Bachelorarbeit

Andreas Windorfer

25. Juli 2020

Inhaltsverzeichnis

1 Multisplay Baum

Der Multisplay Baum [1] T ist eine Variation zum Tango Baum. Ein preferred path wird hier durch einen Splaybaum dargestellt. Amortisiert betrachtet, ist er $\log(\log(n))$-competitive und garantiert $O(\log(n))$ bei einer einzelnen *access* Operation. n steht wieder für der Anzahl der Knoten von T. Da der Splaybaum kein balancierter Baum ist, gibt es zusätzliche mögliche Zustände im Vergleich zu einem Tango Baum mit der gleichen Knotenzahl. Auch der

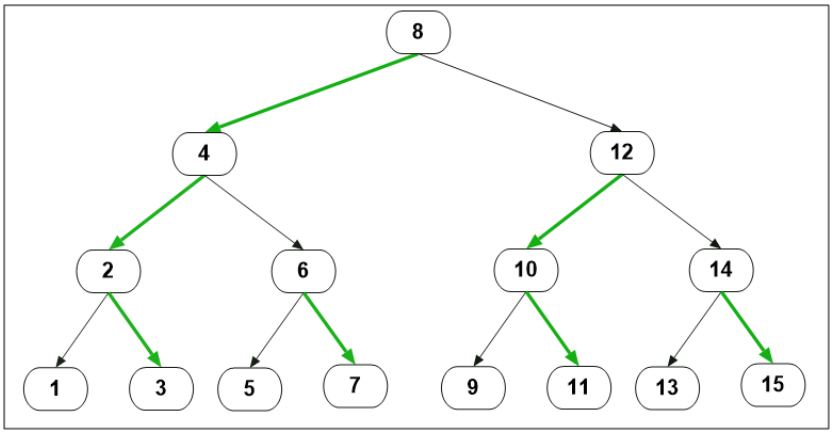

Abbildung 1: Refernzebaum mit grün gezeichneten preferred paths

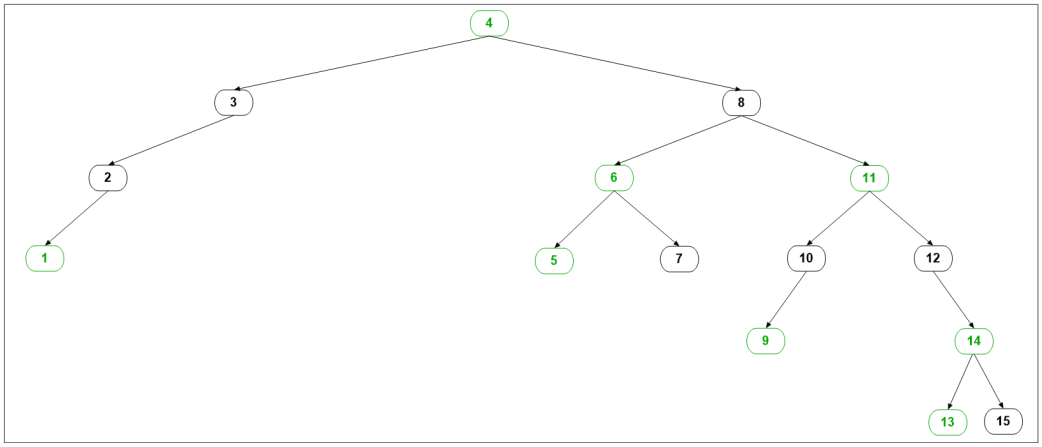

Abbildung 2: Beispielhafter Multisplay Baum zu Abbildung 1.

Multisplay Baum verwendet einige Hilfsdaten je Knoten. Zum einen bereits bekannten Variablen bzw. Konsanten *isRoot*, *depth* und *minDepth*. Aber auch eine die beim Tango Baum nicht verwendet sind. Sei v ein Knoten in T,

dann wird mit v^* wieder der Knoten im Referenzbaum P bezeichnet, mit $key(v) = key(v^*)$. Jeder Knoten v in T enthält eine *prefChild* Variable, mit der gespeichert welches Kind von v^*, das preferred child ist.

