

## Kapitel 4.6: AVL-Bäume

Algorithmen und Datenstrukturen WS 2023/24

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:**

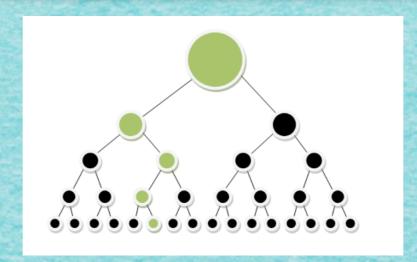
• O(log n): logarithmische Zeit

Wiederholtes Halbieren

Satz 4.4
Suchen, Minimum, Maximum, Nachfolger, Vorgänger können in einem binären Suchbaum der Höhe h in Zeit O(h) durchlaufen werden.

#### **Beweis:**

Klar, der Baum wird nur einmal vertikal durchlaufen!

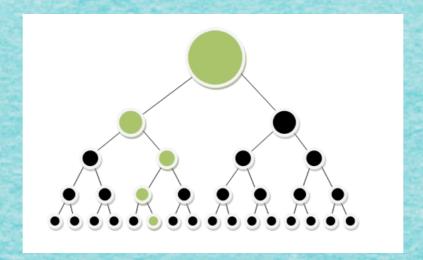


**Satz 4.5** 

Einfügen benötigt O(h) für einen binären Suchbaum der Höhe h.

**Beweis:** 

Klar, der Baum wird nur vertikal abwärts durchlaufen!



**Satz 4.6** 

Löschen benötigt O(h) für einen binären Suchbaum der Höhe h.

**Beweis:** 

Klar, der Baum wird i.W. nur einmal durchlaufen!

#### **Schnell:**

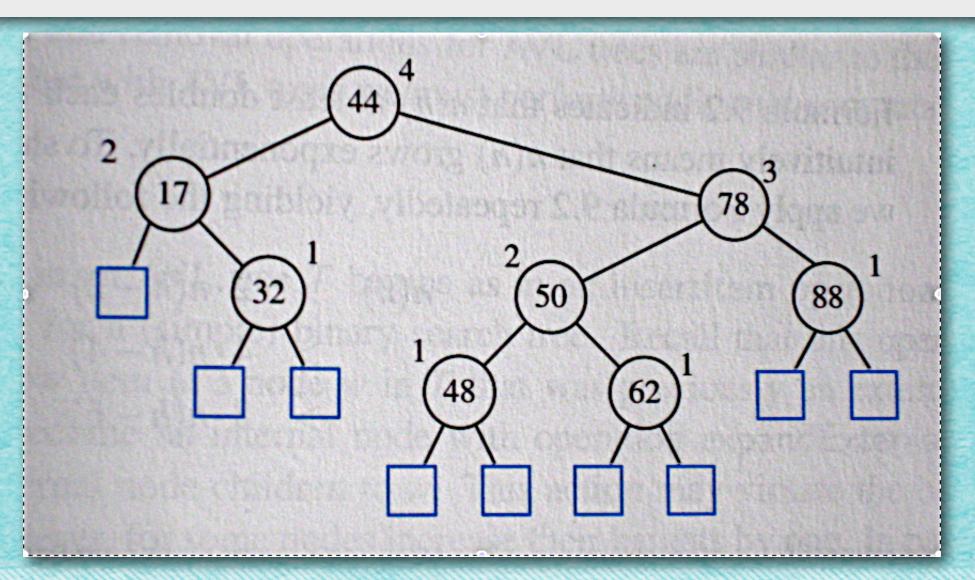
- O(log n): logarithmische Zeit
- O(h): Tiefe des Baumes

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

#### 4.6 AV L-Bäume

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

- (1) Ein binärer Suchbaum ist <u>höhenbalanciert</u>, 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.



## Ab an die Tafel!

s.fekete@tu-bs.de

#### 4.6 AV L-Bäume

**Satz 4.8** 

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

#### **Beweis:**

Wie gesehen!

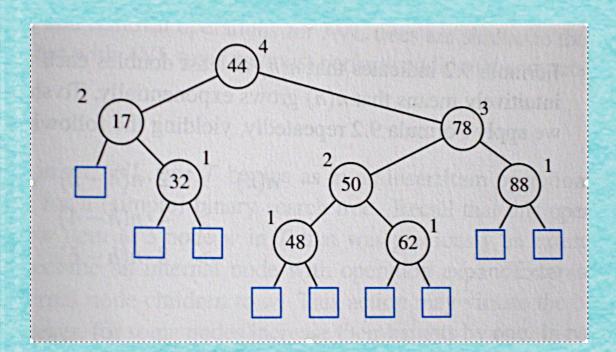
## Damit noch offen:

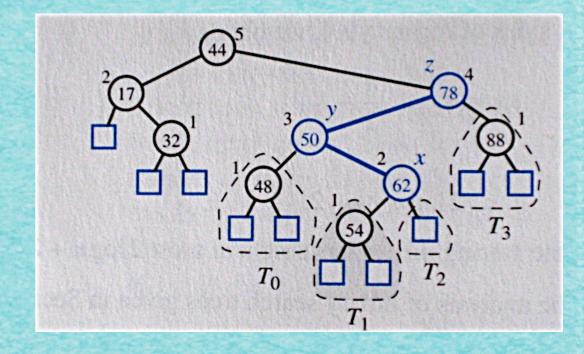
Wie erhält man Höhenbalanciertheit in dynamischen Situationen?

# Einfügen ("INSERT")

## **Aufgabe:**

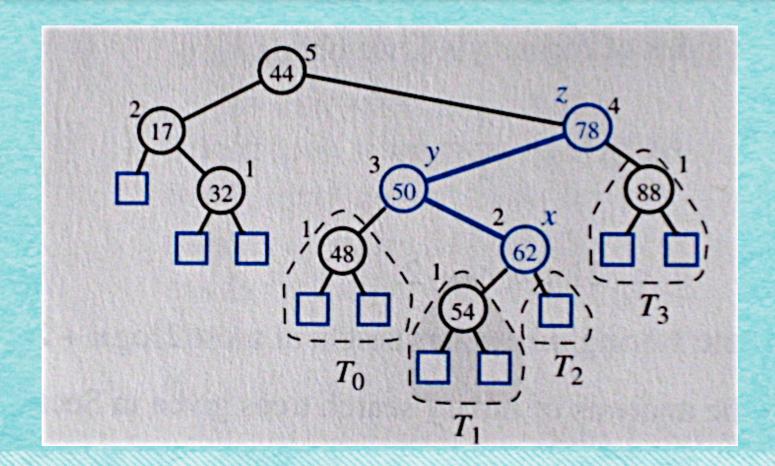
• Füge 54 ein!

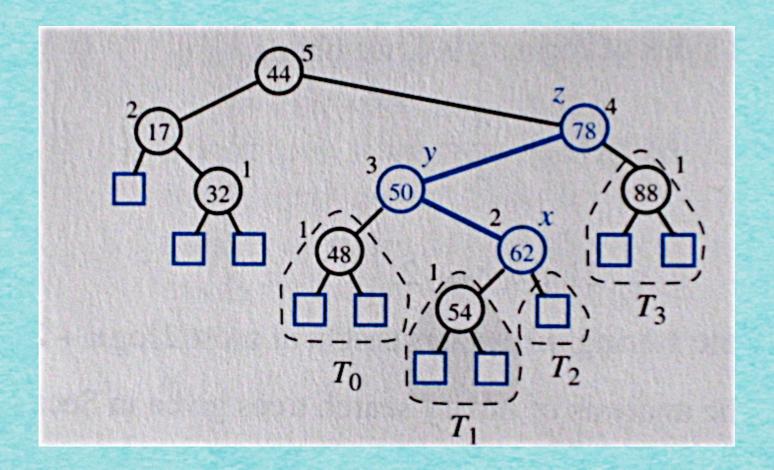




## Idee:

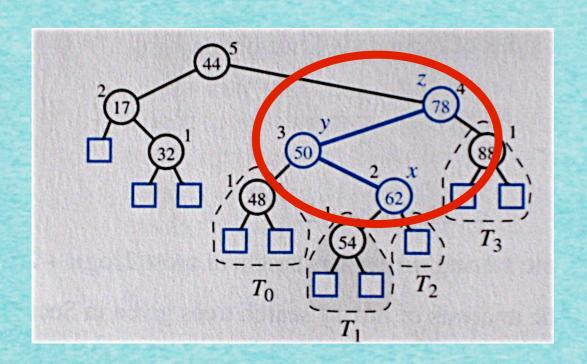
• Höhenbalanciertheit ändert sich beim Einfügen einzelner Elemente nur wenig - und lokal!

