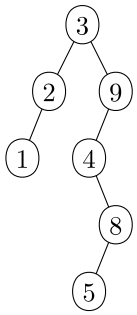
RWTH Aachen **SS 2018**
Lehrgebiet Theoretische Informatik Blatt 4
Rossmanith–Dreier–Kuinke 14.5.2018

Übung zur Vorlesung Algorithmen und Datenstrukturen

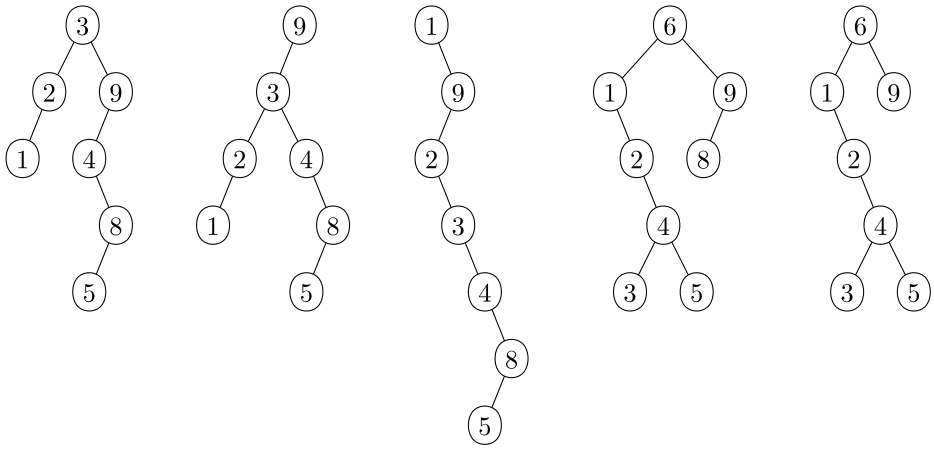
Aufgabe T8

Wir betrachten den rechten Splay-Baum. Was passiert, wenn wir in diesen Baum nach dem Schlüssel 10 suchen, dann nach 1 suchen, dann 6 einfügen und schließlich 8 löschen? Zeichnen Sie den resultierenden Baum nach jeder dieser Anweisungen.



Lösungsvorschlag:

Wir starten mit dem Baum links und suchen 10, suchen 1, fügen 6 ein und löschen schließlich die 8. Hier sind die Bäume, nach jedem dieser Schritte:



Aufgabe T9

Beim Suchen in einem Splay-Baum wird zum Schluss auf dem letzten besuchten Knoten eine Splay-Anweisung ausgeführt. Überlegen Sie sich ein Beispiel, in welchem bei einem anfangs leeren Splay-Baum n Operationen zu einem Zeitaufwand von $\Theta(n^2)$ führen, falls diese Splay-Anweisung nicht ausgeführt wird.

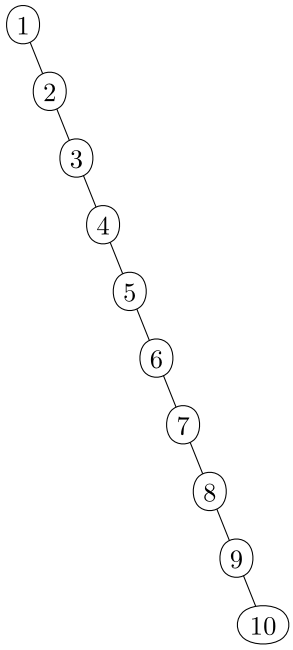
Lösungsvorschlag:

Wenn wir nach einem Schlüssel suchen, der gar nicht im Splay-Baum vorkommt, dann sorgt gerade diese Anweisung dafür, daß der bei der Suche zuletzt besuchte Knoten (und damit indirekt auch alle seine Vorfahren) durch eine Splay-Operation höher an die Wurzel gerückt werden. Das sind gerade die Knoten, welche bei der Suche besucht wurden. Ohne diese Anweisung, könnten wir also bei einer erfolglosen Suche sehr viel Zeit benötigen, weil sie zu einem sehr tief liegenden Blatt führt und könnten diese teuer Suche immer wieder wiederholen.

Als Beispiel fügen wir einfach die Schlüssel $1, \ldots, n$ in einen leeren Splay-Baum ein (was nur $O(n)$ Zeit benötigt), suchen dann aber n-mal nach dem Schlüssel 0. Jede dieser Suchen würde $\Theta(n)$ Schritte benötigen, was zu einer Gesamtzeit von $\Theta(n^2)$ für $2n$ Operationen führt.

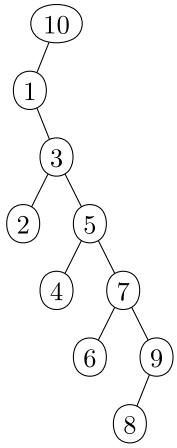
Aufgabe H5 (5 Punkte)

Wir betrachten den rechten Splay-Baum, der recht unbalanciert ist. Was passiert, wenn wir erst nach dem Schlüssel 10 suchen und dann nach dem Schlüssel 9? Ist der Baum jetzt besser balanciert? Was passiert, wenn bei der zig-zig und zag-zag Operation nicht der obere, sondern der untere Knoten zuerst rotiert wird?

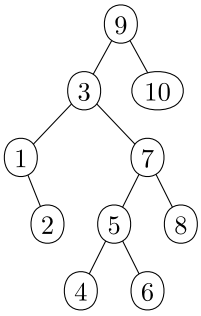


Lösungsvorschlag:

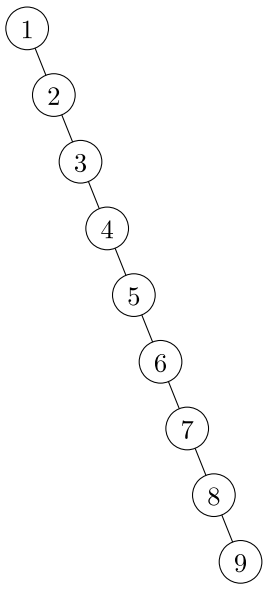
Nach der Suche nach 10 erhalten wir bereits diesen besseren Baum:



Suchen wir in diesem nach der 9, ergibt sich dieses recht balanzierte Ergebnis:



Falls wir beim zig-zig und zag-zag den unteren Knoten zuerst rotieren erhalten wir in diesem Beispiel einen sehr viel schlechter balancierten Baum in dem der gesuchte Knoten in der Wurzel ist und der linke Teilbaum genauso aussieht wie vorher, d.h. ein langer Pfad.



Aufgabe H6 (5 Punkte)

Entwerfen Sie einen effizienten Algorithmus, der einen Splay-Baum B und einen Schlüssel k bekommt und den Baum B in zwei Splay-Bäume B_1 und B_2 teilt, wobei B_1 alle Schlüssel enthält, die kleiner oder gleich k sind und B_2 die übrigen Schlüssel.

Lösungsvorschlag:

Wir können einfach auf B ein $splay(k)$ ausführen, was den Schlüssel k in die Wurzel befördert, oder, falls k in B nicht vorkommt den nächstkleineren oder nächstgrößeren Schlüssel.

Jetzt können wir einfach aus dem linken Unterbaum der Wurzel B_1 machen und aus dem rechten Unterbaum B_2. Die Wurzel hängen wir an B_1 oder B_2, je nachdem ob sie größer, kleiner oder gleich k ist.