

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
oooooooooooo

Tango Bäume

Andreas Windorfer

1. November 2020

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
oooooooooooo

Übersicht

Binärer Suchbaum

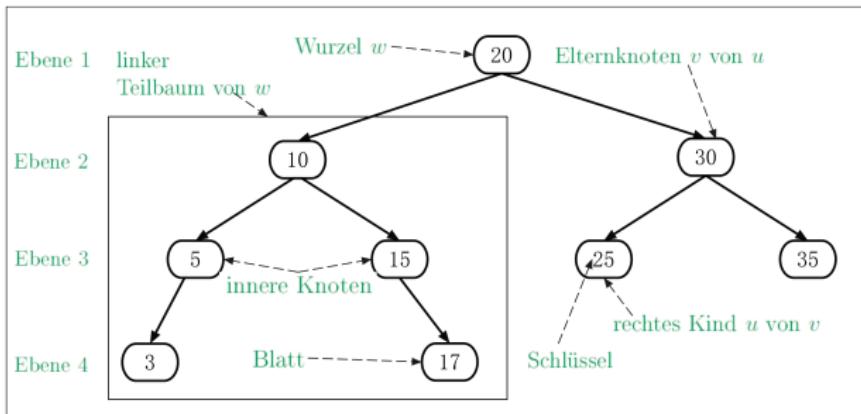
Dynamische Optimalität

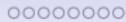
Tango Baum

Splay Baum

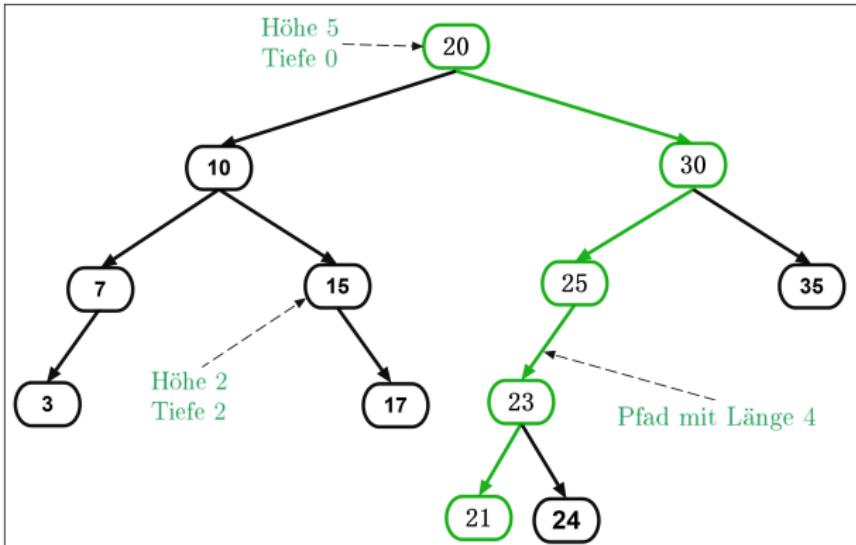
Laufzeittests

Binärer Suchbaum

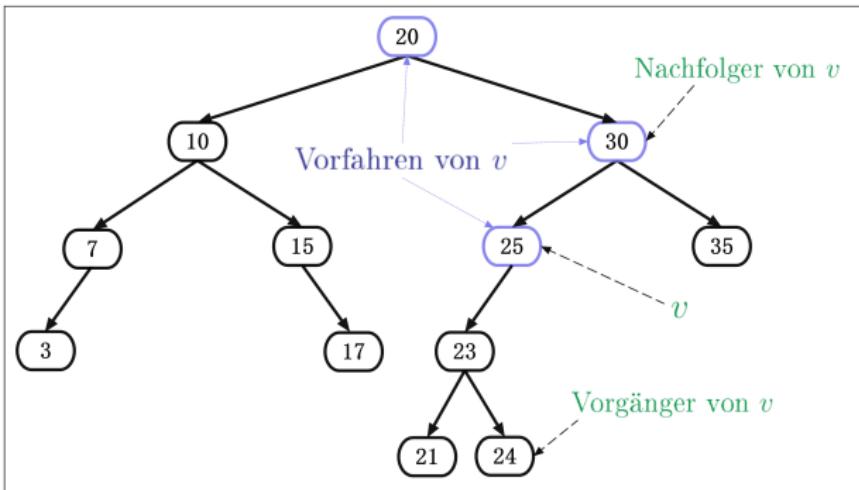




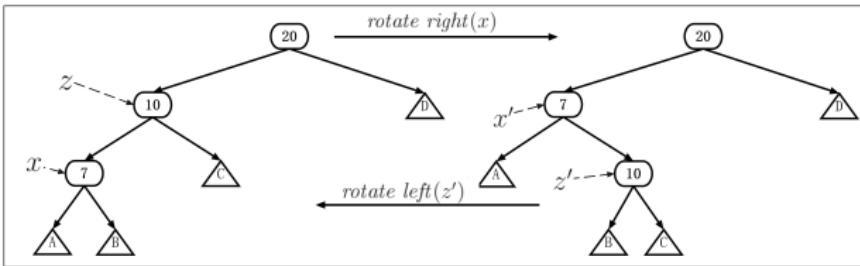
Binärer Suchbaum



Binärer Suchbaum



Binärer Suchbaum



Binärer Suchbaum
oooo

Dynamische Optimalität
●oooo

Tango Baum
oooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
oooooooooooo

Übersicht

Binärer Suchbaum

Dynamische Optimalität

Tango Baum

Splay Baum

Laufzeittests

access(k) Operation

Parameter / Rückgabe

- Parameter k : Schlüssel im BST (Schlüsselmenge K)
- Rückgabe: Knoten mit Schlüssel k

access(k) Operation

Einschränkungen

Ein Zeiger p (berührter Knoten) in die Struktur:

- Setze p auf ein Kind von p
- Setze p auf den Elternknoten von p
- Rotationen

access(k) Operation

Einschränkungen

Ein Zeiger p (berührter Knoten) in die Struktur:

- Setze p auf ein Kind von p
- Setze p auf den Elternknoten von p
- Rotationen

Berechnung der Kosten

Einheitskosten von „1“.

Zugriffsfolgen

Zugriffsfolgen

- $X = x_1, x_2, \dots, x_m$, mit $\forall i \in \{1, 2, \dots, m\} : x_i \in K$
- $access(x_1), access(x_2), \dots, access(x_m)$

Zugriffsfolgen

Zugriffsfolgen

- $X = x_1, x_2, \dots, x_m$, mit $\forall i \in \{1, 2, \dots, m\} : x_i \in K$
- $access(x_1), access(x_2), \dots, access(x_m)$

dynamische BST

- Anpassung der Struktur

Kostenrechnung

Anzahl der Einzelschritte + m

dynamisch Optimal

$OPT(X)$

Niedrigste Kosten zum Ausführen von X

dynamisch Optimal

$OPT(X)$

Niedrigste Kosten zum Ausführen von X

dynamisch Optimal

BST mit Kosten von $O(OPT(X))$, für beliebige X

c-competitive

BST mit Kosten von $O(c \cdot OPT(X))$, für beliebige X

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
●oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
oooooooooooo

Binärer Suchbaum

Dynamische Optimalität

Tango Baum

Splay Baum

Laufzeittests

Tango Baum

Eigenschaften

- Aus BSTs bestehender BST
- $\log(\log(n))$ -competitive

Literatur

Erik D. Demaine, Dion. Harmon, John. Iacono, and Mihai. Patrascu. Dynamic optimality-almost. SIAM Journal on Computing, 37(1):240 251, 2007.

Interleave Lower Bound

Motivation

- Berechnung einer unteren Schranke zu $OPT(X)$
- Beweis der $\log(\log(n))$ -competitiveness

Lower Bound Tree

Definition

Zu $X = x_1, x_2, \dots, x_m$ und $K = \{k \in \mathbb{N} \mid k \text{ ist in } X \text{ enthalten}\}$

Lower Bound Tree

Definition

Zu $X = x_1, x_2, \dots, x_m$ und $K = \{k \in \mathbb{N} \mid k \text{ ist in } X \text{ enthalten}\}$
ist der komplette BST P mit der Schlüsselmenge K der LBT.

Beispiel LBT

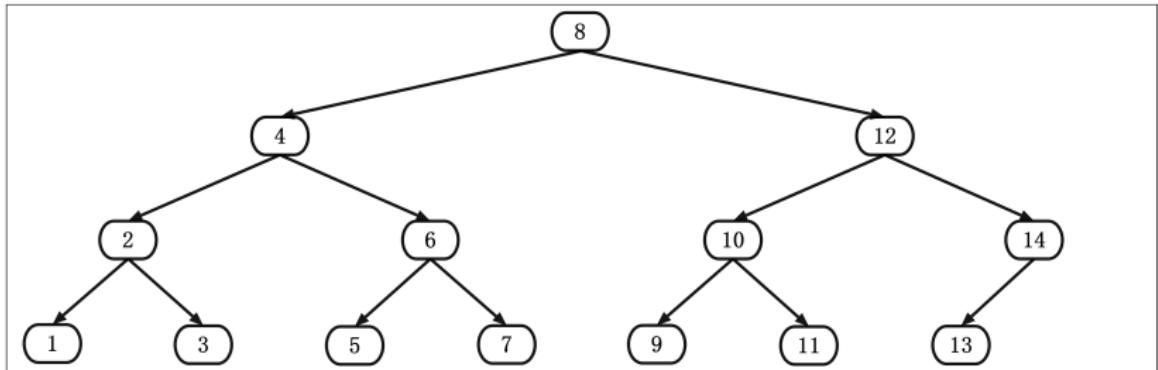


Abbildung: Der Lower Bound Tree zur Zugriffsfolge 1, 2, .., 14.

