ETH zürich

Übungslektion 8 – Bäume & Heaps

Informatik II

8. / 9. April 2025

Willkommen!

Polybox

Passwort: jschul

Personal Website

`https://n.ethz.ch/~jschul`

Heutiges Programm

Wiederholung von Kursinhalten

Prüfungsaufgaben

Hausaufgaben

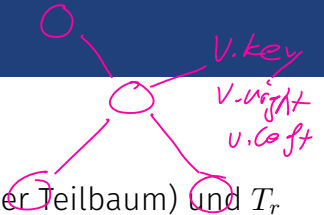
1. Wiederholung von Kursinhalten

Bäume

Ein *Baum* ist

- Verallgemeinerte Liste: Knoten können mehrere Nachfolger haben.
- Spezieller Graph: Graphen bestehen aus Knoten und Kanten. Ein Baum ist ein zusammenhängender, gerichteter, azyklischer Graph.

Binäre Bäume



Ein *binärer Baum* ist

- entweder ein Blatt, d.h. ein leerer Baum,
- oder ein innerer Knoten mit zwei Bäumen T_l (linker Teilbaum) und T_r (rechter Teilbaum) als linken und rechten Nachfolger.

In jedem inneren Knoten **v** wird gespeichert

- ein Schlüssel **v.key** und
- zwei Zeiger **v.left** und **v.right** auf die Wurzeln der linken und rechten Teilbäume.

Ein Blatt wird durch den null-Zeiger repräsentiert. Um überfüllte Diagramme zu vermeiden, lassen wir beim Zeichnen von Bäumen manchmal die Kanten zu den Blättern weg.

Arten von Binären Bäumen

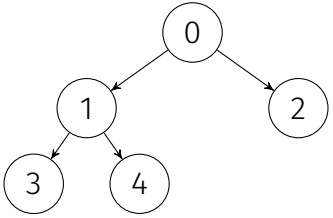
Es gibt drei Arten von Binärbäumen basierend auf der **Anzahl der Kinder**:

- Voller Binärbaum
- Vollständiger Binärbaum
- Perfekter Binärbaum

Hinweis: In der Literatur existieren verschiedene Bezeichnungen und Definitionen!

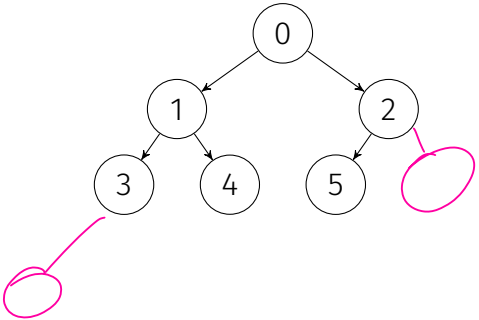
Voller Binärbaum

Ein Binärbaum ist ein voller Binärbaum, wenn jeder Knoten 0 oder 2 Kinder hat.



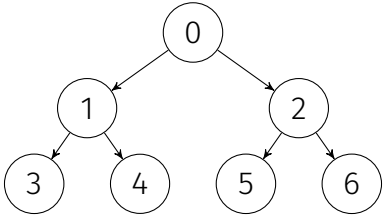
Vollständiger Binärbaum

Ein vollständiger Binärbaum ist ein Baum, in dem alle Ebenen vollständig gefüllt sind, ausser möglicherweise die letzte Ebene, die von links nach rechts gefüllt ist.



Perfekter Binärbaum

Ein Binärbaum ist ein Baum, bei dem alle inneren Knoten zwei Kinder haben und alle Blätter auf derselben Ebene sind.



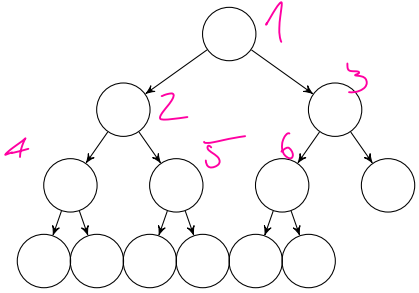
Binäre Bäume: Quiz

Wie viele Knoten enthält ein voller Binärbaum mit 6 Nicht-Blatt-Knoten höchstens:

A. 6 Knoten
B. 9 Knoten
C. 11 Knoten
D. 13 Knoten

Binäre Bäume: Quiz Antwort

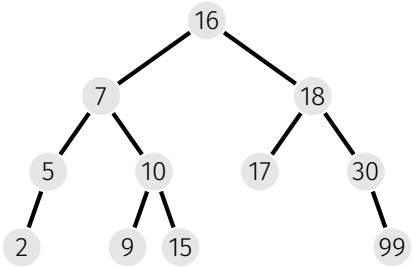
Ein voller Binärbaum mit n Nicht-Blatt-Knoten enthält $2n + 1$ Knoten. Also: $2 \times 6 + 1 = 13$.