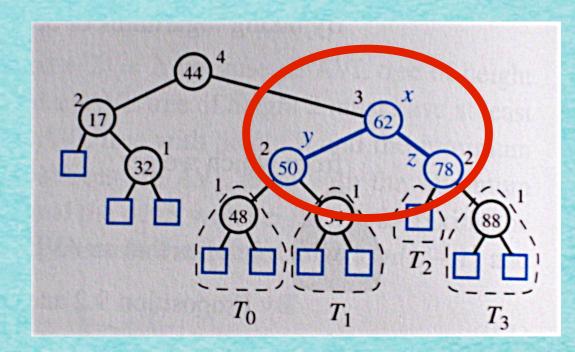




## Was tun?

- 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!





#### 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

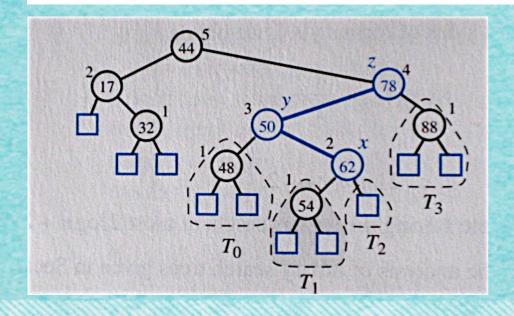
INPUT: Knoten x eines binären Suchbaumes T, Vaterknoten y, Großvaterknoten z

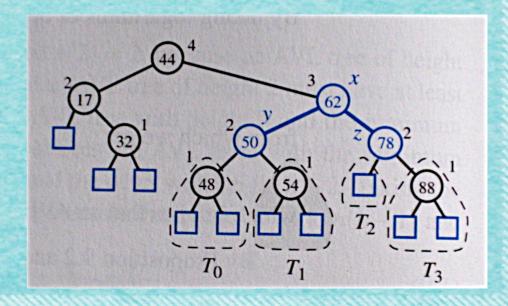
OUTPUT: Binärer Suchbaum T nach Umstrukturierung mit x, y, z

#### RESTRUCTURE(x)

1. Sei (a, b, c) die Größensortierung der Knoten x, y, z; seien (T<sub>0</sub> , T<sub>1</sub> , T<sub>2</sub> , T<sub>3</sub>) 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





#### **Satz 4.10**

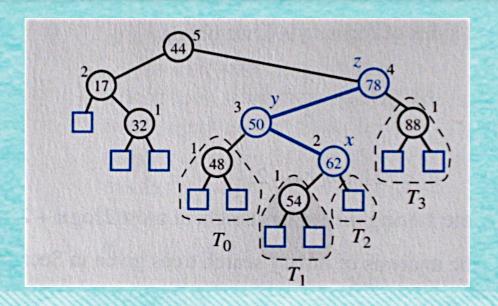
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Einfüge-Operation höhenbalanciert halten. Die Zeit dafür ist O(1).

#### **Beweis:**

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

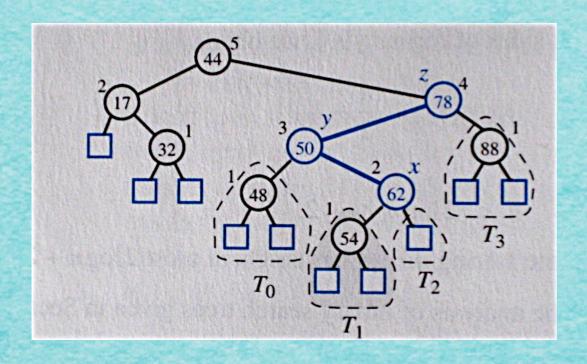
Sei z der nach dem Einfügen niedrigste unbalancierte Vorfahre von v. Sei y das Kind von z, das Vorfahre von v ist; y muss zwei höher sein als das andere Kind von z.

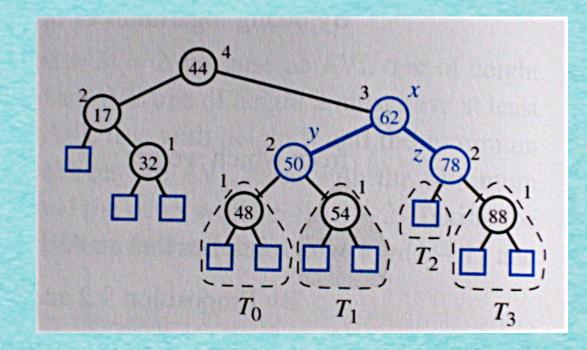
Sei x das Kind von y, das im selben Teilbaum wie v liegt.



#### 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!



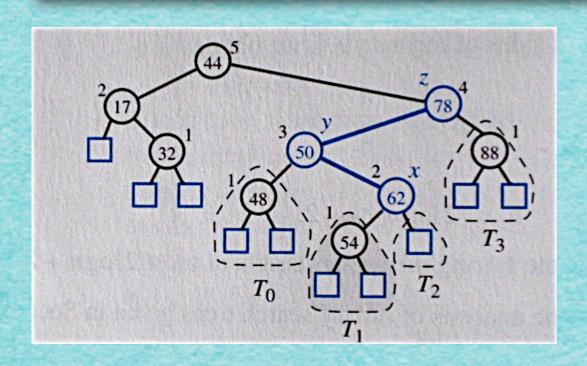


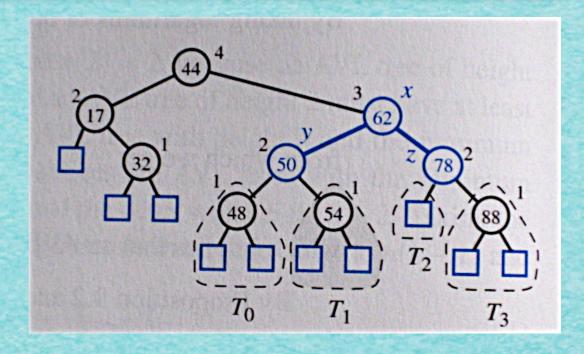
Betrachte jetzt die möglichen Anordnungen von x, y, z!

## Beweis von Satz 4.10 (Forts.):

Welche Anordnungen gibt es?

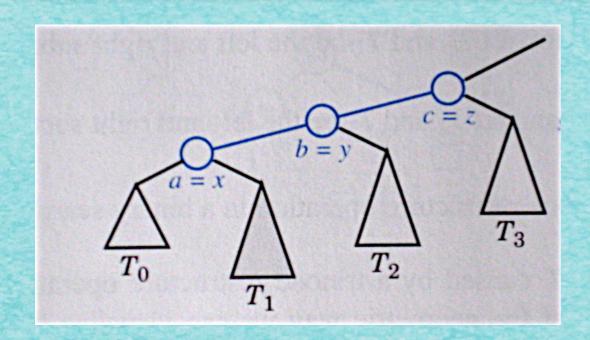
- $(1) \quad x \le y \le z$
- $(2) \quad x \le z \le y$
- $(3) \quad y \le x \le z$
- $(4) \quad y \le z \le x$
- $(5) \quad z \le x \le y$
- $(6) \quad z \le y \le x$

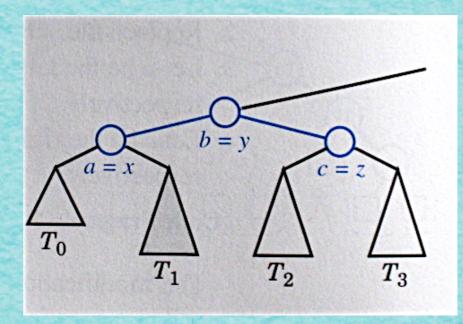




## Beweis von Satz 4.10 (Forts.):

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

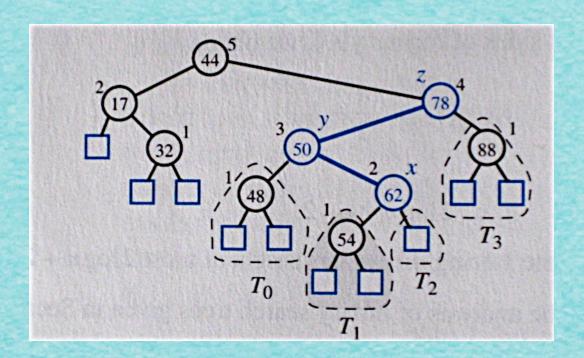




Der Baum ist wieder höhenbalanciert!