Lower Bound Tree

Linke Region eines Kontens v

Schlüssel des linken Teilbaumes von v und $\text{key}(v)$

Linke Region eines Kontens v

Schlüssel des rechten Teilbaumes von v

Interleave durch v

x_{i-1} liegt in der linken Region und x_i in der rechten,
oder umgekehrt.

Lower Bound Tree

inScore (X, v)

Anzahl der Interleaves durch v

IB (X)

$$\text{IB} (X) = \sum_{u \in U} \text{inScore} (X, u)$$

Transition Points

T_0 Startzustand, T_i nach ausführen von access x_i .

Transition Points

T_0 Startzustand, T_i nach ausführen von access x_i . $j \in 0, 1, \dots m$

Zu jedem Knoten u aus P , mit nicht leerer rechter Region, existiert ein transition point in T_j

Transition Points

Sei v der Transition Point zu u und T_j

1. Im Pfad von der Wurzel zu v ist ein Knoten mit einem Schlüssel aus der linken Region von u enthalten.
2. Im Pfad von der Wurzel zu v ist ein Knoten mit einem Schlüssel aus der rechten Region von u enthalten.
3. Kein anderer Knoten mit kleinerer Tiefe erfüllt die Eigenschaften eins und zwei.

Transition Points

Sei U die Menge der Knoten in P mit einer nicht leeren rechten Region.

- Lemma 1: Es gibt zu jedem Knoten $u \in U$ genau einen transition point in T_j .
- Lemma 2: Wird ein transition point nicht berührt, so ist er noch immer der transition point des selben Knotens.
- Lemma 3: Ein Knoten kann nicht der transition point mehrerer Knoten sein.

Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region

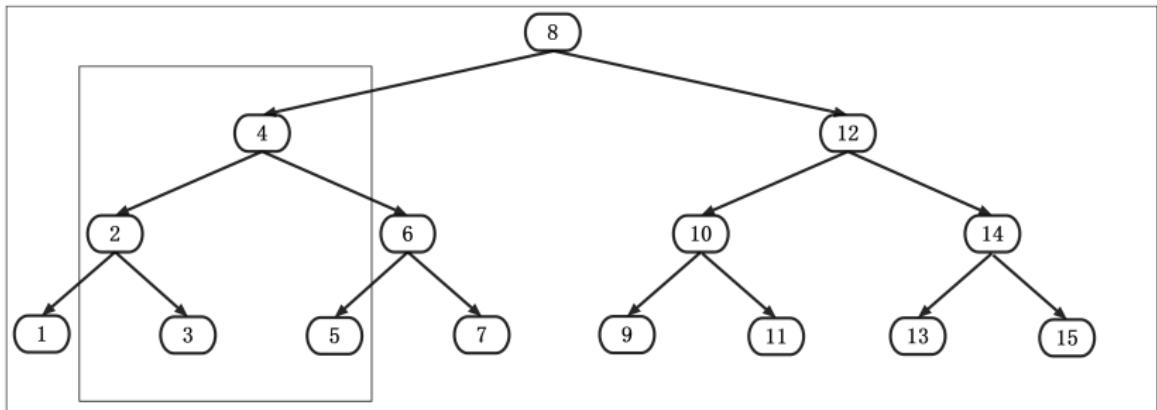
Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region
3. der Teilbaum mit der Wurzel u enthält genau die Schlüssel aus $K_l^r = \{k \in K | k \in [l, r]\}$

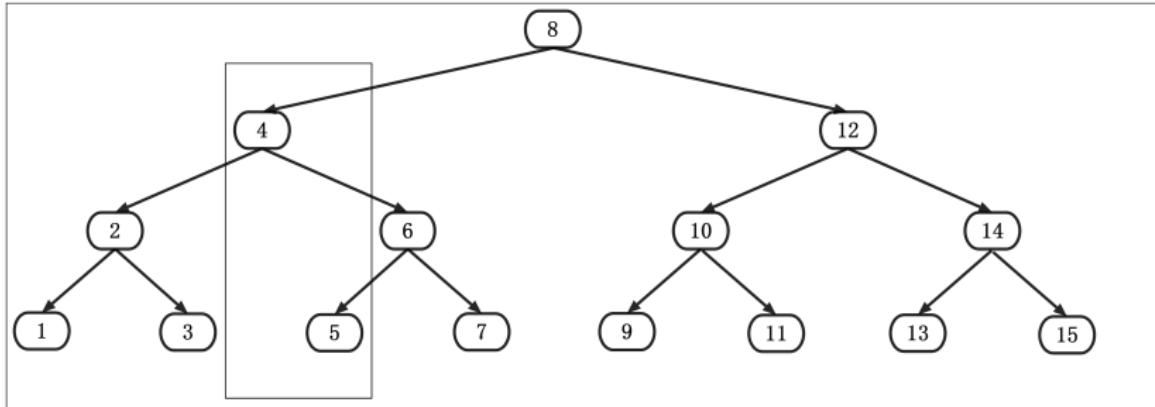
Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region
3. der Teilbaum mit der Wurzel u enthält genau die Schlüssel aus $K_l^r = \{k \in K | k \in [l, r]\}$
4. v_l ist der Vorfahre der Schlüssel der linken Region
5. v_r ist der Vorfahre der Schlüssel der rechten Region
6. w ist der gemeinsame Vorfahre dieser Schlüssel

Beweis Lemma 1



Beweis Lemma 1



Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region

Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region
3. der Teilbaum mit der Wurzel u enthält genau die Schlüssel aus $K_l^r = \{k \in K | k \in [l, r]\}$

Beweis Lemma 1

1. l ist der kleinste Schlüssel der linken Region
2. r ist der größte Schlüssel der rechten Region
3. der Teilbaum mit der Wurzel u enthält genau die Schlüssel aus $K_l^r = \{k \in K | k \in [l, r]\}$
4. v_l ist der Vorfahre der Schlüssel der linken Region
5. v_r ist der Vorfahre der Schlüssel der rechten Region
6. w ist der gemeinsame Vorfahre dieser Schlüssel
7. $w = v_l$ bzw. $w = v_r$
8. Transition point ist v_r bzw. v_l

Transition Point Zuordnung

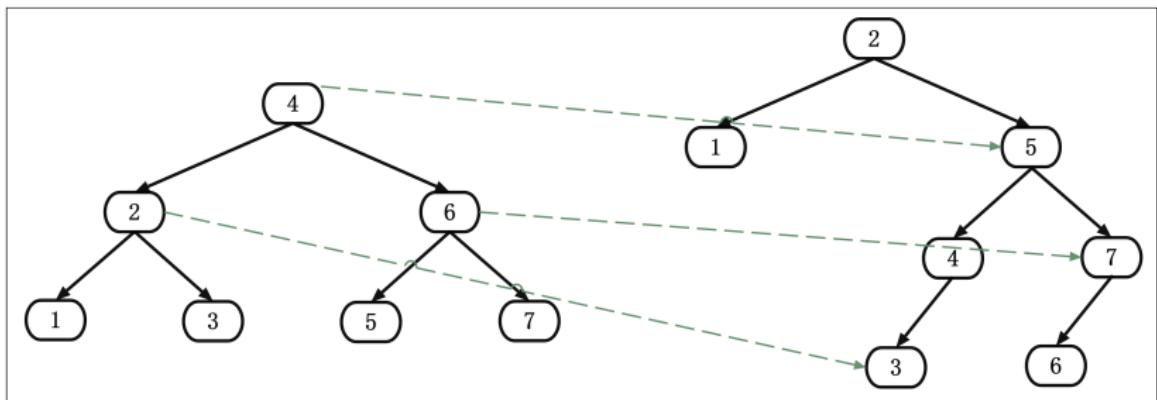


Abbildung: Links ein Lower Bound Tree, rechts ein BST

Satz Interleave Lower Bound

Sei $X = x_0, x_1, \dots, x_m$ eine Zugriffsfolge und n die Anzahl der Knoten im zu X erstellten Lower Bound Tree Y. Dann gilt
 $OPT(X) \geq IB(X)/2 - n$.

Beweis Interleave Lower Bound

1. Zählen der Berührungen von Transition Points
2. Die Anzahl der Berührungen kann für jeden Knoten einzeln bestimmt werden. (Lemma 5.1 und 5.3)

Beweis Interleave Lower Bound

1. Zählen der Berührungen von Transition Points
2. Die Anzahl der Berührungen kann für jeden Knoten einzeln bestimmt werden. (Lemma 5.1 und 5.3)
3. Betrachteter Knoten u . $X_I^{r'} = x_{i_0}, x_{i_1}, \dots, x_{i_p}$ bilden
4. $\text{inScore}(X, u) = p$

Beweis Interleave Lower Bound

1. Zählen der Berührungen von Transition Points
2. Die Anzahl der Berührungen kann für jeden Knoten einzeln bestimmt werden. (Lemma 5.1 und 5.3)
3. Betrachteter Knoten u . $X_I^{r'} = x_{i_0}, x_{i_1}, \dots, x_{i_p}$ bilden
4. $\text{inScore}(X, u) = p$
5. Sei $q \in \mathbb{N}$ mit $1 \leq q \leq \lfloor p/2 \rfloor$

