

## Binäre Suchbäume

### Aufgabe 1:

- a) Abbildung 1 und Abbildung 2 zeigen Beispiele für binäre Suchbäume. Stellen Sie eine Vermutung auf, was man unter einem binären Suchbaum versteht. Welche Vorteile könnte die Verwendung eines binären Suchbaums haben?
- b) Formulieren Sie Bedingungen, die ein Binärbaum erfüllen muss, damit man ihn als binären Suchbaum bezeichnen kann.

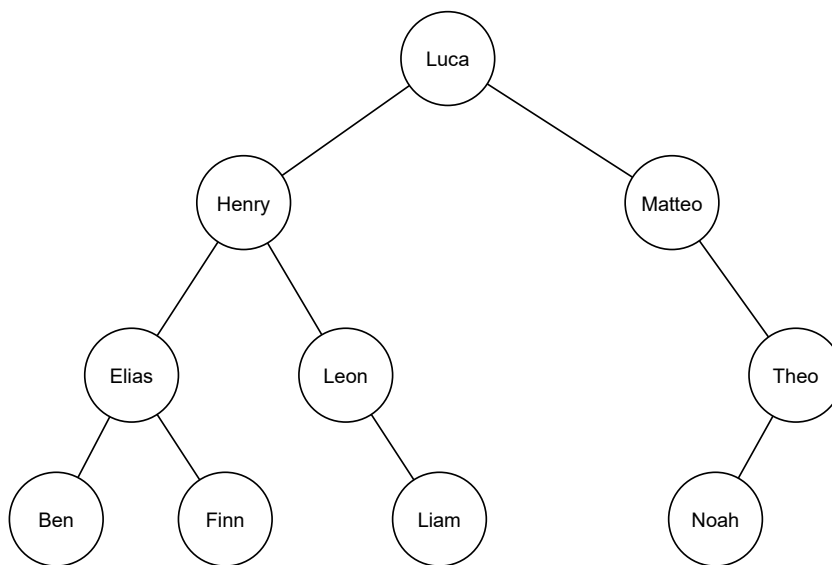


Abbildung 2: binärer Suchbaum mit den 10 beliebtesten Jungennamen eines Jahres

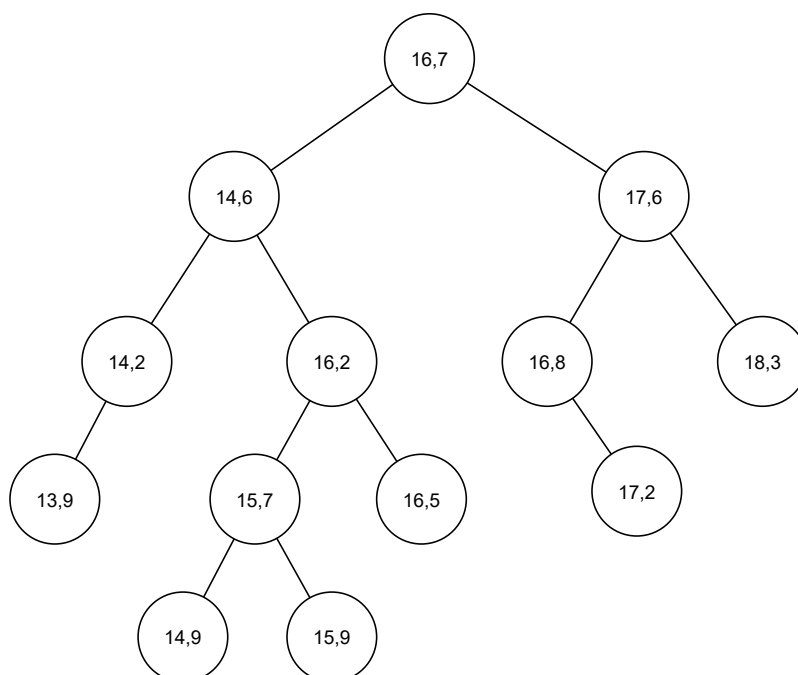


Abbildung 1: Binärer Suchbaum mit den Ergebnissen eines 100-m-Laufs eines Sportkurses

## Definition eines binären Suchbaums

Ein binärer Suchbaum muss für jeden Knoten bzw. jeden Teilbaum die folgenden Bedingungen<sup>1</sup> erfüllen:

- Alle Knoten im linken Teilbaum haben Werte, die kleiner sind als der Wert der Wurzel.
- Alle Knoten im rechten Teilbaum haben Werte, die größer sind als der Wert der Wurzel.

Enthalten die Knoten Objekte mit mehreren Attributen, so können die Bedingungen in der Regel nicht für alle Attribute erfüllt werden. Es muss daher festgelegt werden, für welches Attribut der binäre Baum sortiert sein soll.