1.1 Die *access* Operation beim Multisplay Baum

Zu beachten ist, dass jede BST Darstellung auch eine Splaybaum 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. Eine Abweichung zu den preferred path des Tango Baum ist, dass das preferred child des Knoten mit Schlüssel k zunächst unverändert bleibt. Ist v_k gefunden, werden die Pfadrepräsentationen aktualisiert. Dabei wird bottom up vorgegangen. Nach dem aktualisieren der Pfadrepräsentationen muss noch $splay(k)$ ausgeführt werden, um v_k zur Wurzel von T zu machen. In den Beschreibungen von *cut* und *join* wird von einem zugrunde liegenden preferred child Wechsel vom linken Kind zum Rechten ausgegangen. Der andere Fall ist symmetrisch.
Sei $P_p = {q_1}^*, {q_2}^*, .., {q_m}^*$ ein preferred path und ${q_i}^*$, mit $i \in \{1, 2, .., m\}$, der Knoten an dem das preferred child wechselt. Sei k der Schlüssel von q_i. Sei A der Hilfsbaum der P_p repräsentiert. Sei $U = \{q_1, q_2, .., q_i\}$ und $L = \{q_{i+1}, q_{i+2}, .., q_m\}$. Sei q_r das rechte Kind von q_i und B der Hilfsbaum in dem q_r enthalten ist.

***switch* Operation beim Multisplay Baum** Beim Multisplay Baum werden *cut* und *join* zu einer Operation *switch* zusammengefasst. Abbildung 3 stellt die Zusammenhänge der Schlüssel dar. Das Vorgehen ist sehr ähnlich zu dem aus **??** und **??** sehr ähnlich, weshalb hier weniger detailliert darauf eingegangen wird. Zunächst wird $splay(k)$ auf A ausgeführt. Dadurch entsteht ein Hilfsbaum C mit Wurzel q_i. Sei C_L der linke Teilbaum von q_i und C_R der Rechte.
Es wird der Knoten $q_{l'}$, mit dem kleinsten Schlüssel $l' > key(q_i)$, aus U benötigt. Es wird angenommen, dass dieser existiert. Auf den anderen Fall wird noch eingegangen. ${q_{l'}}^*$ muss eine kleinere Tiefe als q_i^* haben. Deshalb kann $q_{l'}^*$ gefunden werden indem wie folgt vorgegangen wird. p muss auf die Wurzel von C_L gesetzt werden. In einer Schleife wird p so oft auf das linke Kind p_l von p gesetzt, bis der Wert der *minDepth* Variable von p_l größer als die Tiefe von q_i^* ist.
Nun wird $splay(l')$ auf C_l ausgeführt. Ist der rechte Teilbaum D_R von $q_{l'}$ nicht leer, muss bei seiner Wurzel *isRoot* noch auf *true* gesetzt werden. Abbildung

4 stellt es nochmals dar. Der *cut*-Teil ist nun abgeschlossen und wir kommen zum *join*-Teil.
Es wird der Knoten $q_{r'}$, mit dem kleinsten Schlüssel $r' > key(q_i)$, aus U benötigt. Wieder wird angenommen, dass dieser existiert. $q^*_{r'}$ muss eine kleinere Tiefe als q^*_i haben. Deshalb kann $q^*_{r'}$ gefunden werden indem wie folgt vorgegangen wird. p muss auf die Wurzel von C_R gesetzt werden. In einer Schleife wird p so oft auf das rechte Kind p_r von p gesetzt, bis der Wert der *minDepth* Variable von p_r größer als die Tiefe von ${q_i}^*$ ist.
Es wird $splay(r')$ auf C_R ausgeführt. Nun wird der linke Teilbaum D_L von q_r betrachtet. Jeder Knoten in B muss in D_L enthalten sein, denn für $v_R \in R$ gilt $k < key\,(v_R) < r'$. Kein Knoten aus $L \cup U$ kann in D_L enthalten sein, da für $v_L \in L$, $key\,(v_L) < k$ und für für $v_U \in U$, $key\,(v_U) > r'$ gilt. Somit muss das linke Kind von $q_{r'}$ die Wurzel von B sein. Die *isRoot* Variable dieses Knotens wird auf *false* gesetzt. Ist $q_{l'}$ bzw. $q_{r'}$ nicht vorhanden, sind in C_L

