Automatisches Zeichnen von Graphen

Kap. 2: Zeichnen von Bäumen

Prof. Dr. Petra Mutzel

Lehrstuhl für

Algorithm Engineering LS11

Universität Dortmund

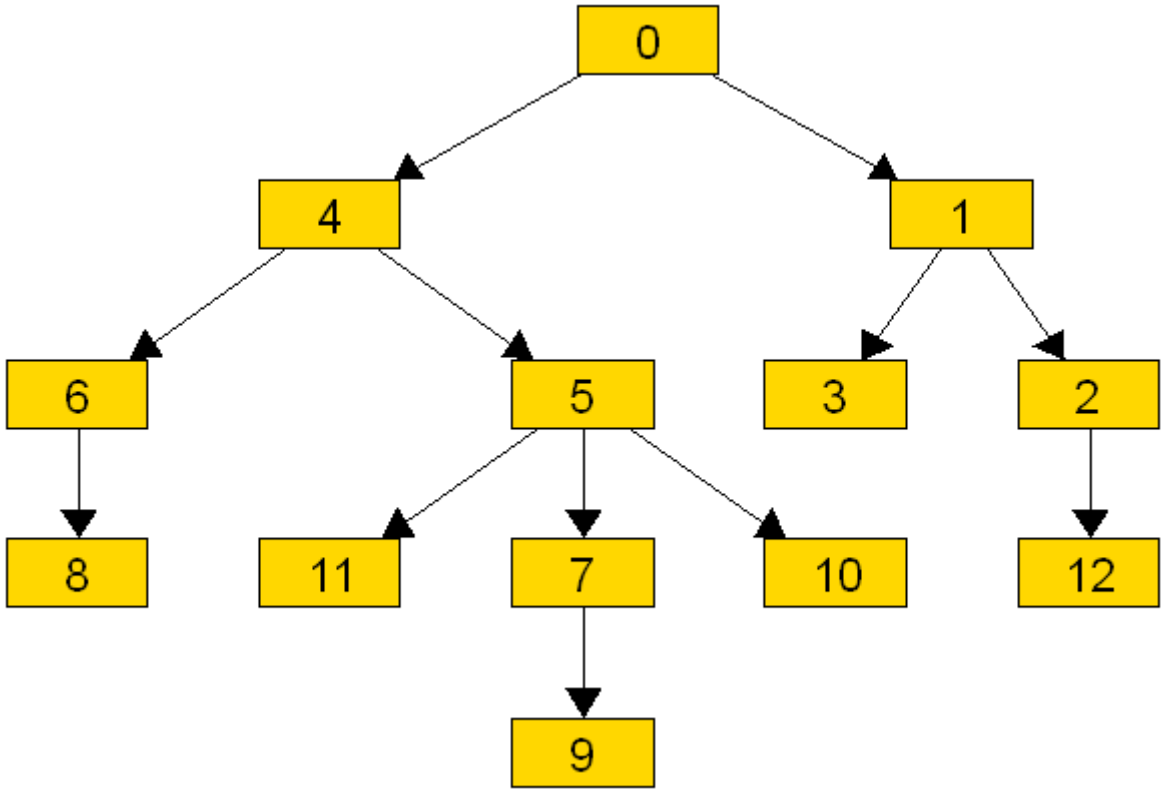
2./3./4. VO WS07/08 16./22./23. Oktober 2007

Literatur für diese VO

- Originalartikel: E.M. Reingold und J.S. Tilford: Tidier Drawings of Trees, IEEE Transactions on Software Engineering SE-7, 1981, 223-228

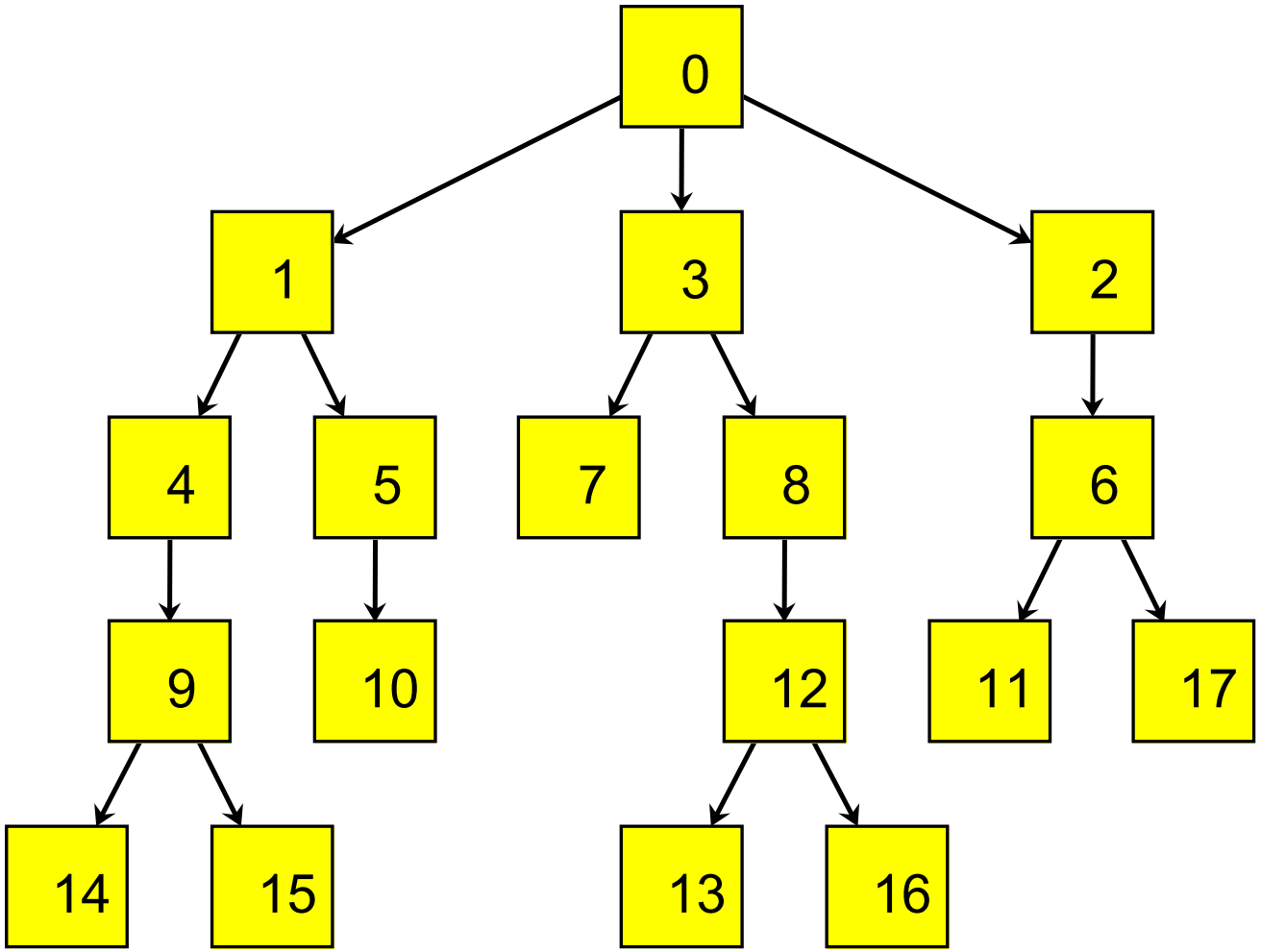
Kapitel 2: Zeichnen von Bäumen

- Orga Diagramme
- Dateistrukturen
- Hierarchien
- Stammbäume
- ...