Binäre Suchbäume

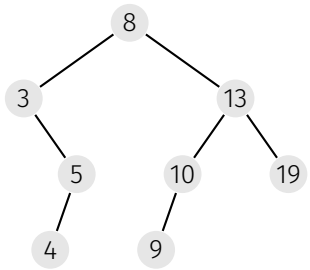
Ein *binärer Suchbaum* ist ein binärer Baum, der die **Suchbaumeigenschaft** erfüllt:

- Jeder Knoten `v` speichert einen Schlüssel
- Schlüssel im linken Teilbaum `v.left` kleiner als `v.key`
- Schlüssel im rechten Teilbaum `v.right` grösser als `v.key`



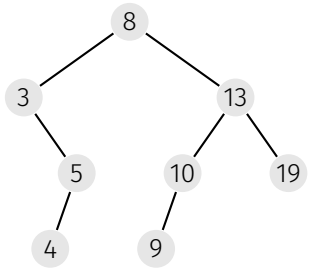
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.



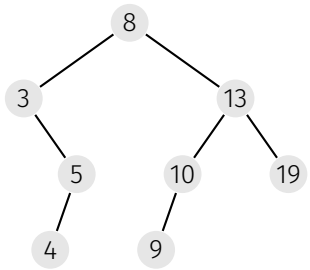
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19



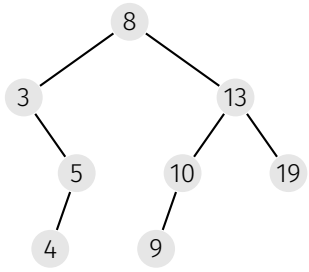
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19
- *Nebenreihenfolge* (*postorder*):
  $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.



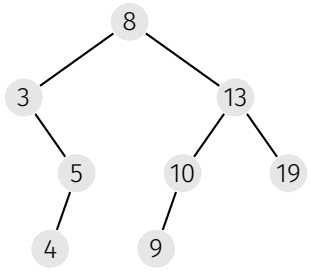
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19
- *Nebenreihenfolge* (*postorder*):
  $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.
  4, 5, 3, 9, 10, 19, 13, 8



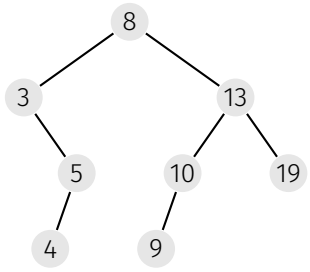
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19
- *Nebenreihenfolge* (*postorder*):
  $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.
  4, 5, 3, 9, 10, 19, 13, 8
- *Symmetrische Reihenfolge* (*inorder*):
  $T_{\text{left}}(v)$, dann v, dann $T_{\text{right}}(v)$.



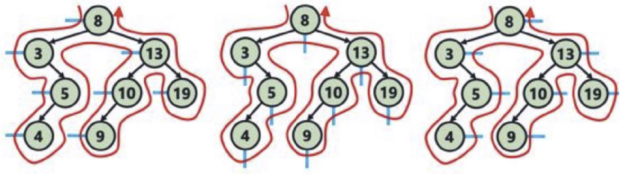
Traversierungsarten

- *Hauptreihenfolge* (*preorder*):
  v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19
- *Nebenreihenfolge* (*postorder*):
  $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.
  4, 5, 3, 9, 10, 19, 13, 8
- *Symmetrische Reihenfolge* (*inorder*):
  $T_{\text{left}}(v)$, dann v, dann $T_{\text{right}}(v)$.
  3, 4, 5, 8, 9, 10, 13, 19



Frame Title

Trick um Traversierungsreihennfolge schneller zu bestimmen:

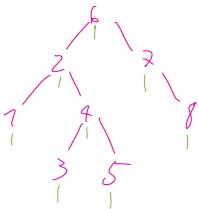


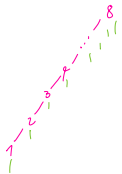
Traversierungsarten: Quiz

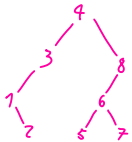
Zeichnen Sie jeweils einen binären Suchbaum, der die folgenden Traversierungen erzeugt. Ist der Baum eindeutig?

