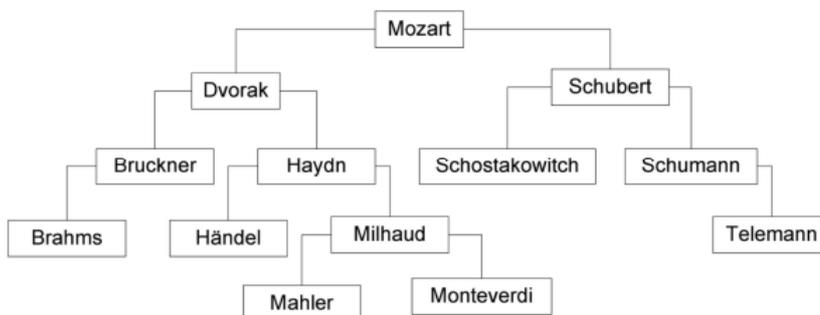


Algorithmen und Datenstrukturen

Ausgeglichene Bäume

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

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



a) 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.

b) Fügen Sie weiter einen Knoten „Mendelsohn“ 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 und bestimmen Sie ebenfalls die Höhe.

Abgabe

Praktikum: ADS6.2

Filename: AVLSearchTree.java

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

Vergleichen Sie so die Höhe Baumes der Rangliste aus dem letzten Praktikum für den unausgeglichene SortedBinary Baum und den AVL Baum. Können Sie den relativ grossen Unterschied erklären?

Aufgabe 4: Implementieren Sie Balanced Check [3 Punkte]

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

Hinweis

- Verwenden Sie dafür die Methode *calcHeight* i.e. verlassen Sie sich nicht auf die Höhenangabe in den Knoten, die ja falsch sein kann.

Abgabe

Praktikum: ADS6.4

Filename: AVLSearchTree.java

Aufgabe 5: Implementation der 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.

Abgabe

Praktikum: ADS6.5

Filename: AVLSearchTree.java