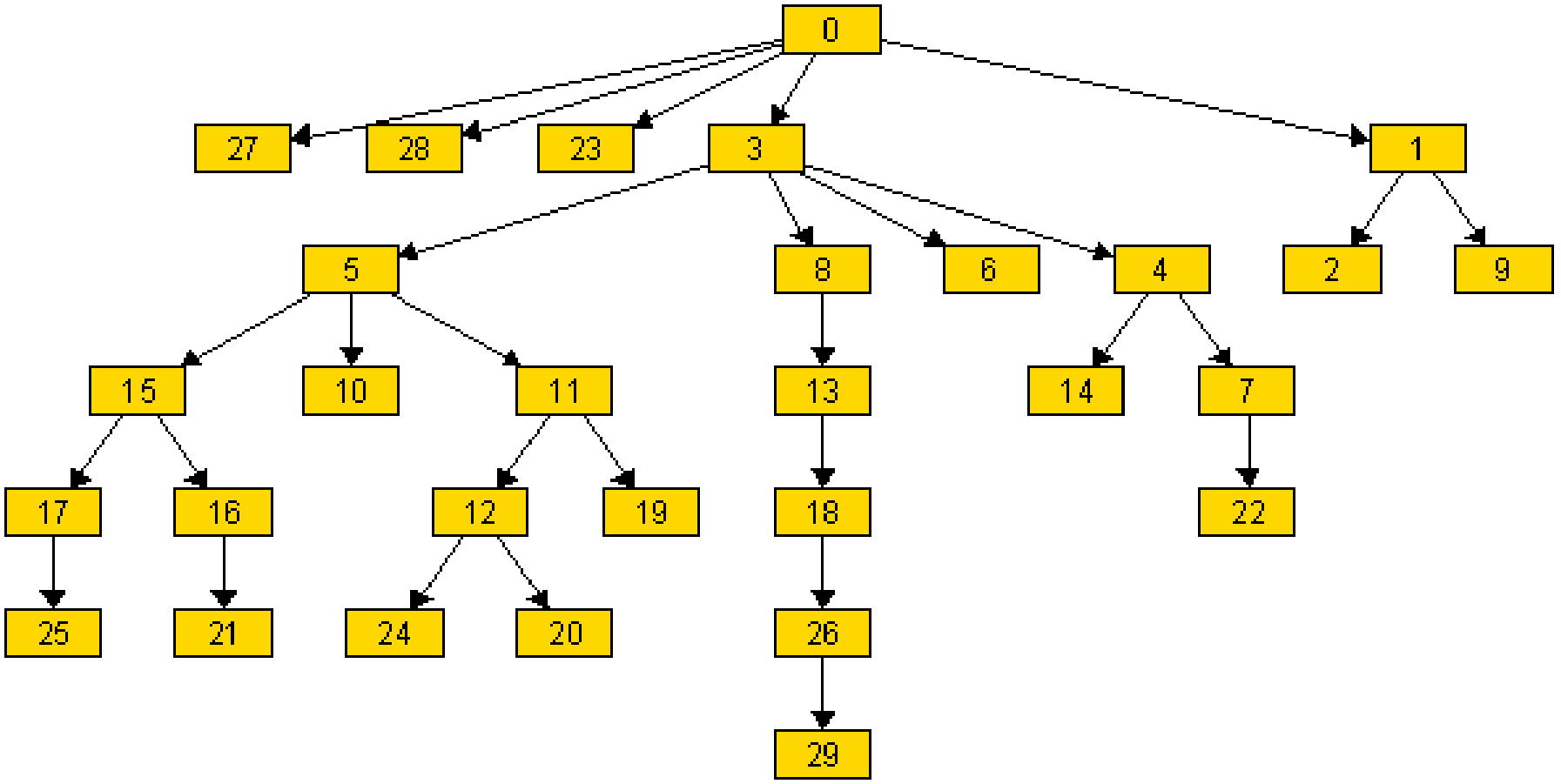


Reingold, Tilford 1981

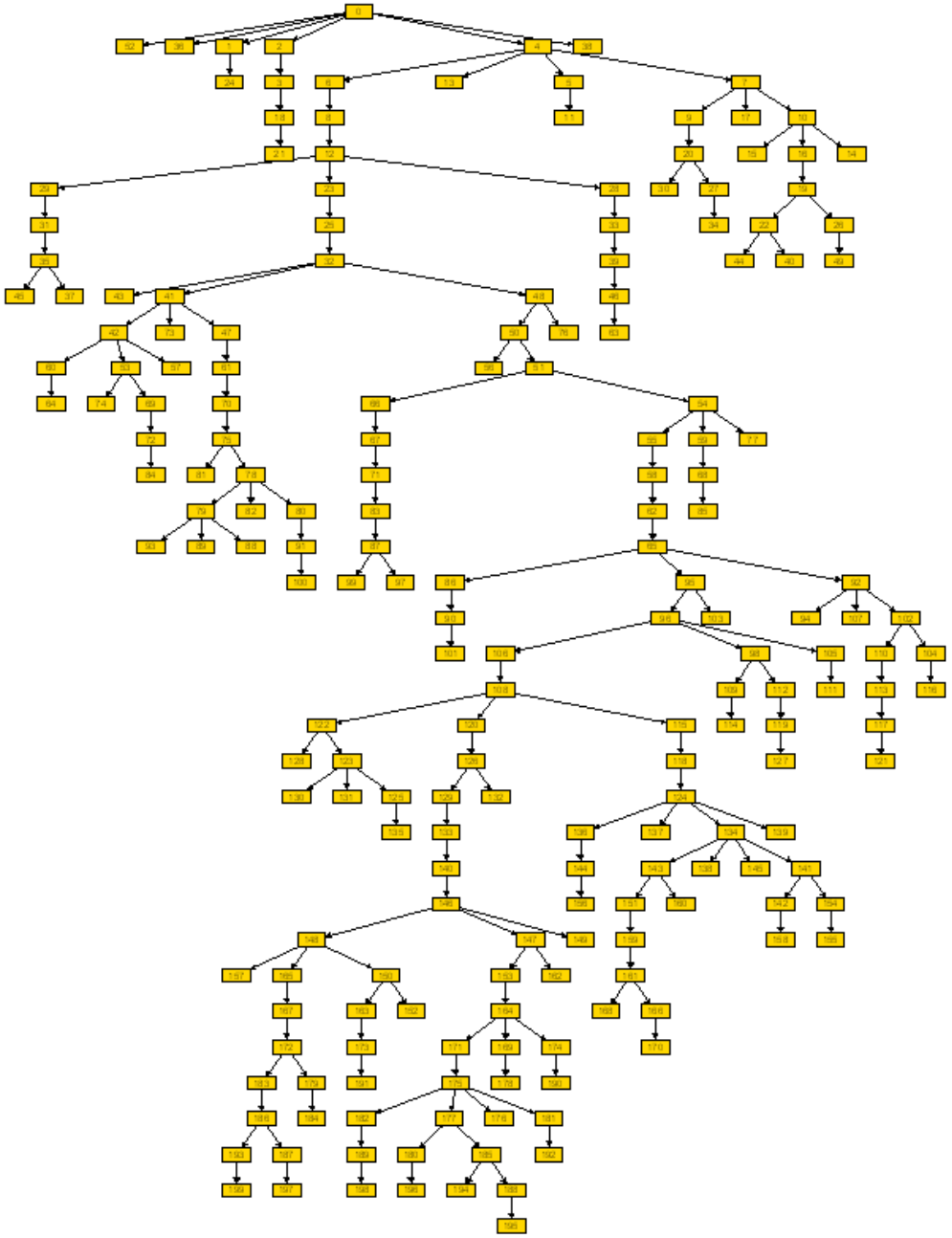
Beispiel: Baumzeichnung



Beispiel: Baumzeichnung



30 Knoten



200 Knoten

Überblick zu Kapitel 2

2.1 Einführung

2.2 Aesthetikkriterien

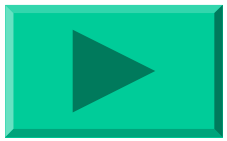
2.3 Algorithmus von Reingold und Tilford (RT)

2.4 Analyse des RT-Algorithmus

2.5 Komplexität

2.6 Lineares Programm

2.7 Alternative Darstellungen für Bäume



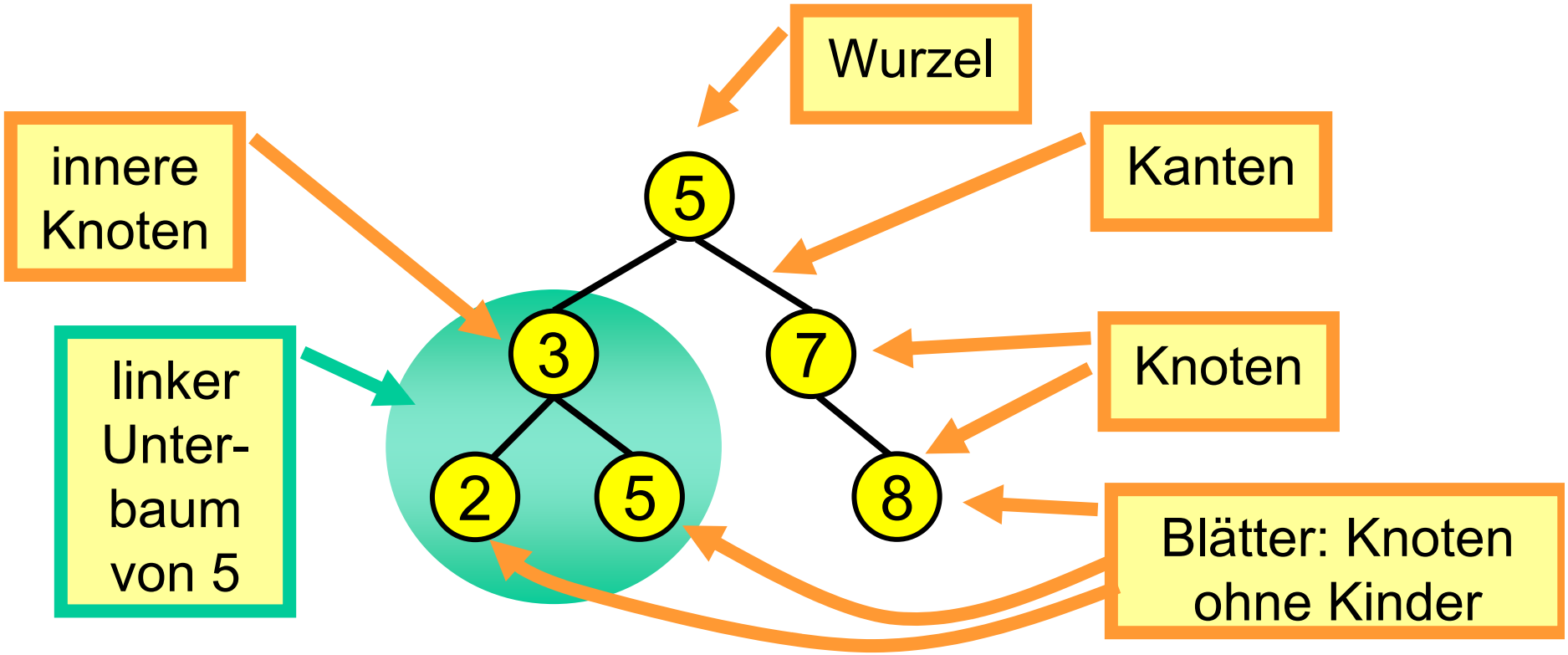
2.1 Einführung

Kurzer Rückblick auf DAP2:

Definitionen

- **Gewurzelter Baum** (rekursiv):
  - entweder leer oder
  - Knoten (Wurzel) mit Verweisen auf mehrere gewurzelte Bäume (Teilbäume) $T_1, T_2, \ldots, T_m$
- **Kind von Knoten v:** Wurzelknoten eines nicht-leeren Teilbaums von v
- **Blatt:** Knoten ohne Kinder

