Bachelorarbeit

Andreas Windorfer

11. August 2020

Zusammenfassung

Inhaltsverzeichnis

1 Weitere binäre Suchbäume

Hier werden kurz zwei Variationen zum Tango Baum vorgestellt. Zum einen der Zipper Baum. Er wurde in [1] vorgestellt und ist ebenfalls $\log(\log(n))$-competitive, garantiert aber auch $O(\log(n))$ im worst case, bei einer einzelnen *access* Operation. n steht wieder für de Anzahl der Knoten von T. Zum anderem den Multisplay Baum [2]. Bei diesem wird ein preferred path durch einen Splay Baum repräsentiert. Amortisiert betrachtet erreicht er $O(\log(n))$ bei *access* und ist $\log(\log(n))$-competitive.

1.1 Zipper Baum

Der Zipper Baum basiert auf dem Tango Baum und nutzt auch preferred paths aus einem Referenzbaum P. Aufbau und Pflege der preferred paths in P unterscheiden sich nicht vom Tango Baum. Ihre Repräsentation im eigentlichen BST T, macht den wesentlichen Unterschied zu einem Tango Baum aus. Abbildung 1 stellt eine solche beispielhaft dar. Sei v ein Knoten in T, dann ist in diesem Kapitel v^* der Knoten in P, mit $key(v) = key(v^*)$. Die Repräsentation eines preferred path $P_p = {p_1}^*, {p_2}^*, .., {p_m}^*$ in T stellt einen Hilfsbaum H dar, der in zwei Teile unterteilt wird, dem **zipper** und dem **bottom tree**. Der bottom tree ist ein balancierter BST der genau die Schlüssel enthält, die in P_p enthalten sind jedoch nicht im zipper. Der zipper besteht aus zwei Teilen, dem **top zipper** z_t und dem **bottom zipper** z_b. z_t und z_b dürfen jeweils maximal $\log_2(\log_2(n))$ Knoten enthalten. Gemeinsam müssen enthalten sie zumindest $\log_2(\log_2(n))/2$ Knoten, wenn ein bottom tree existiert. Bei weniger als $\log_2(\log_2(n))/2$ Knoten im Hilfsbaum existiert kein bottom tree. Es wird im folgenden angenommen, dass ein bottom tree existiert.
P_p wird in **zig Segmente** und **zag Segmente** unterteilt. zig Segmente entsprechen den längst möglichen Pfaden von Knoten mit rechten Kindern in P_p. zag Segmente entsprechen den längst möglichen Pfaden von Knoten mit linken Kindern in P_p. Das Blatt in P_p wird dem Segment seines Elternknoten zugeordnet. In Abbildung 2 sind zig und zag Segmente dargestellt.
Sei t_l bzw. t_r der Knoten in S_{zig} bzw. S_{zag} mit der kleinsten Tiefe. t_l ist die Wurzel von H. t_r ist das rechte Kind von t_l. Der linke Teilbaum von t_l hat Listenform und wird von den restlichen Knoten in S_{zig} gebildet, so dass kein Knoten in diesem Teilbaum ein linkes Kind hat. Der rechte Teilbaum von t_r hat Listenform und wird von den restlichen Knoten in S_{zag} gebildet, so dass kein Knoten in diesem Teilbaum ein rechtes Kind hat.
z_b wird analog aus P_b erzeugt und seien b_l und b_r die Knoten in z_b entsprechend zu t_l und t_r in z_t. b_l ist das linke Kind von t_r. Die Wurzel des bottom Tree ist das linke Kind von t_r. Dass die Links-Rechts-Beziehung eingehalten

wird ergibt aus den Aufbau der zig und zag Segmente. Es ist leicht zu erkennen, dass die Wurzel des bottom tree in konstanter Zeit erreicht werden kann.