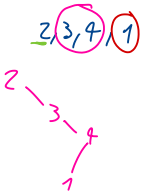
Symmetrische Reihenfolge (inorder)	1 2 3 4 5 6 7 8
Hauptreihenfolge (preorder)	4 3 1 2 8 6 5 7
Nebenreihenfolge (postorder)	1 3 2 5 6 8 7 4

Geben Sie zu jeder Reihenfolge eine Zahlensequenz aus $\{1, \ldots, 4\}$, die nicht aus einem gültigen binären Suchbaum stammen kann.







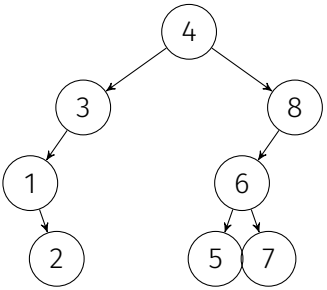


Traversierungsarten: Quiz Antwort

- Jeder Suchbaum mit den Zahlen $\{1, \ldots, 8\}$ ist gültig.
- Der Baum ist nicht eindeutig.
- Wenn die Sequenz nicht sortiert ist, gibt es keinen gültigen Suchbaum. Gegenbeispiel: 1 2 4 3

Traversierungsarten: Quiz Antwort

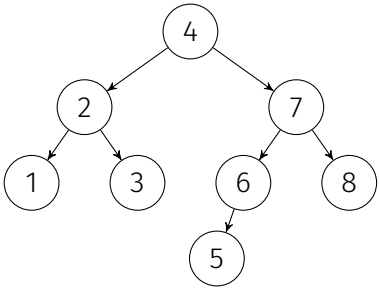
- Hauptreihenfolge (preorder): 4 3 1 2 8 6 5 7



- Der Baum ist eindeutig.
- Es muss rekursiv gelten, dass zuerst eine Gruppe Zahlen kleiner und danach grösser als der erste Wert kommen. Gegenbeispiel: 3 1 4 2

Traversierungsarten: Quiz Antwort

- Nebenreihenfolge (postorder): 1 3 2 5 6 8 7 4



- Der Baum ist eindeutig.
- Konstruktion hier: `https://www.techiedelight.com/build-binary-search-tree-from-postorder-sequence/`, Ähnliches Argument wie vorher, nur von hinten nach vorne. Gegenbeispiel 4 2 1 3

Heaps

Ein *Heap* ist eine baumbasierte Datenstruktur wobei:

- Der Baum ein **vollständiger Binärbaum** ist.
- Ein Heap mit **N** Knoten eine Höhe von **log N** hat (Eigenschaft eines vollständigen Binärbaums).
- Es ist für die schnelle Extraktion von Minimum oder Maximum und für das Sortieren optimiert.

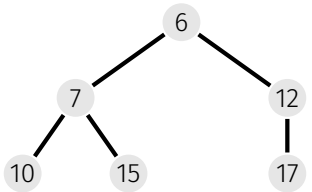
Es gibt zwei Arten von Heap-Implementierungen:

- Min-Heaps
- Max-Heaps

Min-Heaps

Ein *Min-Heap* ist ein binärer Baum wo:

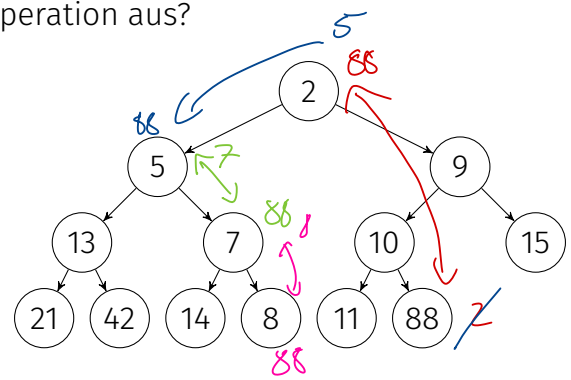
- Der am Wurzelknoten vorhandene Schlüssel ist immer **kleiner oder gleich** den Schlüsseln aller seiner Kinder.
- Dieselbe Eigenschaft muss für alle Teilbäume in diesem Binärbaum rekursiv wahr sein.
- Das **kleinste** Schlüsselelement ist daher immer an der Wurzel vorhanden.



