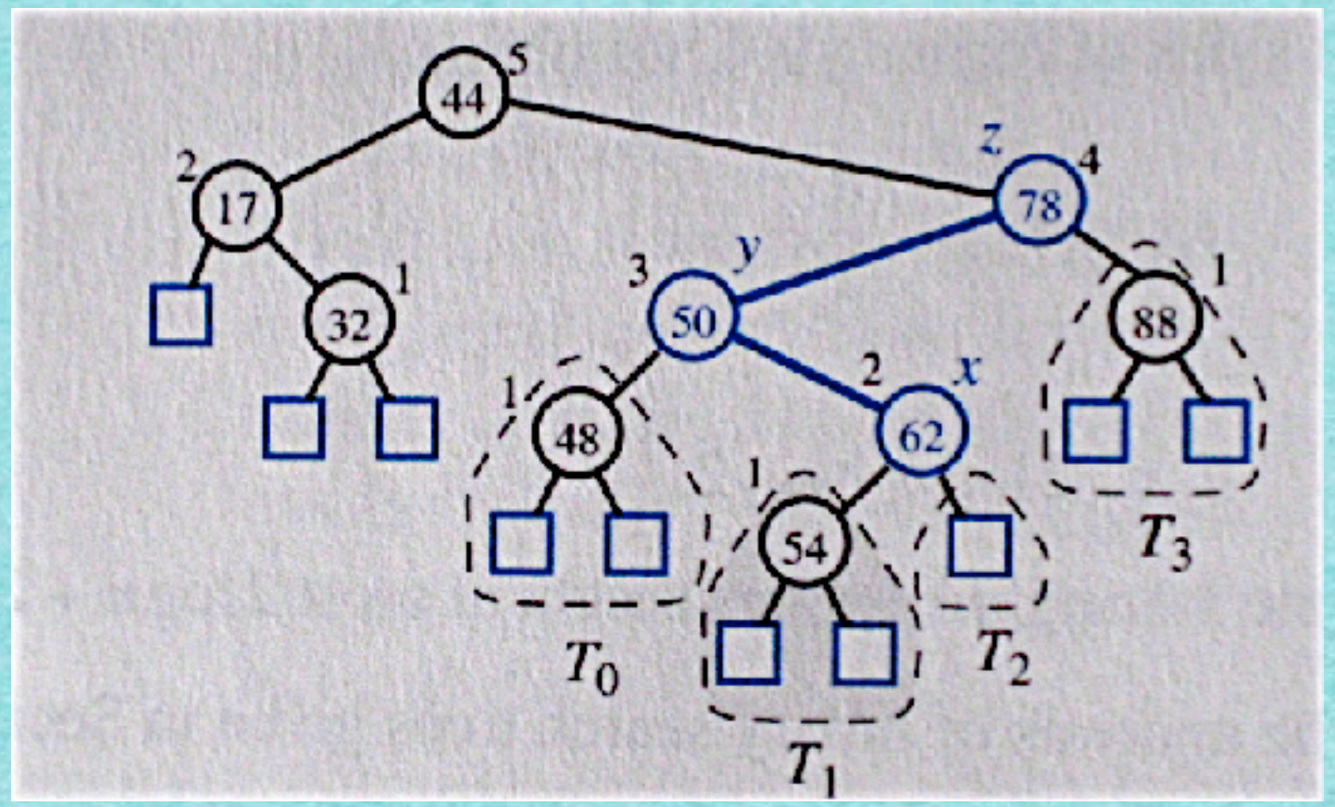


Einfügen



Was tun?

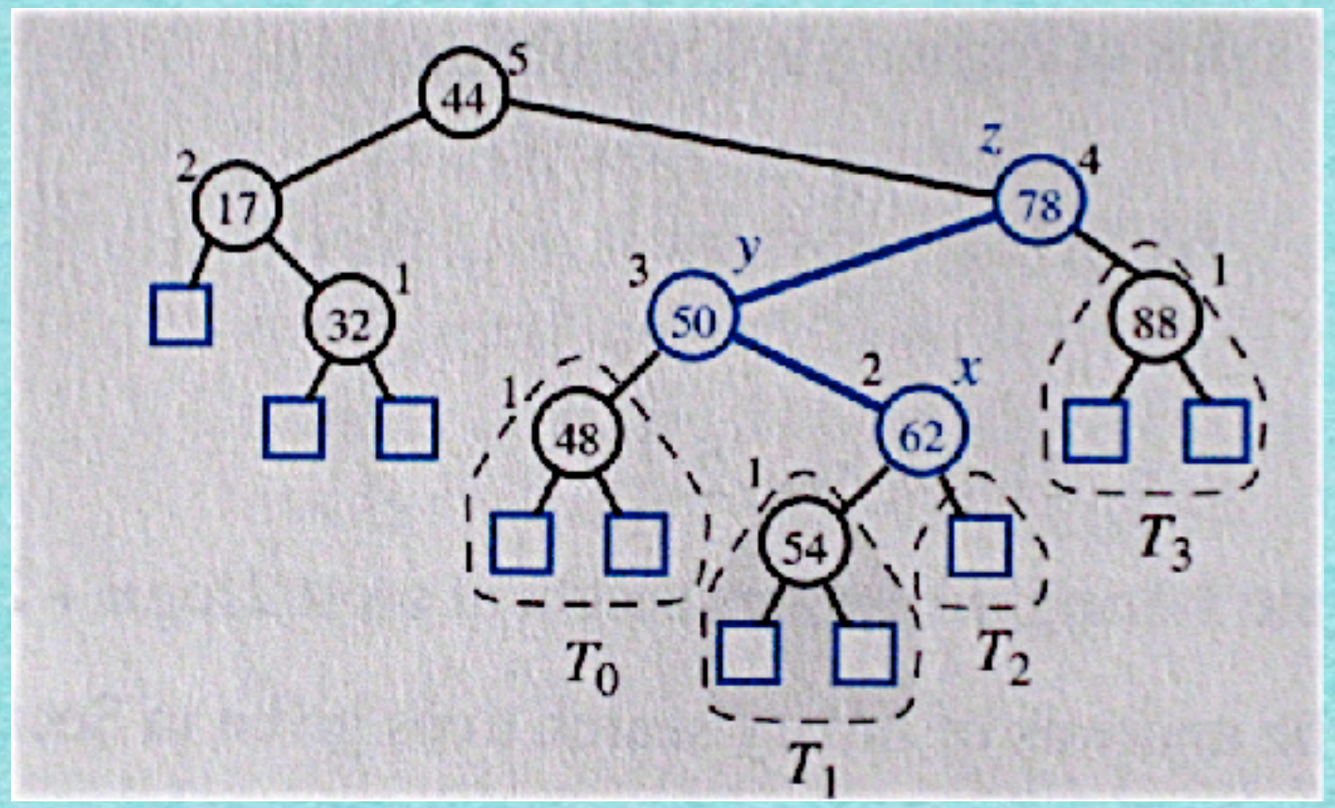
Einfügen



Was tun?

- ***Teilbaum der 78 ist nicht höhenbalanciert.***

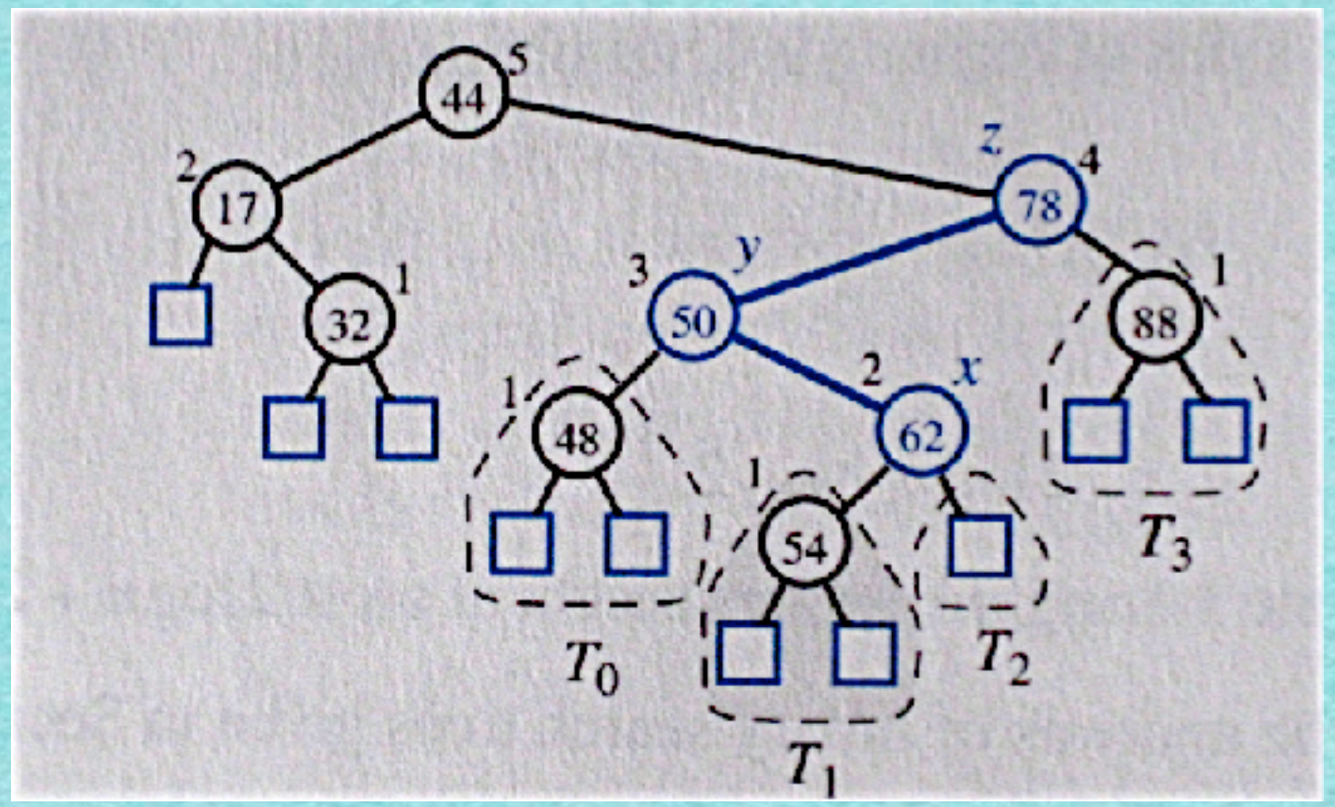
Einfügen



Was tun?

- ***Teilbaum der 78 ist nicht höhenbalanciert.***
- ***Die Höhe sollte höchstens 3 sein, damit auch der ganze Baum unter der 44 höhenbalanciert ist.***

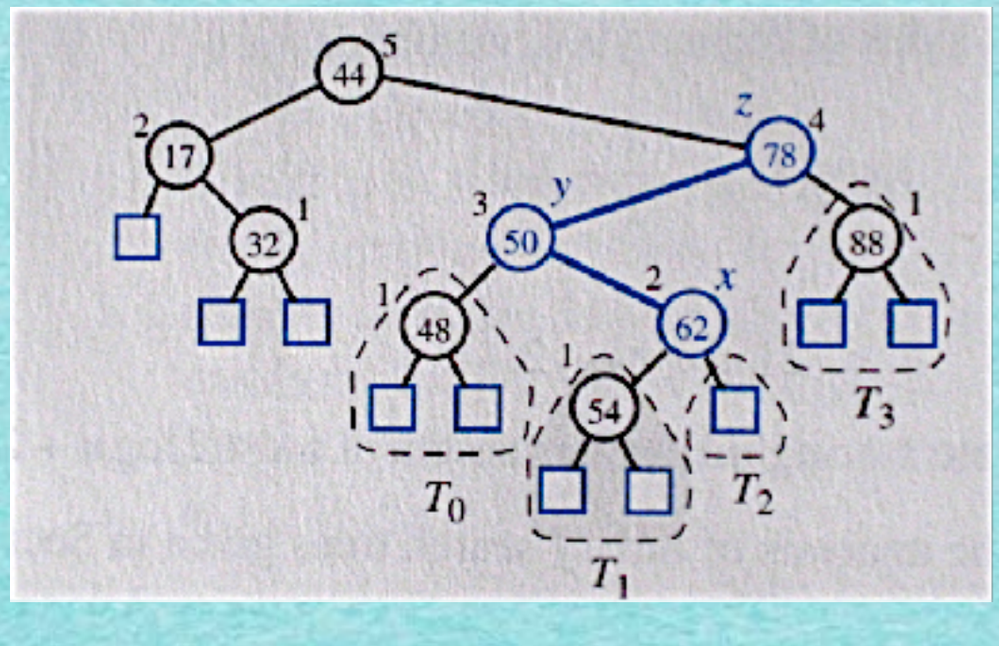
Einfügen



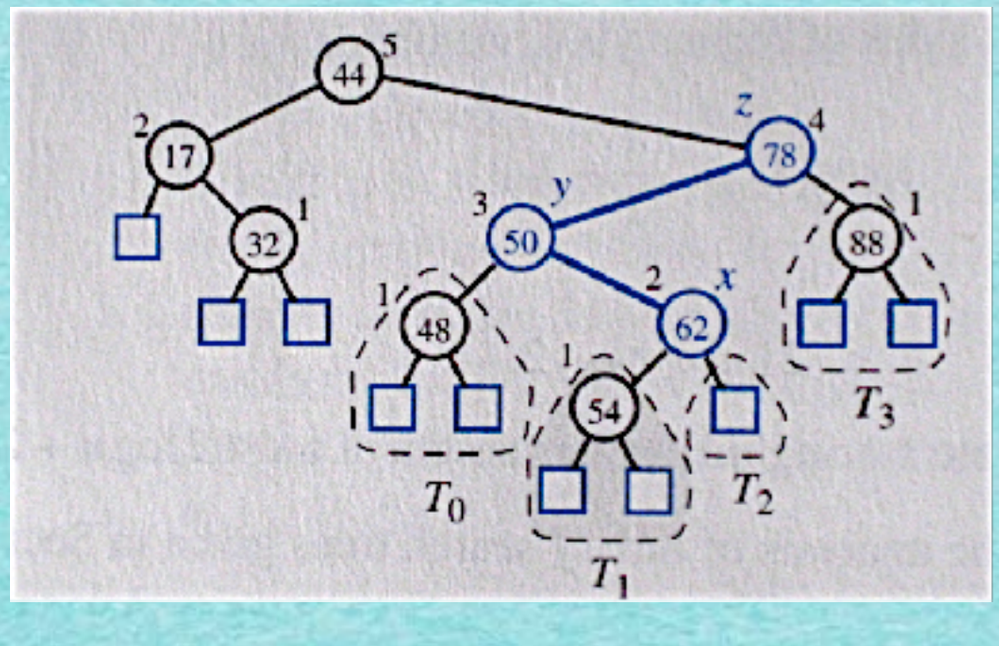
Was tun?

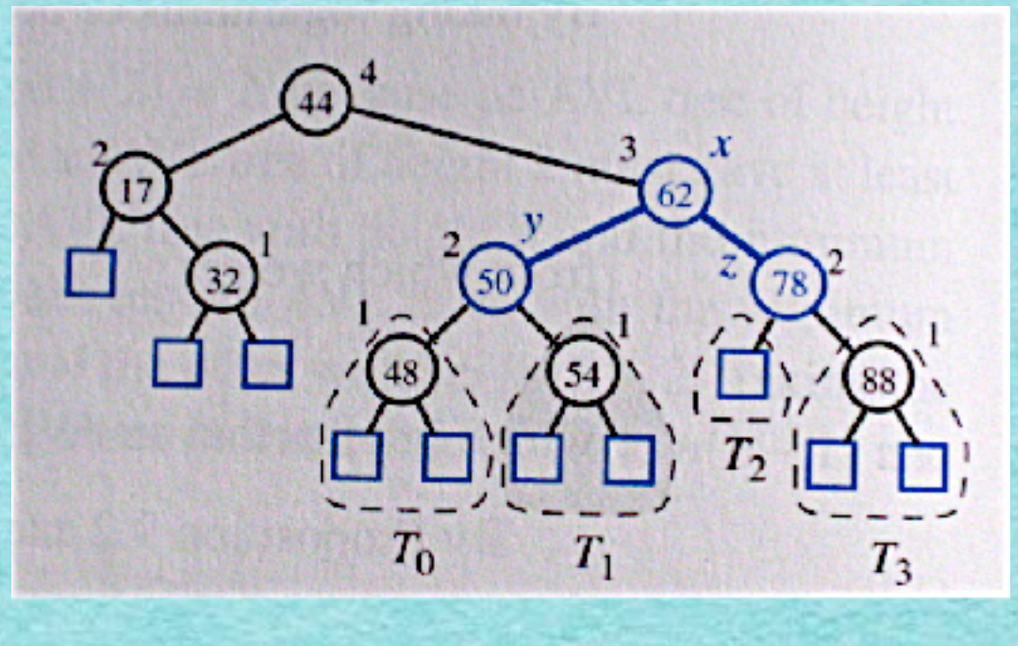
- ***Teilbaum der 78 ist nicht höhenbalanciert.***
- ***Die Höhe sollte höchstens 3 sein, damit auch der ganze Baum unter der 44 höhenbalanciert ist.***
- ***Betrachte Knoten 78, Kind 50, Enkel 62!***

Einfügen

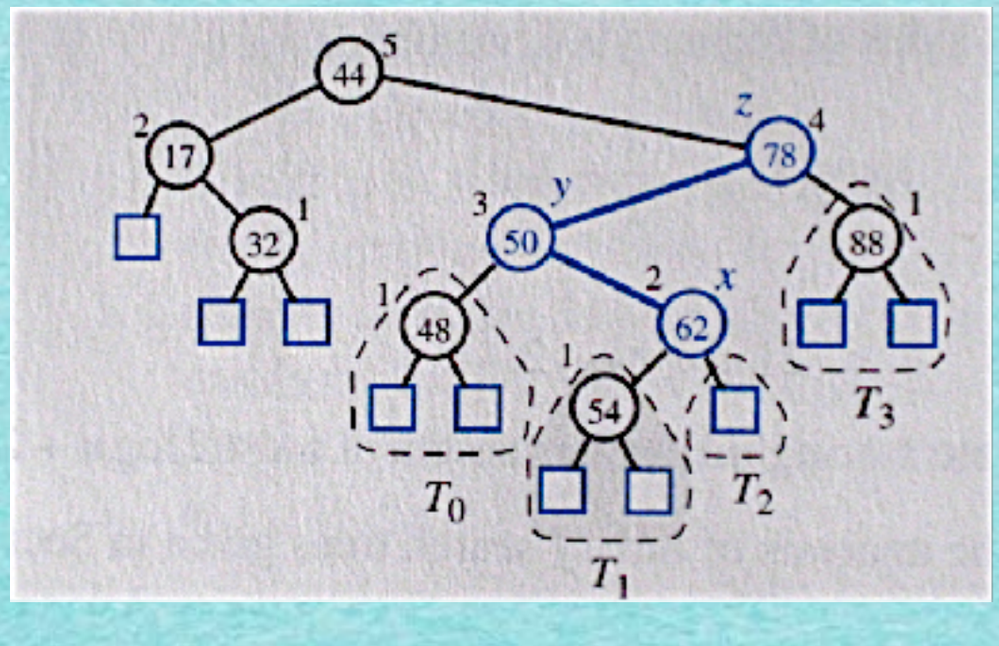


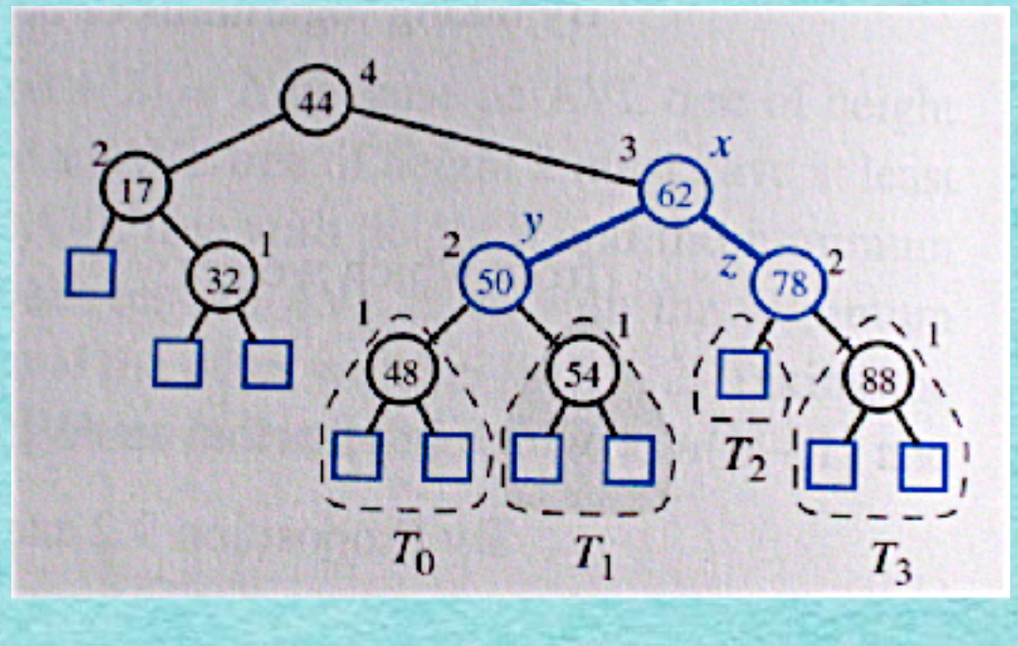
Einfügen


44
17
32
78
50
88
48
62
54
z
y
x
T_0
T_1
T_2
T_3


44
17
32
62
50
78
48
54
88
x
y
z
T_0
T_1
T_2
T_3

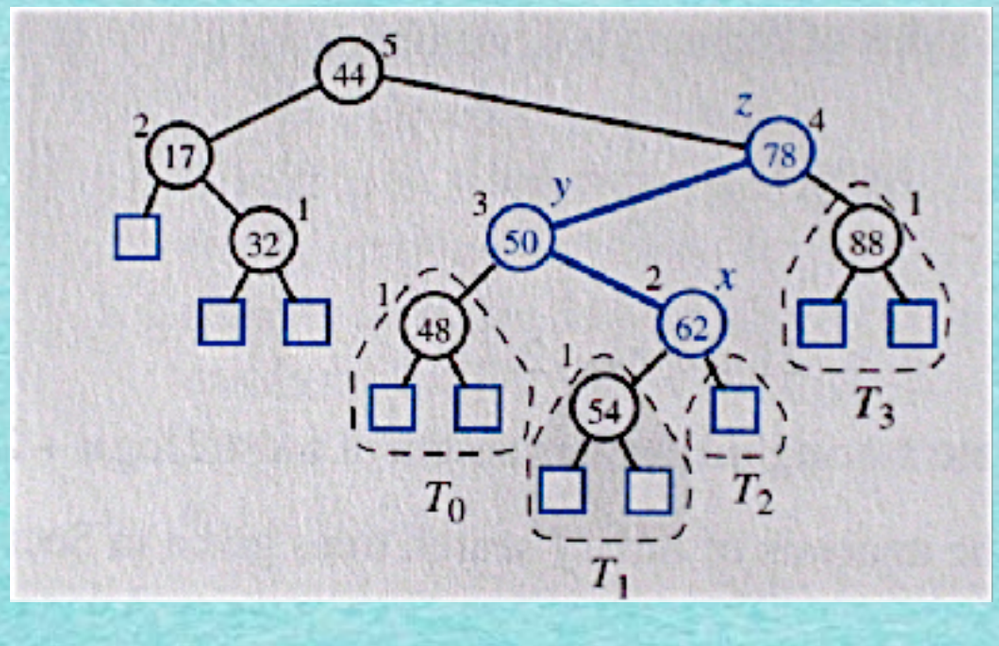
Einfügen

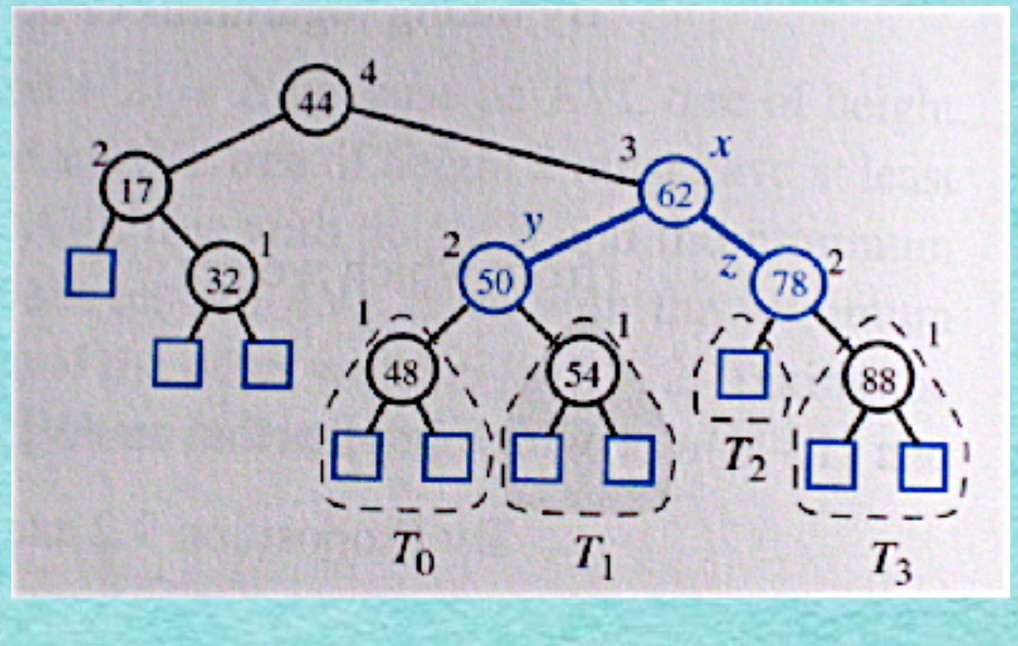




Neuer Baum!

Einfügen

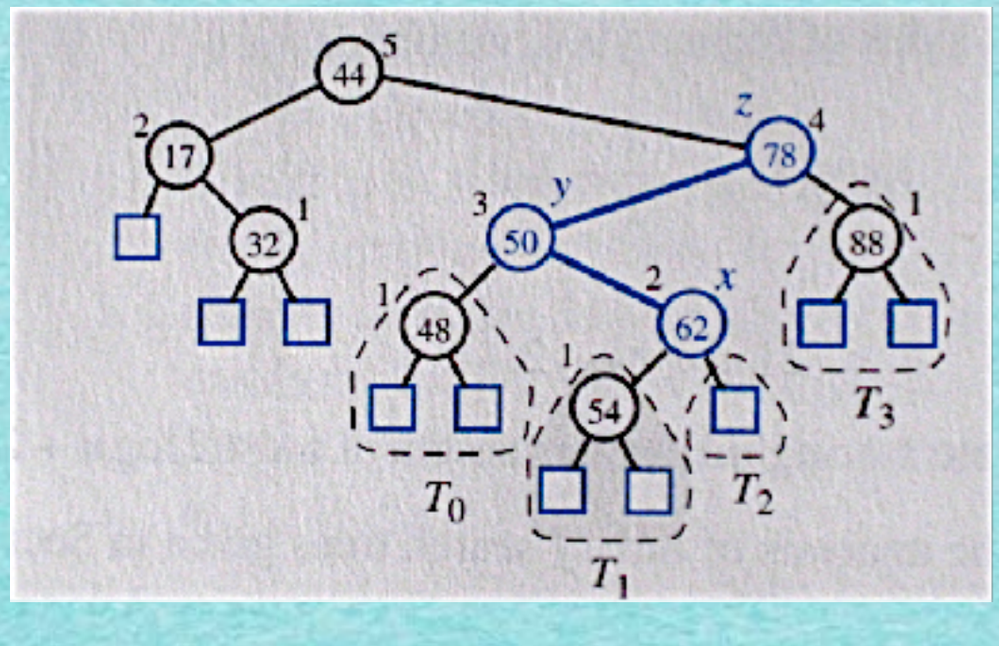


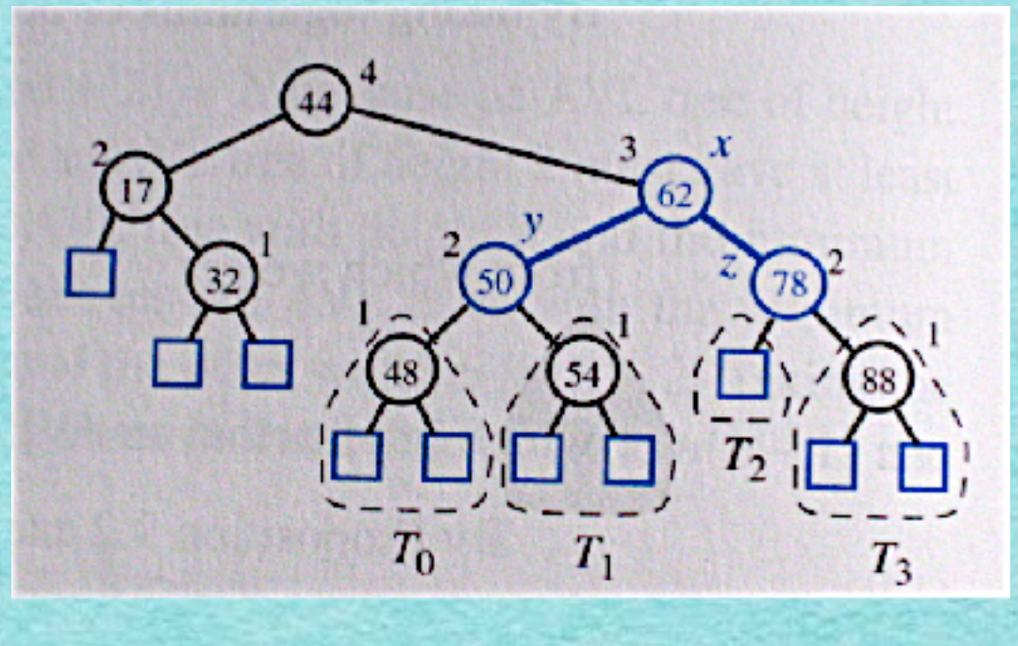


Neuer Baum!

- ***Höhenbalanciert***

Einfügen

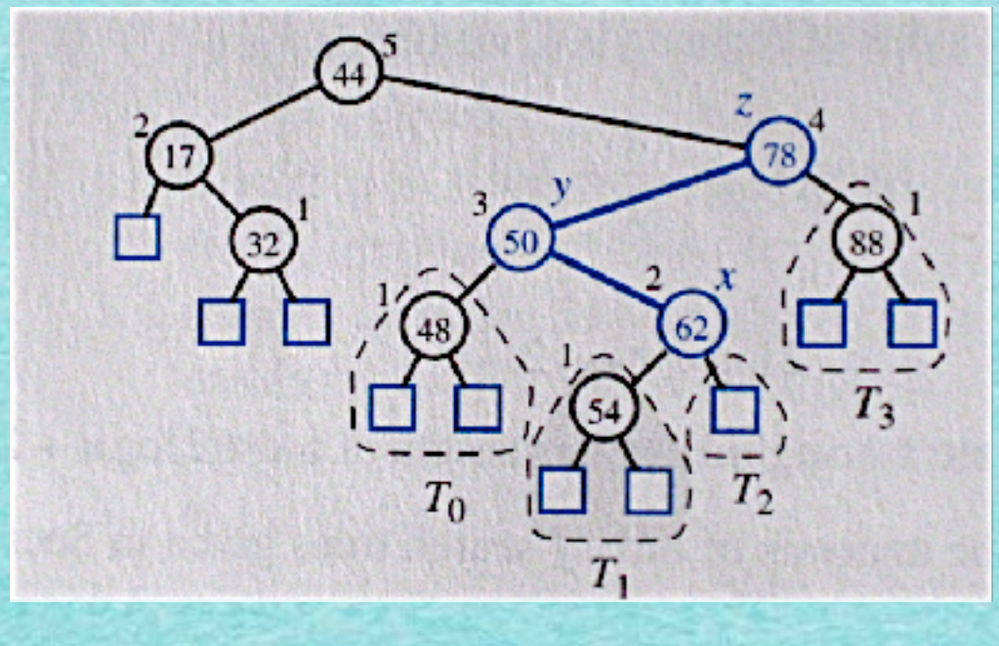


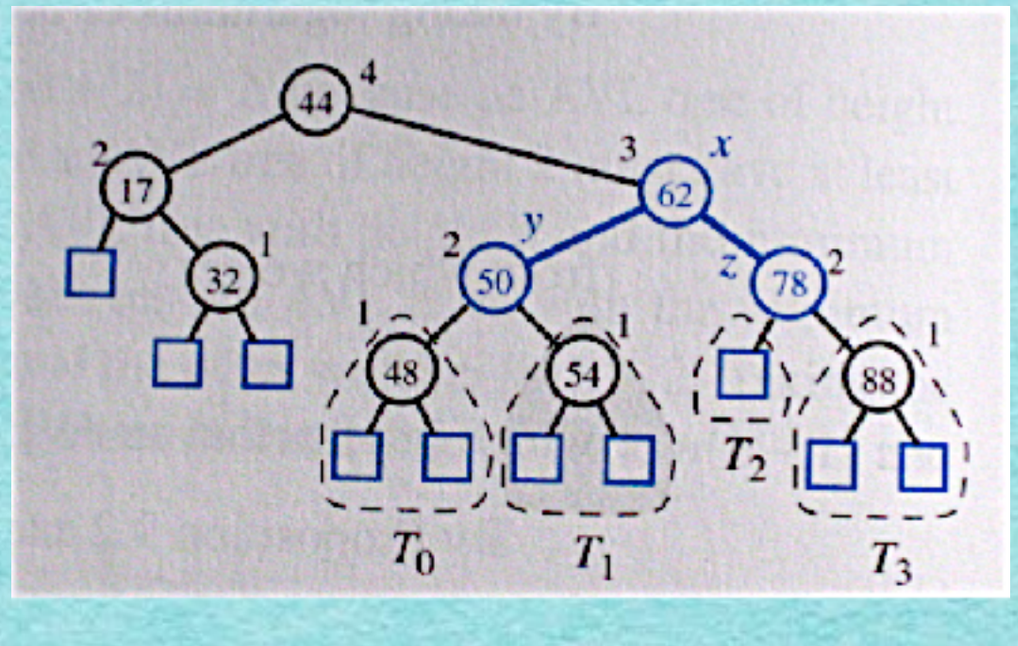


Neuer Baum!

- ***Höhenbalanciert***
- ***Nur lokale Umsetzung der Knoten 78, 50, 62***

Einfügen

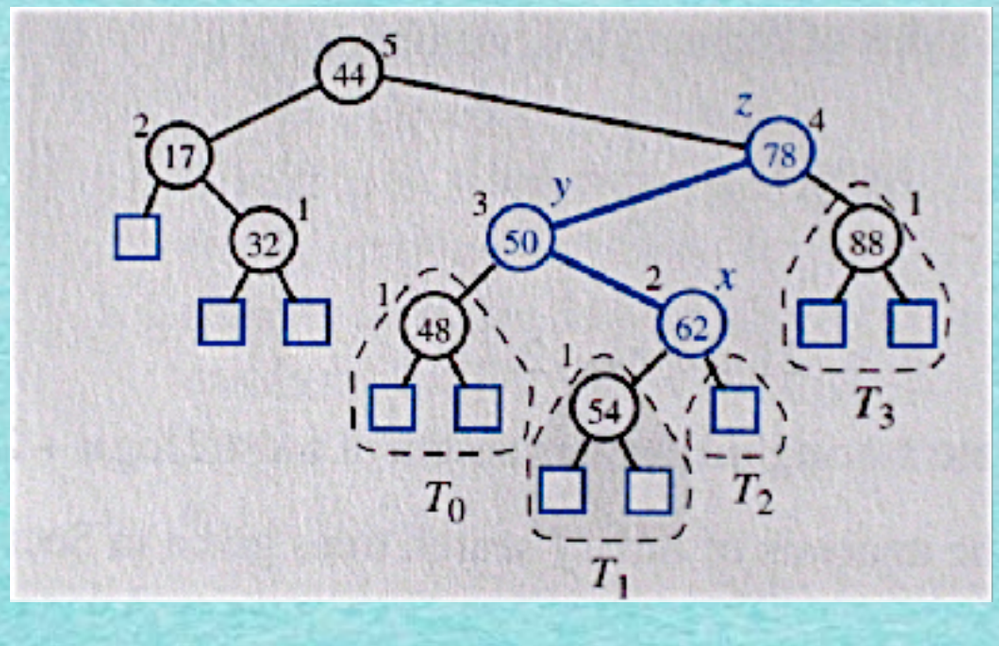


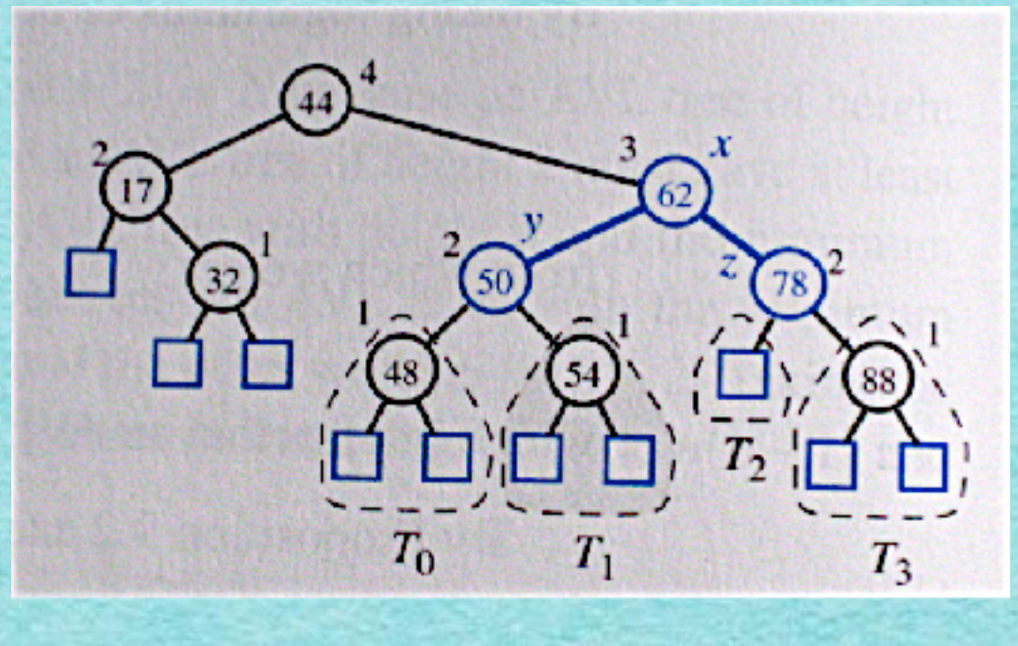


Neuer Baum!

- *Höhenbalanciert*
- *Nur lokale Umsetzung der Knoten 78, 50, 62*
- *Vorher drei Knoten untereinander, jetzt der mittlere über zwei anderen.*

Einfügen

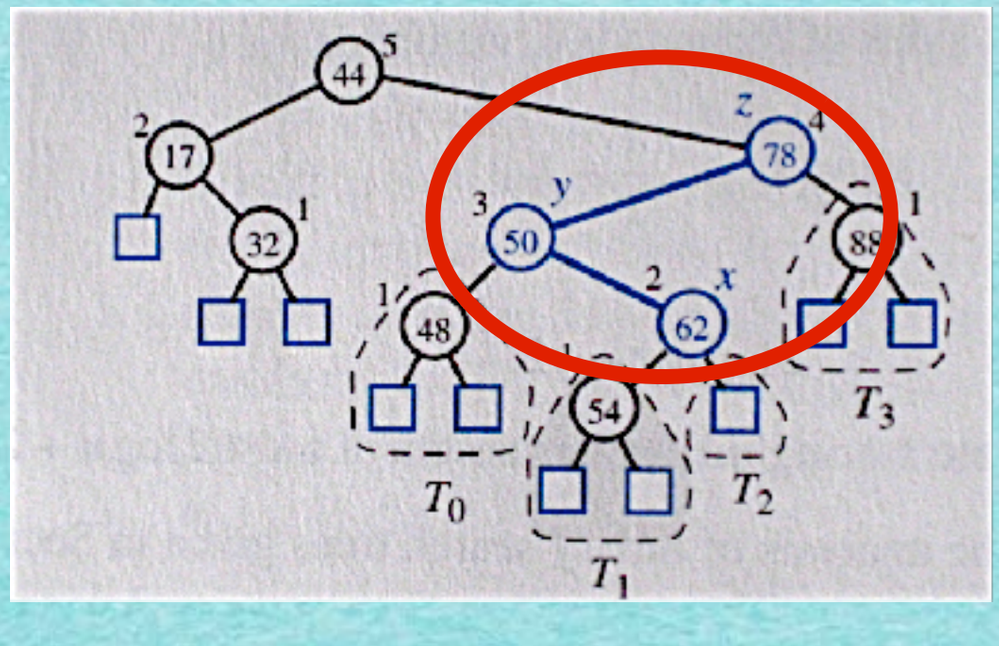


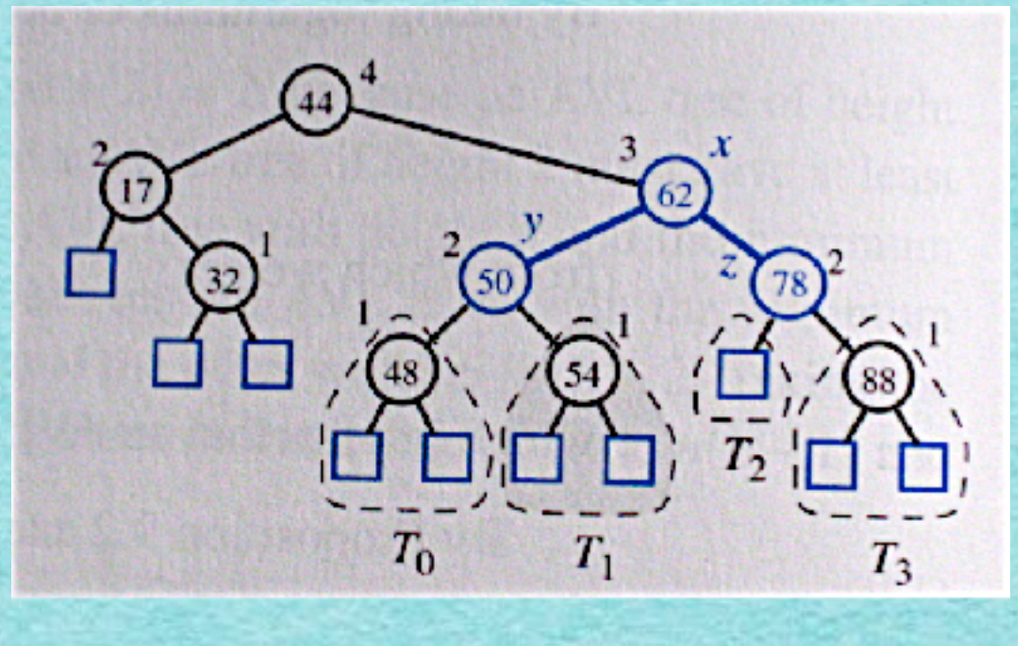


Neuer Baum!

