AUD HAUSÜBUNG VON BLATT 5

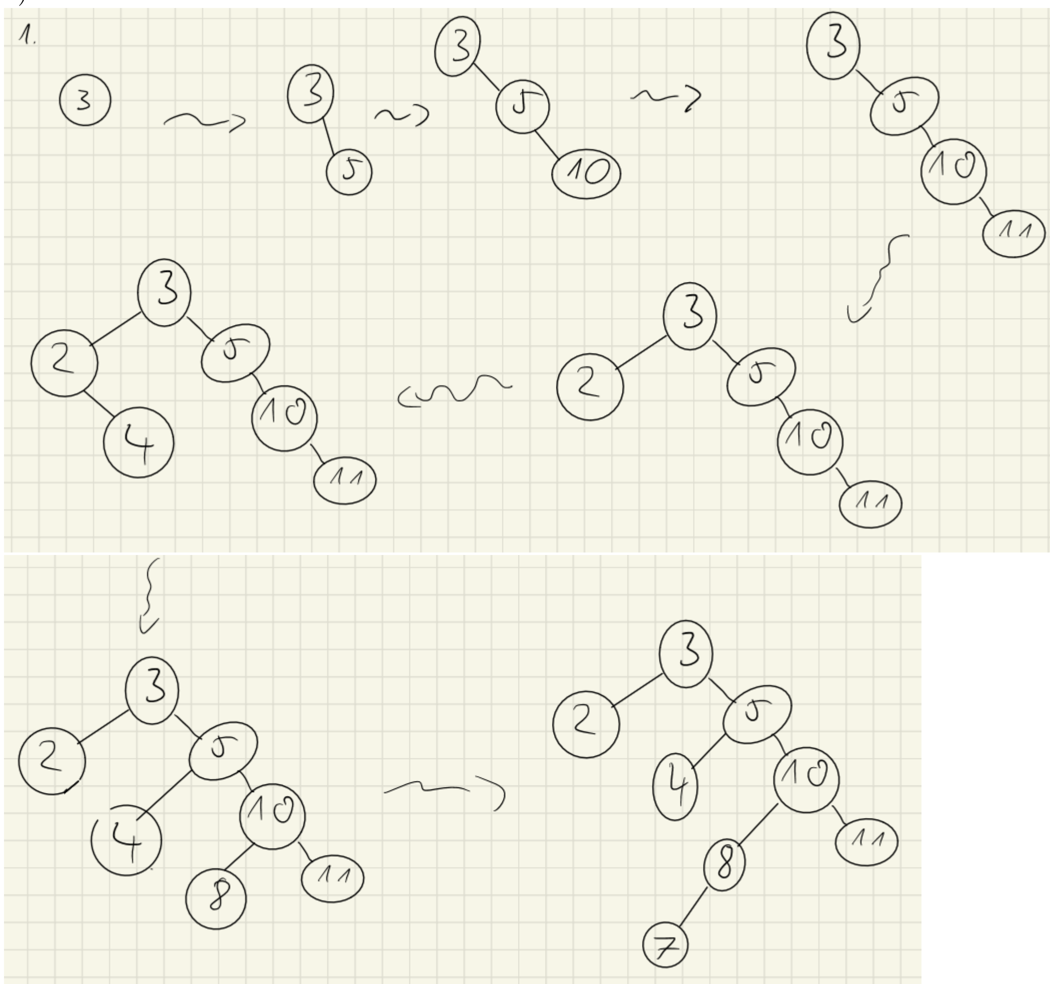
Hannes Albert

