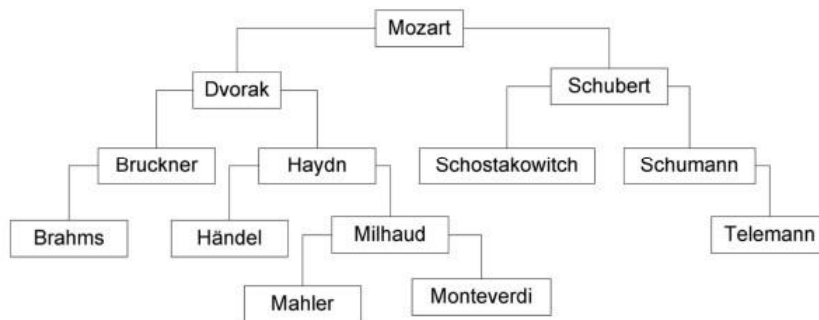


Algorithmen und Datenstrukturen

Ausgeglichene Bäume

Aufgabe 1: Manuelles Einfügen in AVL-Bäume [2 Punkt]

Gegeben ist der folgende AVL-Baum mit den Namen berühmter Komponisten:



- Fügen Sie die beiden Knoten „Beethoven“ und „Zelenka“ ein. Führen Sie alle nötigen Rebalancierungen durch, um die AVL-Eigenschaft zu erhalten. Geben Sie dabei auch die Rotationstypen (r, l, rl, lr) an.
- Fügen Sie weiter einen Knoten „Mendelssohn“ ein und führen Sie analog die nötigen Rebalancierungen durch.

Aufgabe 2: Implementieren Sie den AVL-Baum [2 Punkte]

Vervollständigen Sie den AVL Baum, der als Gerüst vorgegeben ist. Füllen Sie die Werte der Rangliste ein.

Aufgabe 3: Bestimmen der Höhe des Baumes [1 Punkte]

Entwickeln Sie eine Methode *height*, mittels der Sie die Höhe eines (Teil-) Baumes bestimmen können. Bestimmen Sie so die Höhe Baumes der Rangliste aus dem letzten Praktikum für den unausgeglichene SortedBinary Baum und den AVL Baum.

Erklären Sie den grossen Unterschied.

Hinweise:

- Die Methode ist bereits in SortedBinary Tree vorbereitet. *calcHeight*; Eine rekursive Methode, die jeweils die Höhe des linken und rechten Teilbaums bestimmt.
- Das *height*: Attribut im Knoten soll nicht verwendet werden

Aufgabe 4: Implementieren Sie Balanced Check [2 Punkte]

Entwickeln Sie eine Methode *balanced*, die überprüft, ob der Baum AVL ausgeglichen ist.

Hinweis

- Verwenden Sie dafür die Methode *height* von oben und verlassen Sie sich nicht auf die Höhenangabe in den Knoten.

Aufgabe 5: Implementation einer Remove Methode [3 Punkte]

Auch nach dem Löschen von Elementen kann der Baum nicht mehr balanciert sein. Das heisst auch hier muss überprüft werden, ob der Höhenunterschied 2 nicht überschreitet.

Hinweis

- Wie beim Einfügen muss beim Löschen der Baum wieder balanciert werden, d.h. es kann die entsprechende Methode nach der Mutation aufgerufen werden.