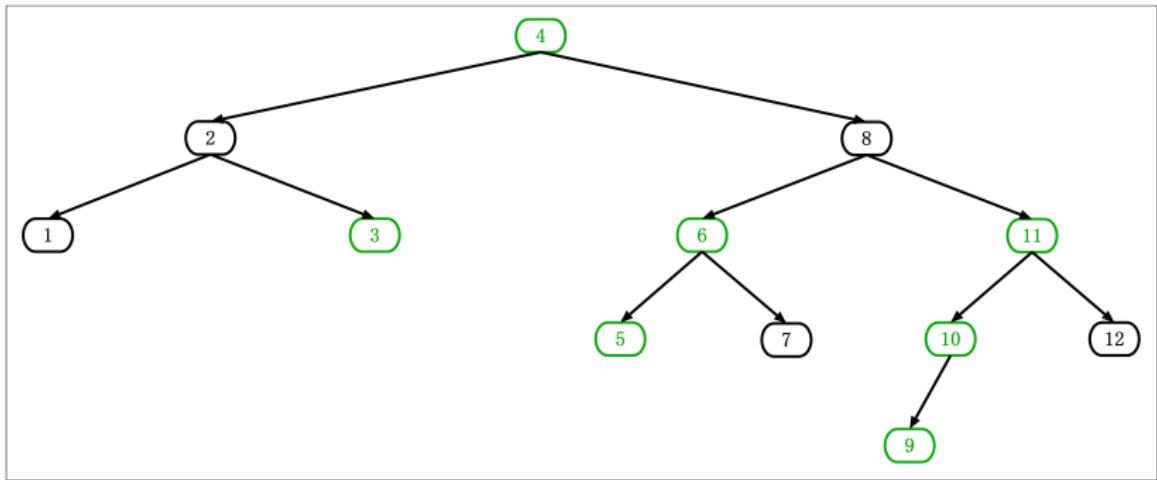
Beweis Interleave Lower Bound

1. Zählen der Berührungen von Transition Points
2. Die Anzahl der Berührungen kann für jeden Knoten einzeln bestimmt werden. (Lemma 5.1 und 5.3)
3. Betrachteter Knoten u . $X_I^{r'} = x_{i_0}, x_{i_1}, \dots, x_{i_p}$ bilden
4. $\text{inScore}(X, u) = p$
5. Sei $q \in \mathbb{N}$ mit $1 \leq q \leq \lfloor p/2 \rfloor$
6. Es folgen mindestens $\lfloor p/2 \rfloor \geq p/2 - 1$ Berührungen des transition points von u

Beweis Interleave Lower Bound

1. Zählen der Berührungen von Transition Points
2. Die Anzahl der Berührungen kann für jeden Knoten einzeln bestimmt werden. (Lemma 5.1 und 5.3)
3. Betrachteter Knoten u . $X_I^{r'} = x_{i_0}, x_{i_1}, \dots, x_{i_p}$ bilden
4. $\text{inScore}(X, u) = p$
5. Sei $q \in \mathbb{N}$ mit $1 \leq q \leq \lfloor p/2 \rfloor$
6. Es folgen mindestens $\lfloor p/2 \rfloor \geq p/2 - 1$ Berührungen des transition points von u
7. $IB(X)/2 - |U| \geq IB(X)/2 - n$

Aufbau des Tango Baumes



Erweiterte Knoten:

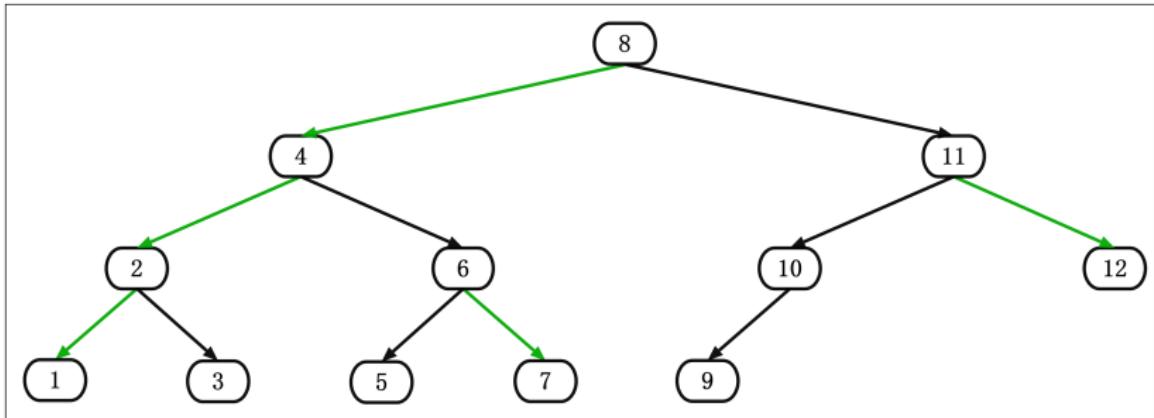
1. *depth*

2. *minDepth*

3. *maxDepth*

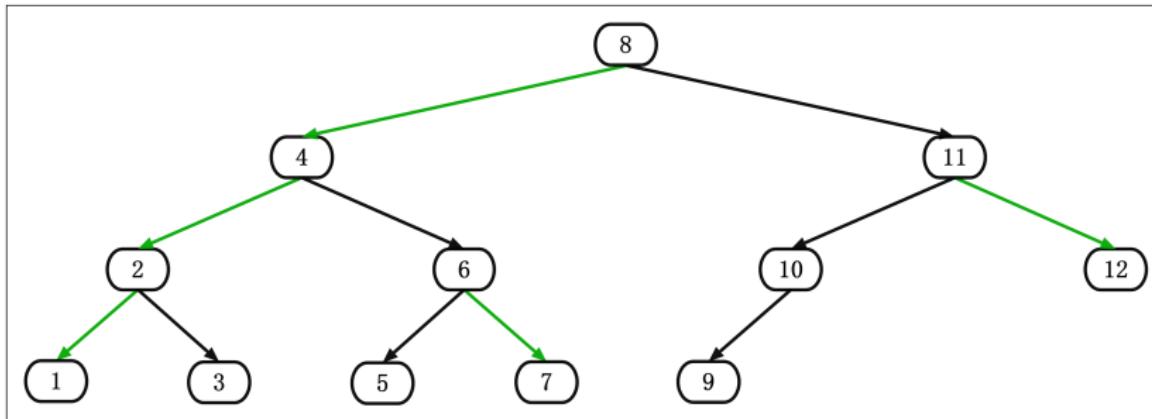
Aufbau des Tango Baumes

- preferred childs



Aufbau des Tango Baumes

- preferred childs



- preferred paths

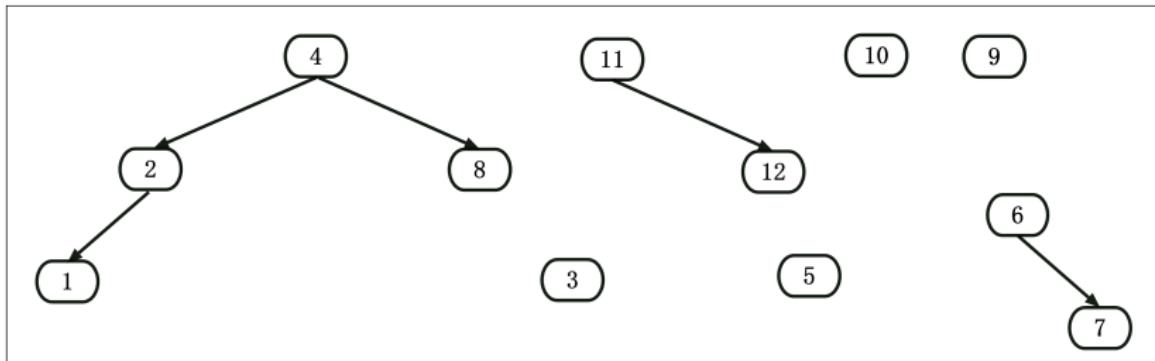
$$P_1 = (8, 4, 2, 1)$$

$$P_2 = (3)$$

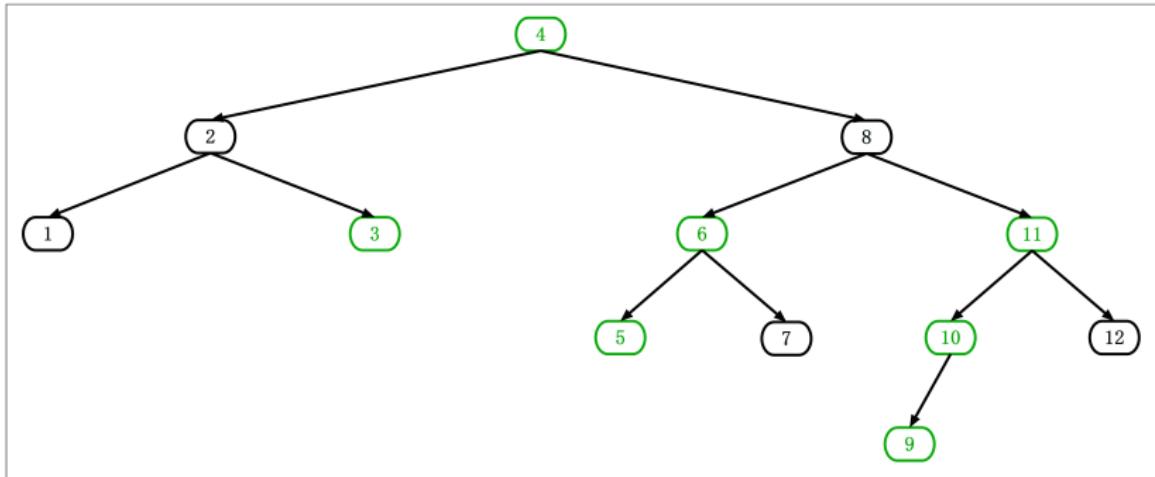
$$P_3 = (6, 7)$$

Aufbau des Tango Baumes

- Hilfsbäume

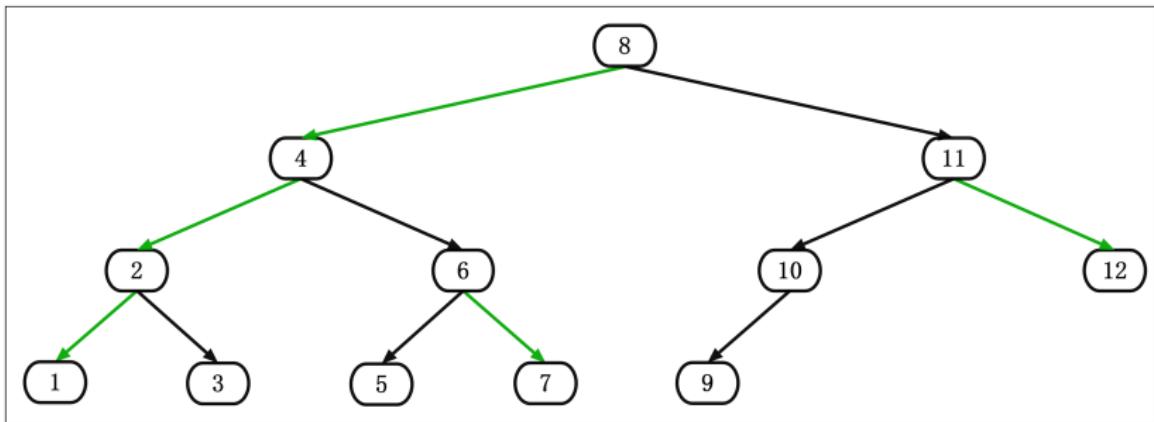


Aufbau des Tango Baumes



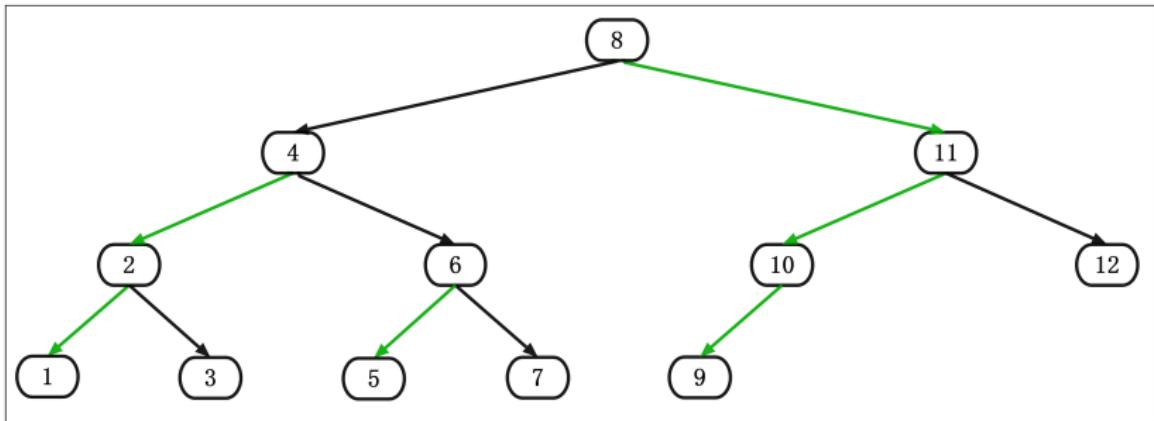
Aufbau des Tango Baumes

- $access(9)$



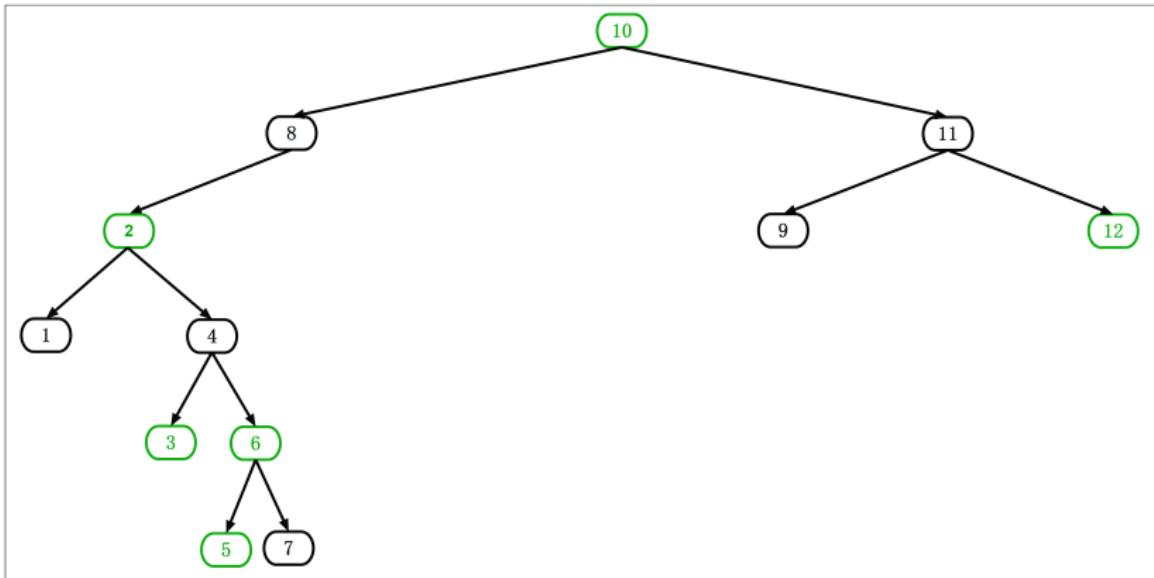
Aufbau des Tango Baumes

- $access(9)$



Aufbau des Tango Baumes

- *access(9)*



access Operation

- Suche findet im HB mit der Tango Baum Wurzel satt.

Anforderungen an die DS der Hilfsbäume

1. $h = O(\log(n))$

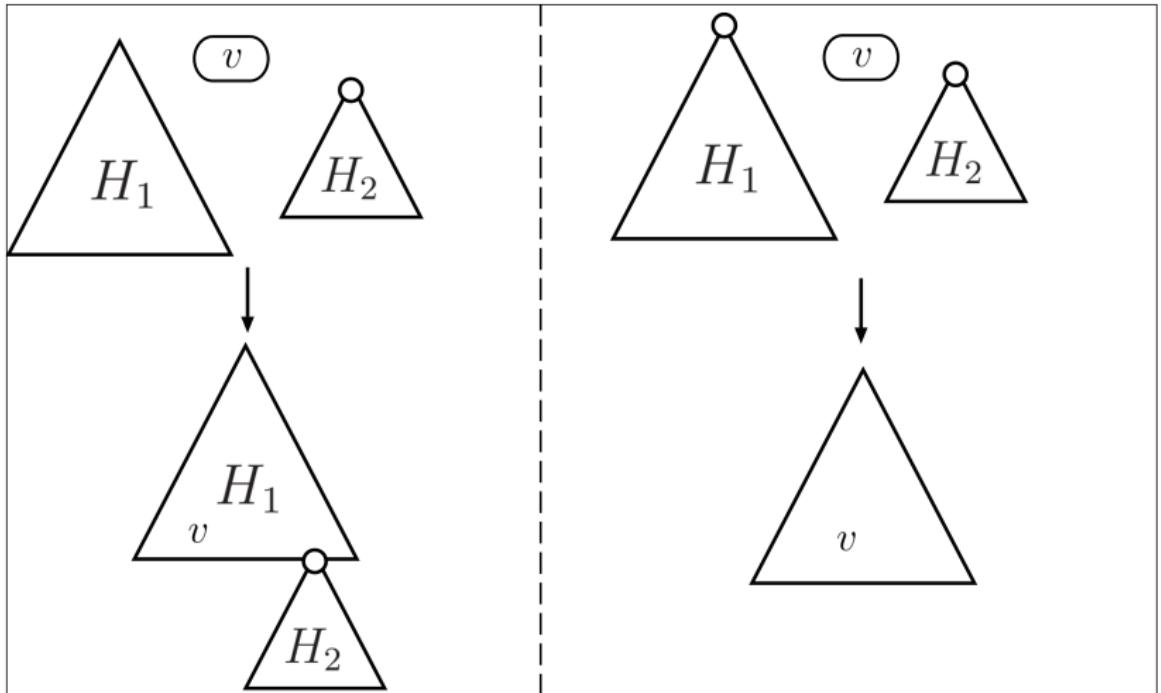
2. $\text{concatenate} (\text{HB } H_1, \text{Knoten } v, \text{HB } H_2)$

