Informatik II

Übung 5

FS 2020

Heutiges Programm

1. Feedback letzte Übung
2. Wiederholung Theorie

Wiederholung: Binäre Bäume, Schlüssel Einfügen

Binäre Suchbäume

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

MinHeap

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

Wiederholung: Binäre Bäume, Schlüssel Einfügen

Binäre Suchbäume

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

MinHeap

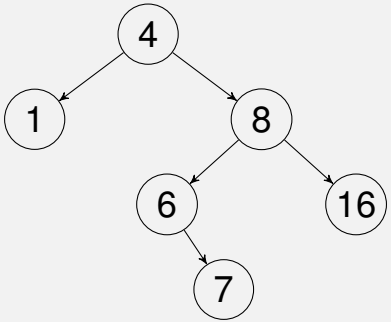
- Zuhinterst im Array einfügen.
- Heap-Bedingung wiederherstellen: `siftUp` (Aufsteigen lassen).

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

Wiederholung: Binäre Bäume, Schlüssel Einfügen

Binäre Suchbäume

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



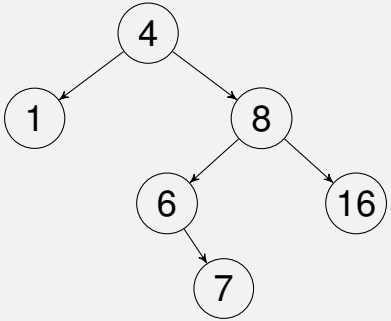
MinHeap

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

Wiederholung: Binäre Bäume, Schlüssel Einfügen

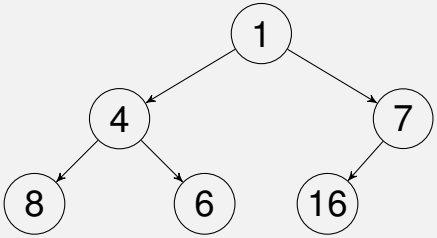
Binäre Suchbäume

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



MinHeap

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



Wiederholung: Binäre Bäume, Schlüssel Löschen

Binäre Suchbäume

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

MinHeap

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

Wiederholung: Binäre Bäume, Schlüssel Löschen

Binäre Suchbäume

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

MinHeap

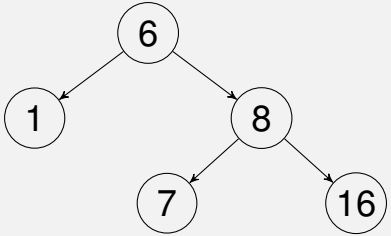
- Schlüssel durch hinterstes Arrayelement ersetzen.
- Heap-Bedingung wiederherstellen: `siftDown` *or* `siftUp`.

Aufgabe: Löschen von 4 in Beispiel-Baum/Heap.

Wiederholung: Binäre Bäume, Schlüssel Löschen

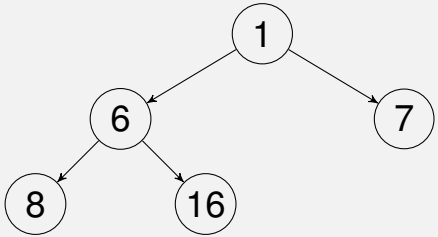
Binäre Suchbäume

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



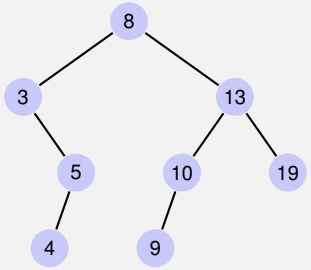
MinHeap

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



Traversierungsarten

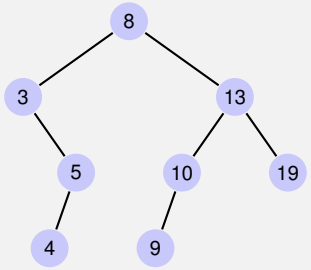
- Hauptreihenfolge (preorder): v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
- Nebenreihenfolge (postorder): $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.
- Symmetrische Reihenfolge (inorder): $T_{\text{left}}(v)$, dann v, dann $T_{\text{right}}(v)$.