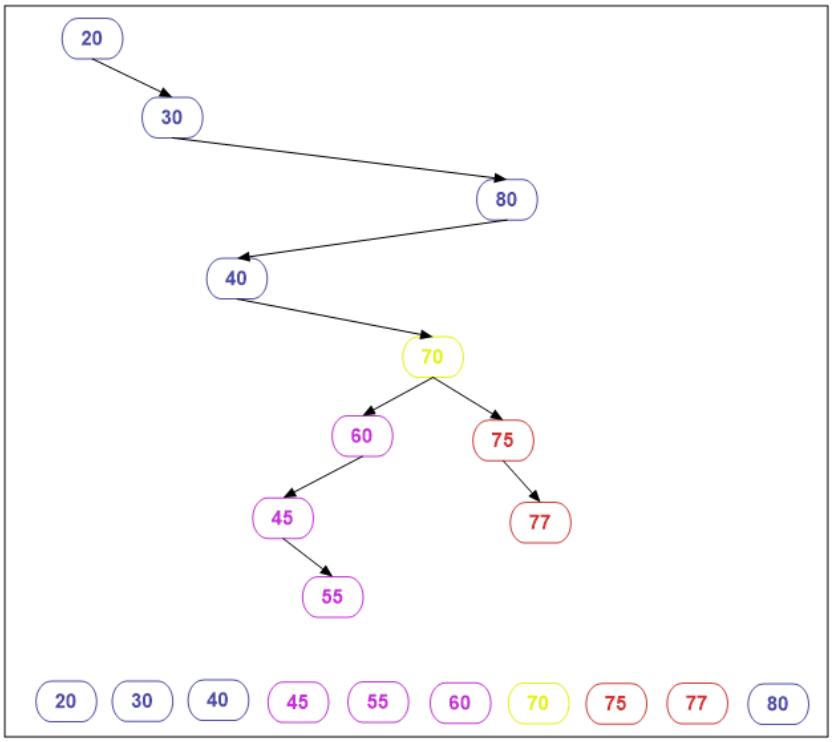

Abbildung 3: Beispielhafte Zusammenhänge der Schlüsselgrößen. U ist blau dargestellt, L lila, R rot und q_i gelb.

bzw. C_R keine Knoten aus U enthalten und die jeweilige *splay* Operation kann entfallen. Aufgrund der Laufzeit, darf aber nach diesen Knoten nur gesucht werden, wenn sie vorhanden sind. Nach Durchführung der ersten *splay*

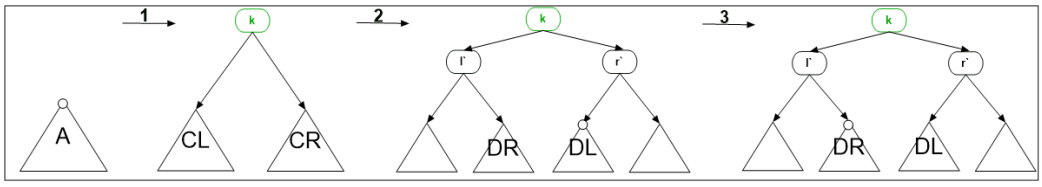

Abbildung 4: Ablauf zum erzeugen einer neuen Pfadrepräsentation, nach einem preferred child Wechsel vom linken Kind zum Rechten.

Operation kann auf die Existenz dieser Knoten geschlossen werden. Es wird die Vorgehensweise zu $q_{l'}$ beschrieben, die zu $q_{r'}$ ergibt sich daraus. Es muss $q_{l'}.depth > q_i.depth$ gelten. Ist die *minDepth* Variable des linken Kindes von v_k kleiner als $q_i.depth$ muss $q_{l'}$ existieren, ansonsten kann er nicht existieren.

1.2 Amortisierte Laufzeitanalyse

Alle Lemmas und Sätze sind in [1] enthalten. Sei v ein Knoten im Multisplay Baum T und H der Hilfsbaum der v enthält. Eine Funktion $w(v)$ liefert zu jedem Knoten eine reelle Zahl > 0, die Gewicht genannt wird. Eine Funktion $tw(v)$ bestimmt die Summe aller Gewichte, der Knoten die im Teilbaum mit Wurzel v und in H enthalten sind. Der Rang $r(v)$ ist definiert durch $r(v) = \log_2 tw(v)$. Sei V die Menge der Knoten von T. Als Potentialfunktion wird

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

verwendet.
Das Generalized Access Lemma verallgemeinert das Access Lemma vom Splay Baum. *splay*(k) stoppt normalerweise erst, wenn der Knoten v_k mit Schlüssel k die Wurzel bildet. Sie kann aber ganz einfach modifiziert werden, so dass sie v_k zu einem Knoten mit Tiefe d macht, mit $v_k.depth > d$.

