

---

## Grundlagen: Algorithmen und Datenstrukturen

---

*Abgabetermin: In der jeweiligen Tutorübung*

### Hausaufgabe 1

Implementieren Sie in der Klasse `UIbiHeap` einen Binomial-Heap. Hierzu muss auch ein BinomialBaum in der Klasse `binomialTree` implementiert werden.

Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `UIbiHeap` und `binomialTree`.

Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klassen `UIbiHeap` und `binomialTree` heißen und auf den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_bi` kompiliert werden können. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.

Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.

### Hausaufgabe 2

Implementieren Sie in der Klasse `IbinaryTree` einen binären Baum, der neben den Operationen `insert`, `remove`, `find` auch die Operationen `preOrder` und `postOrder` implementiert.

Die Funktion `preOrder` druckt ein Element (anfangs die Wurzel) und steigt dann rekursiv zuerst in den linken und dann in den rechten Teilbaum ab (In einer ersten Version des Blattes war links und rechts vertauscht). Die Funktion `postOrder` arbeitet umgekehrt. Sie führt zuerst die rekursiven Aufrufe aus und druckt dann das Element (anfangs die Wurzel).

Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die bereitgestellten Klassen aber nicht deren Interfaces.

Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klasse auf den Rechnern der Rayhalle (`rayhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_bt` kompiliert werden kann. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.

Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.

## Aufgabe 1

Geben Sie einen Algorithmus für eine Build-Funktion an, die  $n$  Elemente  $e_1, \dots, e_n$  als Eingabe erhält und diese in einem Binomial-Heap speichert und die in  $\mathcal{O}(n)$  Zeit läuft. Beweisen Sie die Laufzeit Ihres Algorithmus.

## Aufgabe 2

Fügen Sie die Schlüssel  $a, b, \dots, i$  in der Reihenfolge  $(a, i, e, g, h, f, c, b, d)$  in einen anfänglich leeren AVL-Baum ein. Entfernen Sie anschließend den Schlüssel  $i$ . Zeichnen Sie jeweils den resultierenden Baum (einschließlich notwendiger Rotationen).

## Aufgabe 3

Führen Sie auf einem anfangs leeren  $(2, 4)$ -Baum folgende Operationen aus und zeichnen Sie die Zwischenergebnisse: `insert(23)`, `insert(30)`, `insert(13)`, `insert(6)`, `insert(40)`, `insert(80)`, `insert(62)`, `insert(75)`, `insert(28)`, `insert(21)`, `insert(29)`, `remove(62)`, `remove(75)`, `remove(13)`.