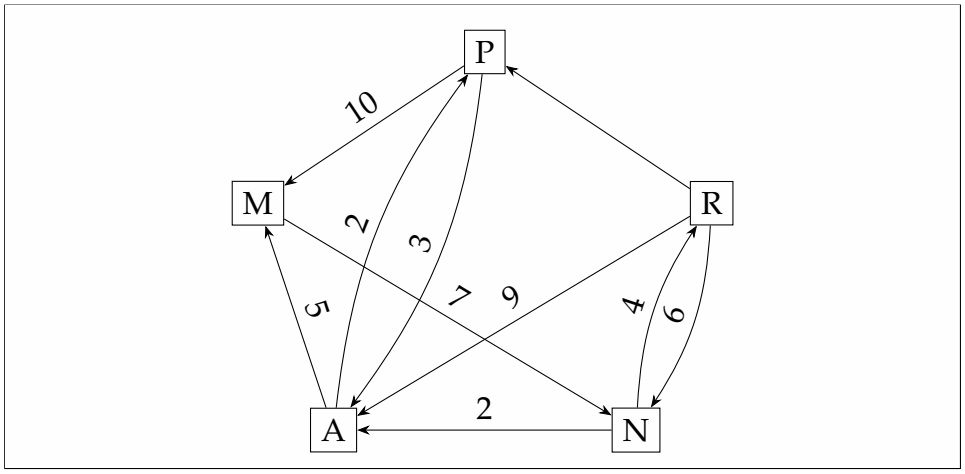
gerichteter Distanzgraph angegeben durch Adjazenzmatrix [Graph M A P R N]

Ein gerichteter Distanzgraph sei durch seine Adjazenzmatrix gegeben (in einer Zeile stehen die Längen der von dem Zeilenkopf ausgehenden Wege.)[1]

$$\begin{array}{c} \\ A \\ M \\ N \\ P \\ R \end{array} \begin{array}{c} \begin{array}{ccccc} A & M & N & P & R \end{array} \\ \begin{pmatrix} * & 5 & - & 2 & - \\ - & * & 7 & - & - \\ 2 & - & * & - & 4 \\ 3 & 10 & - & * & - \\ 9 & - & 6 & 1 & * \end{pmatrix} \end{array}$$

(a) Stellen Sie den Graph in der üblichen Form dar.

Lösungsvorschlag



(b) Bestimmen Sie mit dem Algorithmus von Dijkstra ausgehend von M die kürzeste Wege zu allen anderen Knoten.

Lösungsvorschlag

[1] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen: Aufgabenblatt 6: Graphen, Bäume*, (Check-Up) Seite 3, Aufgabe 5.

Nr.	besucht	A	M	N	P	R
0		∞	0	∞	∞	∞
1	M	∞	**0**	7	∞	∞
2	N	9	|	**7**	∞	11
3	A	**9**	|	|	11	11
4	P	|	|	|	**11**	11
5	R	|	|	|	|	**11**

nach	Entfernung	Reihenfolge	Pfad
M $\rightarrow$ A	9	3	M $\rightarrow$ N $\rightarrow$ A
M $\rightarrow$ M	0	1	
M $\rightarrow$ N	7	2	M $\rightarrow$ N
M $\rightarrow$ P	11	4	M $\rightarrow$ N $\rightarrow$ A $\rightarrow$ P
M $\rightarrow$ R	11	5	M $\rightarrow$ N $\rightarrow$ R

(c) Beschreiben Sie wie ein Heap als Prioritätswarteschlange in diesem Algorithmus verwendet werden kann.

Lösungsvorschlag

Ein Heap kann in diesem Algorithmus dazu verwendet werden, den nächsten Knoten mit der kürzesten Distanz zum Startknoten auszuwählen.

(d) Geben Sie die Operation *„Entfernen des Minimums“* für einen Heap an. Dazu gehört selbstverständlich die Restrukturierung des Heaps.

