

# Bachelorarbeit

Andreas Windorfer

11. Juli 2020

# Inhaltsverzeichnis

<b>1</b>	<b>Splaybaum</b>	<b>3</b>
1.1	<i>access</i> Operation . . . . .	3
1.2	Amortisierte Laufzeitanalyse von <i>splay</i> . . . . .	4

# 1 Splaybaum

Der in [1] vorgestellte Splaybaum ist ein online BST der ohne zusätzliche Hilfsdaten in seinen Knoten auskommt. Nach einer *access* Operation, ist der Knoten mit Schlüssel  $k$  die Wurzel, des Splaybaum. Es gibt keine Invariante, welche eine bestimmte maximale Höhe garantiert. Splaybäume können sogar zu Listen entarten. Amortisiert betrachtet verfügen sie dennoch über sehr gute Laufzeiteigenschaften.

## 1.1 *access* Operation

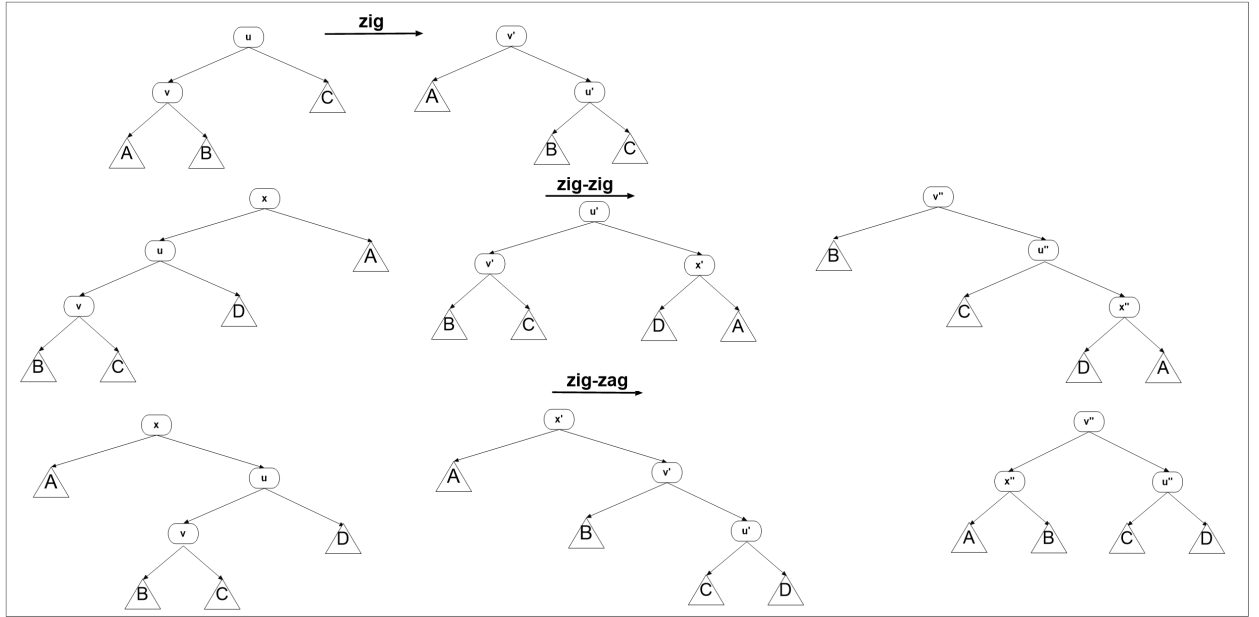
Die wesentliche Arbeit leistet eine Hilfsoperation namens *splay*. Nach deren Ausführung befindet sich der Knoten mit dem gesuchten Schlüssel an der Wurzel und es wird nur noch eine Referenz auf ihn zurückgegeben.

***splay* Operation** Sie  $p$  der Zeiger der Operation in den BST. Zunächst wird eine gewöhnliche Suche ausgeführt bis  $p$  auf den Knoten  $v$  mit Schlüssel  $k$  zeigt. Nun werden iterativ sechs Fälle unterschieden bis  $v$  die Wurzel des Baumes darstellt. Zu jedem Fall gibt es einen der links-rechts-symmetrisch ist. Sei  $u$  der Vater von  $v$ .