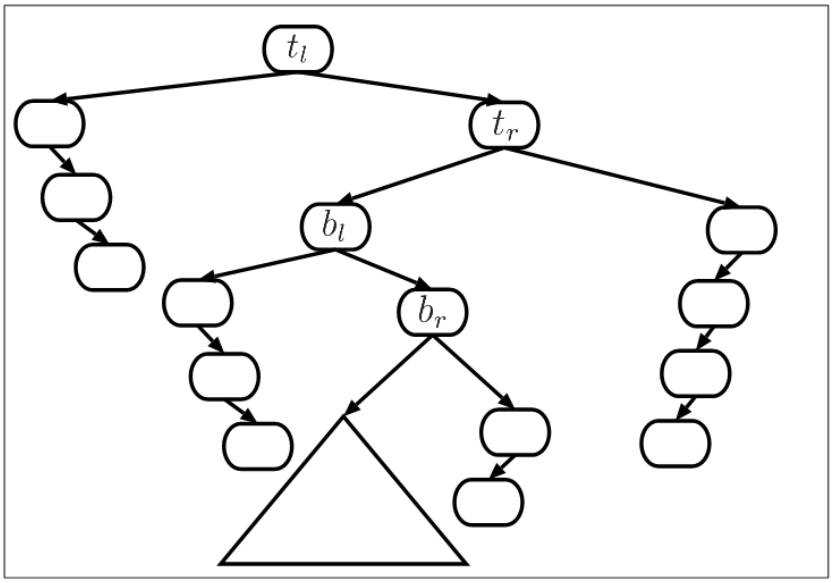

Abbildung 1: Pfadrepräsentation beim Zipper Baum, basiert auf einer Abbildung aus [1].

Besonderheiten bei *access* Sei l die Anzahl der Knoten des top zipper. Beim Suchen nach einem Schlüssel in einem Hilfsbaum H, wird dessen top zipper, mit l Knoten, soweit wie notwendig, in einen Pfad gewandelt, der $p_1, p_2, .., p_l$ entspricht. Ist der top zipper vollständig in einen Pfad umgewandelt, wird der Vorgang beim bottom zipper fortgesetzt. Dieser hat dann die Stellung des top zipper. Außerdem wird dann ein **Extraktionsprozess** angestoßen, der $\log(\log(n))$ Knoten aus dem bottom tree auslagert, um einen neuen bottom zipper zu erzeugen. Ein Extraktionsprozess ist in Abbildung 3 dargestellt.
Ein Knoten aus dem top zipper kann dem Pfad in konstanter Zeit hinzugefügt werden. Ist bei $access(k)$ der top zipper aufgebracht, sind bereits Kosten von $O(\log(\log(n)))$ entstanden. Sei P_p der preferred path den H repräsentiert. Ist der gesuchte Knoten im top zipper enthalten, entstehen in P aufgrund der Knoten von P_p asymptotisch betrachtet die gleichen Kosten, wie in H. Befindet sich der gesuchte Knoten im bottom zipper oder dem bottom tree entstehen in H ebenfalls Kosten von $O(\log(\log(n)))$. Deshalb entstehen asymptotisch betrachtet in H innerhalb P_p die gleichen Kosten. Hieraus wird

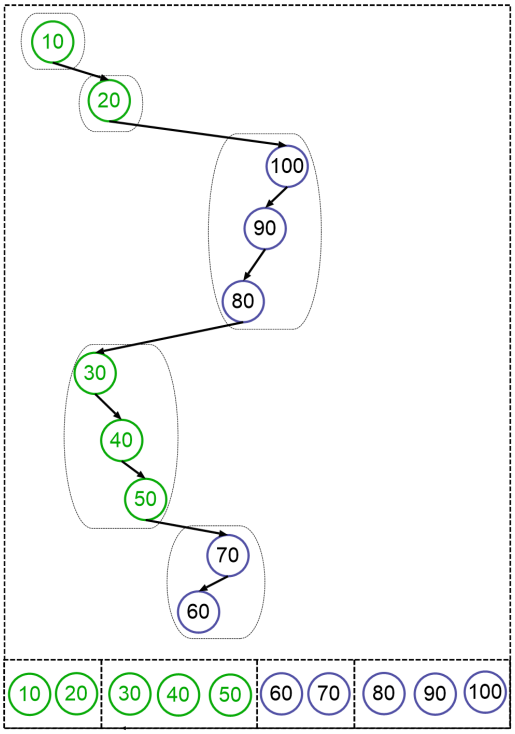

Abbildung 2: zig Segmente sind grün dargestellt. zag Segmente blau

abgeleitet, dass eine *access* Operation in $O(\log(n))$ Zeit ausgeführt werden kann.