Lösungsvorschlag

```
static int NO_VALUE = Integer.MIN_VALUE;

static void heapify(int a[], int i, int last) {
  int smallest = i;
  int l = 2 * i + 1;
  int r = 2 * i + 2;

  if (l <= last && a[l] != NO_VALUE && a[l] < a[smallest]) {
    smallest = l;
  }
```

```
    if (r <= last && a[r] != NO_VALUE && a[r] < a[smallest]) {
      smallest = r;
    }

    if (smallest != i) {
      int swap = a[i];
      a[i] = a[smallest];
      a[smallest] = swap;
      heapify(a, smallest, last);
    }
  }

  static int removeMin(int a[]) {
    int result = a[0];
    int last = a.length - 1;
    while (last > 0 && a[last] == NO_VALUE) {
      last--;
    }
    a[0] = a[last];
    a[last] = NO_VALUE;
    heapify(a, 0, last);
    return result;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/Heap.java`

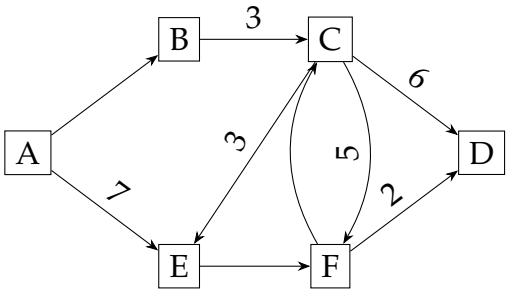
Staatsexamen 46115 / 2012 / Frühjahr / Thema Nr. 1 / Aufgabe Nr. 6

Aufgabe 6 [Graph A-F]

Aufgabe 6[2]

Gegeben sei der folgende gerichtete Graph $G = (V, E, d)$ mit den angegebenen Kantengewichten.

[2] **examen:46115:2012:03**.



(a) Geben Sie eine formale Beschreibung des abgebildeten Graphen G durch Auflistung von V, E und d an.

Lösungsvorschlag

$G = (V, E, d)$

mit

$V = \{A, B, C, D, EF\}$

und

$E = \{(A,B), (A,E), (B,C), (C,D), (C,E), (C,F), (E,F), (F,C), (F,D), \}$

und

$d = \{1, 7, 3, 6, 3, 5, 1, 1, 2\}$

Als Adjazenzliste

A: $\rightarrow$ B $\quad \xrightarrow{7}$ E

B: $\xrightarrow{3}$ C

C: $\xrightarrow{6}$ D $\quad \xrightarrow{3}$ E $\quad \xrightarrow{5}$ F

D:

E: $\rightarrow$ F

F: $\rightarrow$ C $\quad \xrightarrow{2}$ D

(b) Erstellen Sie die Adjazenzmatrix A zum Graphen G.

Lösungsvorschlag

$$
\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}
\begin{array}{c}
\begin{array}{cccccc} A & B & C & D & E & F \end{array} \\
\begin{pmatrix} * & 1 & - & - & 7 & - \\ - & * & 3 & - & - & - \\ - & - & * & 6 & 3 & 5 \\ - & - & - & * & - & - \\ - & - & - & - & * & 1 \\ - & - & 1 & 2 & - & * \end{pmatrix}
\end{array}
$$

(c) Berechnen Sie unter Verwendung des Algorithmus nach Dijkstra - vom Knoten A beginnend - den kürzesten Weg, um alle Knoten zu besuchen. Die Restknoten werden in einer Halde (engl. Heap) gespeichert. Geben Sie zu jedem Arbeitsschritt den Inhalt dieser Halde an.

Lösungsvorschlag

Nr.	**besucht**	**A**	**B**	**C**	**D**	**E**	**F**
0		0	∞	∞	∞	∞	∞
1	A	**0**	1	∞	∞	7	∞
2	B	|	**1**	4	∞	7	∞
3	C	|	|	**4**	10	7	9
4	E	|	|	|	10	**7**	8
5	F	|	|	|	10	|	**8**
6	D	|	|	|	**10**	|	|

nach	**Entfernung**	**Reihenfolge**	**Pfad**
A → A	0	1	
A → B	1	2	A → B
A → C	4	3	A → B → C
A → D	10	6	A → B → C → D
A → E	7	4	A → E
A → F	8	5	A → E → F

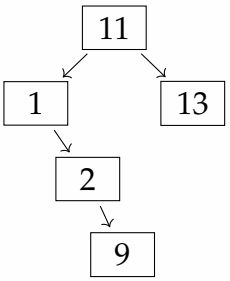