Min-Heaps: Quiz

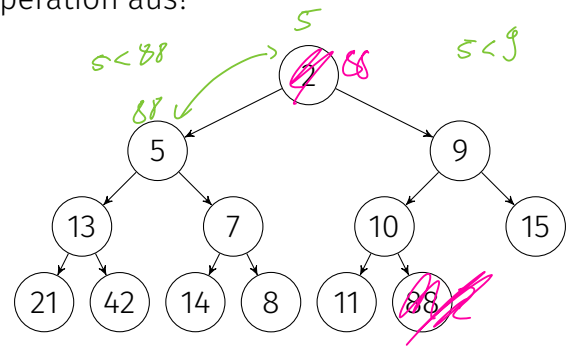
Führen Sie auf folgendem Min-Heap eine Extract-Min (entferne den kleinsten Schlüssel) Operation aus, wie in der Vorlesung vorgestellt, einschliesslich der Wiederherstellung der Heap-Bedingung. Wie sieht der Heap nach der Operation aus?

1
2
3
4

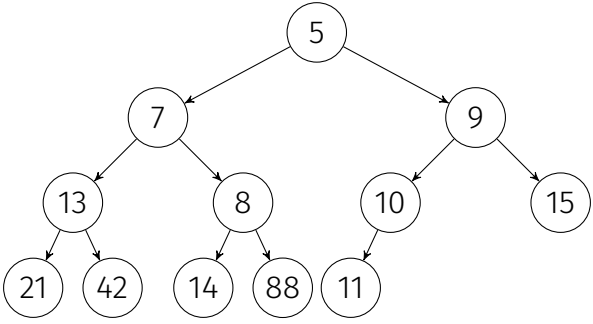


Min-Heaps: Quiz

Führen Sie auf folgendem Min-Heap eine Extract-Min (entferne den kleinsten Schlüssel) Operation aus, wie in der Vorlesung vorgestellt, einschliesslich der Wiederherstellung der Heap-Bedingung. Wie sieht der Heap nach der Operation aus?



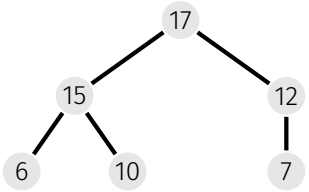
Min-Heaps: Quiz Antwort



Max-Heaps

Ein *Max-Heap* ist ein binärer Baum wo:

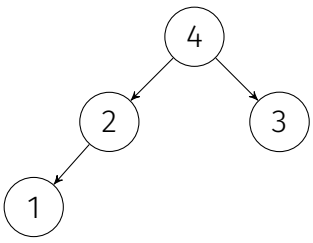
- Der am Wurzelknoten vorhandene Schlüssel ist immer **größer oder gleich** den Schlüsseln aller seiner Kinder.
- Dieselbe Eigenschaft muss für alle Teilbäume in diesem Binärbaum rekursiv wahr sein.
- Das **größte** Schlüsselelement ist daher immer an der Wurzel vorhanden.

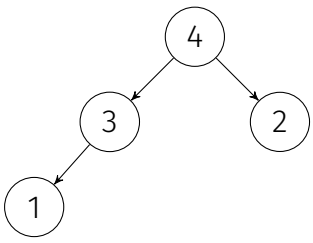


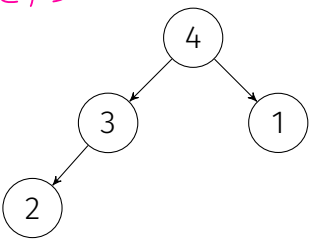
Anzahl Max-Heaps

Sei $N(n)$ die Anzahl verschiedener Max-Heaps, welche aus allen Schlüsseln $1, 2, \ldots, n$ gebildet werden können. Beispielsweise ist $N(1) = 1,\ N(2) = 1,\ N(3) = 2,\ N(4) = 3$ und $N(5) = 8$.
Finde die Werte $N(6)$ und $N(7)$.

1, 2, 3

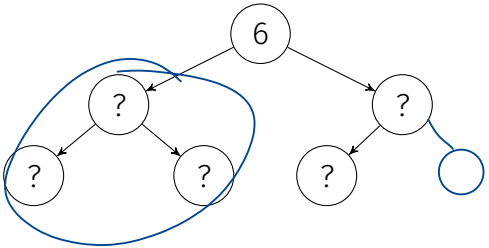






Anzahl Max-Heaps

Ein die Elemente $1, 2, 3, 4, 5, 6$ enthaltender Max-Heap sieht so aus:



Anzahl Max-Heaps

Möglichkeiten, Elemente des linken Teilbaums zu wählen: $\binom{5}{3}$.

$$\Rightarrow N(6) = \binom{5}{3} \cdot N(3) \cdot N(2) = 10 \cdot 2 \cdot 1 = 20.$$

$$\text{und } N(7) = \binom{6}{3} \cdot N(3) \cdot N(3) = 20 \cdot 2 \cdot 2 = 80.$$