- *Höhenbalanciert*
- *Nur lokale Umsetzung der Knoten 78, 50, 62*
- *Vorher drei Knoten untereinander, jetzt der mittlere über zwei anderen.*
- *„Rotation“*

Einfügen

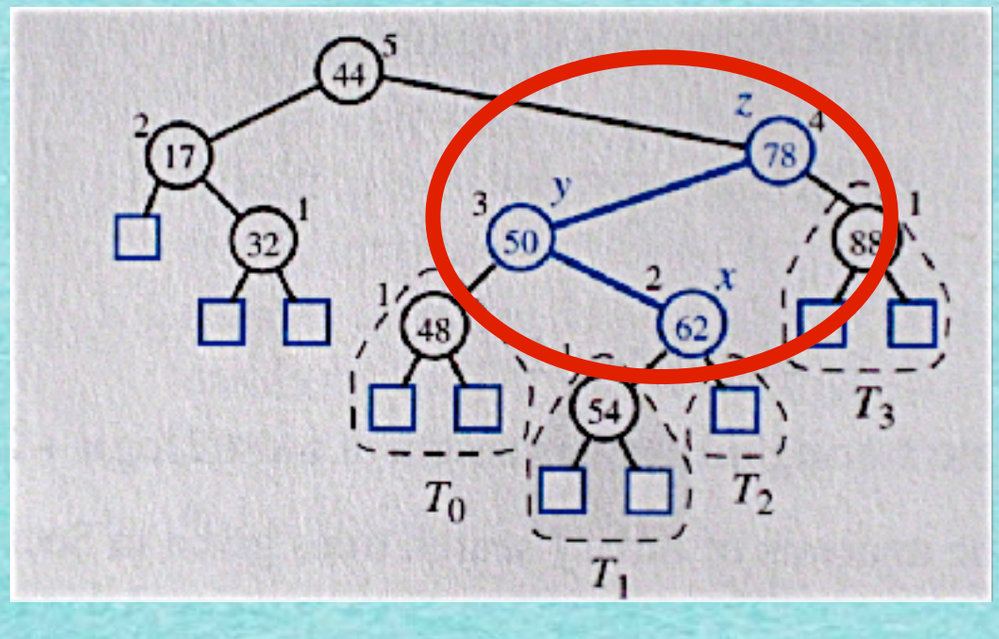


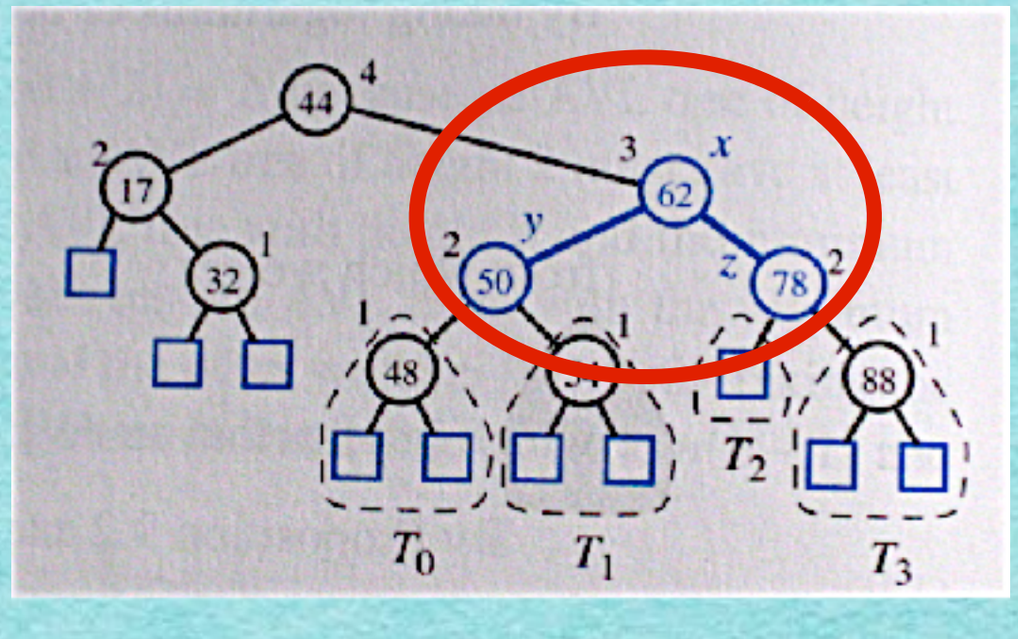


Neuer Baum!

- *Höhenbalanciert*
- *Nur lokale Umsetzung der Knoten 78, 50, 62*
- *Vorher drei Knoten untereinander, jetzt der mittlere über zwei anderen.*
- *„Rotation“*

Einfügen





Neuer Baum!

- *Höhenbalanciert*
- *Nur lokale Umsetzung der Knoten 78, 50, 62*
- *Vorher drei Knoten untereinander, jetzt der mittlere über zwei anderen.*
- *„Rotation“*

Algorithmus 4.9

INPUT:

OUTPUT:

Algorithmus 4.9

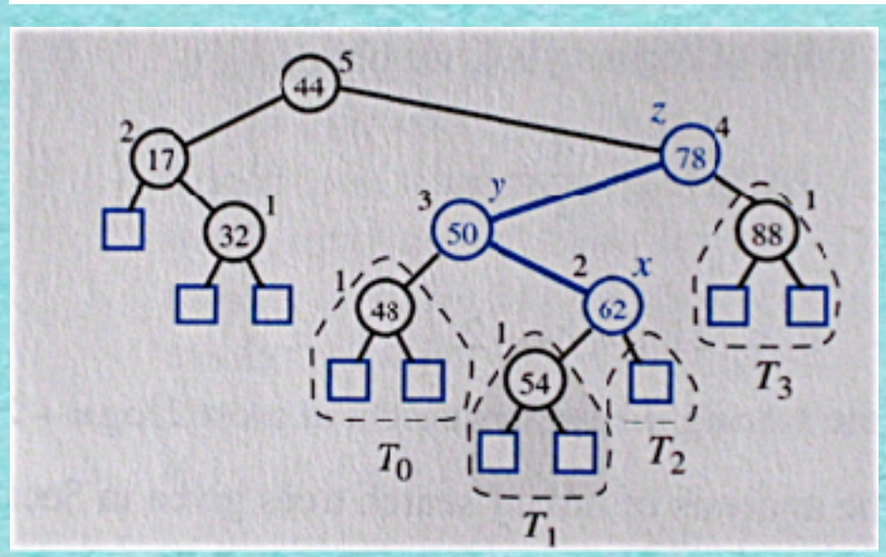
INPUT: Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z

OUTPUT:

Algorithmus 4.9

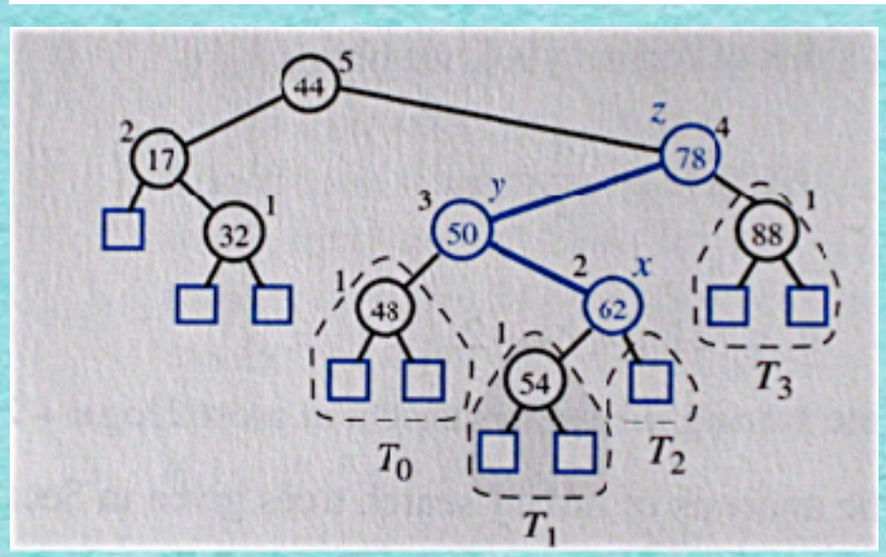
INPUT: Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z

OUTPUT:



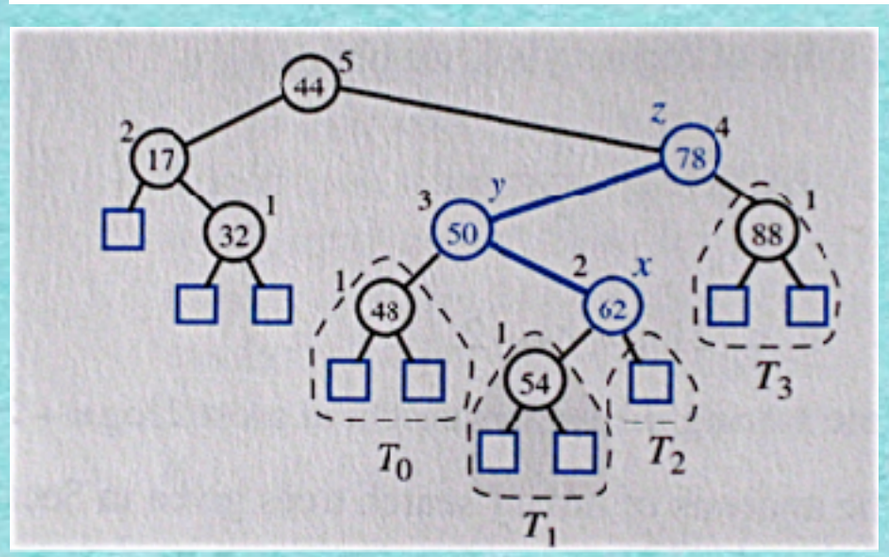
Algorithmus 4.9

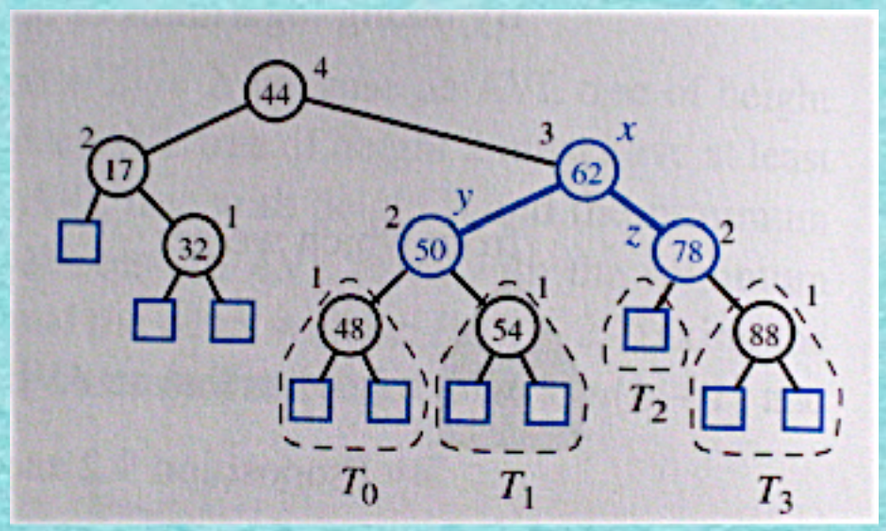
INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z



Algorithmus 4.9

INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

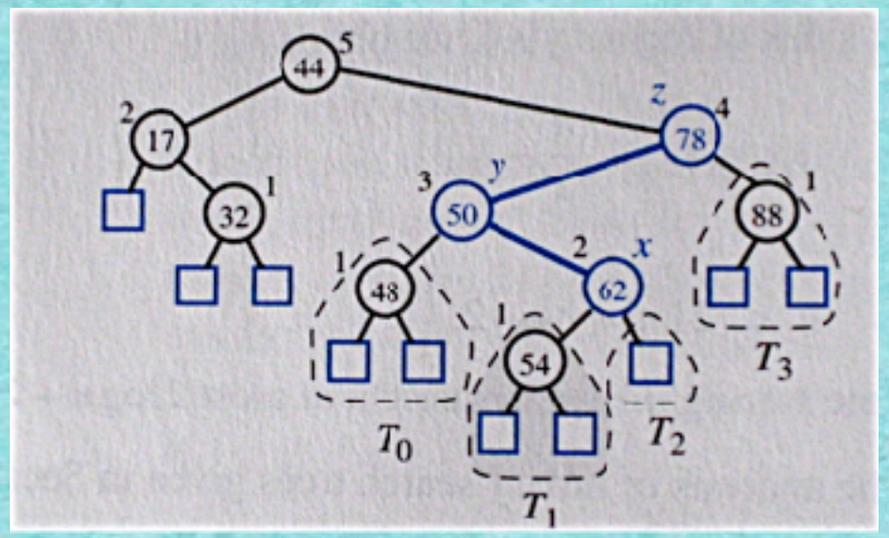


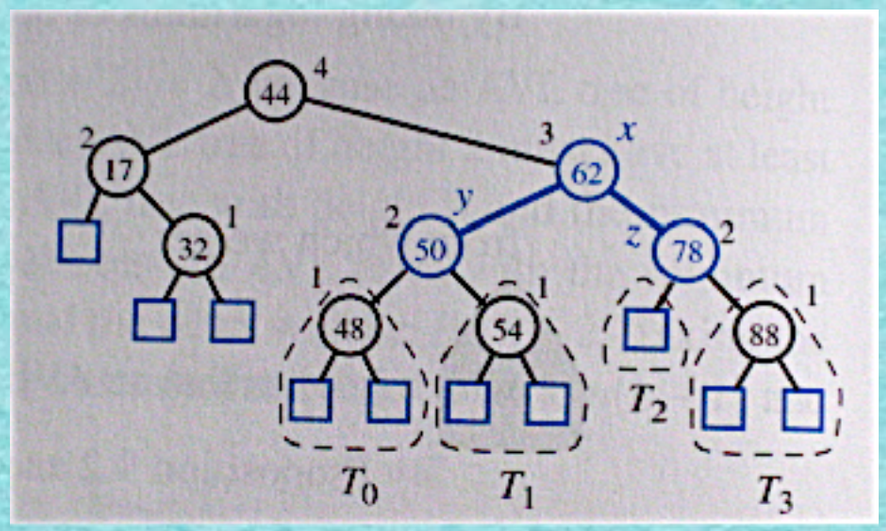


Algorithmus 4.9

INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

RESTRUCTURE(x)



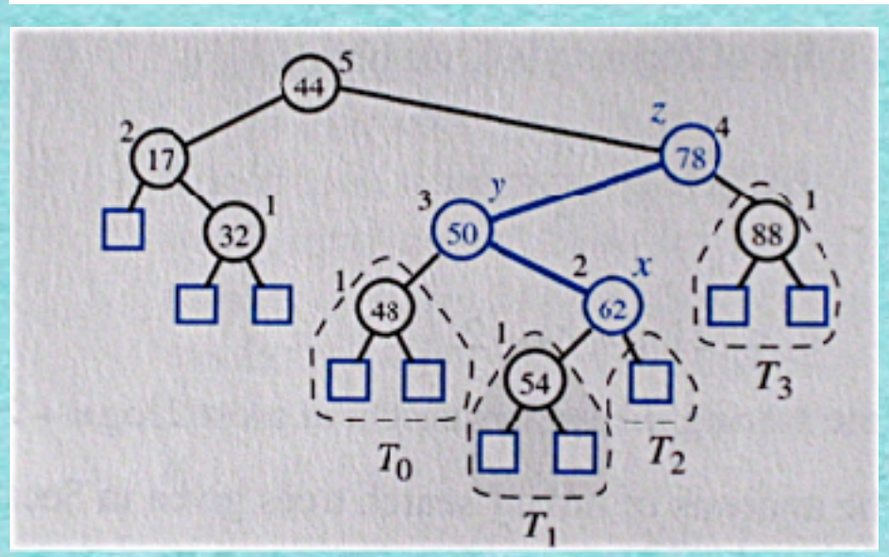


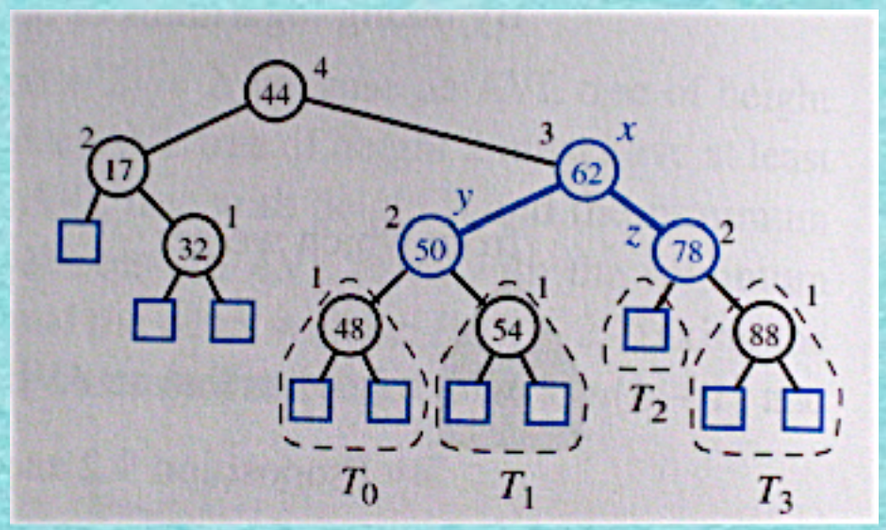
Algorithmus 4.9

INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;



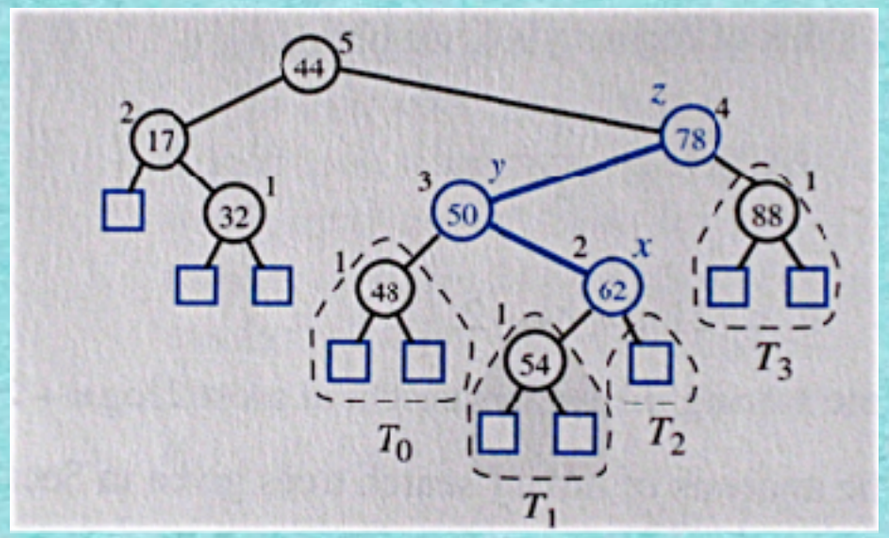


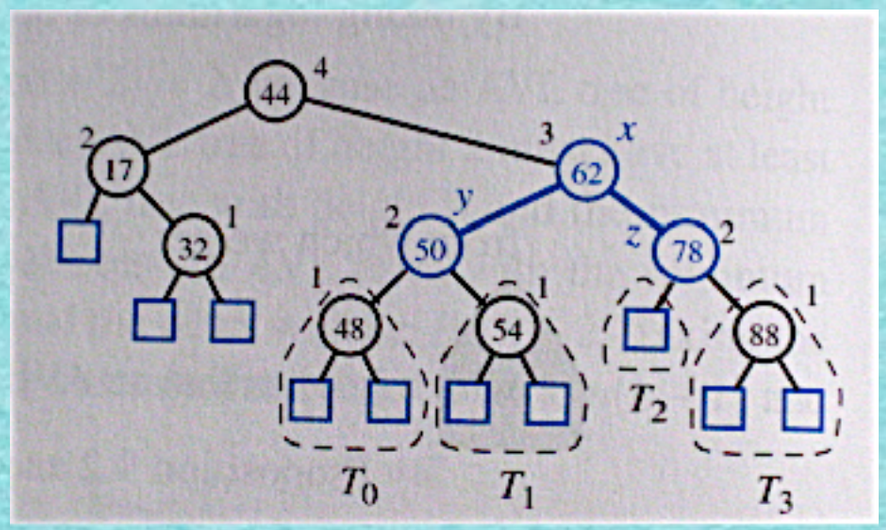
Algorithmus 4.9

INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;
   seien (T_0 , T_1 , T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben





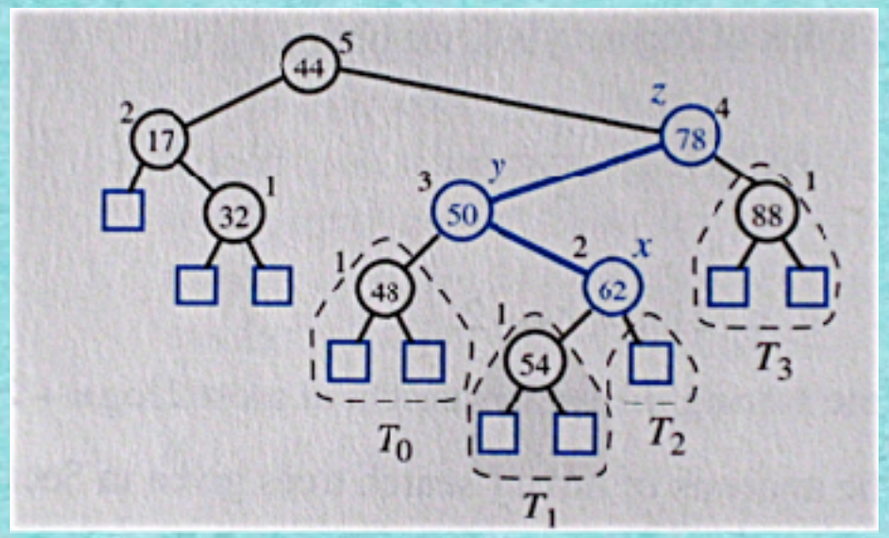
Algorithmus 4.9

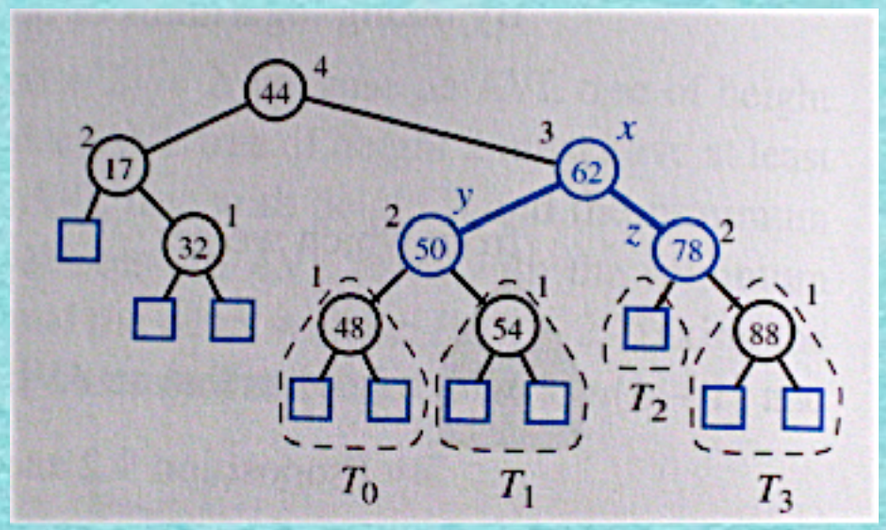
INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;
   seien (T_0 , T_1 , T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben

2. Ersetze den Teilbaum mit Wurzel z durch einen neuen Teilbaum mit Wurzel b.





Algorithmus 4.9

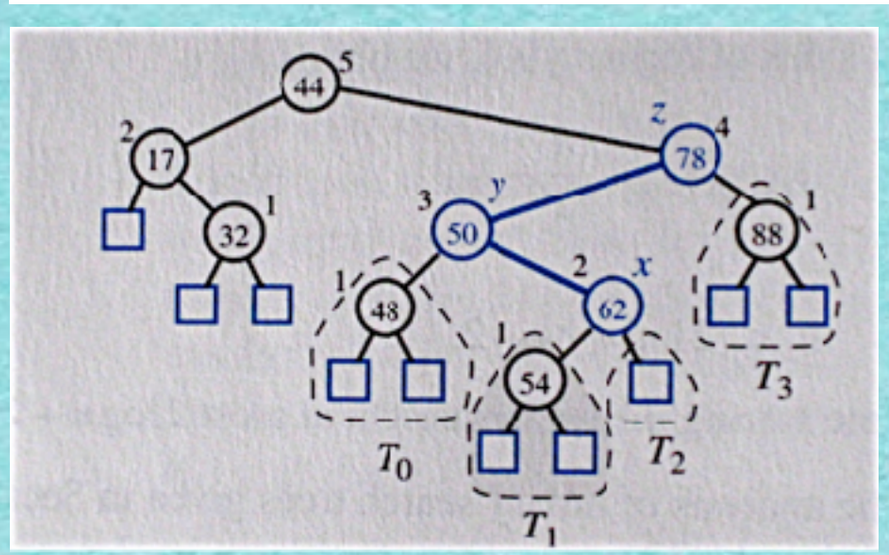
INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

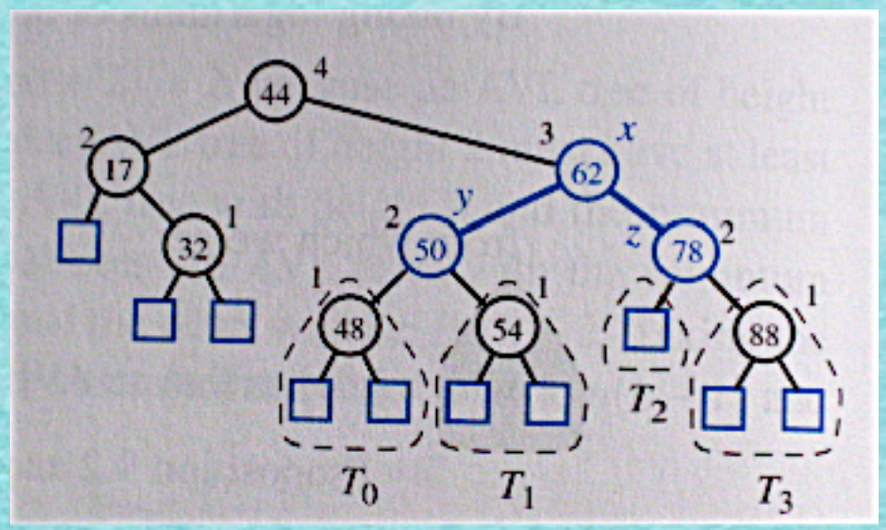
RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;
   seien (T_0 , T_1 , T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben

2. Ersetze den Teilbaum mit Wurzel z durch einen neuen Teilbaum mit Wurzel b.

3. Setze a als linkes Kind von b, mit T_0 und T_1 als linken und rechten Teilbaum unter a;





Algorithmus 4.9

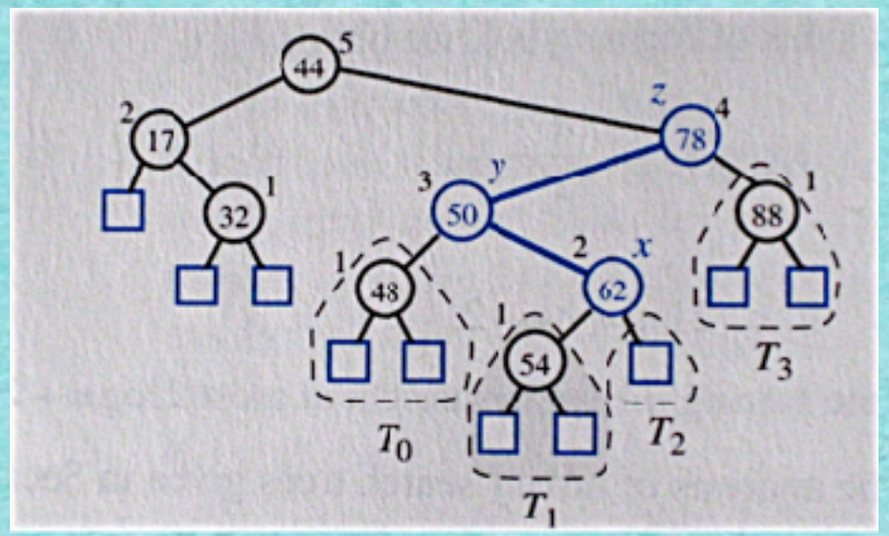
INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

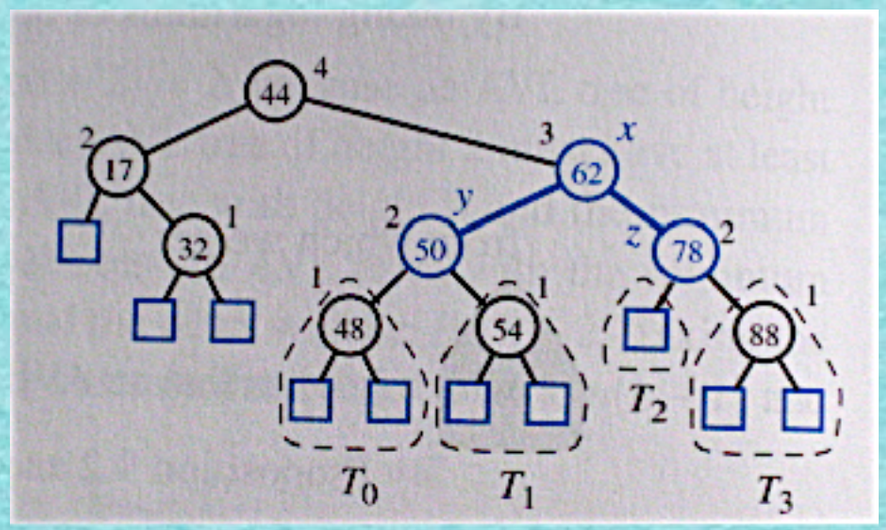
RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;
   seien (T_0 , T_1 , T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben

2. Ersetze den Teilbaum mit Wurzel z durch einen neuen Teilbaum mit Wurzel b.

3. Setze a als linkes Kind von b, mit T_0 und T_1 als linken und rechten Teilbaum unter a;
   setze c als rechtes Kind von b, mit T_2 und T_3 als linken und rechten Teilbaum unter c.





Algorithmus 4.9

INPUT:	Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z
OUTPUT:	Binärer Suchbaum T nach Umstrukturierung mit x, y, z

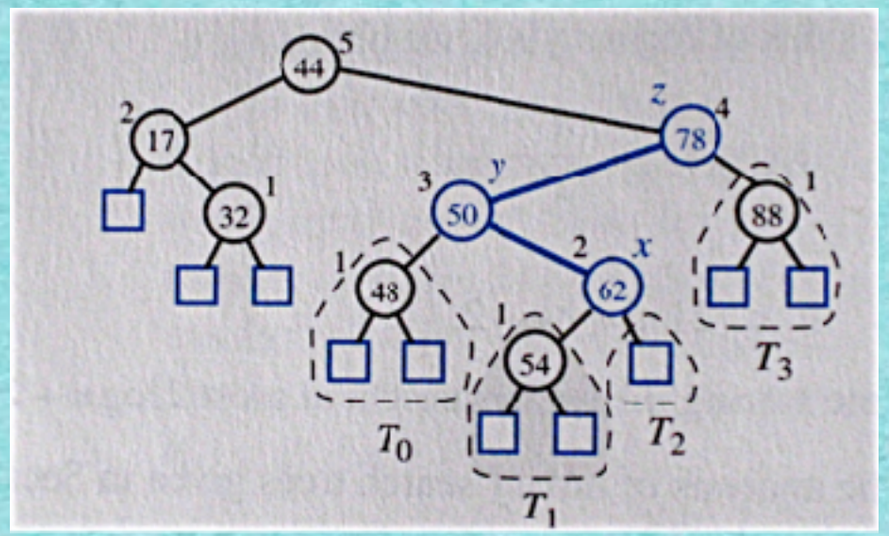
RESTRUCTURE(x)

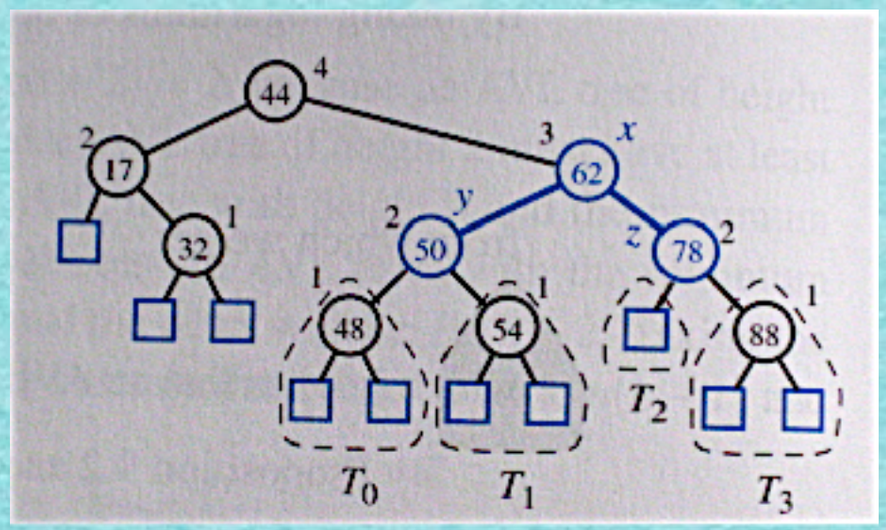
1. Sei (a, b, c) die Größensortierung der Knoten x, y, z;
   seien (T_0 , T_1 , T_2, T_3) die Größensortierung der vier Teilbäume unter x, y, z, die nicht Wurzeln x, y, z haben

2. Ersetze den Teilbaum mit Wurzel z durch einen neuen Teilbaum mit Wurzel b.

3. Setze a als linkes Kind von b, mit T_0 und T_1 als linken und rechten Teilbaum unter a;
   setze c als rechtes Kind von b, mit T_2 und T_3 als linken und rechten Teilbaum unter c.

4. RETURN





Einfügen

Satz 4.10

Satz 4.10

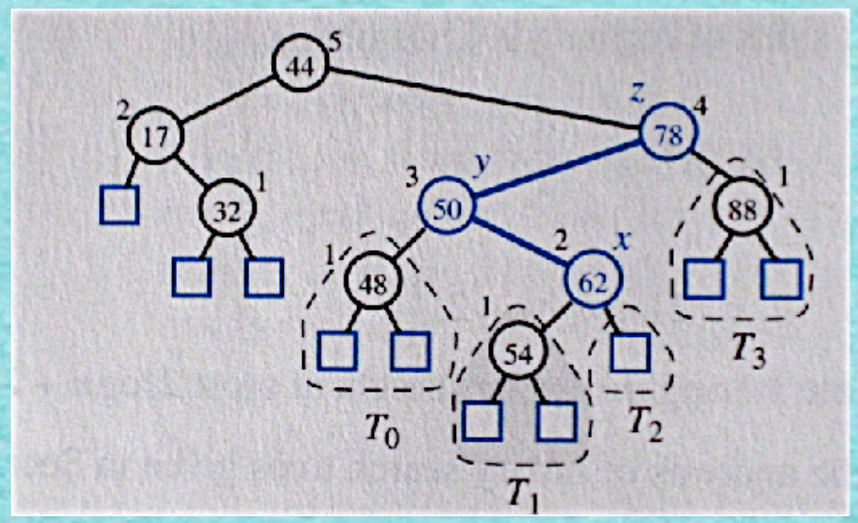
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.

Satz 4.10

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Satz 4.10

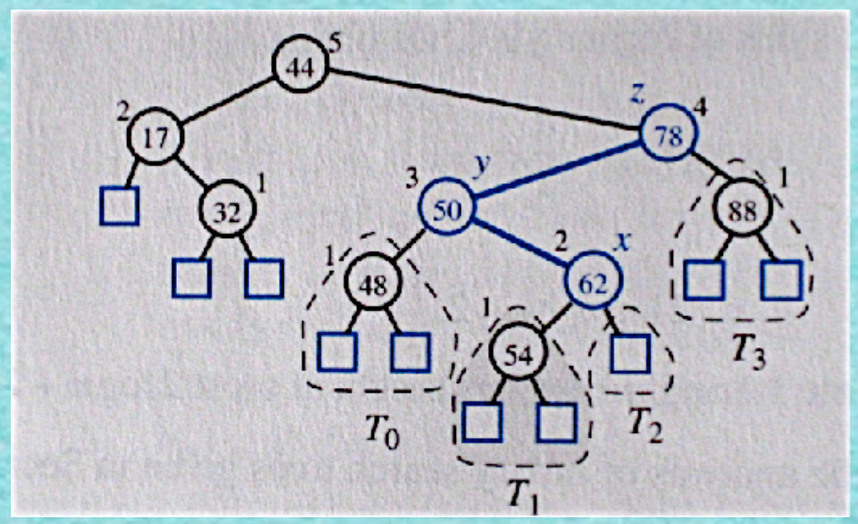
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).



Satz 4.10

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Beweis:

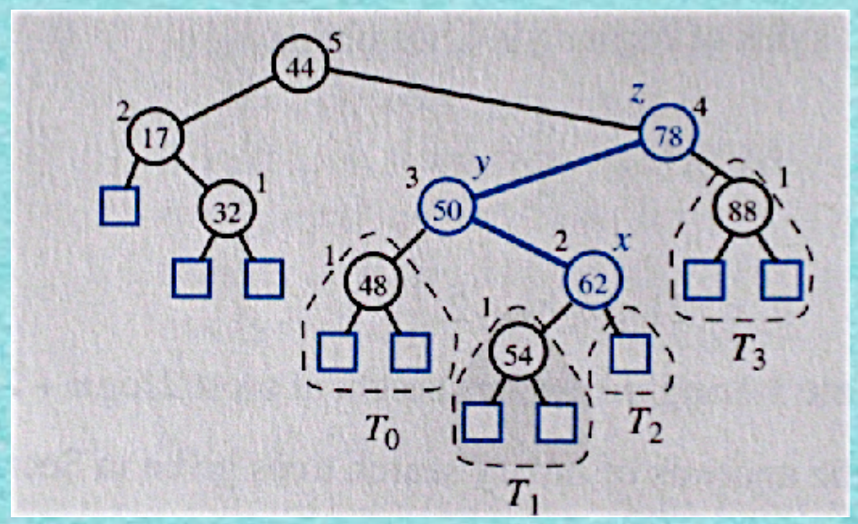


Satz 4.10

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Beweis:

Angenommen, durch Hinzufügen eines Knotens v ist der Baum unbalanciert geworden.

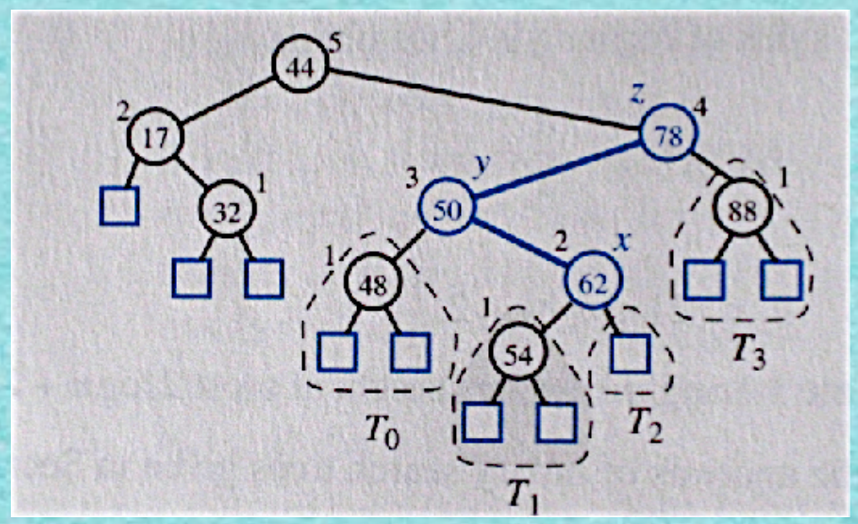


Satz 4.10

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Beweis:

Angenommen, durch Hinzufügen eines Knotens v ist der Baum unbalanciert geworden.
Sei z der nach dem Einfügen niedrigste unbalancierte Vorfahre von v.

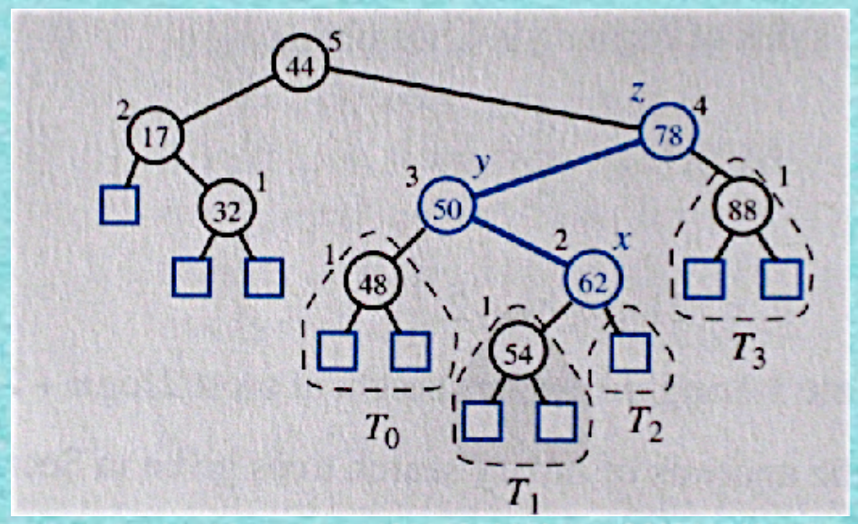


Satz 4.10

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Beweis:

Angenommen, durch Hinzufügen eines Knotens v ist der Baum unbalanciert geworden.
Sei z der nach dem Einfügen niedrigste unbalancierte Vorfahre von v.
Sei y das Kind von z, das Vorfahre von v ist; y muss zwei höher sein als das andere Kind von z.

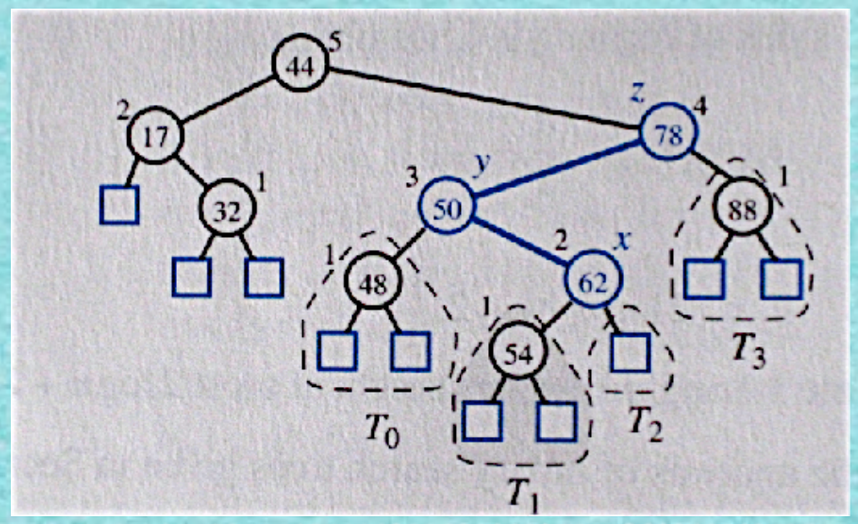


Satz 4.10

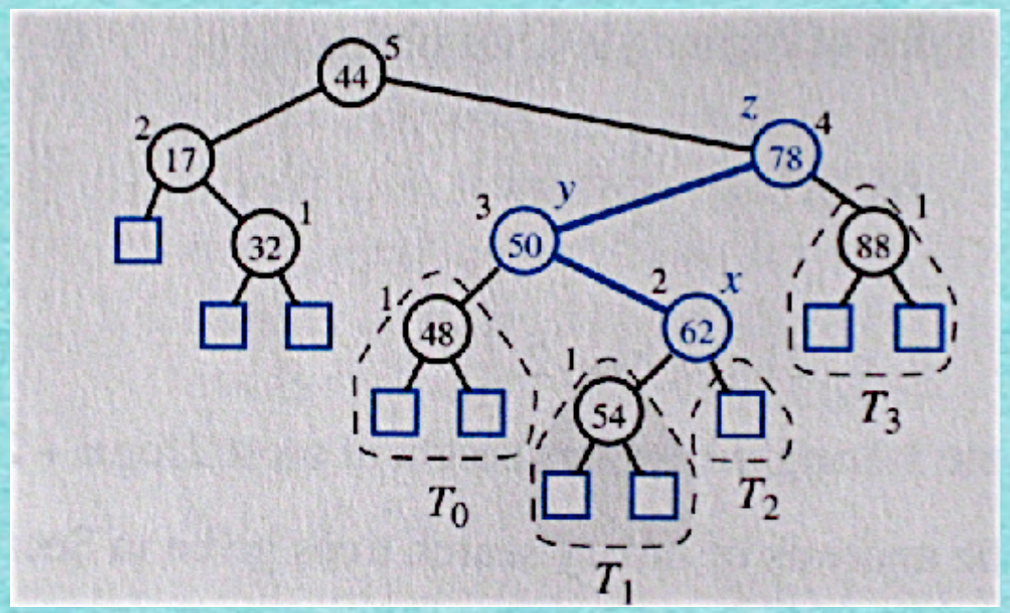
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten.
Die Zeit dafür ist O(1).

Beweis:

Angenommen, durch Hinzufügen eines Knotens v ist der Baum unbalanciert geworden.
Sei z der nach dem Einfügen niedrigste unbalancierte Vorfahre von v.
Sei y das Kind von z, das Vorfahre von v ist; y muss zwei höher sein als das andere Kind von z.
Sei x das Kind von y, das im selben Teilbaum wie v liegt.