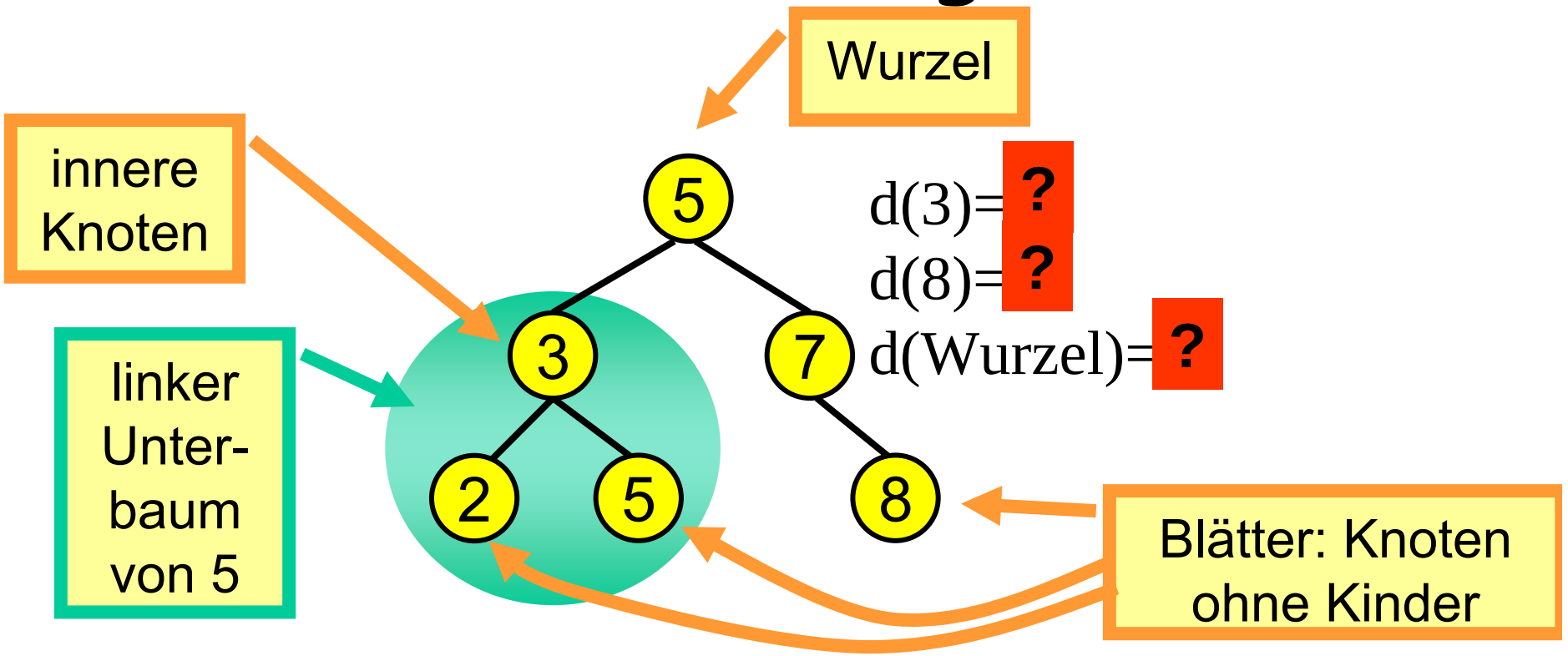
Bezeichnungen: Gewurzelte Bäume



- Tiefe $d(v)$: # der Elter-Knoten bis Wurzel (inkl.)

- 3 ist Wurzel des linken Unterbaums von 5
- 3 ist Elter von 2, 3 ist Elter von 5
- 2 ist linkes Kind von 3, 5 ist rechtes Kind von 3

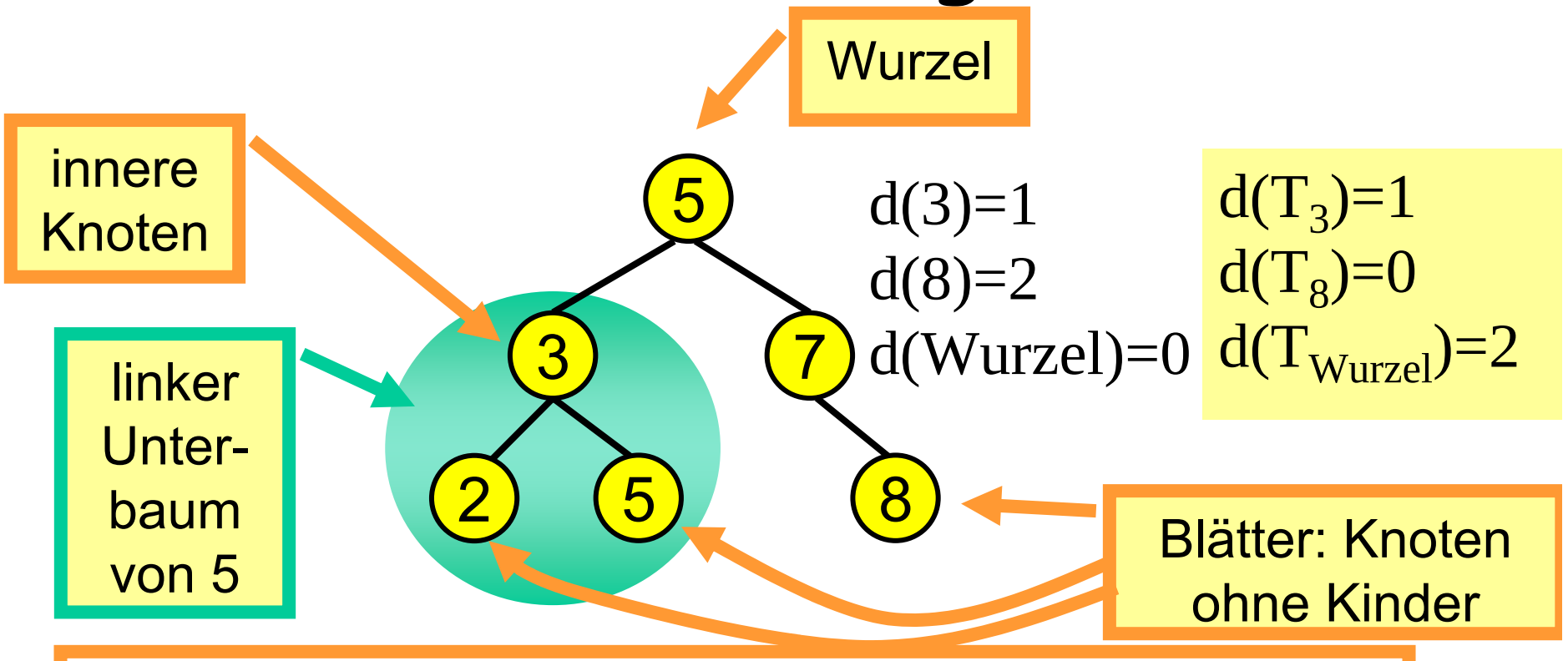
Bezeichnungen



- Tiefe d(v): # der Elter-Knoten bis Wurzel (inkl.)

Ebene: Menge aller Knoten mit gleicher Tiefe

Bezeichnungen



Nachfolger(v): Menge aller Knoten im Unterbaum mit Wurzel v

Höhe $h(T_r)$ eines (Teil-)baumes T_r mit Wurzel r:
max { d(v): v ist Knoten in T_r }

Weitere Definitionen

- Ein gewurzelter Baum heißt **geordnet**, wenn die Reihenfolge der Kinder festgelegt ist
- Ein **binärer** gewurzelter Baum ist ein gewurzelter Baum, bei dem jeder Knoten genau 0,1, oder 2 Kinder hat.
- Ein **geordneter** gewurzelter binärer Baum unterscheidet also zwischen **linkem und rechtem** Kind T_L/T_R bzw. linkem und rechtem Unterbaum.
- Ein binärer Baum heißt **vollständig**, wenn alle Blätter die gleiche Tiefe haben.

HIER: Binärbaum:= geordneter gewurzelter Binärbaum

Binärbäume

- Achtung: Binärbäume sind keine Untermenge von Bäumen

- Dies sind die „gleichen Bäume“ aber unterschiedliche Binärbäume

Implementierung binärer Bäume

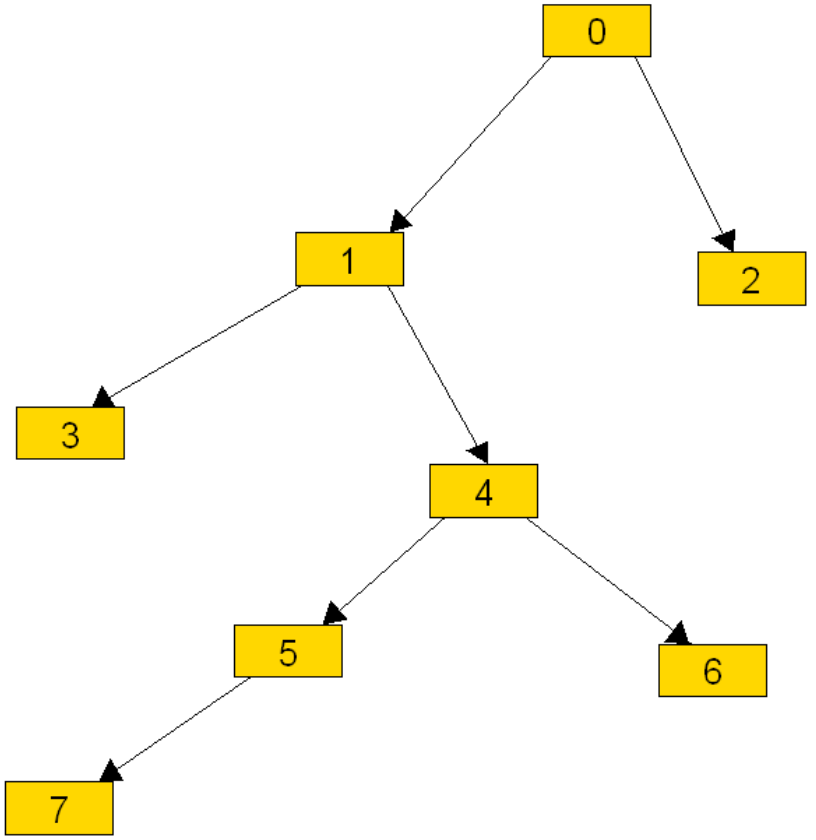
- Realisierung als verallgemeinerte Listen mit bis zu zwei Nachfolgern:
  - **x.info**: zum Schlüssel zu speichernde Daten
  - **x.left**: linkes Kind von Knoten x
  - **x.right**: rechtes Kind von Knoten x

- Zugriff auf den Baum erfolgt über seinen Wurzelknoten root

Traversierungen für binäre Bäume

Inorder-Traversierung:

- Durchsuche rekursiv zunächst den linken Unterbaum,
- dann die Wurzel,
- durchsuche dann den rechten Unterbaum.