Zusammenführen von T_1 und T_2 . Laufzeit $O(\log(n))$

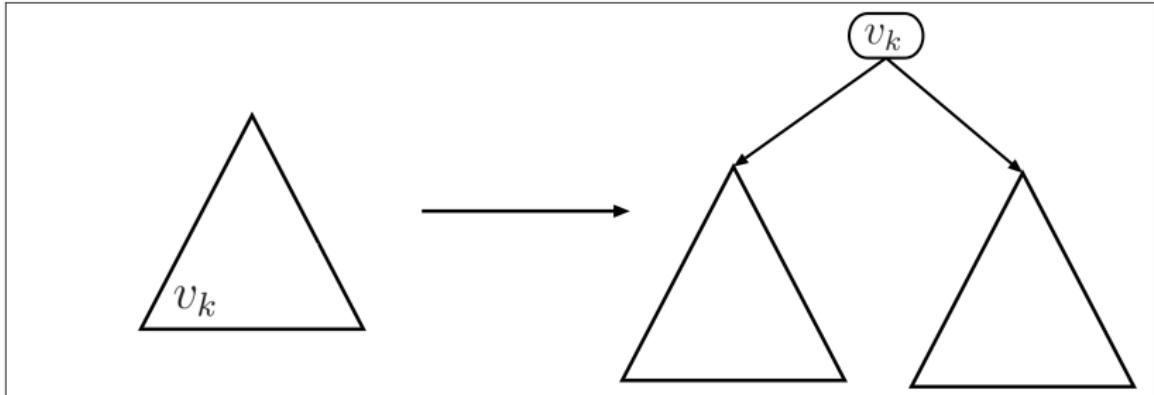
3. $\text{split} (\text{Schlüssel } k)$

Zerteilen der eigenen Struktur. Laufzeit $O(\log(n))$

concatenate (HB H_1 , Knoten v , HB H_2)



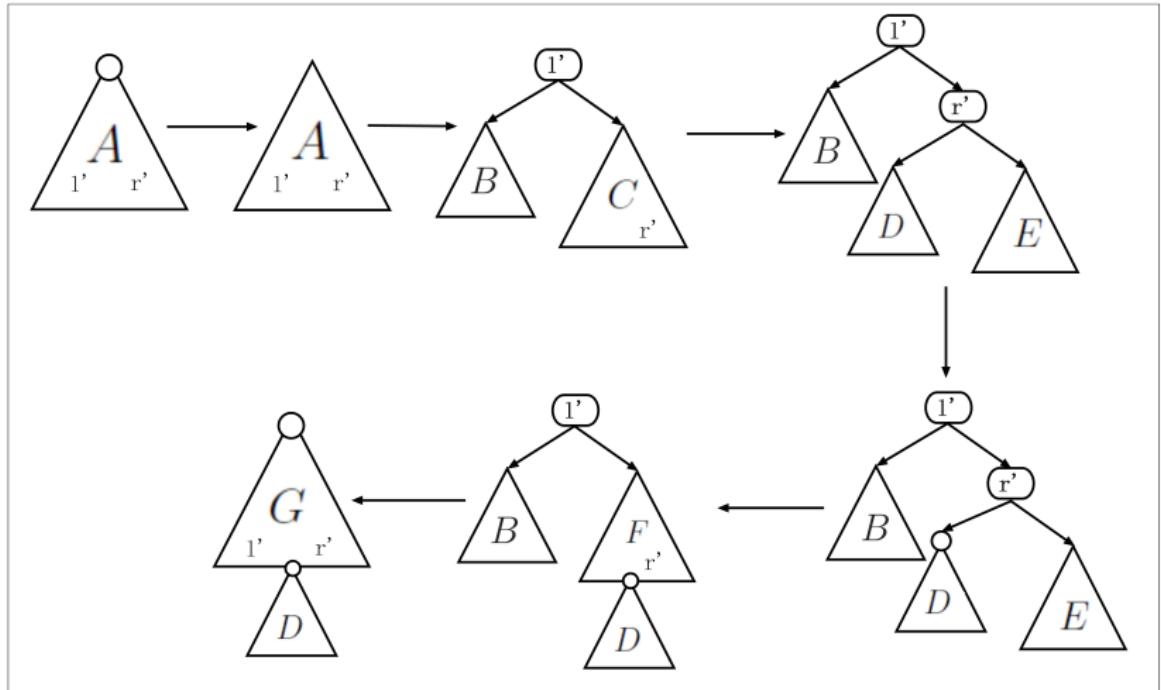
split (Schlüssel k)



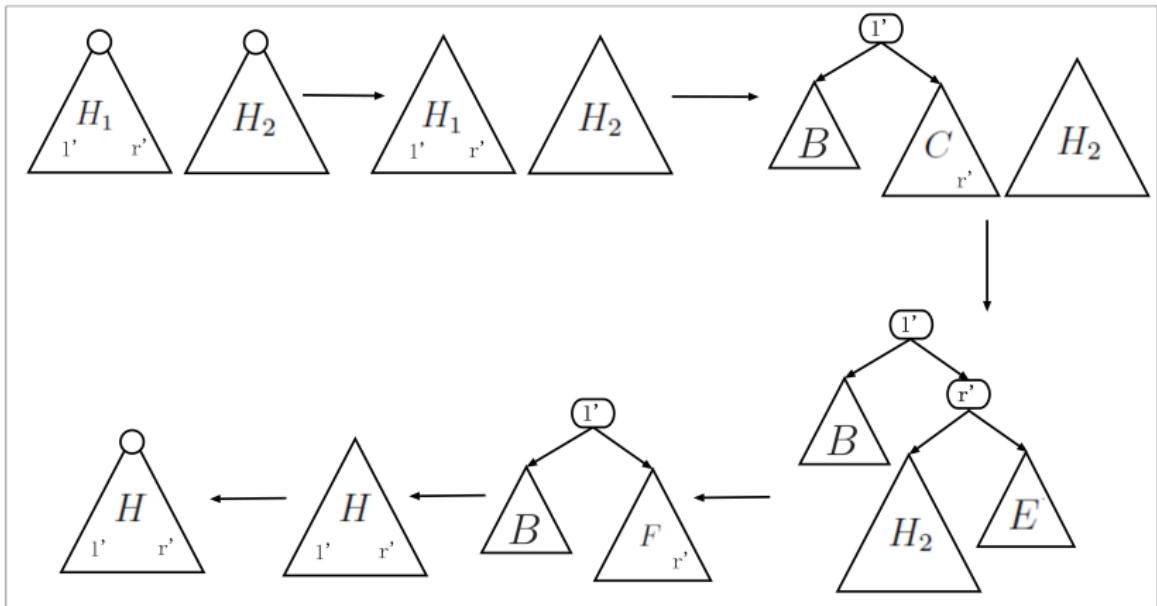
Hilfsoperationen für access

1. *join (HB T_1 , HB T_2)*
Vereinigen von T_1 und T_2
2. *cut (depth d)*
Zerteilen der eigenen Struktur beim Knoten mit Tiefe d

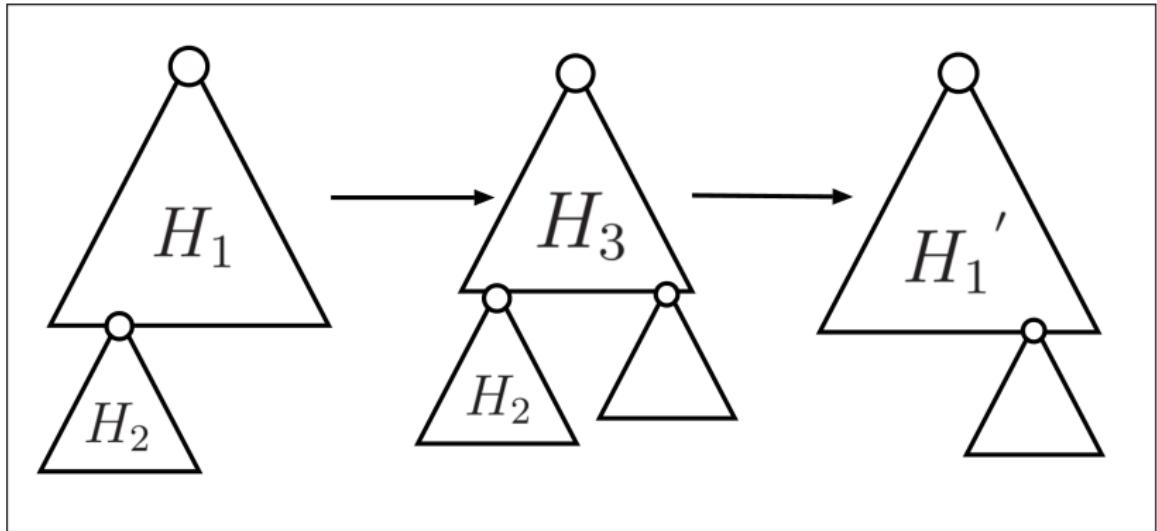
cut Operation



join Operation



access Operation



Laufzeit von access

1. $IB(X)$ Interleaves führen zu $IB(X)$ Wechsel bei preferred children von links nach rechts, oder umgekehrt.
2. maximal n zusätzliche Wechsel bei preferred children, durch Erstzugriffe in Teilbäume

Laufzeit von access

1. $IB(X)$ Interleaves führen zu $IB(X)$ Wechsel bei preferred children von links nach rechts, oder umgekehrt.
2. maximal n zusätzliche Wechsel bei preferred children, durch Erstzugriffe in Teilbäume
3. Das ergibt maximal $IB(X) + n$ Wechsel bei preferred children

Laufzeit von access

1. $IB(X)$ Interleaves führen zu $IB(X)$ Wechsel bei preferred children von links nach rechts, oder umgekehrt.
2. maximal n zusätzliche Wechsel bei preferred children, durch Erstzugriffe in Teilbäume
3. Das ergibt maximal $IB(X) + n$ Wechsel bei preferred children
4. Es wird maximal $IB(X) + n + 1$ mal an der Wurzel des Tango Baumes gestartet.
5. Für die Höhe eines Hilfsbaumes gilt $O(\log(\log(n)))$

Laufzeit von access

1. $IB(X)$ Interleaves führen zu $IB(X)$ Wechsel bei preferred children von links nach rechts, oder umgekehrt.
2. maximal n zusätzliche Wechsel bei preferred children, durch Erstzugriffe in Teilbäume
3. Das ergibt maximal $IB(X) + n$ Wechsel bei preferred children
4. Es wird maximal $IB(X) + n + 1$ mal an der Wurzel des Tango Baumes gestartet.
5. Für die Höhe eines Hilfsbaumes gilt $O(\log(\log(n)))$
6. $O((IB(X) + n + m)(1 + \log(\log(n))))$