Aufgabe 7 [Zahlenfolge in binärer Suchbaum, Min-Heap und AVL-Baum]

Fügen Sie nacheinander die Zahlen 11, 1, 2, 13, 9, 10, 7, 5[3]

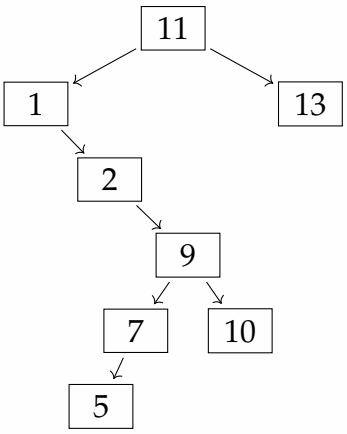
(a) in einen leeren binären Suchbaum und zeichnen Sie den Suchbaum jeweils nach dem Einfügen von „9“ und „5“

Lösungsvorschlag

Nach dem Einfügen von „9“:



Nach dem Einfügen von „5“:

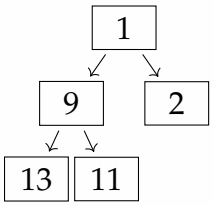


[3] *Staatsexamen 46115 Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft) 2014 Frühjahr.*

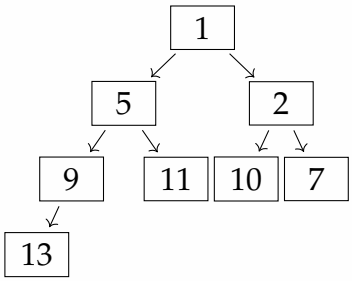
(b) in einen leeren Min-Heap ein, der bzgl. „$\leq$“ angeordnet ist und geben Sie den Heap nach „9“ und nach „5“ an

Lösungsvorschlag

Nach dem Einfügen von „9“:



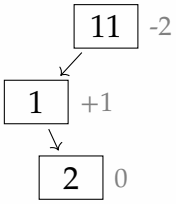
Nach dem Einfügen von „5“:



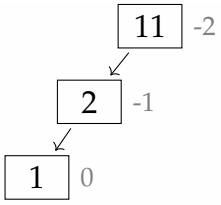
(c) in einen leeren AVL-Baum ein! Geben Sie den AVL Baum nach „2“ und „5“ an und beschreiben Sie die ggf. notwendigen Rotationen beim Einfügen dieser beiden Elemente!

Lösungsvorschlag

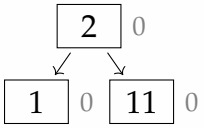
Nach dem Einfügen von „2“:



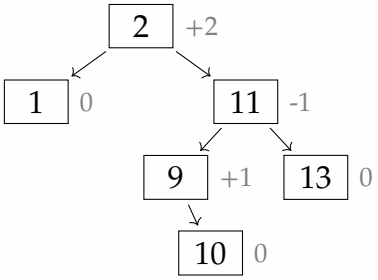
Nach der Linksrotation:



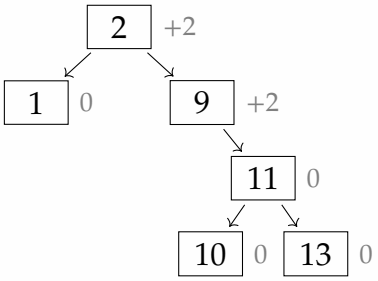
Nach der Rechtsrotation:



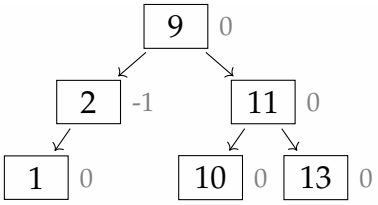
Nach dem Einfügen von „10“:



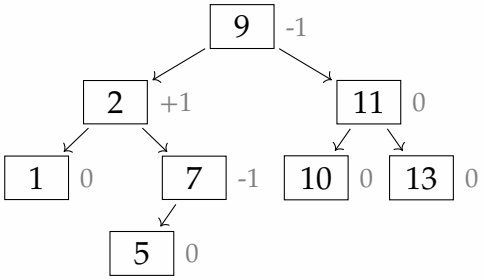
Nach der Rechtsrotation:



Nach der Linksrotation:



Nach dem Einfügen von „5“:



Staatsexamen 46115 / 2017 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 6

Aufgabe 6: [Halden - Heaps]

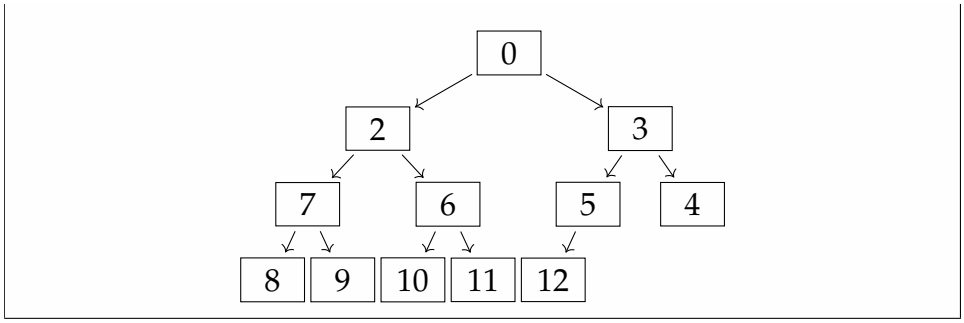
Gegeben sei folgende Feld-Einbettung (Array-Darstellung) einer Min-Halde:[4]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	2	3	7	6	5	4	8	9	10	11	12

(a) Stellen Sie die Halde graphisch als (links-vollständigen) Baum dar.

Lösungsvorschlag

[4] *Staatsexamen 46115 Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft) 2017 Herbst.*



(b) Entfernen Sie das kleinste Element (die Wurzel 0) aus der obigen initialen Halde, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	6	3	7	10	5	4	8	9	12	11

(c) Fügen Sie nun den Wert 1 in die obige initiale Halde ein, stellen Sie die Haldeneigenschaft wieder her und geben Sie nur das Endergebnis in Felddarstellung an.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	2	1	7	6	3	4	8	9	10	11	12	5

Staatsexamen 46115 / 2019 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 7

Aufgabe 7 (Heapify) [Heapify]

Schreiben Sie in Pseudocode eine Methode `heapify(int[] a)`, welche im übergebenen Array der Länge n die Heapeigenschaft in $\mathcal{O}(n)$ Schritten herstellt. D. h. als Ergebnis soll in a gelten, dass $a[i] \leq a[2i+1]$ und $a[i] \leq a[i+2]$.[5]

[5] **examen:46115:2019:09**.