## Beweis von Satz 4.10 (Forts.):

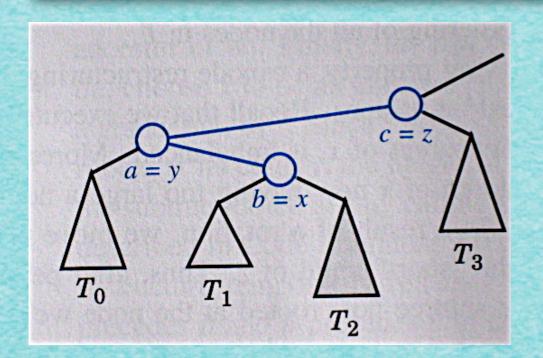
 $(2) \quad x \le z \le y$ 

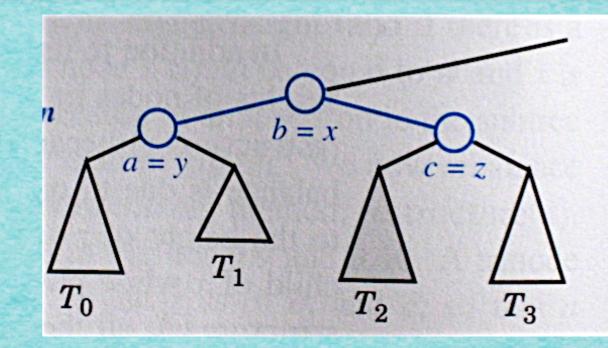


Der Fall kann nicht auftreten!

## Beweis von Satz 4.10 (Forts.):

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

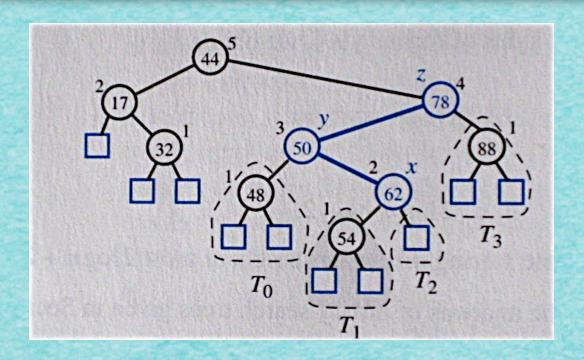




Der Baum ist wieder höhenbalanciert!

## Beweis von Satz 4.10 (Forts.):

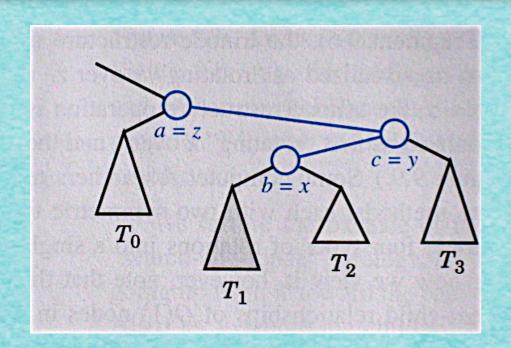
 $(4) \quad y \le z \le x$ 

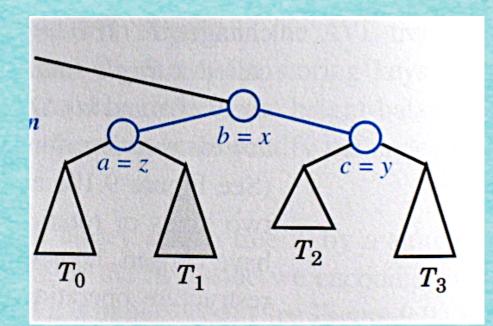


Der Fall kann nicht auftreten!

## Beweis von Satz 4.10 (Forts.):

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

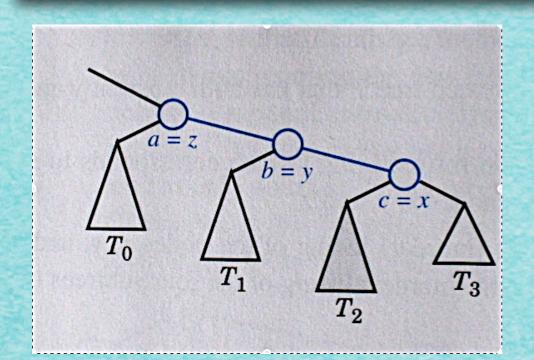


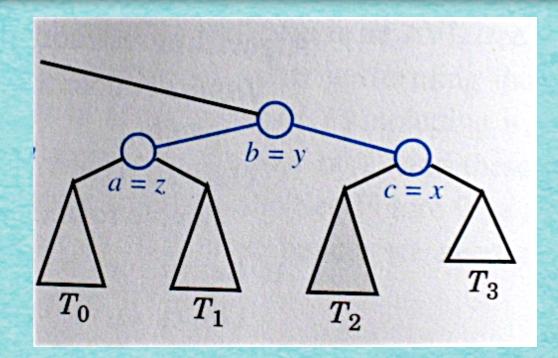


Der Baum ist wieder höhenbalanciert!

### Beweis von Satz 4.10 (Forts.):

**(6)** z≤y≤x





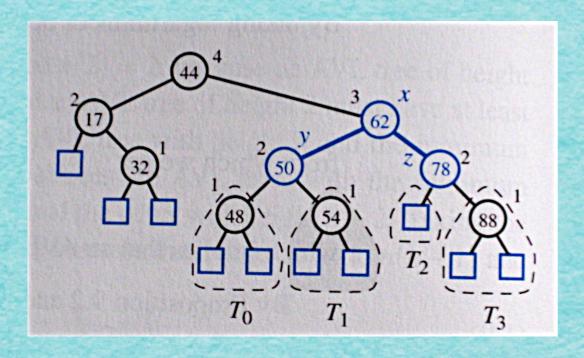
Der Baum ist wieder höhenbalanciert!

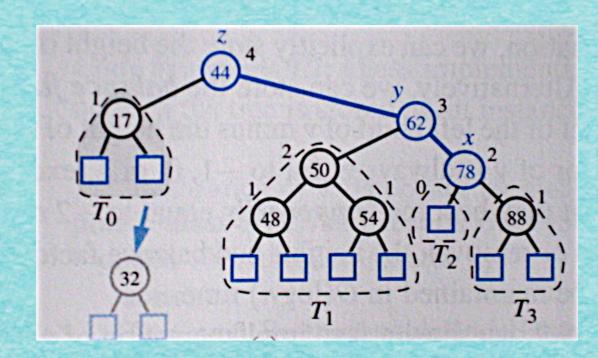
Alle Schritte erfordern nur konstant viele Rechenoperationen.

## Löschen ("PELETE")

## **Aufgabe:**

• Lösche 32!

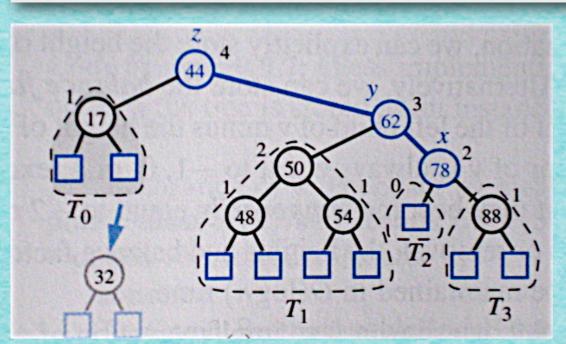


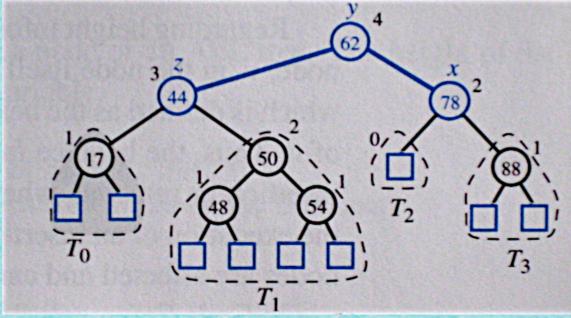


#### Löschen

### **Wieder:**

#### • Verwende RESTRUCTURE!





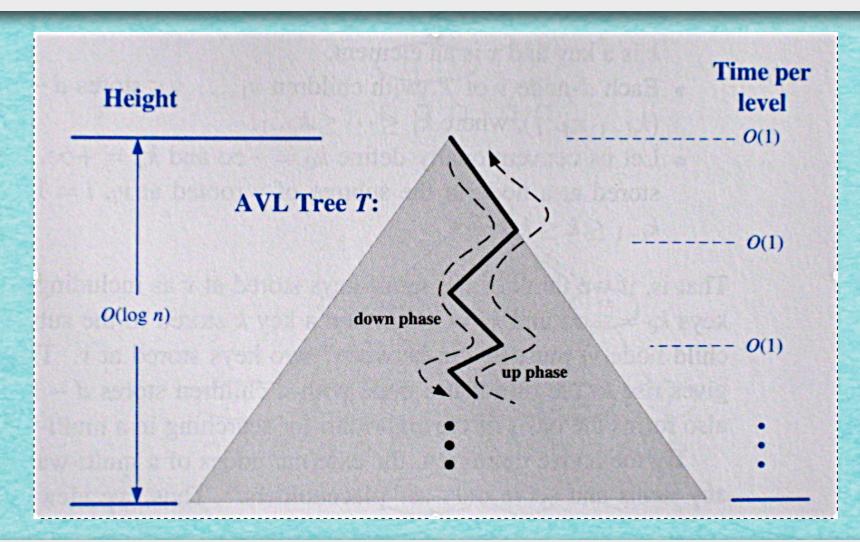
Das reicht in diesem Beispiel!