Generalized Access Lemma 1.1. *Sei T ein Multisplay Baum. Sei v ein Knoten im Hilfsbaum H von T. H enthalte n Knoten. Sei u ein Vorfahre von v in H. Es wird eine (modifizierte) splay Operation verwendet, so dass $v'.depth = u.depth$ gilt. Die amortisierten Kosten von splay(k) ist maximal* $3(r(u) - r(v)) + 1 = O(\log(tw(u))/tw(v)) = O(\log(n))$

Beweis. Folgt direkt aus **??**. Dass u die Wurzel ist wird dort nicht verwendet. □

Im nächsten Lemma geht es um die Gesamtlaufzeit einer *access* Operation beim Multisplay Baum T mit Referenzbaum P. Sei p^* ein Knoten in P. Es werden drei Funktionen benötigt. $des\,(p^*)$ liefert die Menge der Nachfahren von p^*. $pa\,(p^*)$ liefert die Menge der Pfade von p^* zu einem Blatt. $node(k)$ liefert den Knoten aus T der Schlüssel k enthält. An die Gewichtsfunktion werden zwei Bedingungen gestellt. Sei $f \geq 2$.

1. $w\,(p) \geq \max\{w\,(v)\,|v^* \in des\,(p^*)\}$
2. $fw\,(p) \geq \max\{\sum_{v^* \in P_p} w\,(v)\,|P_p \in pa\,(p^*)\}$

Multisplay Baum Access Lemma 1.1. *Sei T ein Multisplay Baum mit Referenzbaum P. Sei v^* die Wurzel von P. Sei f eine Konstante ≥ 2. Die amortisierte Laufzeit zum Ausführen der Zugriffsfolge $X = x_1, x_2, .., x_m$ ist*

$$O\left(\sum_{i=1}^{m} \log_2\left(\frac{w\,(v)}{w\,(node\,(x_i))}\right) + (\log_2 f)\,(IB\,(X) + m)\right)$$

Beweis. Der Beweis enthält drei Teile. Zuerst geht es um die Kosten einer *switch* Operation, dann um die Kosten einer *access* Operation. Zum Schluss wird die Relation zur Interleave Bound hergestellt.
Es wird eine *switch* Operation während $access(x_i)$ betrachtet. Seien ${y_1}^*, {y_2}^*, .., {y_k}^*$ die Knoten in P bei denen sich, während $access(x_i)$ der preferred child ändert. Für $i \in \{1, 2, .., k\}$ sei t_i die Wurzel des Splay Baumes der y_i enthält und t_{k+1} die Wurzel des Splay Baumes der den Knoten v, mit $key(v) = x_i$, enthält.
Es werden nun die Kosten von *switch* aufgrund dem Wechsel des preferred child an ${y_i}^*$ betrachtet. Und hierbei zunächst die Potentialänderung aufgrund der Änderung an den *isRoot* Variablen in Schritt 3 (Abbildung 4). Variablen ohne Hochstrich beziehen sich auf den Zustand vor Schritt 3, jedoch nach den *splay* Operationen. Sei y_l das linke Kind von y_i und y_r das Rechte, vor Ausführung von Schritt 3. Sei w_1 das rechte Kind von y_l und w_2 das linke Kind von y_r (w_2 ist die Wurzel eines Hilfsbaumes) . Abbildung 4 ist zu entnehmen, dass sich die Werte von $tw\,()$ nur an den Knoten y_l, y_i und y_r ändern kann.

$$
\begin{aligned}
\Delta\Phi &= (r(y_l') - r(y_l)) + (r(y_i') - r(y_i)) + (r(y_r') - r(y_r)) && (1)\\
&\leq (r(y_i') - r(y_i)) + (r(y_r') - r(y_r)) && (2)\\
&= \log_2(tw(y_i')/tw(y_i)) + \log_2(tw(y_r')/tw(y_r)) && (3)\\
&\leq \log_2\left(\frac{tw(y_i) + tw(w_2)}{tw(y_i)}\right) + \log_2\left(\frac{tw(y_r) + tw(w_2)}{tw(y_r)}\right) && (4)\\
&\leq \log_2(1 + tw(w_2)/w(y_i)) + \log_2(1 + tw(w_2)/w(y_i)) && (5)\\
&\leq 2\log_2(1+f) && (6)\\
&= O(\log_2(f)) && (7)
\end{aligned}
$$

Begründungen:

1. -
2. $r(y_l') \leq r(y_l)$
3. -
4. $tw(w_1) \geq 0$
5. Bedingung 1 an die Gewichtsfunktion und $w(y_i) \leq tw(y_i)$
6. Bedingung 2 an die Gewichtsfunktion. Die Knoten aus dem Hilfsbaum mit Wurzel w_2 repräsentieren einen Pfad in P. ${w_2}^*$ muss ein Nachfolger von y_i* sein.