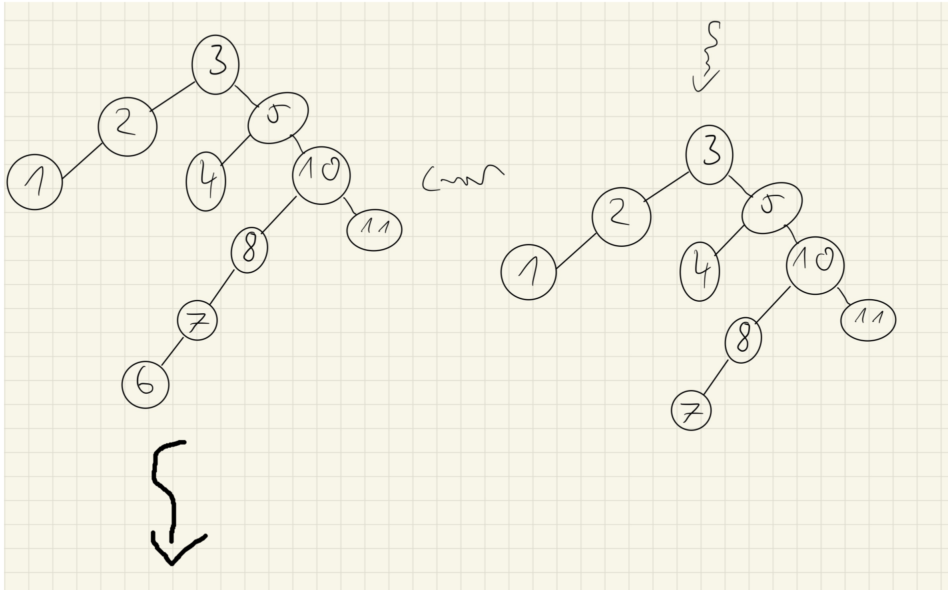
Gruppe: 36
Tutor: Julian Eulenburg