### Aufgabe 2:

- Fügen Sie in den binären Suchbaum in Abbildung 3 nacheinander die Werte 8, 7 und 12 ein. Die Vorhandenen Eltern-Kind-Beziehungen sollen dabei nicht verändert werden.
- Beschreiben Sie allgemein das Vorgehen beim Einfügen von Werten in einen binären Suchbaum.
- Zeichnen Sie einen binären Suchbaum, der die Zahlen 1 bis 15 enthält und die Höhe vier hat. Gibt es mehr als eine Lösung? Begründen Sie Ihre Antwort.
- Geben Sie für den binären Suchbaum in Abbildung 3 jeweils die Reihenfolge der Knoten an, wenn der Suchbaum in *inorder*, *preorder*, *postorder* bzw. *levelorder* Reihenfolge durchlaufen wird. Die in a) eingefügten Werte müssen dabei nicht berücksichtigt werden.
- Zeichnen Sie für die 4 Ausgabereihenfolgen aus Aufgabe c) jeweils den resultierenden Baum, wenn die Zahlen in der jeweiligen Reihenfolge sukzessive durch Einfügeoperationen in einen leeren binären Suchbaum eingefügt werden.

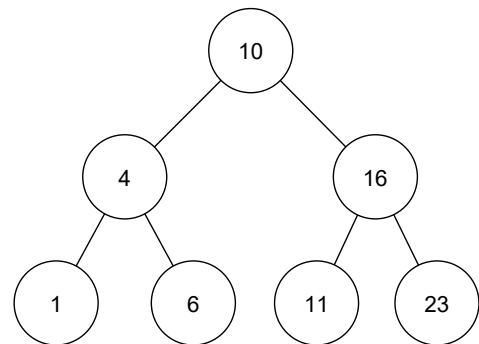


Abbildung 3: Binärer Suchbaum vom Typ Ganzzahl

### Aufgabe 3:

- Begründen Sie, dass der Aufwand für die Suche nach einem Knoten mit einem bestimmten Wert in einem binären Suchbaum abhängig von der Höhe des binären Suchbaums ist.
- Vergleichen Sie den Aufwand für die Suche in einem binären Suchbaum mit dem Aufwand für die Suche in einer sortierten Reihung. Schätzen Sie den Aufwand anhand der benötigten Vergleiche ab.

<sup>1</sup> Hier wird nur mit dieser Definition gearbeitet. Denkbar wäre aber auch eine spiegelverkehrte Sortierung, so dass die Werte im linken Teilbaum größer und die im rechten Teilbaum kleiner als die Wurzel sind. Dies ist zu berücksichtigen, wenn für einen gegebenen Suchbaum in einem anderen Kontext die Ordnung ermittelt werden soll.

**Aufgabe 4:** Abbildung 4 zeigt einen binären Suchbaum, dessen Knoten Objekte einer Klasse `Person` mit den Attributen `ID`, `Name` und `Alter` enthalten.

- a) Geben Sie an, nach welchem Attribut der binäre Suchbaum sortiert ist.  
b) Geben Sie die Werte des Attributs `Name` in der Reihenfolge aus, die entsteht, wenn der Baum in *inorder*-Reihenfolge durchlaufen wird.

Erzeugen Sie anschließend den binären Suchbaum, der entsteht, wenn die Personen in dieser Reihenfolge sukzessive in einen anfangs leeren Binärbaum eingefügt werden. Der Baum soll diesmal nach dem Attribut `Name` sortiert sein.

- c) Erläutern Sie, weshalb das Attribut `Alter` für die Sortierung der Knoten in einem binären Suchbaum ungeeignet ist.