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

## **Wieder:**

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



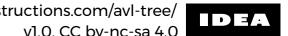
Höchstens O(log n) RESTRUCTURE-Operationen, jeweils in O(1)!

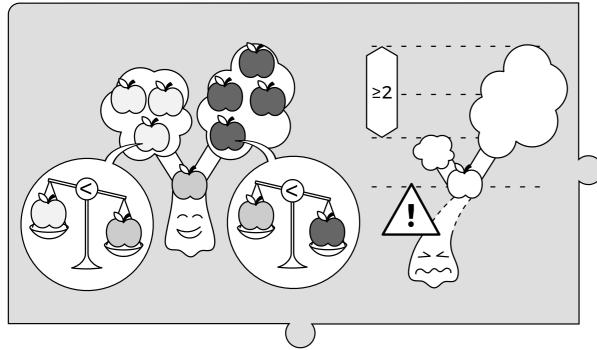
#### Löschen

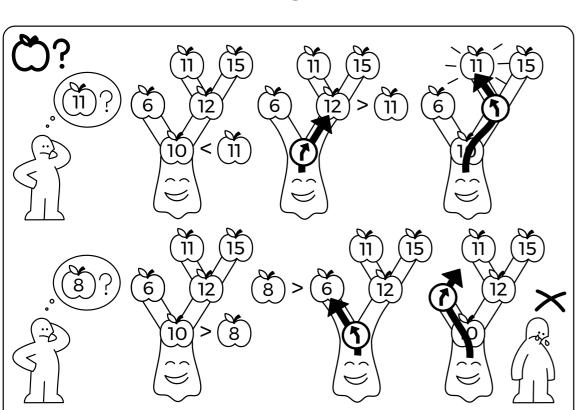
#### **Satz 4.11**

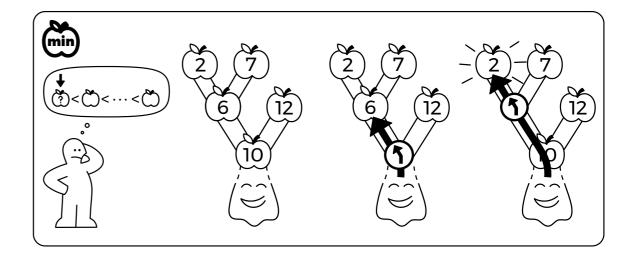
Mithilfe von RESTRUCTURE kann man einen AVL-Baum auch nach einer Lösch-Operation höhenbalanciert halten. Die Zeit dafür ist O(log n).