```java
import org.bschlangaul.helfer.Konsole;

/**
 * Nach Pseudocode nach
 * https://www.oreilly.com/library/view/algorithms-in-
↪  a/9780596516246/ch04s06.html
 */
public class Heapify {

  public static void buildHeap(int a[]) {
    int n = a.length;
    for (int i = n / 2 - 1; i >= 0; i--) {
      heapify(a, i, n);
    }
  }

  public static void heapify(int a[], int index, int max) {
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    int smallest;

    if (left < max && a[left] < a[index]) {
      smallest = left;
    } else {
      smallest = index;
    }

    if (right < max && a[right] < a[smallest]) {
      smallest = right;
    }

    if (smallest != index) {
      int tmp = a[index];
      a[index] = a[smallest];
      a[smallest] = tmp;
      heapify(a, smallest, max);
    }
  }

  public static void main(String[] args) {
    int[] a = new int[] { 5, 3, 16, 2, 10, 14 };
    buildHeap(a);
    Konsole.zeigeZahlenFeld(a); // 2 3 14 5 10 16
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2019/herbst/Heapify.java`

Staatsexamen 46115 / 2020 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 7

Aufgabe 7 [Heaps]

Sei H ein Max-Heap, der n Elemente speichert. Für ein Element v in H sei $h(v)$ die Höhe von v, also die Länge eines längsten Pfades von v zu einem Blatt im Teilheap mit Wurzel v.[6]

(a) Geben Sie eine rekursive Definition von $h(v)$ an, in der Sie sich auf die Höhen der Kinder v.left und v.right von v beziehen (falls v Kinder hat).

(b) Geben Sie eine möglichst niedrige obere asymptotische Schranke für die Summe der Höhen aller Elemente in H an, also für $\sum_{v \in H} h(v)$ und begründen Sie diese.

Tipp: Denken Sie daran, wie man aus einem beliebigen Feld einen Max-Heap macht.

(c) Sei H' ein Feld der Länge n. Geben Sie einen Algorithmus an, der in Linearzeit testet, ob H ein Max-Heap ist.

Staatsexamen 66115 / 2012 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 7

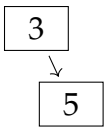
Aufgabe 7 [3,5,1,2,4 in leerer Suchbaum und Heap]

(a) Fügen Sie nacheinander die Zahlen $3, 5, 1, 2, 4$[7]

(i) in einen leeren binären Suchbaum ein

Nach dem Einfügen von „3“:

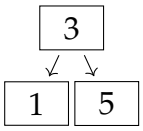
Nach dem Einfügen von „5“:



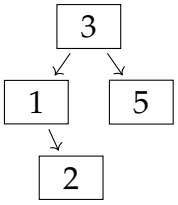
[6] **examen:46115:2020:03**.

[7] *Staatsexamen 66115 Theoretische Informatik / Algorithmen (vertieft) 2012 Herbst*.

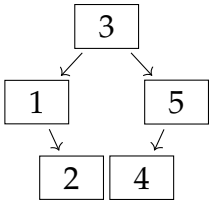
Nach dem Einfügen von „1“:



Nach dem Einfügen von „2“:

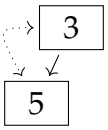


Nach dem Einfügen von „4“:

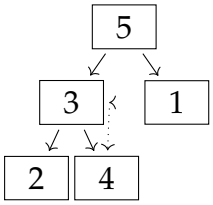


(ii) in einen leeren Heap ein

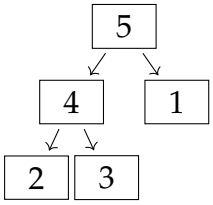
Erstellen einer Max.-Halde, einfügen von 3 und 5, Versickern notwendig:



Einfügen von 1 und 2 ohne Änderungen, Einfügen von 4, versickern notwendig:



Fertiger Heap:



8

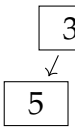
Ausführlicher als Max-Halde

Nach dem Einfügen von „3“:

0
3

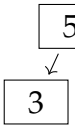
Nach dem Einfügen von „5“:

0	**1**
3	5



Nach dem Vertauschen von „5“ und „3“:

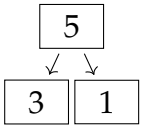
0	**1**
5	3



Nach dem Einfügen von „1“:

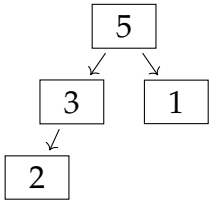
0	**1**	**2**
5	3	1

[8] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen: Aufgabenblatt 7: Wiederholung*, Zeichnen der Heap).



