2.2 Rot-Schwarz-Bäume

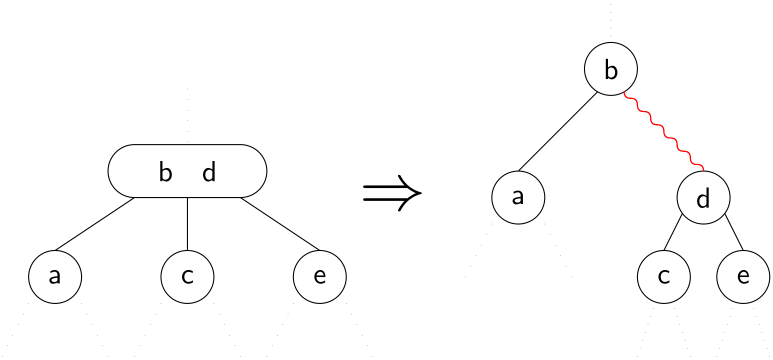
Definition 15

Rot-Schwarz-Bäume sind externe Binärbäume (jeder Knoten hat 0 oder 2 Kinder) mit roten und schwarzen Kanten, so dass gilt:

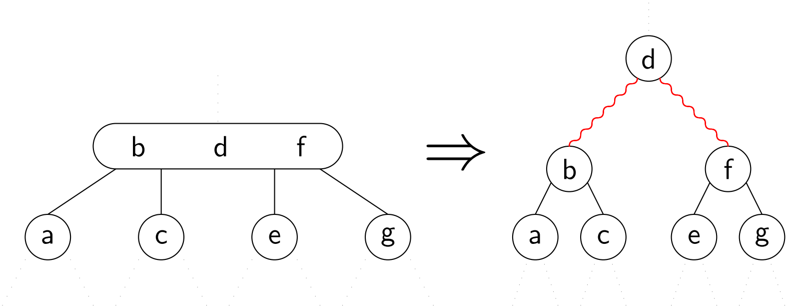
1. alle Blätter hängen an schwarzen Kanten (durchgezogene Linien)
2. alle Blätter haben die gleiche „Schwarztiefe“
3. kein Pfad von der Wurzel zu einem Blatt enthält (zwei oder mehr) aufeinanderfolgende rote Kanten (gewellte Linien).

Dabei ist die „Schwarztiefe“ eines Knoten die Anzahl der schwarzen Kanten auf dem Pfad von der Wurzel zu diesem Knoten.

Rot-Schwarz-Bäume können zur Implementierung von $(2,3)$- oder $(2,4)$-Bäumen dienen, mit dem Vorteil, dass alle internen Knoten Verzweigungsgrad $= 2$ haben!



Implementierung eines 4-Knotens:



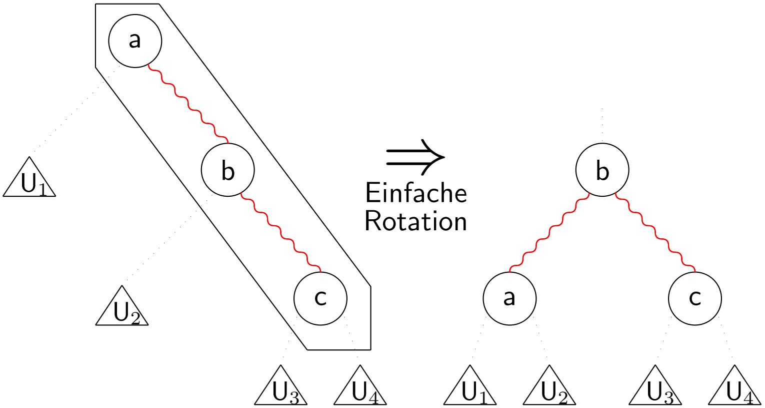
Operationen auf Rot-Schwarz implementierten (a, b)-Bäumen:

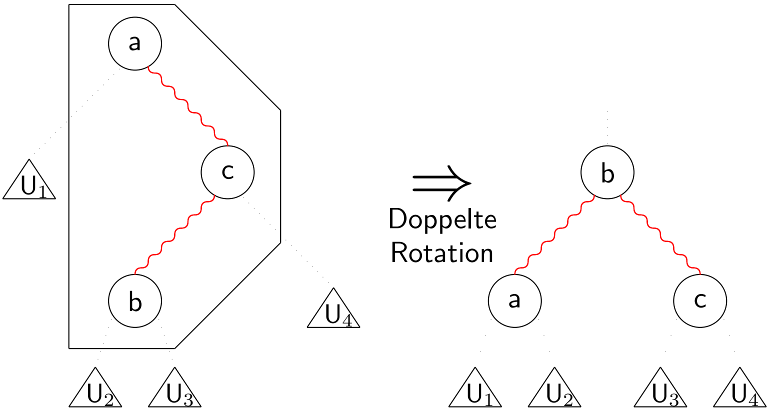
1. *IsElement*(k, T): vgl. (a, b)-Bäume

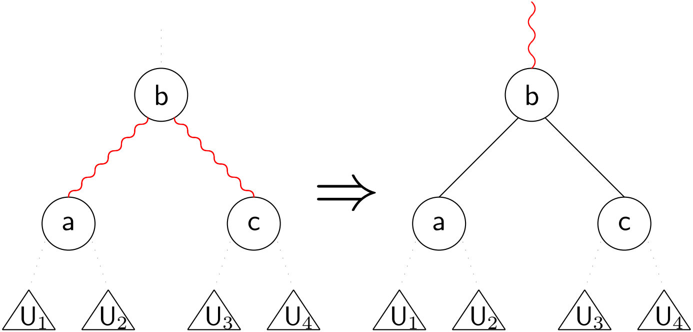
   $$\boxed{\text{Zeit } \mathcal{O}(\log n)}$$

2. *Insert*(k, T): Führe *IsElement*(k, T) aus $\rightsquigarrow$ Blatt w. Dort sei o.B.d.A. nicht das Element v gespeichert. Ersetze w durch neuen internen Knoten w' mit Kindern v, w, wobei v (bzw. w) ein neues Blatt mit Schlüssel k ist. w' erhält eine rote Eingangskante.

   Falls nun zwei aufeinanderfolgende rote Kanten an w' vorliegen, führe Rotationen zur Rebalancierung durch.


a
b
c
U1
U2
U3
U4
Einfache
Rotation
b
a
c
U1
U2
U3
U4

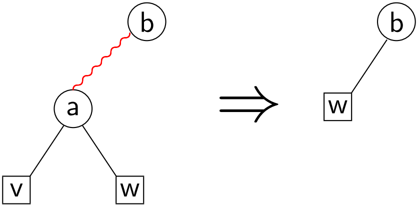

a
c
b
U1
U2
U3
U4
Doppelte
Rotation
b
a
c
U1
U2
U3
U4


b
a
c
U1
U2
U3
U4
⇒
b
a
c
U1
U2
U3
U4

3. *Delete*(k, T): Das Löschen des Blattes v macht es erforderlich, dass der Vater des Blattes durch den Bruder des Blattes ersetzt wird, damit der Binärbaumstruktur erhalten bleibt. Der Bruder von v ist eindeutig bestimmt.

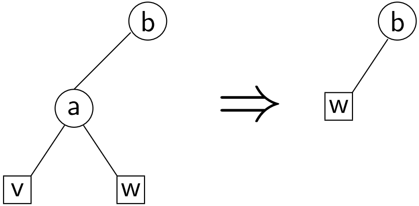
   Dabei treten zwei Fälle auf:

   1. Fall



   Dadurch ändert sich die Schwarztiefe von w nicht, denn die Eingangskante zu w wird (notwendig) zu schwarz umgefärbt. Auch kann es hierdurch nicht zu zwei aufeinanderfolgenden roten Kanten kommen.

2. Fall



Hierdurch verkleinert sich die Schwarztiefe von w um 1. Wir nennen einen Knoten, dessen Schwarztiefe um 1 zu gering ist, „zu hoch“. Falls die Eingangskante eines zu hohen Knoten rot ist, wird sie nach schwarz umgefärbt. Wir unterscheiden nun die folgenden Fälle:

1. Vater von w schwarz, Bruder von w rot: Der Baum wird so rotiert und umgefärbt, dass der Vater rot und der Bruder schwarz ist.
2. Bruder von w schwarz: Die Eingangskante des Bruders wird rot gefärbt und die Prozedur rekursiv mit dem Vater fortgesetzt, der nun (zusammen mit allen Knoten seines Unterbaums) zu hoch ist (Achtung: Falls Vater zuvor rot, Rebalancierung, da dann zwei aufeinander folgende rote Kanten).

Bemerkung: Falls der Bruder von w eine rote Ausgangskante hat, ist im zweiten Fall eine nicht-rekursive Vereinfachung möglich, indem die obersten Ebenen des Unterbaums, dessen Wurzel der Vater von w ist, geeignet umgeordnet werden.

Satz 16

Die Tiefe eines Rot-Schwarz-Baumes mit n Blättern ist $\mathcal{O}(\log n)$.

Beweis:

Hausaufgabe! $\square$

Korollar 17

Die Wörterbuch-Operationen auf Rot-Schwarz-Bäumen benötigen Zeit $\mathcal{O}(\log n)$.

Beweis:

Hausaufgabe! $\square$

Rot-Schwarz-Bäume wurden von R. Bayer unter der Bezeichnung symmetric binary B-trees erfunden und von Guibas und Sedgewick benannt und weiter erforscht.