Laufzeit von access

1. $IB(X)$ Interleaves führen zu $IB(X)$ Wechsel bei preferred children von links nach rechts, oder umgekehrt.
2. maximal n zusätzliche Wechsel bei preferred children, durch Erstzugriffe in Teilbäume
3. Das ergibt maximal $IB(X) + n$ Wechsel bei preferred children
4. Es wird maximal $IB(X) + n + 1$ mal an der Wurzel des Tango Baumes gestartet.
5. Für die Höhe eines Hilfsbaumes gilt $O(\log(\log(n)))$
6. $O((IB(X) + n + m)(1 + \log(\log(n))))$
7. $O((OPT(X) + n)(1 + \log(\log(n))))$, mit $OPT(X) \geq IB(X)/2 - n$

Laufzeit von access

Einzelne Access Operation:

1. $\Theta(\log(n))$ Wechsel bei preferred children

Laufzeit von access

Einzelne Access Operation:

1. $\Theta(\log(n))$ Wechsel bei preferred children
2. $O(\log(n)(\log(\log(n)) + 1))$

Laufzeit von access

Einzelne Access Operation:

1. $\Theta(\log(n))$ Wechsel bei preferred children
2. $O(\log(n)(\log(\log(n))+1))$
3. Balancierte BSTs erreichen $O(\log(n))$

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
●oooooooooooooooooooo

Laufzeittests
oooooooooooo

Binärer Suchbaum

Dynamische Optimalität

Tango Baum

Splay Baum

Laufzeittests

Splay Baum

1. dynamisch
2. keine zusätzliche Attribute
3. kein balancierter BST
4. $\log(n)$ -competitive bewiesen
5. dynamische Optimalität vermutet

access

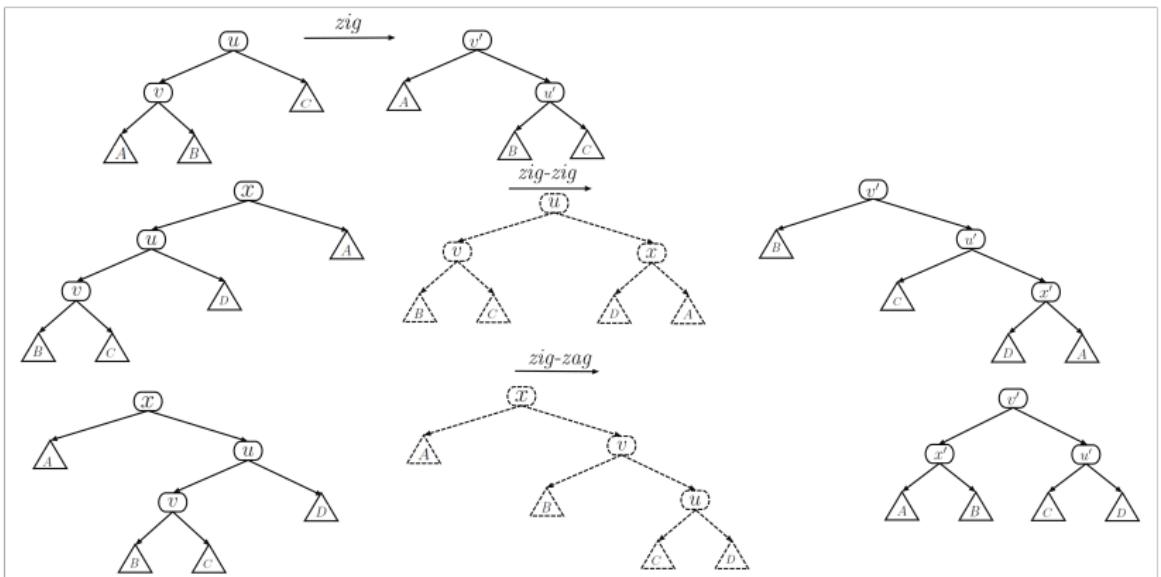
1. *splay* aufrufen
2. Knoten an der Wurzel zurückgeben

access

1. *splay* aufrufen
2. Knoten an der Wurzel zurückgeben

splay

- *zig*
- *zag*
- *zig-zig*
- *zag-zag*
- *zig-zag*
- *zag-zig*

splay

Eine einzige splay Operation

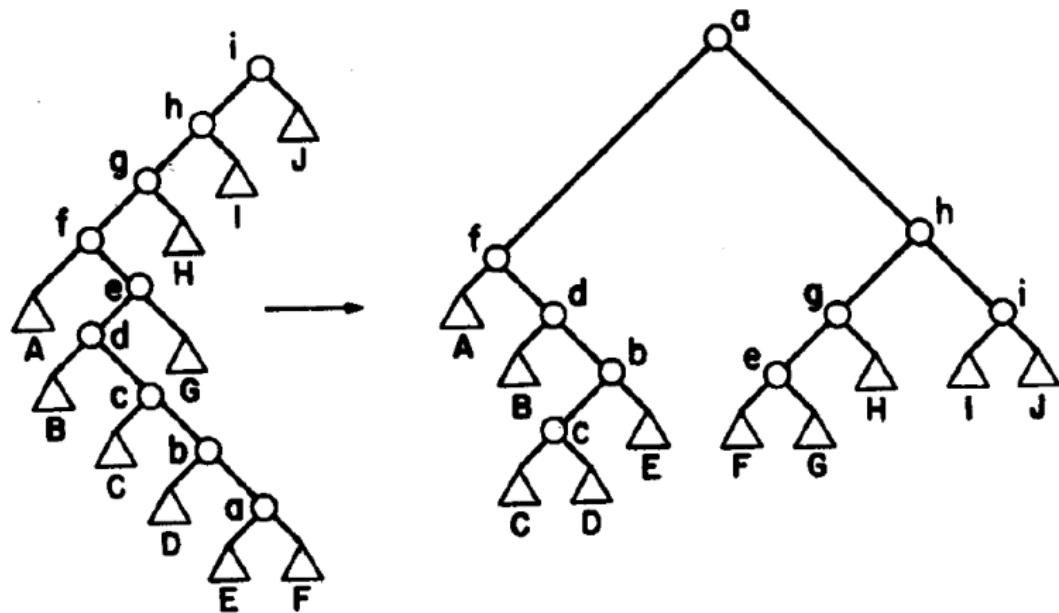


FIG. 4. Splaying at node *a*.

Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. J. ACM, 32(3):652-686, July 1985.

Access Lemma

Sei T ein Splay Baum mit n Knoten, Wurzel w und einem Knoten v mit dem Schlüssel k . Die amortisierten Kosten von $splay(k)$ sind maximal

$$3(r(w) - r(v)) + 1 = O(\log(tw(w)/tw(v))) = O(\log(n)).$$

Beweis Access Lemma

Potentialfunktionsmethode

- Gewichtsfunktion $w(v) \geq 0$

Beweis Access Lemma

Potentialfunktionsmethode

- Gewichtsfunktion $w(v) \geq 0$
- U ist Menge der Knoten im Teilbaum mit Wurzel v
- $tw(v) = \sum_{u \in U} w(u)$

Beweis Access Lemma

Potentialfunktionsmethode

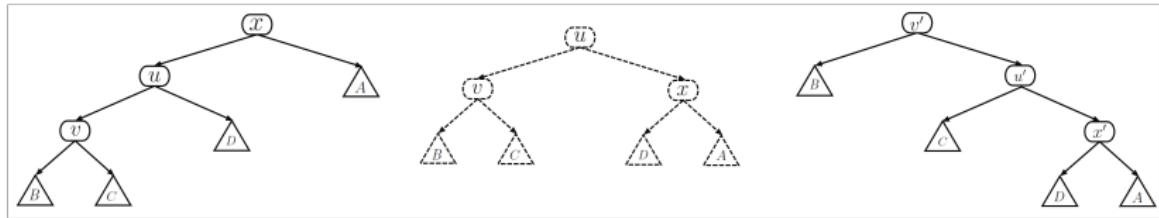
- Gewichtsfunktion $w(v) \geq 0$
- U ist Menge der Knoten im Teilbaum mit Wurzel v
- $tw(v) = \sum_{u \in U} w(u)$
- Rang $r(v) = \log_2(tw(v))$

Beweis Access Lemma

Potentialfunktionsmethode

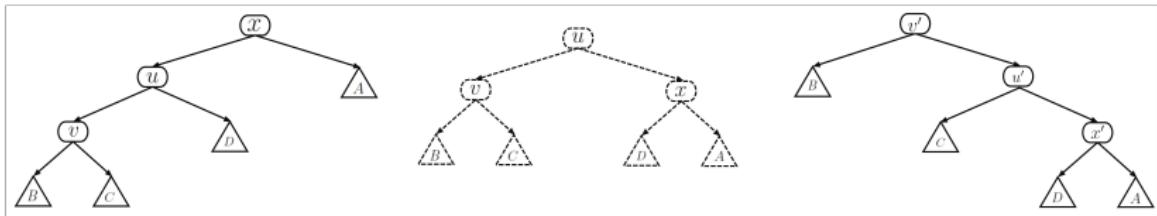
- Gewichtsfunktion $w(v) \geq 0$
- U ist Menge der Knoten im Teilbaum mit Wurzel v
- $tw(v) = \sum_{u \in U} w(u)$
- Rang $r(v) = \log_2(tw(v))$
- V ist Menge der Knoten im Gesamtbaum
- $\Phi = \sum_{v \in V} r(v)$

