Aufgabe 11.6

Donnerstag, 20. Juli 2017 21:35

a) [1] Beweisen Sie per Induktion, dass es in einem Baum mit $n \geq 1$ Knoten genau $n - 1$ Kanten gibt.

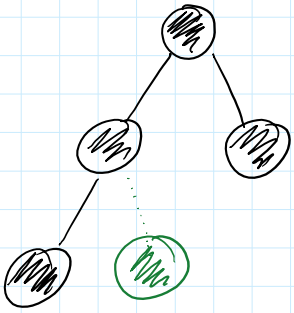
IA.: Für $n = 1$:  $n - 1 = 0$

Ein Baum mit einem Knoten kann keine Kanten besitzen. ✓

IV.: Angenommen es gilt, dass in einem Baum $n \geq 1$ $n \in \mathbb{N}$ genau $n-1$ Kanten gibt

IS.: $n \rightarrow n+1$

Sei ein beliebiger Baum mit n Knoten gegeben.



Will man einen weitere Kante hinzufügen, funktioniert das nicht ohne einen Kreis zu bilden und somit das Kriterium eines Baums zu verletzen.

Will man eine Kante entfernen, zerstört man den Baum.

⇒ Nur möglich einen Knoten mit einer Kante hinzuzufügen

IV.: n ist passender Baum $n = n - 1$

↳ $n + 1 = (n + 1) - 1$ q.e.d

b) [0.5] Die Laufzeit einer Tiefensuche auf einem Graphen mit n Knoten und m Kanten ist $\Theta(n + m)$. Für die Laufzeit in Abhängigkeit von n ergibt sich $\mathcal{O}(n^2)$, da es bis zu $n^2 - n$ Kanten geben kann. Geben Sie eine möglichst kleine Abschätzung der Laufzeit einer Tiefensuche **auf einem Baum** in Abhängigkeit nur von der Zahl n der Knoten an, wenn die Suche bei der Wurzel startet.

Anzahl Kanten Baum: Laut a) $n - 1$

↳ $O(n + n - 1) = O(n)$

c) [1] Gegeben seien nun zwei Zahlen a und b mit $a < b$ und ein binärer Suchbaum B mit $\mathbf{n}$ Knoten als Suchstruktur über einer Liste von paarweise verschiedenen, aufsteigend sortierten Zahlen. In der Liste gebe es $\mathbf{m} > 1$ Zahlen z aus dem Bereich zwischen a und b (d.h. $a \leq z \leq b$).

Betrachten Sie das folgende Verfahren, um alle Elemente zwischen a und b auszugeben:

> Lokalisiere (mittels Suche in B) das Listenelement, dessen Eintrag z die kleinste Zahl ist, für die $z \geq a$ gilt. Gehe die Liste durch bis zum ersten Auftreten eines Eintrags größer als b (oder bis zum Ende der Liste).

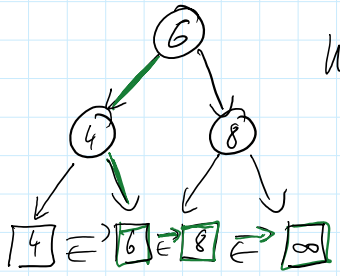
a := 3 b := 7

binärer Suchbaum

Geben Sie für dieses Verfahren den asymptotischen Worst-Case-Aufwand in $\mathcal{O}$-Notation in Abhängigkeit von m und n an und begründen Sie Ihre Antwort kurz.

a = 6

b = 9



Worst-Case: - ganze Liste durchgehen
↳ O(m)
- binärer Baum als „Liste"
↳ O(n)
⇒ O(m+n)

d) [3.5] Fritzchen, ein unerfahrener Programmierer, hat vergessen, die Liste zu verketten. Sie sollen ihm nun helfen, trotzdem alle Elemente zwischen a und b zu finden.

Implementieren Sie eine entsprechende Suche auf B nach den entsprechenden zum Bereich zwischen a und b gehörenden Blättern, wobei nicht in Unterbäumen gesucht werden soll, deren Einträge nicht im Bereich zwischen a und b liegen können.

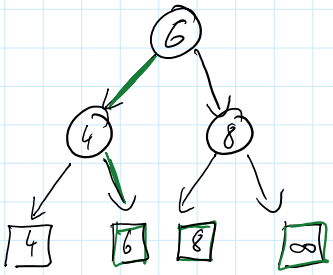
Benutzen Sie bei Ihrer Implementierung die folgenden Methoden und Attribute:

- `isLeaf(t)` – liefert true genau dann, wenn `t` ein Blatt ist
- `t.number` – in `t` gespeicherter Schlüssel; wenn `t` ein Blatt ist, entspricht dieser einem Eintrag in der (gedachten, nicht verketteten) Liste
- `t.leftChild` – gibt das linke Kind von `t` zurück
- `t.rightChild` – gibt das rechte Kind von `t` zurück

Die gefundenen Zahlen z aus dem gesuchten Bereich sollen aufsteigend durch den Aufruf `out(z);` ausgegeben werden. Vervollständigen Sie folgende Funktion:

```
public static void find(Tree t, int a, int b) {

}
```




```
public static void find(Tree t, int a, int b) {
  if ((t.number ≥ a && isLeaf(t.leftChild)) || (t.number < b && isLeaf(t.rightChild)) {
    out(t.number);
  if( t.number >= a )
    find(t.leftChild, int a, int b);
  else if (t.number < b )
    find(t.rightChild, int a, int b);
}
```