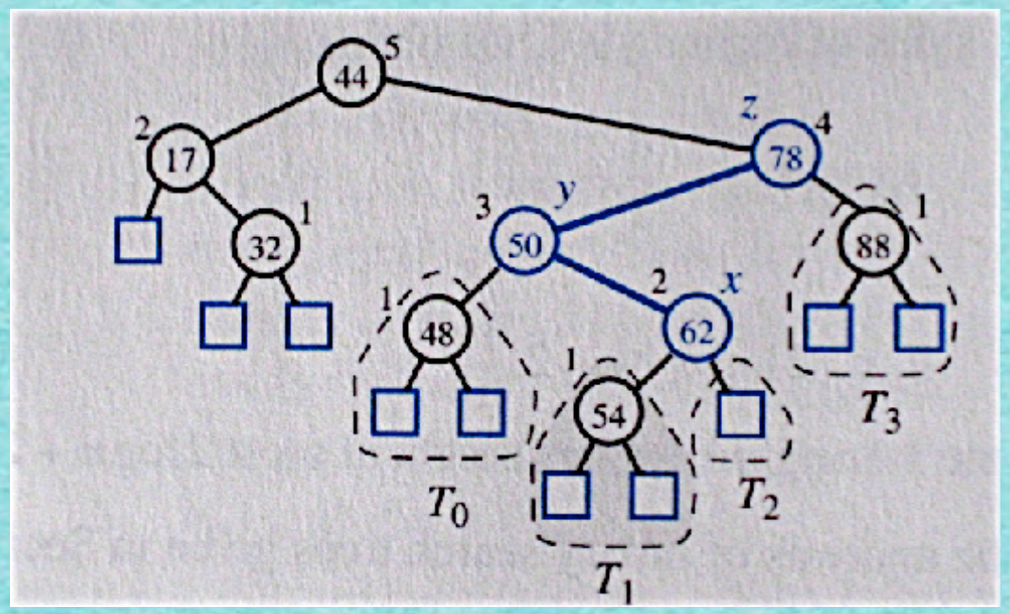


Einfügen



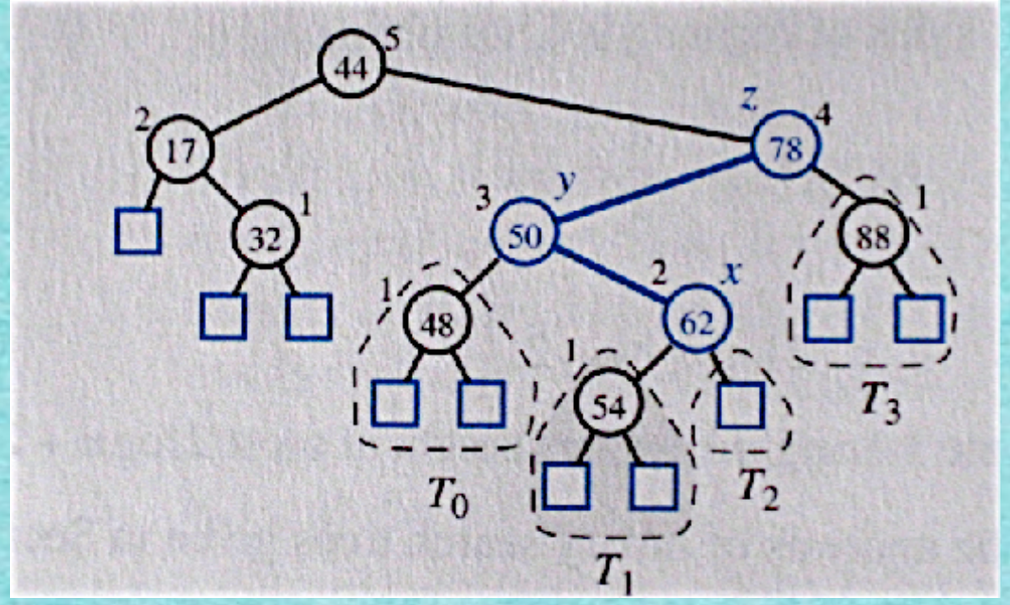
Beweis von Satz 4.10 (Forts.):

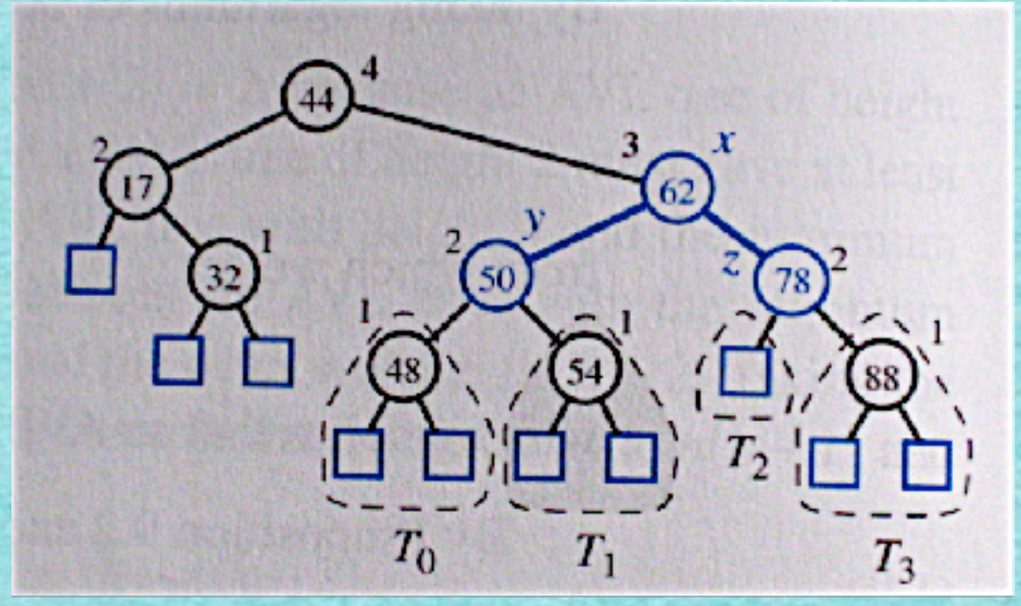
Jetzt ersetzen wir die Teilstruktur z, y, x (3 Knoten untereinander) durch eine Teilstruktur mit 2 Knoten unter einem.
Z.z.: Danach ist der Baum ein AVL-Baum!



Beweis von Satz 4.10 (Forts.):

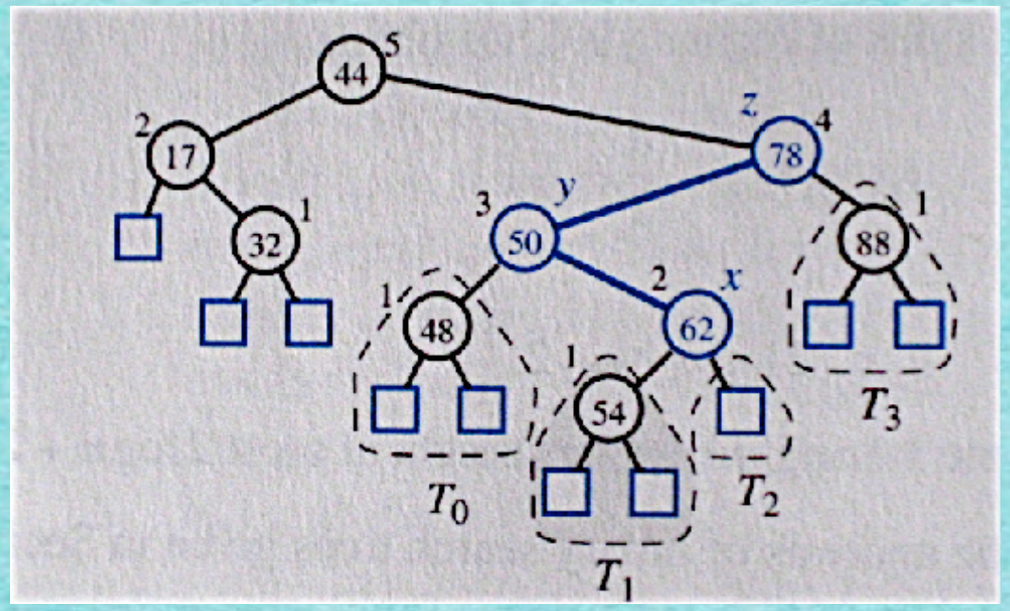
Jetzt ersetzen wir die Teilstruktur z, y, x (3 Knoten untereinander) durch eine Teilstruktur mit 2 Knoten unter einem.
Z.z.: Danach ist der Baum ein AVL-Baum!

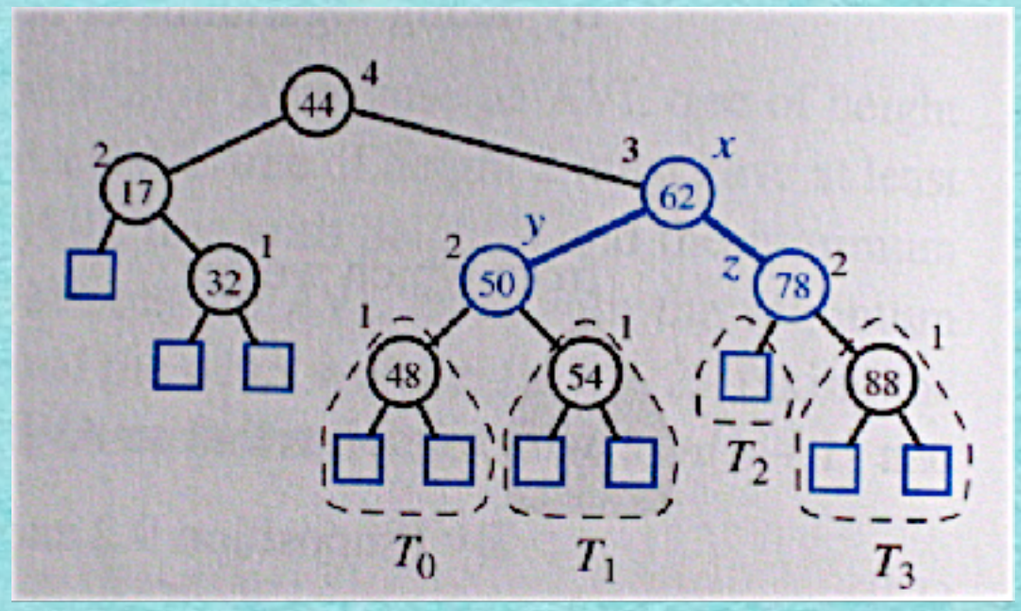




Beweis von Satz 4.10 (Forts.):

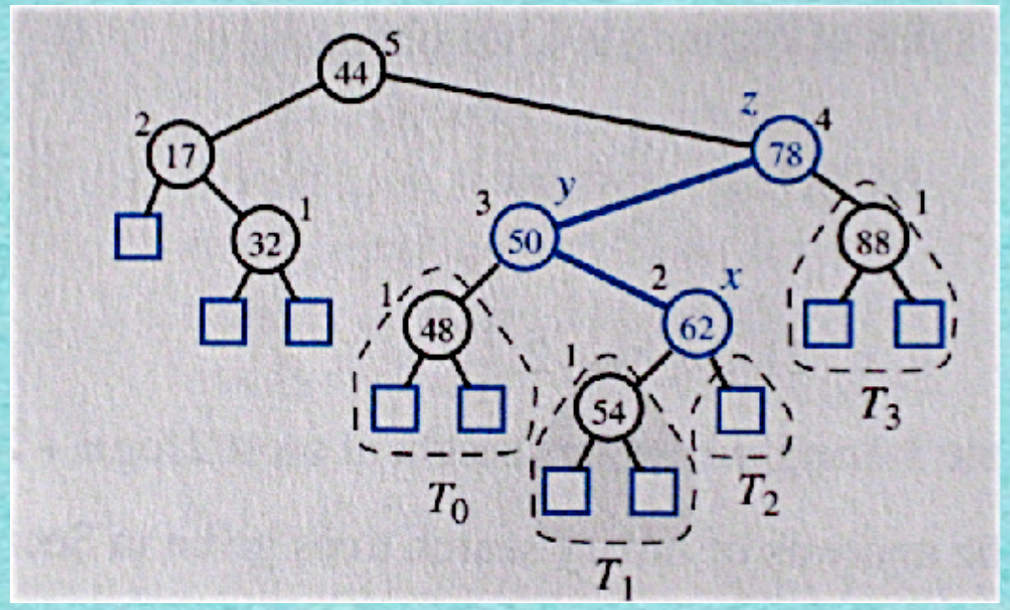
Jetzt ersetzen wir die Teilstruktur z, y, x (3 Knoten untereinander) durch eine Teilstruktur mit 2 Knoten unter einem.
Z.z.: Danach ist der Baum ein AVL-Baum!

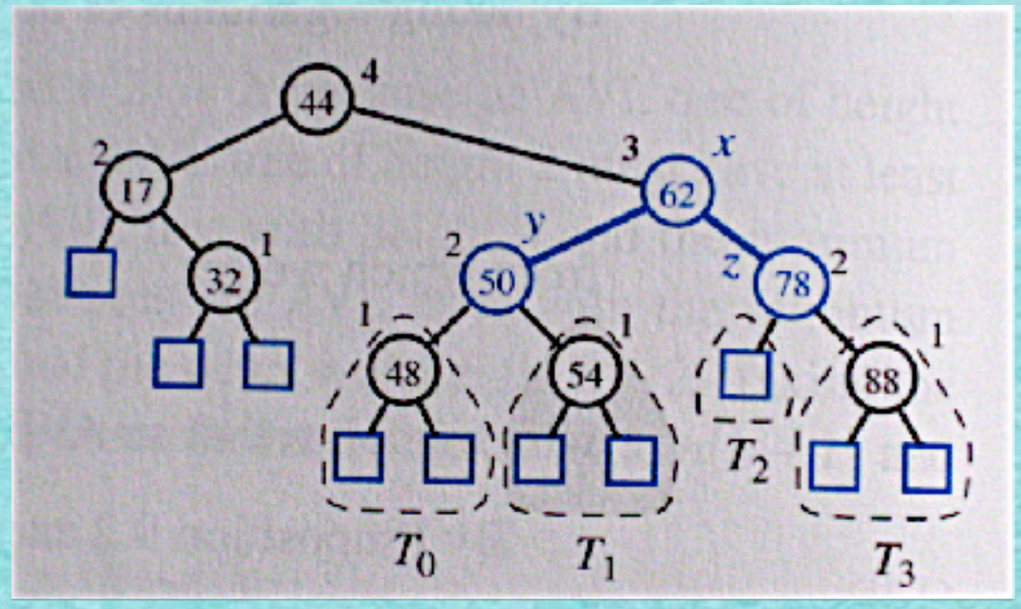




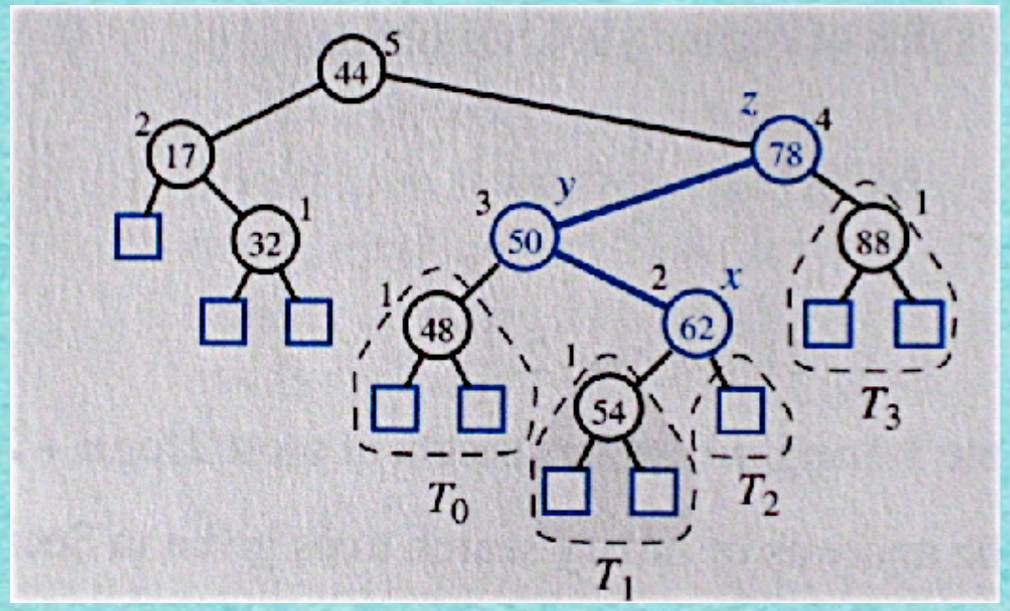
Betrachte jetzt die möglichen Anordnungen von x, y, z!

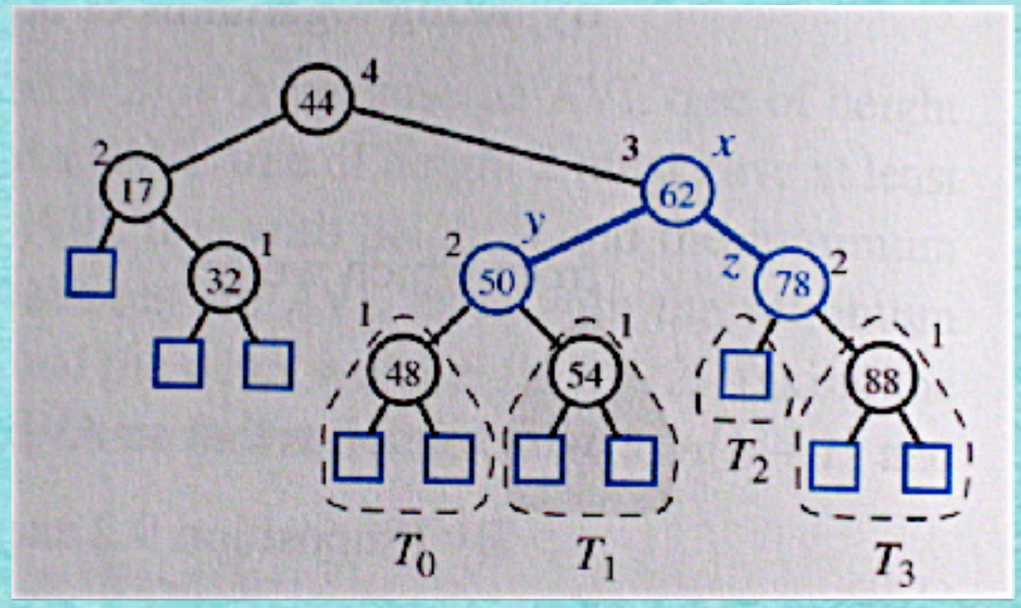
Einfügen


44
5
17
2
32
1
z
78
4
3
y
50
88
1
1
48
2
x
62
54
1
T_0
T_1
T_2
T_3


44
4
17
2
32
1
3
x
62
2
y
50
z
78
2
1
48
1
54
88
1
T_0
T_1
T_2
T_3

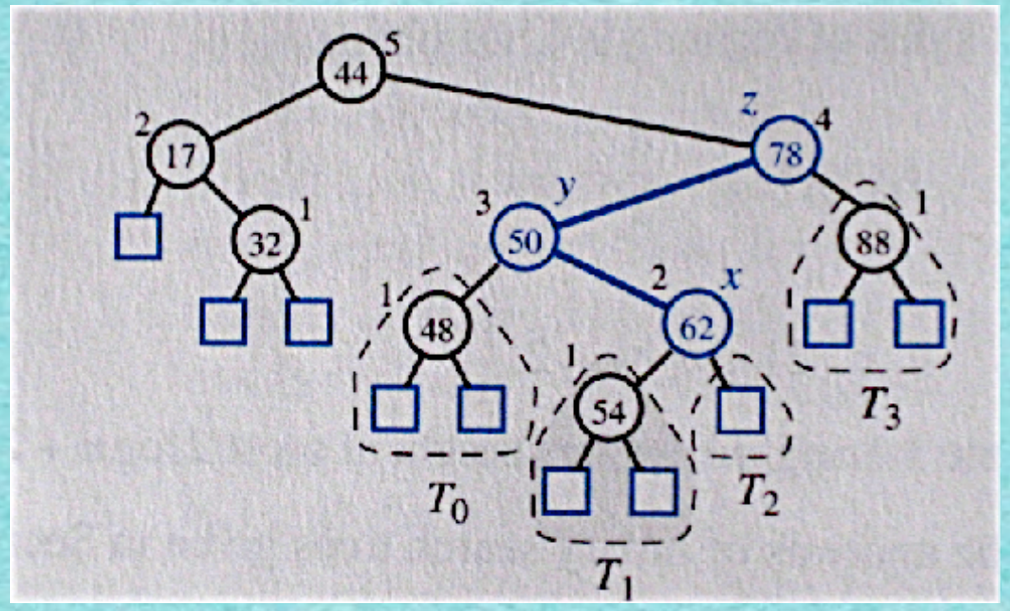
Beweis von Satz 4.10 (Forts.):

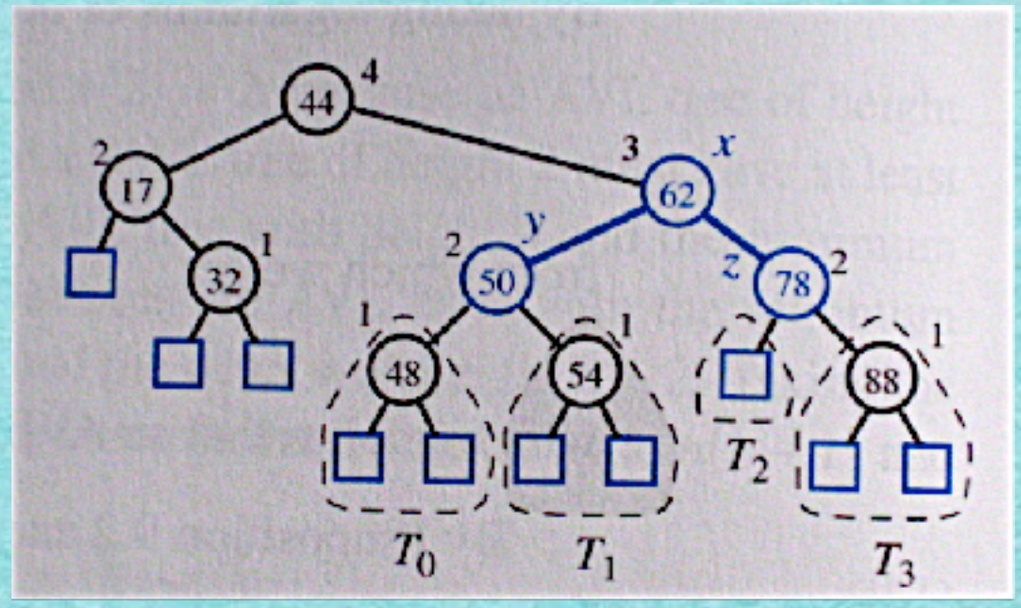




Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

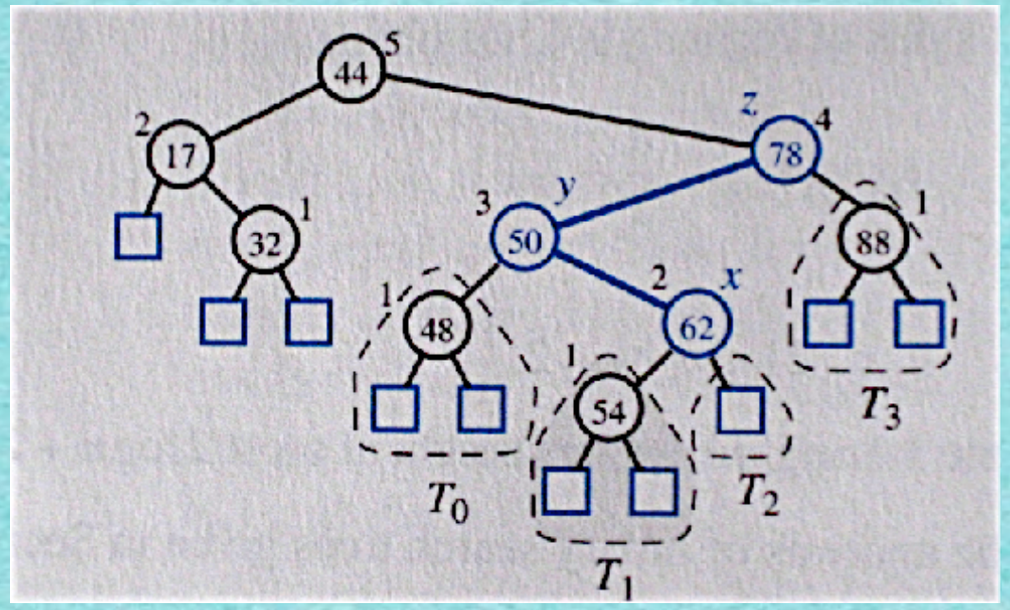


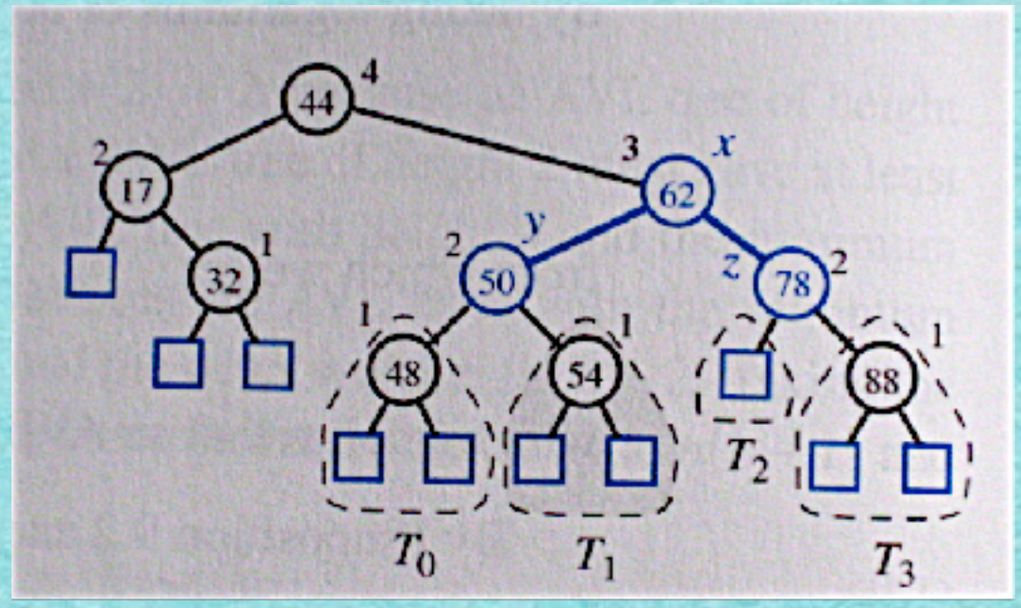


Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

(1) $x \le y \le z$



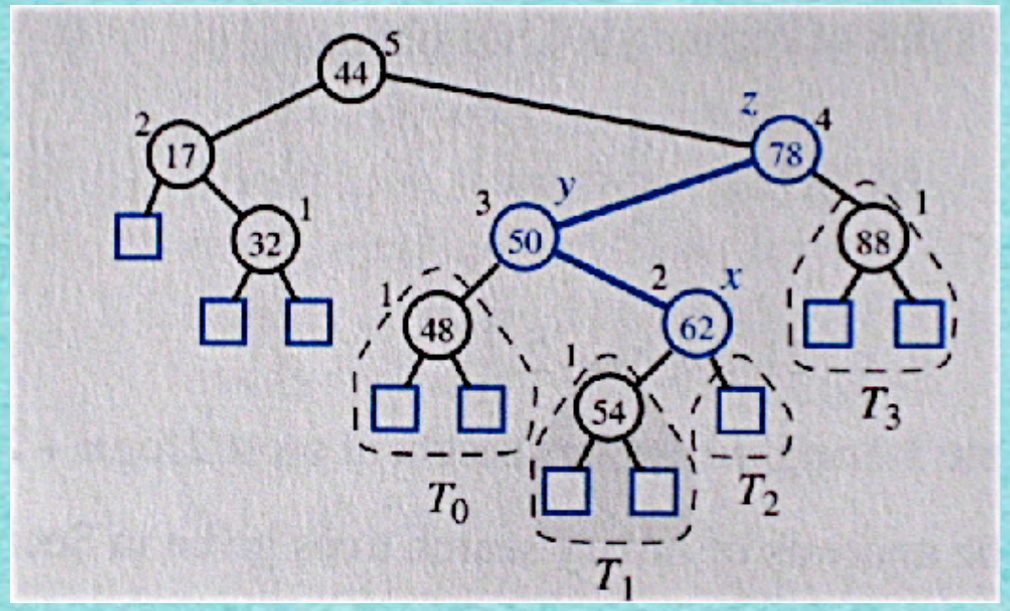


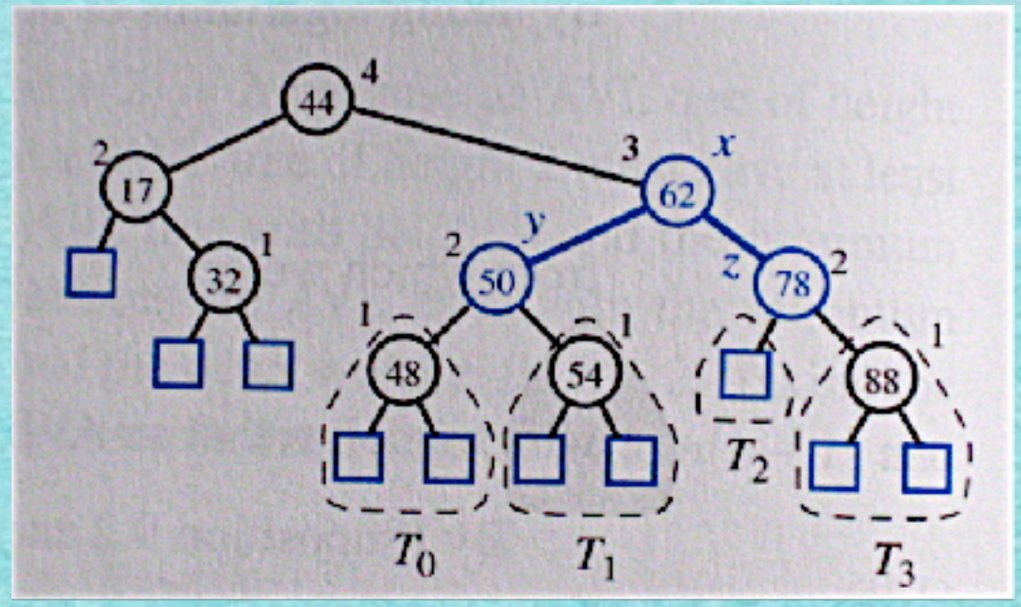
Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

(1) $x \le y \le z$

(2) $x \le z \le y$





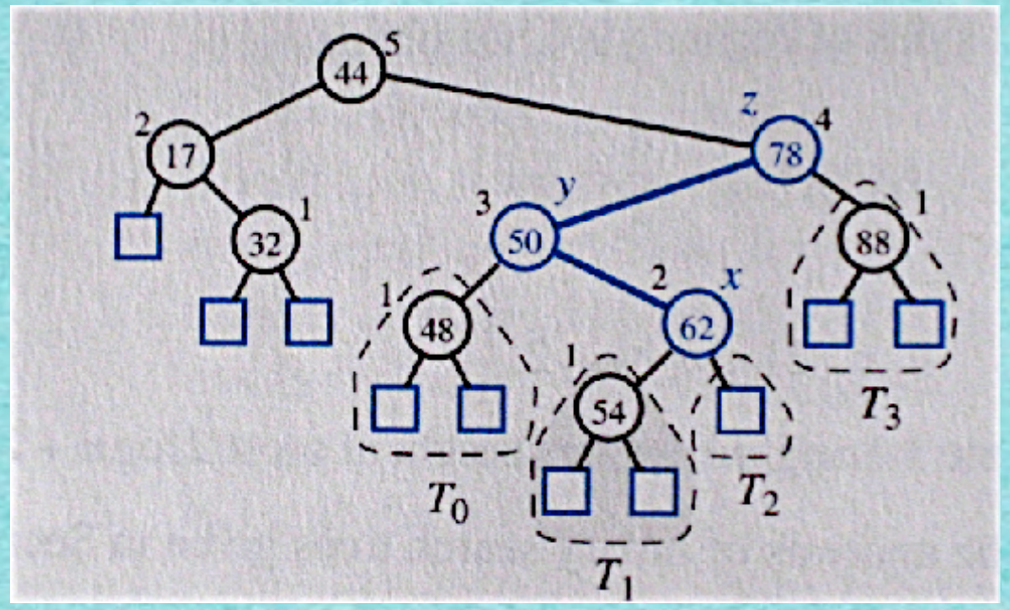
Beweis von Satz 4.10 (Forts.):

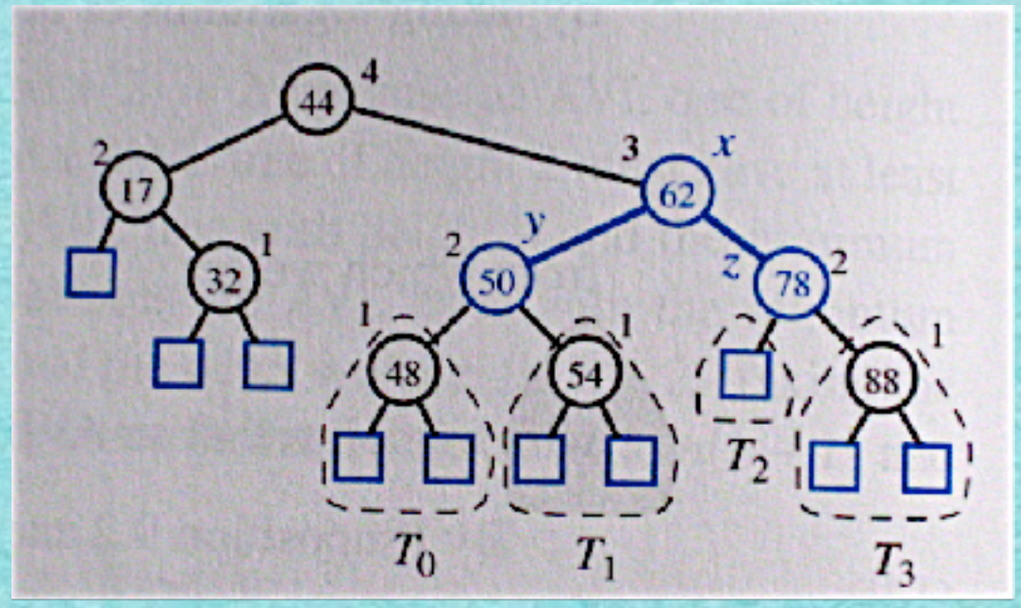
Welche Anordnungen gibt es?

(1) $x \le y \le z$

(2) $x \le z \le y$

(3) $y \le x \le z$





Beweis von Satz 4.10 (Forts.):

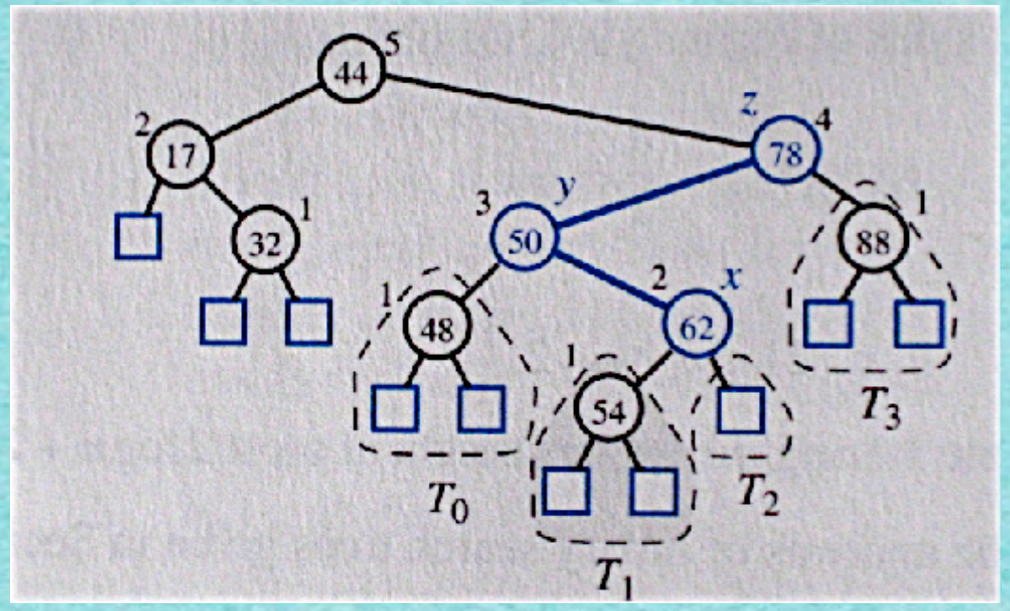
Welche Anordnungen gibt es?

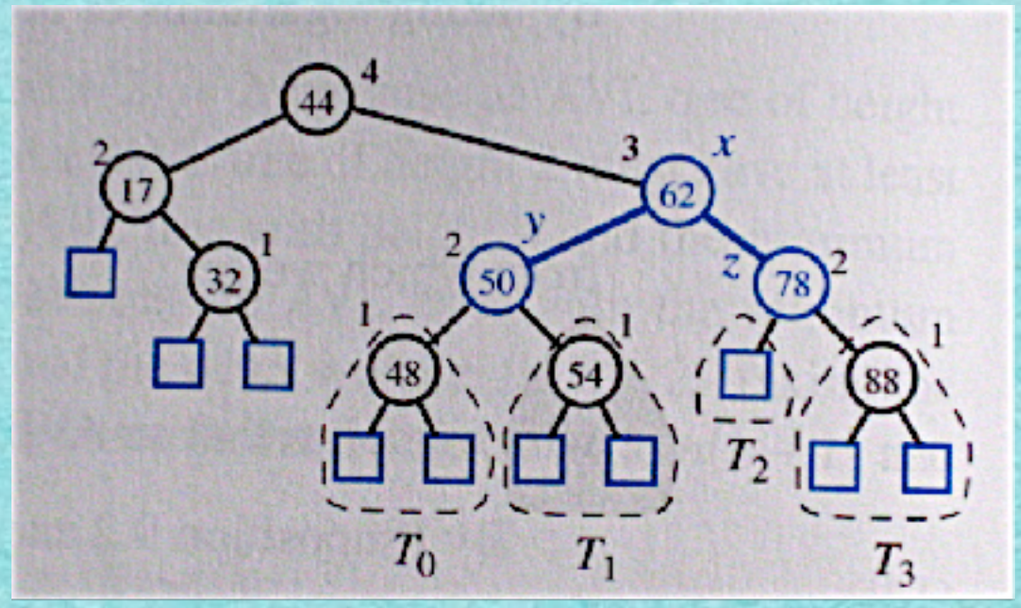
(1) $x \le y \le z$

(2) $x \le z \le y$

(3) $y \le x \le z$

(4) $y \le z \le x$





Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

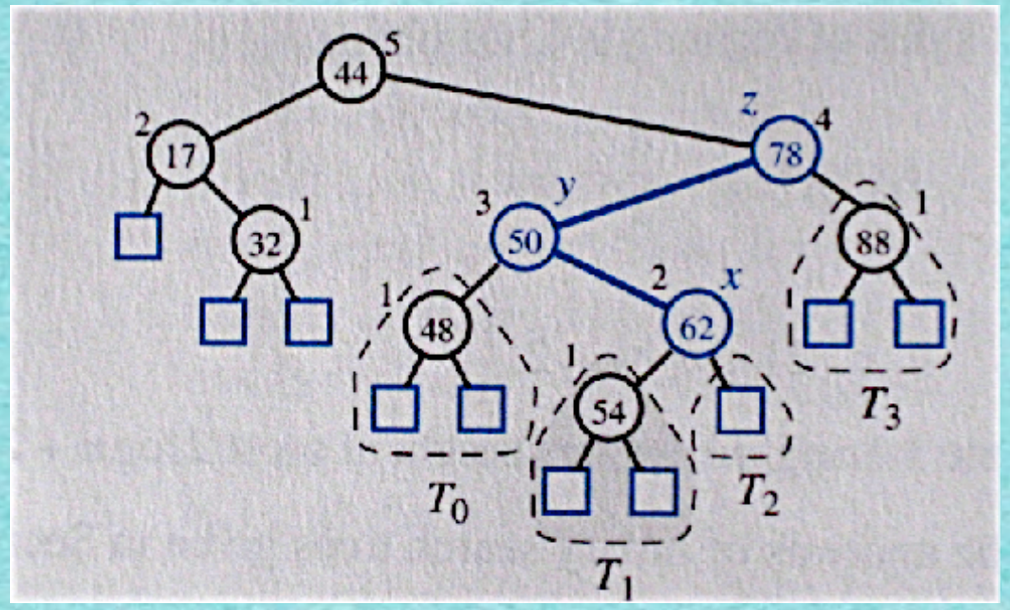
(1) $x \le y \le z$

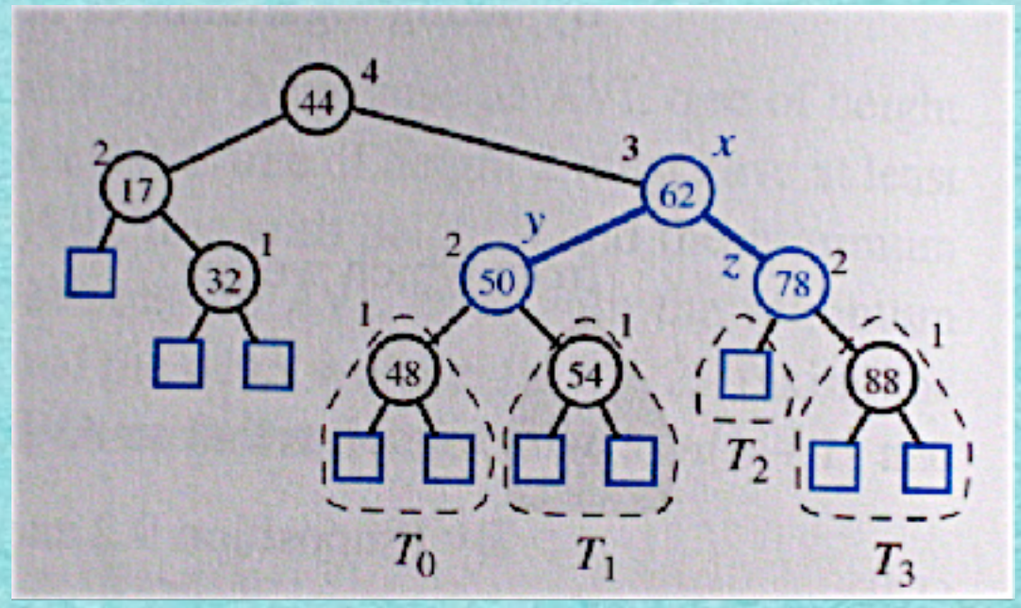
(2) $x \le z \le y$

(3) $y \le x \le z$

(4) $y \le z \le x$

(5) $z \le x \le y$





Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

(1) $x \le y \le z$

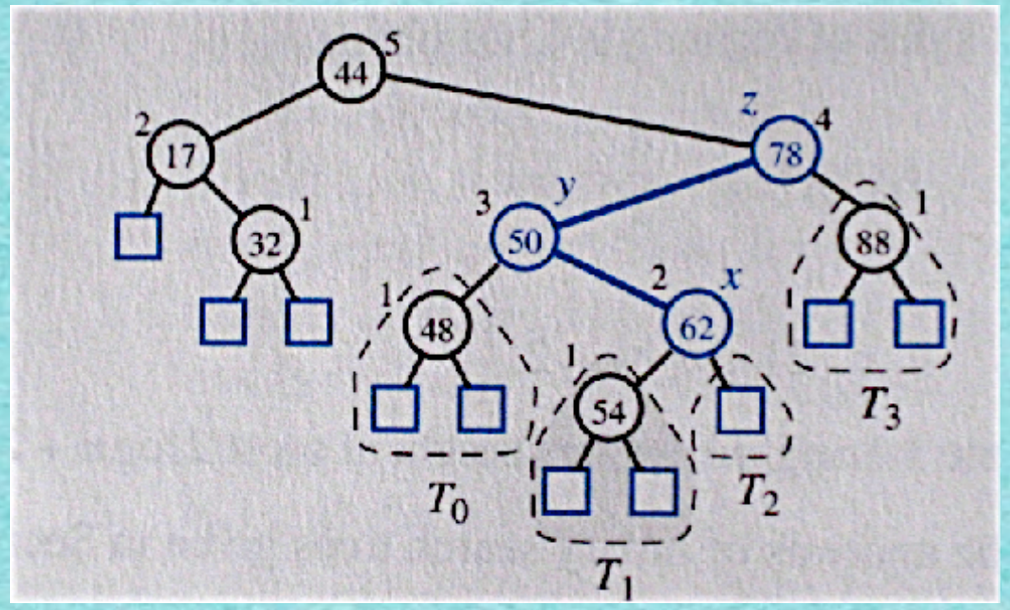
(2) $x \le z \le y$

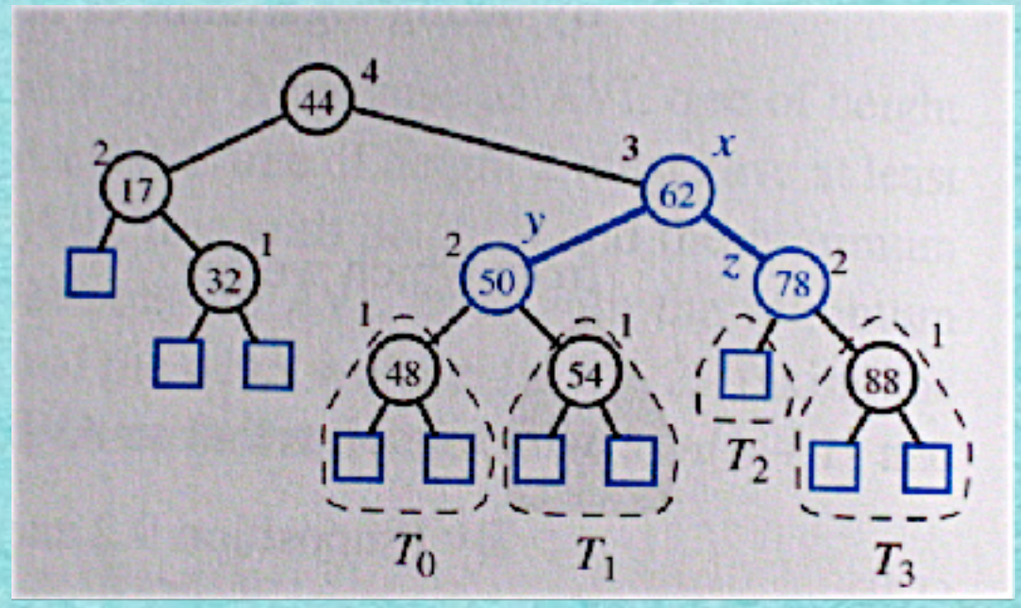
(3) $y \le x \le z$

(4) $y \le z \le x$

(5) $z \le x \le y$

(6) $z \le y \le x$





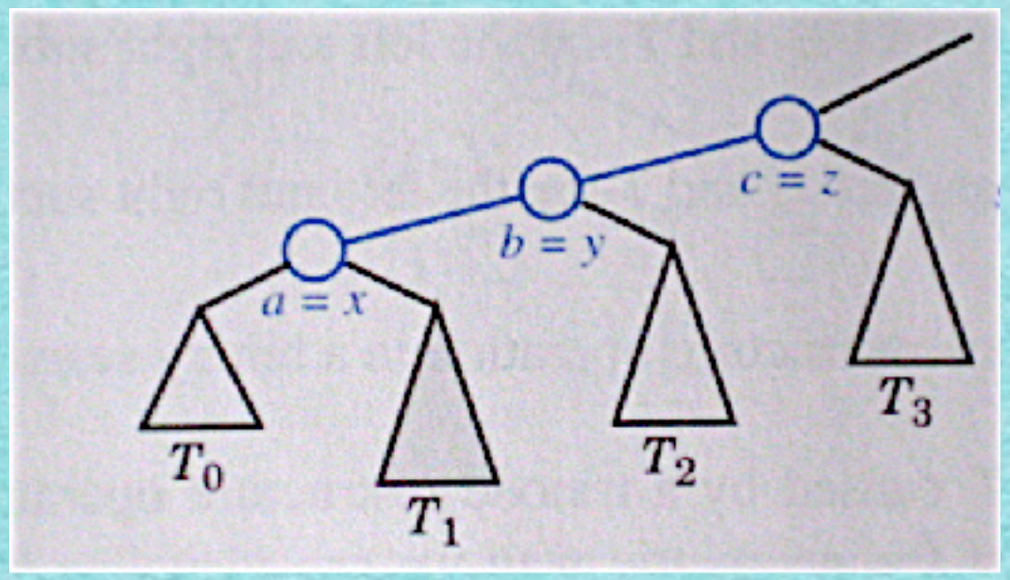
Einfügen

Beweis von Satz 4.10 (Forts.):