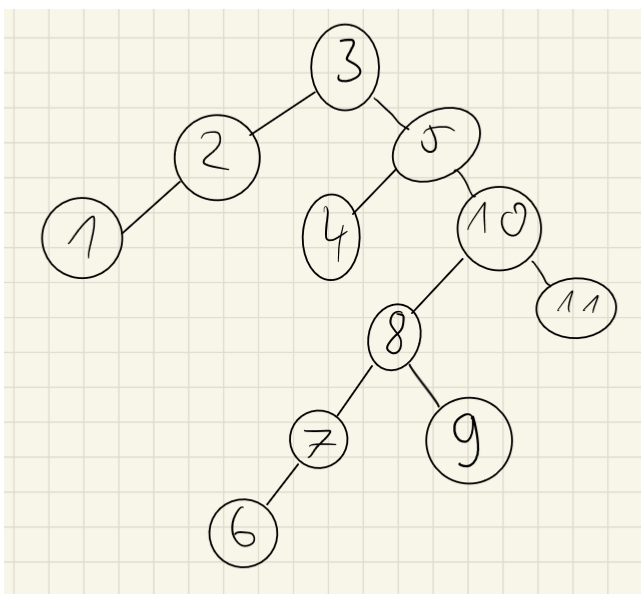
Mai 2023

1 H1

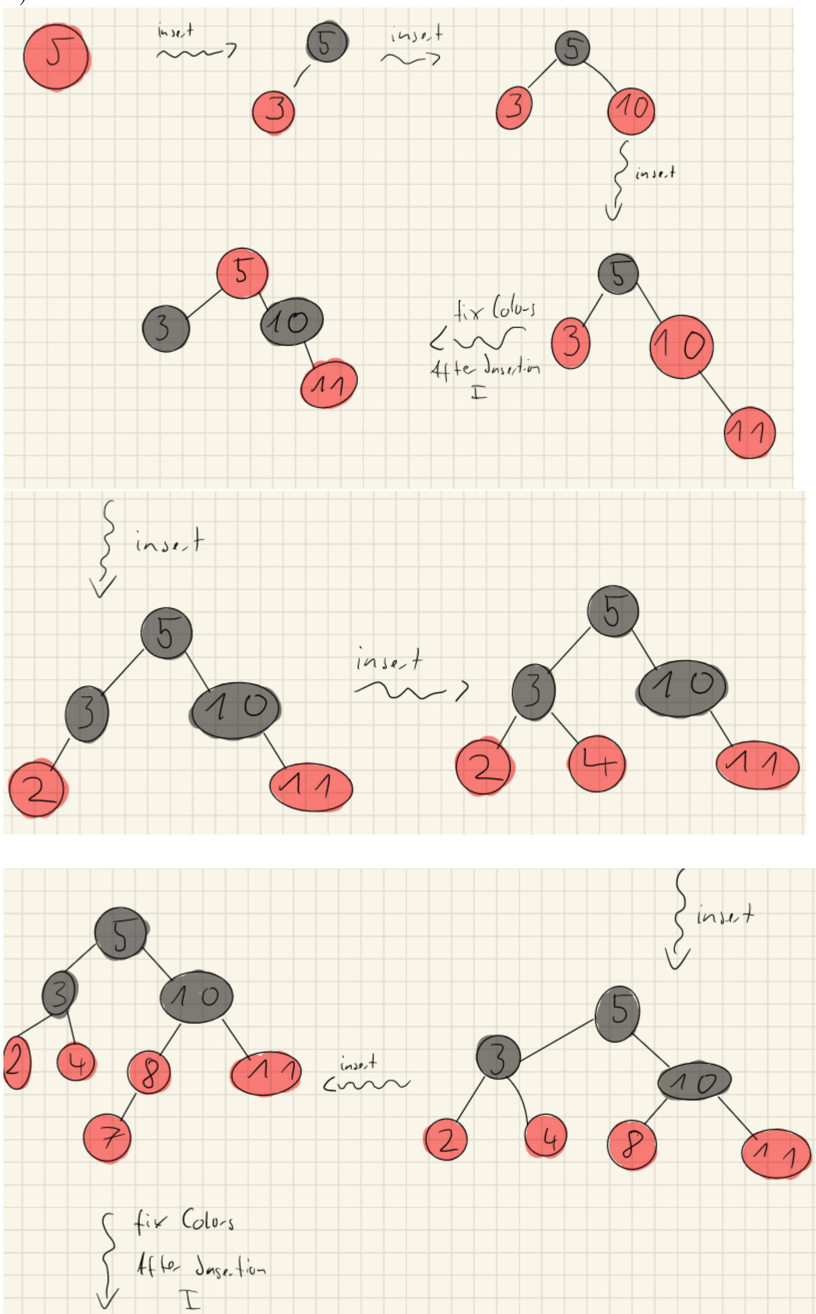
a)