$5 = \binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} = \frac{5!}{2! \cdot 3!} = \frac{5 \cdot 4}{2} = 10$

Komplexitätsanalyse von Min-Heap und Max-Heap

- Erhalten des größten oder kleinsten Elements: **O(1)**
- Element in Max-Heap oder Min-Heap einfügen: **O(log N)**
- Größte- oder Kleinste-Element entfernen: **O(log N)**

Schlüssel Einfügen

Binärer Suchbaum

- Nach Schlüssel suchen.
- Bei erreichtem leeren Blatt (`null`) einfügen.

Min-Heap

- Zuhinterst im Array einfügen.
- Heap-Bedingung wiederherstellen: `siftUp` (Aufsteigen lassen).

Schlüssel Einfügen

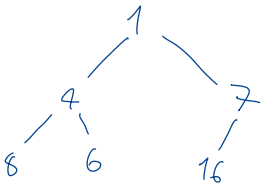
Binärer Suchbaum

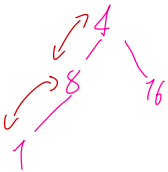
- Nach Schlüssel suchen.
- Bei erreichtem leeren Blatt (`null`) einfügen.

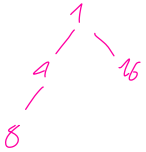
Min-Heap

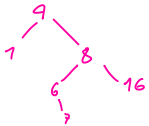
- Zuhinterst im Array einfügen.
- Heap-Bedingung wiederherstellen: `siftUp` (Aufsteigen lassen).

Aufgabe: Einfügen von $4, 8, 16, 1, 6, 7$ in leeren Baum/Heap.


1
4
7
8
6
16


4
8
16
1

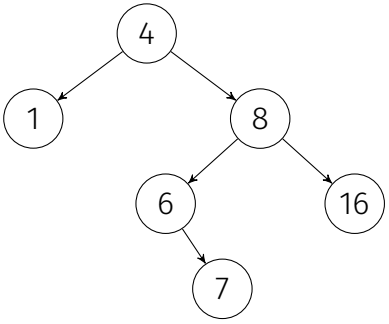

1
4
16
8


9
1
8
6
16
7

Schlüssel Einfügen

Binärer Suchbaum

- Nach Schlüssel suchen.
- Bei erreichtem leeren Blatt (`null`) einfügen.



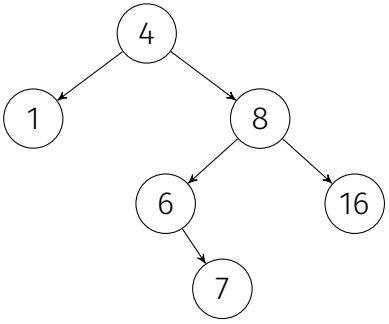
Min-Heap

- Zuhinterst im Array einfügen.
- Heap-Bedingung wiederherstellen: `siftUp` (Aufsteigen lassen).

Schlüssel Einfügen

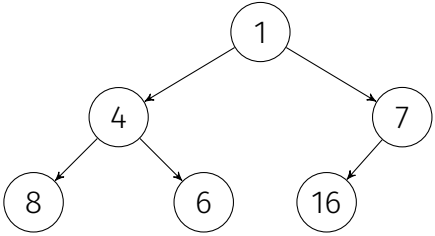
Binärer Suchbaum

- Nach Schlüssel suchen.
- Bei erreichtem leeren Blatt (`null`) einfügen.



Min-Heap

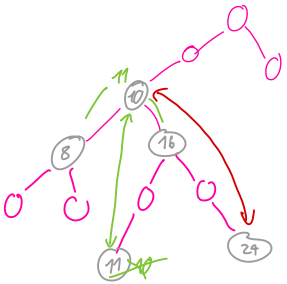
- Zuhinterst im Array einfügen.
- Heap-Bedingung wiederherstellen: `siftUp` (Aufsteigen lassen).



Schlüssel Löschen

Binärer Suchbaum

- Schlüssel k durch symm. Nachfolger n ersetzen.
- Achtung: Wohin mit rechtem Kind von n?