(1) $x \leq y \leq z$

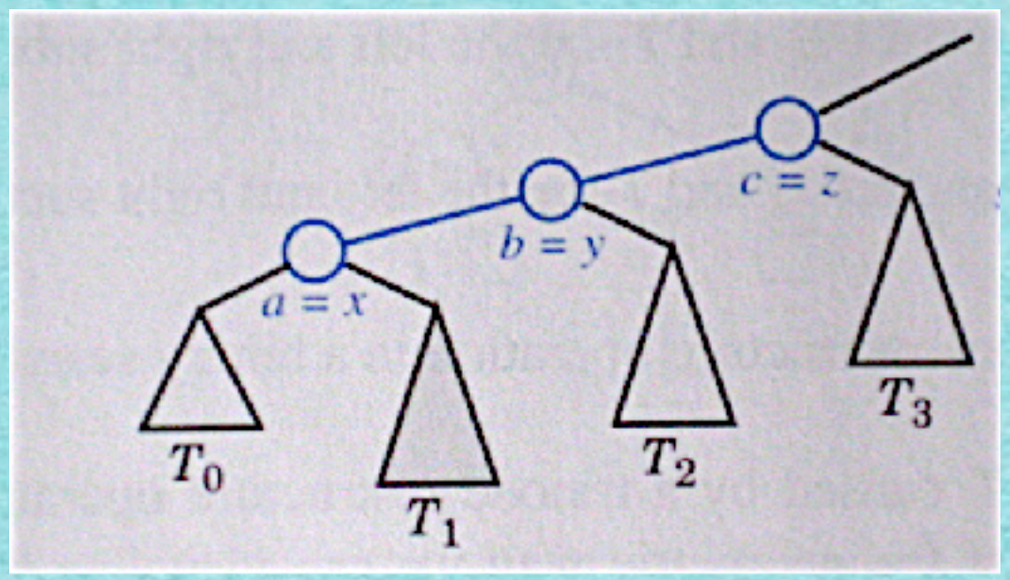
Beweis von Satz 4.10 (Forts.):

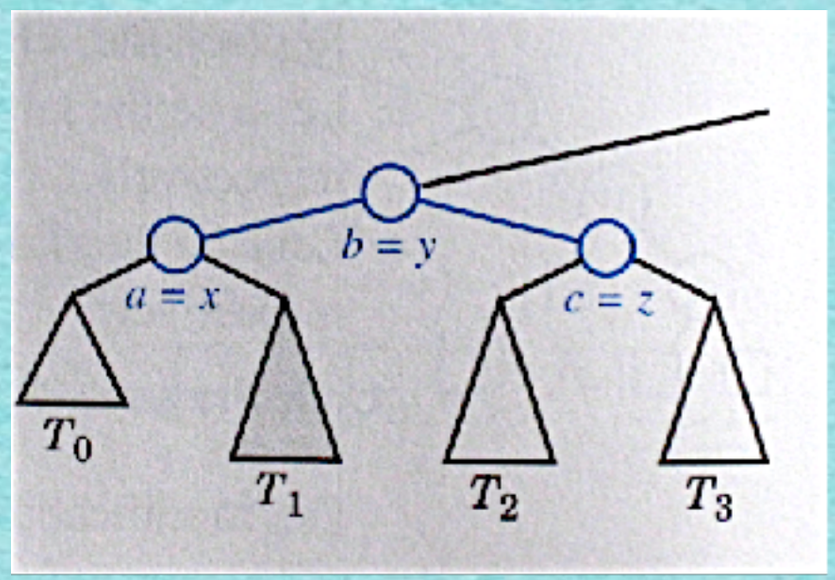
(1) $x \le y \le z$



Beweis von Satz 4.10 (Forts.):

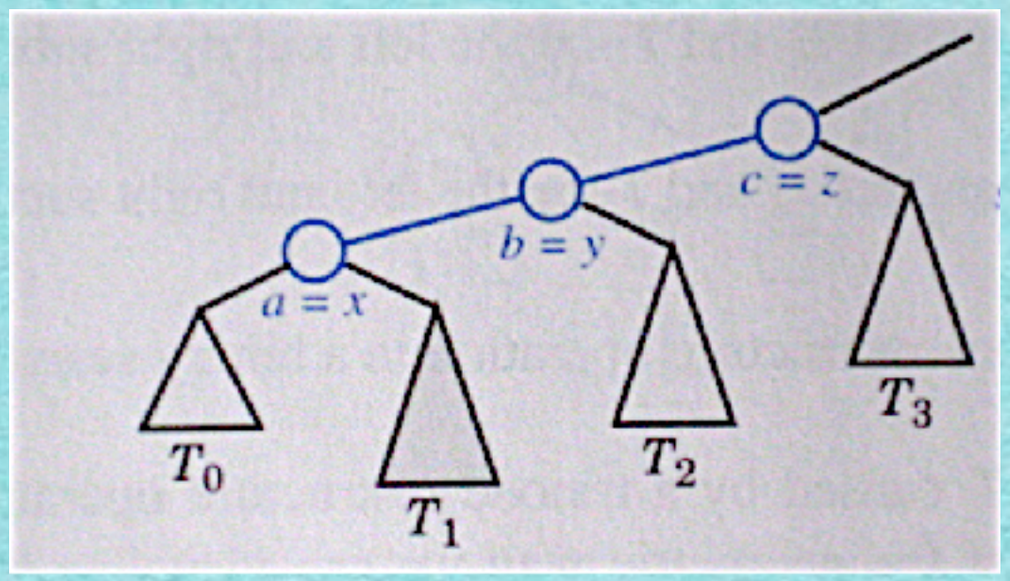
(1) $x \le y \le z$

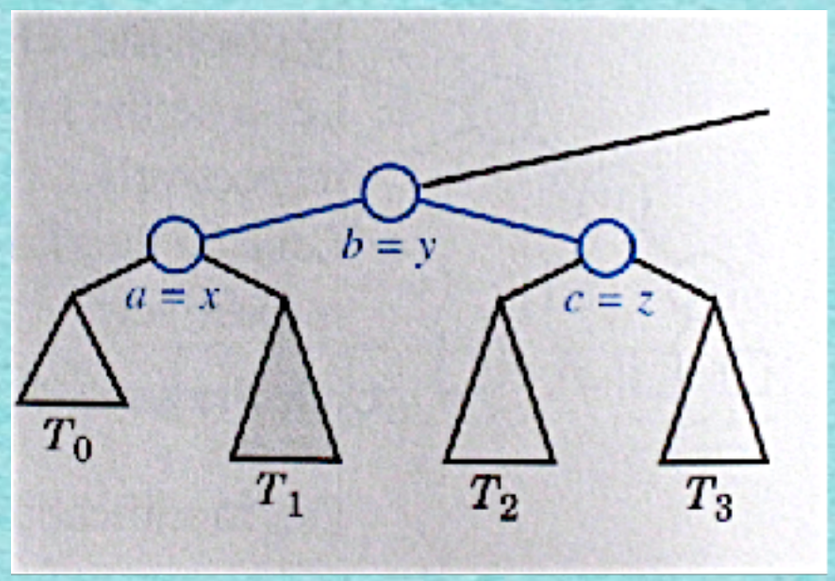




Beweis von Satz 4.10 (Forts.):

(1) $x \le y \le z$





Der Baum ist wieder höhenbalanciert!

Einfügen

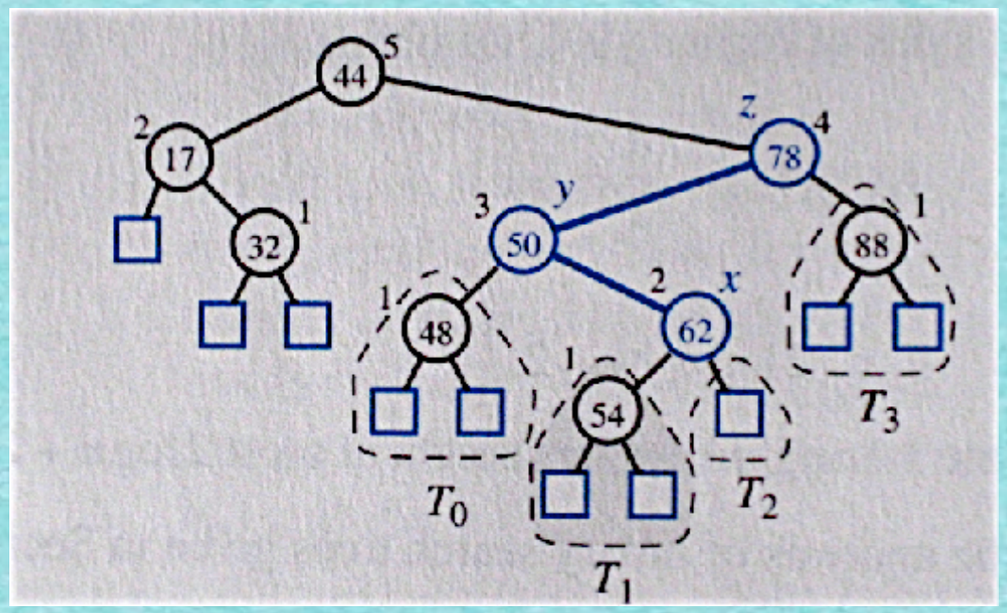
Beweis von Satz 4.10 (Forts.):

Beweis von Satz 4.10 (Forts.):

(2) $x \leq z \leq y$

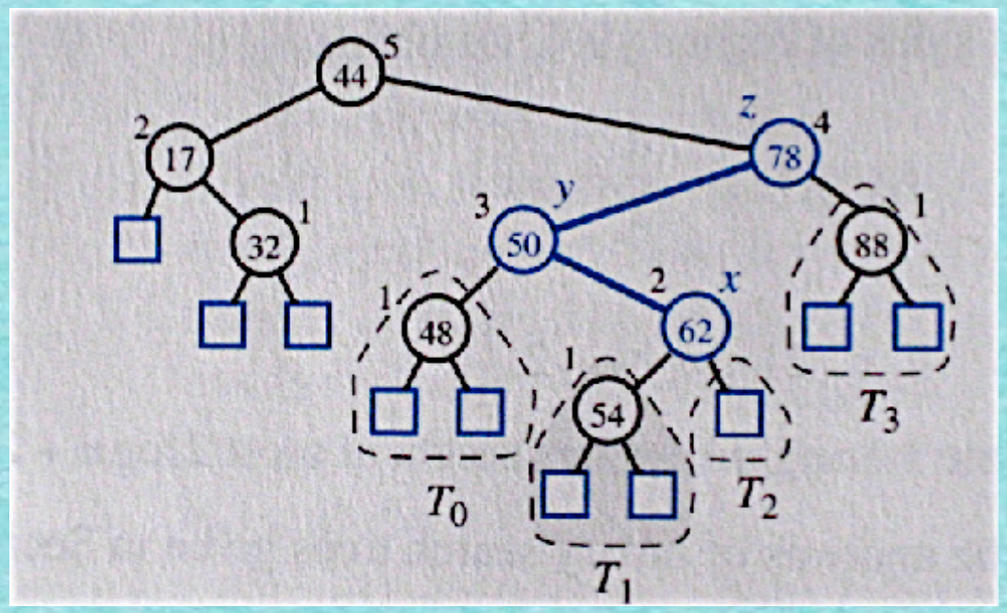
Beweis von Satz 4.10 (Forts.):

(2) $x \le z \le y$



Beweis von Satz 4.10 (Forts.):

(2) $x \leq z \leq y$



Der Fall kann nicht auftreten!

Beweis von Satz 4.10 (Forts.):

(2) $x \leq z \leq y$

Der Fall kann nicht auftreten!

Einfügen

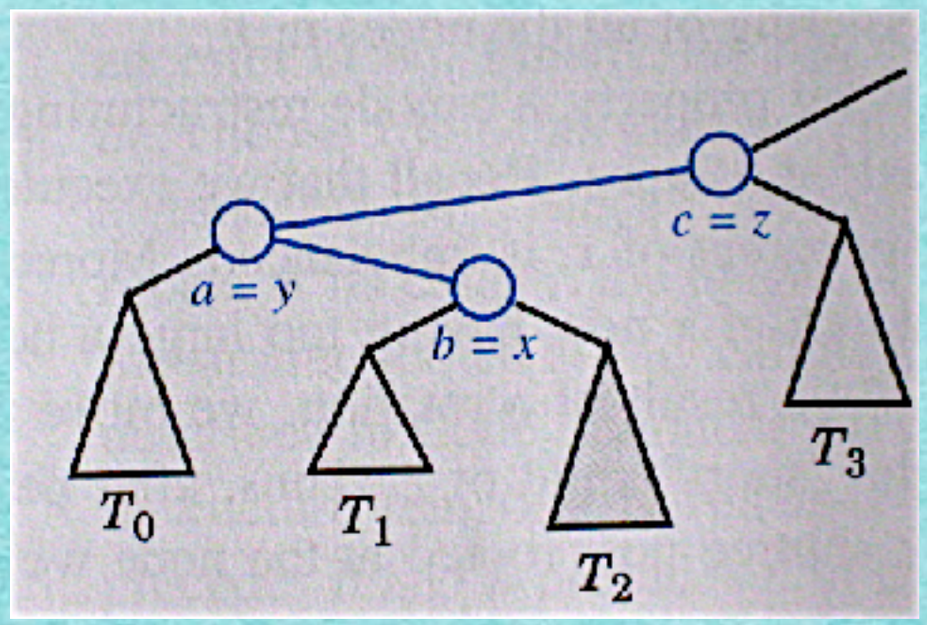
Beweis von Satz 4.10 (Forts.):

Beweis von Satz 4.10 (Forts.):

(3) $y \leq x \leq z$

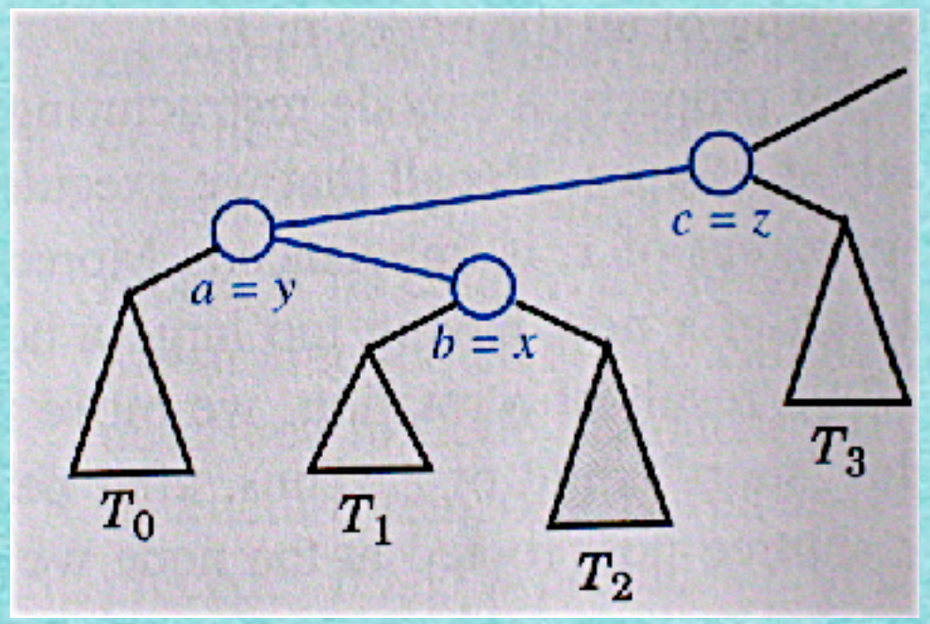
Beweis von Satz 4.10 (Forts.):

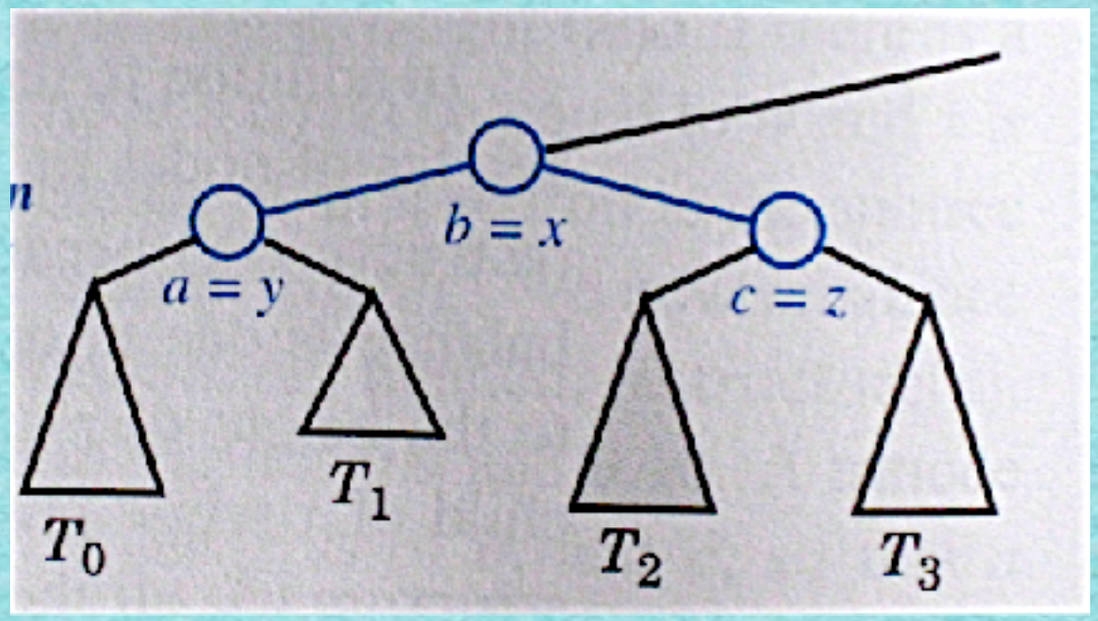
(3) $y \le x \le z$



Beweis von Satz 4.10 (Forts.):

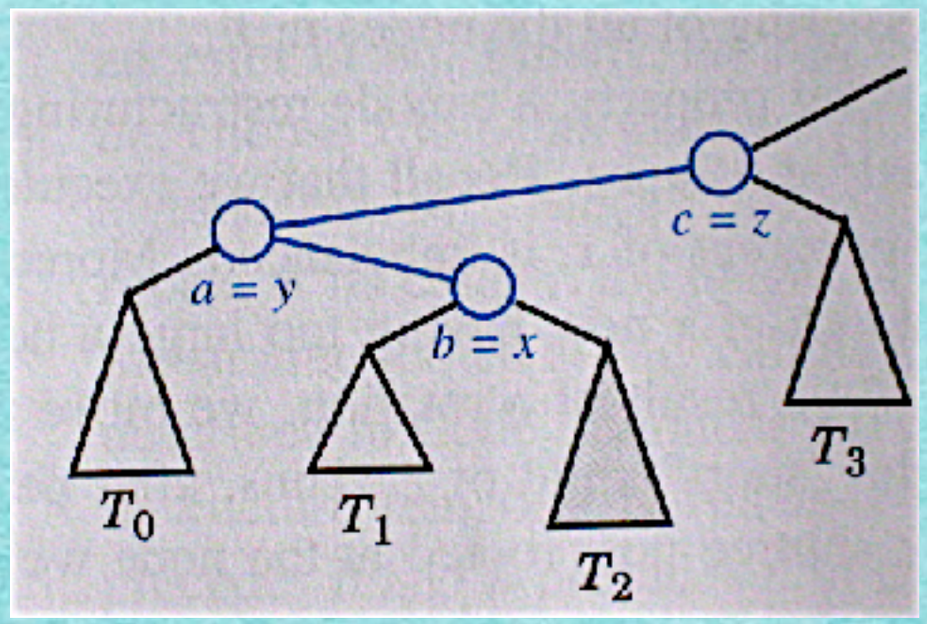
(3) $y \le x \le z$

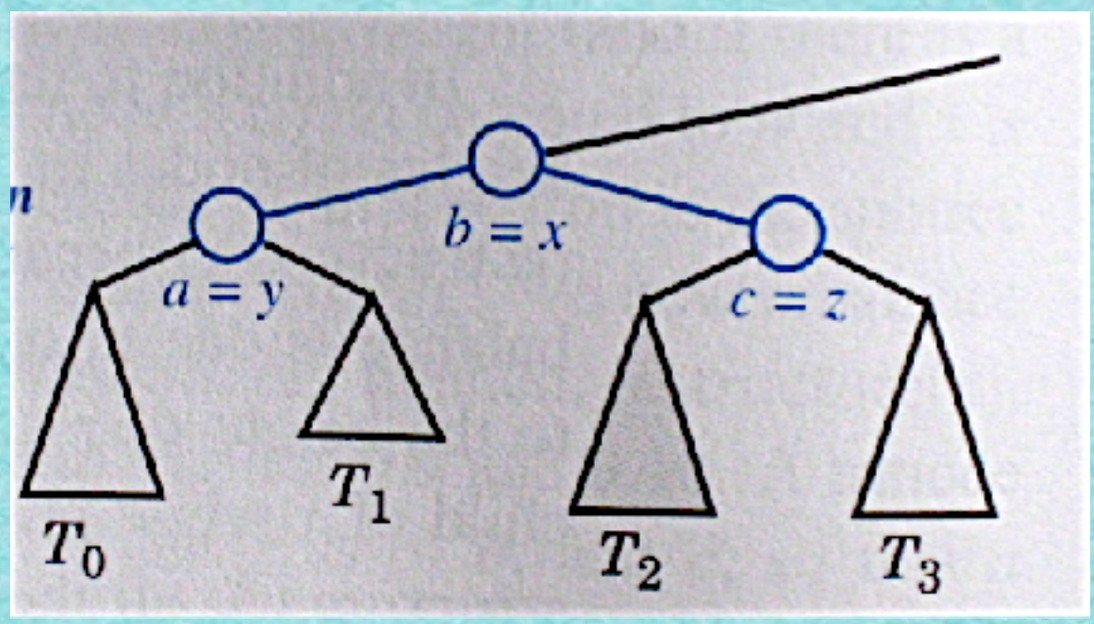




Beweis von Satz 4.10 (Forts.):

(3) $y \le x \le z$





Der Baum ist wieder höhenbalanciert!

Einfügen

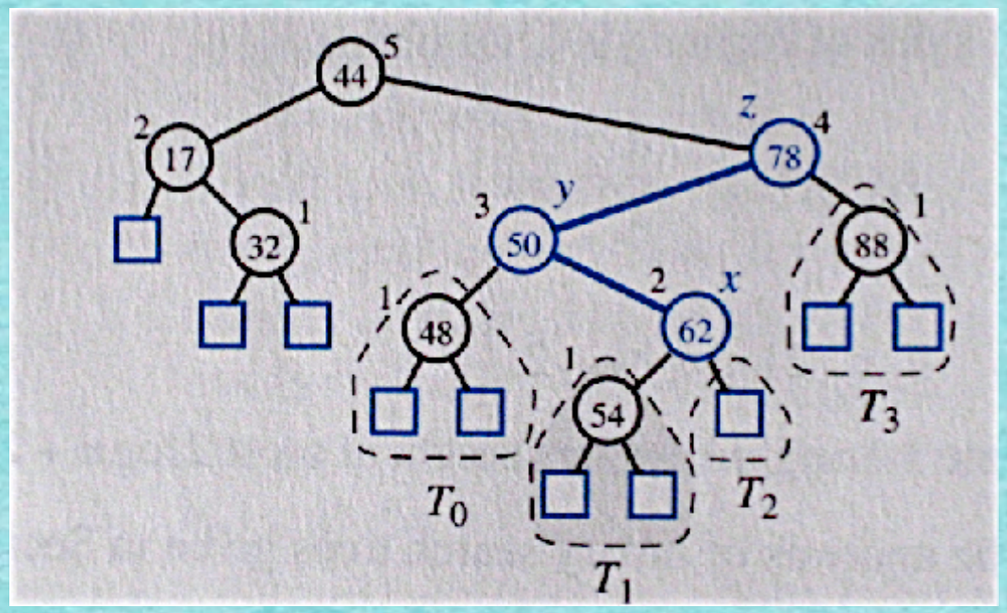
Beweis von Satz 4.10 (Forts.):

Beweis von Satz 4.10 (Forts.):

(4) $y \leq z \leq x$

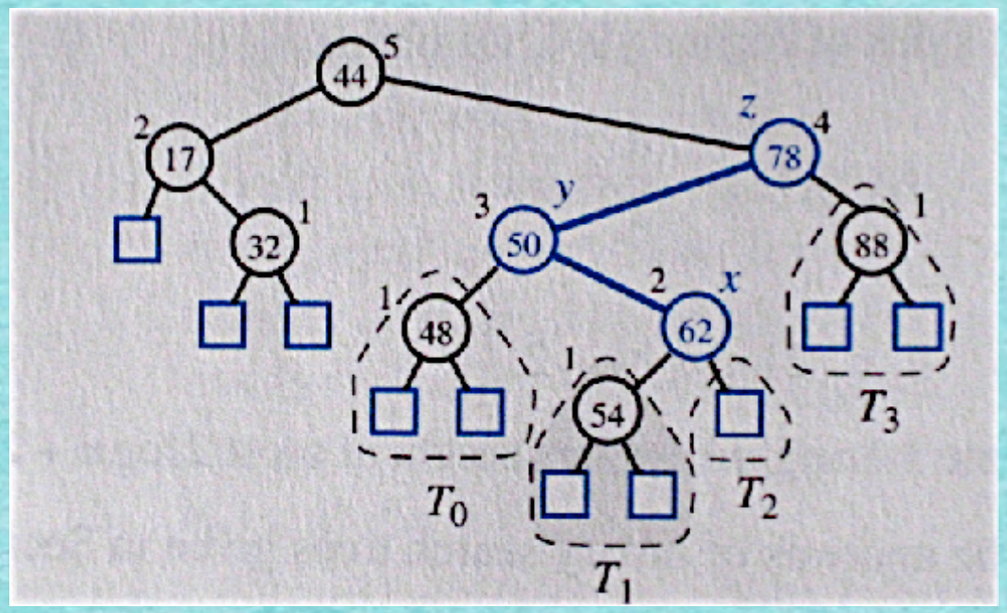
Beweis von Satz 4.10 (Forts.):

(4) $y \le z \le x$



Beweis von Satz 4.10 (Forts.):

(4) $y \le z \le x$



Der Fall kann nicht auftreten!

Beweis von Satz 4.10 (Forts.):

(4) $y \leq z \leq x$

Der Fall kann nicht auftreten!

Einfügen

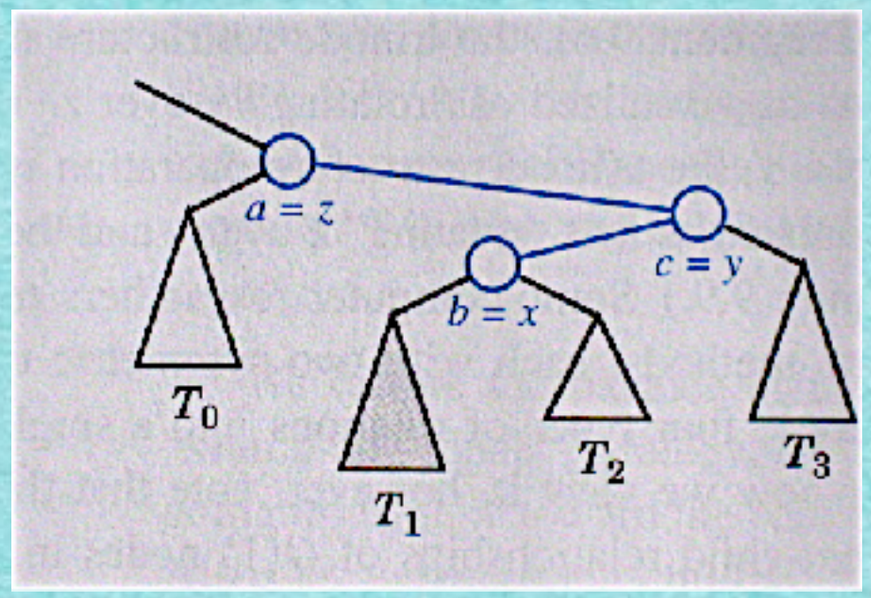
Beweis von Satz 4.10 (Forts.):

Beweis von Satz 4.10 (Forts.):

(5) $z \leq x \leq y$

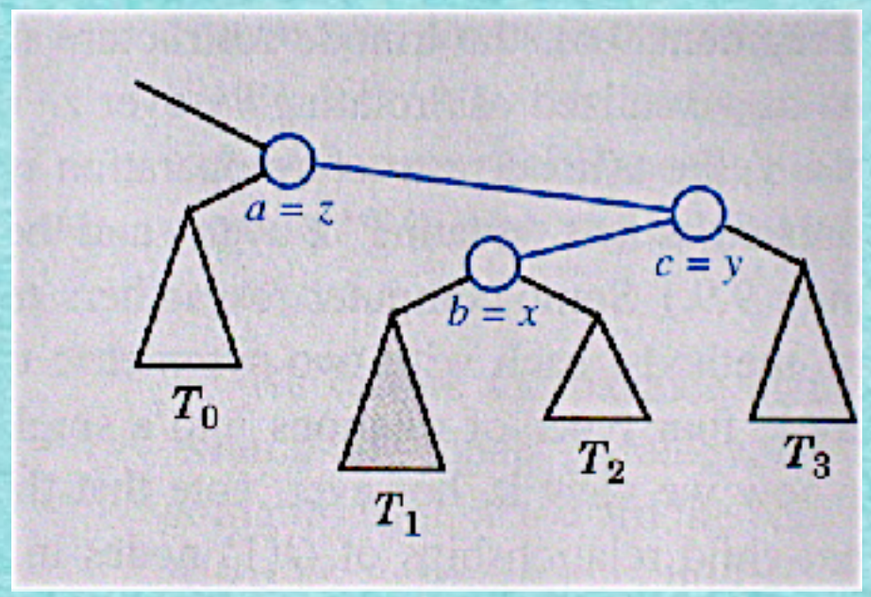
Beweis von Satz 4.10 (Forts.):

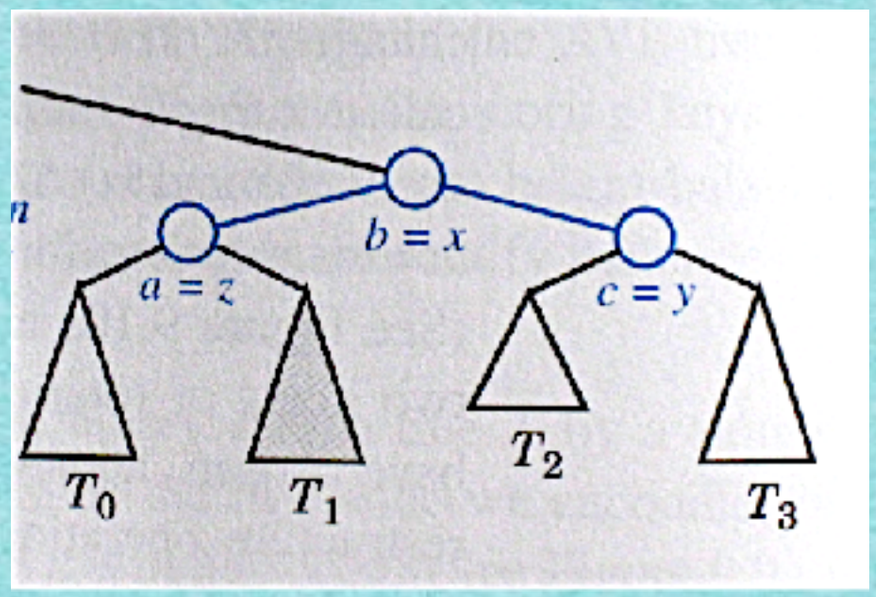
(5) z≤x≤y



Beweis von Satz 4.10 (Forts.):

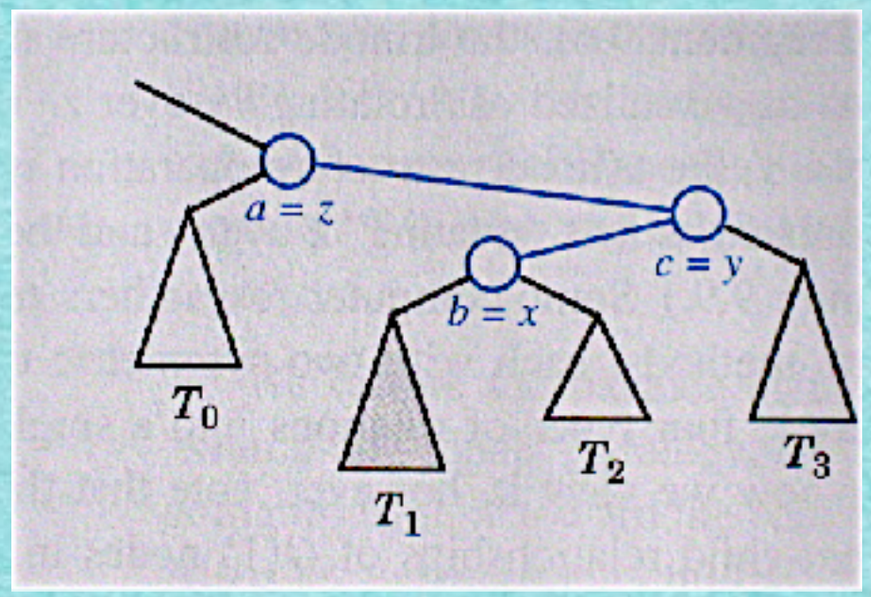
(5) z≤x≤y

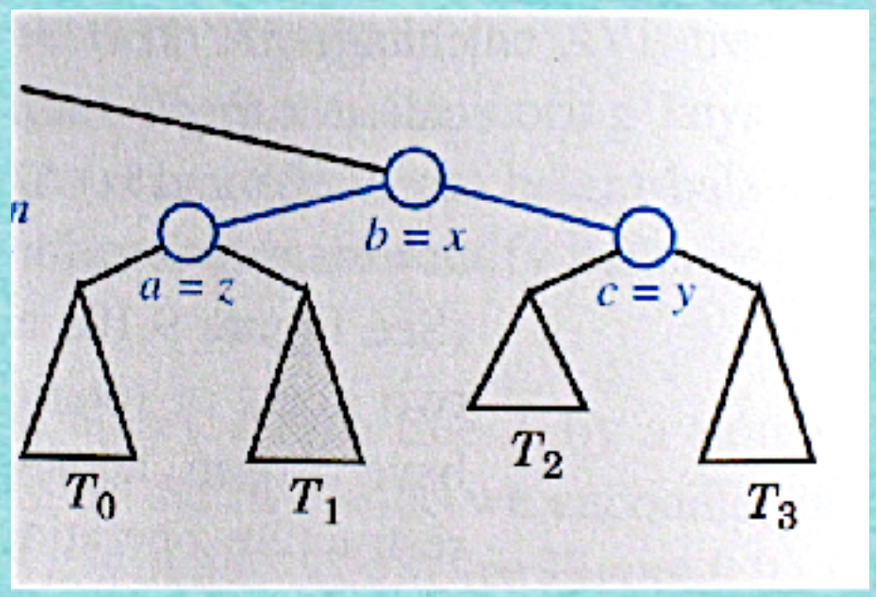




Beweis von Satz 4.10 (Forts.):

(5) z≤x≤y





Der Baum ist wieder höhenbalanciert!