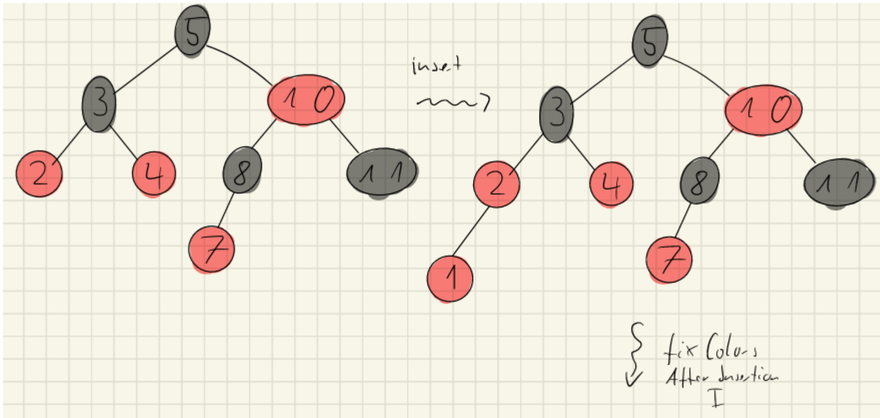


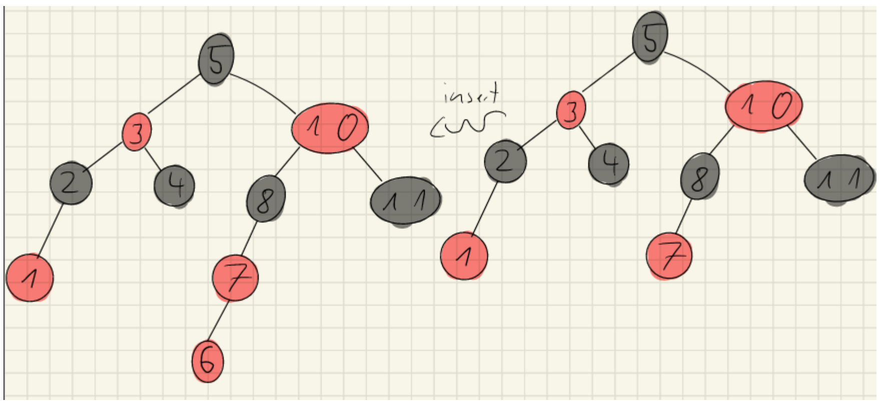

3
2
5
1
4
10
8
11
7
6
3
2
5
1
4
10
8
11
7

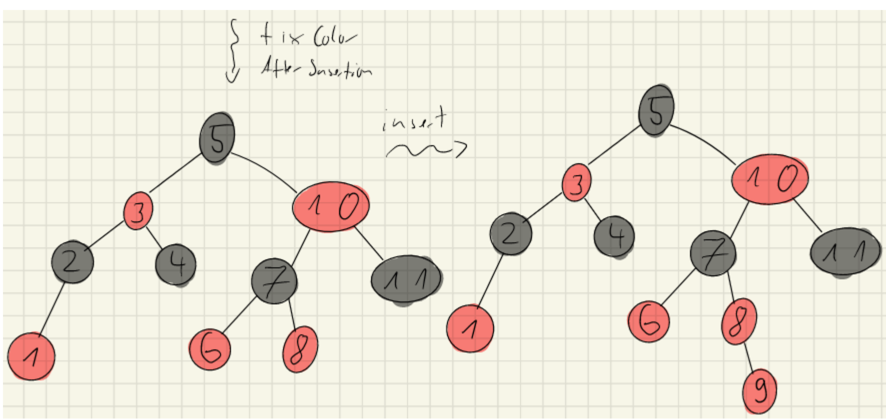

3
2
5
1
4
10
8
11
7
9
6

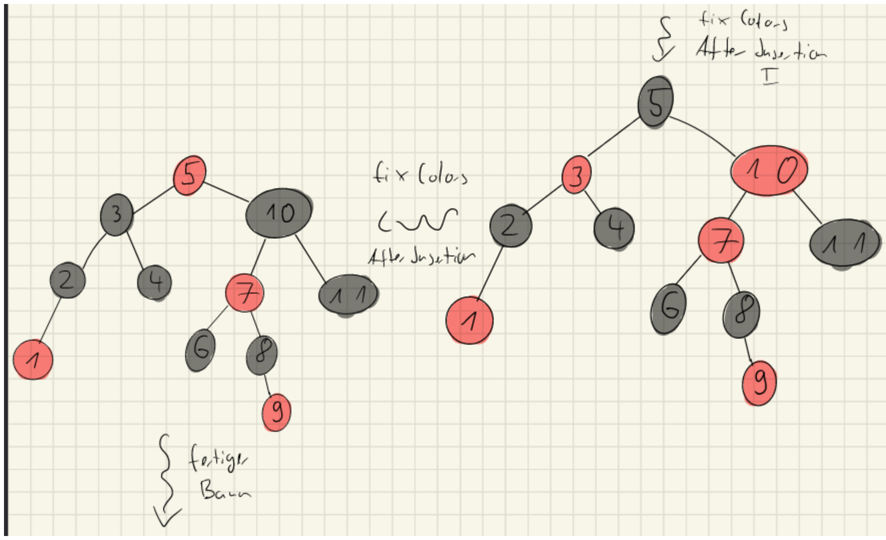
b)

