Bachelorarbeit

Andreas Windorfer

24. Juni 2020

Inhaltsverzeichnis

1 Tango Baum

Der Tango Baum ist ein aus BSTs, den **Hilfsbäumen**, bestehender BST. Auf die Anforderungen an die Hilfsbäume wird in Abschnitt 1.2 eingegangen und mit dem Rot-Schwarz-Baum wird eine mögliche Variante noch detailliert vorgestellt. Der Tango Baum wurde in [1],von Demaine, Harmon, Iacono und Patrascu beschrieben, inklusive eines Beweises über seine *lg lg n*-competitiveness. Ebenfalls in [1] enthalten ist eine als **Interleave Lower Bound** bezeichnete Variation der ersten unteren Schranke von Wilber. Da diese für das Verständnis des Tango Baumes wesentlich ist, wird mit ihr gestartet, bevor es zur Beschreibung der Struktur selbst kommt.

1.1 Interleave Lower Bound

Sei X eine Zugriffsfolge und sei $K = \{k \in \mathbb{N} | k \textit{ ist in } X \textit{ enthalten}\}$. Auch hier wird ein lower bound tree verwendet, dieser ist jedoch etwas anders definiert als in Abschnitt **??**. Hier ist der lower bound tree Y zu einer Zugriffsfolge X, der komplette BST mit Schlüsselmenge K. Anders als in Abschnitt **??** gibt es hier somit zu jeder Zugriffssequenz nur genau einen lower bound tree. Abbildung 1 zeigt den lower bound tree zur Zugriffsfolge $1, 2, .., 15$. Zu jedem Knoten v in Y werden zwei Mengen definiert. Die **linke Region** von v enthält den Schlüssel von v, sowie die im linken Teilbaum von v enthaltenen Schlüssel. Die **rechte Region** von v enthält die im rechten Teilbaum von v enthaltenen Schlüssel. Sei l der kleinste Schlüssel im Teilbaum mit Wurzel v und r der größte. Sei $X = x_1, x_2, ., x_m$ die Zugriffssequenz und $X_l^r = x_{1'}, x_{'2}, .., x_{jm'}$ wie in Abschnitt **??** definiert. $i \in \{2, 3, .., m'\}$ ist ein „**Interleave** durch v“ wenn $x_{(i-1)}$ in der linken Region von v liegt und x_i in der rechten Region von v, oder umgekehrt. In Y sind Knoten enthalten, bei denen die rechte Region leer ist. Durch diese kann es keinen Interleave geben. Sei U die Menge der Knoten von Y, mit einer nicht leeren rechten Region. Sei *inScore(v)* die Funktion die zu dem Knoten $v \in U$ die Anzahl der Interleaves durch v zurückgibt. Die Funktion $IB(X)$ ist definiert durch:

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

Sei T_0 der BST mit Schlüsselmenge K auf X ausgeführt wird. Für $i \in \{1, 2, .., m\}$ sei T_i der BST, der entsteht nachdem $access(x_i)$ auf T_{i-1} ausgeführt wurde. Zu $u \in U$ und $j \in \{0, 1, .., m\}$ gibt es einen **transition point** v in T_j. v ist ein Knoten mit folgenden Eigenschaften:

1. Der Pfad von der Wurzel von T_j zu v enthält einen Knoten dessen Schlüssel in der linken Region von u enthalten ist.

2. Der Pfad von der Wurzel von T_j zu v enthält einen Knoten dessen Schlüssel in der rechten Region von u enthalten ist.

3. In T_i ist kein Knoten mit Eigenschaft 1 und 2 enthalten, der eine kleinere Tiefe als v hat.

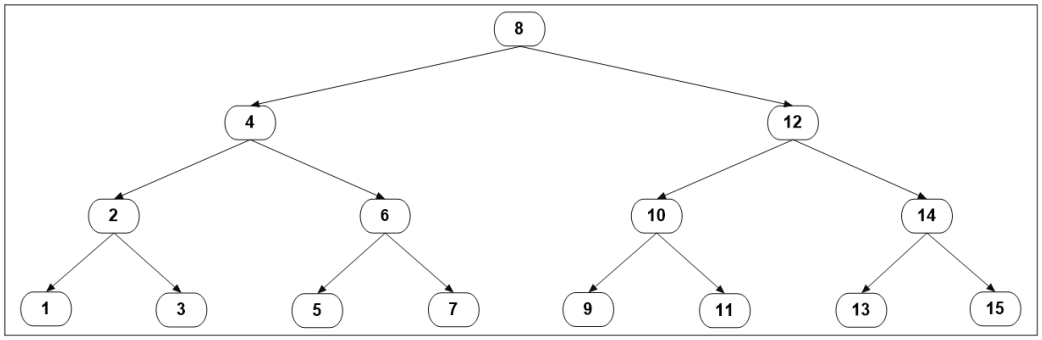

Abbildung 1: Der lower bound tree zur Zugriffsfolge $1, 2, .., 15$

Im Beweis dieses Abschnittes wird gezeigt das $OPT(X) \geq \frac{IB(X)}{2} - n$ gilt, wobei n die Anzahl der Knoten im lower bound tree ist. Dafür werden jedoch noch drei Lemmas zu den Eigenschaften von Y benötigt.

Lemma 1.1. *Sei $X = x_1, x_2, .., x_m$ eine Zugriffssequenz und Y ein zu X erstellter lower bound tree mit Schlüselmenge K. Sei T_0 der BST mit Schlüsselmenge K auf dem X ausgeführt wird. Für $i \in \{1, 2, .., m\}$ sei T_i der BST der durch Ausführen von access(x_i) auf $T_{(i-1)}$ entsteht. Sei U die Menge der Knoten von Y. Dann gibt es zu jedem Knoten $u \in U$ und $j \in \{0, 1, .., m\}$ genau einen transition point in T_j.*

Beweis. Sei l der kleinste Schlüssel in der linken Region von u und r der Größte. Im Teilbaum mit Wurzel u sind genau die Schlüssel $K_l^r = \{k \in K | k \in [l, r]\}$ enthalten. Sei v_l der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der linken Region von u in T_j, mit der größten Tiefe. Sei v_r der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der rechten Region von u in T_j, mit der größten Tiefe. $key(l)$ bzw. $key(r)$ muss selbst in der linken bzw. rechten Region von u enthalten sein, vergleiche **??**. Sei w der gemeinsame Vorfahre aller Schlüssel aus der linken und der rechten Region von u in T_l^r mit der größten Tiefe. Es muss $key(w) \in [l, r]$ gelten. Somit muss $key(w)$ entweder in der linken oder rechten Region von u enthalten sein. Da

w der Knoten mit der größten Tiefe sein muss, für den $key\,(w) \in [l, r]$ gilt, muss entweder $w = v_l$ oder $w = v_r$ gelten, je nachdem wessen Tiefe kleiner ist. Für den Fall $w = v_l$ ist v_r der transition point in T_j zu u und für den Fall $w = v_r$ ist es v_l. Es wird der Fall $w = v_l$ betrachtet, der andere kann direkt daraus abgeleitet werden. Im Pfad $P_u = v_0, v_1, .., v_r$ von der Wurzel v_0 zu v_r ist v_l enthalten und da v_r ein gemeinsamer Vorfahre der Schlüssel aus der rechten Region von u ist muss v_r der einzige Knoten mit einem Schlüssel aus der rechten Region von u in P_u sein. Jeder Pfad P in T_j von der Wurzel zu einem Knoten mit einem Schlüssel aus der rechten Region von u muss mit $v_0, v_1, .., v_r$ beginnen, somit kann es keinen weiteren transition point für u in T_j geben.

□

Der Knoten auf den der Zeiger p zum ausführen von *access* gerade zeigt wird als **berührter** Knoten bezeichnet. Im zweiten Lemma geht es darum, dass sich der transition point v eines Knoten nicht verändern kann, solange v nicht wenigstens einmal der berührte Knoten war. In den zwei verbleibenden Lemmas und dem Satz seien T_j, X, Y, U und u wie in Lemma 1.1 definiert.

