Moritz Ruge & Lennard Wittenberg
Bachelor Informatik
Freie Universitäten Berlin

08 Mai 2025

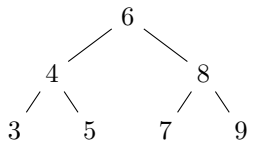
ASSIGNMENT 2 — Algorithmen und Datenstrukturen

Problem 1: Induktion und binäre Bäume

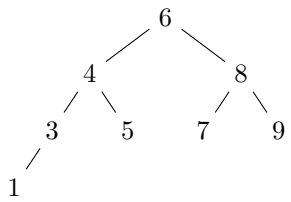
Ein binärer Baum heißt vollständig, falls jeder Knoten entweder null oder zwei Kinder besitzt.

a) Zeichnen Sie einen binären Suchbaum, der vollständig ist, und einen binären Suchbaum, der nicht vollständig ist.

- Balancierter vollständiger BTS:



- Unvollständiger BTS:



⟶ Answer

b) Beweisen Sie durch eine geeignete Induktion: In jedem vollständigen binären Suchbaum ist die Anzahl der Blätter genau um eins größer als die Anzahl der inneren Knoten.

- n jedem vollständigem BTS gilt: jeder Knoten n hat 0 oder genau 2 Kindknoten.
- innerer Knoten: jeder Knoten v gilt als innerer Knoten gdw. v min. 1 Kindknoten hat.
- Blätter Knoten: Jeder Knoten v gilt als Blatt, wenn v keine Kinderknoten hat.

Annahme: In jedem vollständigen binären Suchbaum ist die Anzahl der Blätter genau um eins größer als die Anzahl der inneren Knoten.

base-case BTS mit einem inneren Knoten (root):
Angenommen der BTS erfüllt die Bedingungen eines vollständigen BTS und die Werte der Knoten spielen keine Rolle, dann muss gelten:
$\boldsymbol{num}_{\text{Blätter}} = \boldsymbol{num}_{\text{innerer Knoten}} + \mathbf{1}$

Da die Werte in diesem Fall unerheblich sind, gilt die Zeichnung für einen vollständingen BTS mit einem inneren Knoten:

- $\boldsymbol{num}_{\text{innerer Knoten}} = \boldsymbol{a}\ \boldsymbol{num}_{\text{innere Knoten}} = \mathbf{1}$
- $\boldsymbol{num}_{\text{Blätter}} = \boldsymbol{b, c}\ \boldsymbol{num}_{\text{Blätter}} = \mathbf{2}$

Die Anahme gilt im base-case.

I.S: Wenn die Annahme bei n inneren Knoten gilt, gilt sie auch $\boldsymbol{n} + \mathbf{1}$ inneren Knoten. Ein vollständiger BTS mit n inneren Knoten erfüllt die Gleichung:
$\boldsymbol{num}_{\text{Blätter}} = \boldsymbol{num}_{\text{innerer Knoten}} + \mathbf{1}$
Wenn es einen inneren Knoten zusätzlich geben soll, wird ein Blattknoten zu diesem Knoten und bekommt nach der Definition von vollständigen BTS 2 neue Kindknoten, da:

1. nur 1 Kind Widerspruch mit Definition von vollständigen BTS.
2. 0 Kindknoten, dann bleibt $\boldsymbol{num}_{\text{innerer Knoten}}$ gleich, da das Blatt nicht zum Inneren Knoten werden konnte.

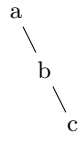
$\longrightarrow$ Answer

c) Formulieren Sie eine ähnliche Aussage für allgemeine binäre Suchbäume und beweisen Sie sie.

Annahme/Eigenschaft: In einem Baum gilt:$num_{node} = num_{edges} + 1 \Rightarrow$ die Anzahl der Knoten i m Baum ergibt sich aus Anzahl der Kanten + 1.

- Jeder Knoten ist durch genau eine Kante mit seinem Elternknoten verbunden.
- Bei Insertion wird ein neuer Knoten mit seinem Elternknoten verbunden.

base-case: $num_{nodes} = 3$



$num_{Knoten} = \{a, b, c, \}\ |num_{Knoten}| = 3$
$num_{Kanten} = \{a - b, b - c\}\ |num_{Kanten}| = 2$
es gilt: $num_{node} = num_{edges} + 1 \Rightarrow 3 = 2 + 1$
Die Annahme gilt für den base-case.