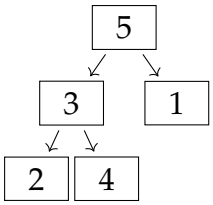
Nach dem Einfügen von „2“:

0	**1**	**2**	**3**
5	3	1	2



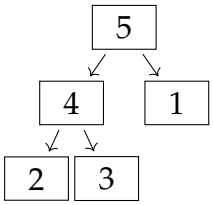
Nach dem Einfügen von „4“:

0	**1**	**2**	**3**	**4**
5	3	1	2	4



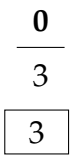
Nach dem Vertauschen von „4“ und „3“:

0	**1**	**2**	**3**	**4**
5	4	1	2	3

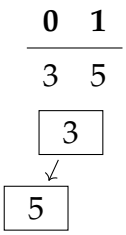


Ausführlicher als Min-Halde

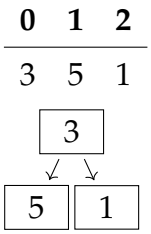
Nach dem Einfügen von „3“:



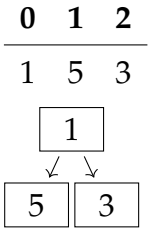
Nach dem Einfügen von „5“:



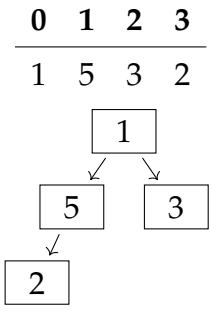
Nach dem Einfügen von „1“:



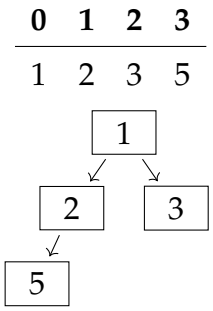
Nach dem Vertauschen von „1“ und „3“:



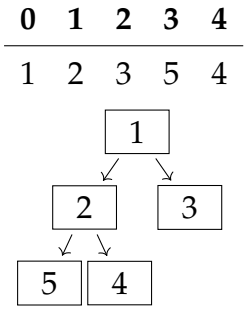
Nach dem Einfügen von „2“:



Nach dem Vertauschen von „2“ und „5“:



Nach dem Einfügen von „4“:



Geben Sie die Ergebnisse an (Zeichnung)

(b) Geben Sie zwei Merkmale an, bei denen sich Heaps und binäre Suchbäume wesentlich unterscheiden. Ein wesentlicher Unterschied zwischen Bubblesort und Mergesort ist z. B. die *worst case* Laufzeit mit $\mathcal{O}(n^2)$ für Bubblesort und $\mathcal{O}(n \log n)$ für Mergesort.

Lösungsvorschlag

	Binärer Suchbaum	Heap
Suchen beliebiger Wert (worst case)	$\mathcal{O}(\log(n))$	$\mathcal{O}(n)$
Suchen Min-Max (average case)	$\mathcal{O}(\log(n))$	$\mathcal{O}(1)$

a

[a] `https://cs.stackexchange.com/q/27860`

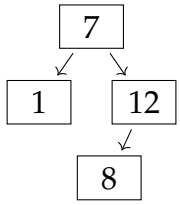
Staatsexamen 66115 / 2013 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 7

Aufgabe 7 [Heap und binärer Suchbaum]

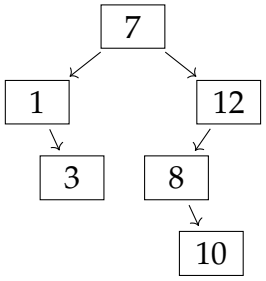
(a)

(i) Fügen Sie nacheinander die Zahlen 7, 1, 12, 8, 10, 3, 5 in einen leeren binären Suchbaum ein und zeichnen Sie den Suchbaum nach „8“ und nach „3“.[9]

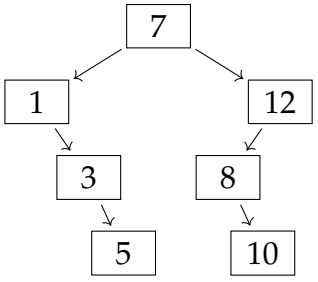
Nach dem Einfügen von „8“:



Nach dem Einfügen von „3“:



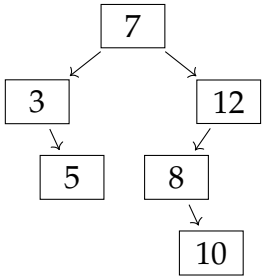
Nach dem Einfügen von „5“:



(ii) Löschen Sie die „1“ aus dem in (i) erstellten Suchbaum und zeichnen Sie den Suchbaum.

Nach dem Löschen von „1“:

[9] **examen:66115:2013:09**.