Lemma 1.1. *Sei v der transition point zu u in T_j. Sei $l \in N$, mit $j < l \leq m$. Gilt für alle x_i, mit $i \in [j, l]$, während der Ausführung von access(x_i), v war nicht wenigstens einmal der berührte Knoten, dann ist v während der gesamten Ausführungszeit von $access\,(x_j)\,, access\,(x_{j+1})\,, .., access\,(x_l)$ der transition point zu u in T_l.*

Beweis. Sei v_l der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der linken Region von u in T_j, mit der größten Tiefe. Sei v_r der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der rechten Region von u in T_j, mit der größten Tiefe. Hier wird wieder ohne Verlust der Allgemeinheit der Fall $v = v_r$ betrachtet. Da v_r nicht berührt wird, wird auch kein Knoten in der rechten Region von u berührt. v_r ist somit während der gesamten Ausführungszeit von $access\,(x_j)\,, access\,(x_{j+1})\,, .., access\,(x_l)$ der gemeinsame Vorfahre der Schlüssel aus der rechten Region von u mit der größten Tiefe. Knoten mit Schlüssel in der linken Region von u könnten berührt werden. Zu einem Ausführungszeitpunkt t kann deshalb ein Knoten $v_{li} \neq v_l$ mit einem Schlüssel aus der linken Region von u der gemeinsame Vorfahre der Knoten mit diesen Schlüsseln mit der größten Tiefe sein. Da v_r nicht berührt wird kann zu keinem Zeitpunkt v_l im Teilbaum mit Wurzel v_r enthalten sein. Somit kann auch v_{lt} nicht in diesem Teilbaum enthalten sein. Somit muss die Tiefe von v_{lt} kleiner sein, als die von v_r und v_r bleibt der transition point von u. □

Im dritten Lemma wird gezeigt dass ein Knoten v in T_j nur der transition point zu einem Knoten aus U sein kann.

Lemma 1.1. *Sei $u_1, u_2 \in U$, mit $u_1 \neq u_2$. Sei v ein Knoten in T_j. v kann nicht sowohl der transition point von u_1, als auch der von u_2 sein.*

Beweis. Sei v_l bzw. v_r der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der linken bzw. rechten Region von u_1 in T_j, mit der größten Tiefe. Sei w_l bzw. w_r der gemeinsame Vorfahre aller Knoten mit einem Schlüssel aus der linken bzw. rechten Region von u_2 in T_j, mit der größten Tiefe. Ist weder u_1 ein Vorfahre von u_2 noch u_2 einer von u_1, dann muss auch $w_l \neq v_l \wedge w_l \neq v_r$ sowie $w_r \neq v_l \wedge w_r \neq v_r$ gelten, die Teilbäume mit Wurzel u_1 und u_2 dann über disjunkte Schlüsselmengen verfügen. Somit müssen die transition points von u_1 und u_2 unterschiedlich sein. Sei, ohne Verlust der Allgemeinheit, u_1 ein Vorfahre von u_2. werden drei Fälle unterschieden:

1. Ist $mathitkey\,(v_1)$ ist nicht im Teilbaum mit Wurzel u_2 enthalten, so kann v_1 nicht der transition point von u_2 sein.

2. $key\,(v_1)$ ist im Teilbaum mit Wurzel u_2 enthalten und $key\,(v_1)$ ist in der linken Region von u_1 enthalten:
   Da u_1 Vorfahre von u_2 ist, müssen alle Schlüssel im Teilbaum mit Wurzel u_2 in der linken Region von u_1 enthalten sein. Da der Schlüssel von v_1 in der linken Region von u_1 liegt, muss v_r ein Vorfahre von v_l in T_j sein.$key\,(v_1)$ muss somit der Schlüssel von w_l bzw. w_r sein, je nachdem wessen Tiefe kleiner ist. Denn andererseits könnte man einen Pfad von der Wurzel von T_j zu v_1 angeben der zwei Knoten aus der linken Region von u_1 enthält, dass ist jedoch ein Widerspruch dazu, dass $key\,(v_1)$ in der linken Region von u_1 enthalten ist und v_1 zudem der transition point für u_1 ist.
   v_2 ist entweder der Knoten w_l oder w_r je nachdem wessen Tiefe größer ist, somit gilt $v_1 \neq v_2$.