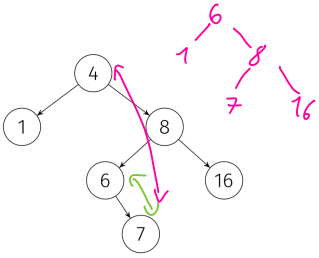
Min-Heap

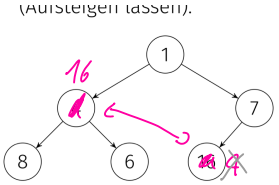
- Schlüssel durch hinterstes Arrayelement ersetzen.
- Heap-Bedingung wiederherstellen: `siftDown` *oder* `siftUp`.

Schlüssel Löschen

Binärer Suchbaum

- Schlüssel k durch symm. Nachfolger n ersetzen.
- Achtung: Wohin mit rechtem Kind von n?





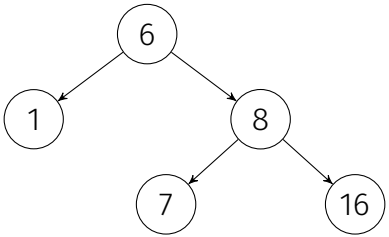
Min-Heap

- Schlüssel durch hinterstes Arrayelement ersetzen.
- Heap-Bedingung wiederherstellen: `siftDown` *oder* `siftUp`.

Schlüssel Löschen

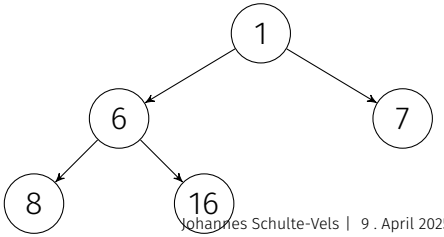
Binärer Suchbaum

- Schlüssel k durch symm. Nachfolger n ersetzen.
- Achtung: Wohin mit rechtem Kind von n?



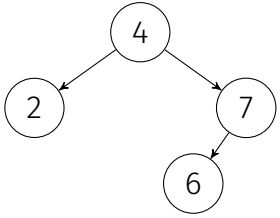
Min-Heap

- Schlüssel durch hinterstes Arrayelement ersetzen.
- Heap-Bedingung wiederherstellen: `siftDown` *oder* `siftUp`.



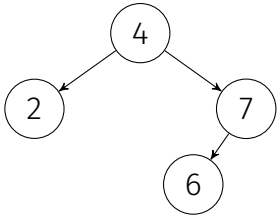
Schlüssel Löschen: Quiz

Wenn Sie zwei Schlüssel aus einem Binären Suchbum löschen wollen, spielt es dann eine Rolle, in welcher Reihenfolge Sie dies tun? Mit anderen Worten, ist das Löschen kommutativ?



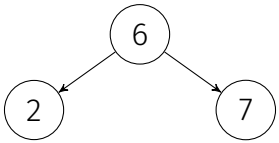
Schlüssel Löschen: Quiz Antwort

Ja, die Reihenfolge kann wichtig sein. Betrachten wir ein Gegenbeispiel. Angenommen, wir löschen Schlüssel 4 aus diesem Baum.



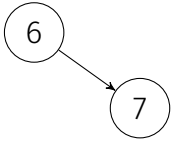
Schlüssel Löschen: Quiz Antwort

Der Schlüssel 6 ist jetzt die neue Wurzel. Löschen wir nun Schlüssel 2.



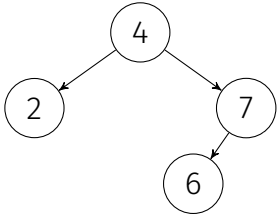
Schlüssel Löschen: Quiz Antwort

Schlüssel 2 wurde gelöscht. Dies ist der Binäre Suchbaum, den wir erhalten, nachdem wir Schlüssel 4 und dann Schlüssel 2 gelöscht haben.



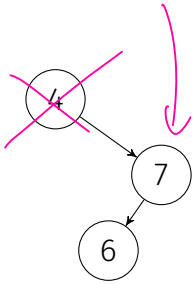
Schlüssel Löschen: Quiz Antwort

Kehren wir zum ersten Baum zurück. Angenommen, wir löschen zuerst Schlüssel 2.



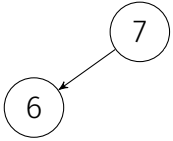
Schlüssel Löschen: Quiz Antwort

Löschen wir nun Schlüssel 4.



Schlüssel Löschen: Quiz Antwort

Schlüssel 7 ist nun die Wurzel des Baumes, nicht Schlüssel 6 wie zuvor. Wir haben gezeigt, dass das Löschen von Schlüsseln nicht immer kommutativ ist.