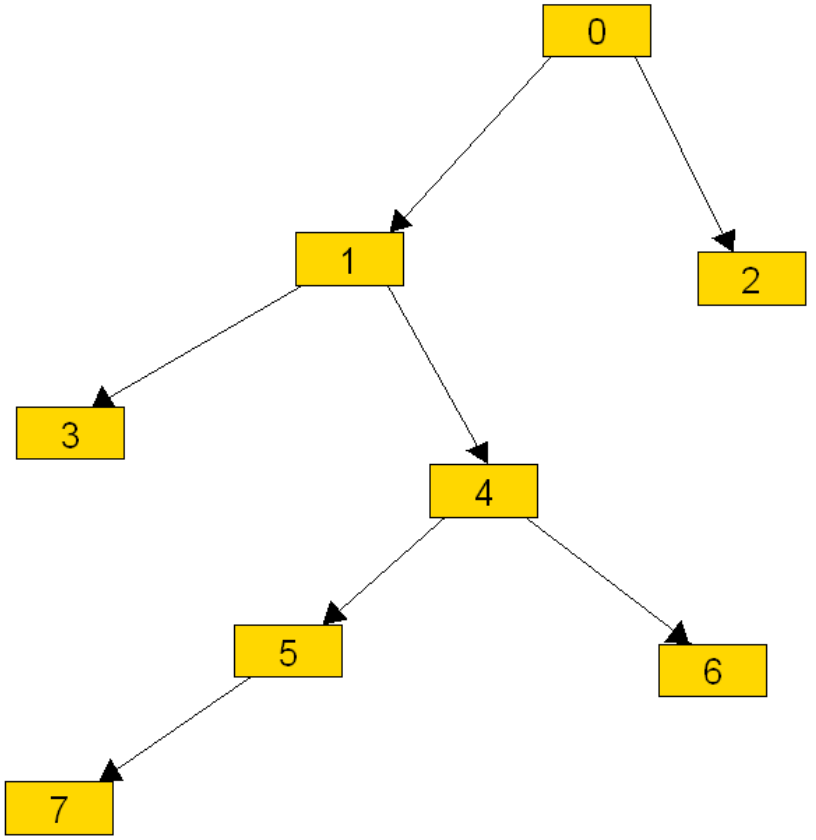


3,1,7,5,4,6,0,2

Traversierungen für binäre Bäume

Preorder-Traversierung:

- Durchsuche rekursiv zunächst die Wurzel,
- dann den linken Unterbaum,
- durchsuche dann den rechten Unterbaum.

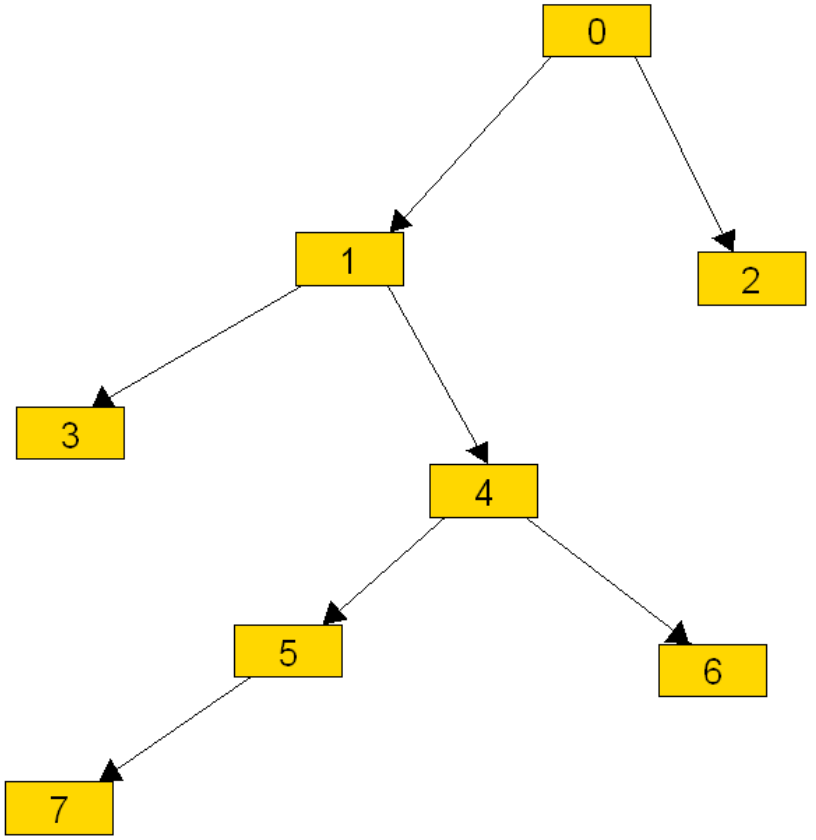


0,1,3,4,5,7,6,2

Traversierungen für binäre Bäume

Postorder-Traversierung:

- Durchsuche rekursiv zunächst den linken Unterbaum,
- dann den rechten Unterbaum,
- durchsuche dann die Wurzel.



3,7,5,6,4,1,2,0

Traversierungen für binäre Bäume

- **Level-Order-Traversierung:** Durchsuche zunächst alle Ebenen von links nach rechts, wobei die Ebenen von oben nach unten besucht werden.

Inorder-Traversierung

```
(1)  Procedure INORDER(p)
(2)  if p ≠ NULL {
(3)      INORDER(p.left)
(4)      Ausgabe von p
(5)      INORDER(p.right)
(6)  }
```

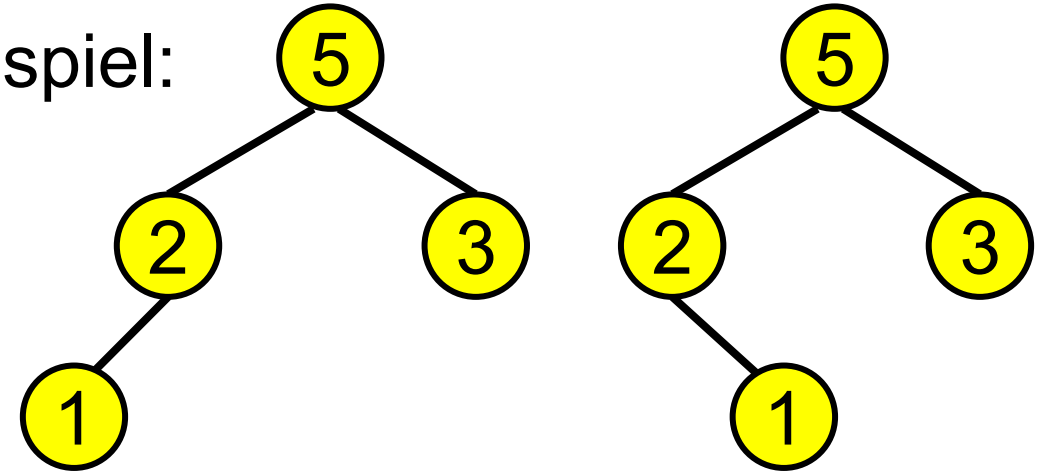
Aufruf: INORDER(root)

Traversierungen für binäre Bäume

- Aus **Inorder-Traversierung und Preorder-Traversierungsreihenfolge** kann der binäre Baum eindeutig rekonstruiert werden, wenn er lauter verschiedene Schlüssel enthält.

- Dies gilt nicht, wenn lediglich Preorder- und Postorder-Reihenfolge gegeben sind.

Beispiel:



Preorder: 5,2,1,3
Postorder: 1,2,3,5

ENDE Rückblick auf DAP2

2.2 Aesthetikkriterien für Binärbäume

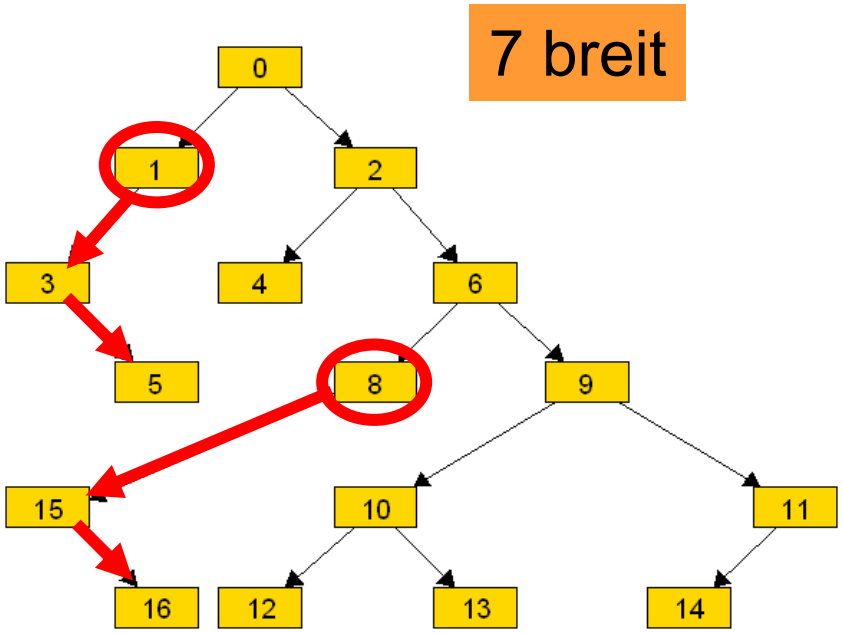
1. „Schichtenzeichnungen“, d.h. Knoten mit gleicher Entfernung zur Wurzel erhalten gleiche Schicht (y-Koordinaten)
2. Linkes Kind soll links vom Elter liegen, rechtes Kind soll rechts vom Elter liegen
3. Elter sollte zentriert über den Kindern liegen
4. Zwei isomorphe Unterbäume sind gleich gezeichnet
5. Ein Baum und sein Spiegelbild erhalten spiegelbildliche Zeichnungen