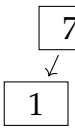
(iii) Fügen Sie 7, 1, 12, 8, 10, 3, 5 in einen leeren MIN-Heap ein, der bzgl. „$\leq$“ angeordnet ist. Geben Sie den Heap nach jedem Element an.

Nach dem Einfügen von „7“:

0
7

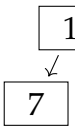
Nach dem Einfügen von „1“:

0	**1**
7	1



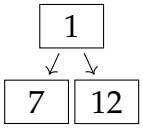
Nach dem Vertauschen von „1“ und „7“:

0	**1**
1	7



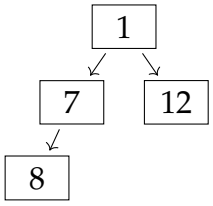
Nach dem Einfügen von „12“:

0	**1**	**2**
1	7	12



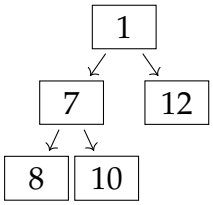
Nach dem Einfügen von „8“:

0	**1**	**2**	**3**
1	7	12	8



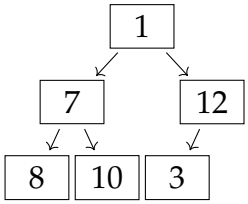
Nach dem Einfügen von „10“:

0	**1**	**2**	**3**	**4**
1	7	12	8	10



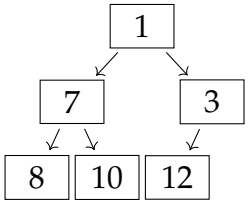
Nach dem Einfügen von „3“:

0	**1**	**2**	**3**	**4**	**5**
1	7	12	8	10	3



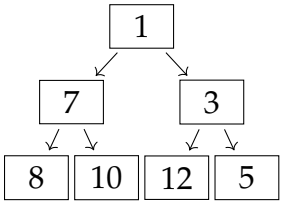
Nach dem Vertauschen von „3“ und „12“:

0	**1**	**2**	**3**	**4**	**5**
1	7	3	8	10	12



Nach dem Einfügen von „5“:

0	**1**	**2**	**3**	**4**	**5**	**6**
1	7	3	8	10	12	5



(b) Was ist die worst-case Laufzeit in O-Notation für das Einfügen eines Elements in einen Heap der Größe n? Begründen Sie ihre Antwort.

Lösungsvorschlag

Die worst-case Laufzeit berechnet sich aus dem Aufwand für das Durchsickern eines eingefügten Elementes. Da das Durchsickern entlang eines Pfades im Baum erfolgt, entspricht der Aufwand im ungünstigsten Fall der Höhe des Baumes, $\eth\log_2 n$. Insgesamt ergibt sich somit eine worst-case Laufzeit von $\mathcal{O}(\log n)$.[a]

[a] Saake und Sattler, *Algorithmen und Datenstrukturen*, Seite 413.

Staatsexamen 66115 / 2014 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 6

Aufgabe 6 [Selectionsort]

Gegeben[10]sei ein einfacher Sortieralgorithmus, der ein gegebenes Feld A dadurch sortiert, dass er das *Minimum* m von A *findet*, dann das Minimum von A ohne das Element m usw.[11]

[10] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen: Aufgabenblatt 3: Algorithmenmuster*, Seite 4, Aufgabe 4.

[11] *Staatsexamen 66115 Theoretische Informatik / Algorithmen (vertieft) 2014 Herbst*, Thema 2 Aufgabe 6 Seite 5.

(a) Geben Sie den Algorithmus in Java an. Implementieren Sie den Algorithmus *in situ*, ðso, dass er außer dem Eingabefeld nur konstanten Extraspeicher benötigt. Es steht eine Testklasse zur Verfügung.