2. Prüfungsaufgaben

Alte Prüfungsaufgaben

In der folgenden Tabelle ist ein Min-Heap in seiner üblichen Form gespeichert. Wie sieht die Tabelle aus, nachdem die Zahl 17 eingefügt wurde?

1	2	3	4	5	6	7	8	9	10	11	12
2	9	7	25	23	32	26	48	37	39	39	37

1	2	3	4	5	6	7	8	9	10	11	12	13

Alte Prüfungsaufgaben

In der folgenden Tabelle ist ein Min-Heap in seiner üblichen Form gespeichert. Wie sieht die Tabelle aus, nachdem die Zahl 17 eingefügt wurde?

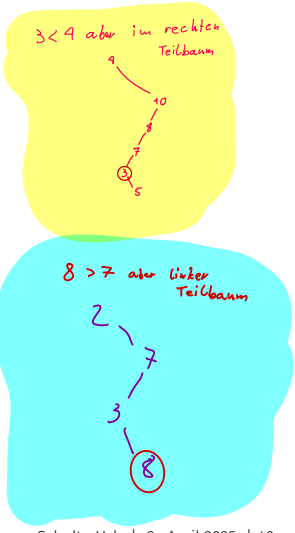
1	2	3	4	5	6	7	8	9	10	11	12
2	9	7	25	23	32	26	48	37	39	39	37

1	2	3	4	5	6	7	8	9	10	11	12	13
2	9	7	25	23	17	26	48	37	39	39	37	32

Binärbaum

Angenommen, ein bestimmter binärer Suchbaum hat ganze Zahlen zwischen 1 und 10 als Schlüssel und wir suchen nach der Zahl 5. Welche der folgenden Sequenzen können **nicht** die untersuchte Schlüsselfolge sein?

- ◯ 10, 9, 8, 7, 6, 5
- ⊗ 4, 10, 8, 7, 3, 5
- ◯ 1, 10, 2, 9, 3, 8, 4, 7, 6, 5
- ⊗ 2, 7, 3, 8, 4, 5

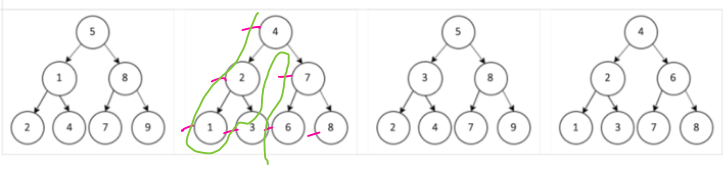


Binärbaum

Binärbaum

2.b)

Nach einer Traversierung eines Baumes mit der Preorder Reihenfolge bekommen wir die Elemente: 4, 2, 1, 3, 7, 6, 8. Welche der folgenden Abbildungen entspricht dem tatsächlichen Baum? / *A preorder traversal of a tree yields: 4, 2, 1, 3, 7, 6, 8. Which of the following figures correspond to the actual tree?*



(A) (B) (C) (D)

○ A ⊠ B ○ C ○ D

Binärbaum

Gegeben ein perfekt balancierter binärer Suchbaum mit n Elementen, was ist asymptotisch die schlechtest mögliche Anzahl Vergleiche um das Element mit dem kleinsten Wert zu finden, mit der besten Ihnen bekannten Methode?

Given a perfectly balanced Binary Search Tree with n elements, what is the asymptotical worst case number of comparisons for finding the minimum value element, using the best method you know?

Anzahl Vergleiche? / *Number of comparisons?*

⊗ $\log n$ ○ $\sqrt{n}$ ○ n ○ $n \log n$ ○ n^2 ○ e^n ○ $n!$

Binärbaum

Gegeben ein Binärbaum mit 31 Elementen.
Was ist die Minimale/Maximale Baumtiefe?

Given a binary tree with 31 elements. What is the minimal/maximal tree depth?

2^0

$\lceil 31 \rceil \rightarrow 32$

(3.c.1)

Minimale Baumtiefe? / *Minimal tree depth?*

◯ 1 ◯ 4 ◯ 5 ◯ 16 ◯ 31 ◯ 63

$2^x \stackrel{!}{=} 32$

$x = (\log_2(32)$

(3.c.2)

Maximale Baumtiefe? / *Maximum tree depth?*

◯ 1 ◯ 4 ◯ 5 ◯ 16 ◯ 31 ◯ 63

Binärbaum

1. Fügen Sie den Schlüssel 16 in den Baum ein.

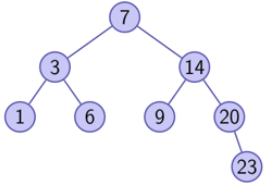
2. Löschen Sie den Schlüssel 14. Nutzen Sie hierbei den symmetrischen Nachfolger, falls nötig.