Beweis Access Lemma



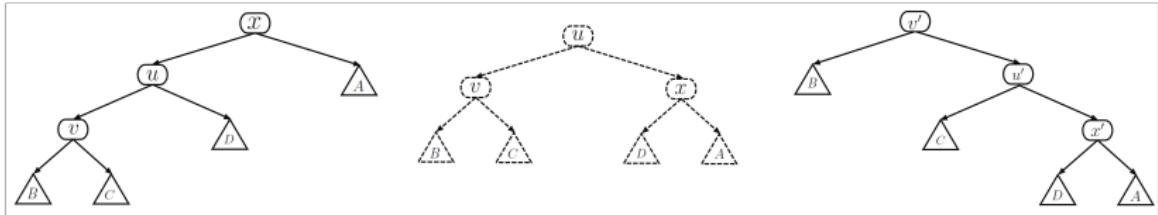
Kosten Einzelschritt
 $\leq 3 \left(r(v)' - r(v) \right) + 1$

Beweis Access Lemma



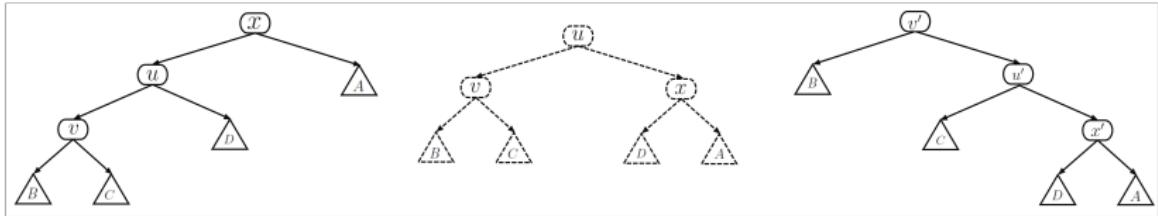
$$\begin{aligned} & 2 + r(u)' + r(v)' + r(x)' - r(u) - r(v) - r(x) \\ &= 2 + r(u)' + r(x)' - r(u) - r(v) \\ &\leq 2 + r(v)' + r(x)' - 2r(v) \end{aligned}$$

Beweis Access Lemma



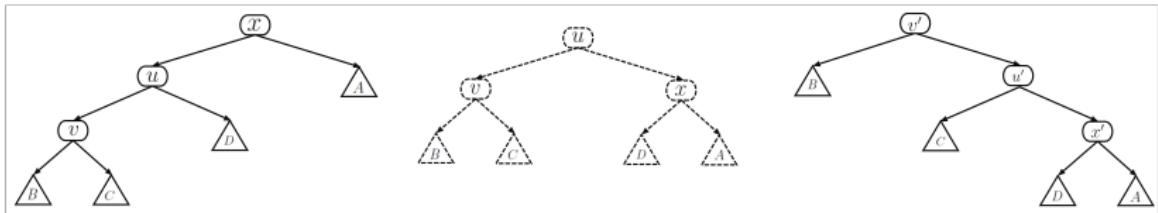
$$2 + r(v)' + r(x)' - 2r(v) \leq 3(r(v)' - r(v))$$

Beweis Access Lemma



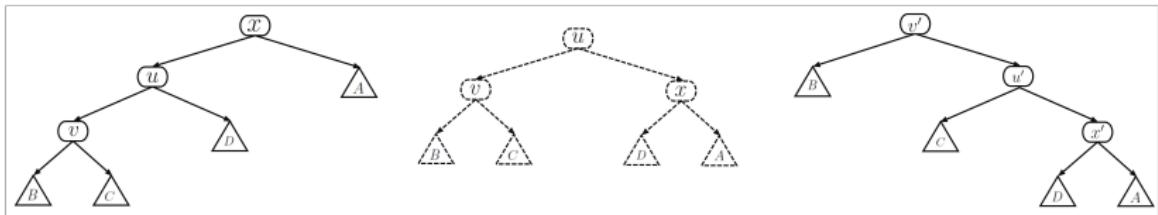
$$\begin{aligned} & 2 + r(v)' + r(x)' - 2r(v) \leq 3(r(v)' - r(v)) \\ \Leftrightarrow & 2 \leq 2r(v)' - r(x)' - r(v) \end{aligned}$$

Beweis Access Lemma



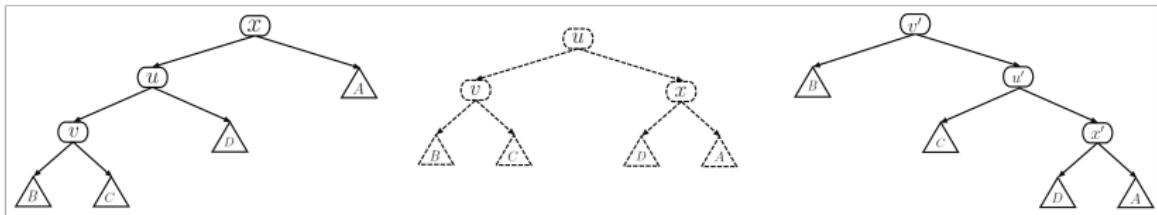
$$\begin{aligned} & 2 + r(v)' + r(x)' - 2r(v) \leq 3(r(v)' - r(v)) \\ \Leftrightarrow & 2 \leq 2r(v)' - r(x)' - r(v) \\ \Leftrightarrow & -2 \geq -2r(v)' + r(x)' + r(v) \end{aligned}$$

Beweis Access Lemma



$$\begin{aligned} & 2 + r(v)' + r(x)' - 2r(v) \leq 3(r(v)' - r(v)) \\ \Leftrightarrow & 2 \leq 2r(v)' - r(x)' - r(v) \\ \Leftrightarrow & -2 \geq -2r(v)' + r(x)' + r(v) \\ \Leftrightarrow & -2 \geq \log_2(tw(x') / tw(v')) + \log_2(tw(v) / tw(v')) \end{aligned}$$

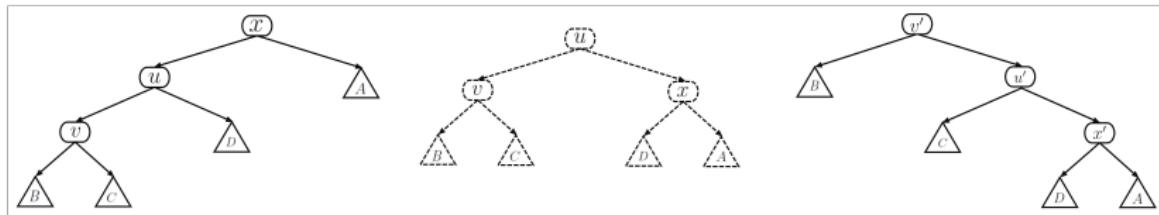
Beweis Access Lemma



$$-2 \geq \log_2 (tw(x') / tw(v')) + \log_2 (tw(v) / tw(v'))$$

Für $a, b \in R$ mit $a, b > 0$ und $a + b \leq 1$ gilt:
 $\log_2(a) + \log_2(b) \leq -2$

Beweis Access Lemma



$$-2 \geq \log_2 (tw(x') / tw(v')) + \log_2 (tw(v) / tw(v'))$$

Für $a, b \in R$ mit $a, b > 0$ und $a + b \leq 1$ gilt:
 $\log_2(a) + \log_2(b) \leq -2$

Beweis Access Lemma

$$\begin{aligned} & 3(r'(v) - r(v)) + 3(r''(v) - r'(v)) + 3(r'''(v) - r''(v)) \dots + 1 \\ &= 3(r(w) - 3r(v)) + 1 \end{aligned}$$

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
●oooooooo

Binärer Suchbaum

Dynamische Optimalität

Tango Baum

Splay Baum

Laufzeittests

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

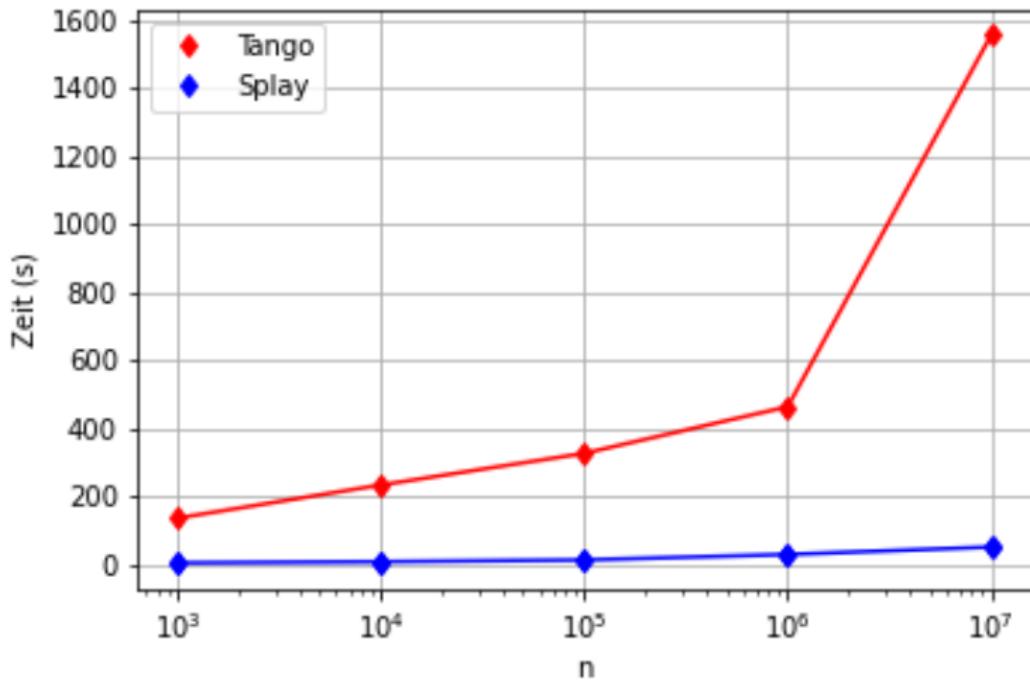
Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
●oooooooo