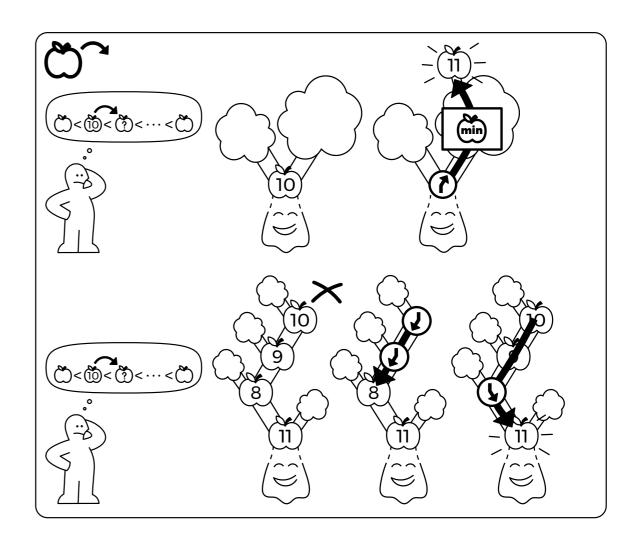
## **BÄLÄNCE TREE**







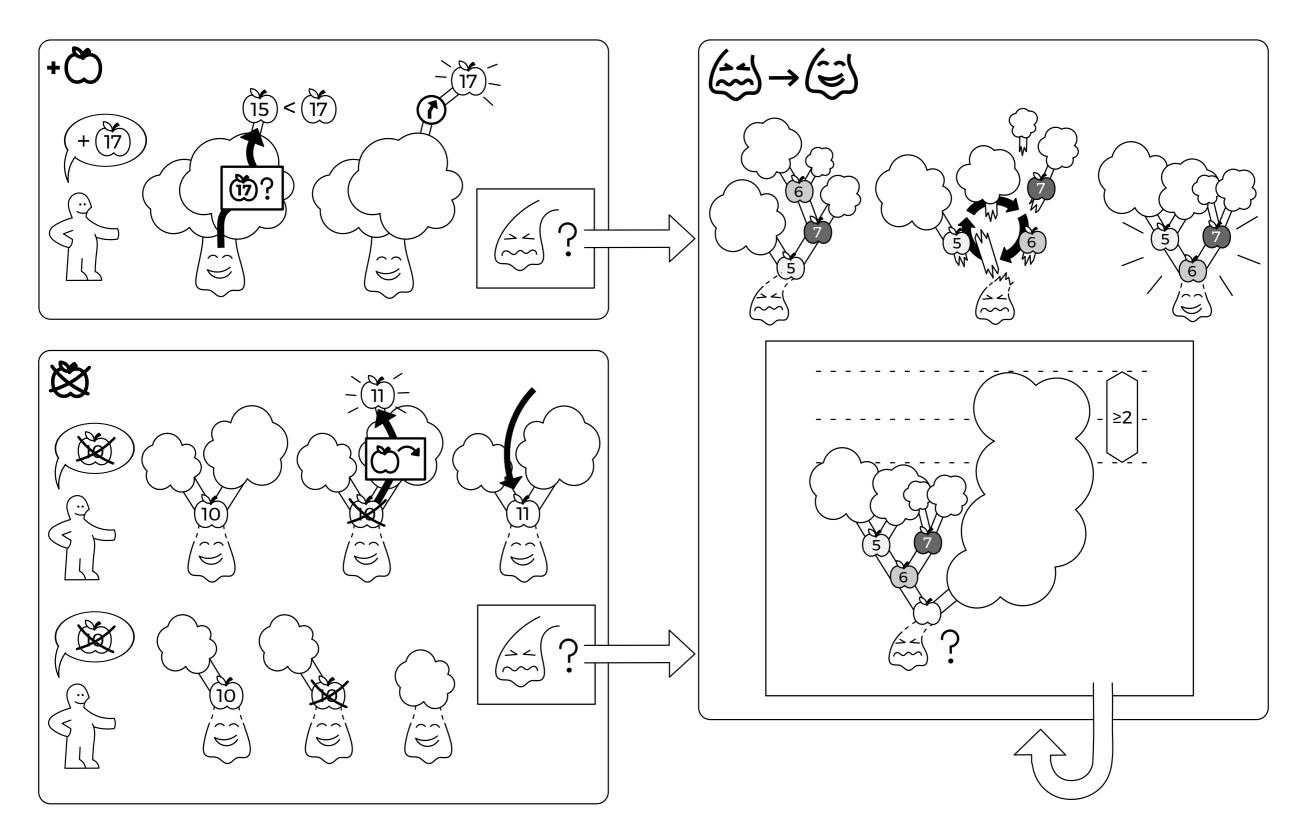




## **BÄLÄNCE TREE**

2/2



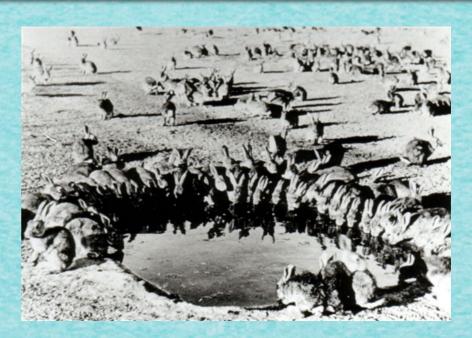




Leonardo da Pisa, gen. Fibonacci 1180-1241

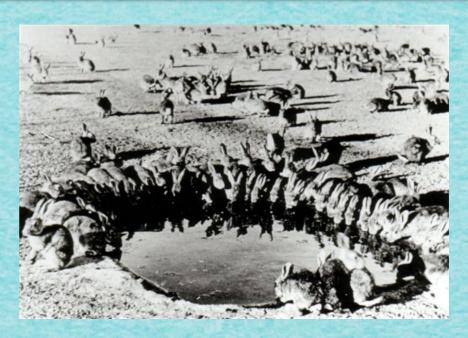
gemmar. que fe fo mele para ; er quib" i uno mile ouo pomane gemmar in ico mele parta coniclos. The fi pirta 4 Tho in le er quib'i que parar para 7 fri ique mele para s er qb parta 4 gemmar alia parta 4 quill'additte que ge face क्ष्म महिमारा । इ विभाग melecer वि विभाग व व geminatu fuere राहि men gentut ito mie falla a purupanant fie fe i ferto mele para : । वो वृष्टि वर्रवार parest । हते geminar र दिएतं एमरे । मूर्व pura ? + cu quib adduit purift : i q geminar ? commo mete. ertifo para 44 cu quib adduct parift 7 4 q geminat ino no mete ert î मिं parta s o द्या quib adduit rurft parte 94 व geminat i deamo. ert ino para 1 + + तो quib adduit rurfit sariff s न वे genninar ? undecimo mele ert î po paria e t ? en qb 4 adding parife ; it + q geminar in ultimo mele erne self para 7 7 7 gror para pepir fin par 7 pfaro loco 7 capite uni 21 im. potet e moe i kao margine quali boc opan fumilit quirmi Sepn कृता गाँत त्य कि पार्रक । ता र शित हे देवक नंदर्श ती वृत्त्या न्यू £# मां त्यं वृताक निर्वे क्वंक्ट्रिं काल्य प्राप्तान क्वामार्थे त्यं प्राप्तकतमार्थन The pollet fice pordine & ffiniat mil meftle! Vatura horer ft, quoy pin - रिकेर नरेली hat केरिका, क्लिर साक् नरेली न्यूंपर 8 % ITT VI har ofter = + tel after om bur orier = eren apmaft ंद ने वृंतार टिक्मित करी प्रांत ठालेक्य मिल्यू . मानी hoini. 1000 व्या मान स्था र १ ५००६ र व लोग जोगां हो है जिसमा कार्य ะ ะเก็ filma erqua fi ermint driot pmi/2ft ? tei boirf. = ? vemanebir rief rogen boil temanebit pimo boi de le Purfit fi de drift. ernmeric ह + A के चंता न्त्रात फिल न्यात किरा remanche fo के remamebrica de 6 constitution de print boil ci ? िर्म द्वा क रेवा करता । वृत्ता मामान्ये दिव प्रथित में द rien bite divot & i co ine toil girti ? p ine girti zonni ?? offiniler fi pomoit quiq tolm poster quiq n. Si ur ife q tolm posse ab hiftqui folui ni will cognolate tule o moini embenis moches मर अठेवा मामाम कृष्णा नृति तो मावि क्यान्त्रिया. नि क्या विमान क्या मिल muo firen agen apini to folubit eru affio. fi at lequal fille to a मर्वे कृति विभा cognotat मार्गिया वृत्तिकात न वृ कृति विकास = > व्य स्ती न्वार के में क्यार कार मार कार के कि कार के कि

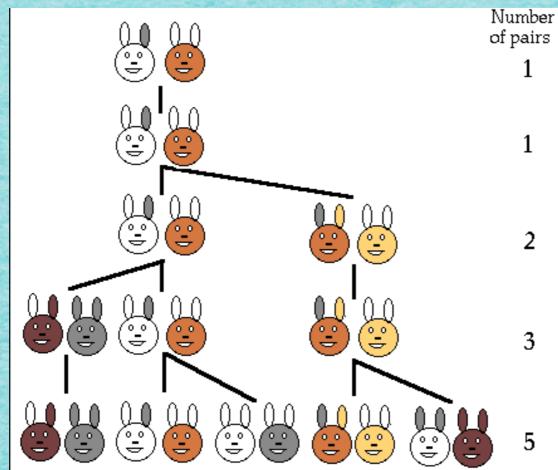
## Indian Ocean Eighty Mile Beach Wallal Downs Cape Keraudren No. 1 Fence No. 2 Fence No. 3 Fence Route taken in film Rabbit Proof Fence Jigalong WESTERN AUSTRALIA Kalbarr Dongara Eucla Moore River Perth Esperance Jerdacuttup Bremer Bay Southern Ocean Albany