1.2 Multisplay Baum

Ein preferred path wird hier durch einen Splay Baum dargestellt, um dessen Laufzeiteigenschaften nutzen zu können. Da der Splay Baum kein balancierter Baum ist, gibt es zusätzliche mögliche Zustände im Vergleich zu einem Tango Baum mit der gleichen Knotenzahl. Zu den genannten Eigenschaften bezüglich der Laufzeit sind Beweise in [2] enthalten. Der Multisplay Baum erfüllt die working set Eigenschaft [3].

Die *access* Operation beim Multisplay Baum Zu beachten ist, dass jede BST Darstellung auch eine Splay Baum Darstellung ist. Anders als beim Tango oder Zipper Baum, muss ein neu erzeugter Hilfsbaum also nicht so angepasst werden, dass er weitere Invarianten einhält. Nach einer $access(k)$ Operation ist der Knoten v_k mit dem Schlüssel k die Wurzel von T. Zunächst wird eine gewöhnliche Suche in T durchgeführt, bis der Zeiger p der Operation auf v_k zeigt. Ist v_k gefunden, werden die Pfadrepräsentationen aktualisiert.

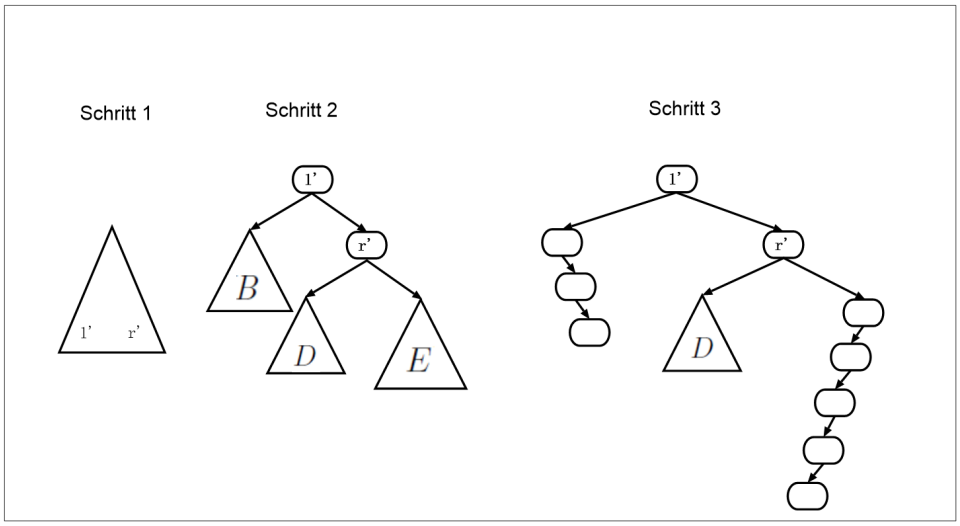

Abbildung 3: Extraktionsprozess beim Zipper Baum

Hierzu muss der Hilfsbaum, der die Wurzel von T enthält neu erzeugt werden.