Einfügen

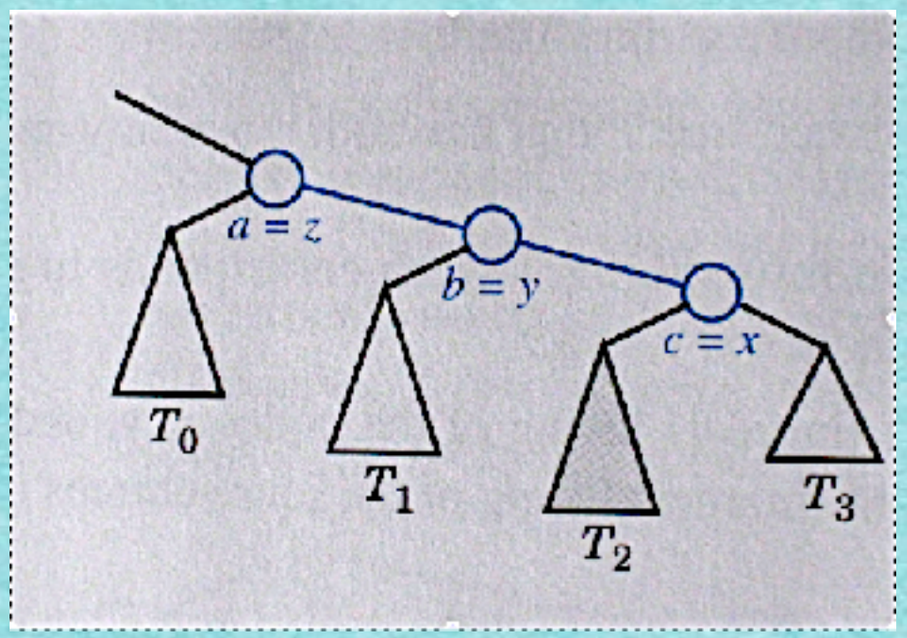
Beweis von Satz 4.10 (Forts.):

Beweis von Satz 4.10 (Forts.):

(6) $z \leq y \leq x$

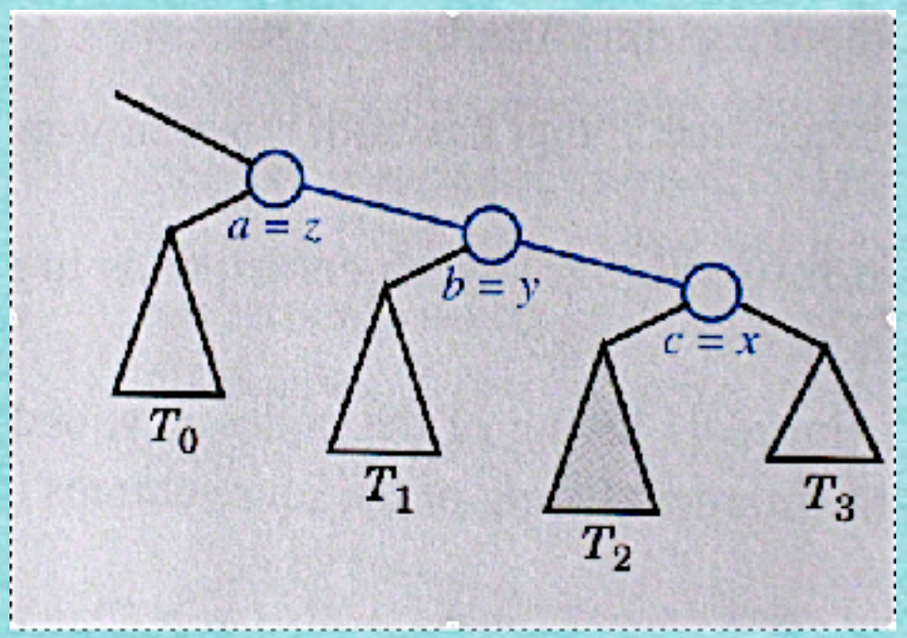
Beweis von Satz 4.10 (Forts.):

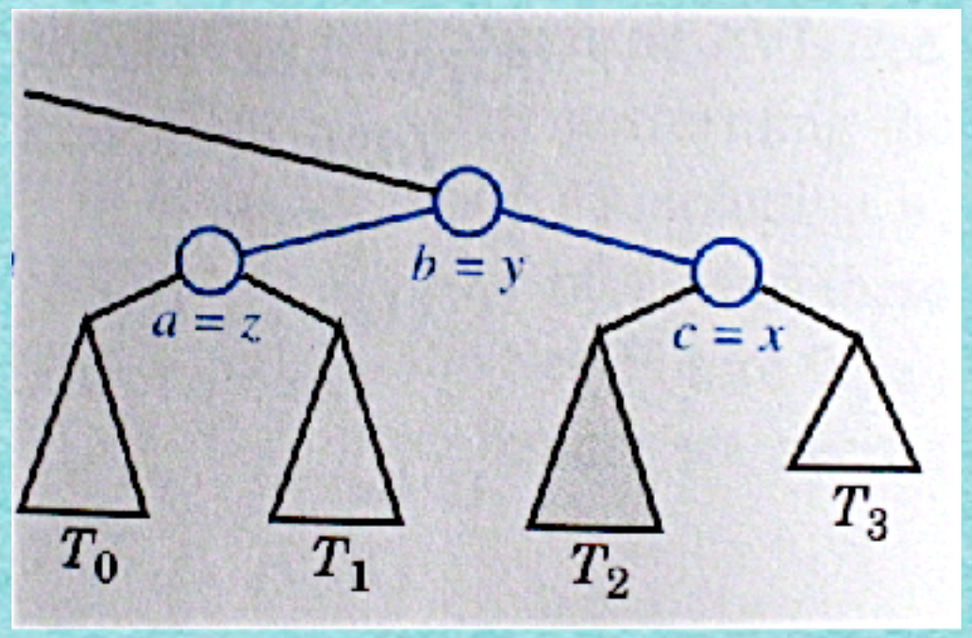
(6) $z \le y \le x$



Beweis von Satz 4.10 (Forts.):

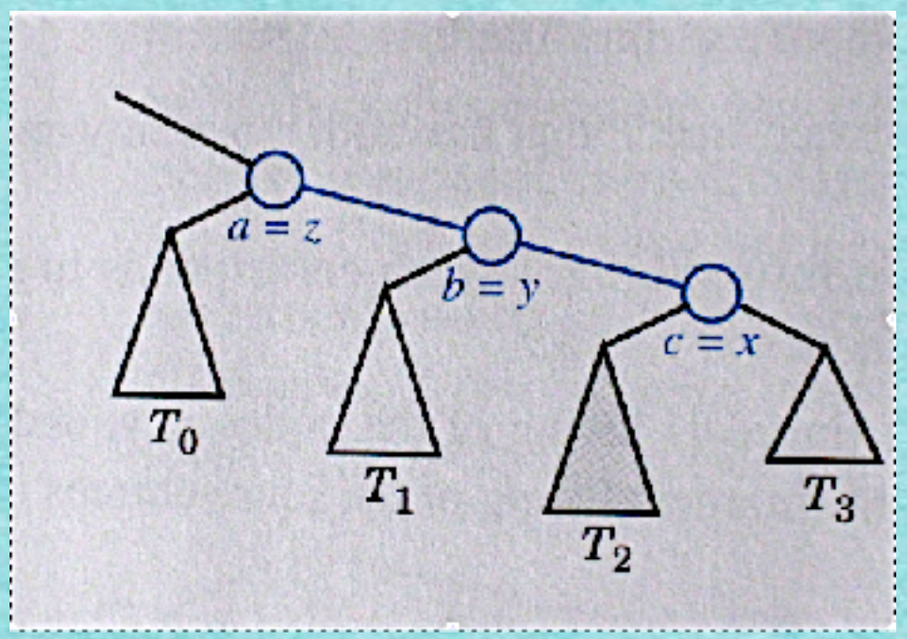
(6) $z \le y \le x$

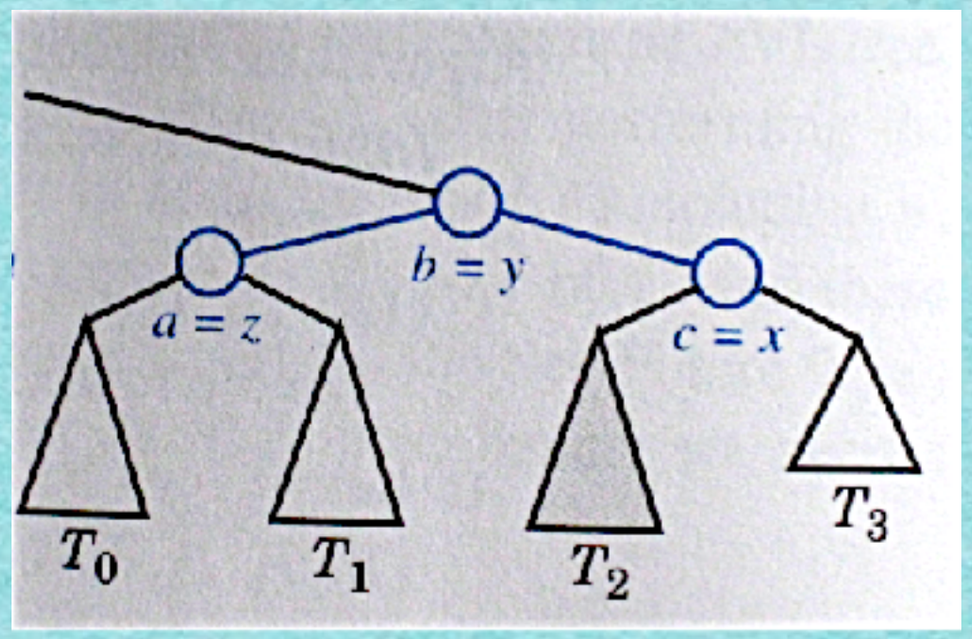




Beweis von Satz 4.10 (Forts.):

(6) z≤y≤x

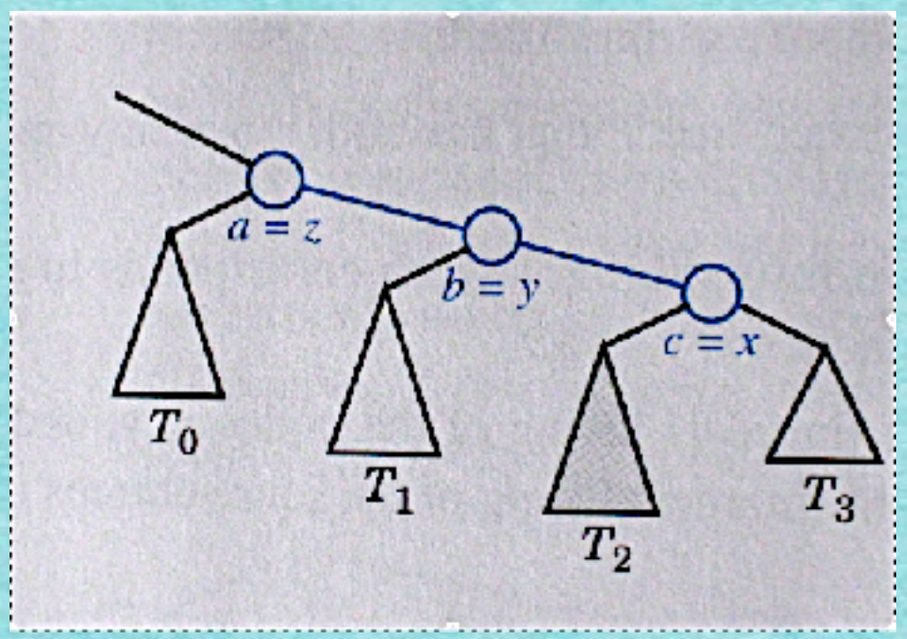


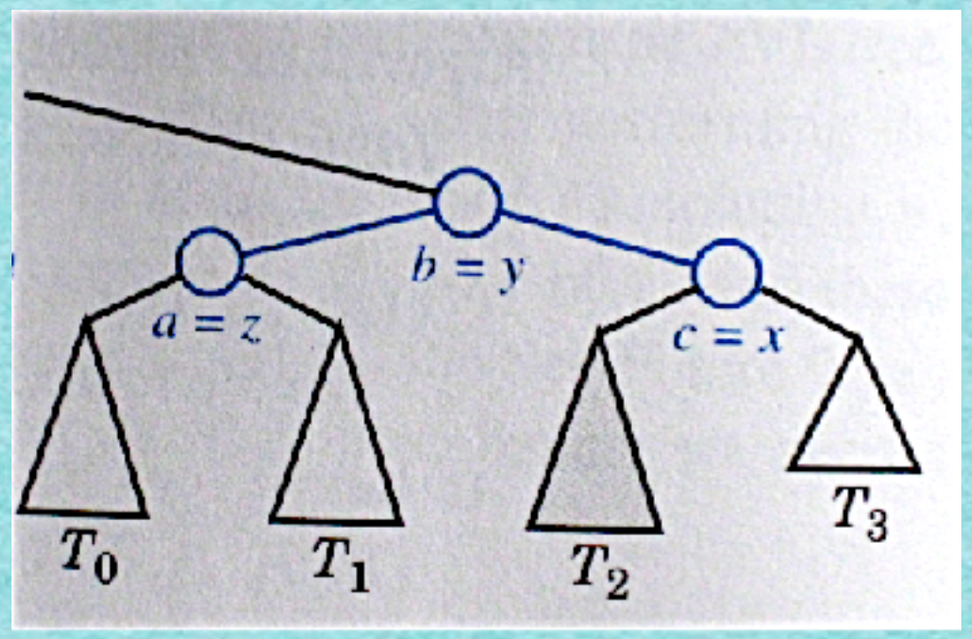


Der Baum ist wieder höhenbalanciert!

Beweis von Satz 4.10 (Forts.):

(6) z≤y≤x



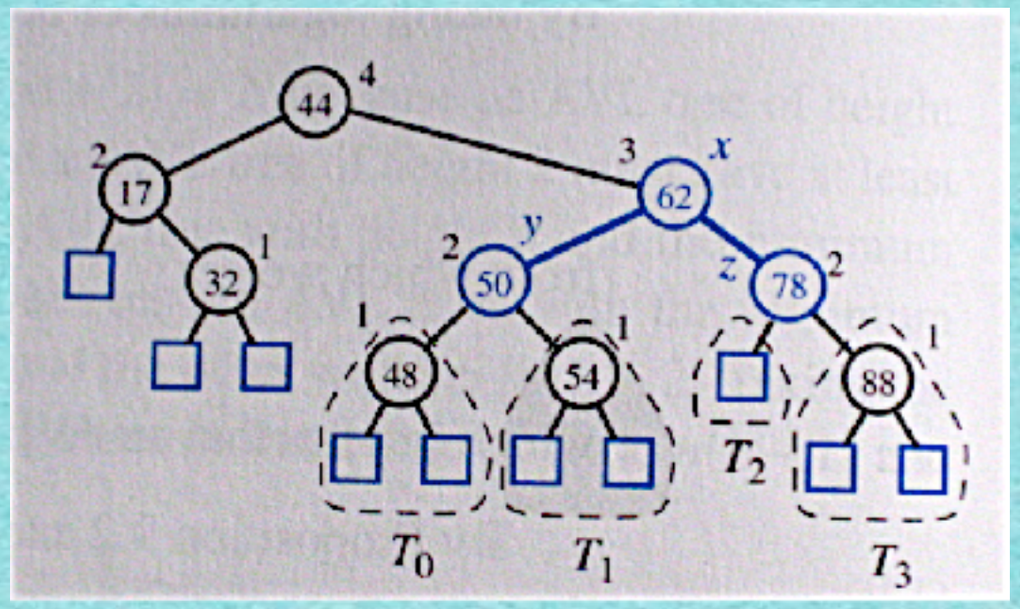


Der Baum ist wieder höhenbalanciert!

Alle Schritte erfordern nur konstant viele Rechenoperationen.

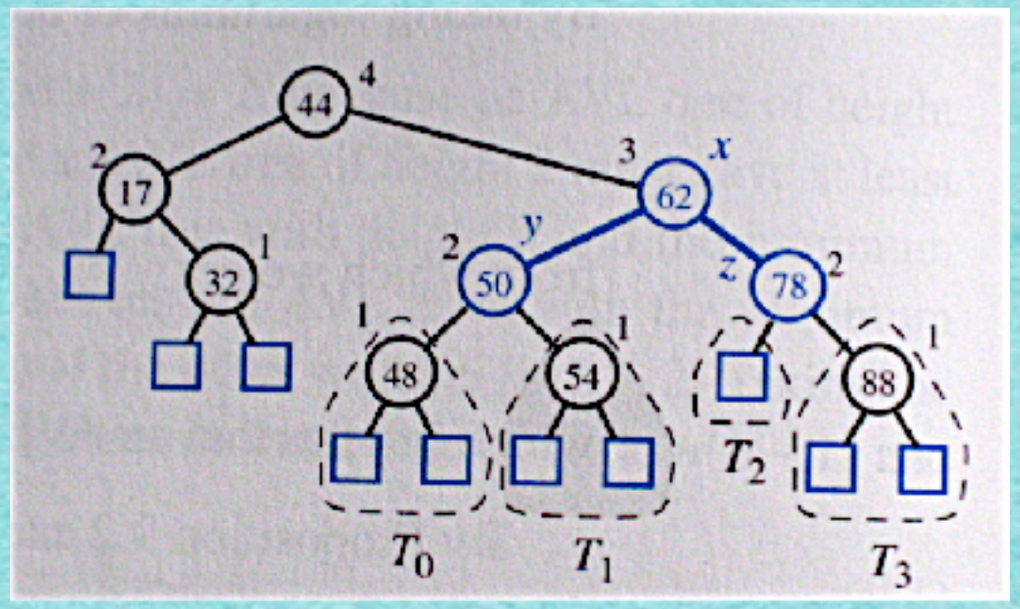
Löschen („DELETE“)

Löschen („DELETE“)



Löschen („DELETE“)

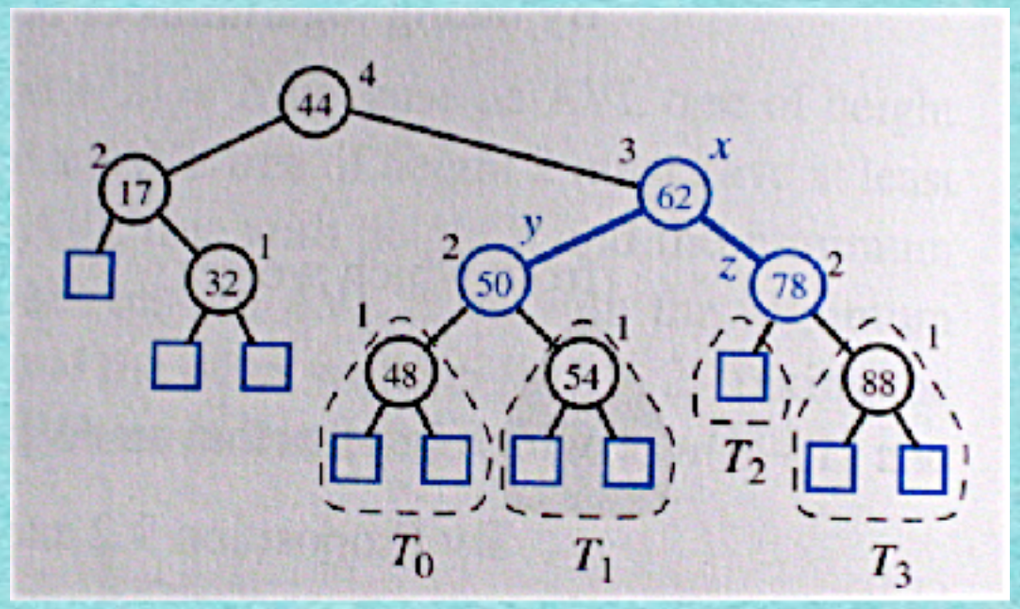
Aufgabe:



Löschen („DELETE“)

<u>Aufgabe:</u>

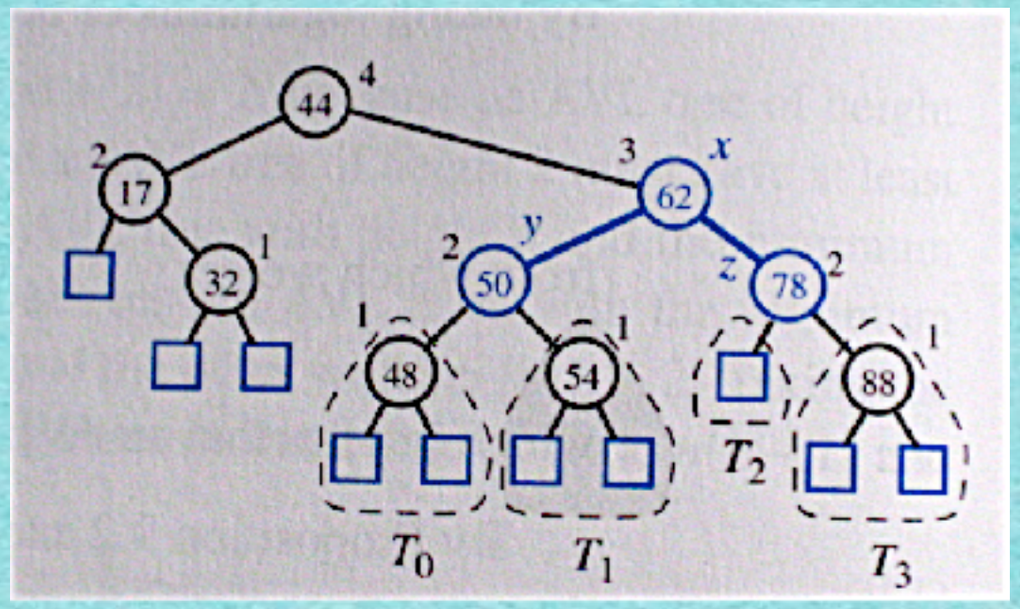
- ***Lösche 32!***

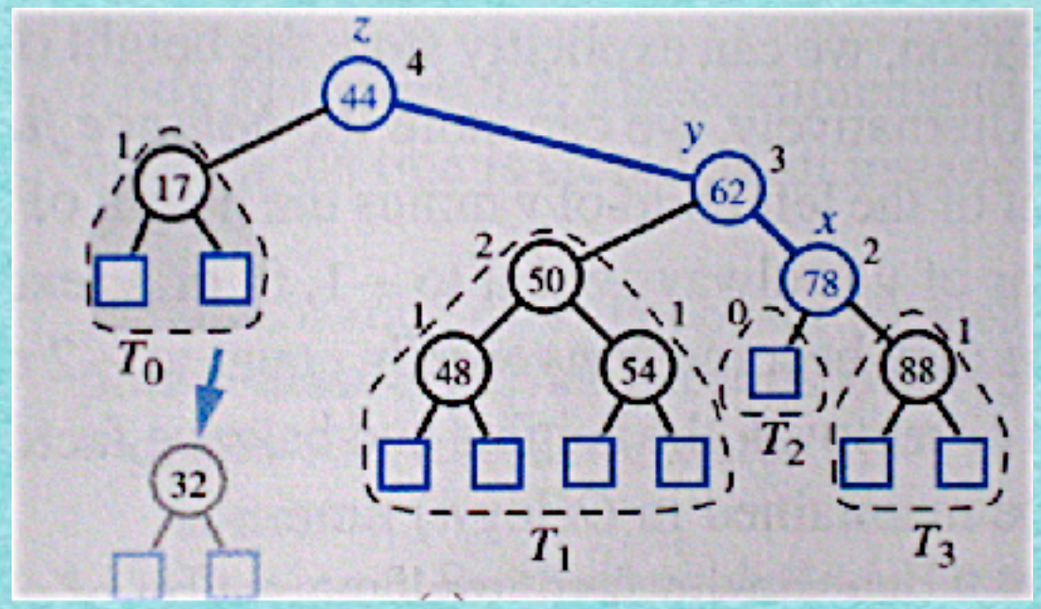


Löschen („DELETE“)

<u>Aufgabe:</u>

- ***Lösche 32!***





Löschen

Löschen

Wieder:

Löschen

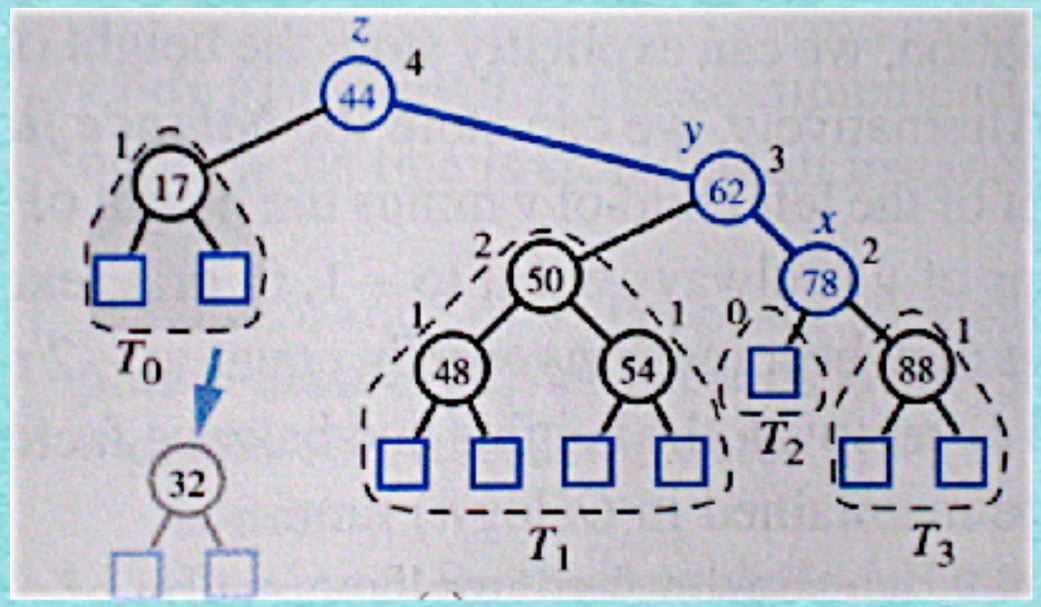
Wieder:

- ***Verwende RESTRUCTURE!***

Löschen

Wieder:

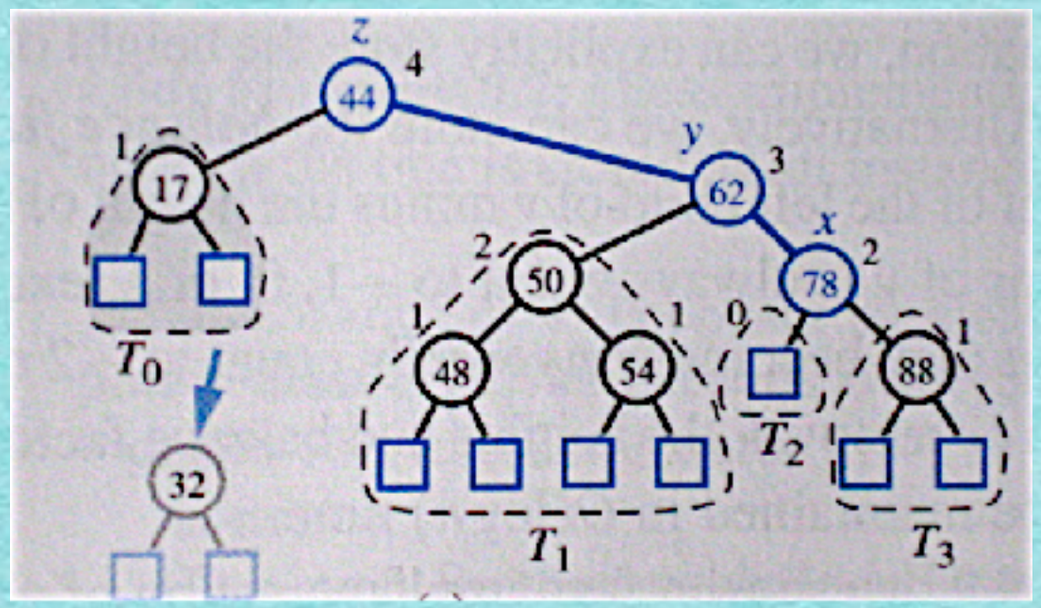
- ***Verwende RESTRUCTURE!***

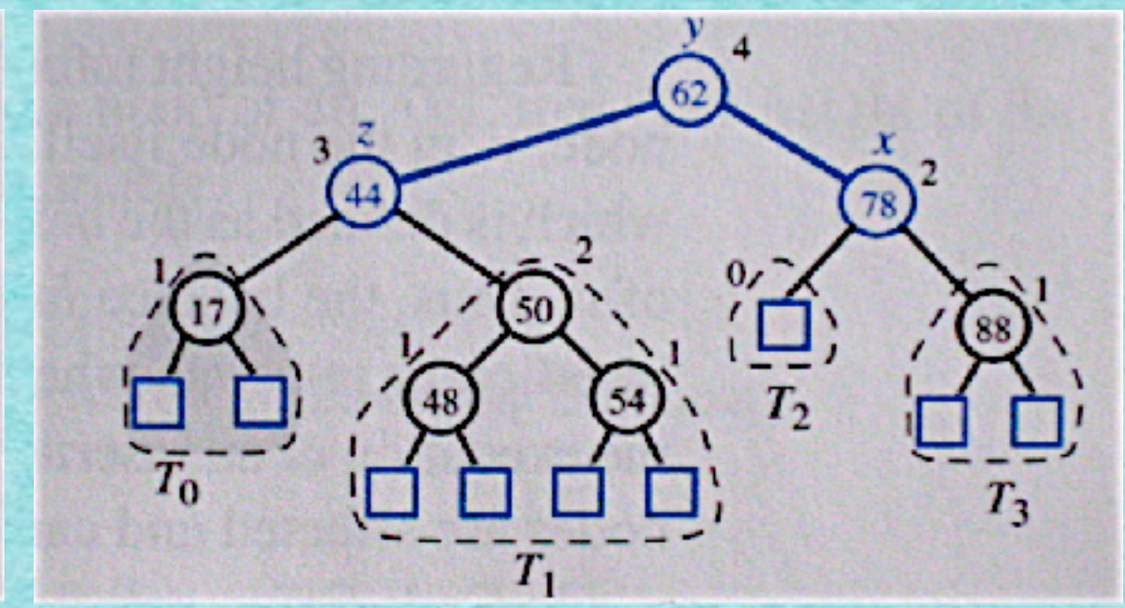


Löschen

Wieder:

- ***Verwende RESTRUCTURE!***

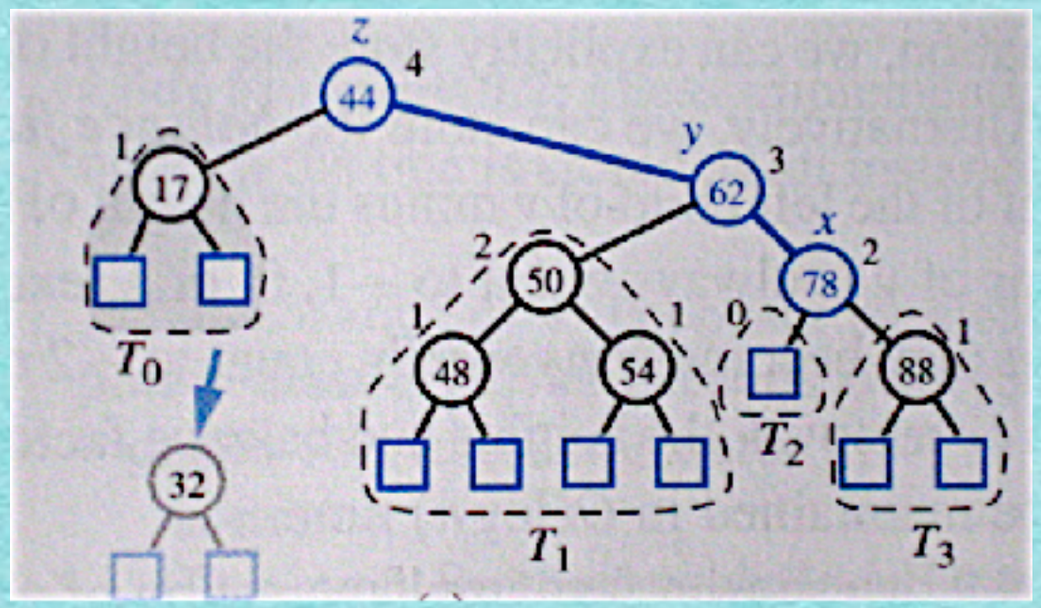


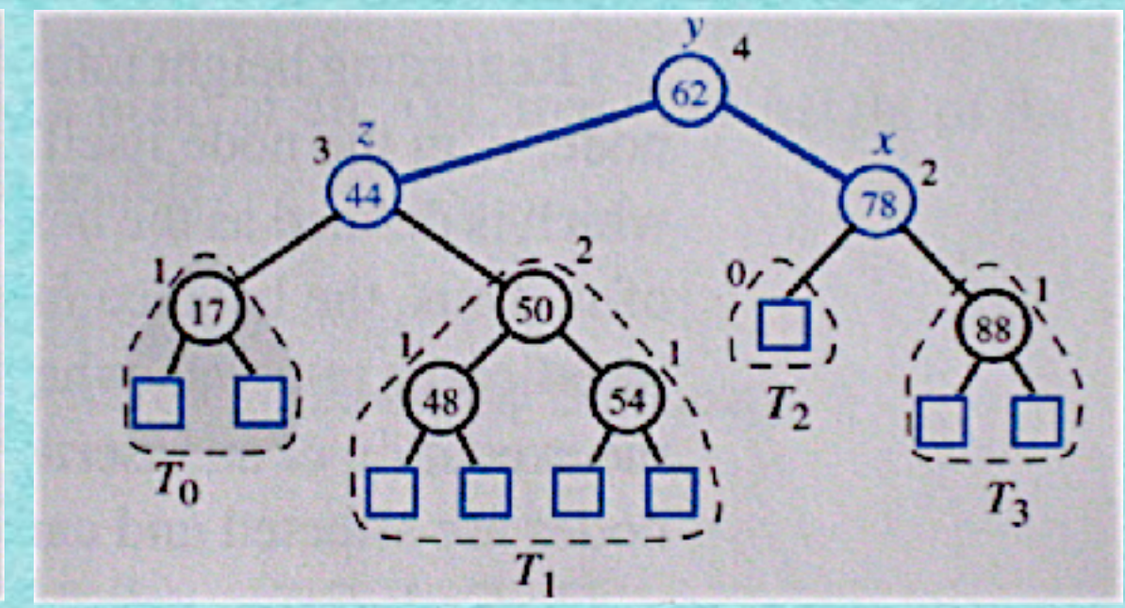


Löschen

Wieder:

- ***Verwende RESTRUCTURE!***



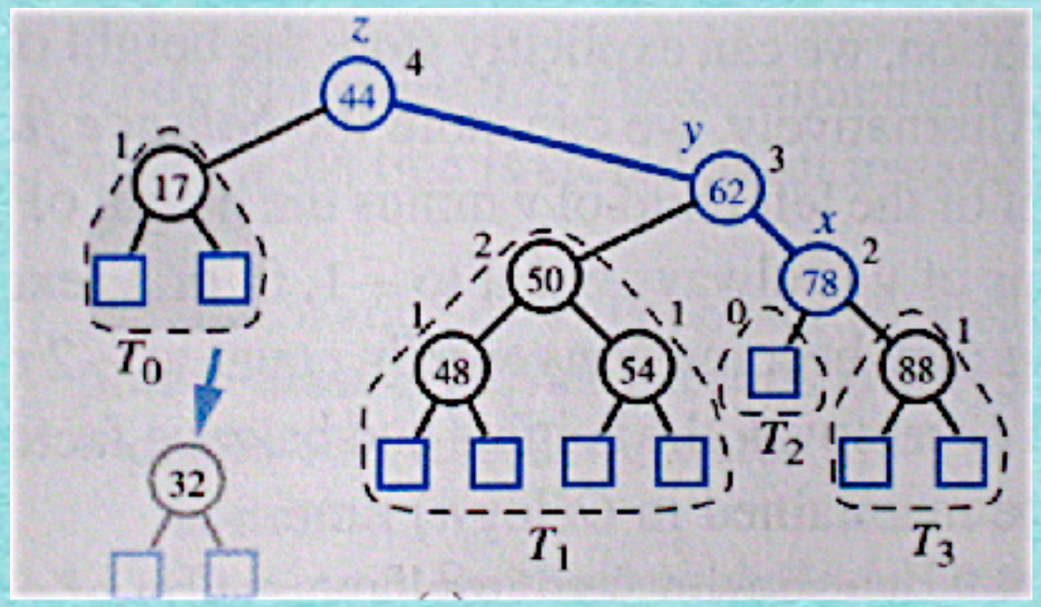


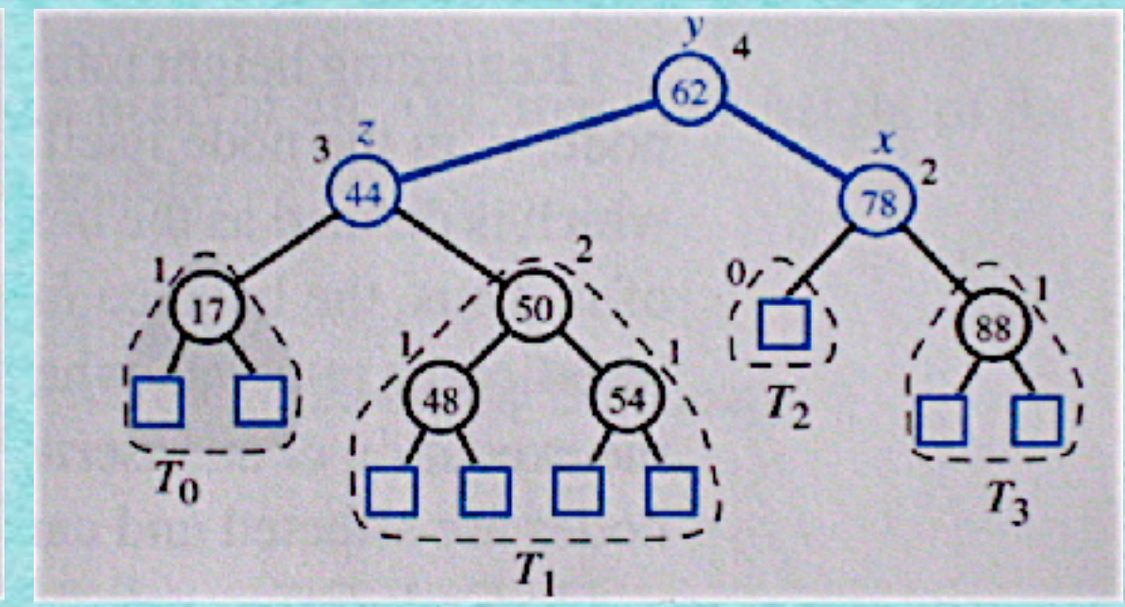
Das reicht in diesem Beispiel!

Löschen

Wieder:

- ***Verwende RESTRUCTURE!***





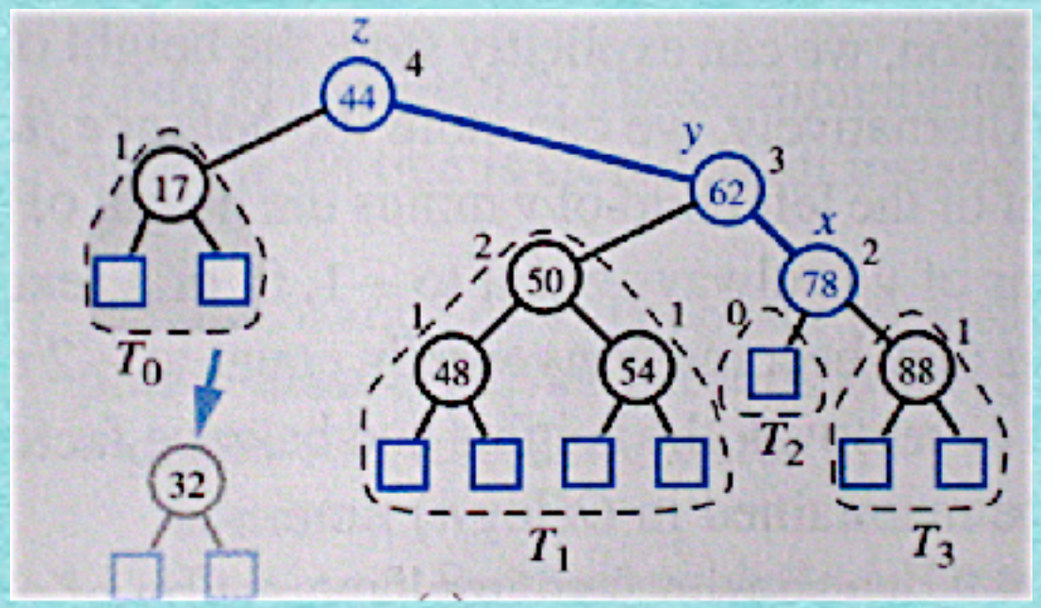
Das reicht in diesem Beispiel!

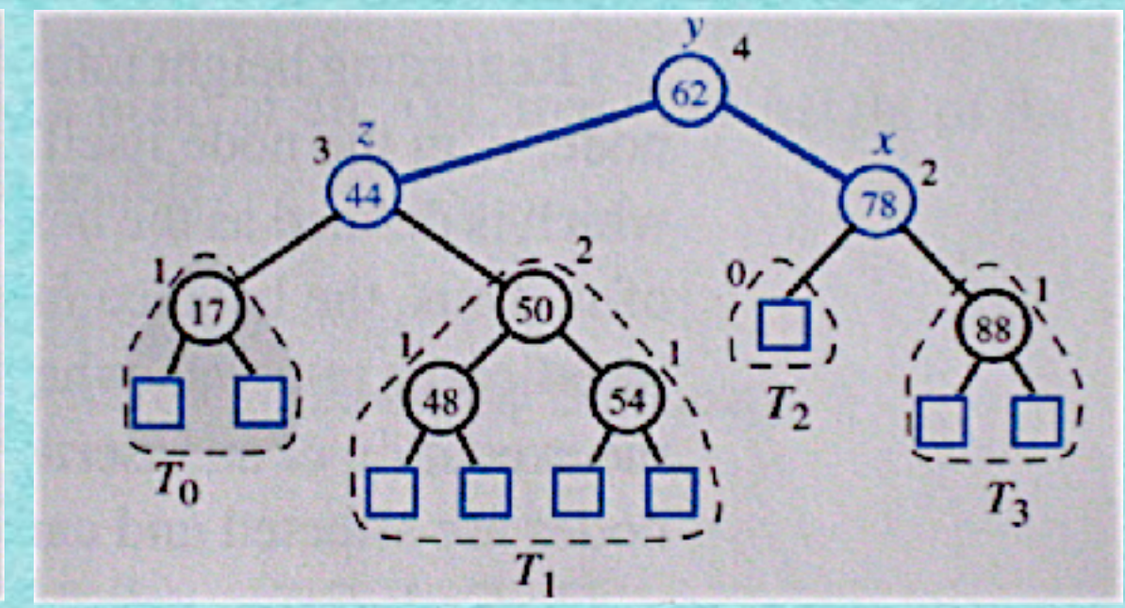
Allerdings kann im allgemeinen ein neues Problem auftauchen:

Löschen

Wieder:

- ***Verwende RESTRUCTURE!***





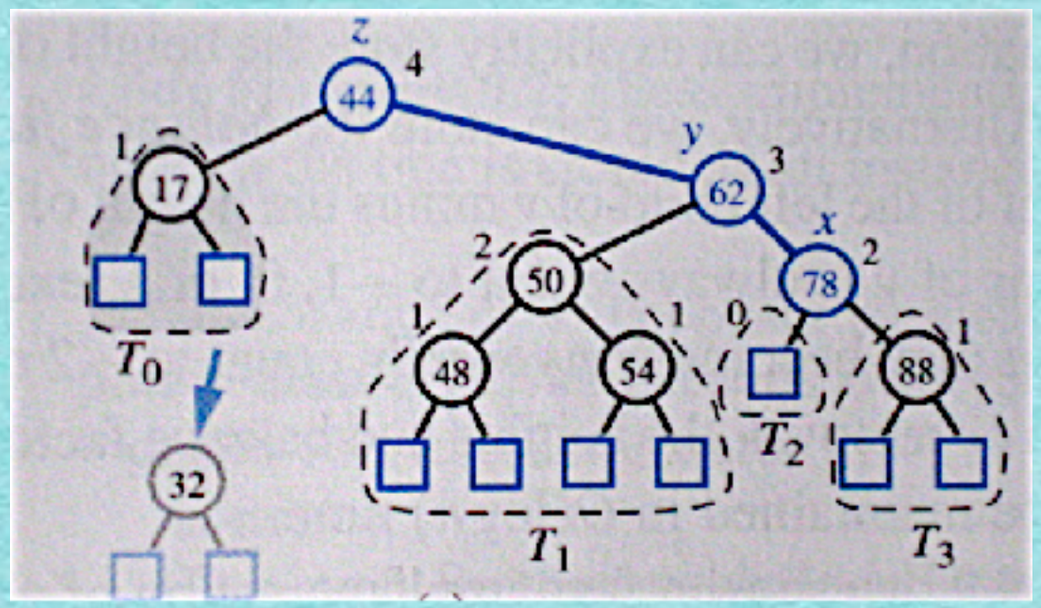
Das reicht in diesem Beispiel!

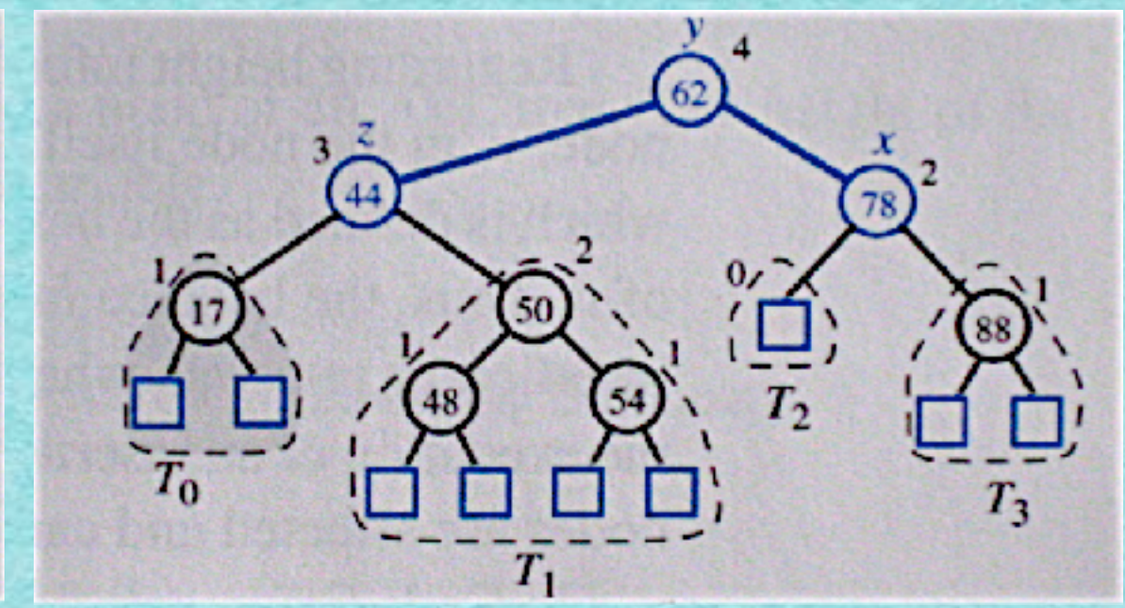
Allerdings kann im allgemeinen ein neues Problem auftauchen: Der neu höhenbalancierte Teilbaum von b kann niedriger sein als

Löschen

Wieder:

- ***Verwende RESTRUCTURE!***





Das reicht in diesem Beispiel!

Allerdings kann im allgemeinen ein neues Problem auftauchen: Der neu höhenbalancierte Teilbaum von b kann niedriger sein als der vorher von z. Dadurch wird eventuell der Vater von z unbalanciert!