Weitere Ziele: Gitterzeichnung, d.h. ganzzahlige Koordinaten

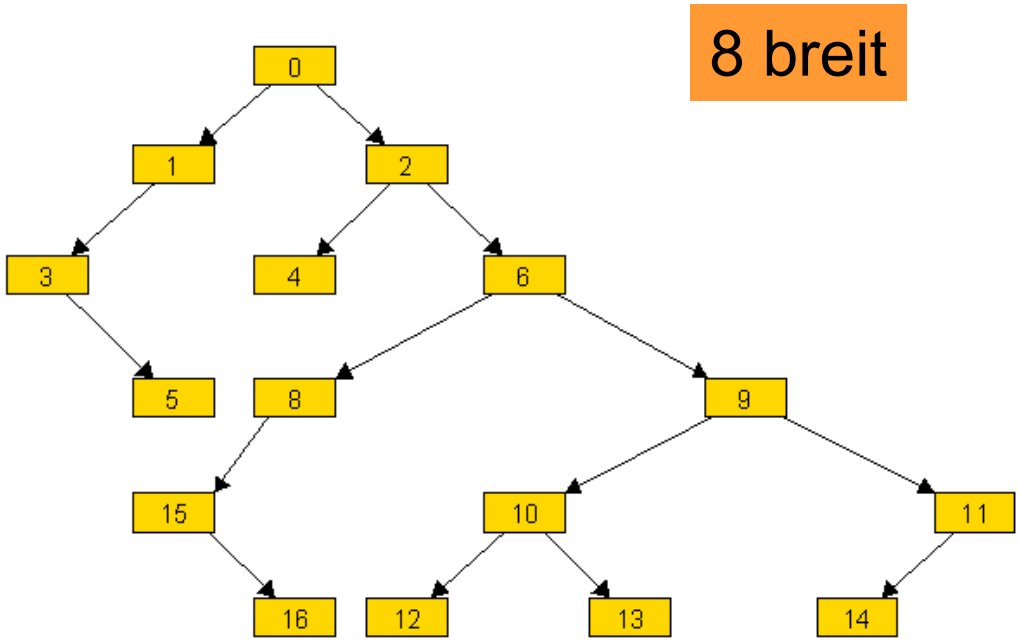
Aufgabe: Bestimme die x-Koordinaten, so dass Weite minimal

Bemerkung

- Aesthetikkriterium 4. widerspricht sich mit der Forderung der minimalen Weite:



minimale Weite



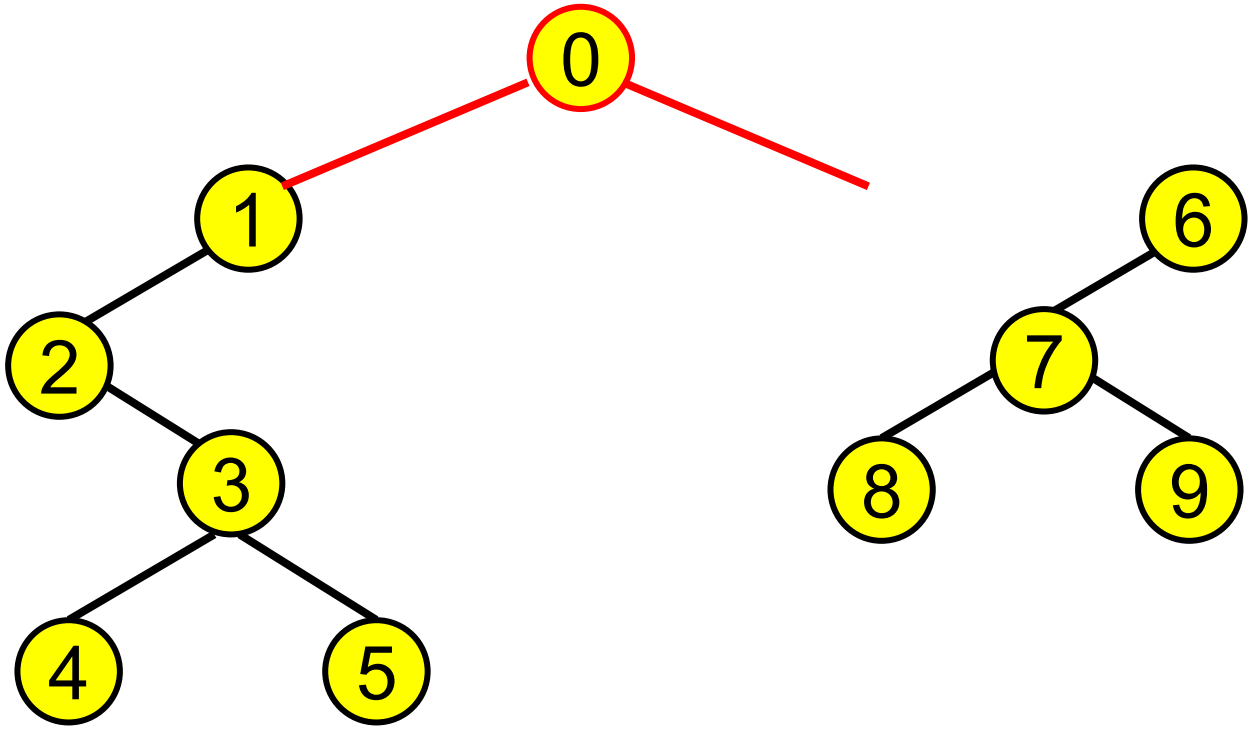
Zeichnung mit Bed. 4

Algorithmus von Reingold und Tilford

Idee: „Postorder“-Traversal

- Zeichne den linken Unterbaum (UB)
- Zeichne den rechten UB
- Füge die Zeichnung der UB so zusammen, dass diese Mindestabstand 2 haben
- Platziere die Wurzel der beiden UB eine Schicht weiter oben genau in der Mitte der beiden Kinder
- Falls ein Elter nur 1 Kind besitzt, platziere Wurzel in horizontaler Distanz 1