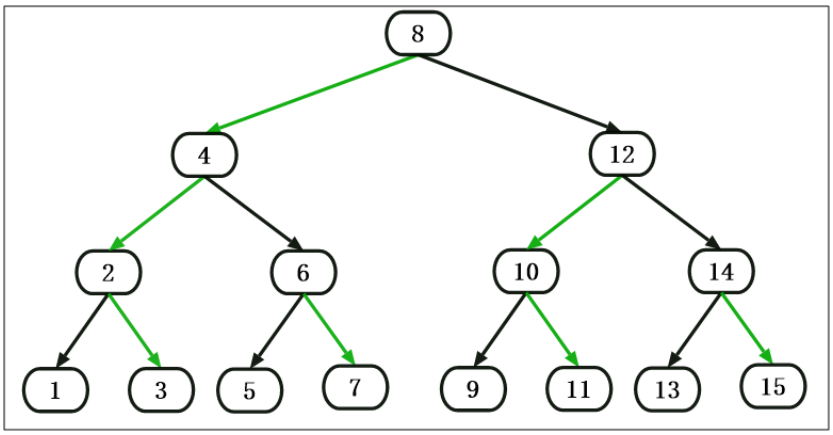

Abbildung 4: Referenzbaum mit grün gezeichneten preferred paths

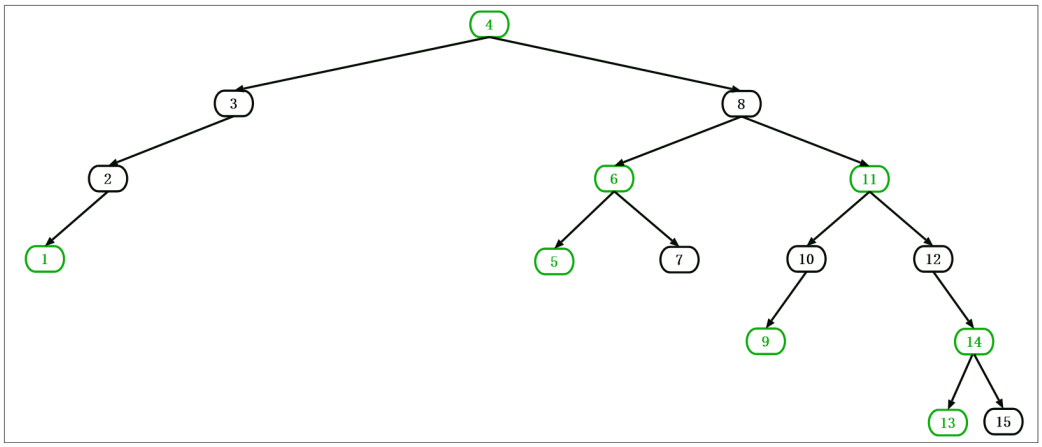

Abbildung 5: Beispielhafter Multisplay Baum zu Abbildung 4.

Literatur

[1] Prosenjit Bose, Karim Douïeb, Vida Dujmović, and Rolf Fagerberg. An o(log log n)-competitive binary search tree with optimal worst-case access times. *Algorithm Theory - SWAT 2010*, page 38–49, 2010.

[2] Daniel Dominic Sleator and Chengwen Chris Wang. Dynamic optimality and multi-splay trees. Technical report, 2004.

[3] Chengwen Chris Wang. *Multi-Splay Trees*. PhD thesis, USA, 2006.

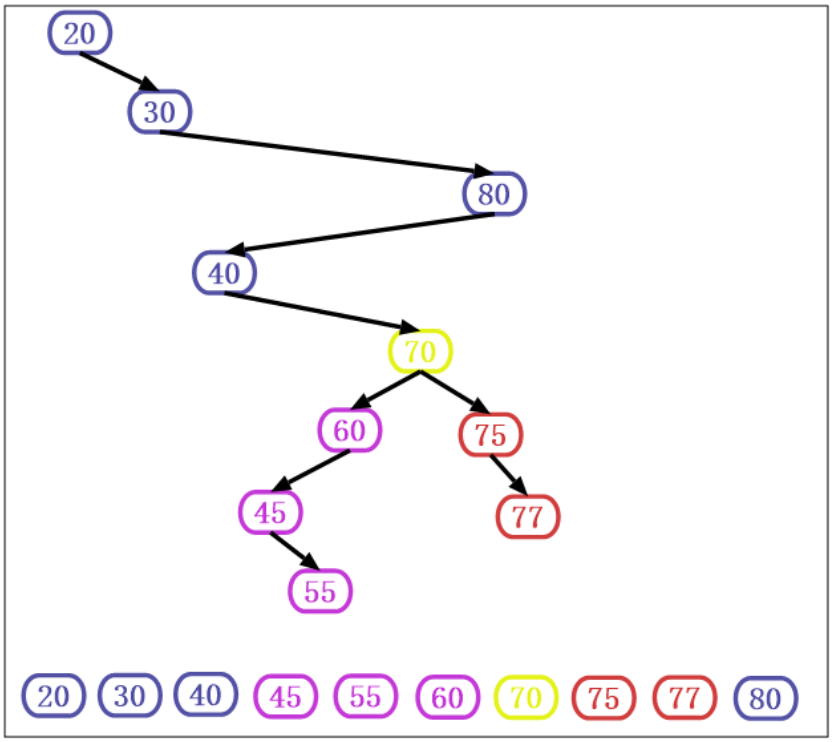

Abbildung 6: Beispielhafte Zusammenhänge der Schlüsselgrößen. U ist blau dargestellt, L lila, R rot und q_i gelb.