3. $key\,(v_1)$ ist im Teilbaum mit Wurzel u_2 enthalten und $key\,(v_1)$ ist in der rechten Region von u_1 enthalten:
   Symmetrisch zu Fall 2.

□

Satz 1.1. *Für eine Zugriffssequenz $X = x_1, x_2, .., x_m$ und n die Anzahl der Knoten im zu X erstellten lower bound tree. Dann gilt*
$OPT\,(X) \geq IB\,(X)\,/2 - n$.

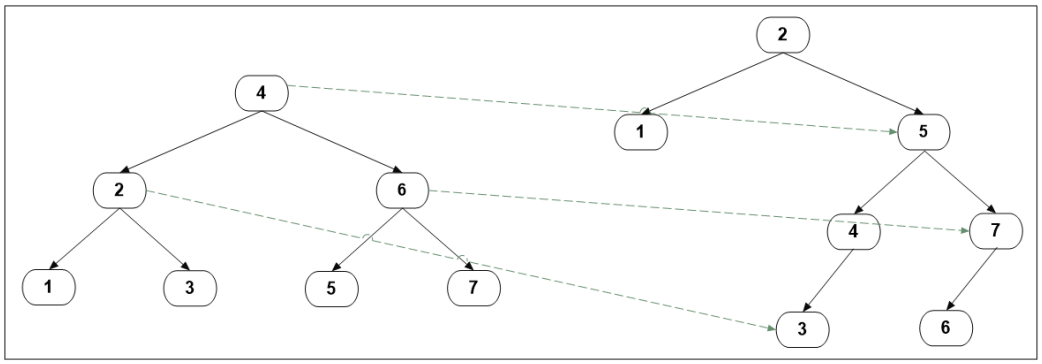

Abbildung 2: Transition point Zuordnung. Links ein lower bound tree, rechts ein möglicher T_j.

Beweis. Es wird die Mindestanzahl der Berührungen von transition points gezählt. Durch 1.1 kann die Anzahl der Berührungen für jedes $y \in P$ einzeln bestimmt werden, diese müssen dann lediglich noch aufaddiert werden. Sei l, r, v_r und v_l wie in Lemma1 zu y definiert, so dass entweder l oder r der transition point zu y sein muss, je nachdem welcher der beiden Knoten die größere Tiefe hat (**??**). Sei $X_l^{r\prime} = x_{i_1}, x_{i_2}, .., x_{i_p}$ die Folge die entsteht, wenn aus X_l^r alle x_k entfernt werden, für die gilt x_k ist in der gleichen Region von y wie x_{k-1}. Nun wird angenommen, dass die x_{j_i} mit i ist gerade in der rechten Region von y liegen, und die x_{j_i} mit i ist ungerade in der linken Region. Der andere Fall kann wieder direkt abgeleitet werden. Sei $q \in \mathbb{N}$ mit $1 \geq q \geq \lfloor p/2 \rfloor$. $access\left(x_{i_{2q-1}}\right)$ muss v_l berühren und $access\left(x_{i_{2q}}\right)$ muss v_r berühren. Sei ki_{2q-1} der Schlüssel des transition point von y zu Beginn von $access\left(x_{i_{2q-1}}\right)$ und ki_{2q} der Schlüssel des transition point von y zu Beginn von $access\left(x_{i_{2q}}\right)$. Gilt $ki_{2q-1} = ki_{2q}$ so muss der transition point von y in $access\left(x_{i_{2q}}\right)$ berührt worden sein. Gilt $ki_{2q-1} \neq ki_{2q}$ so muss der transition point von y, nach 1.1, in $access\left(x_{i_{2q-1}}\right)$ berührt worden sein. Aus der Konstruktion von $X_l^{r\prime}$ folgen daraus mindestens $\lfloor p/2 \rfloor \geq p/2 - 1$ Berührungen des transition point von y. Aufaddieren über alle Knoten ergibt eine Anzahl von Berührungen von transition points von zumindest $IB\left(X\right)/2 - |U| \geq IB\left(X\right)/2 - n$.