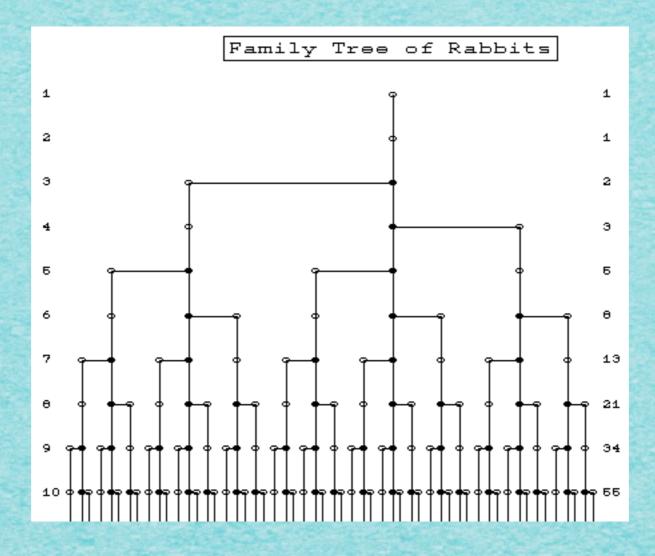
$$F(n)=F(n-1)+F(n-2)$$



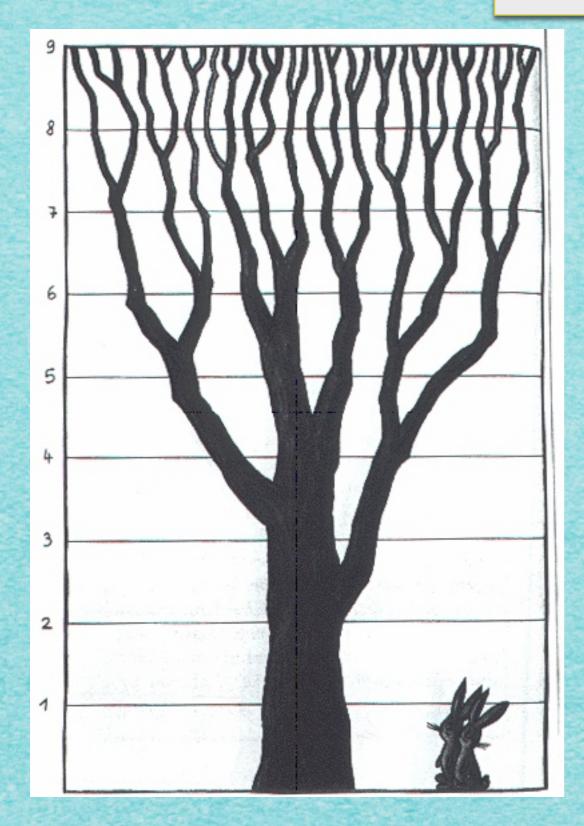




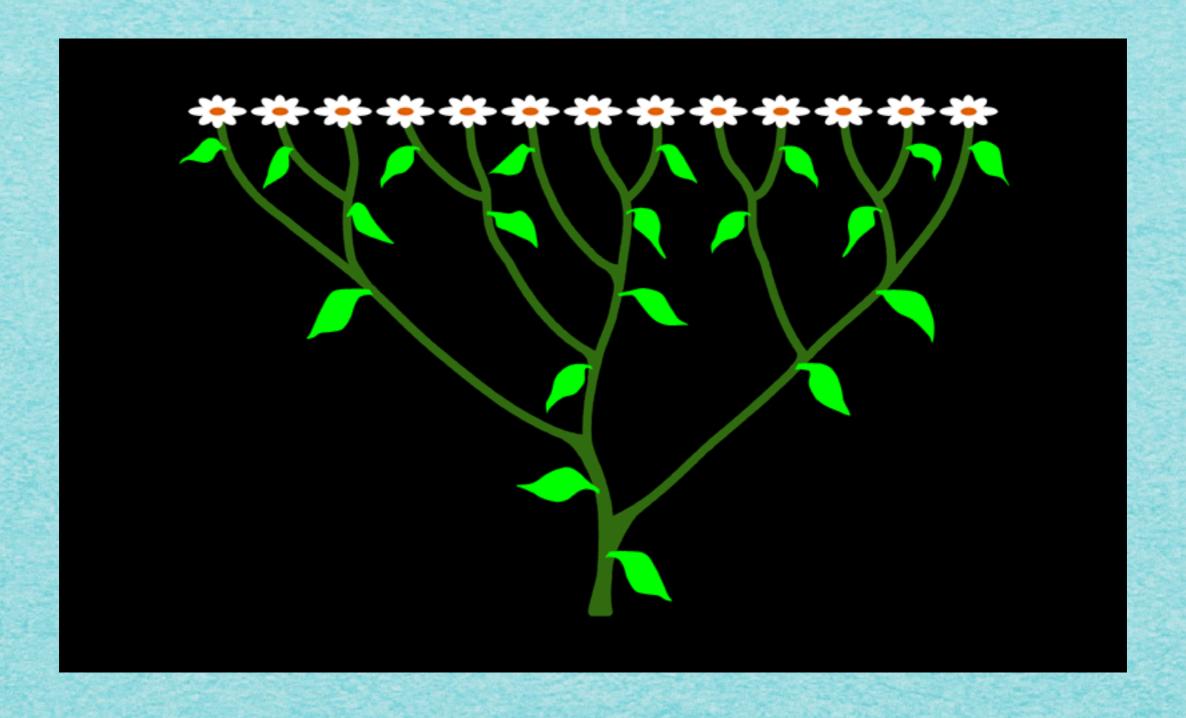
$$F(n)=F(n-1)+F(n-2)$$

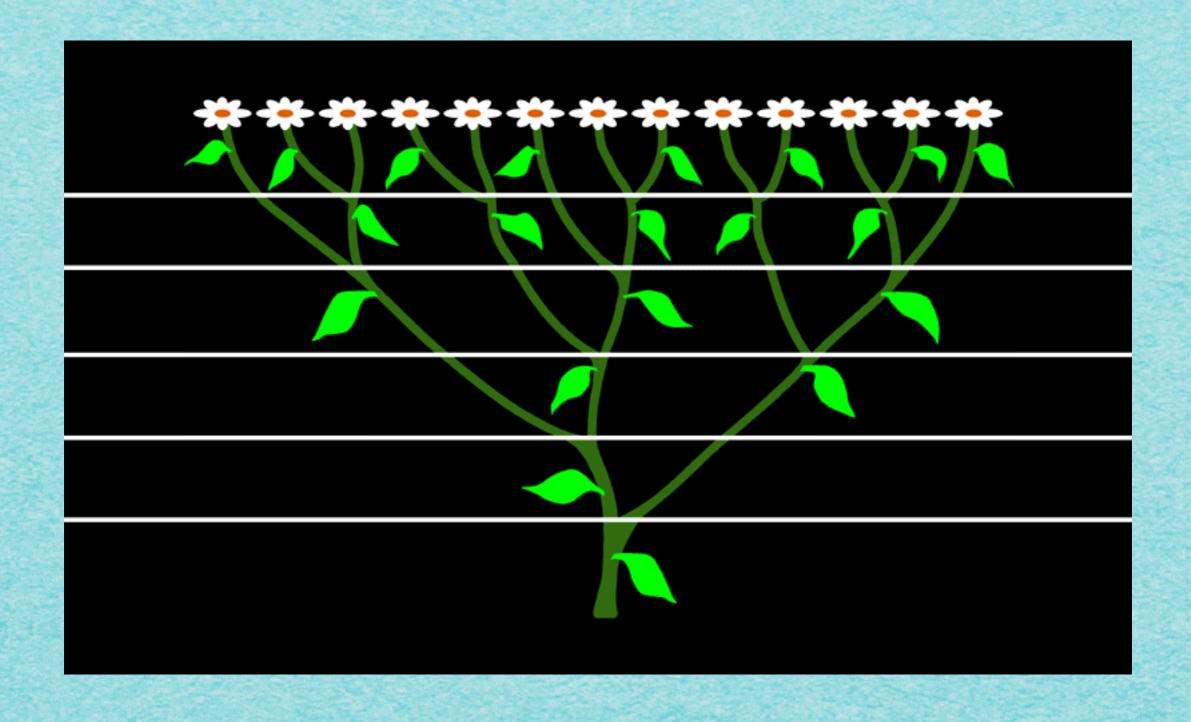


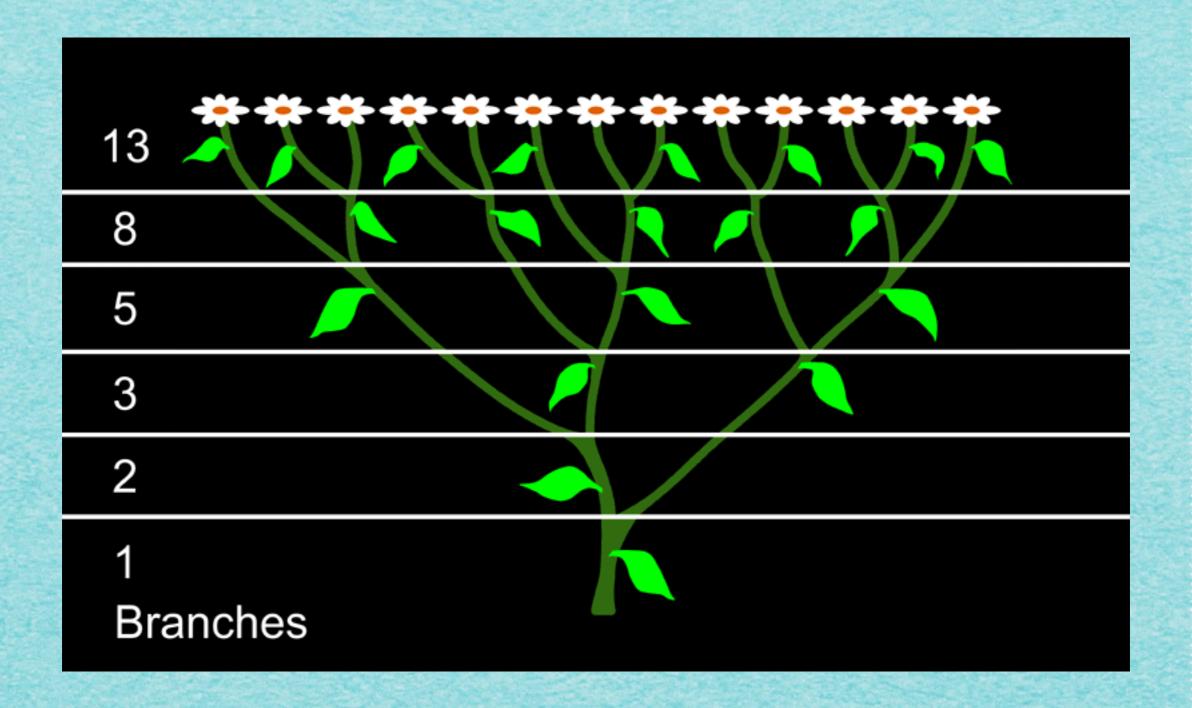
$$F(n)=F(n-1)+F(n-2)$$