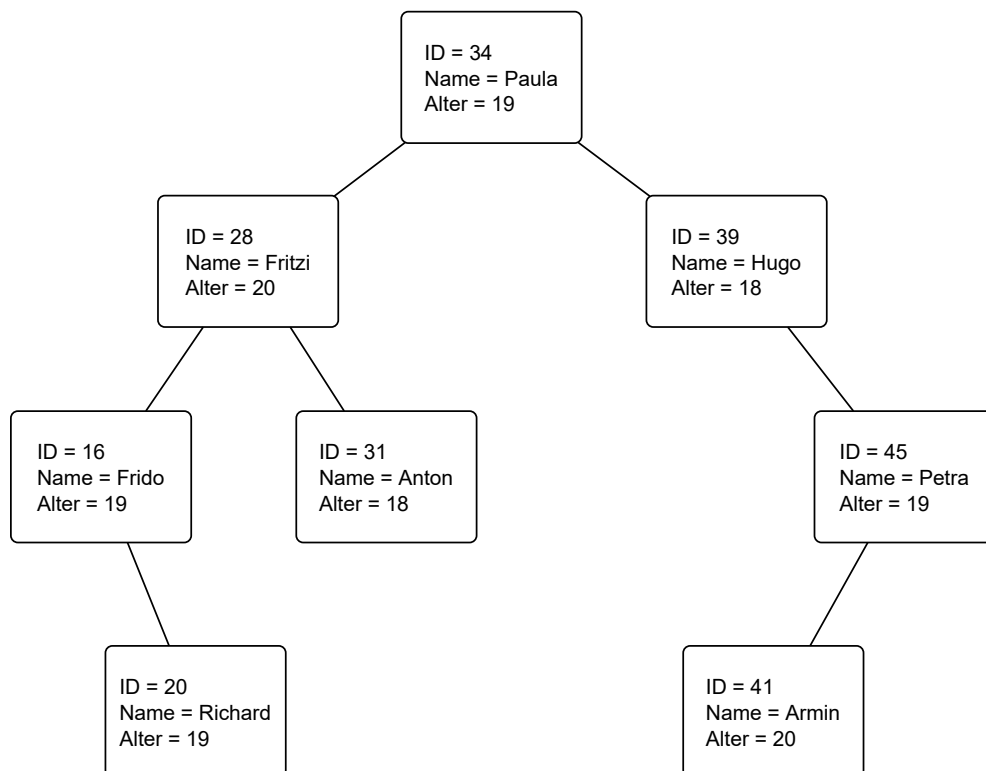


Abbildung 4: binärer Suchbaum mit Knoten vom Typ `Person`

**Aufgabe 5:** Im beiliegenden Programm `VorlageAufgabe5` wird mit folgenden Anweisungen ein binärer Suchbaum erzeugt, der in der globalen Variablen `suchbaum` gespeichert wird.

```
1 suchbaum.setItem("Luna");
2 suchbaum.setLeft(new BinTree("Hanna"));
3 suchbaum.setRight(new BinTree("Robert"));
4 suchbaum.getLeft().setLeft(new BinTree("Emil"));
5 suchbaum.getLeft().setRight(new BinTree("Igor"));
6 suchbaum.getRight().setLeft(new BinTree("Max"));
7 suchbaum.getRight().setRight(new BinTree("Siri"));
8 suchbaum.getLeft().getLeft().setRight(new BinTree("Frida"));
9 suchbaum.getRight().getLeft().setRight(new BinTree("Ole"));
```

Abbildung 5: Anweisungen zum Erzeugen eines binären Suchbaums

- Zeichnen Sie den binären Suchbaum, der durch die Anweisungen in Abbildung 5 entsteht.
- Das Programm enthält bereits eine Implementierung der Operation `preorder`. Implementieren Sie die Operationen `inorder`, `postorder` und `levelorder`, so dass die Inhaltswerte des global definierten Suchbaums `suchbaum` passend ausgegeben werden.
- Implementieren Sie eine Operation `suche`, die als Parameter eine Zeichenkette und einen Binärbaum erhält und rekursiv nach dem Teilbaum sucht, der als Inhaltswert der Wurzel die übergebene Zeichenkette enthält. Die Operation gibt den gefundenen Teilbaum zurück bzw. den Wert `null`, wenn kein entsprechender Teilbaum gefunden wird.  
Testen Sie Ihre Operation `suche`, indem Anwendende die Möglichkeit erhalten, einen Namen einzugeben, nach dem gesucht werden soll. Geben Sie den gefundenen Teilbaum in *inorder*-Reihenfolge aus.  
Untersuchen Sie, wie oft die Operation `suche` maximal rekursiv aufgerufen wird.
- Implementieren Sie eine rekursive Operation `einfügen`, die als Parameter den einzufügenden Wert, also Namen als Zeichenkette erhält sowie den Suchbaum, in den der Wert eingefügt werden soll. Testen Sie die Operation, indem Anwendende die Möglichkeit erhalten, einen Namen einzugeben, der eingefügt werden soll. Geben Sie den `suchbaum` nach dem Einfügen eines neuen Namens in *inorder*-Reihenfolge aus.

**Aufgabe 6:** In dem Programm aus Aufgabe 5 soll es auch möglich sein, einen Namen aus dem binären Suchbaum zu löschen. Dazu müssen die folgenden Fälle unterschieden werden:

- Der zu löschende Knoten ist ein Blatt.
  - Der zu löschende Knoten hat genau ein Kind.
  - Der zu löschende Knoten hat zwei Kinder.
- Entwickeln Sie für jeden der drei Fälle ein mögliches Vorgehen, welches den entsprechenden Knoten so löscht, dass der Baum weiterhin sortiert ist und alle anderen Knoten von der Wurzel aus erreicht werden können.
  - Begründen Sie, dass man den Nachfolger eines Knotens findet, wenn man im rechten Teilbaum, immer wieder den linken Teilbaum wählt, bis es keinen linken Teilbaum mehr gibt.
  - Skizzieren Sie für den Suchbaum in Abbildung 6, welcher Baum jeweils entsteht, wenn nacheinander die Knoten „Klaus“, „Bert“ und „Till“ gelöscht werden.
  - Implementieren Sie eine Operation `loeschen(wert: Zeichenkette, b: BinTree): BinTree`, die alle drei Fälle berücksichtigt. Implementieren Sie dazu eine Hilfsoperation `sucheKleinsten(b: BinTree): BinTree2`, die in dem Binärbaum `b` den Binärbaum mit dem kleinsten Inhaltswert in der Wurzel bestimmt, sowie eine Hilfsoperation `sucheVater(wert: Zeichenkette, b: BinTree): BinTree`, die zu einem Wert den Binärbaum mit dem Vaterknoten als Wurzel bestimmt.

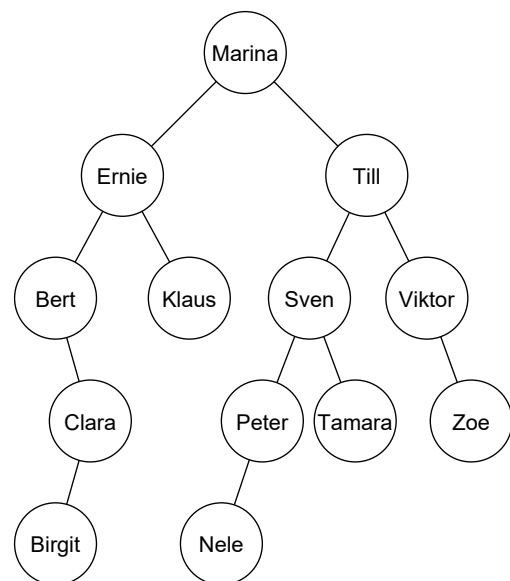


Abbildung 6: Binärer Suchbaum alphabetisch sortiert

<sup>2</sup> Abhängig vom gewählten Vorgehen für Fall 3 kann alternativ eine Operation `sucheGrößten` hilfreich sein.

**Aufgabe 7:** Das Programm *VorlageAufgabe7* enthält einen global definierten binären Suchbaum `tierBaum` vom Typ `Tier`. Die Klasse `Tier` enthält die Attribute `Name` und `Tierart` vom Typ `Zeichenkette` sowie `Gewicht` vom Typ `Fließkommazahl`. Aktuell ist der Suchbaum nach den Namen der Tiere sortiert.

- a) Gehen Sie davon aus, dass der binäre Suchbaum `tierbaum` alle Tiere eines Zoos enthält. Der Zoo möchte keinen Namen doppelt vergeben. Implementieren Sie eine Operation `suche(name: Zeichenkette, BinTree baum): Wahrheitswert`, die den Wert `wahr` zurückgibt, wenn der als Parameter übergebene binäre Suchbaum `baum` bereits ein Tier mit dem Namen `name` enthält. Testen Sie die Operation, indem Anwendende die Möglichkeit erhalten, einen Namen für ein Tier vorzuschlagen, und das Programm zurückmeldet, ob der Name bereits vergeben ist.
- b) Der Zoo ändert seine Regel für die Namensvergabe, da es immer schwieriger wird, Namen zu finden, die noch nicht vergeben sind. Die Namen sollen nur noch für Tiere einer Tierart eindeutig sein. Diskutieren Sie, inwieweit das Programm und die Speicherung der Namen mithilfe binärer Suchbäume geändert werden sollte, um die neue Regel zu berücksichtigen.
- c) Für die Tiere im `tierBaum` soll nun in der globalen Variablen `neuerBaum` ein neuer Suchbaum erstellt werden, der nach den Gewichten der Tiere sortiert ist. Implementieren Sie eine entsprechende Operation `sortiereNeu(b: BinTree)` und eine Hilfsoperation `ein fuegen(neu: Tier, b: BinTree)`, die ein Tier auf Basis des Gewichtes in den übergebenen Baum einfügt.  
Erläutern Sie, welche Voraussetzung die Gewichte der Tiere erfüllen müssen, damit die Erstellung eines binären Suchbaums, der nach dem Gewicht sortiert ist, möglich ist.
- d) Ein Medikament darf nur an Tiere verabreicht werden, die schwerer als 5 kg sind. Beschreiben Sie, wie der nach Gewicht sortierte Suchbaum genutzt werden kann, um alle Tiere, die schwerer als 5 kg sind, zu finden.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#). Von der Lizenz ausgenommen ist das InfSII-Logo.

Für die korrekte Ausführbarkeit der Quelltexte in diesem Arbeitsblatt wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.