□

1.2 Aufbau des Tango Baum

Wie bereits erwähnt besteht ein Tango Baum T mit Schlüsselmenge K aus Hilfsbäumen. T bietet lediglich eine *access* Operation an. Ist T also erst einmal für K erzeugt, ist seine Schlüsselmenge unveränderlich. Sei P der lower bound tree aus Abschnitt 1.1 mit Schlüsselmenge K. P ist kein Hilfsbaum

und muss in Implementierungen auch nicht erstellt werden. Er dient aber dazu den Aufbau von T vor und nach einer *access* Operation zu veranschaulichen. Jeder innere Knoten p in P kann ein **preferred child** haben. Wurde während der Ausführungszeit von X noch keine *access* Operation mit einem im Teilbaum mit Wurzel p enthalten Schlüssel als Parameter ausgeführt, so hat p kein preferred child. Ansonsten sei $access(k)$ die zuletzt ausgeführte Operation mit einem Schlüssel der im Teilbaum mit Wurzel p enthalten ist. Liegt k in der linken Region von p, dann ist das linke Kind von p, das preferred child von p. Ist k in der rechten Region von p enthalten, dann ist das rechte Kind von p, das preferred child von p. Wir erweitern die Knoten von P mit einer weiteren Variable *prefChild* welche drei Werte annehmen kann. Sie enthält *none* wenn ihr Knoten kein preferred Child besitzt, *left* wenn das linke Kind das preferred child ist, ansonsten entsprechend *right*. Hier kann man bereits die Kopplung zur interleave lower bound erkennen. Ein Wechsel von *prefChild* von *left* zu *right*, oder umgekehrt, findet genau dann statt, wenn es zu einem interleave durch den Knoten der Variable kommt. Abbildung 3 stellt einen möglichen Zustand von P zwischen zwei *access* Operationen dar. Man erkennt zum Beispiel sofort, dass der Parameter der letzten *access* Operation 8, 4, 2 oder 1 gewesen sein muss, da man von der Wurzel aus über preferred child zu den Knoten mit diesen Schlüsseln gelangen kann. Die Schlüssel 10 und 9 können noch nie Parameter einer *access* Operation gewesen sein, ansonsten müsste der Knoten mit dem Schlüssel 10 ein preferred child haben

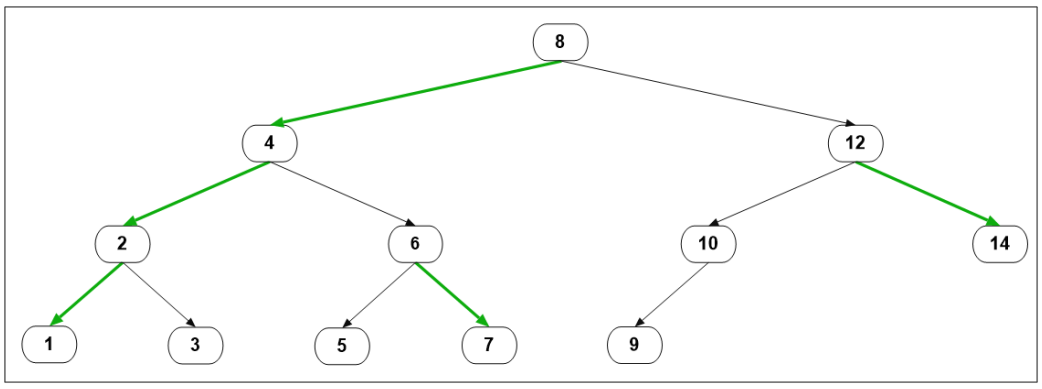

Abbildung 3: Die preferred childs werden durch die grünen Pfeile markiert.

1.3 Die *access* Operation beim Tango Baum

1.4 Laufzeitanalyse

Literatur

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