Nach Durchführung aller Operationen, beantworten Sie die Fragen über den resultierenden Graph. Schreiben Sie NONE falls das Kind nicht existiert.

1.Insert the key 16 into the tree.

2.Delete the key 14. Use the symmetric successor, if needed.

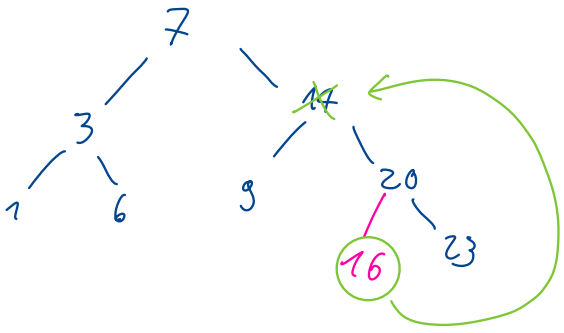
After the operations are performed, answer the questions about the resulting graph. Enter NONE if the child does not exist.



Binärbaum

S: 16, 20, None

1. 16 > 7, .. > 14, .. < 20
2. Symm. Nachfolger



Binärbaum

(3.e.1)

Was ist der Wert des rechten Kind von 7? / *What is the value of 7's right child?*

16

(3.e.2)

Was ist der Wert des rechten Kind von 16? / *What is the value of 16's right child?*

20

(3.e.3)

Was ist der Wert des linken Kind von 20? / *What is the value of 20's left child?*

NONE

Binärbaum

Heaps

Das folgende Array a repräsentiert einen Min-Heap. Zeigen Sie das resultierende Array a nachdem Sie das Minimum extrahieren und die Heap-Bedingung wieder hergestellt haben.

The following array a represents a min-heap. Show the resulting array a after you extract the minimum and reestablish the heap-condition.

1	3	2	6	5	4	9	11	8	13	7
										-

Tragen Sie die Lösung in Moodle Komma-getrennt ohne Leerschläge ein, also zum Beispiel `0,1,2`. / *Enter the solution in Moodle comma-separated without spaces, e.g., `0,1,2`.*

Heaps

(3.d) Das folgende Array a repräsentiert einen Min-Heap. Zeigen Sie das resultierende Array a nachdem Sie das Minimum extrahieren und die Heap-Bedingung wieder hergestellt haben.

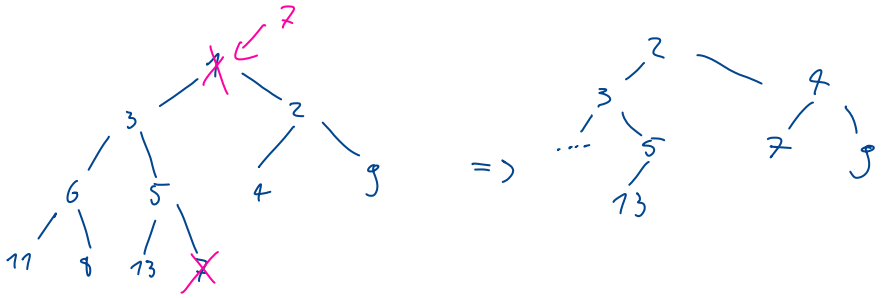
The following array a represents a min-heap. Show the resulting array a after you extract the minimum and reestablish the heap-condition.

1	3	2	6	5	4	9	11	8	13	7
2	3	4	6	5	7	9	11	8	13	-

Tragen Sie die Lösung in Moodle Komma-getrennt ohne Leerschläge ein, also zum Beispiel `0,1,2`. / *Enter the solution in Moodle comma-separated without spaces, e.g., `0,1,2`.*

Heaps

Heaps 1: 2, 3, 4, 6, 5, 7, 9, 11, 8, 13



3. Hausaufgaben

Übung 7: Trees

Auf `https://expert.ethz.ch/enrolled/SS25/mavt2/exercises`

Exercise 7: Trees Einfach, Mittel, Schwer

- Binary Search Trees and Heaps Gute Übung, ZF, Slides
- Implementing a Binary Search Tree Sehr gut für Versändnis!
- Concatenating Heaps Tricky, achtet auf Definitionen
- Heapsort Vorlesung / ZF zur Hilfe

Abgabedatum: Montag 14.04.2025, 20:00 MEZ

KEINE HARDCODIERUNG

Feedback

https://n.ethz.ch/~jschul/Feedback