Nun geht es um die Gesamtkosten *cost(switch)*(x_i) der *switch* Operation. Dazu werden die Kosten der max. drei *splay* Operationen mit $\Delta\Phi$ addiert. $tw(t_i)$ kann bei jeder *splay* Operation eingesetzt werden, da sich das Gesamtgewicht an der Wurzel, durch *splay* nicht verändert.

$$
\begin{aligned}
& O(\log_2(tw(t_i)/tw(y_i))) + O(\log_2(tw(t_i)/tw(y_l))) + O(\log_2(tw(t_i)/tw(y_r))) + O(\log_2(f))\\
\leq & O(\log_2(tw(t_i)/w(y_i))) + O(\log_2(tw(t_i)/w(y_l))) + O(\log_2(tw(t_i)/w(y_r))) + O(\log_2(f))\\
\leq & O(\log_2(tw(t_i)/w(y_i))) + O(\log_2(tw(t_i)/w(y_i))) + O(\log_2(tw(t_i)/w(y_i))) + O(\log_2(f))\\
= & O(\log_2(tw(t_i)/w(y_i))) + O(\log_2(f))\\
\leq & O(\log_2(tw(t_i)/(tw(t_{i+1})/f))) + O(\log_2(f)) \ \ (\textit{zweite Bedingung von } w())\\
= & O(\log_2(tw(t_i)/tw(t_{i+1}))) + O(\log_2(f))
\end{aligned}
$$

Für die zweite Ungleichung wird die erste Bedingung an die Gewichtsfunktion benötigt. □

Die Kosten der k *switch* Operationen müssen mit den Kosten der abschließenden *splay* Operation addiert werden, um auf Gesamtkosten $cost(access)$ zu erhalten.

$$\begin{aligned}
cost\left(access\left(x_i\right)\right) &= \sum_{i=1}^{k} O\left(cost\left(switch\left(y_i\right)\right)\right) + cost\left(splay\left(x_i\right)\right) \\
&= \sum_{i=1}^{k} O\left(\log_2\left(tw\left(t_i\right)/tw\left(t_{i+1}\right)\right)\right) + \sum_{i=1}^{k} O\left(\log_2\left(f\right)\right) + O\left(\log_2\left(tw\left(t_1\right)/tw\left(v\right)\right)\right) \\
&= O\left(\log_2\left(tw\left(t_1\right)/tw\left(t_{i+1}\right)\right)\right) + O\left(k\log_2\left(f\right)\right) + O\left(\log_2\left(fw\left(t\right)/w\left(v\right)\right)\right) \\
&= O\left(\log_2\left(w\left(t\right)/w\left(v\right)\right)\right) + O\left(\left(k+1\right)\log_2\left(f\right)\right)
\end{aligned}$$

Jetzt wird die Verbindung zur Interleave Bound hergestellt.Sei U die Menge der in P enthaltenen Knoten. Sei $X_1 = x_1, x_2, .., x_{i-1}$ und $X_2 = x_1, x_2, .., x_i$. k entspricht $IB\left(X_2\right) - IB\left(X_1\right)$, vergleiche Kapitel **??**.

$$\begin{aligned}
cost\left(access\left(X\right)\right) &= \sum_{i=1}^{m} access\left(x_i\right) \\
&= \sum_{i=1}^{m} O\left(\log_2\left(w\left(t\right)/w\left(node\left(x_i\right)\right)\right)\right) + \sum_{i=2}^{m} O\left(\left(1 + IB\left(X_2\right) - IB\left(X_1\right)\right)\log_2\left(f\right)\right) \\
&= \sum_{i=1}^{m} O\left(\log_2\left(w\left(t\right)/w\left(node\left(x_i\right)\right)\right)\right) + O\left(\left(IB\left(X\right) + m\right)\log_2\left(f\right)\right) \\
&= O\left(\sum_{i=1}^{m} \log_2\left(\frac{w\left(v\right)}{w\left(node\left(x_i\right)\right)}\right) + \left(\log_2 f\right)\left(IB\left(X\right) + m\right)\right)
\end{aligned}$$

Satz 1.1. *Sei T ein Multisplay Baum. Sei v ein Knoten im Hilfsbaum H von T. H enthalte n Knoten Sei u ein Vorfahre von v in H. Es wird eine (modifizierte) splay Operation verwendet, so dass $v'.depth = u.depth$ gilt. Die amortisierte Kosten dafür von splay(k) ist maximal $3\left(r\left(u\right) - r\left(v\right)\right) + 1 = O\left(\log\left(tw\left(u\right)\right)/tw\left(v\right)\right) = O\left(\log\left(n\right)\right)$*

Beweis. Folgt direkt aus **??**. Dass u die Wurzel ist wird dort nicht verwendet. □

Literatur

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