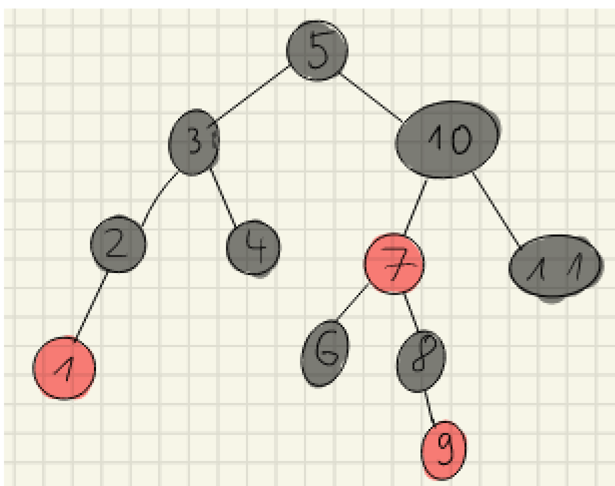



insert
fix Colors
After Insertion
I


insert


fix Color
After Insertion
insert





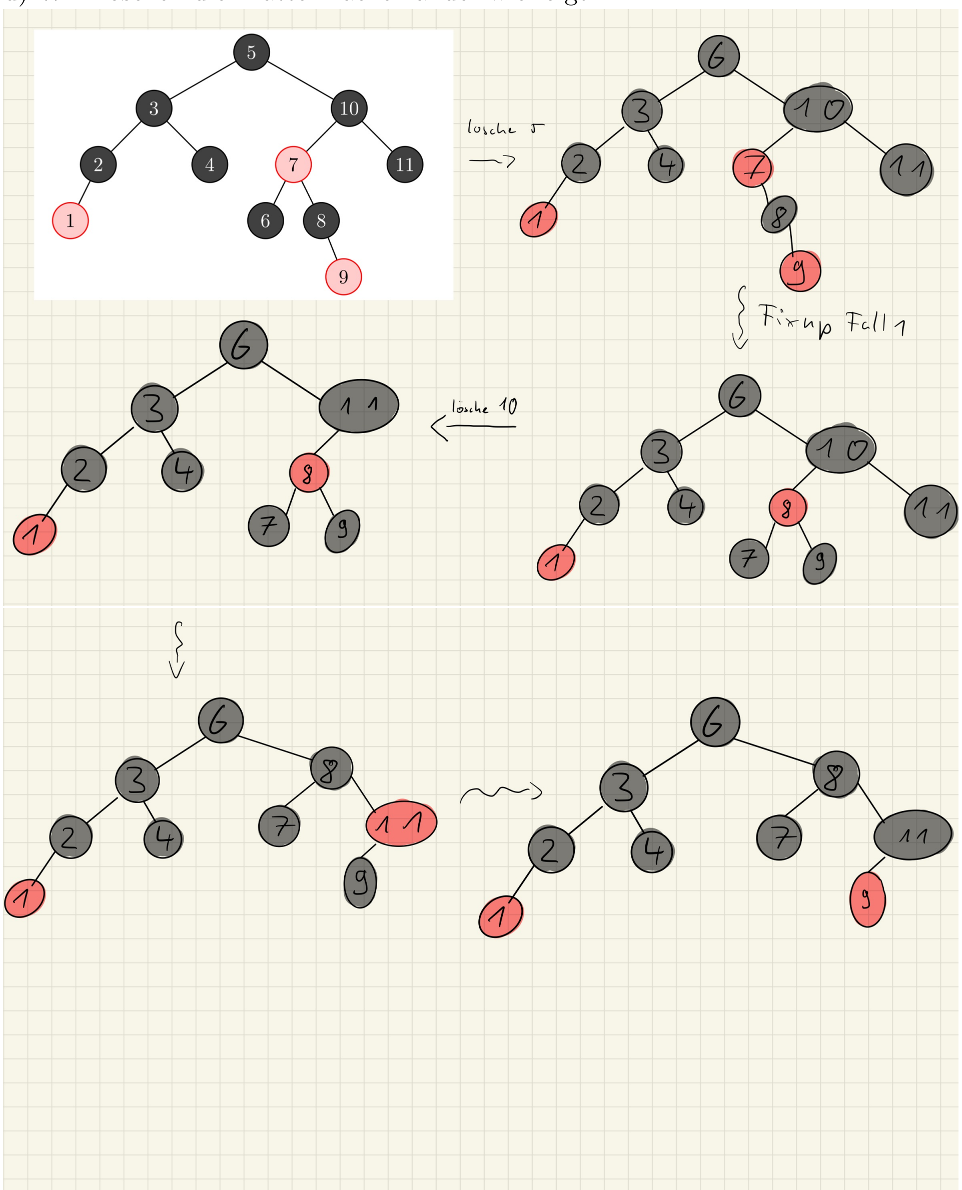
Mithilfe der Algorithmen von Foliensatz 03, S. 47 + 51 erhalten wir für den BST aus Aufgabe H1 a) folgenden Reihenfolgen für die verschiedenen Traversierungs-Möglichkeiten:

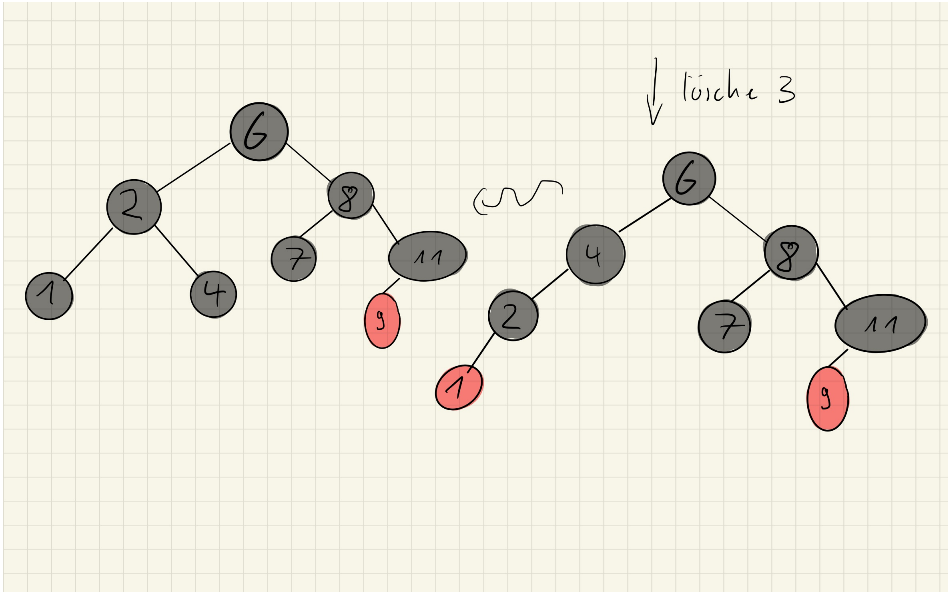
$$\begin{aligned} Preorder &: \ 3\ 2\ 1\ 5\ 4\ 10\ 8\ 7\ 6\ 9\ 11 \\ Inorder &: 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11 \\ Postorder &: 1\ 2\ 4\ 6\ 7\ 9\ 8\ 11\ 10\ 5\ 3 \end{aligned}$$

Für den Rot-Schwarz-Baum hingegen erhalten wir die folgenden Reihenfolgen:

$$Preorder: 5\ 3\ 2\ 1\ 4\ 10\ 7\ 6\ 8\ 9\ 11$$
$$Inorder: 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11$$
$$Postorder: 1\ 2\ 4\ 3\ 6\ 9\ 8\ 7\ 11\ 10\ 5$$

d) Wir Löschen die Blätter nacheinander wie folgt:





2 H2

a)
Induktionsanfang:
Sei n = 1. Da ein einzelner Knoten keine Kanten hat, da es ja schließlich keine Knoten zum verbinden gibt. D.h. Anzahl der Knoten entspicht 0. gilt die Formel:

$$n - 1 \overset{n=1}{=} 1 - 1 = 0$$

Induktionsvorraussetung:
Sei n $\in \mathbb{N}^*$, die Anzahl der Knoten eines Baumes. So entspricht die Anzahl der Kanten in einem Baum n - 1.
Induktionsschritt:
Für den Induktionsschritt setzen wir die Anzahl der Knoten auf n + 1 für n $\in \mathbb{N}^*$

$$n + 1 \overset{IV}{=} n - 1 + 1 = n$$

Die Fomel wurde hiermit gezeight, da (n + 1) - 1 = n und wir sind fertig. □

b)
Die Best-Case Laufzeit erhalten wir unter der Annahme, dass durch das Einfügen aller Knoten ein möglichst ausgeglichener Baum entsteht. Damit erhalten wir für die Laufzeit der Einfüge-operation n * O(log_2 n) = O(n log_2 n) (Foliensatz 03, S. 81) und für die Inorder-Traversierung erhalten wir eine Laufzeit von O(n) (Foliensatz 03, S. 48). Damit gilt:

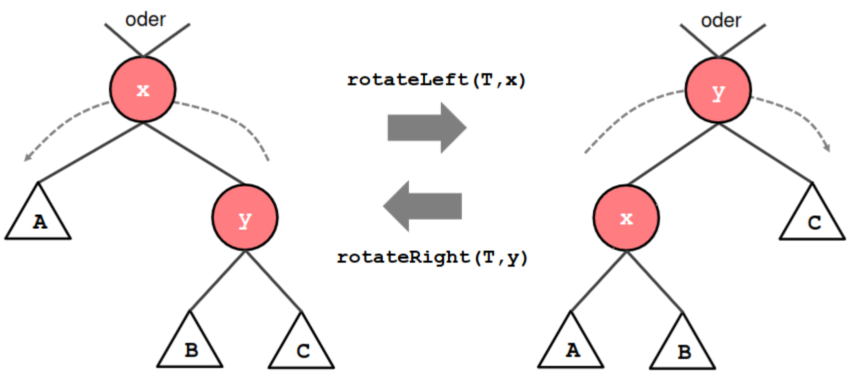
$$O(n * log_2 n) + O(n) = O(n * log_2 n + n) = O(n * log_2 n)$$