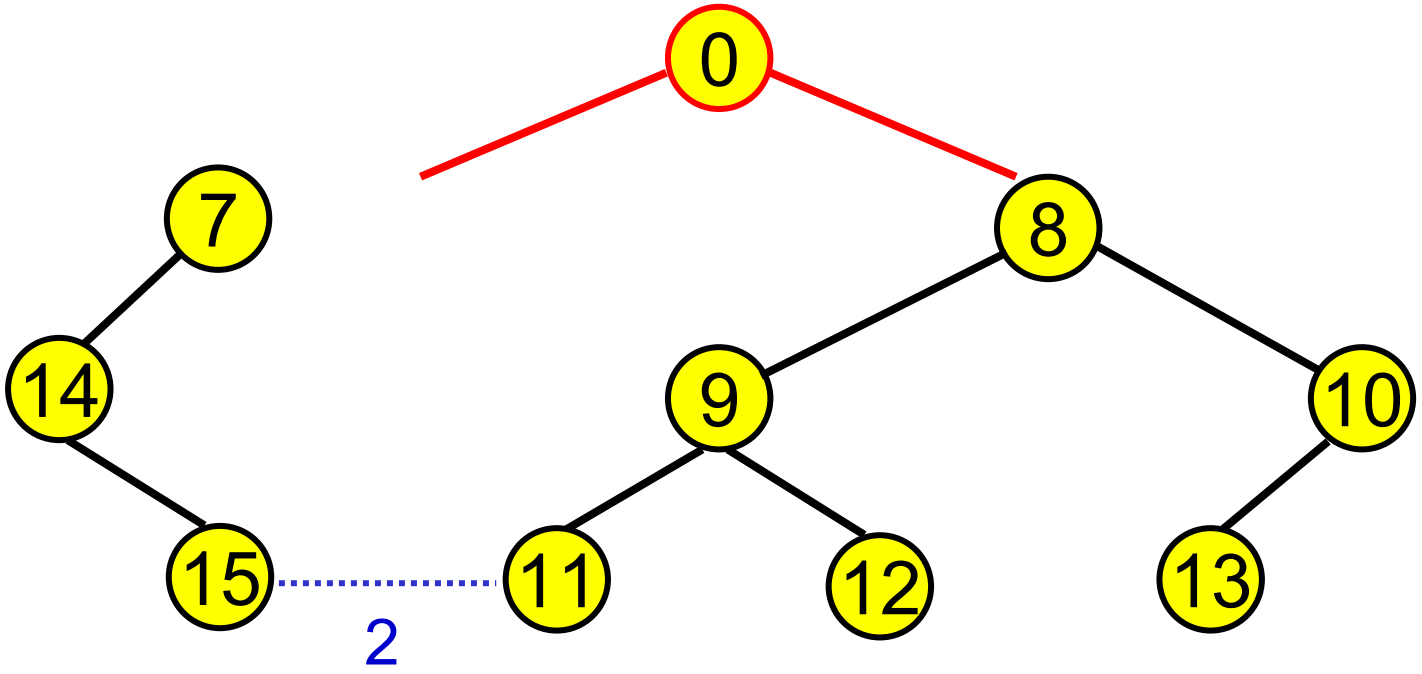
Beispiel des RT-Algorithmus



Probleme: Beispiel

Problem: ungerader Abstand → nicht-ganzzahlige Koordinate

Lösung: rücke rechten UB um 1 weiter nach rechts

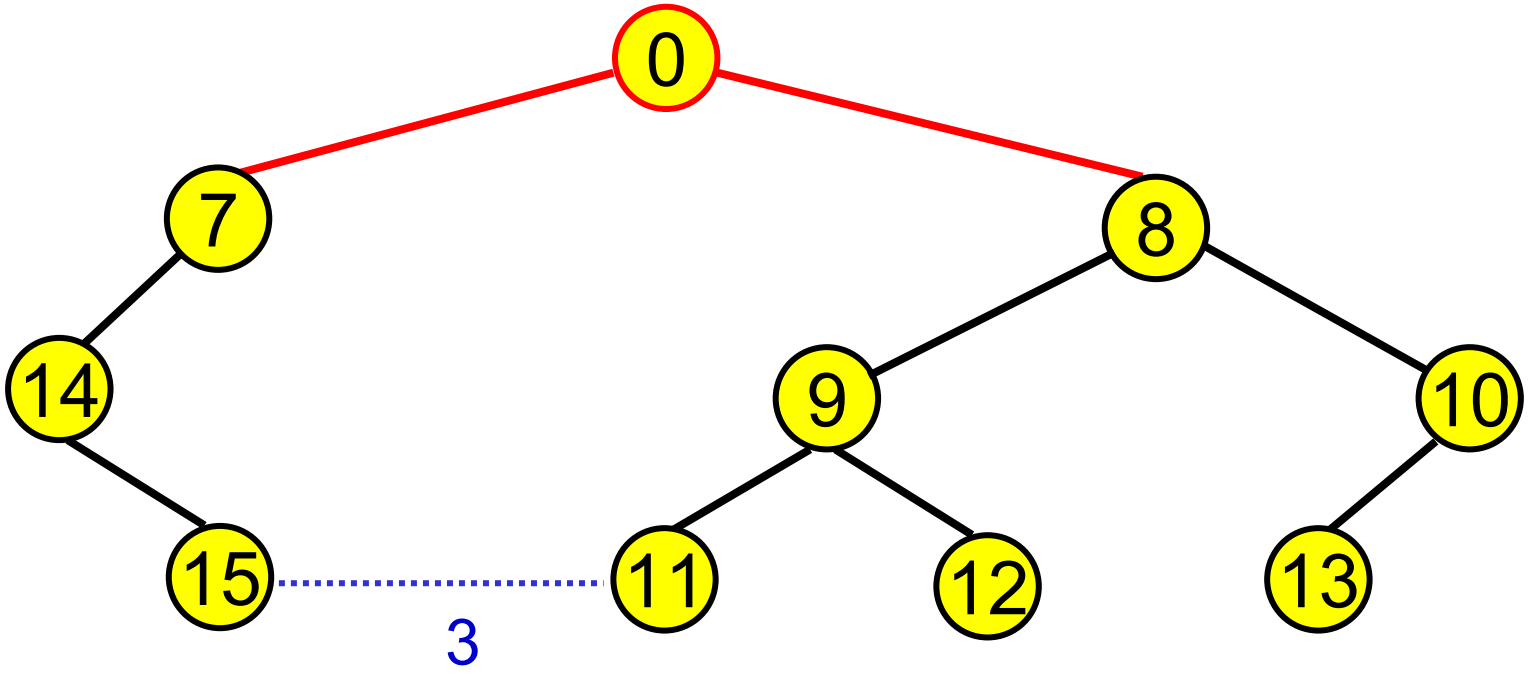


Beispiel des RT-Algorithmus

Problem: ungerader Abstand → nicht-ganzzahlige Koordinate

Lösung: rücke rechten UB um 1 weiter nach rechts

Problem: Weite größer als notwendig

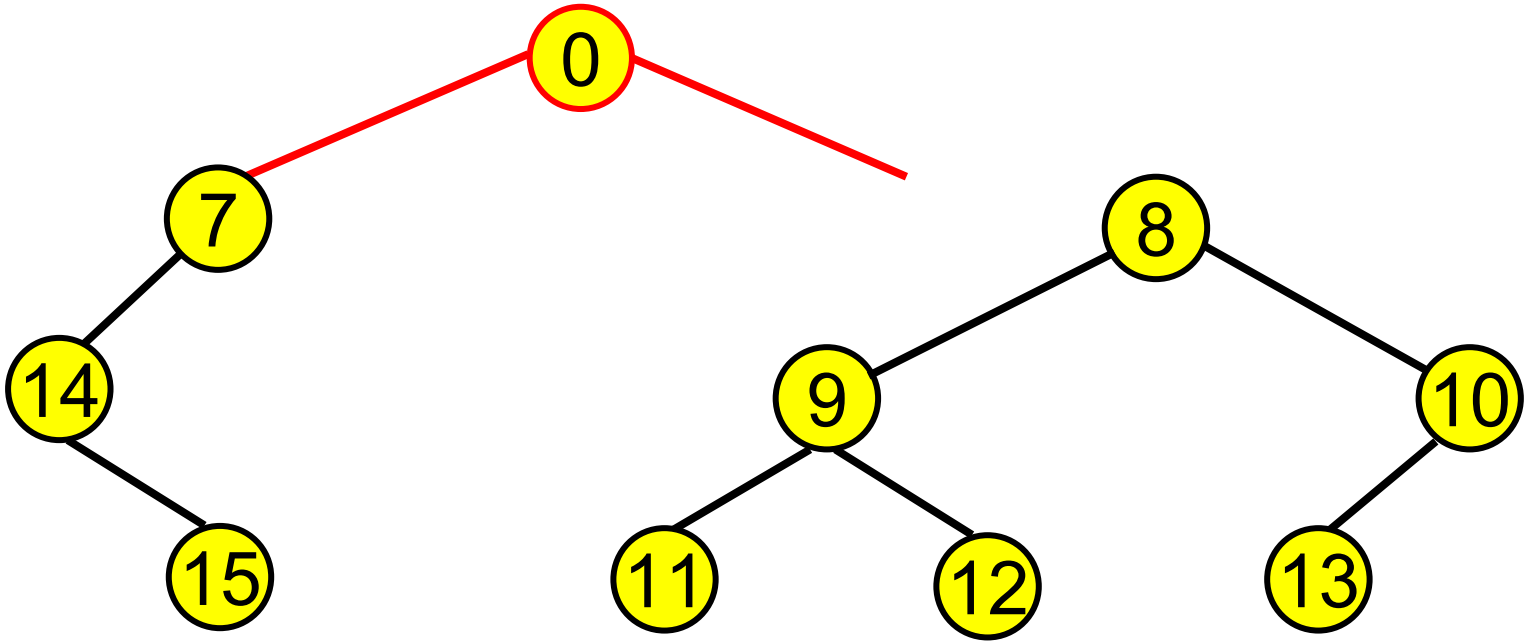


Beispiel des RT-Algorithmus

Problem: ungerader Abstand → nicht-ganzzahlige Koordinate

Lösung A: rücke rechten UB um 1 weiter nach rechts

Problem: Weite größer als notwendig



Lösung: Lösung A in Verbindung mit separiere mit Distanz 1

Algorithmus von Reingold und Tilford

Ausgefeiltere Idee: „Postorder“-Traversal

- Zeichne den linken UB
- Zeichne den rechten UB
- Füge die Zeichnung der UB so zusammen, dass diese Mindestabstand **1** haben
- Platziere die Wurzel der beiden UB eine Schicht weiter oben genau in der Mitte der beiden Kinder (falls der Abstand ungerade, dann schiebe rechten UB um 1 weiter nach rechts)
- Falls ein Elter nur 1 Kind besitzt, platziere Wurzel in horizontaler Distanz 1

Algorithmus Idee von RT

Genauer:

(1) Starte mit Wurzel W_L und W_R

(2) Bestimme die minimale Distanz

(3) Falls Kinder existieren, dann

(4) Wandere eine Schicht tiefer; Gehe zu (2)

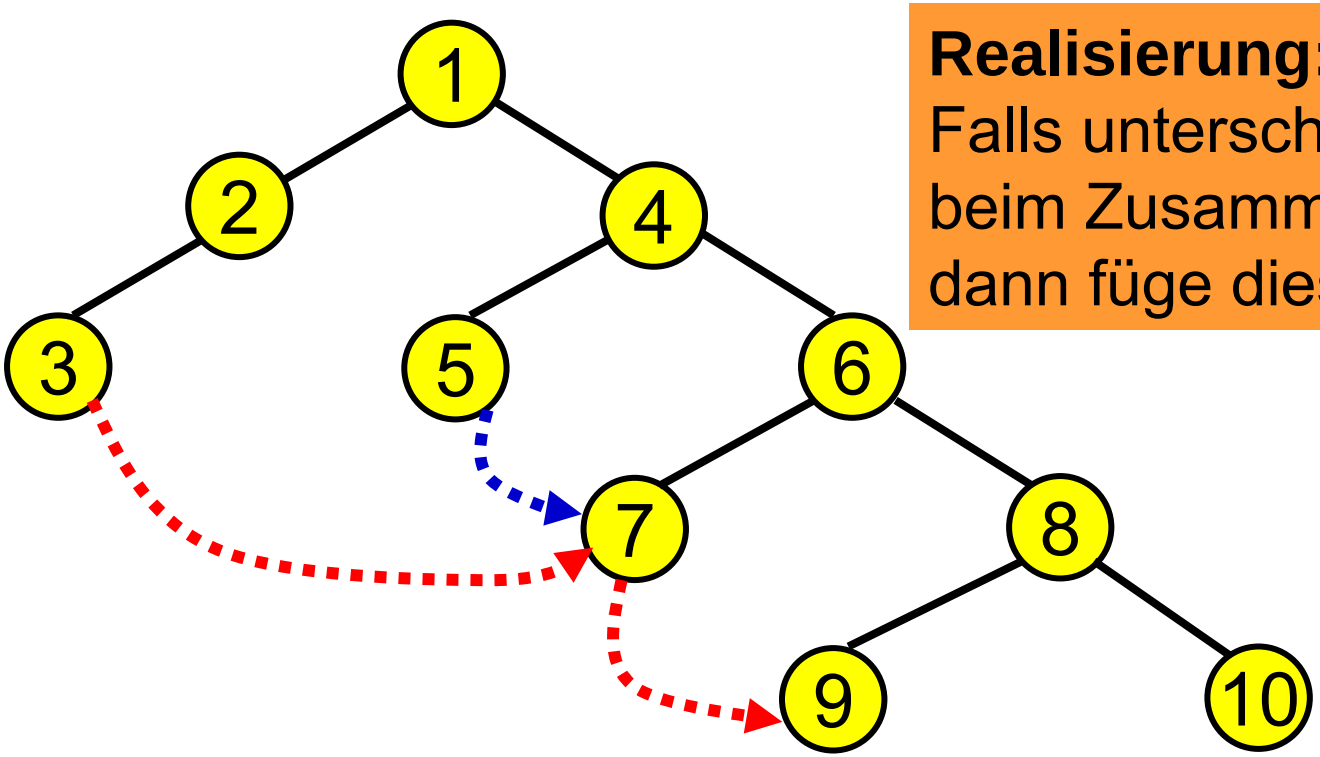
(5) Beim Hochwandern: Fixiere Position der UB relativ zu ihrem Elter, der über ihnen zentriert wird