Löschen

Löschen

Wieder:

Löschen

Wieder:

- ***Verwende RESTRUCTURE!***

Löschen

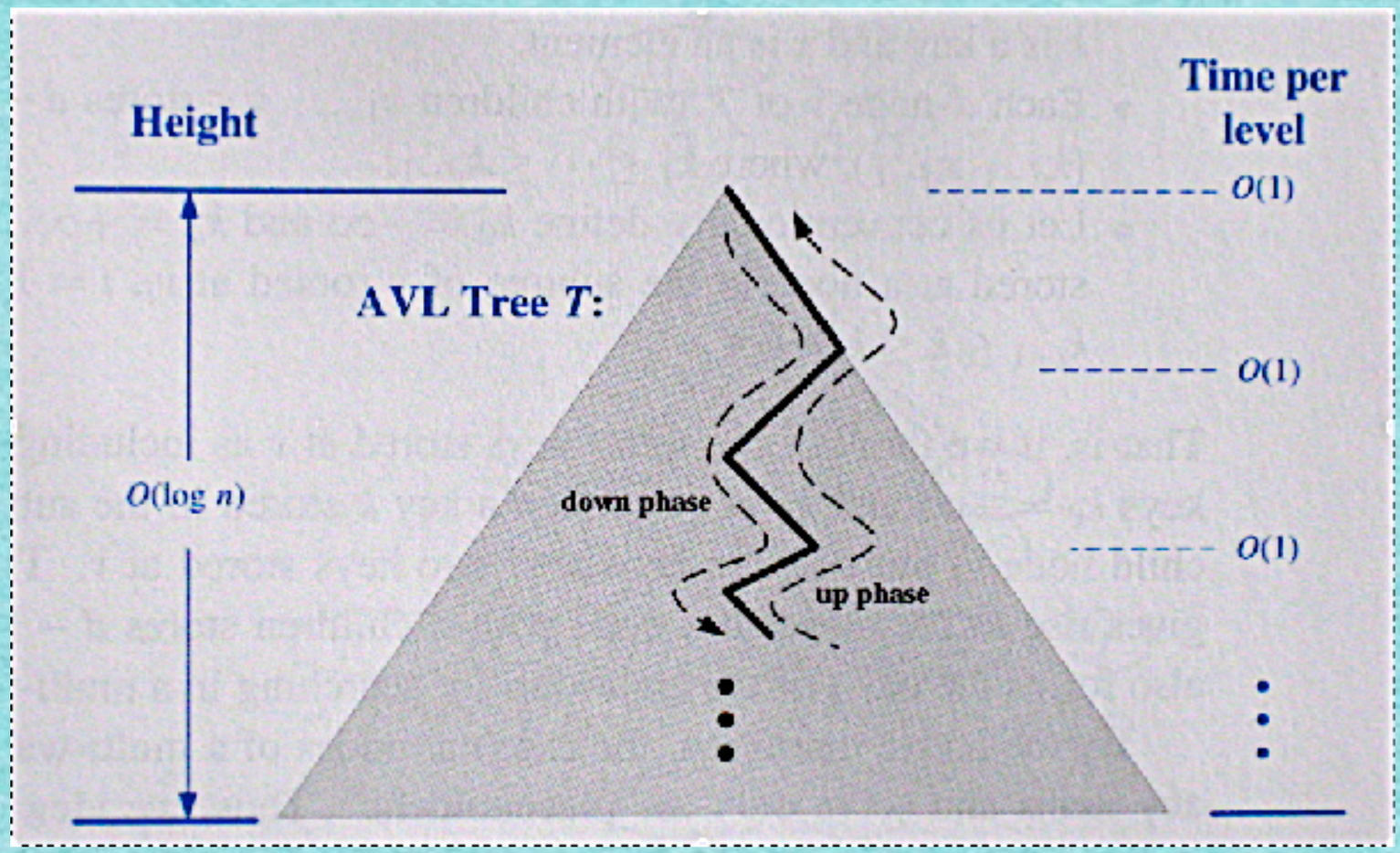
Wieder:

- ***Verwende RESTRUCTURE!***
- ***Ggf. immer wieder…***

Löschen

Wieder:

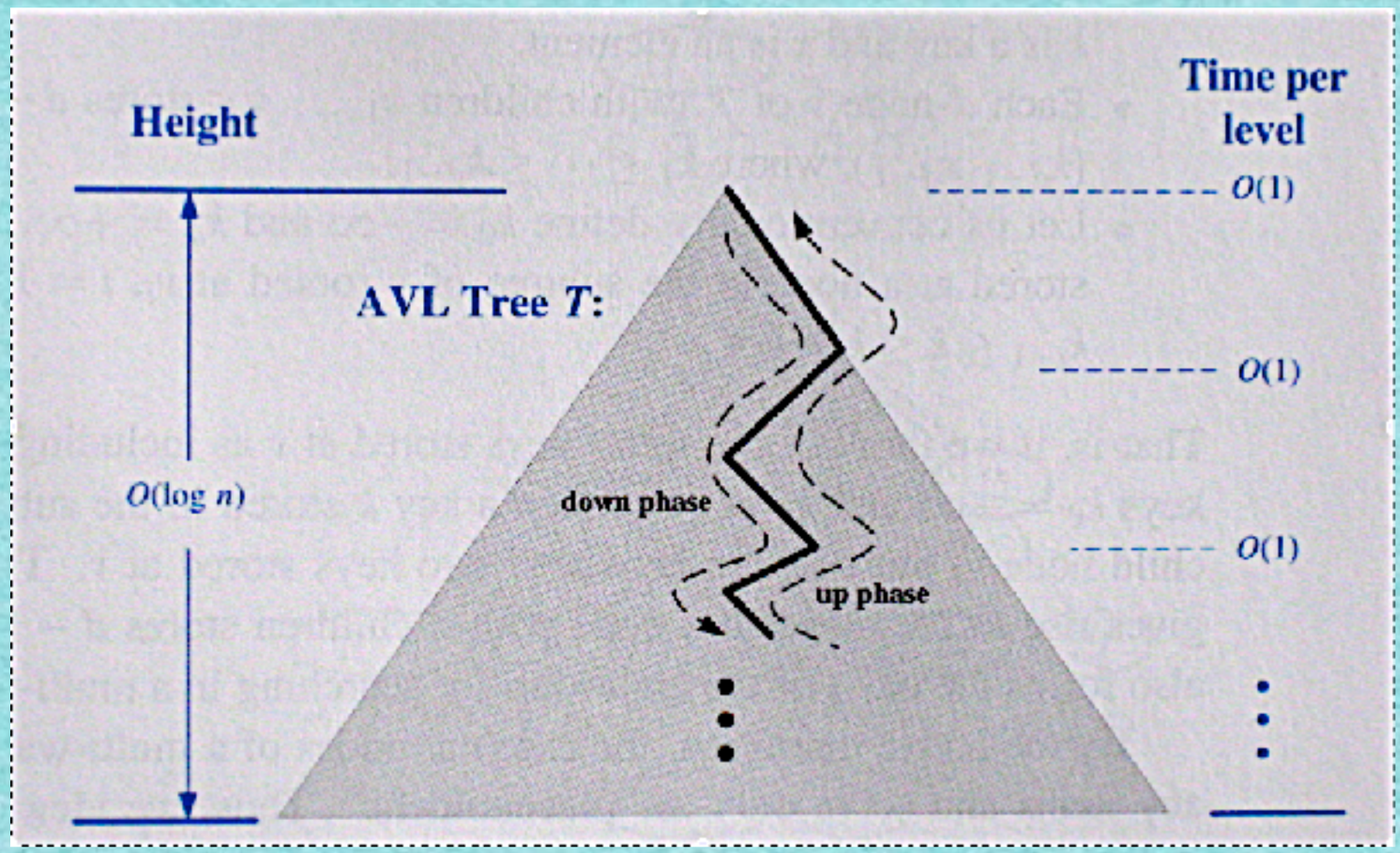
- ***Verwende RESTRUCTURE!***
- ***Ggf. immer wieder…***



Löschen

Wieder:

- ***Verwende RESTRUCTURE!***
- ***Ggf. immer wieder…***



Höchstens O(log n) RESTRUCTURE-Operationen, jeweils in O(1)!

Löschen

Löschen

Satz 4.11

Löschen

Satz 4.11

Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Lösch-Operation höhenbalanciert halten.

Löschen

Satz 4.11

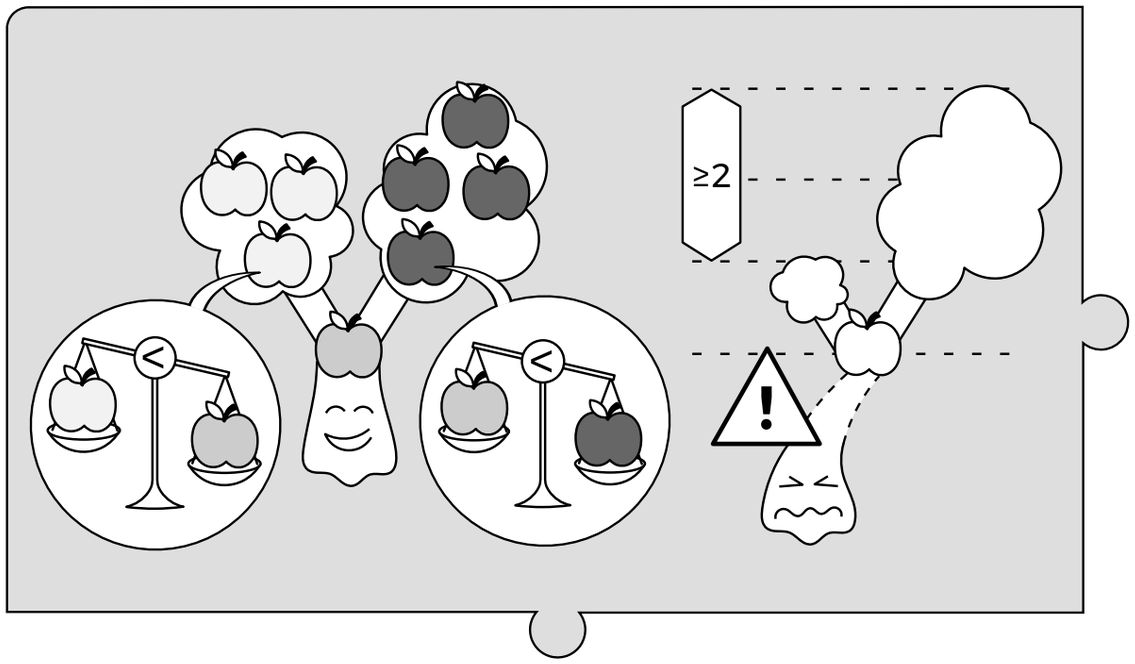
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Lösch-Operation höhenbalanciert halten.
Die Zeit dafür ist O(log n).

BÄLÄNCE TREE 1/2

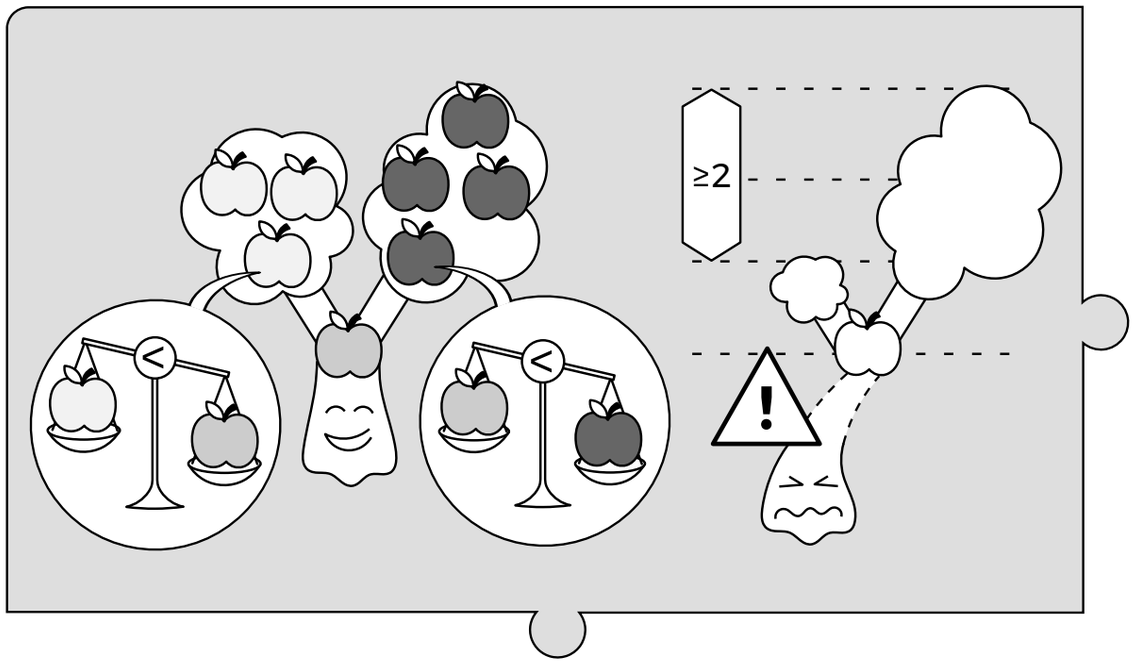
BÄLÄNCE TREE 1/2

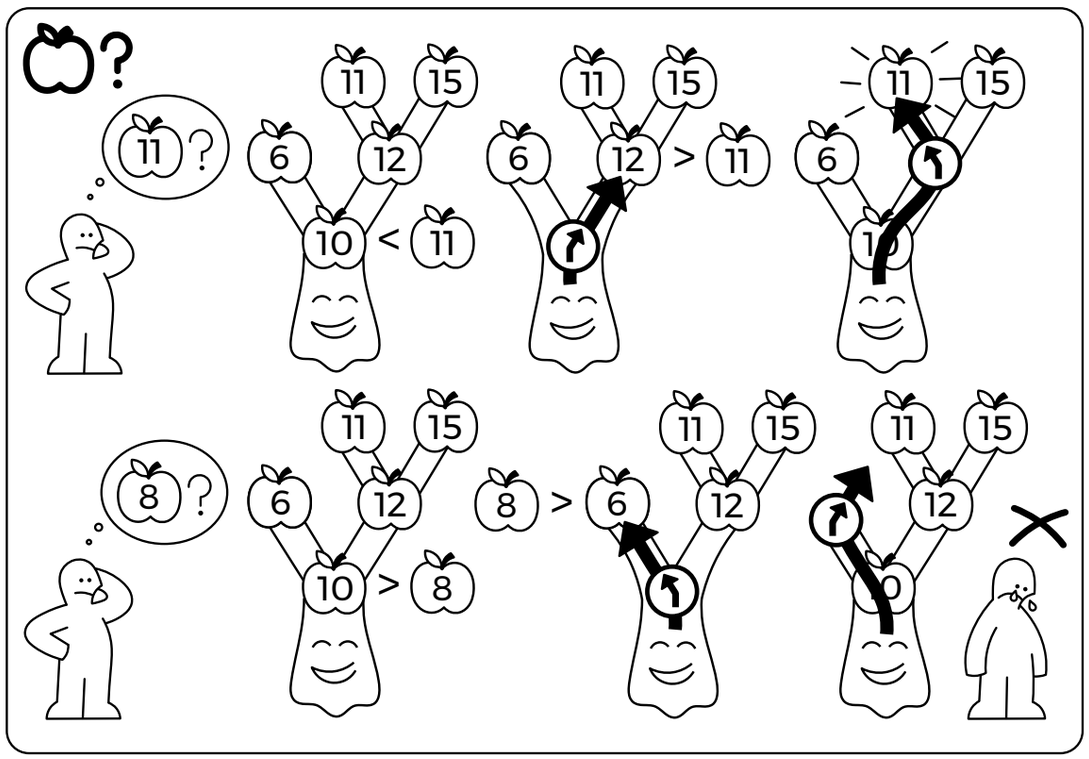
idea-instructions.com/avl-tree/
v1.0, CC by-nc-sa 4.0

IDEA



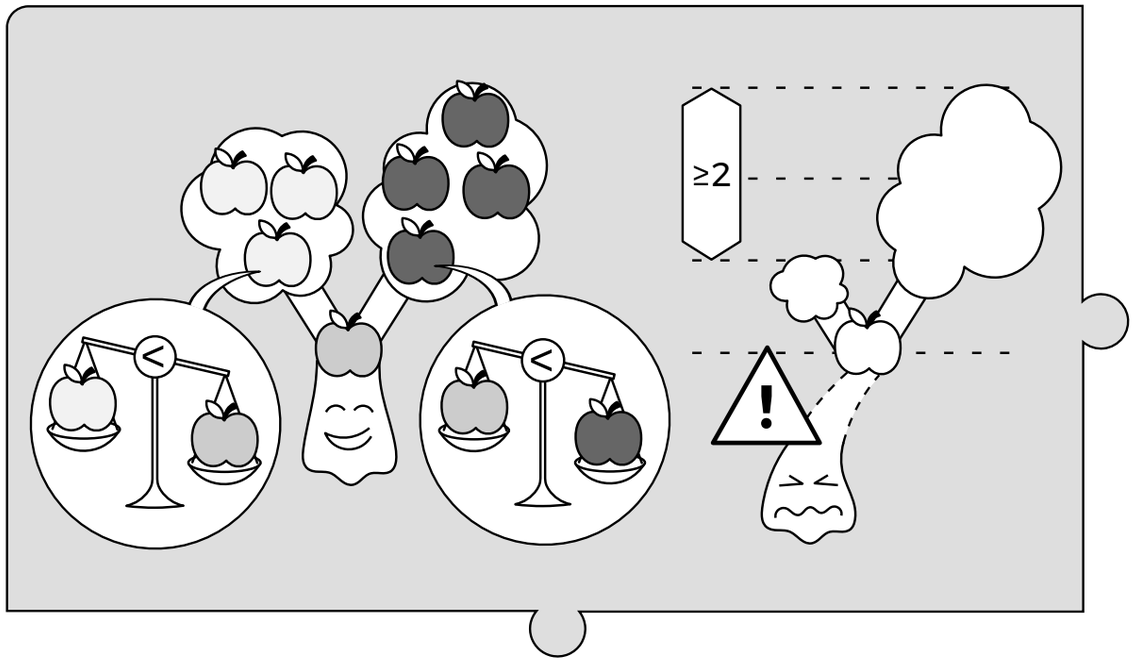
BÄLÄNCE TREE 1/2

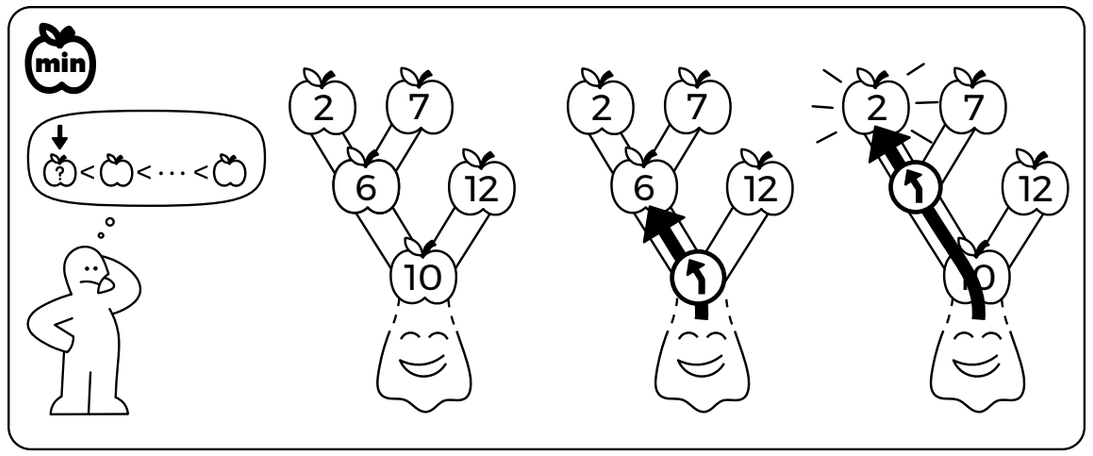


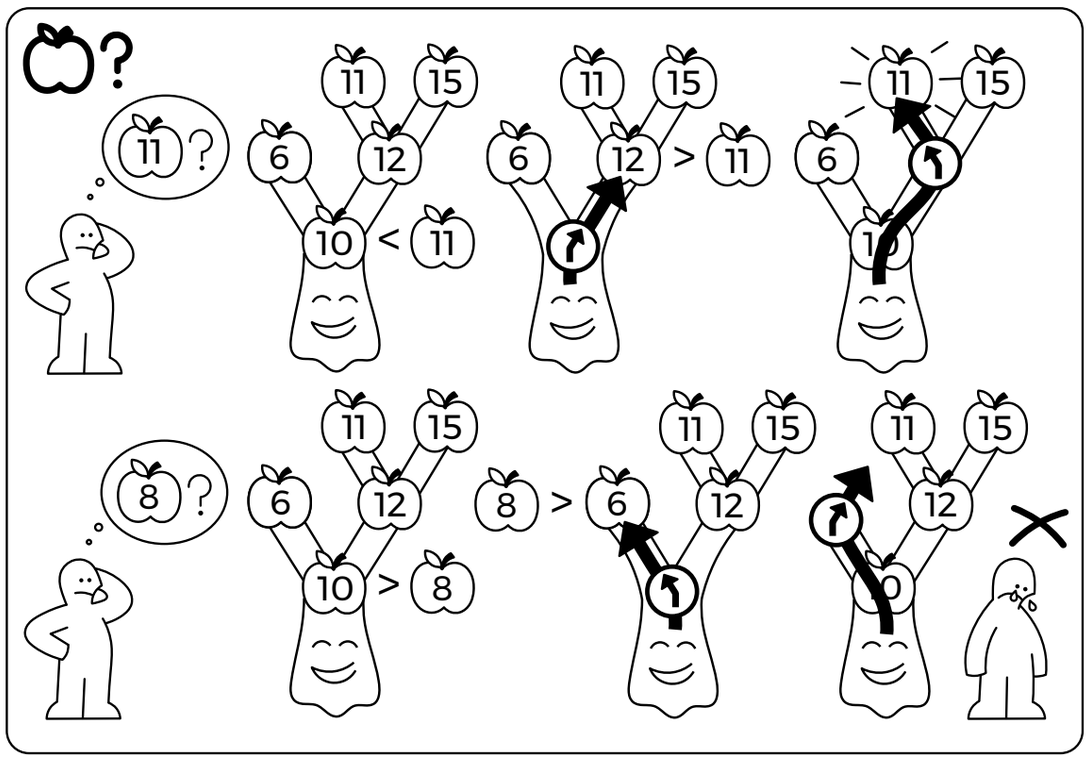


BÄLÄNCE TREE 1/2

idea-instructions.com/avl-tree/
v1.0, CC by-nc-sa 4.0

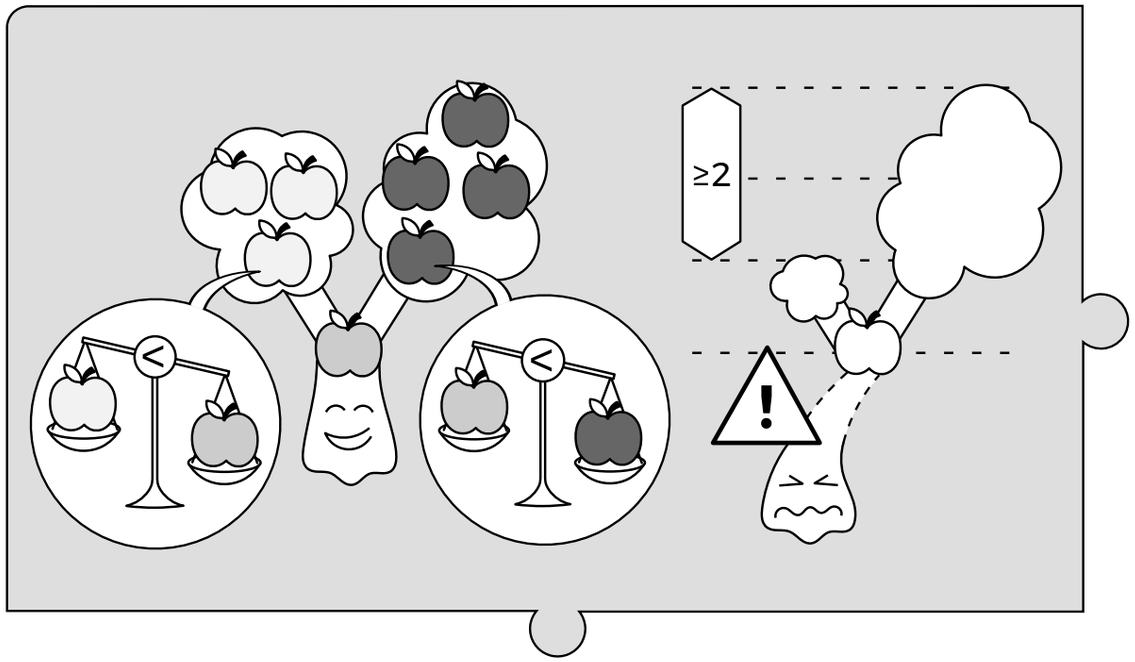


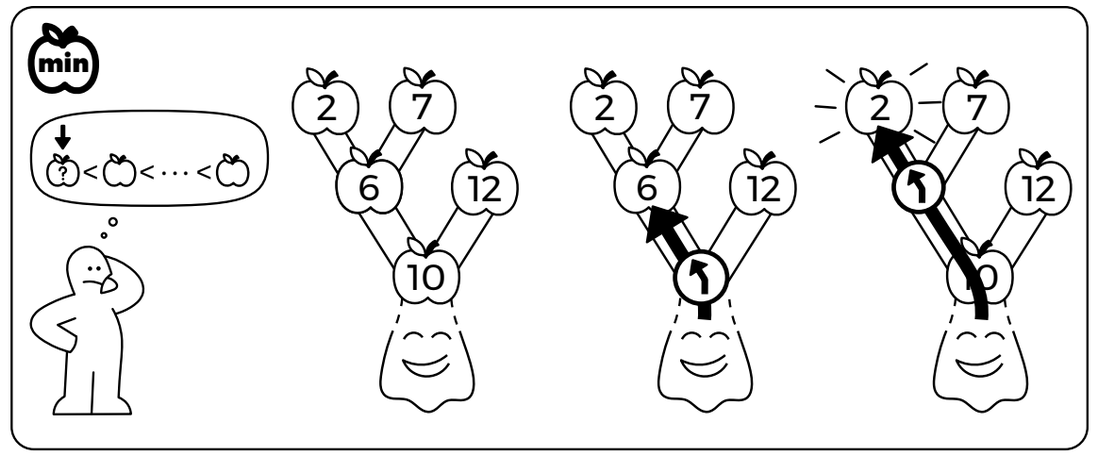


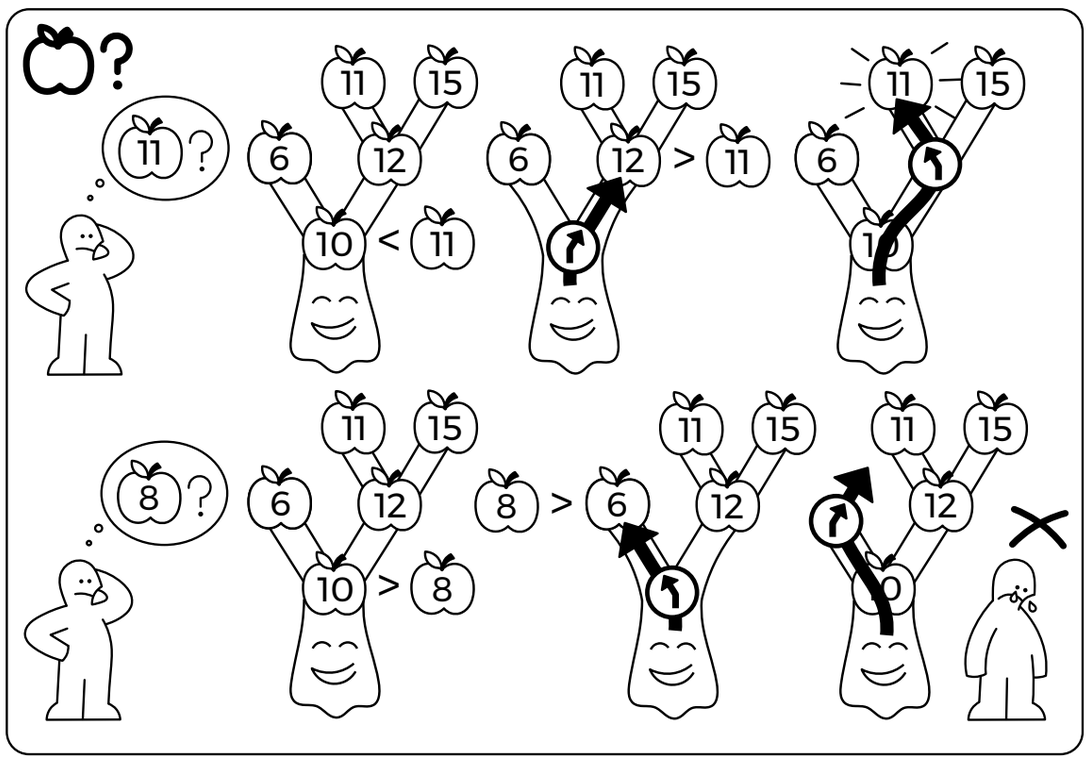


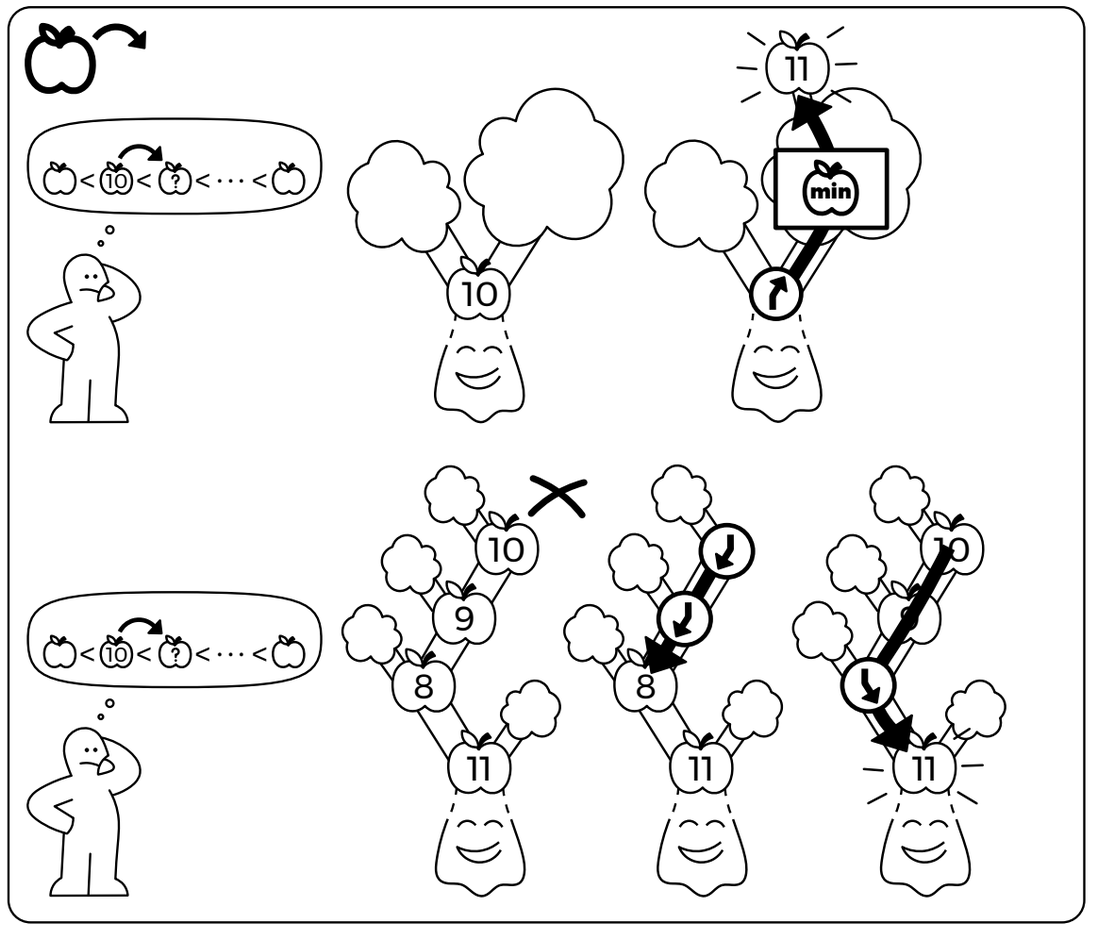
BÄLÄNCE TREE 1/2

idea-instructions.com/avl-tree/
v1.0, CC by-nc-sa 4.0



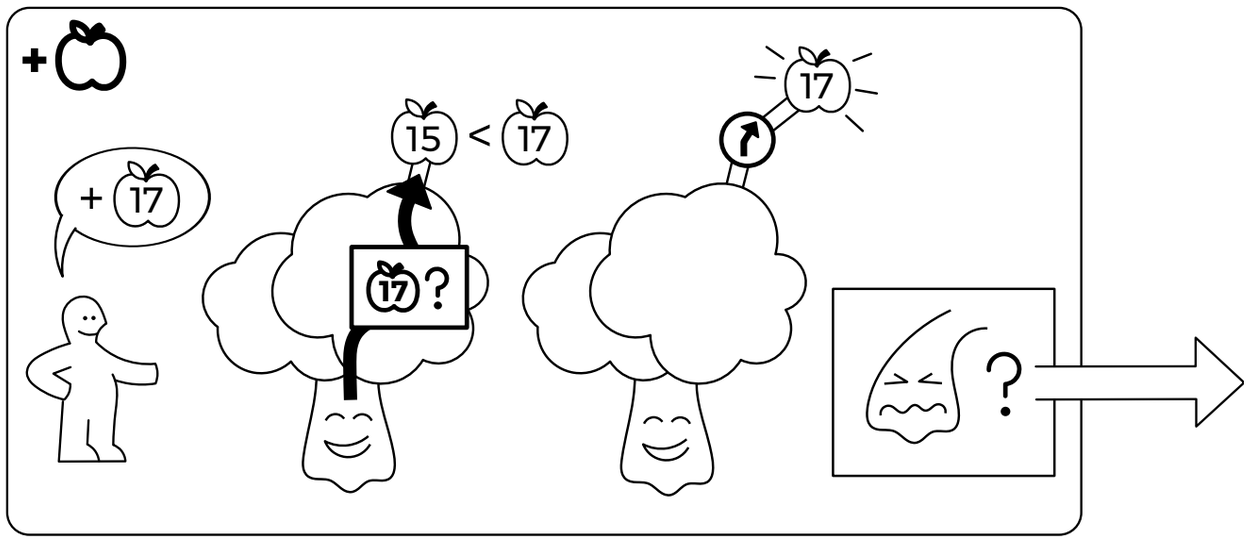






BÄLÄNCE TREE 2/2

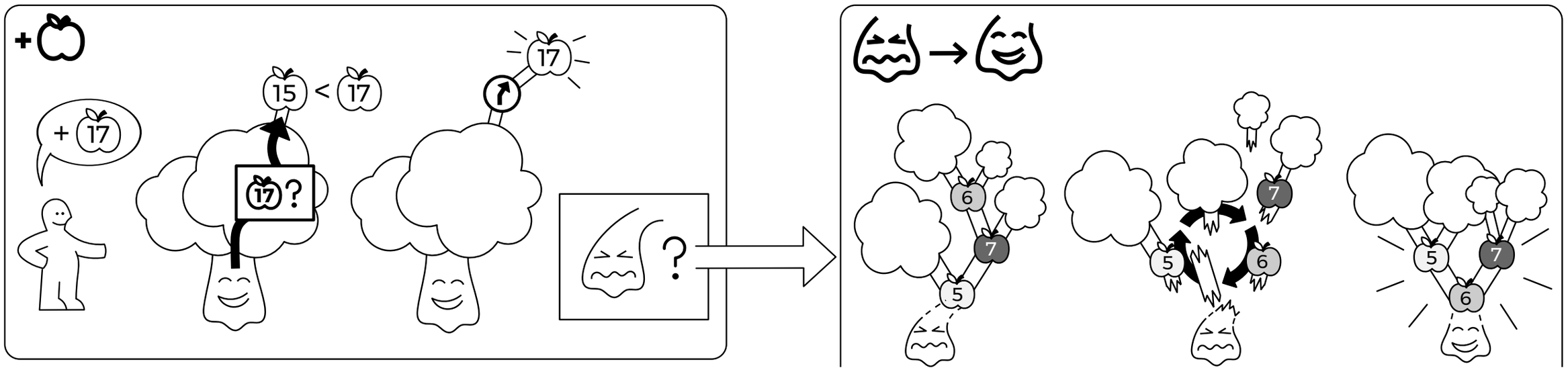
BÄLÄNCE TREE 2/2



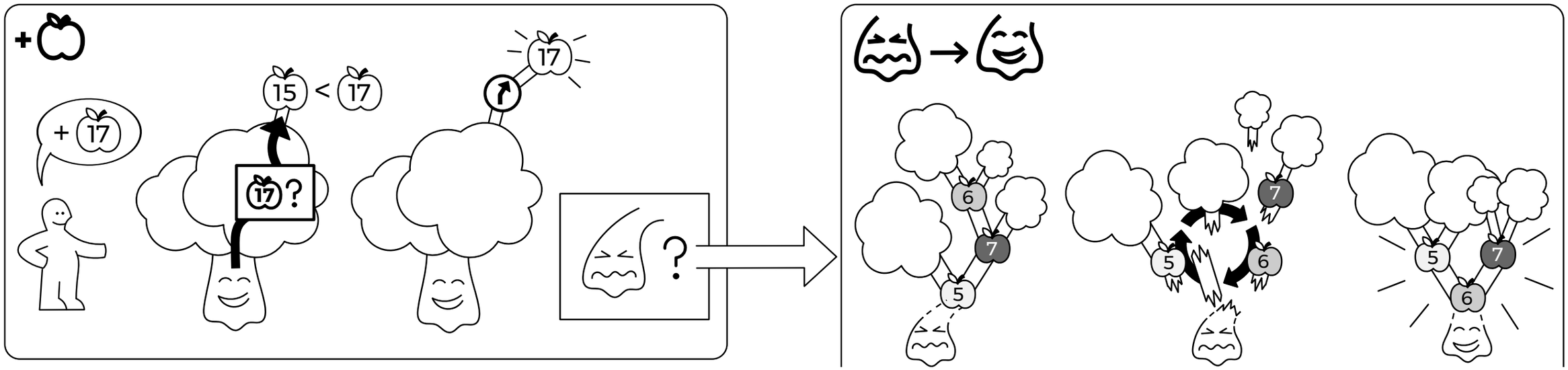
BÄLÄNCE TREE 2/2

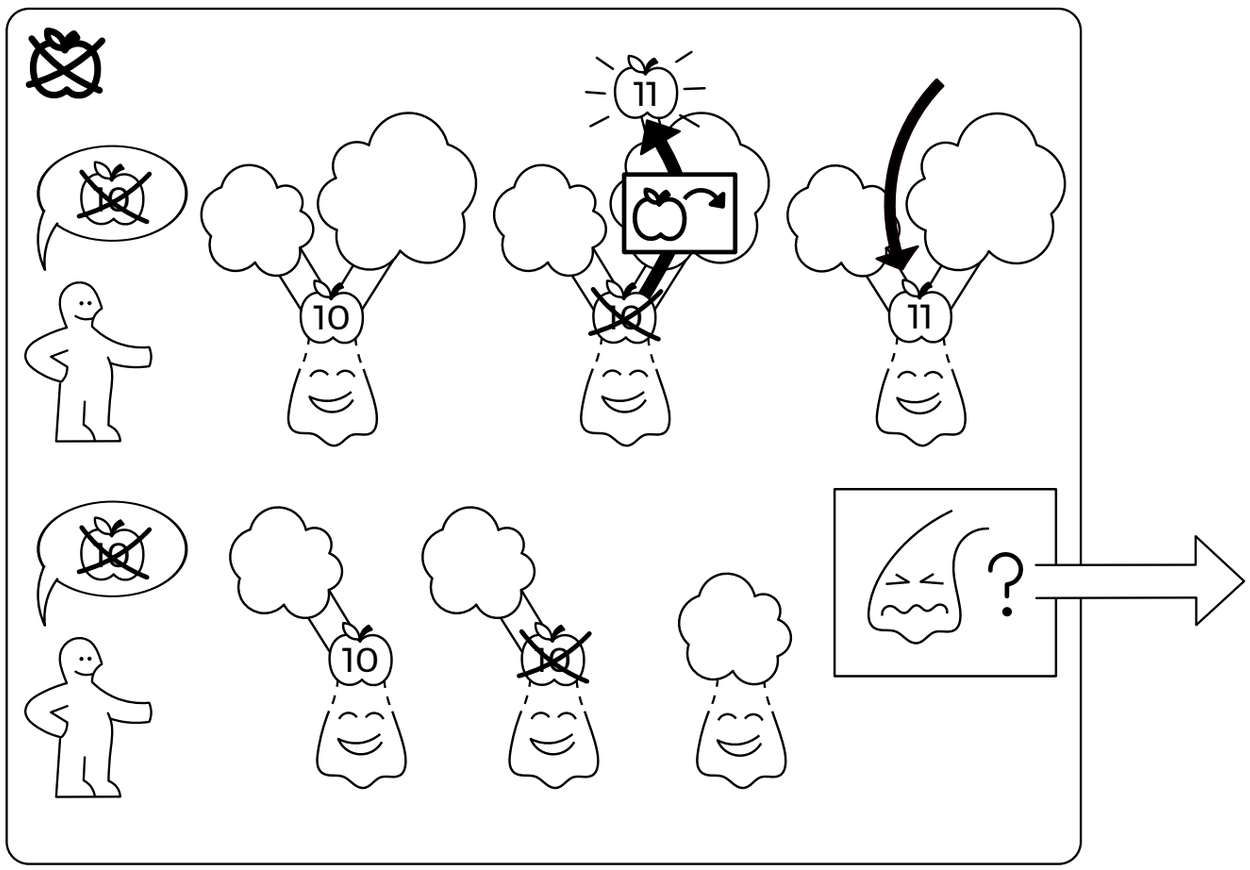
idea-instructions.com/avl-tree/
v1.0, CC by-nc-sa 4.0

IDEA



BÄLÄNCE TREE 2/2

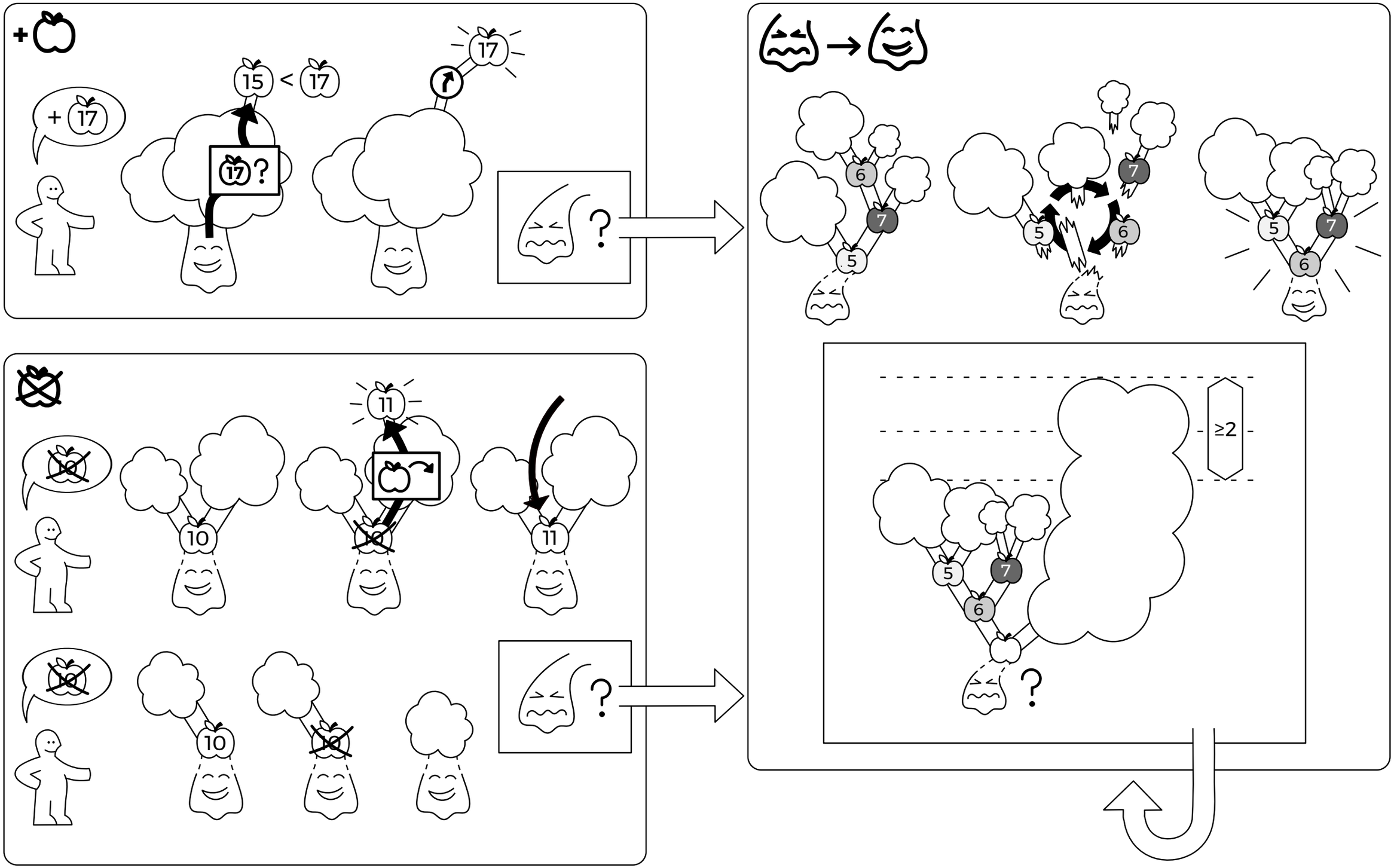




BÄLÄNCE TREE 2/2

idea-instructions.com/avl-tree/
v1.0, CC by-nc-sa 4.0

IDEA



QUIZ!