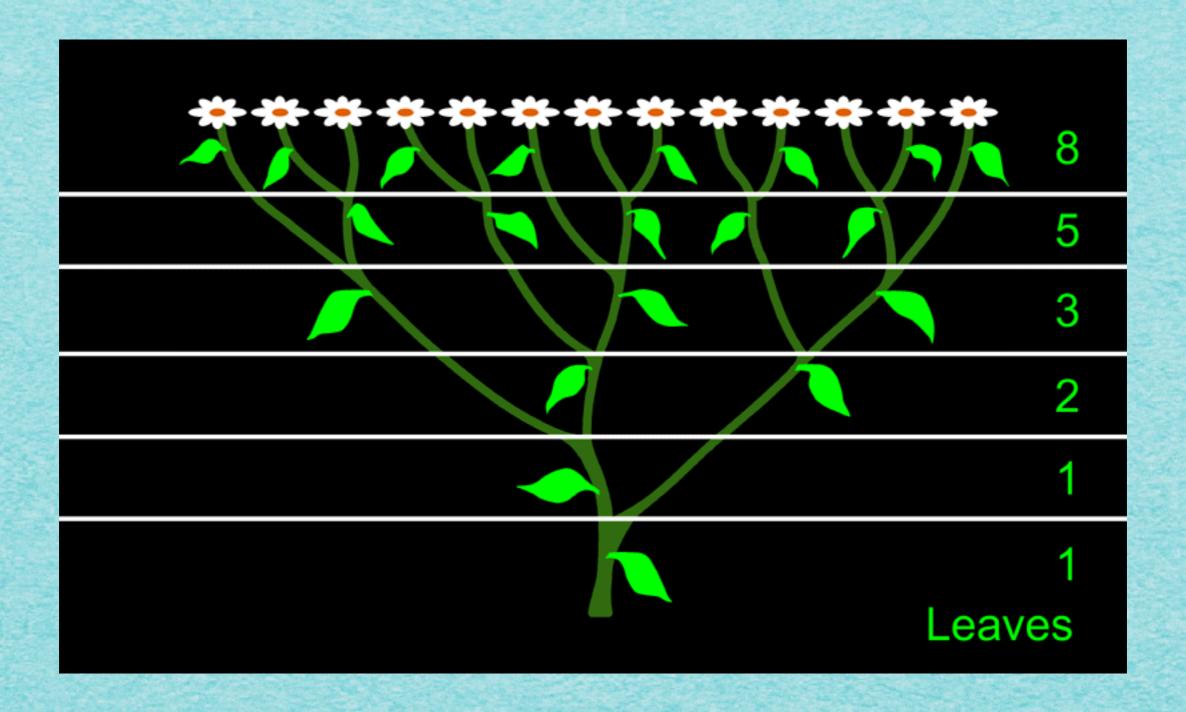




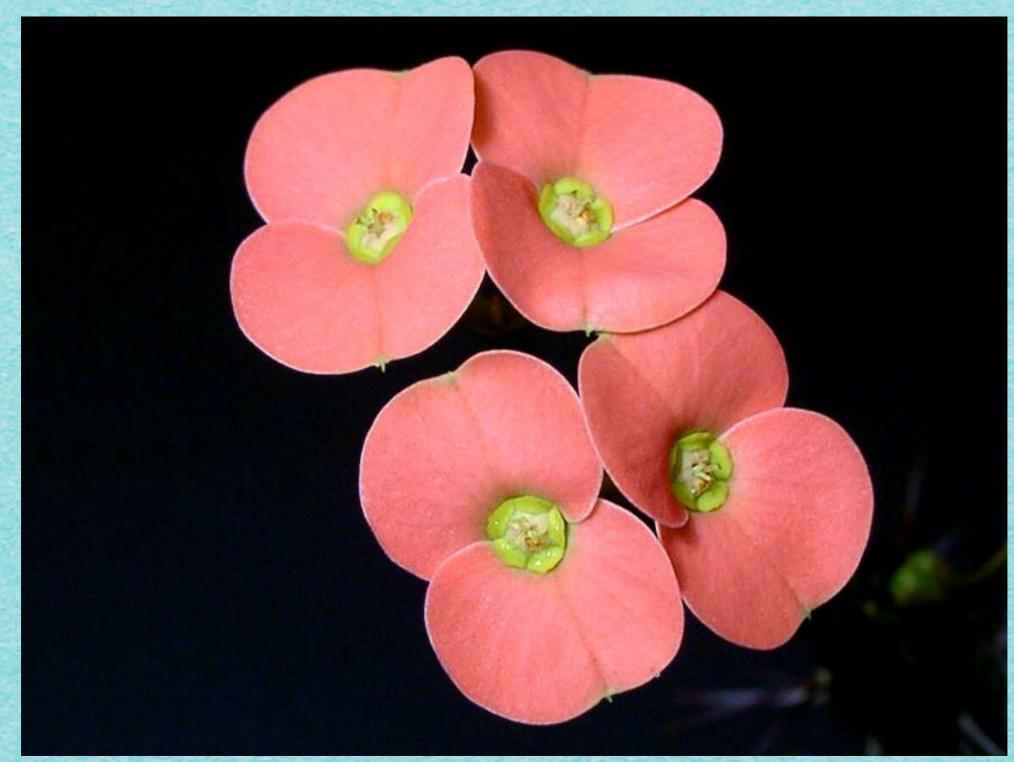








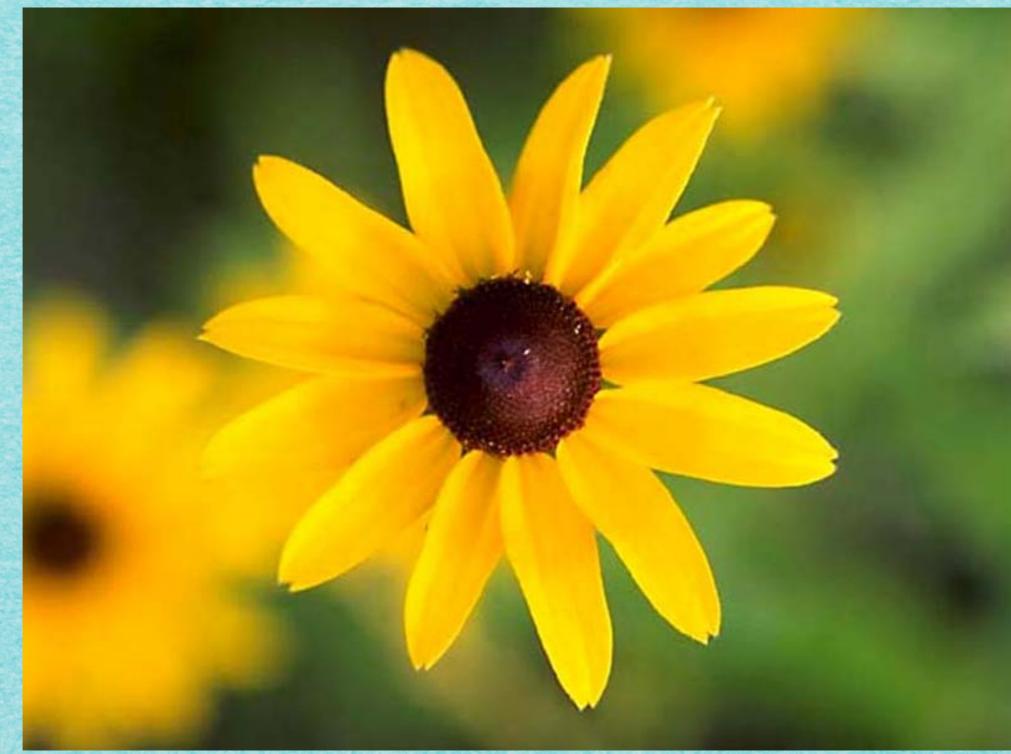


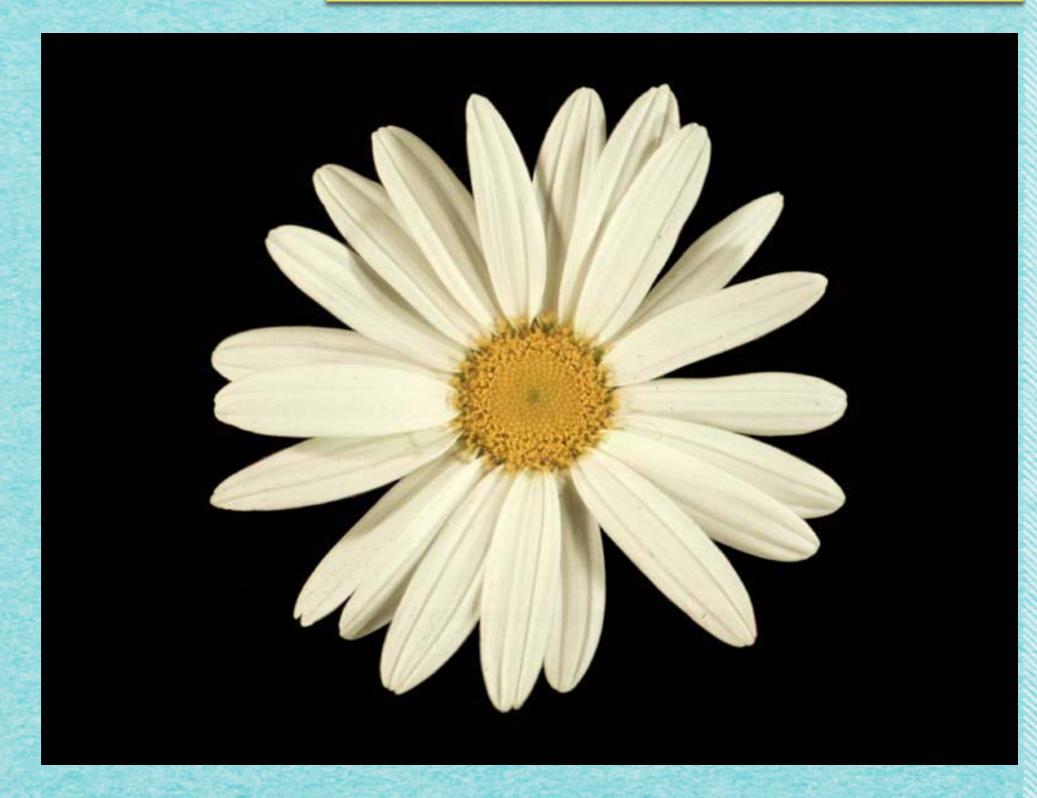




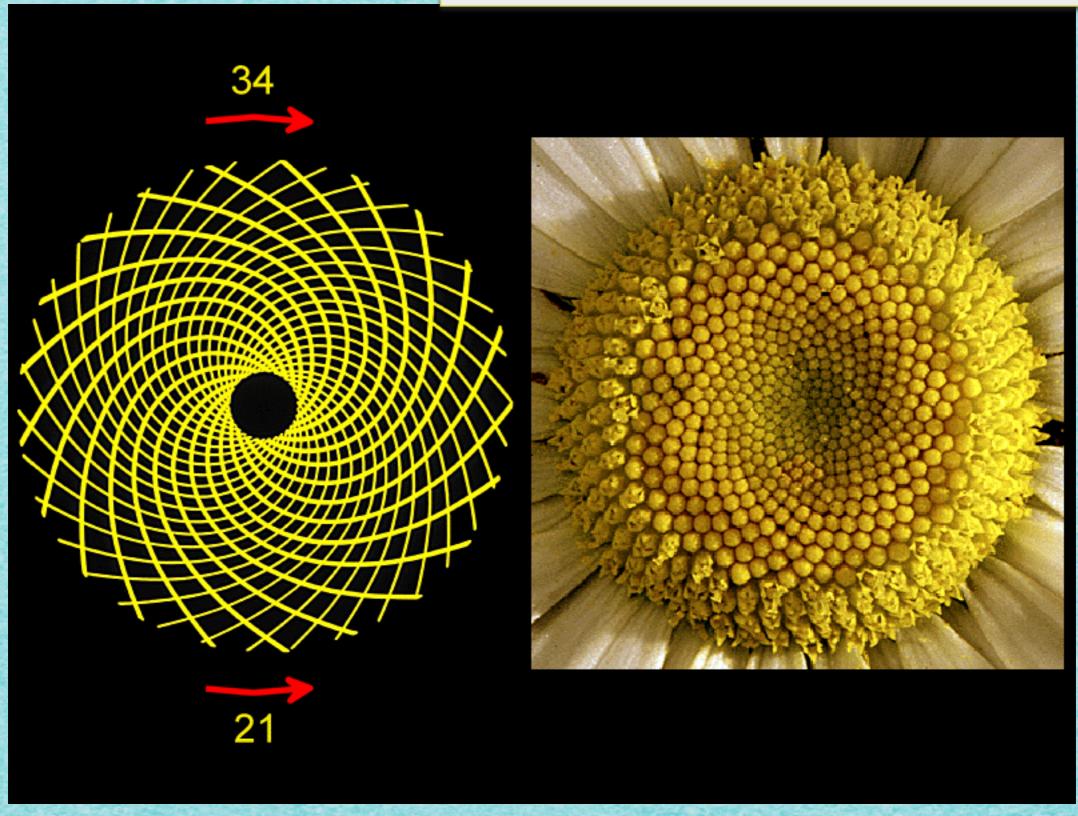






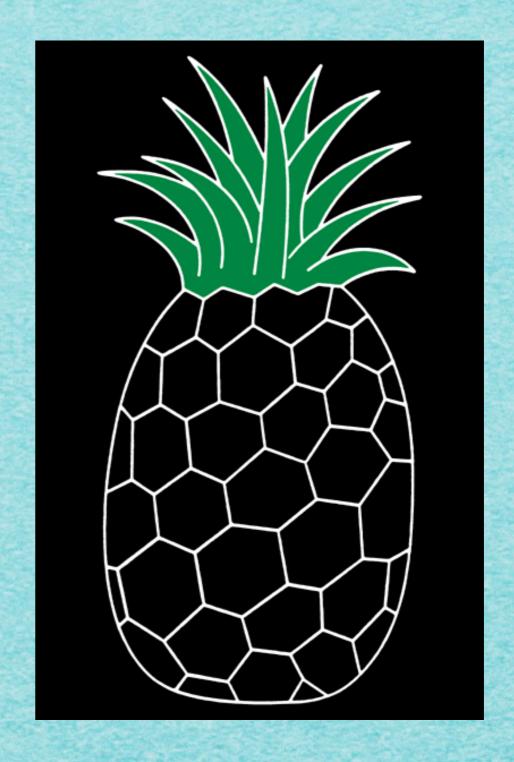




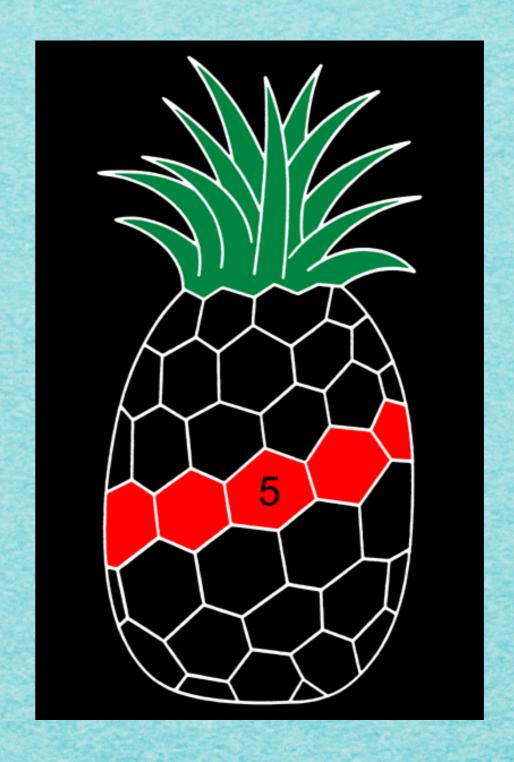