Traversierungsarten

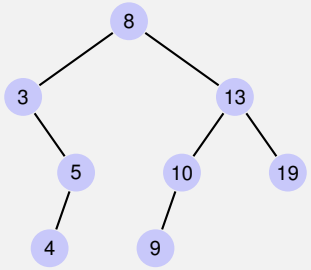
- Hauptreihenfolge (preorder): v, dann $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$.
  8, 3, 5, 4, 13, 10, 9, 19
- Nebenreihenfolge (postorder): $T_{\text{left}}(v)$, dann $T_{\text{right}}(v)$, dann v.

- Symmetrische Reihenfolge (inorder): $T_{\text{left}}(v)$, dann v, dann $T_{\text{right}}(v)$.



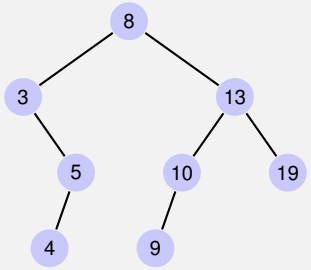
Traversierungsarten

- 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 $T_{\text{right}}(v)$.



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



Quiz (tricky)

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.

Antworten

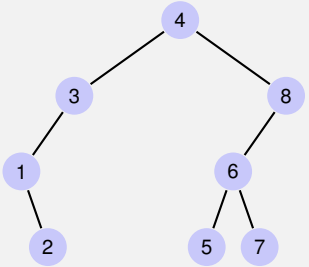
Symmetrische Reihenfolge: jeder Suchbaum mit den Zahlen $\{1, \ldots, 8\}$ ist gültig.

Der Baum ist nicht eindeutig

Es gibt keinen Suchbaum, welcher nicht die aufsteigend sortierte Sequenz ausgeben würde. Gegenbeispiel 1 2 4 3

Antworten

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

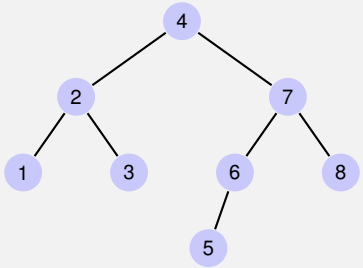


Der Baum ist eindeutig

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

Antworten

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

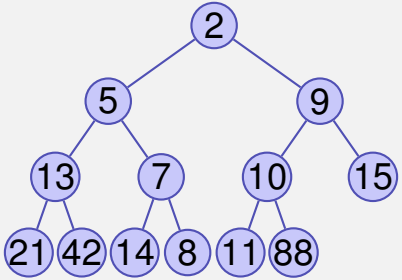


Der Baum ist eindeutig

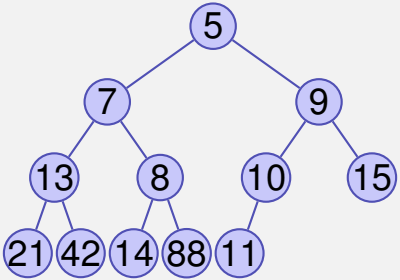
Konstruktion hier: `https://www.techiedelight.com/build-binary-search-tree-from-postorder-sequence/`, Ähnliches Argument wie vorher, nur von hinten nach vorne. Gegenbeispiel 4 2 1 3

Heap

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



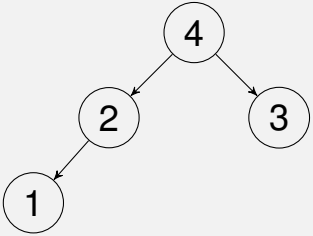
Lösung

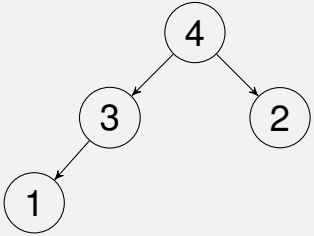


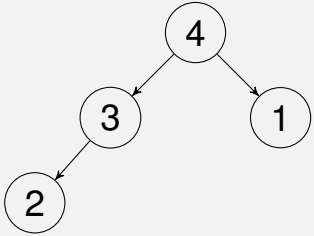
Anzahl MaxHeaps mit n verschiedenen Schlüsseln

Sei $N(n)$ die Anzahl verschiedener MaxHeaps, 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 $N(7)$.

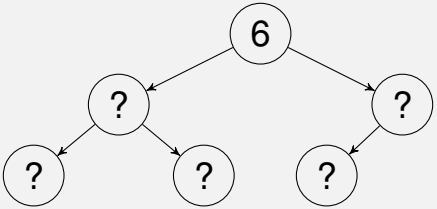






Anzahl MaxHeaps mit n verschiedenen Schlüsseln

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



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

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

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

Fragen oder Anregungen?