4.7 Fibonacci-Zahlen

4.7 Fibonacci-Zahlen

4.7 Fibonacci-Zahlen

Leonardo da Pisa,
gen. Fibonacci
1180-1241

4.7 Fibonacci-Zahlen

Leonardo da Pisa,
gen. Fibonacci
1180-1241

Fibonacci-Zahlen

$$F(n)=F(n-1)+F(n-2)$$

Fibonacci-Zahlen

$$F(n)=F(n-1)+F(n-2)$$

Fibonacci-Zahlen

$$F(n)=F(n-1)+F(n-2)$$

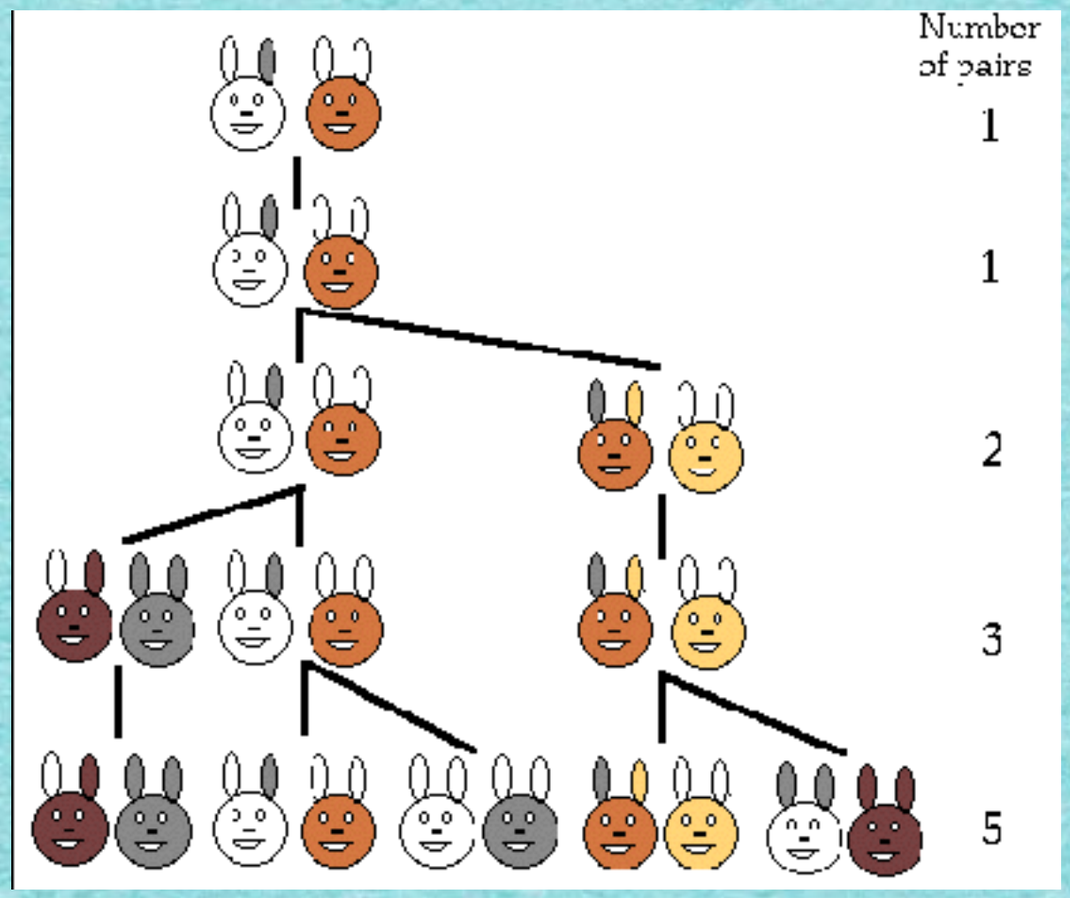
Fibonacci-Zahlen

$$F(n)=F(n-1)+F(n-2)$$

Fibonacci-Zahlen

$$F(n)=F(n-1)+F(n-2)$$

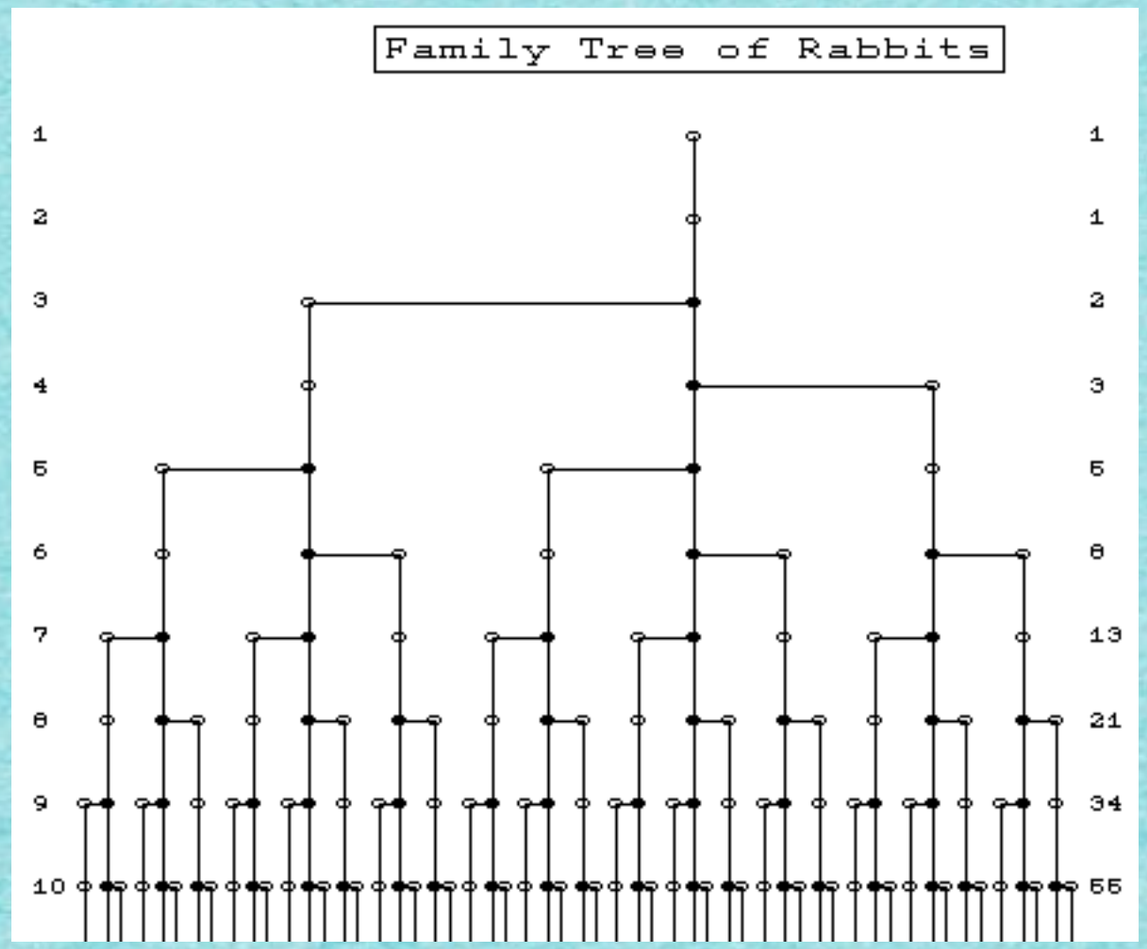
Fibonacci-Zahlen



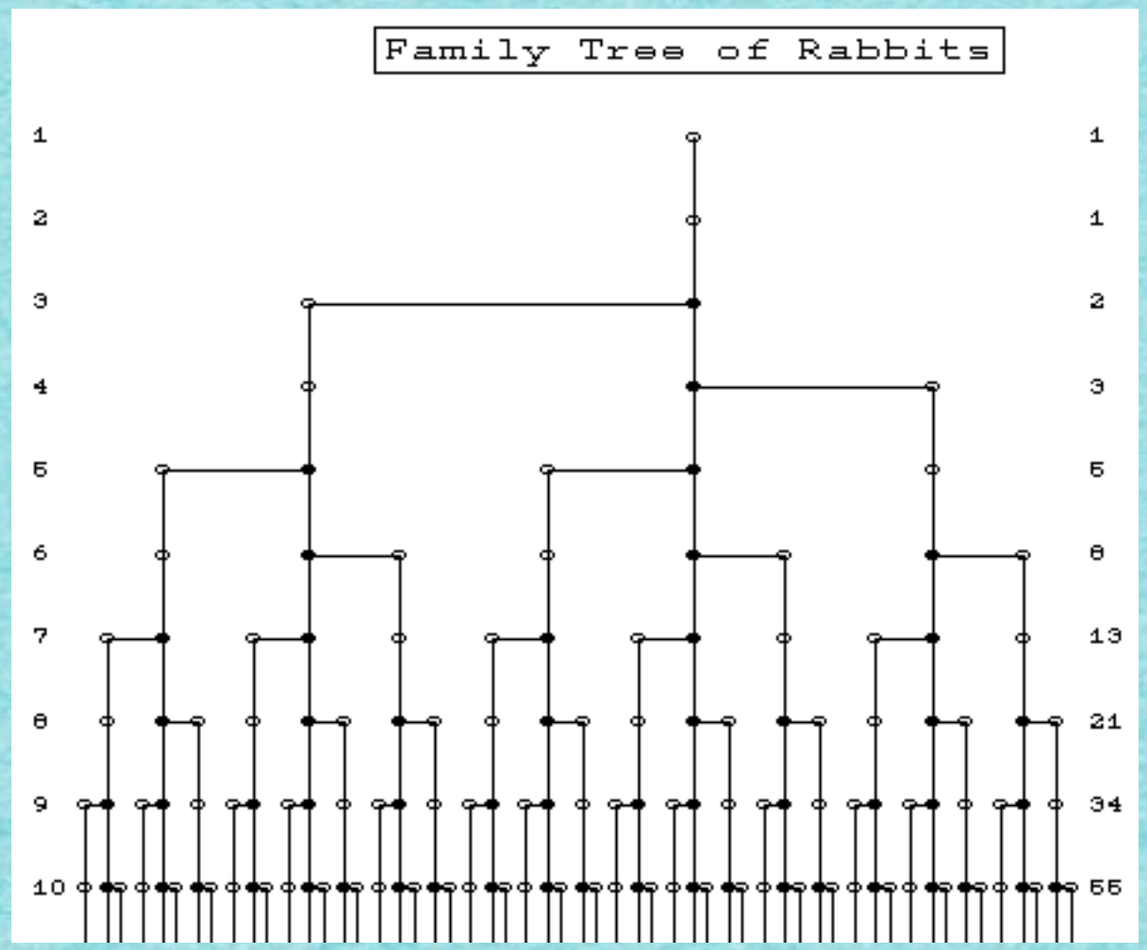
$$F(n)=F(n-1)+F(n-2)$$

Fibonacci-Zahlen

Fibonacci-Zahlen



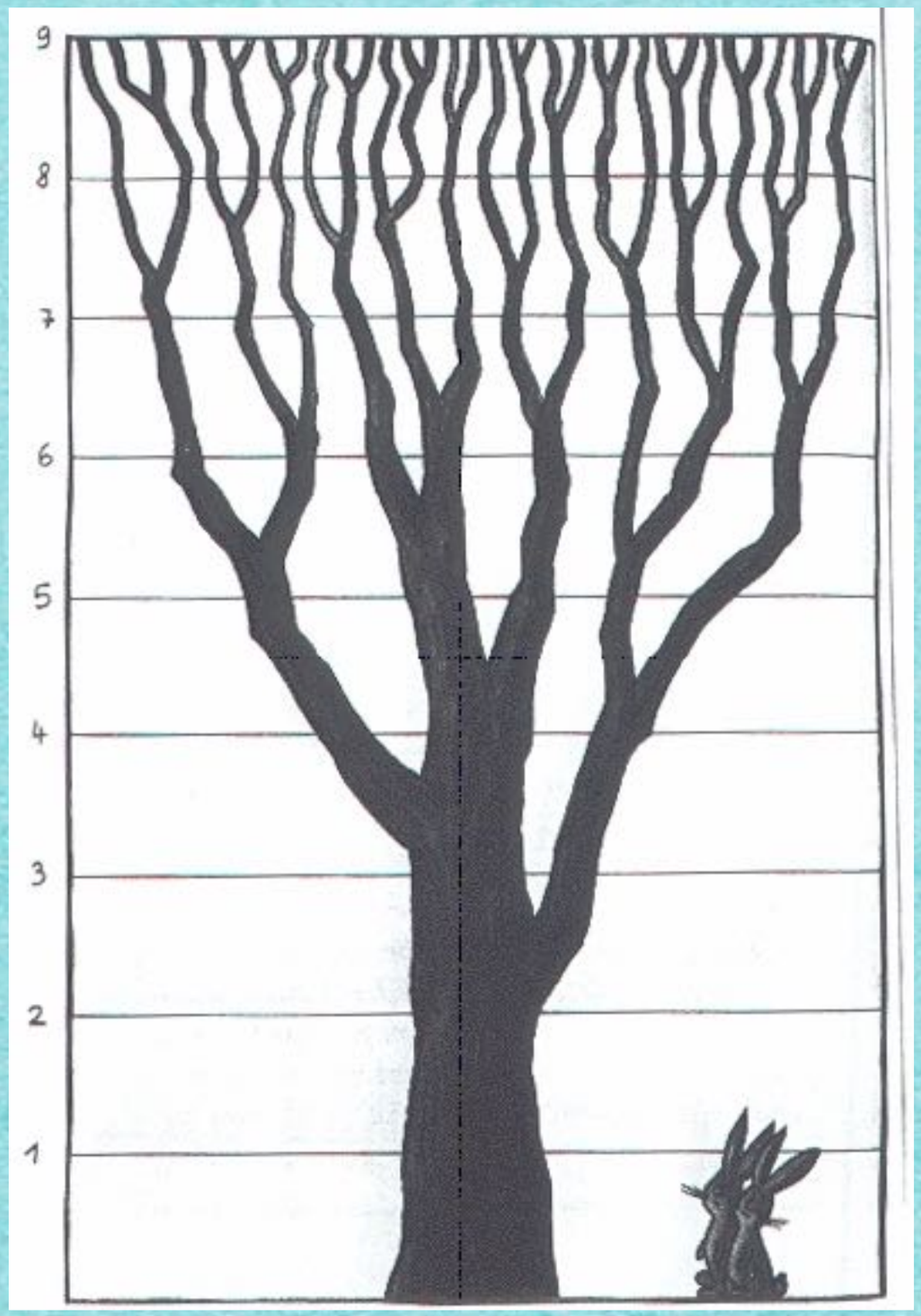
Fibonacci-Zahlen



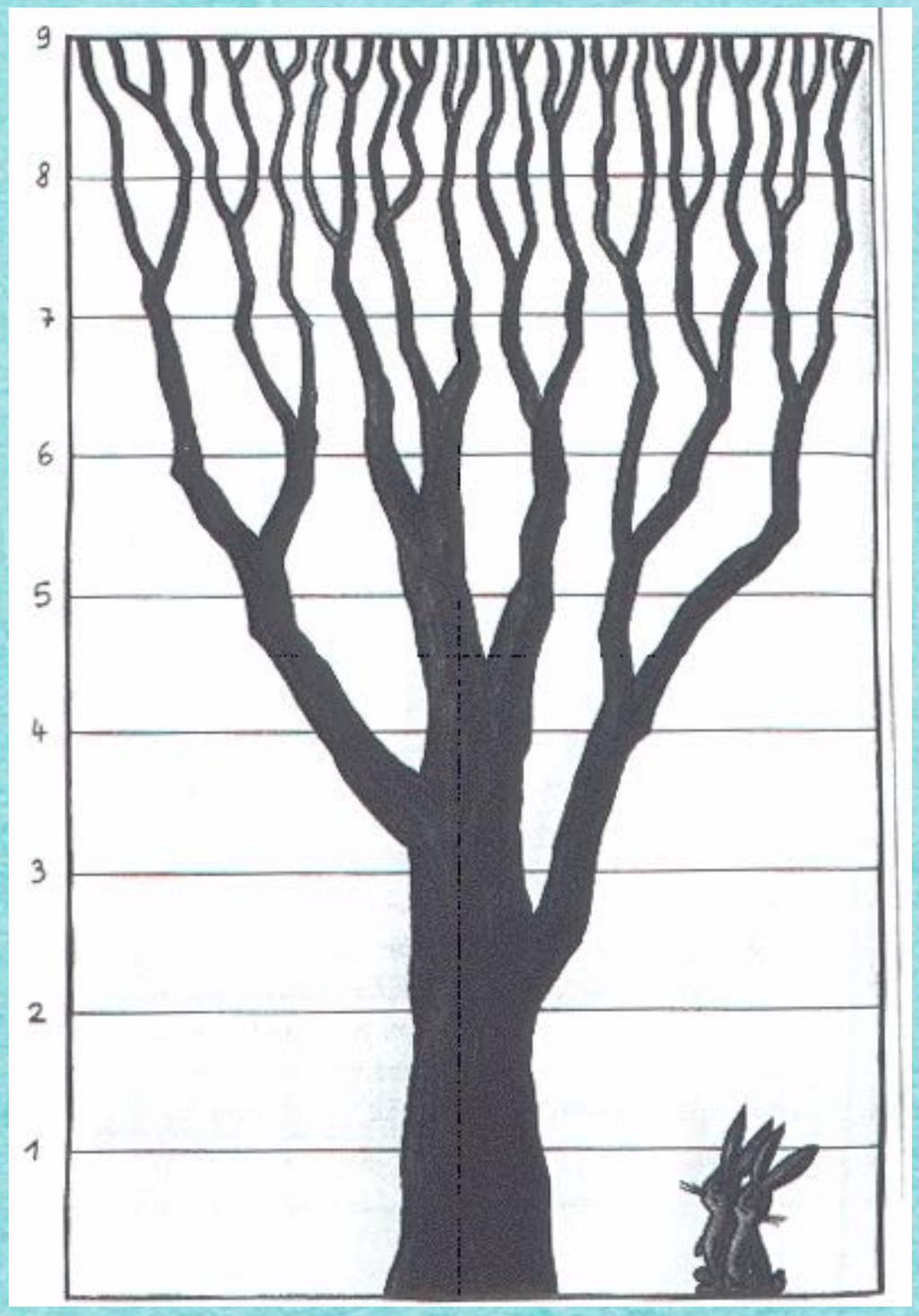
$$F(n)=F(n-1)+F(n-2)$$

Fibonacci-Zahlen

Fibonacci-Zahlen

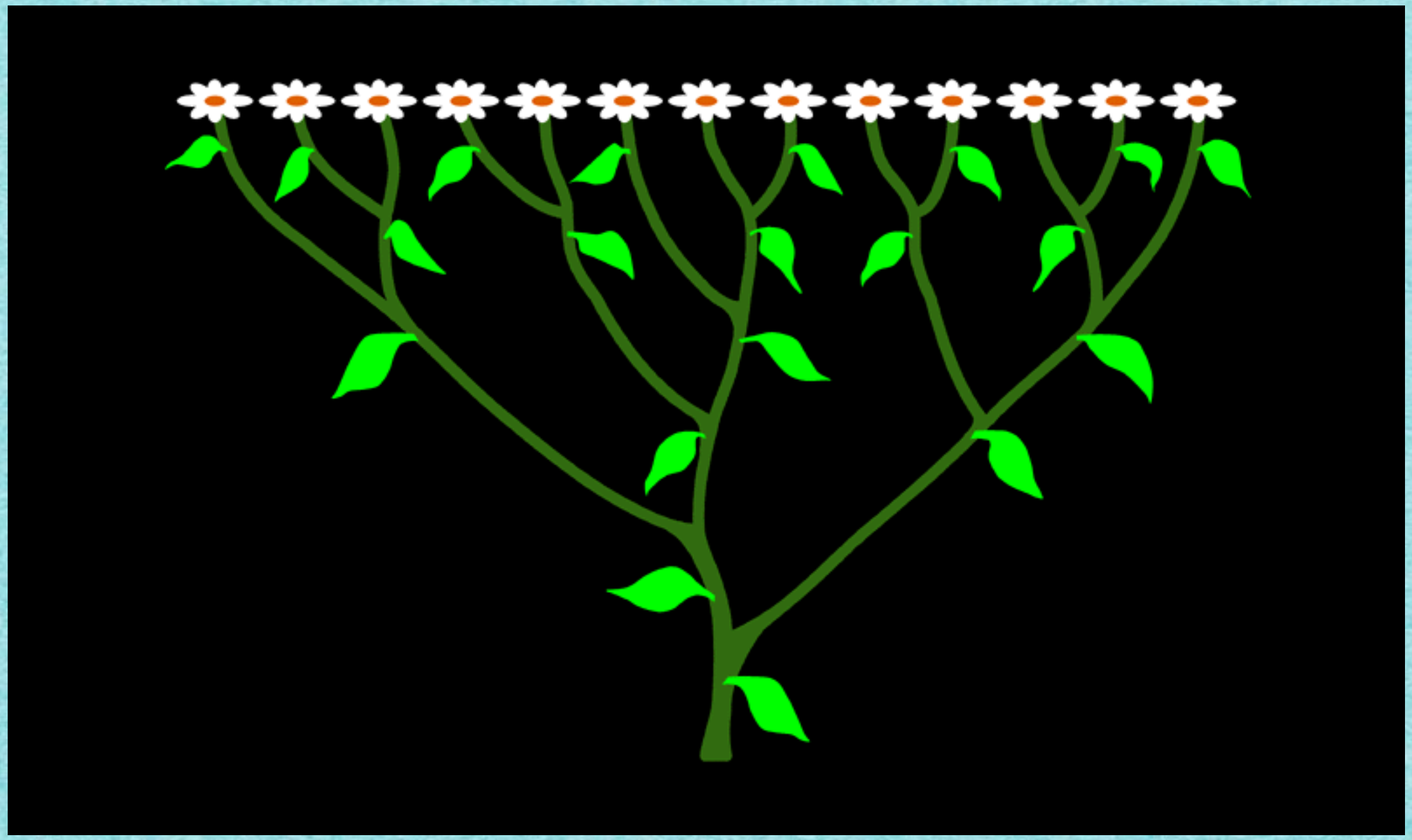


Fibonacci-Zahlen

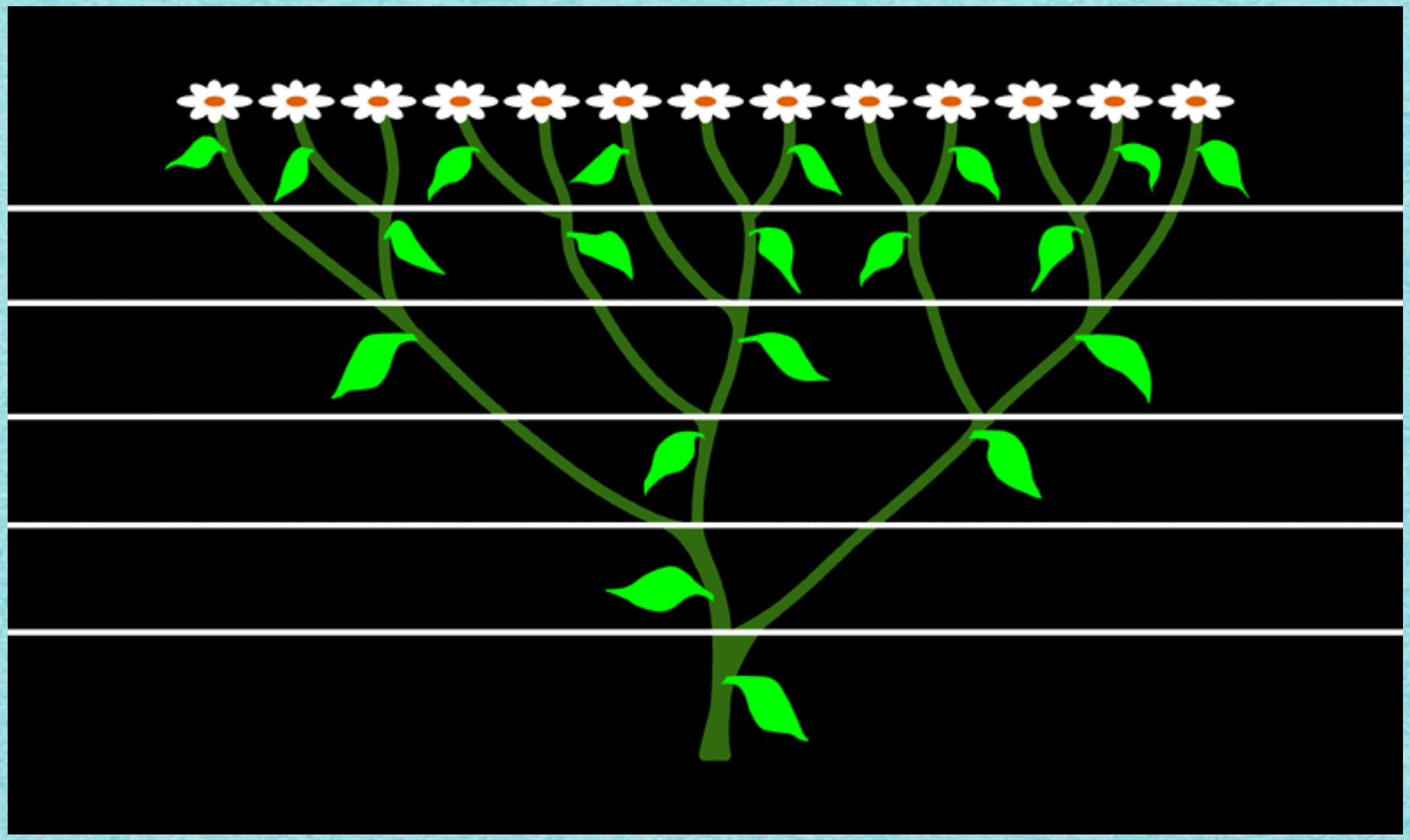


Fibonacci-Zahlen

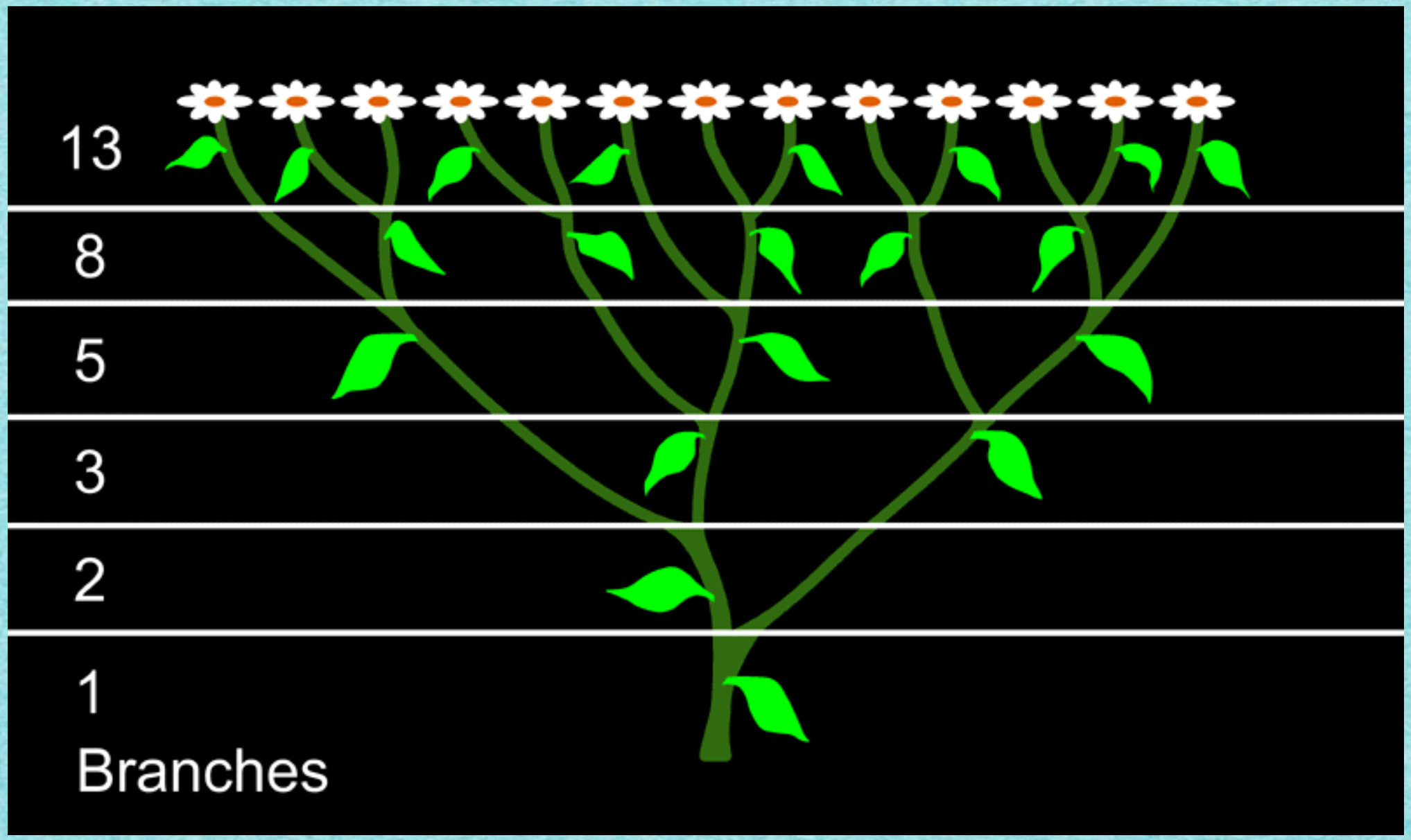
Fibonacci-Zahlen



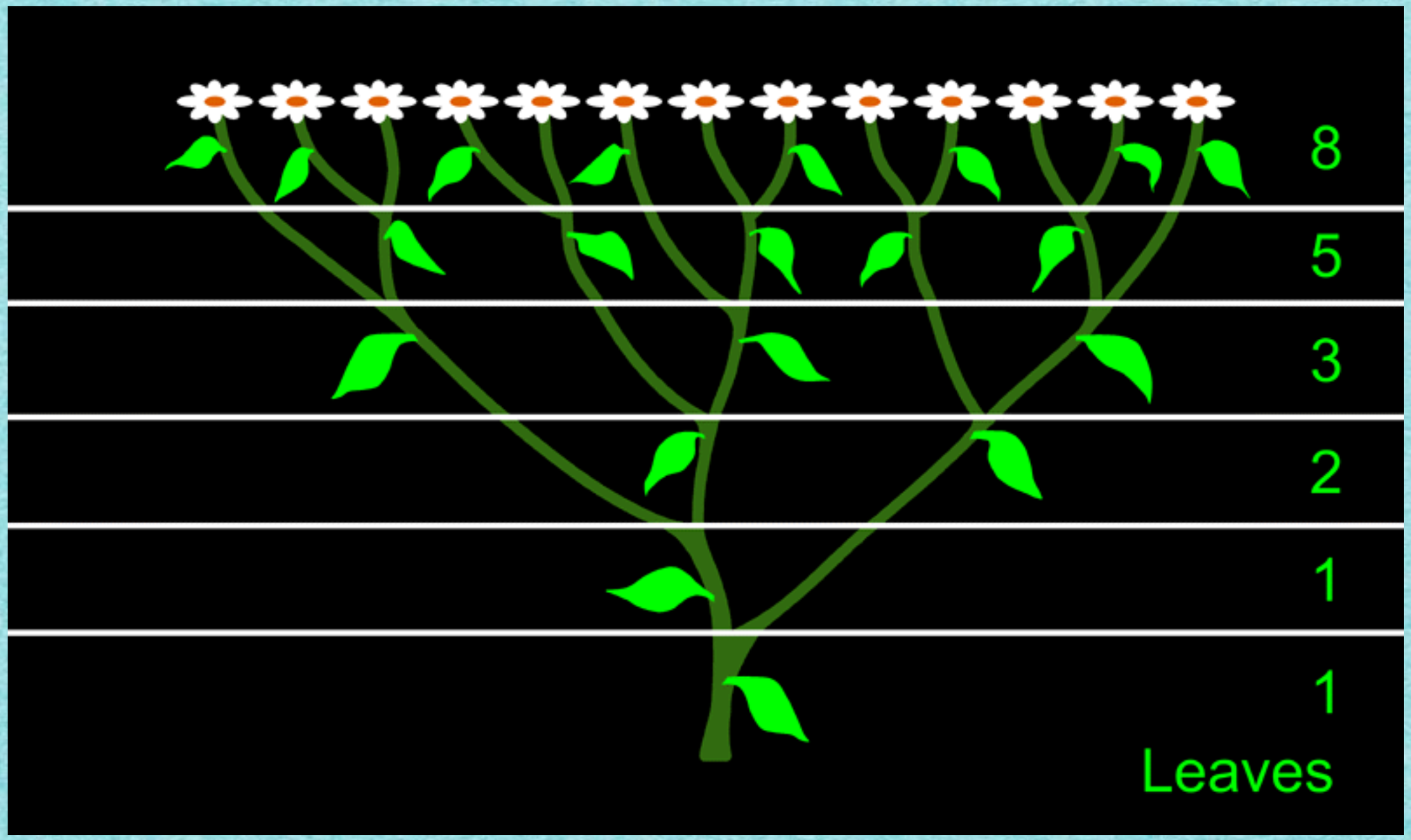
Fibonacci-Zahlen



Fibonacci-Zahlen



Fibonacci-Zahlen



Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

1

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

2

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

3

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

5

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

8

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

1 3

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

2.1

Fibonacci-Zahlen

Fibonacci-Zahlen

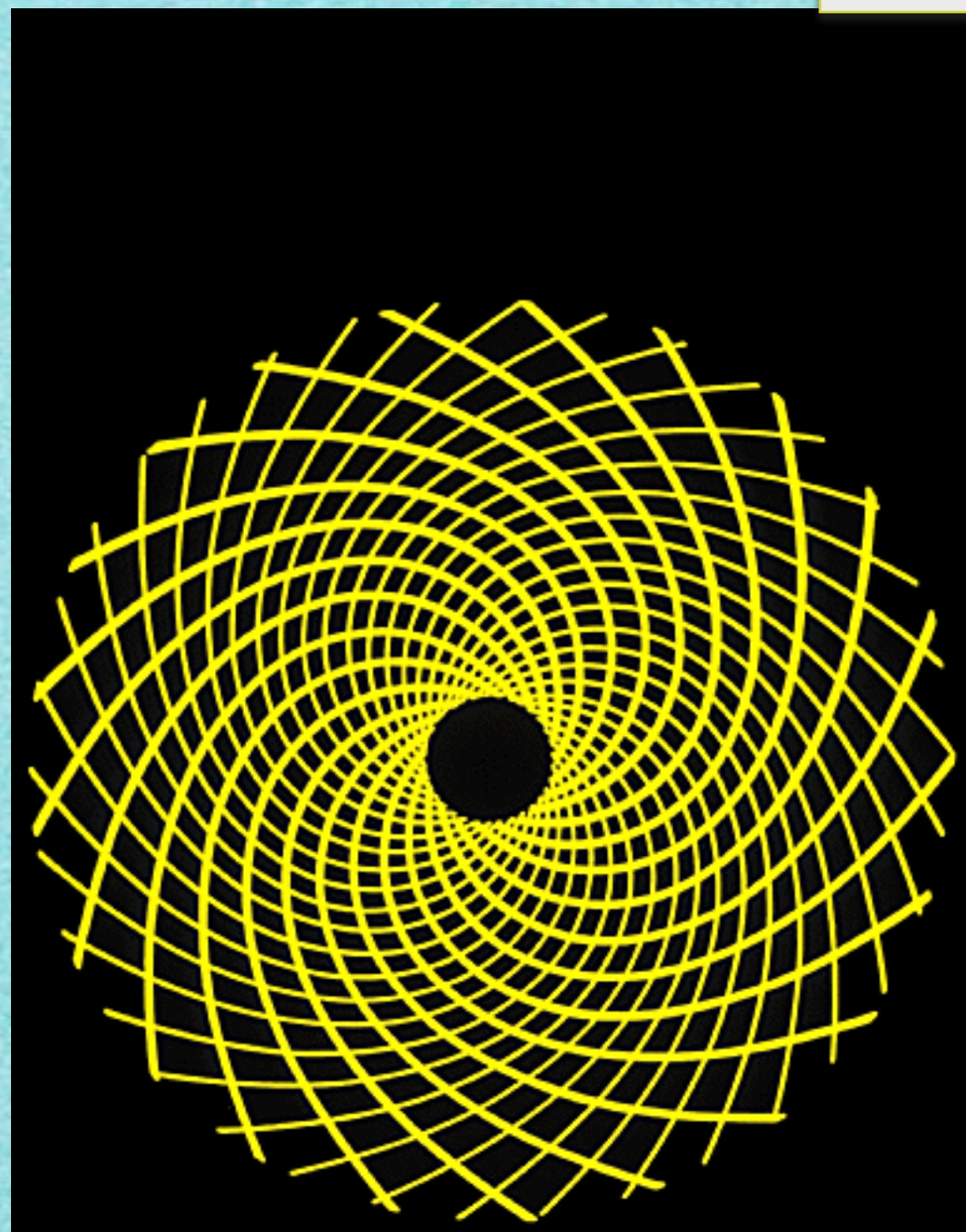
Fibonacci-Zahlen

34

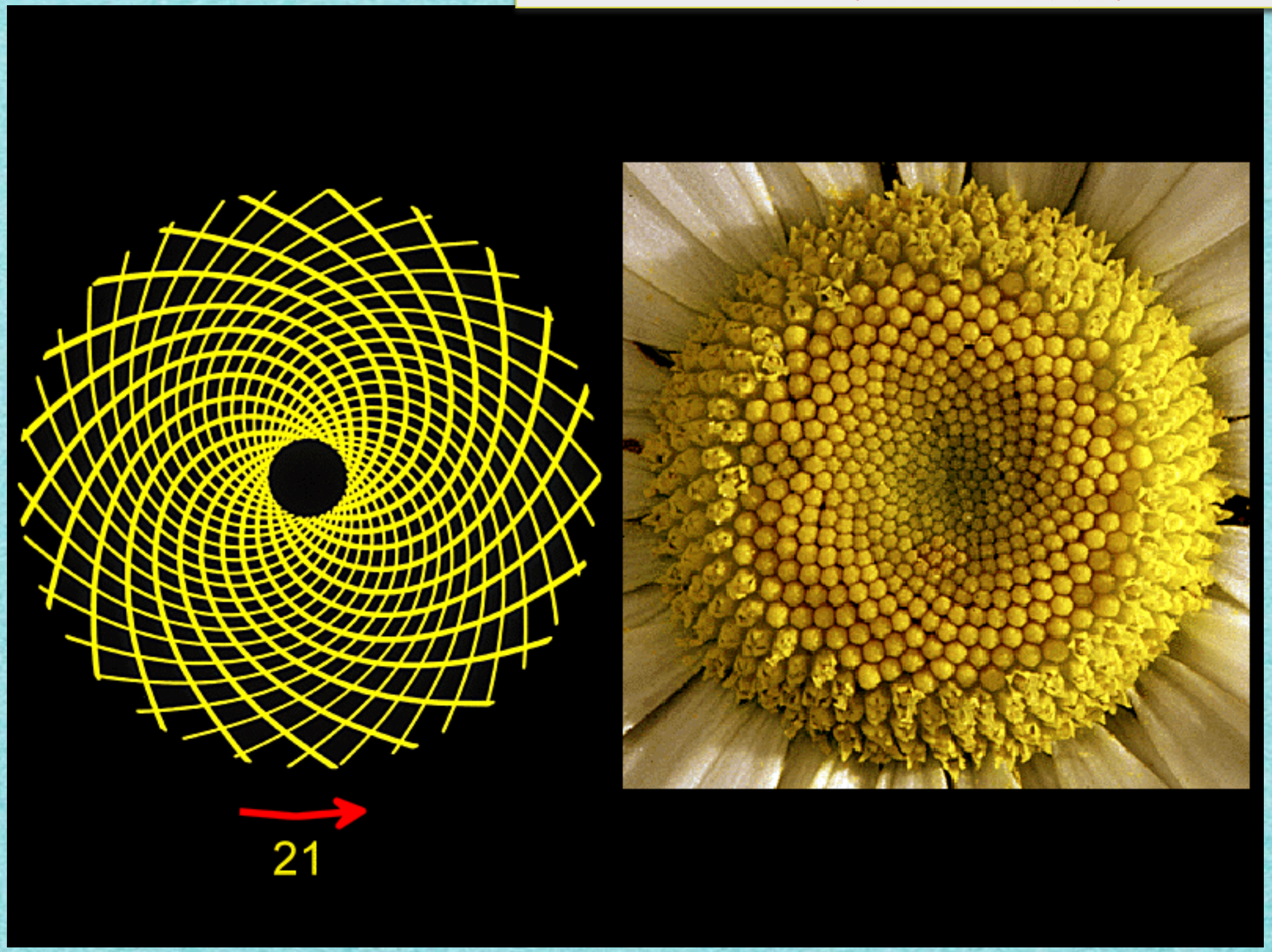
Fibonacci-Zahlen

Fibonacci-Zahlen

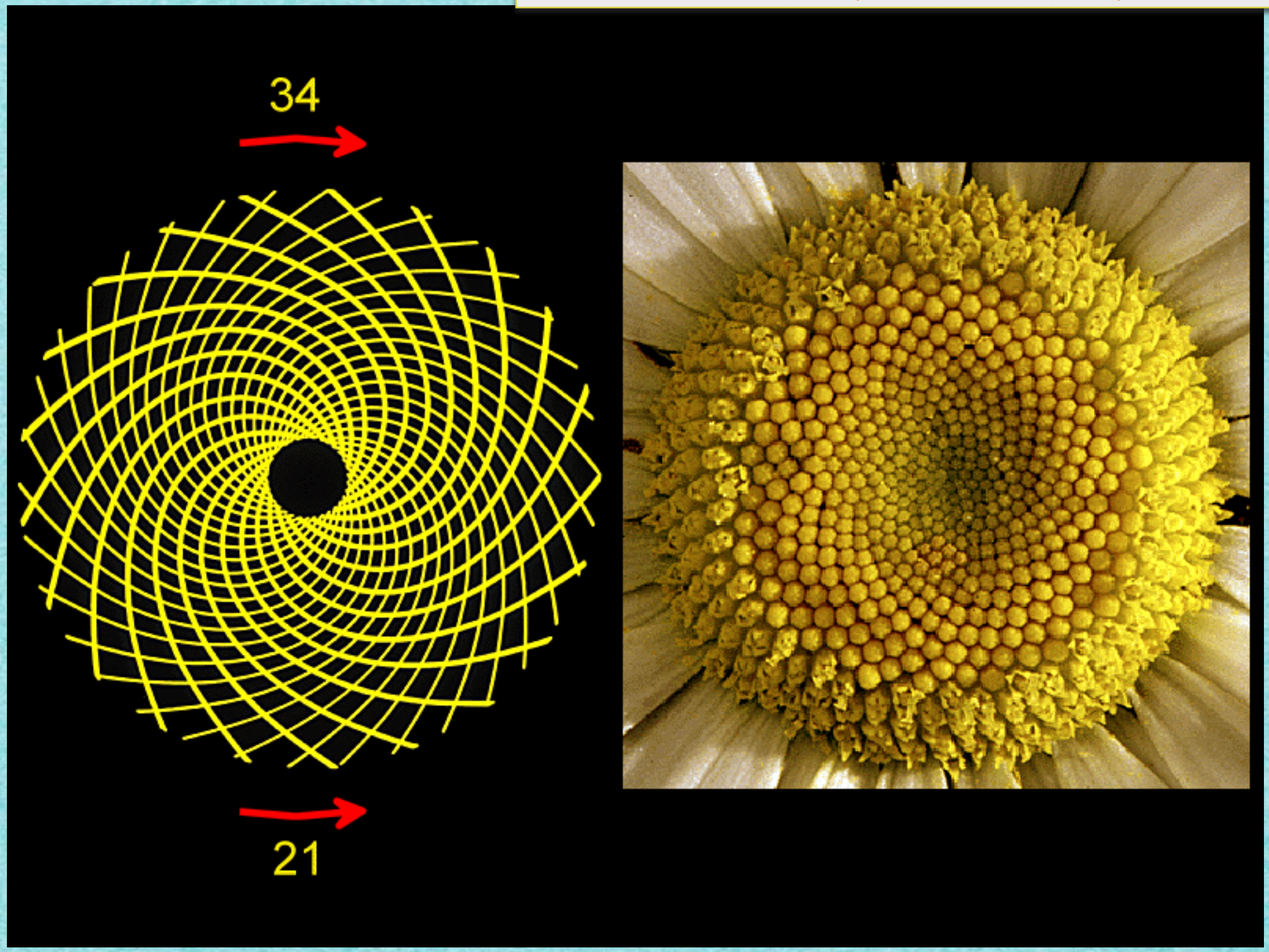
Fibonacci-Zahlen



Fibonacci-Zahlen

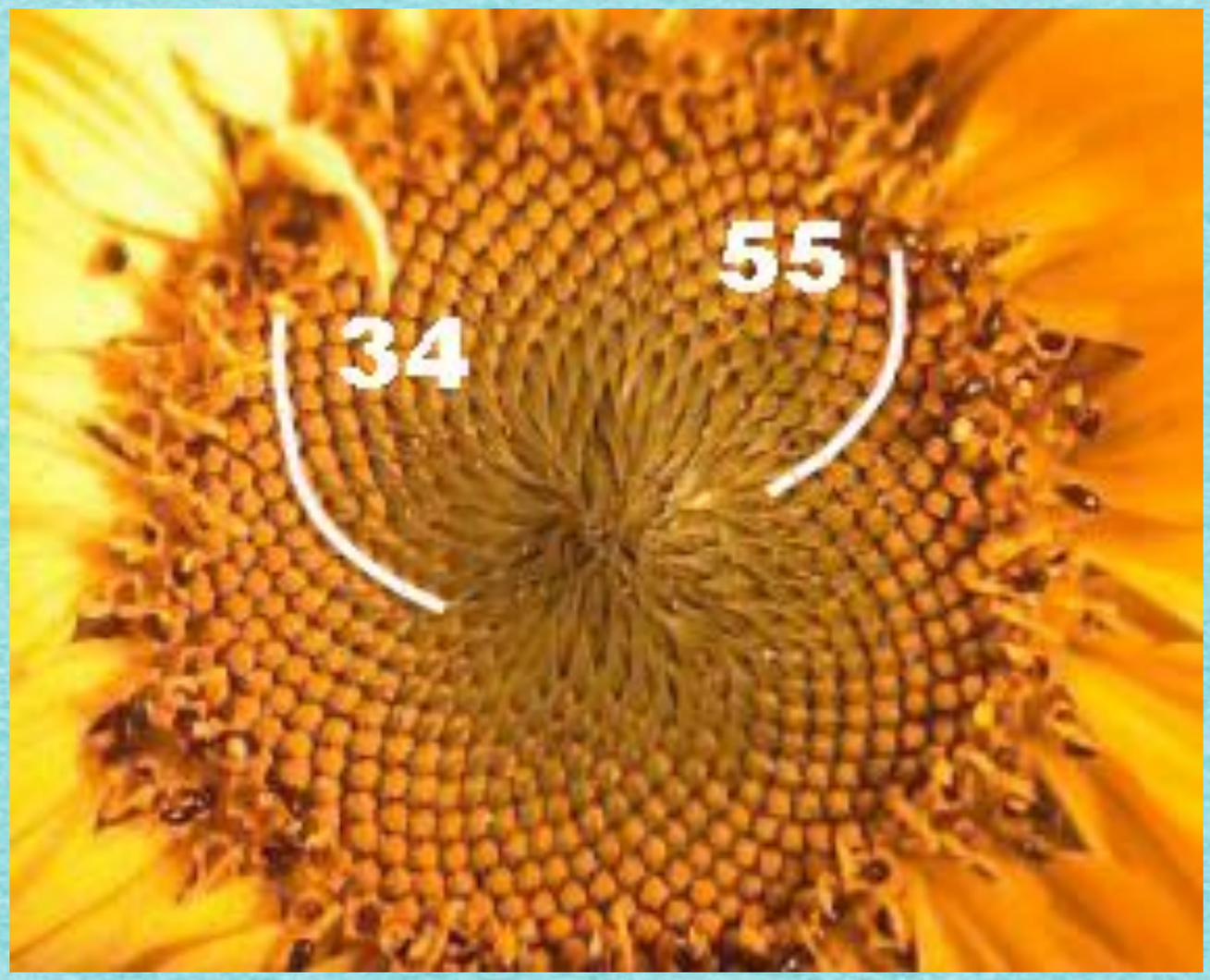


Fibonacci-Zahlen



Fibonacci-Zahlen

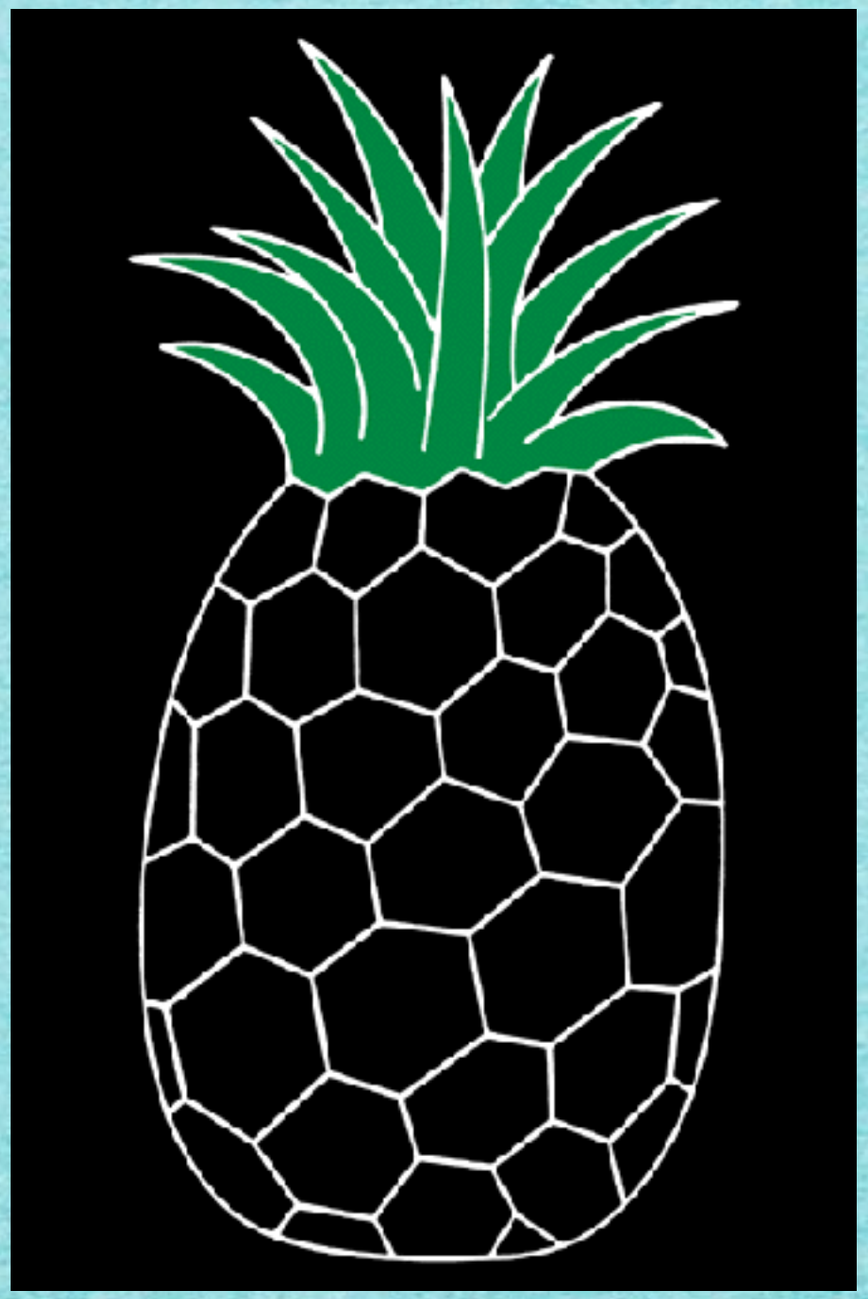
Fibonacci-Zahlen



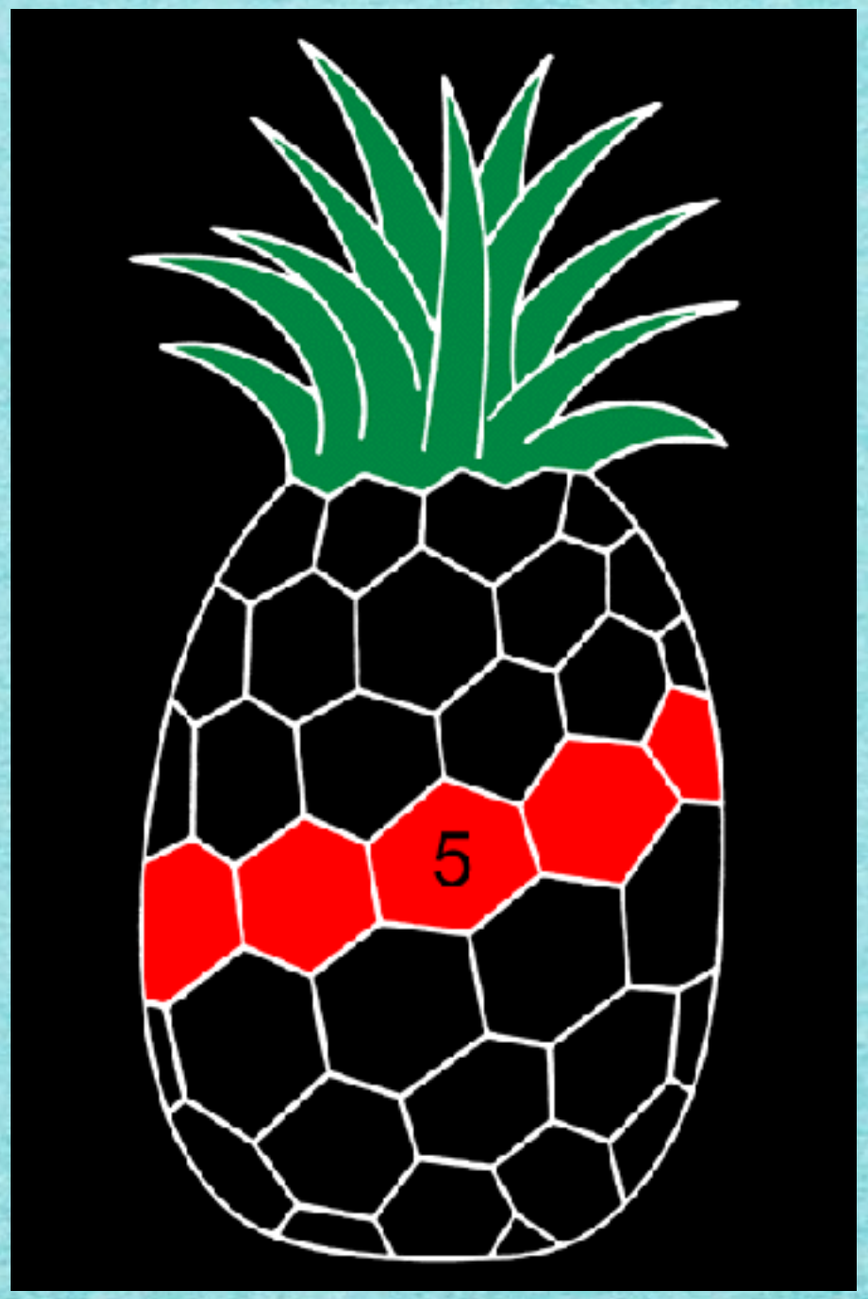
Fibonacci-Zahlen

Fibonacci-Zahlen

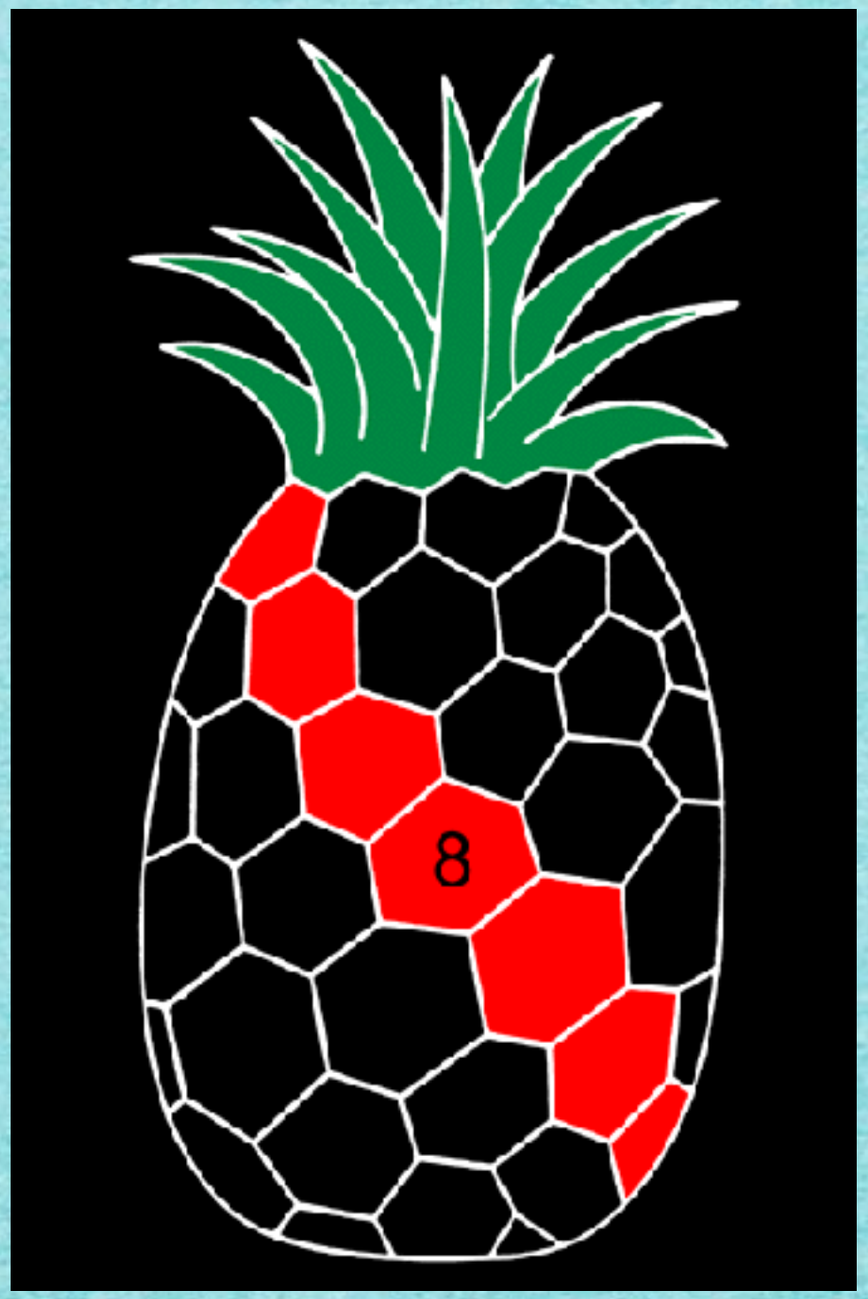
Fibonacci-Zahlen



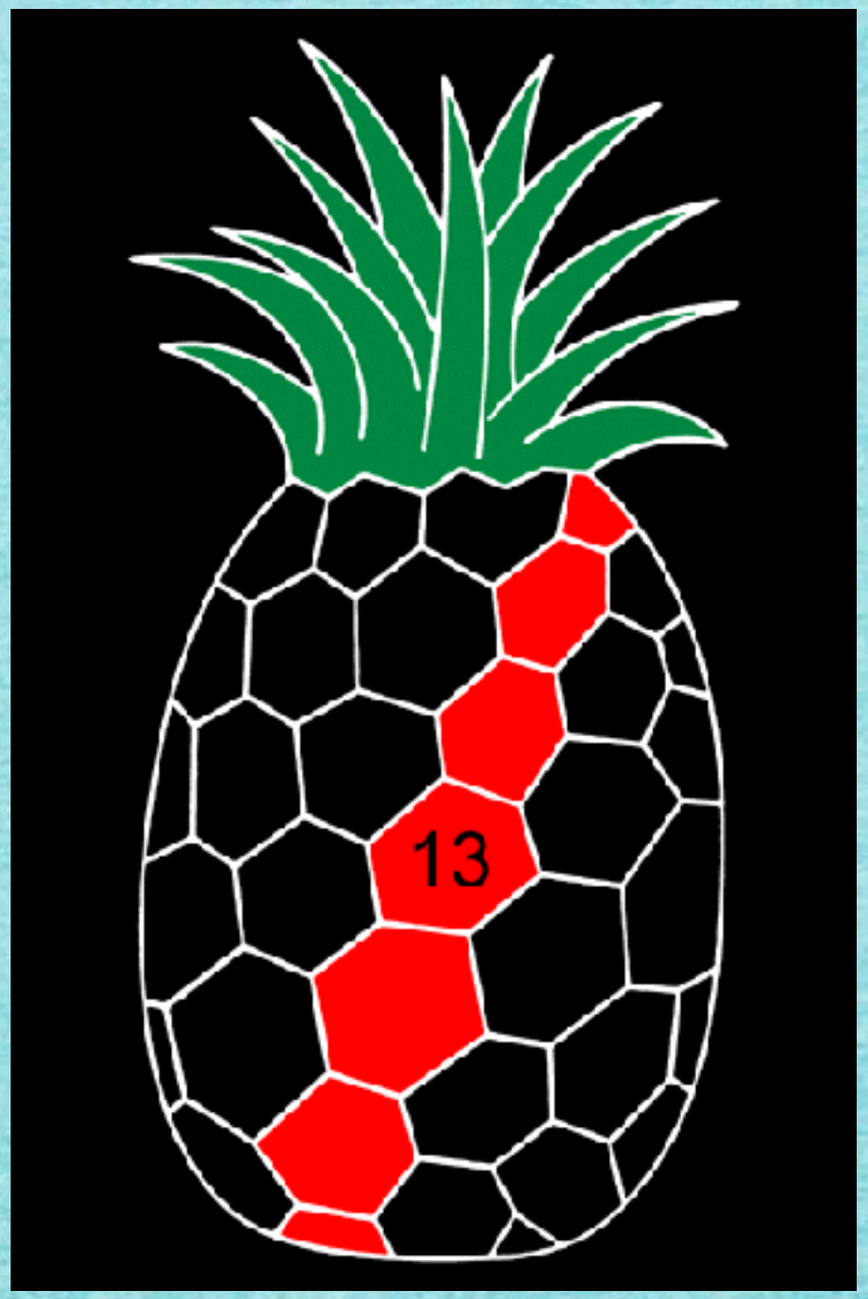
Fibonacci-Zahlen



Fibonacci-Zahlen



Fibonacci-Zahlen



Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

3:2=1.500

Fibonacci-Zahlen

3:2=1.500

5:3=1.666

Fibonacci-Zahlen

3:2=1.500

5:3=1.666

8:5=1.600

Fibonacci-Zahlen

3:2=1.500

5:3=1.666

8:5=1.600

13:8=1.625

Fibonacci-Zahlen

3:2=1.500

5:3=1.666

8:5=1.600

13:8=1.625

21:13=1.615

Fibonacci-Zahlen

3:2=1.500

5:3=1.666

8:5=1.600

13:8=1.625

21:13=1.615

34:21=1.619

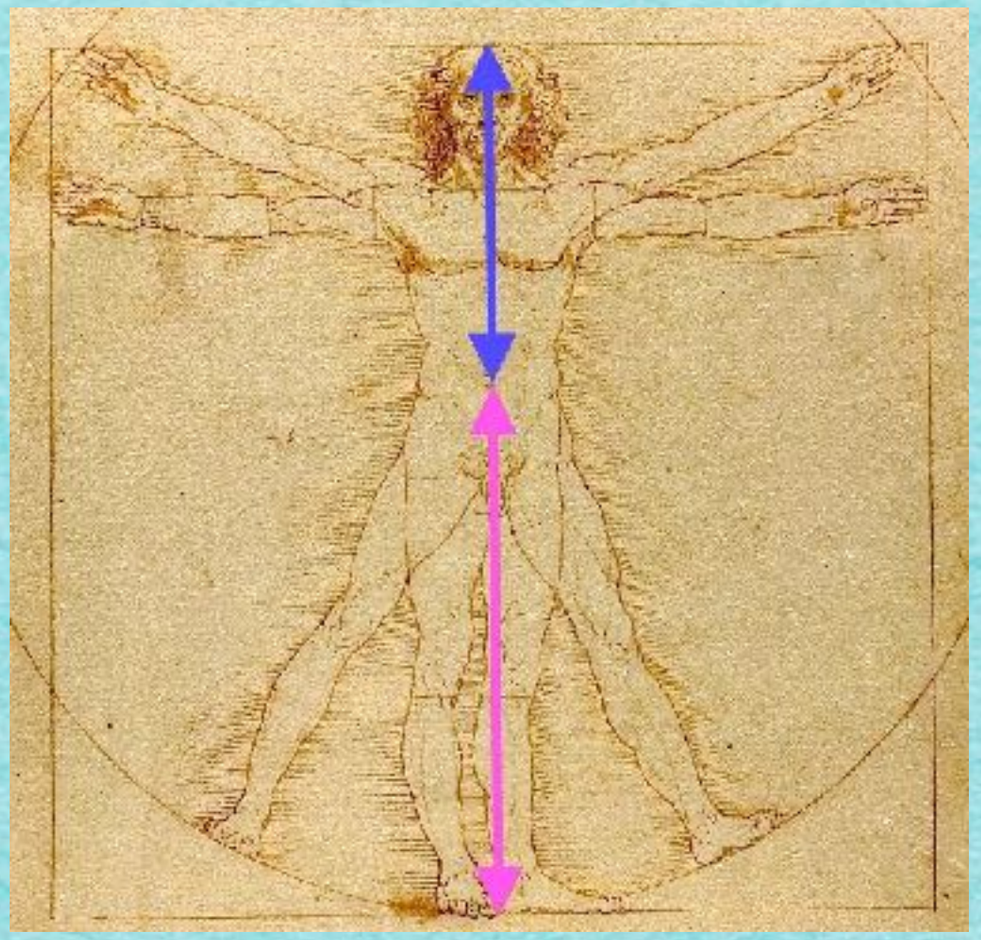