1.  $v$  ist das linke Kind der Wurzel (zig-Fall):  
Es wird eine Rechtsrotation auf  $v$  ausgeführt. Nach dieser ist  $v$  die Wurzel des Splaybaum und die Operation wird beendet.
2.  $v$  ist das linke Kind der Wurzel (zag-Fall):  
Symmetrisch zu zig.
3.  $v$  ist ein linkes Kind und  $u$  ist ein linkes Kind. (zig-zig-Fall):  
Dieser Fall unterscheidet den Splaybaum vom einem anderen BST (move-to-root), mit schlechteren Laufzeiteigenschaften. Es wird zuerst eine Rechtsrotation auf  $u$  ausgeführt und erst danach eine Rechtsrotation auf  $v$ . Bei move-to-root ist es genau anders herum.
4.  $v$  ist ein rechtes Kind und  $u$  ist ein rechtes Kind. (zag-zag-Fall):  
Symmetrisch zu zig-zig.
5.  $v$  ist ein linkes Kind und  $u$  ist ein rechtes Kind. (zig-zag-Fall):  
Es wird eine Rechtsrotation auf  $v$  ausgeführt. Im Anschluss wird eine Linksrotation auf  $u$  ausgeführt.
6.  $v$  ist ein rechtes Kind und  $u$  ist ein linkes Kind. (zag-zig-Fall):  
Symmetrisch zu zig-zag.

Abbildung 1 zeigt drei der Fälle. Trotz der Einfachheit kann die Auswirkung einer einzelnen *splay* Operation sehr groß sein. Abbildung 2 aus [1] zeigt eine solche Konstellation.

Die Laufzeit von *access* auf einem BST mit  $n$  Knoten ist  $O(n)$ .



**Abbildung 1:** Darstellung von zig, zig-zag und zig-zig.

## 1.2 Amortisierte Laufzeitanalyse von *splay*

Es wird die Potentialfunktionsmethode aus Kapitel ?? verwendet. Sei  $v$  ein Knoten im Splaytree  $T$ . Eine Funktion  $w(v)$  liefert zu jedem Knoten eine reelle Zahl  $> 0$ , die Gewicht genannt wird. Das Gewicht eines Knotens ist unveränderlich. Eine Funktion  $tw(v)$  bestimmt die Summe aller Gewichte im Teilbaum mit Wurzel  $v$ . Der Rang  $r(v)$  ist definiert durch  $r(v) = \lfloor \log_2 tw(v) \rfloor$ . Sei  $V$  die Menge der Knoten von  $T$ . Als Potentialfunktion wird

$$\Phi = \sum_{v \in V} r(v)$$

verwendet.

**Access Lemma 1.1.** *Sei  $T$  ein Splaybaum mit  $n$  Knoten und Wurzel  $w$  und einem Knoten  $v$  mit Schlüssel  $k$ . Die amortisierte Laufzeit von  $splay(k)$  ist maximal  $3(r(w) - r(v)) + 1 = O(\log(tw(w)) / tw(v)) = O(\log(n))$*