Lösungsvorschlag

```java
public class SortierungDurchAuswaehlen {
  static void vertausche(int[] zahlen, int index1, int index2) {
    int tmp = zahlen[index1];
    zahlen[index1] = zahlen[index2];
    zahlen[index2] = tmp;
  }

  static void sortiereDurchAuswählen(int[] zahlen) {
    // Am Anfang ist die Markierung das erste Element im
    ↪  Zahlen-Array.
    int markierung = 0;
    while (markierung < zahlen.length) {
      // Bestimme das kleinste Element.
      // 'min' ist der Index des kleinsten Elements.
      // Am Anfang auf das letzte Element setzen.
      int min = zahlen.length - 1;
      // Wir müssen nicht bis letzten Index gehen, da wir 'min'
      ↪  auf das letzte Element
      // setzen.
      for (int i = markierung; i < zahlen.length - 1; i++) {
        if (zahlen[i] < zahlen[min]) {
          min = i;
        }
      }

      // Tausche zahlen[markierung] mit gefundenem Element.
      vertausche(zahlen, markierung, min);
      // Die Markierung um eins nach hinten verlegen.
      markierung++;
    }
  }

  public static void main(String[] args) {
    int[] zahlen = { 5, 2, 7, 1, 6, 3, 4 };
    sortiereDurchAuswählen(zahlen);
    for (int i = 0; i < zahlen.length; i++) {
      System.out.print(zahlen[i] + " ");
    }
  }
}
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/herbst/SortierungDurchAuswaehlen.java

(b) Analysieren Sie die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Beim ersten Durchlauf des *Selectionsort*-Algorithmus muss $n-1$ mal das Minimum durch Vergleich ermittel werden, beim zweiten Mal $n-2$. Mit Hilfe der *Gaußschen Summenformel* kann die Komplexität gerechnet werden:

$$(n-1)+(n-2)+\cdots+3+2+1 = \frac{(n-1)\cdot n}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{n^2}{2} \approx n^2$$

Da es bei der Berechnung des Komplexität um die Berechnung der asymptotischen oberen Grenze geht, können Konstanten und die Addition, Subtraktion, Multiplikation und Division mit Konstanten z. b. $\frac{n^2}{2}$ vernachlässigt werden.

Der *Selectionsort*-Algorithmus hat deshalb die Komplexität $\mathcal{O}(n^2)$, er ist von der Ordnung $\mathcal{O}(n^2)$.

(c) Geben Sie eine Datenstruktur an, mit der Sie Ihren Algorithmus beschleunigen können.

Lösungsvorschlag

Der *Selectionsort*-Algorithmus kann mit einer Min- (in diesem Fall) bzw. einer Max-Heap beschleunigt werden. Mit Hilfe dieser Datenstruktur kann sehr schnell das Minimum gefunden werden. So kann auf die vielen Vergleiche verzichtet werden. Die Komplexität ist dann $\mathcal{O}(n \log n)$.

Staatsexamen 66115 / 2016 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 7

Verständnis Suchbäume [Vergleich Suchbäume]

Wofür eignen sich die folgenden Baum-Datenstrukturen im Vergleich zu den anderen angeführten Baumstrukturen am besten, und warum. Sprechen Sie auch die Komplexität der wesentlichen Operationen und die Art der Speicherung an.[12]

[12] *Staatsexamen 66115 Theoretische Informatik / Algorithmen (vertieft) 2016 Frühjahr*, Seite 9.