Zufällig erzeugte Zugriffsfolgen.

$$m = 40.000.000$$



Zufällig erzeugte Zugriffsfolgen

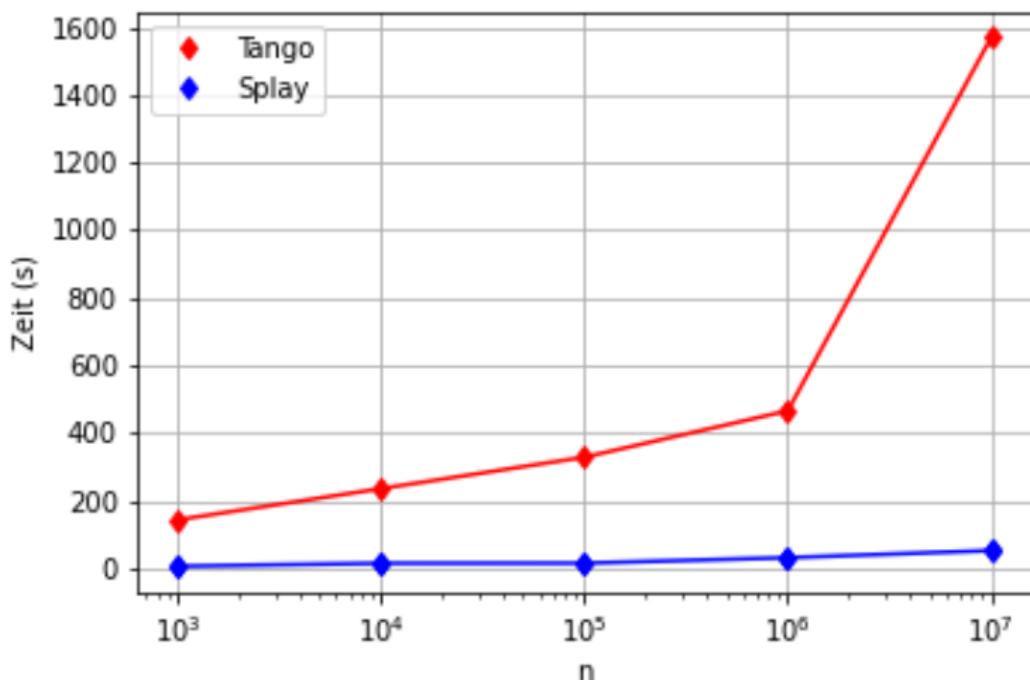


Abbildung: Beste Zeiten des Splay Baumes.

Zufällig erzeugte Zugriffsfolgen

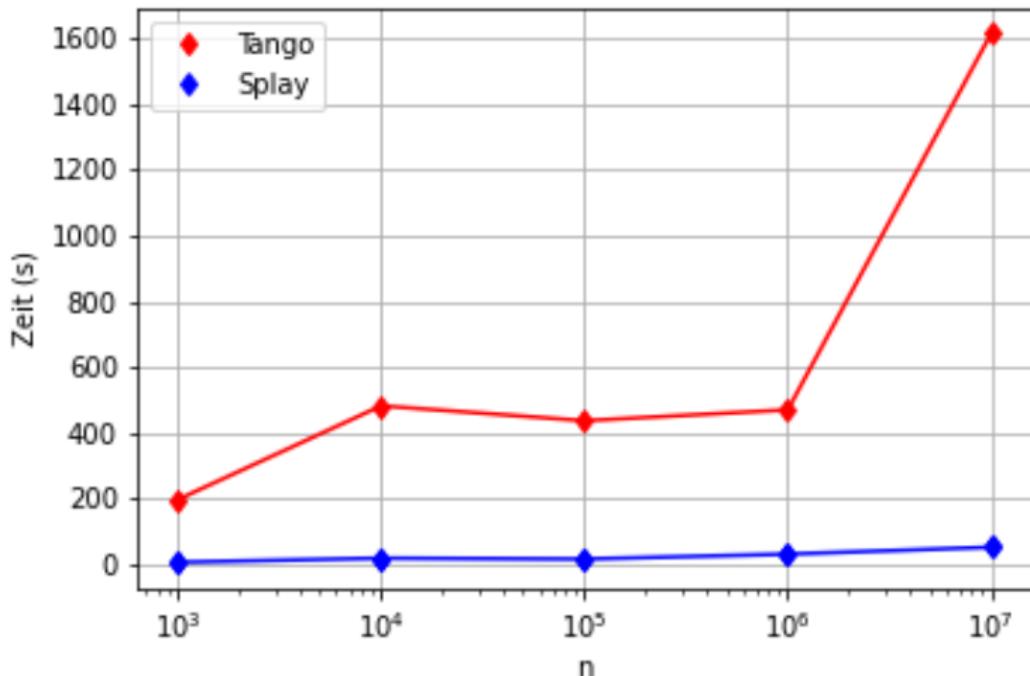


Abbildung: Durchschnittliche Werte.

Auffälligkeiten

1. Tango Kennlinie zu Durchschnittswerten
2. Große Differenzen zwischen Tango- und Splay Baum

Tango Kennlinie

Durchschnittswerte $n = 10.000$

i	Zeit Tango (s)	Zeit Splay (s)
1	354	16
2	841	21
3	345	15
4	819	31
5	352	15
6	344	16
7	334	16
8	329	16
9	328	15
10	332	16
avg	438	18

Dynamic Finger Property

- $X = x_1, x_2, \dots, x_m$
- $K = \{1, 2, \dots, n\}$

Definition

$$O \left(m + n + \sum_{i=2}^m \log (|x_{i-1} - x_i| + 1) \right)$$

Dynamic Finger Property

Splay Baum: ja

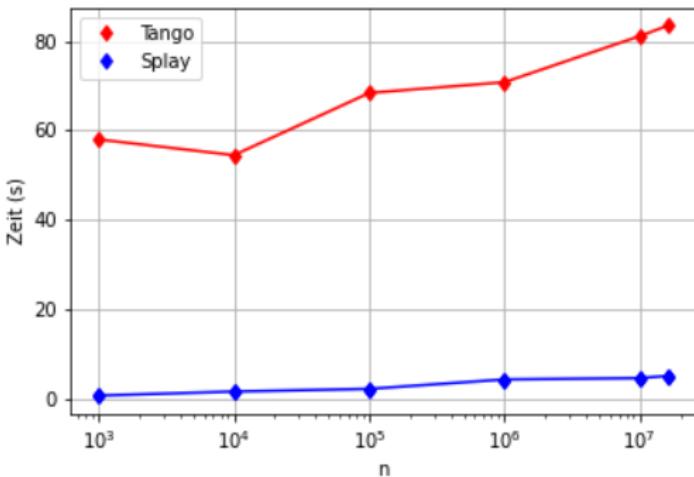
Richard Cole, Bud Mishra, Jeanette Schmidt, and Alan Siegel. On the dynamic finger conjecture for splay trees. part i: Splay sorting log n-block sequences. SIAM Journal on Computing, 2000.

Tango Baum: nein

Dynamic Finger Property

Zugriffsfolgen der Form:

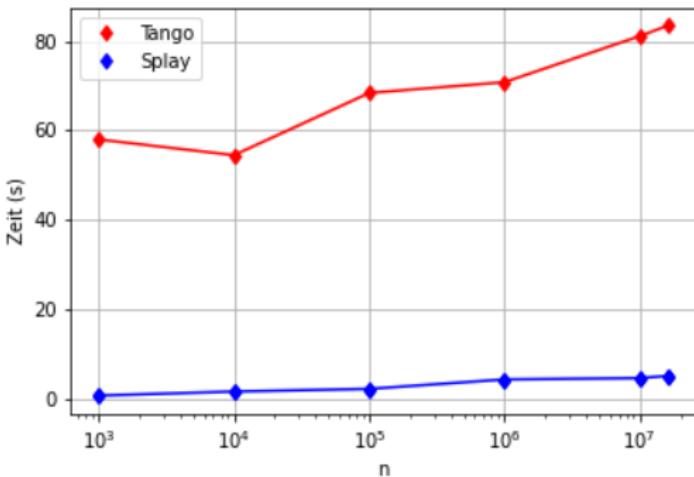
1, 3, 5, ..., $n - 1$, $n - 3$, $n - 5$, ..., 1, 3, 5..., $n - 1$, ...



Dynamic Finger Property

Zugriffsfolgen der Form:

1, 3, 5, ..., $n - 1$, $n - 3$, $n - 5$, ..., 1, 3, 5..., $n - 1$, ...



Dynamic Finger Property

n	Zeit Tango in (s)	Zeit Splay in (s)
10^3	58	0,8
10^4	54,5	1,7
10^5	68,4	2,3
10^6	70,8	4,4
10^7	81,1	4,7
$1,6 * 10^7$	83,3	5,2

Binärer Suchbaum
oooo

Dynamische Optimalität
ooooo

Tango Baum
oooooooooooooooooooooooooooo

Splay Baum
oooooooooooooooooooo

Laufzeittests
oooooooooooo

Vielen Dank für Ihre Aufmerksamkeit