I.S: Wenn die Annahme bei n inneren Knoten gilt, gilt sie auch n+1 inneren Knoten. Ein Baum mit n inneren Knoten erfüllt die Gleichung: $num_{Knoten} = num_{Kanten} + 1$
. Bei $n+1$ Knoten muss ein zusätzlicher Knoten in den Baum eingefügt werden (insertion). Durch den neuen Knoten einstehet immer eine neue Kanten, da ein Knoten nur EIN linkes und rechtes haben kann. ein Knoten

kann nicht zwei linke oder rechte Knoten haben. Daher gilt:

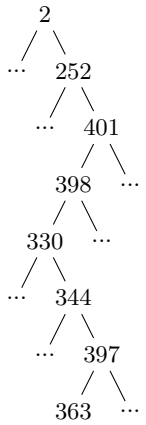
1. Hat der neue Knoten denselben Wert, wie ein anderer Knoten, so wird der alte durch den neuen ersetzt. $\Rightarrow$ immer noch n Knoten Gleichung bleibt unverändert und gilt daher immer noch.
2. Der neue Knoten wird zum Kind eines anderen Knotens und mit einer neuen Kanten mit diesem Elternknoten verbunden. $\Rightarrow n = n + 1$ für die Gleichung gilt: $num_{Knoten} + 1 = num_{Kanten} + 1 + 1 = num_{Knoten} + 1 = num_{Kanten} + 2 \equiv num_{Knoten} = num_{Kanten} + 1$
3. Die Annahme gilt für $n + 1$

□

Problem 2: Binäre Suchbäume

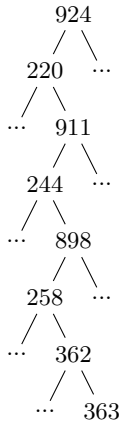
a) Angenommen, wir haben einen binären Suchbaum T , welcher die Zahlen von 1 bis 1000 als Schlüssel speichert. Wir suchen in T nach dem Schlüssel 363. Bestimmen Sie für jede der folgenden Schlüsselfolgen, ob sie als Folge der Einträge auf dem Suchpfad nach 363 auftreten kann. Begründen Sie jeweils Ihre Antwort.

1. **2, 252, 401, 398, 330, 344, 397, 363.**



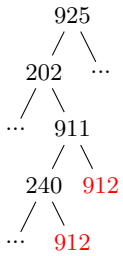
Schritt	Aktueller Knoten	Vergleich mit Ziel 363	Nächster Schritt	Intervall für nächsten Knoten
1	2	$363 > 2 \Rightarrow$ rechts	252	$[3, 1000]$
2	252	$363 > 252 \Rightarrow$ rechts	401	$[253, 1000]$
3	401	$363 < 401 \Rightarrow$ links	398	$[253, 400]$
4	398	$363 < 398 \Rightarrow$ links	330	$[253, 397]$
5	330	$363 > 330 \Rightarrow$ rechts	344	$[331, 397]$
6	344	$363 > 344 \Rightarrow$ rechts	397	$[345, 397]$
7	397	$363 < 397 \Rightarrow$ links	363	$[345, 396]$
8	363	Ziel erreicht	—	—

2. **924, 220, 911, 244, 898, 258, 362, 363.**



Schritt	Aktueller Knoten	Vergleich mit Ziel 363	Nächster Schritt	Intervall für nächsten Knoten
1	924	$363 < 924 \Rightarrow$ links	220	$[1, 923]$
2	220	$363 > 220 \Rightarrow$ rechts	911	$[221, 923]$
3	911	$363 < 911 \Rightarrow$ links	244	$[221, 910]$
4	244	$363 > 244 \Rightarrow$ rechts	898	$[245, 910]$
5	898	$363 < 898 \Rightarrow$ links	258	$[245, 897]$
6	258	$363 > 258 \Rightarrow$ rechts	362	$[259, 897]$
7	362	$363 > 362 \Rightarrow$ rechts	363	$[363, 897]$
8	363	Ziel erreicht	—	—

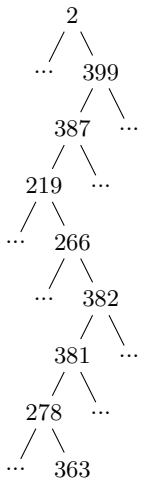
3. **925, 202, 911, 240, 912, 245, 363.**



Schritt	Aktueller Knoten	Vergleich mit Ziel 363	Nächster Schritt	Intervall für nächsten Knoten
1	925	$363 < 925 \Rightarrow$ links	202	$[1, 924]$
2	202	$363 > 202 \Rightarrow$ rechts	911	$[203, 924]$
3	911	$363 < 911 \Rightarrow$ links	240	$[203, 910]$
4	240	$363 > 240 \Rightarrow$ rechts	912	$[241, 910]$
5	912	$363 < 912 \Rightarrow$ links		$912 > [910] \Rightarrow$ Fehler

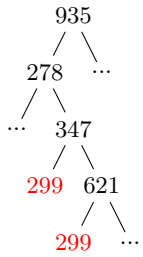
- Der Knoten 912 liegt nicht mehr im Intervall von [241,910] und somit ist der Knoten nicht in der **Schlüsselfolge**.