Fibonacci-Zahlen

Fibonacci-Zahlen

Fibonacci-Zahlen

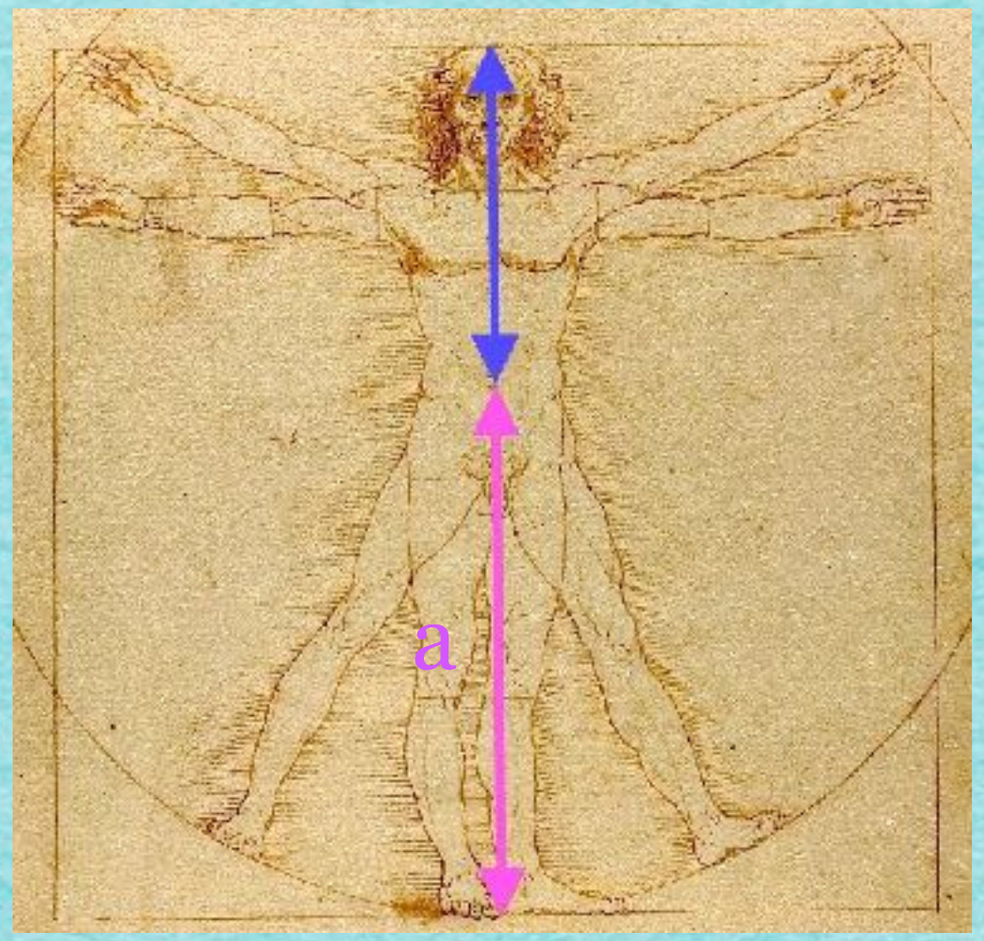
Goldener Schnitt:

Fibonacci-Zahlen



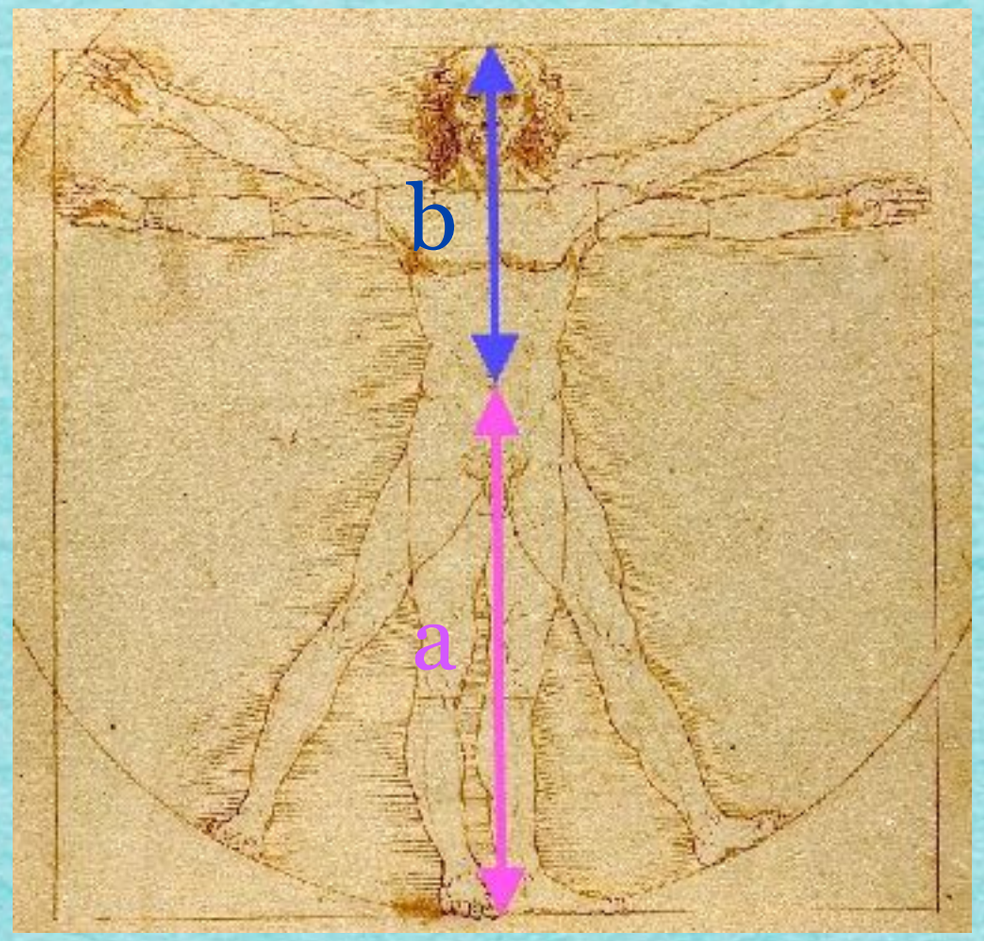
Goldener Schnitt:

Fibonacci-Zahlen



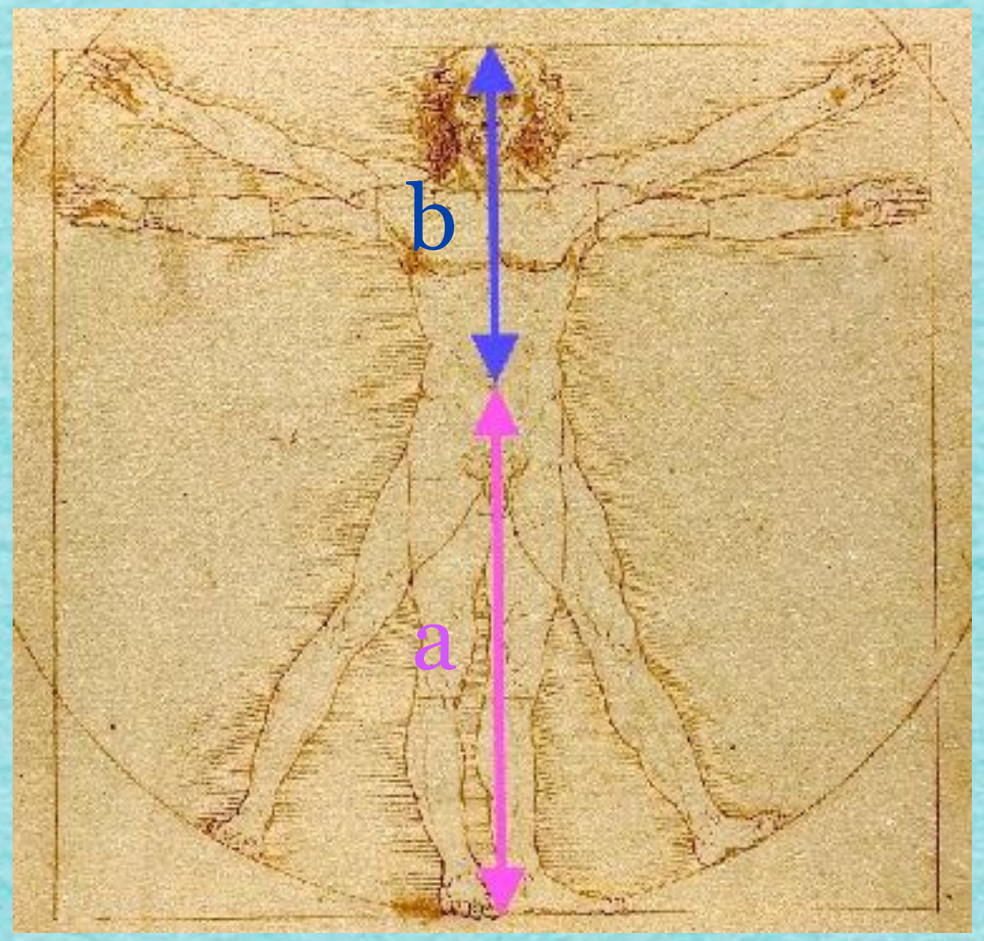
Goldener Schnitt:

Fibonacci-Zahlen



Goldener Schnitt:

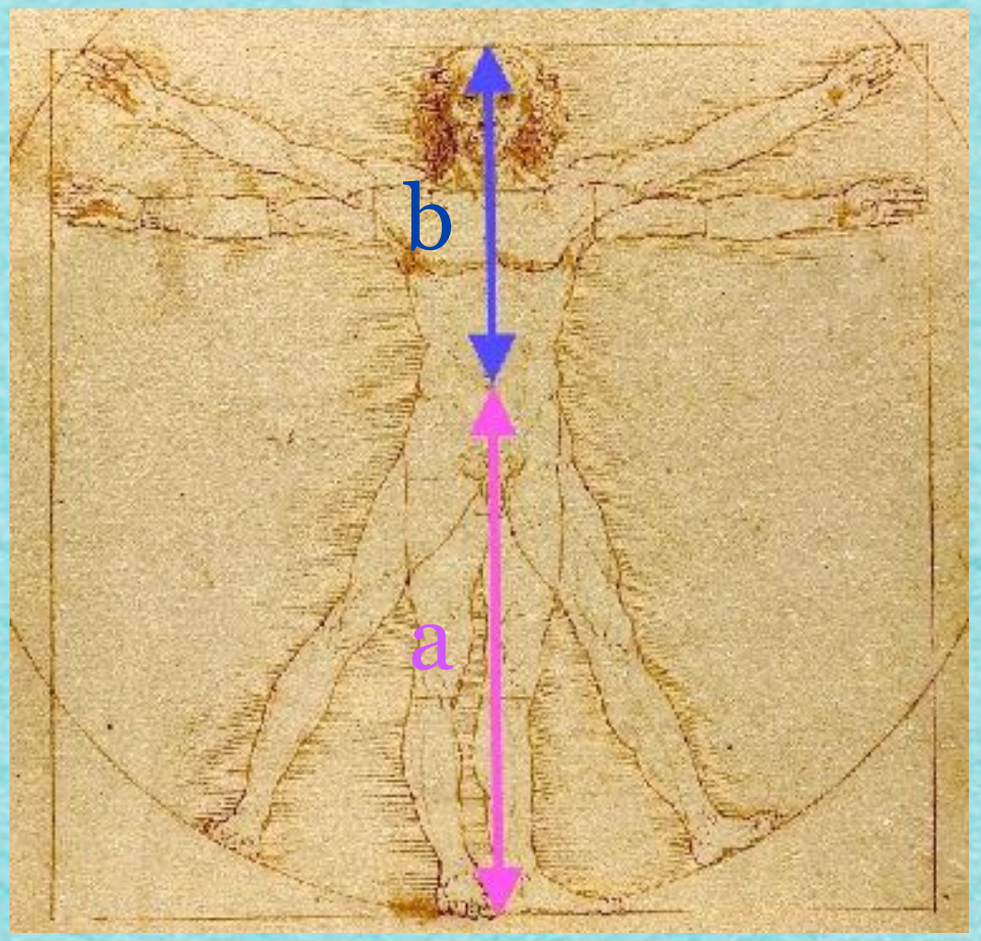
Fibonacci-Zahlen



Goldener Schnitt:

$$\frac{a}{a+b} = \frac{b}{a}$$

Fibonacci-Zahlen



Goldener Schnitt:

$$\frac{a}{a+b} = \frac{b}{a}$$

Fibonacci-Zahlen

Fibonacci-Zahlen

$$\frac{a}{a+b} = \frac{b}{a}$$

Fibonacci-Zahlen

$$\frac{a}{a+b} = \frac{b}{a}$$

Herleitung des Zahlenwertes [Bearbeiten]

Aus der oben angegebenen Definition

$$\frac{a}{b} = \frac{a+b}{a} = 1 + \frac{b}{a}$$

bzw.

$$\frac{a}{b} - 1 - \frac{b}{a} = 0$$

folgt mit $\Phi = \frac{a}{b}$ und $\frac{1}{\Phi} = \frac{b}{a}$

$$\Phi - 1 - \frac{1}{\Phi} = 0\,.$$

Multiplikation mit Φ ergibt die quadratische Gleichung

$$\Phi^2 - \Phi - 1 = 0\,.$$

Diese Gleichung hat genau die beiden Lösungen

$$\Phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618\,033$$

und

$$\bar{\Phi} = \frac{1-\sqrt{5}}{2} = 1 - \Phi = -\frac{1}{\Phi} \approx -0{,}618\,033\,.$$

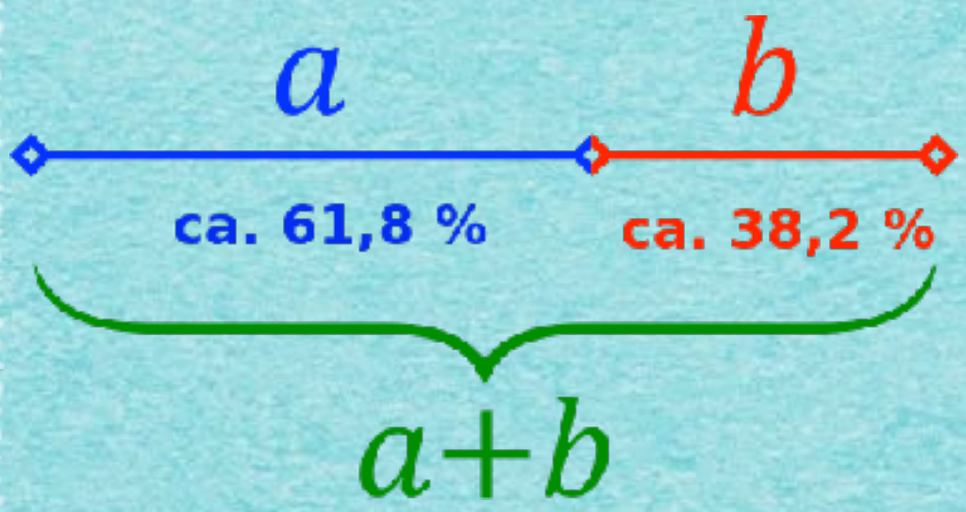
Da $\bar{\Phi}$ negativ ist, ist Φ die gesuchte Goldene Zahl.

Aus diesen Betrachtungen folgt unmittelbar die interessante Beziehung:

$$\frac{1}{\Phi} + 1 = \Phi = \Phi^2 - 1$$

Fibonacci-Zahlen

$$\frac{a}{a+b} = \frac{b}{a}$$



Herleitung des Zahlenwertes [Bearbeiten]

Aus der oben angegebenen Definition

$$\frac{a}{b} = \frac{a+b}{a} = 1 + \frac{b}{a}$$

bzw.

$$\frac{a}{b} - 1 - \frac{b}{a} = 0$$

folgt mit $\Phi = \frac{a}{b}$ und $\frac{1}{\Phi} = \frac{b}{a}$

$$\Phi - 1 - \frac{1}{\Phi} = 0\,.$$

Multiplikation mit Φ ergibt die quadratische Gleichung

$$\Phi^2 - \Phi - 1 = 0\,.$$

Diese Gleichung hat genau die beiden Lösungen

$$\Phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618\,033$$

und

$$\bar{\Phi} = \frac{1-\sqrt{5}}{2} = 1 - \Phi = -\frac{1}{\Phi} \approx -0{,}618\,033\,.$$

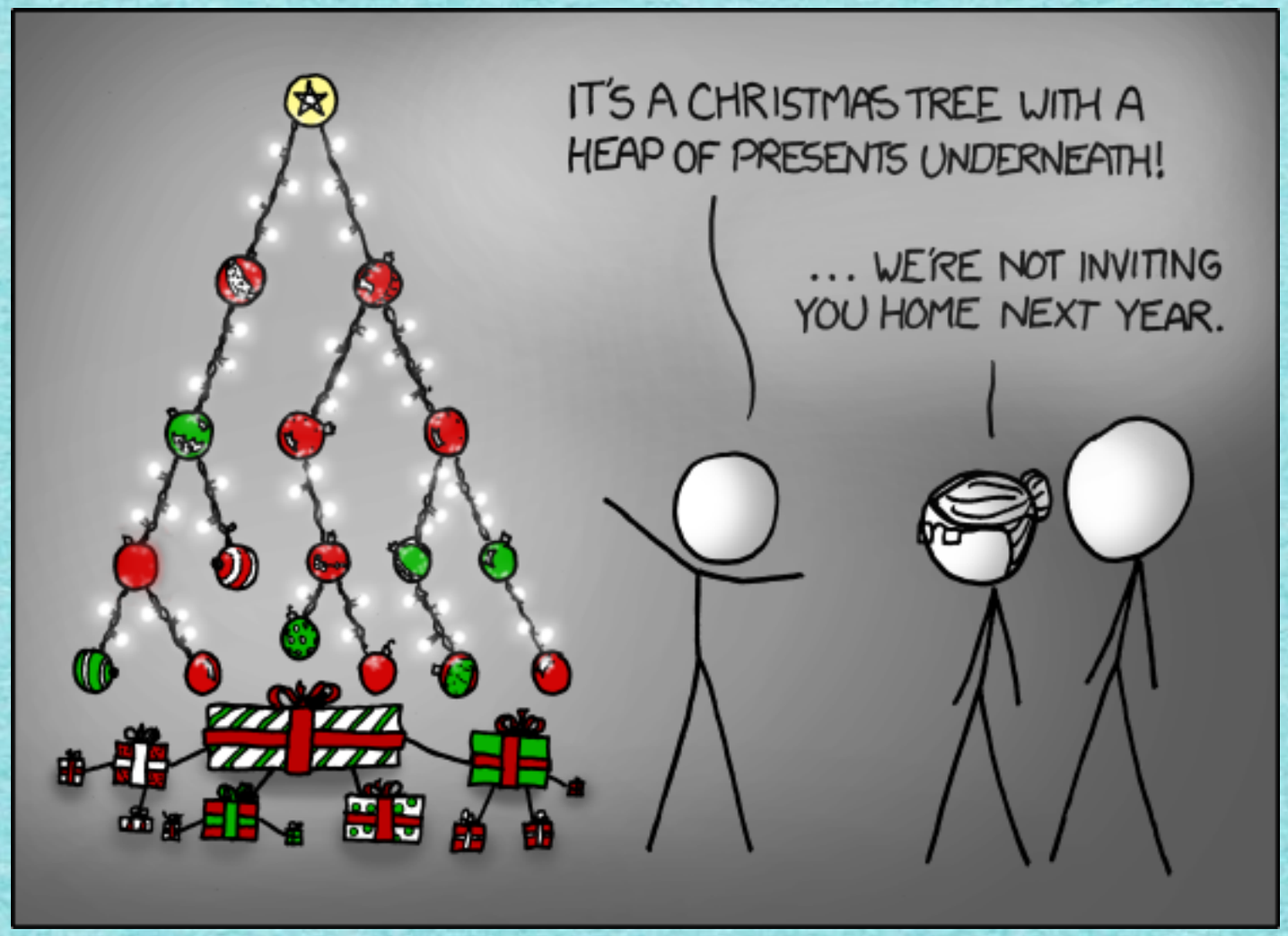
Da $\bar{\Phi}$ negativ ist, ist Φ die gesuchte Goldene Zahl.

Aus diesen Betrachtungen folgt unmittelbar die interessante Beziehung:

$$\frac{1}{\Phi} + 1 = \Phi = \Phi^2 - 1$$

Zusammenfassung Kapitel 4!

Zusammenfassung Kapitel 4!

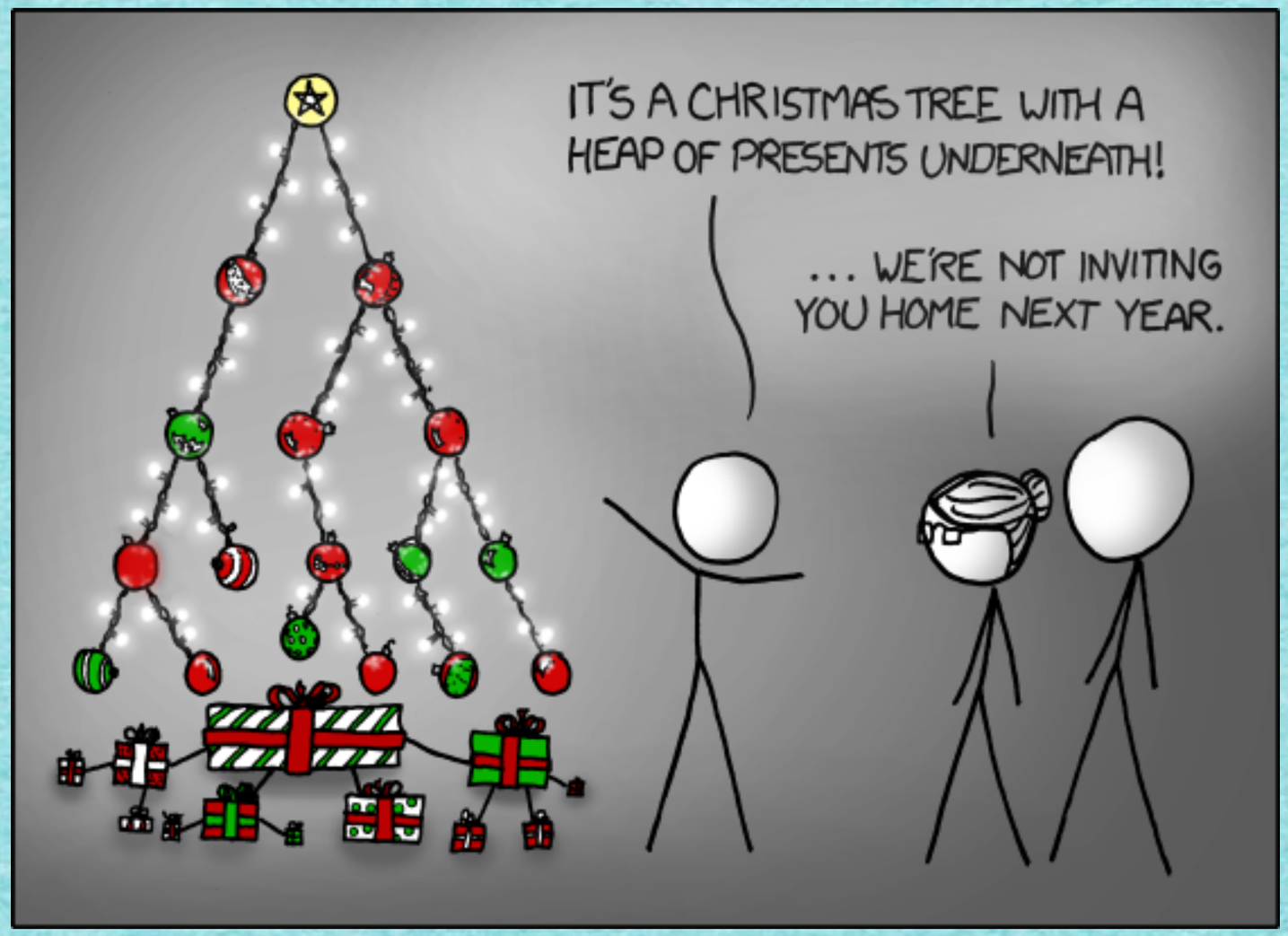


Zusammenfassung Kapitel 4!

Zusammenfassung Kapitel 4!

NI.KO L.AUS

Zusammenfassung Kapitel 4!



Frohe Weihnachten!

Mehr demnächst!

s.fekete@tu-bs.de