Somit erhalten wir eine Best-Case Laufzeit von O(n log_2 n). Im Falle der Worst-Case Laufzeit entsteht durch die Einfüge-Operationen im degenerierten Sinne eine lineare Liste was insgesamt zu einer Laufzeit von n * O(n) = O(n^2) führt. Für die Inorder-Traversierung erhalten wir weiterhin O(n). Damit bekommen wir:

$$O(n^2) + O(n) = O(n^2 + n) = O(n^2)$$

Also ergibt sich für die Worst-Case Laufzeit O(n^2). c)

Wir verwenden die gleiche Namensgebung für die beteiligten Elemente wie die Folien:



Um zu zeigen, dass die BST-Eigenschaften erhalten bleiben müssen wir zeigen, dass nach der

Rotation folgende Eigenschaften gelten:

1. $\forall k \in A : k.key <= x.key$
2. $\forall k \in B \cup C \cup \{y\} : k.key >= x.key$
3. $\forall k \in B : k.key <= y.key$
4. $\forall k \in C : k.key >= y.key$
5. x.key <= x.parent.key $\vee$ x.key >= x.parent.key , falls T.root != x

Nun arbeiten wir die einzelnen Punkte der Reihe nach ab:
1.
Da bereits vor der Rotation A der linke Teilbaum von x war müssen die geforderten Eigenschaften schon vor der Rotation gültig gewesen sein und da sich für den Teilbaum nichts geändert hat sind sie es weiterhin.

2.
Da B vor der Rotation der rechte Teilbaum von x war, gilt auch nach der Rotation, dass der Schllüssel aller Knoten in B größer gleich dem Wert des Knoten von x ist. Da x vor der Rotation Teil des linken Teilbaums von y war gilt: x.key <= y.key. Das heißt, dass

y.key >= x.key, weshalb die von uns geforderte Eigenschaft erfüllt ist. Da C vor der Rotation Teil des rechten Teilbaums von y war gilt nach der Rotation:

$$\forall k \in C : y.key <= k.key$$

Das bedeutet auch x.key <= y.key <= k.key für alle k $\in$ C. Somit ist auch die zweite Eigenschaft nach der Rotation erfüllt.

3.
Da B vor der Rotation auch schon Teil des linken Teilbaums von y war, gilt bereits für alle Elemente von B, dass ihr Schlüsselwert kleiner ist als der von y. Dies ist logischerweise auch nach der Rotation schon so.

4.
Da C vor der Rotation der rechte Teilbaum von y war und dies sich während der Rotation auch nicht geändert hat gilt dies natürlcih auch noch nach Rotation, weshalb dieser Punkt auch erfüllt ist.

5.
Wir treffen hier eine Fallunterscheidung:
1.Fall: x.parent.left = x
Da x vor der Rotation Teil des linken Teilbaums war, gilt auch weiterhin, dass x.key ¡= x.parent.key.
2.Fall: x.parent.right = x
Da x vor der Rotation Teil des rechten Teilbaums war, gilt auch weiterhin, dass x.key ¿= x.parent.key.
Somit ist auch die fünfte geforderte Eigenschaft erfüllt und wir sind fertig.

Damit eine Rechtsrotation funktioniert muss natürlich gelten, dass x und y ungelich nil sind. Bei A, B und C ist dies nicht so wichtig: wenn der Teilbaum nicht existiert gibt es keine Elemente deren Schlüssel größer, kleiner oder gleich x.key oder y.key sein können. Dementsprechend würde für A = nil 1. außer Acht gewerden lassen können. Für B = nil könnte 3. ignoriert werden und in 2. würde einfach B durch die Menge ohne Elemente ersetzt werden. Gleiches gilt für den Fall, dass C = nil wäre, wobei dann 4. ignoriert werden könnte.