Bachelorarbeit

Andreas Windorfer

3. Juli 2020

Zusammenfassung

Inhaltsverzeichnis

1 Zipper-Baum

Der Zipper-Baum basiert auf dem Tango-Baum und nutzt auch preferred paths aus einem lower-bound-tree P. Aufbau und Pflege der preferred paths in P unterscheiden sich nicht vom Tango-Baum, wohl aber ihre Repräsentation im eigentlichen BST T. Der Zipper-Baum wurde in [1] vorgestellt. Er ist ebenfalls $\log(\log(n))$-competitive, garantiert aber auch $O(\log(n))$ bei einer einzelnen *access* Operation. n steht wieder für de Anzahl der Knoten von T. Das Verhalten der Operationen *cut* und *concatenate* unterscheidet sich deutlich.

1.1 Repräsentation eines preferred path

Ein Hilfsbaum H zur Repräsentation eines preffered path $P_p = p_1, p_2, .., p_p$ wird in einen **zipper** und einem **bottom tree** unterteilt. Enthält der preferred path nicht mehr als $O\log(\log(n))$ Knoten, besteht der Hilfsbaum allein aus dem zipper. Die Anzahl der Knoten des zipper liegt in $[\log(\log(n))/2, 2\log(\log(n))]$, wenn ein bottom tree existiert, ansonsten in $[0, 2\log(\log(n))]$. Der bottom tree ist ein balancierter BST, der genau die Schlüssel aus P_p enthält, die im zipper fehlen. Enthält der zipper q Knoten, dann entsprechen deren Schlüsseln denen aus dem Pfad $p_1, p_2, .., p_q$. Der zipper ist ein BST und die Wurzel des bottom tree ist das Kind eines Knotens aus dem zipper, so dass H ein BST ist.
Die Konstruktion des zipper ist so ausgelegt, dass innerhalb konstanter Zeit von der Wurzel von H auf die Wurzel des bottom tree zugegriffen werden kann. Es gibt in der Regel mehrere mögliche Darstellungen eines zipper zu P_p. Ein zipper z besteht ebenfalls wieder aus zwei Bestandteilen dem **oberen zipper** z_1 und dem **unterem zipper** z_2. Insgesamt müssen in z zumindest $\log(\log(n))/2$ enthalten sein. Pro Bestandteil dürfen maximal $\log(\log(n))$ Knoten enthalten sein. Sei a_1 die Anzahl der Knoten in z_1 und a_2 die in z_2, so dass die genannten Anforderungen eingehalten werden. Konstellationen in denen das nicht möglich ist, bleiben zunächst außen vor. In z_1 sind die Schlüssel der Knoten des Pfades $P_1 = p_1, p_2, .., p_{a_1}$ enthalten. In z_2 die aus $P_2 = p_{a_1+1}, p_{a_1+2}, .., p_{a_1+a_2}$.
P_1 wird in **zig Segmente** und **zag Segmente** unterteilt. zig Segmente entsprechen den längst möglichen Teilpfaden von Knoten mit linken Kindern in P_p. zag Segmente entsprechen den längst möglichen Teilpfaden von Knoten mit rechten Kindern in P_p. Enthält P_2 das Blatt aus P_p, wird dieses dem Segment seines Vaters zugeordnet. In Abbildung 1 sind zig und zag Segmente dargestellt. Sei S_1 die Folge der Knoten der zig Segmente, aufsteigend sortiert nach der Tiefe und S_2 die Folge der Knoten der zag Segmente, auf-

steigend sortiert nach der Tiefe der enthaltenen Knoten. Ist u der tiefste Knoten eines zig bzw. zag Segmentes, so können in den Segmenten mit Knoten größerer Tiefe nur noch größere bzw. kleinere Schlüssel enthalten sein, vergleiche Abschnitt **??** und Abbildung 1. Deshalb müssen die Knoten in $S_1 \circ S_2$ aufsteigend sortiert nach Schlüssel sein. Da die Knoten in S_1 bzw. S_2 aber auch aufsteigend bzw. absteigend nach Tiefe sortiert sind, können aus $S_1 \circ S_2$ Intervalle für die Schlüsselmengen, von Pfaden in P_p mit Endknoten p_p abgeleitet werden, siehe Abbildung 1, diese werden in den nachfolgend vorgestellten Operationen benötigt.
Sei nun l der Knoten in S_1 mit der kleinsten Tiefe, und r der Knoten in S_2 mit der kleinsten Tiefe. Der Knoten mit Schlüssel $key\,(l)$ ist die Wurzel von H. Der Linke Teilbaum von l enthält die Schlüssel der Knoten in S_1 mit einer Tiefe $\leq a_1$

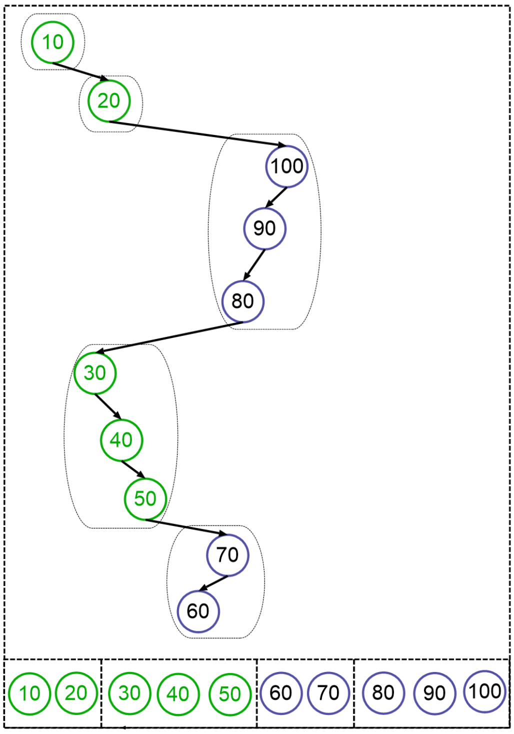

Abbildung 1: zig Segmente sind grün dargestellt. zag Segmente blau

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.