(6) Am Ende werden die relativen Positionen in absolute Koordinaten umgewandelt mittels preorder

Bemerkung: Eine Zeichnung eines UB ändert sich während des Verfahrens nicht mehr

Problem: Äußerste Kontur des Baumes?

Kontur folgt nicht immer von Elter zu Kind



Realisierung:
Falls unterschiedliche Höhe beim Zusammenschieben, dann füge diesen Extra-Link ein

Beobachtung: In diesen Fällen ist Knoten ein Blatt

Lösung: Benutze normale Kind-Zeiger in Verbindung mit Unterscheidungsbit

Algorithmus von RT: Teil 1

(1) Durchlaufe den geg. Baum in postorder:

(2) // Sei T der aktuelle (Teil-)Baum

(3) Initialisiere: MINDIST=1

(4) Platziere die Wurzeln der UB in MINDIST zueinander

(5) Wandere entlang der rechten Kontur von T_L und der linken Kontur von T_R nach unten und erhöhe dabei MINDIST, falls notwendig

(6) Setze OFFSET(T_L) = -(MINDIST+1)/2 (DIV rundet ab)

(7) Setze OFFSET(T_R) = (MINDIST+1)/2

(8) Bei ungleicher Tiefe von T_L und T_R: Bestimme neue Konturen von T (setze Link zum Rand des tieferen)

Algorithmus von RT: Teil 2

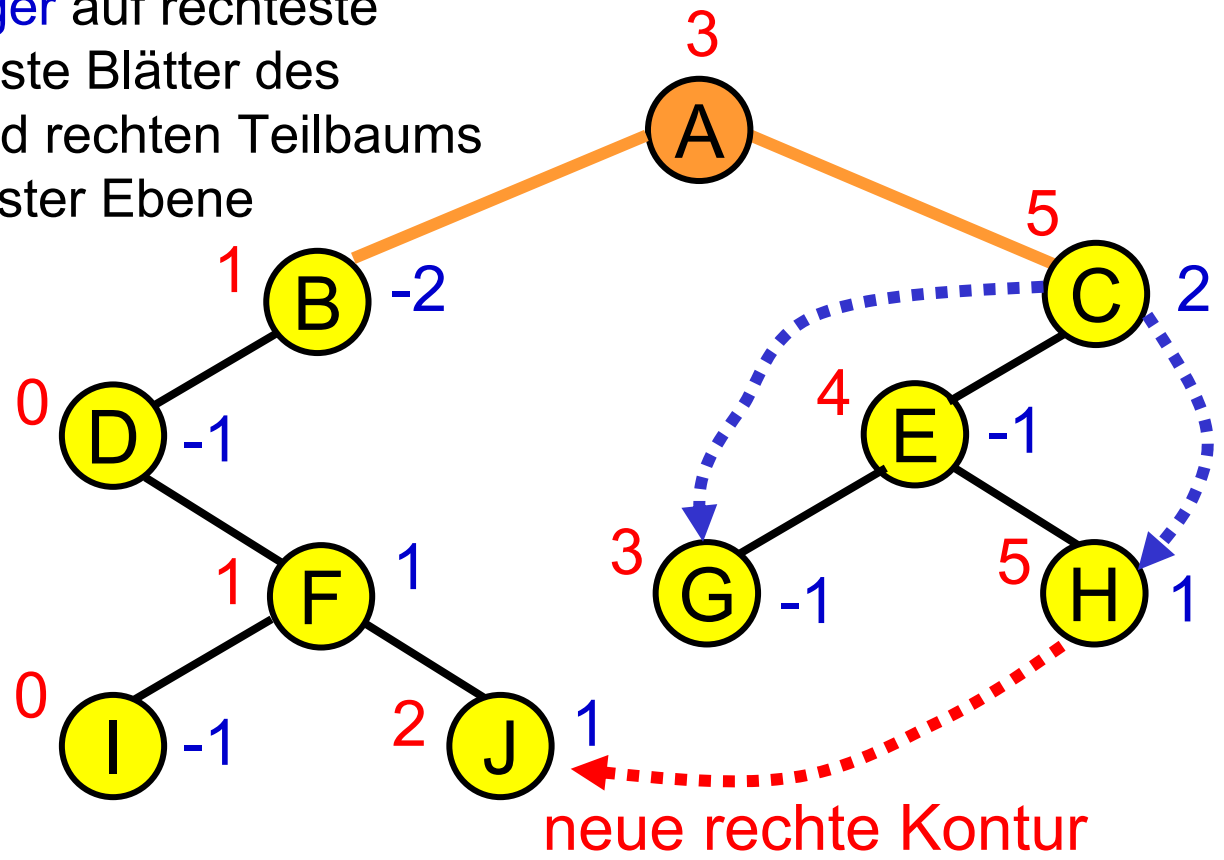
// Jetzt liegen die relativen Koordinaten fest

(9) Laufe durch linken Rand um absoluten OFFSET von Wurzel w zu erhalten $\rightarrow$ XCOORD(w)

(10) Durchlaufe T in Preorder-Sequenz und lege die x-Koordinaten aller Knoten fest

Beispiel für RT Algorithmus

neue Kontur: halte hierfür Extrazeiger auf rechteste und linkeste Blätter des linken und rechten Teilbaums auf unterster Ebene

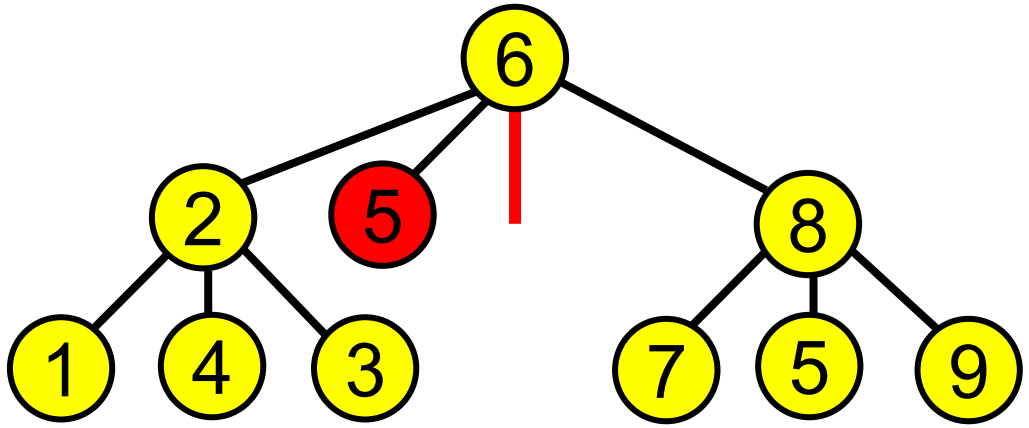


OFFSET-Werte
XCOORD

Beispiel durchrechnen

Verallgemeinerung auf Bäume

- Der RT-Algorithmus kann einfach für allgemeine Bäume verallgemeinert werden
- Problem: Man muss kleine UB zwischen großen UB gleichmäßig aufteilen



- Algorithmus von Walker 1990, Korrektur in
- Buchheim, Jünger, Leipert: Drawing rooted trees in linear time, 2006

2.4 Analyse des RT Algorithmus

- Die gewünschten Eigenschaften (1)-(5) aus 2.2 (Aesthetikkriterien) sind bei einer RT Zeichnung erfüllt. Alle Koordinaten sind ganzzahlig.