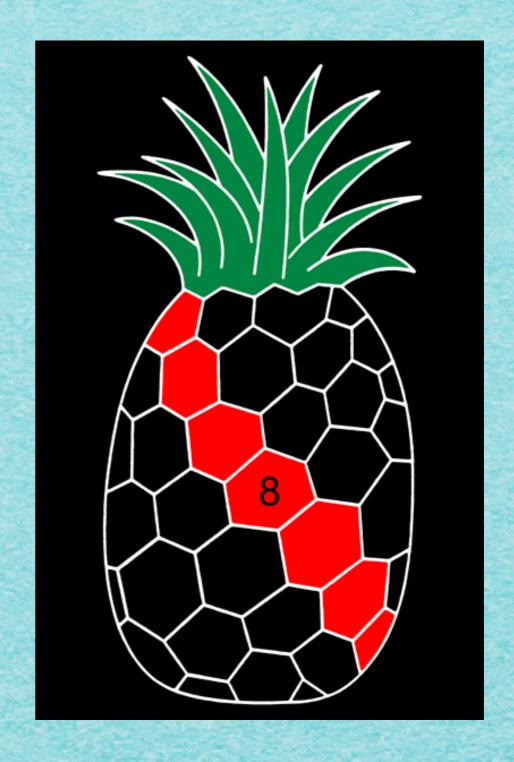




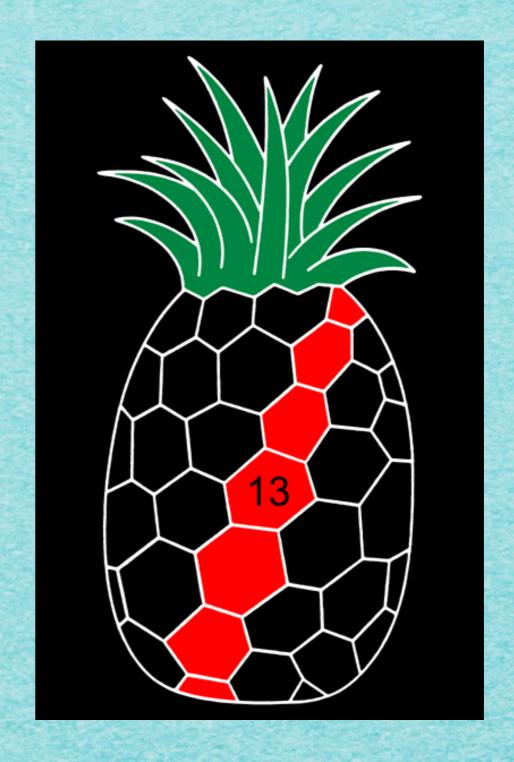




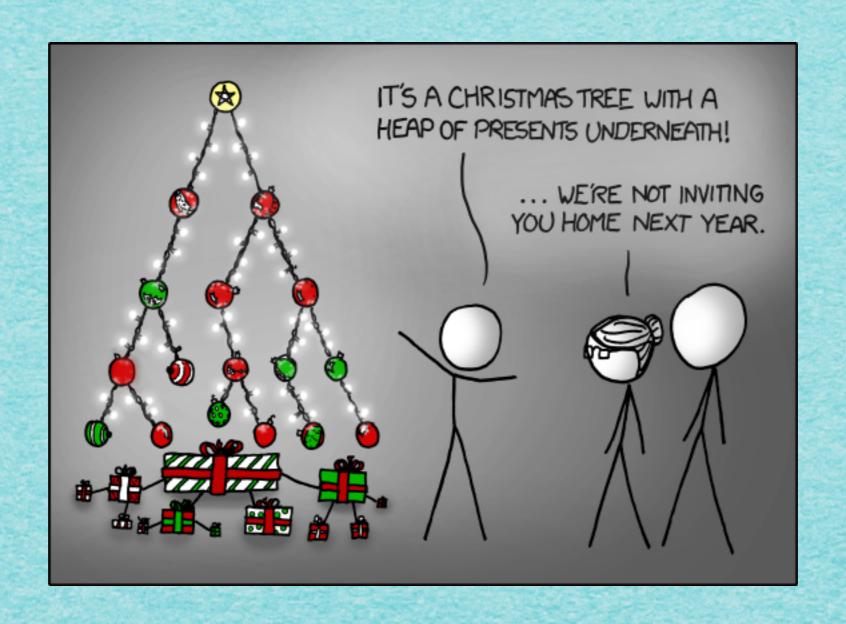








#### Zusammenfassung Kapitel 4!



#### Zusammenfassung Kapitel 4!





## Frohe Weihnachten!



# Mehr demnächst!

s.fekete@tu-bs.de