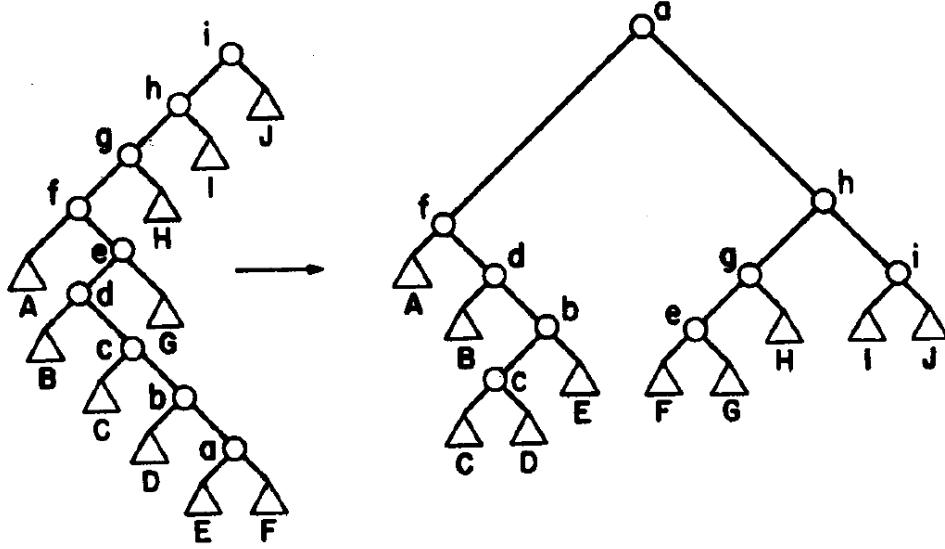


FIG. 4. Splaying at node  $a$ .

Abbildung 2: Eine einzige *splay* Operation. [1]

*Beweis.* Zunächst wird für *zig*, *zig-zag* und *zig-zig* gezeigt dass die amortisierten Kosten nicht größer als  $3(r(v)' - r(v)) + 1$ . Für die anderen drei Fälle folgt es dann aus der Symmetrie. Im Anschluss wird die gesamte Operation betrachtet. An Abbildung 1 ist zu erkennen, dass sich der Wert von  $tw()$  nur bei den Knoten  $v$ , dessen Vater  $u$  und dem Vater  $x$  von  $u$  verändern kann. Damit gilt

$$\Phi' - \Phi = r(u)' + r(v)' + r(x)' - r(u) - r(v) - r(x)$$

**zig** In diesem Fall existiert  $x$  nicht, damit gilt

$\Phi' - \Phi = r(u)' + r(v)' - r(u) - r(v)$ . Der Wert von  $tw()$  für die Wurzel ist unabhängig von Zustand des Splaytree, da immer alle vorhandenen Gewichte aufsummiert werden. Deshalb muss  $tw(v)' = tw(u)$  gelten. Daraus folgt  $\Phi' - \Phi = r(u)' - r(v)$ . Aus  $r(v)' \geq r(u)'$  folgt  $\Phi' - \Phi \leq r(v)' - r(v) \leq 3(r(v)' - r(v))$ . Addieren von 1 aufgrund der Rotation ergibt Kosten  $\leq 3(r(v)' - r(v)) + 1$ .

**zig-zig** Es müssen zwei Rotationen ausgeführt werden, deshalb entstehen amortisierte Kosten von

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

Nun wird zunächst die Behauptung aufgestellt, dass diese Summe klein genug ist, dies wird dann über Äquivalenzen gezeigt.

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

Dass die letzte Ungleichung gilt, kann man an einer Eigenschaft des  $\log_2$  ableiten. Für  $a, b \in R$  mit  $a, b > 0$  und  $a + b \leq 1$  gilt  $\log_2(a) + \log_2(b) \leq -2$ . An Abbildung 1 ist zu erkennen, dass sich  $tw(v)$  vom Ausgangszustand zum Zwischenzustand hin nicht verändert.  $tw(x')$  ist ebenfalls unverändert zum Zwischenschritt. Es kann also bei beiden Knoten mit den Werten aus dem Zwischenschritt gearbeitet werden.  $tw(v') = tw(x') + tw(v) + w(u)$ . Daraus folgt  $((x') + (v))/tw(v') < 1$  und mit der Eigenschaft von  $\log_2$  folgen die Ungleichungen.

**zig-zag**

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

Nun wird wie bei zig-zig vorgegangen.

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

Mit der  $\log_2$  Regel aus zig-zig und Abbildung 1 folgt die Behauptung. Betrachtet man die Kosten der Gesamtoperation so bildet sich eine Teleskopsumme, die möglicherweise, wenn ein zig bzw. zag Fall enthalten ist, mit eins addiert werden muss. Daraus folgt das Lemma.  $\square$

## Literatur

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