4. **2, 399, 387, 219, 266, 382, 381, 278, 363.**



Schritt	Aktueller Knoten	Vergleich mit Ziel 363	Nächster Schritt	Intervall für nächsten Knoten
1	2	$363 > 2 \Rightarrow$ rechts	399	$[3, 1000]$
2	399	$363 < 399 \Rightarrow$ links	387	$[3, 398]$
3	387	$363 < 387 \Rightarrow$ links	219	$[3, 386]$
4	219	$363 > 219 \Rightarrow$ rechts	266	$[220, 386]$
5	266	$363 > 266 \Rightarrow$ rechts	382	$[267, 386]$
6	382	$363 < 382 \Rightarrow$ links	381	$[267, 381]$
7	381	$363 < 381 \Rightarrow$ links	278	$[267, 380]$
8	278	$363 > 278 \Rightarrow$ rechts	363	$[279, 380]$
9	363	Ziel Erreicht!	-	-

5. **935, 278, 347, 621, 299, 392, 358, 363.**



Schritt	Aktueller Knoten	Vergleich mit Ziel 363	Nächster Schritt	Intervall für nächsten Knoten
1	935	$363 < 935 \Rightarrow$ links	278	$[1, 934]$
2	278	$363 > 278 \Rightarrow$ rechts	347	$[279, 934]$
3	347	$363 > 347 \Rightarrow$ rechts	621	$[348, 934]$
4	621	$363 < 621 \Rightarrow$ links	299	$[348, 620]$
5	299	$363 > 299 \Rightarrow$ rechts		$299 < [348] \Rightarrow$ Fehler

- Der Knoten 299 liegt nicht mehr im Intervall von [348,620] und somit ist der Knoten nicht in der **Schlüsselfolge**.

b) Sei T ein binärer Baum mit n Knoten, und sei K eine total geordnete Menge von n Schlüsseln. Zeigen Sie, dass es genau eine Möglichkeit gibt, die Schlüssel aus K auf die Knoten von T zu verteilen, so dass die binäre Suchbaumeigen- schaft erfüllt ist.

Gegeben:

- T ist ein binärer Baum mit n Knoten
- $K = \{k_1 < k_2 < ... < k_n\}$ eine total geordnete Menge von n Schlüsseln
- Dann gibt es genau eine Möglichkeit, die Schlüssel aus K auf die Knoten von T zu verteilen, sodass die binäre Suchbaumeigenschaft erfüllt ist.

Induktionsanfang

Ein Baum mit nur einen Knoten $n = 1$ kann nur genau einen Schlüssel enthalten (K_1), damit sind die BST(binary search tree)-Eigenschaften erfüllt.

Induktionsvoraussetzung

Wir nehmen an, dass das für alle binären Bäume mit weniger als n Knoten zutrifft.

Induktionsschritt

T ist nun ein binärer Baum mit $n \geq 2$ Knoten.

Dabei ist der linke Teilbaum T_L und seine Knoten n_L, der rechte Teilbaum T_R und seine Knoten n_R
Es gilt: $n = n_L + 1 + n_R$

Nun müssen wir die Schlüssel $K = \{k_1 < k_2 < ... < k_n\}$ auf die Knoten verteilen damit die BST-Eigenschaft noch erfüllt ist.

- Dabei bekommt der Wuzelknoten von T den Schlüssel k_{n_L+1}
- die kleineren Schlüssel landen in den linken Teilbaum T_L
  - also kriegt der linke Teilbaum die n_L Schlüssel $\{k_1, ..., k_{n_L}\}$
- Schlüssel größer als der Wurzelknoten landen in dem rechten Teilbaum T_R
  - also kriegt der rechte Teilbaum die n_R Schlüssel $\{k_{n_L+2}, ..., k_n\}$

$\Rightarrow$ Wenn wir das ganze nun auf die Induktionsvoraussetzung anwenden, erhalten wir für:

- T_L mit den Schlüsseln n_L $\{k_1, ..., k_{n_L}\}$ $\Rightarrow$ genau eine Möglichkeit die Schlüssel zu verteilen, damit die BST-Eigenschaft gilt.
- T_R mit den Schlüsseln n_R $\{k_{n_L+2}, ..., k_n\}$ $\Rightarrow$ auch genau eine Möglichkeit die Schlüssel zu verteilen, damit die BST-Eigenschaft gilt.
- Damit das ganze funktioniert, muss die Wurzel genau den Schlüssel k_{n_L+1} erhalten

Es gibt also für jeden binären Baum mit n Knoten und jede total geordnete Menge von n Schlüsseln genau eine Möglichkeit, die Schlüssel auf die Knoten zu verteilen, damit die BST-Eigenschaft erfüllt ist.

□

Submitted by Moritz Ruge & Lennard Wittenberg on 08 Mai 2025.