- R. Bayer:
  Symmetric binary B-trees: Data structure and maintenance algorithms,
  Acta Inf. **1**, pp. 290–306, 1972

- Leo J. Guibas, Robert Sedgewick:
  A dichromatic framework for balanced trees,
  Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science, pp. 8–21. IEEE Computer Society, 1978

3. Binäre Suchbäume

3.1 Natürliche binäre Suchbäume

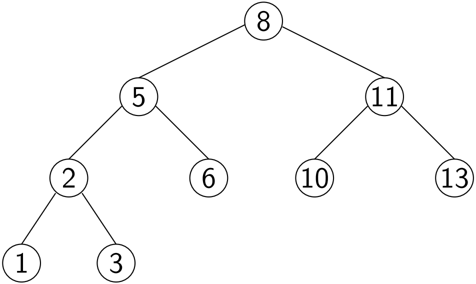
Definition 18

Ein natürlicher binärer Suchbaum über einem durch $\leq$ total geordneten Universum U ist ein als interner Suchbaum organisierter Binärbaum (also: Schlüssel an den internen Knoten, Blätter entsprechen leeren Knoten und werden nur wenn nötig angegeben).

Sortierungsbedingung:

Für jeden Knoten v und alle Knoten u im linken und alle Knoten w im rechten Unterbaum von v gilt

$$k(u) < k(v) < k(w)\,.$$



Lemma 19

Die In-Order-Linearisierung eines binären Suchbaumes mit obigen Suchwegbedingungen ergibt die Folge der Schlüssel in (strikt) aufsteigender Folge.

Beweis:

Klar! □

Die Wörterbuch-Operationen:

- *IsElement*(k, T):

$v :=$ Wurzel von T;
while $v \neq$ Blatt **do**
  if $k(v) = k$ **then**
    return $v(v)$
  elif $k(v) > k$ **then**
    $v :=$linkes Kind von v
  else
    $v :=$rechtes Kind von v
  fi
od
return NIL

- *Insert*(k, T):

  Führe *IsElement*(k, T) aus;

  if k nicht in T **then**

    IsElement ergibt Blatt w (NIL-Pointer);

    Füge neuen internen Knoten v mit $k(v) = k$ an Stelle von w ein

  fi

- *Delete*(k, T):

  Führe *IsElement*(k, T) aus;

  if k in T enthalten **then**

    IsElement führt zu Knoten w mit $k = k(w)$

    Falls w keine internen Kinder hat: klar

    Falls w nur ein internes Kind hat: Ersetze w durch dieses

    Ansonsten ersetze w durch seinen In-Order-Vorgänger oder -Nachfolger

  fi

Problem bei natürlichen Suchbäumen:
Bei bestimmten Abfolgen der Wörterbuchoperationen entarten natürliche Suchbäume stark, z.B. bei Einfügen der Schlüssel in monoton aufsteigender bzw. absteigender Folge zu einer linearen Liste der Tiefe n.

Daraus ergibt sich für die Wörterbuch-Operationen *Insert*, *Delete*, *IsElement* eine *worst case*-Komplexität von

$$\Theta(n)\,.$$

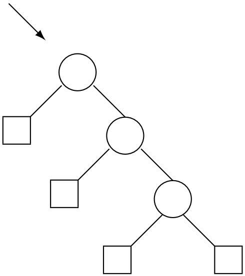
Falls alle Einfügefolgen gleichwahrscheinlich sind, gilt für die Höhe des Suchbaums

$$\mathbb{E}[h] = \mathcal{O}(\log n)\,.$$

Falls alle topologischen (von der Form her gleichaussehenden) Bäume der Größe n gleichwahrscheinlich sind, dann gilt

$$\mathbb{E}[h] = \Theta(\sqrt{n})\,.$$

Ohne Beweis!



3.2 Höhenbalancierte binäre Suchbäume (AVL-Bäume)

Definition 20

AVL-Bäume sind (interne) binäre Suchbäume, die die folgende Höhenbalancierung erfüllen:

Für jeden Knoten v gilt:

$$|\text{Höhe}(\text{linker UB}(v)) - \text{Höhe}(\text{rechter UB}(v))| \leq 1\,.$$

Bemerkung: AVL-Bäume sind nach ihren Erfindern G. Adelson-Velskii und Y. Landis (1962) benannt.

Satz 21

Ein AVL-Baum der Höhe h enthält mindestens $F_{h+2} - 1$ und höchstens $2^h - 1$ interne Knoten, wobei F_n die n-te Fibonacci-Zahl ($F_0 = 0, F_1 = 1$) und die Höhe die maximale Anzahl von Kanten auf einem Pfad von der Wurzel zu einem (leeren) Blatt ist.