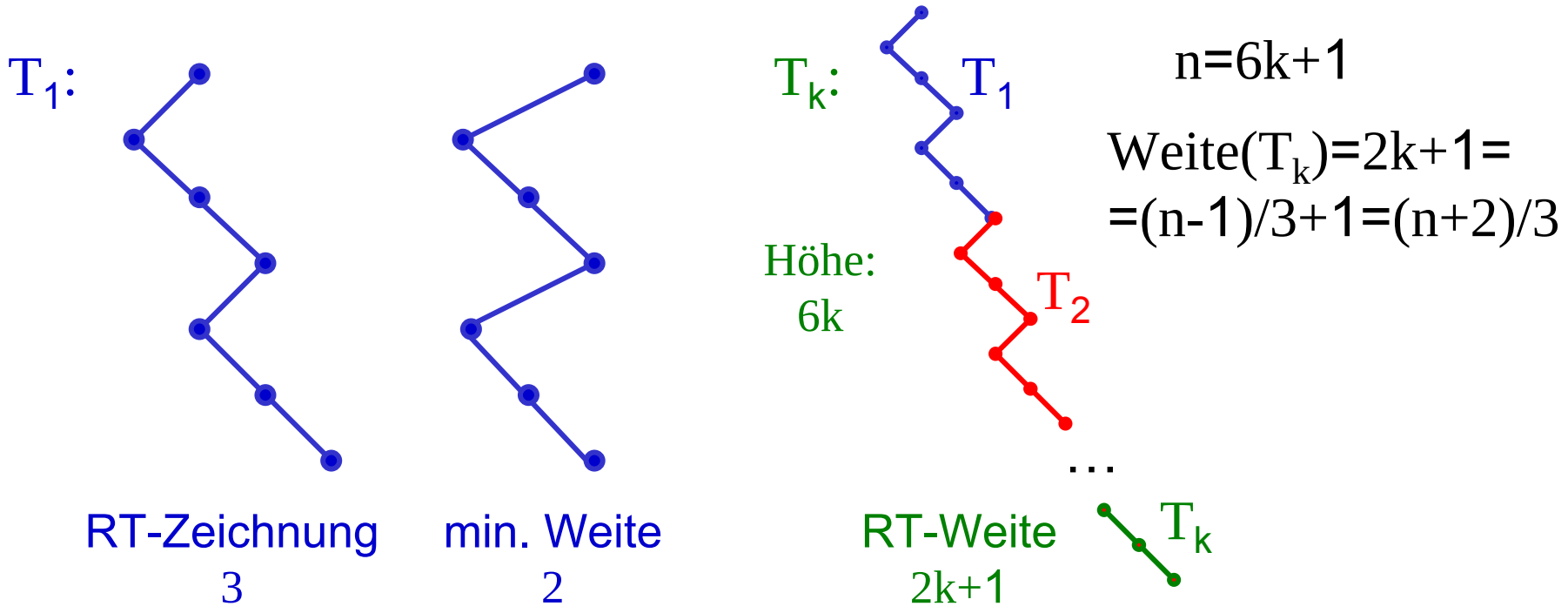
- Ganzzahlige Koordinaten
- (1) Schichtenzeichnungen
- (2) Kinder links/rechts von Elter
- (3) Elter zentriert
- (4) isomorphe Unterbäume gleich
- (5) Baum und Spiegelbild identisch

Wie ist es mit minimaler Weite unter diesen Bedingungen?

Analyse des RT Algorithmus

- Der RT Algorithmus erreicht nicht immer die minimale Weite unter den gewünschten Bedingungen.

- **Schlimmer:** Es existieren Binärbäume T, für die der RT-Algorithmus Zeichnungen mit Weite $(n+2)/3$ produziert, während die optimale Weite gleich 2 ist.



Laufzeitanalyse des RT Algorithmus

- Lemma: Der RT-Algorithmus läuft in linearer Zeit.

h: Anzahl Knoten auf längstem Weg: altes $h(T)+1$

- $F(T)$: Zeit des Algorithmus für Baum T
- T_L: linker UB, T_R rechter UB, $h(T)$ Höhe von T
- Rekursionsgleichung: $F(T) = F(T_L)+F(T_R)+\min(h(T_L),h(T_R))$
- Auflösung (Beh.): $F(T) = n(T)-h(T)$ ohne Konstanten

- Beweis: Induktion über Anzahl der Knoten
- $n(T)=0$: $F(T)=0$ ✓ $n(T)=1$: $F(T)=1-1=0$ ✓
- Aussage sei wahr für Bäume mit $<N$ Knoten, sei $n(T)=N$
- $F(T) = F(T_L) + F(T_R) + \min(h(T_L),h(T_R)) =$

$= (K - h(T_L))+(N-K-1- h(T_R)) + \min(h(T_L),h(T_R)) =$

$= N - (\max(h(T_L),h(T_R)) + 1) = N - h(T) = n(T) - h(T)$

2.5 Komplexität des Baumzeichenproblems

- **Lemma:** Das Problem eine Zeichnung eines Binärbaums mit Weite ≤ W zu erstellen, bei der die Bedingungen (1)-(5) erfüllt sind und (6) die x-Koordinaten ganzzahlig sind, ist NP-vollständig.

Beweis: Reduktion von 3-SAT (s. Bw. in Supowit, Reingold: The complexity of drawings trees nicely, 1983, sowie Akkerman, Buchheim, Jünger, Teske: corrigendum, 2004)

- **Lemma:** Das Problem eine Zeichnung eines Binärbaums mit Weite ≤ W zu erstellen, bei der die obigen Bedingungen ohne (5) und ohne (6) erfüllt sind, ist äquivalent zu Linearer Programmierung.

2.6 Lineares Programm

Variablen

- $x(v)$: für x-Koordinaten von Knoten v
- und 2 zusätzlichen Variablen L und R (für linken und rechten Rand) mit $L \leq x(v)$ für alle v und $R \geq x(v)$ für alle v.

- **Zielfunktion:** minimiere R-L
- unter den Nebenbedingungen: ???

2.6 Lineares Programm

- **Zielfunktion: minimiere R-L**
- unter den Nebenbedingungen:
- zu 1.: ✓
- zu 2.: $x(v.right) - x(v) \geq 1$ und $x(v) - x(v.left) \geq 1$ für alle v
- zu 3.: $x(v.right) - x(v) = x(v) - x(v.left)$
- Abstände: $x(v_2) - x(v_1) \geq 1$ für alle v_1, v_2 auf derselben Schicht mit v_2 level-order Nachfolger von v_1
- zu (4): Bestimmung aller Paare von isomorphen UB: Sei $\{v_1, v_2, \ldots, v_k\}$ Menge von Wurzeln isomorpher UB:
  $x(v_i.right) - x(v_i) = x(v_{i+1}.right) - x(v_{i+1})$ für alle i, $1 \leq i < k$, für die $v_i.right$ existiert ODER dasselbe für $v_i.left$:
  $x(v_i) - x(v_{i+1}.left) = x(v_{i+1}) - x(v_{i+1}.left)$

Aufstellung via preorder

level-order

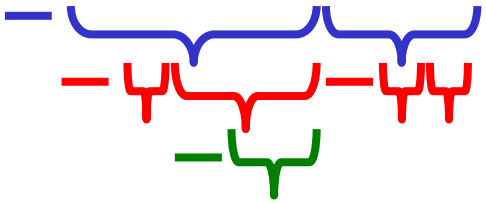
???

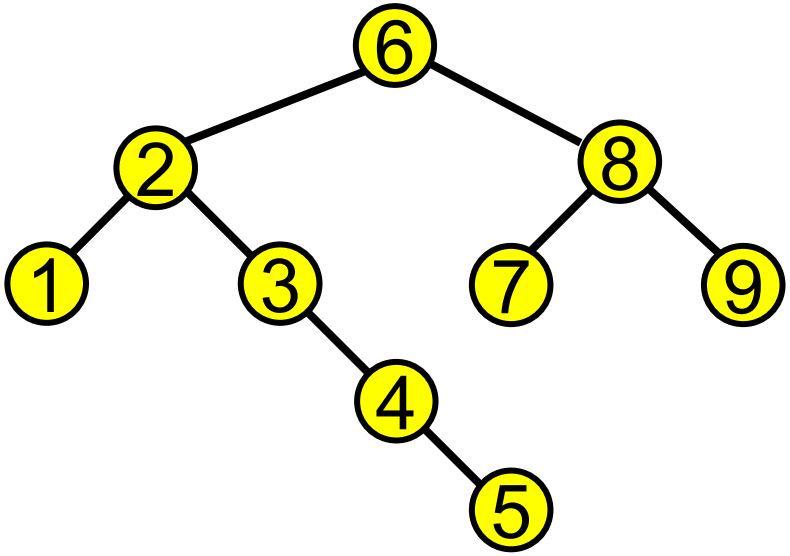
Lineares UGLS und Lineare Zielfunktion → polynomielle Zeit

Wie bestimmt man die isomorphen UB?

- Merke die Größe (Anzahl der Knoten) jeden UB
- Jeder UB T_k der Größe k erhält eine Nummer: Rank(T_k), so dass isomorphe UB gleichen Rang erhalten.
- Aufbau von Rank: 1. Stelle: Wurzel, Stellen danach, die kleiner sind als Wurzel hängen am linken UB, die restlichen am rechten UB.

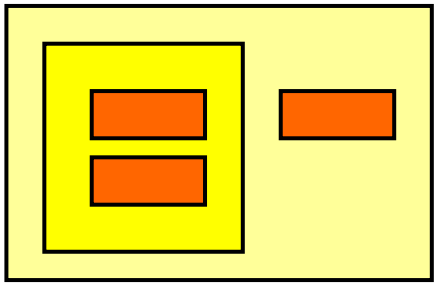
Beispiel: 6 2 1 3 4 5 8 7 9



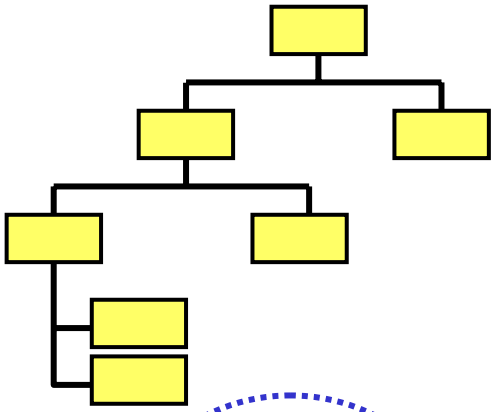


2.7 Alternative Darstellungen für Bäume

- Inclusion Drawings:
  - „Kinder sind enthalten in“



- „Tip-Over“ Zeichnungen:
  - „Kinder entweder horizontal oder vertikal“



- Radiale Baumzeichnungen