Splay trees

1 Einleitung

Splay-trees sind selbstbalancierende binäre Suchbäume mit der zusätzlichen Eigenschaft, kürzlich gesuchte Elemente sehr schnell zurückzugeben. In einer gewissen Weise vereinen sie die Eigenschaften von AVL-Bäumen (selbstbalancierend), und einer MFT-Liste (kürzlich angesprochene Elemente sind sehr schnell erreichbar). Die Nachteile sind aber, dass Splaytrees nur amortisiert logarithmische Laufzeit haben, einzelne Operationen können teurer sein. Zudem können sie in der Praxis etwas langsamer sein (wenn auch nicht asymptotisch), wenn die Zugriffe gleichverteilt sind.

Ein weiterer Vorteil ist der sehr einfache Aufbau von Splaytrees: Eine Implementation ist deutlich einfacher als etwa von AVL-Bäumen. Zudem sind keinerlei Zusatzinformationen nötig, die gespeichert werden müssen.

2 Operationen

2.1 Splaying

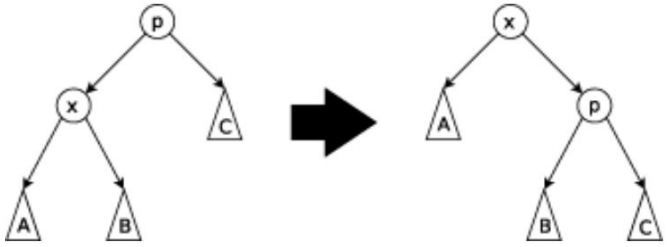
Immer wenn eine Element x angesprochen wird, kommt die splay-operation zum Einsatz um das Element an die Wurzel zu bewegen. Dabei kommt es auf drei Faktoren an

- Ist x linkes oder rechtes Kind von p(x)?
- Ist p(x) die Wurzel?, und wenn nicht:
- Ist p(x) linkes oder rechtes Kind von q(x), dem „Grossvater von x“.

Al erstes wird das Element x im Baum gesucht. Wird es nicht gefunden, so wird x auf den Vater der freien Position gesetzt, an welcher x zu liegen kommen würde in einem normalen Binärbaum. Dann wird das Element x zu Wurzel bewegt, wobei x immer um zwei „Ebenen“ im Baum nach oben steigt (ausser am Ende möglicherweise nur eine). Beim Bewegen des Elementes zur Wurzel gibt es drei Fälle:

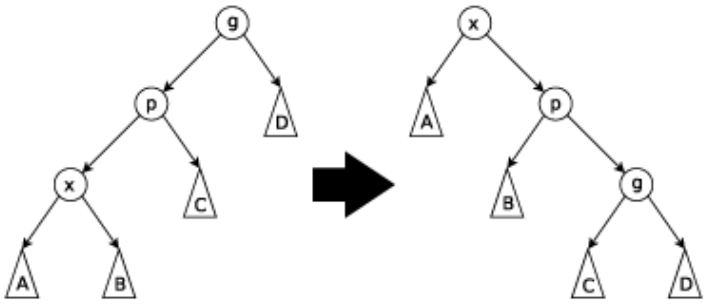
2.1.1 Zig

Ist p(x) bereits die Wurzel, muss nur x und p(x) durch eine einfach Rotation vertauscht werden.



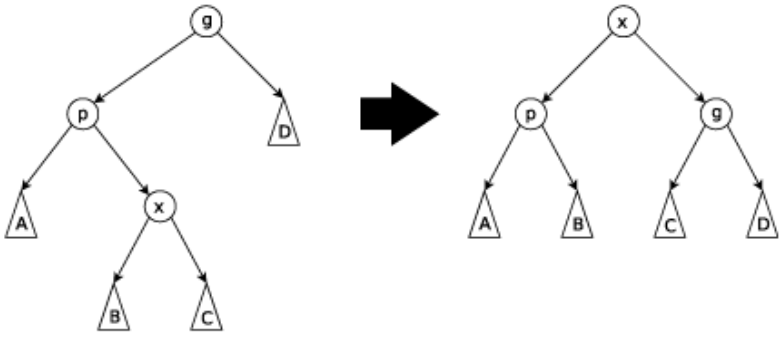
2.1.2 Zig-Zig

x und p(x) sind beide linke oder beides rechte Kinder, dann werden zwei Rotationen ausgeführt: Zuerst bei p(x), und dann eine Rotation bei x (wenn eine Rotation bei a heisst, dass a und p(a) vertauscht werden).



2.1.3 Zig-Zag

x ist linkes und p(x) rechtes Kind oder umgekehrt: Nun müssen ebenfalls zwei Rotationen durchgeführt werden, und zwar wie beim AVL-Baum: Zwei einfache Rotationen bei x.



2.2 Einfügen

Um ein Element k in den Baum einzufügen, wird nun zuerst splay(k) ausgeführt und damit der Vater y des einzufügenden Elements an die Wurzel gebracht. (Ist das Element bereits im Baum, so ist y == k und es ist nichts weiter zu tun). Da sich nun der symmetrische Vorgänger (oder Nachfolger) an der Wurzel befindet, kann k einfach an die Wurzel gesetzt werden. y und eines der beiden Kinder von y werden dann zu den Kindern von k und so entsteht wieder ein korrekter binärer Suchbaum.

2.3 Entfernen

Um ein Element k aus dem Baum zu entfernen, wird zuerst splay(k) ausgeführt. Findet sich dann an der Wurzel ein Element ungleich k, kam k im Baum nicht vor, und es ist nichts weiter zu tun. Andernfalls wird im linken Teilbaum von k (welches nun an der Wurzel steht), das grösste Element gesplayed. Damit erhält man einen Baum mit Wurzel k, dessen linker Teilbaum das grösste Element an der Wurzel hat. Deshalb gibt es dort kein rechtes Kind, und k kann einfach aus der Wurzel entfernt werden.