(a) Rot-Schwarz-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(b) AVL-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log_2 n)$ (im Durchschnitt)
$\mathcal{O}(\log_2 n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(c) Binärer-Heap

Verwendungszweck zum effizienten Sortieren von Elementen.
[a]

Einfügen (Zeitkomplexität)
$\mathcal{O}(1)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(n)$ (im Durchschnitt)
$\mathcal{O}(n)$ (im schlechtesten Fall) [b]

[a] deut. Wikipedia
[b] engl. Wikipedia

(d) B-Baum

Lösungsvorschlag

Einfügen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Löschen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall)

Suchen (Zeitkomplexität)
$\mathcal{O}(\log n)$ (im Durchschnitt)
$\mathcal{O}(\log n)$ (im schlechtesten Fall) [a]

[a] tutorialspoint.com

(e) R-Baum

Lösungsvorschlag

Verwendungszweck Ein R-Baum erlaubt die schnelle Suche in mehrdimensionalen ausgedehnten Objekten. [a]

Suchen (Zeitkomplexität)
$\mathcal{O}(\log_M n)$ (im Durchschnitt) [b]
$\mathcal{O}(n)$ (im schlechtesten Fall) [c]

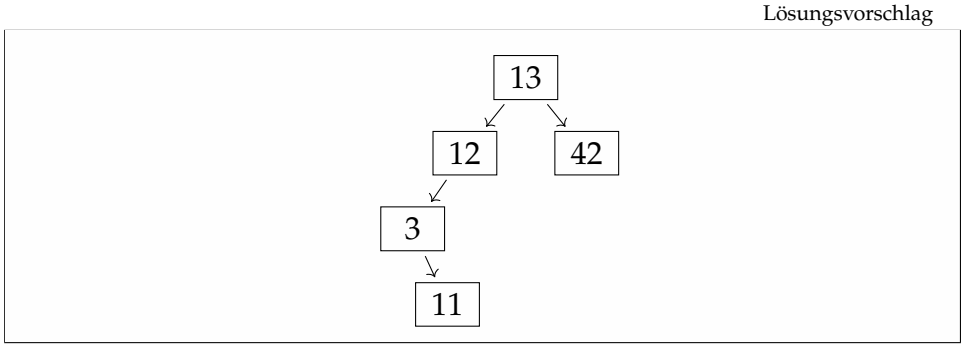
[a] deut. Wikipedia
[b] eng. Wikipedia
[c] Simon Fraser University, Burnaby, Kanada

Staatsexamen 66115 / 2017 / Frühjahr / Thema Nr. 2 / Aufgabe Nr. 8

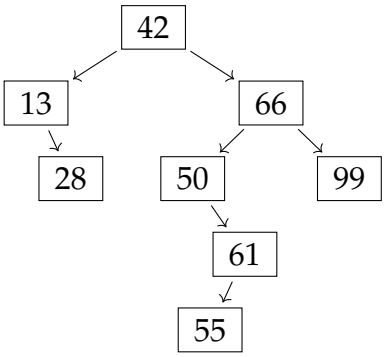
Aufgabe 8 [Binärbaum, Halde, AVL]

(a) Fügen[13] Sie die Zahlen 13, 12, 42, 3, 11 in der gegebenen Reihenfolge in einen zunächst leeren binären Suchbaum mit aufsteigender Sortierung ein. Stellen Sie nur das Endergebnis dar.[14]

Lösungsvorschlag



(b) Löschen Sie den Wurzelknoten mit Wert 42 aus dem folgenden *binären* Suchbaum mit aufsteigender Sortierung und ersetzen Sie ihn dabei durch einen geeigneten Wert aus dem *rechten* Teilbaum. Lassen Sie möglichst viele Teilbäume unverändert und erhalten Sie die Suchbaumeigenschaft.[15]

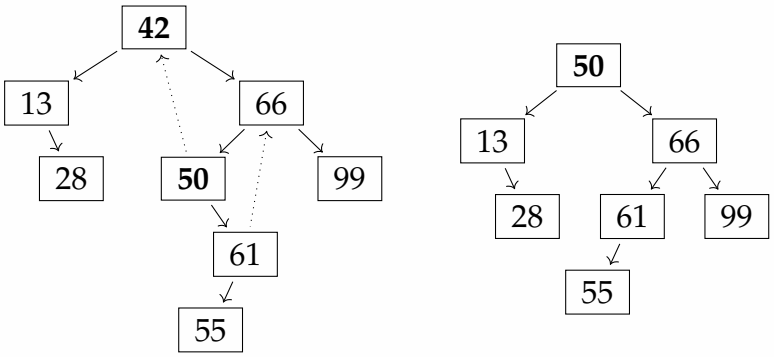


Lösungsvorschlag

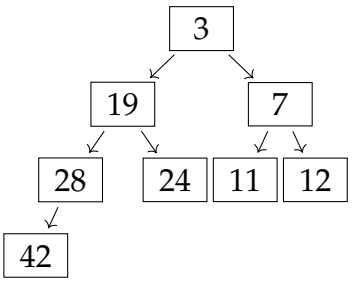
[13] *Staatsexamen 66115 Theoretische Informatik / Algorithmen (vertieft) 2017 Herbst*.

[14] *Qualifizierungsmaßnahme Informatik: Algorithmen und Datenstrukturen: Aufgabenblatt 5: Hashing, Bäume, Listen*, Aufgabe 2.

[15] *Staatsexamen 66115 Theoretische Informatik / Algorithmen (vertieft) 2017 Herbst*, Seite 12.

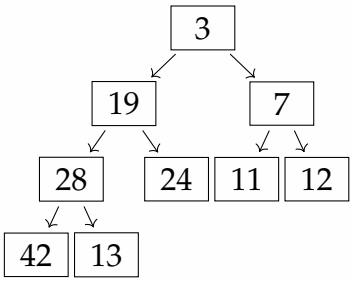


(c) Fügen Sie einen neuen Knoten mit dem Wert 13 in die folgende Min-Halde ein und stellen Sie anschließend die Halden-Eigenschaft vom neuen Blatt aus beginnend wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

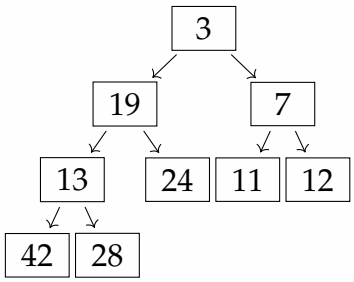


Lösungsvorschlag

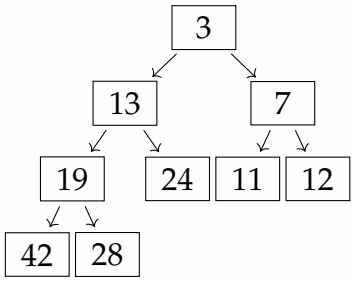
Nach dem Einfügen von „13“:



Nach dem Vertauschen von „13“ und „28“:



Nach dem Vertauschen von „13“ und „19“:

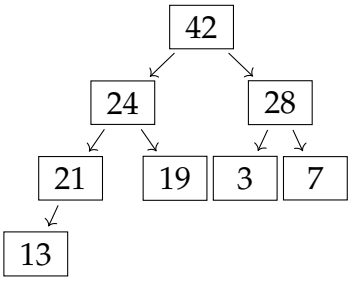


(d) Geben Sie für die ursprüngliche Min-Halde aus Teilaufgabe c) (ohne den neu eingefügten Knoten mit dem Wert 13) die Feld-Einbettung (Array-Darstellung) an.

Lösungsvorschlag

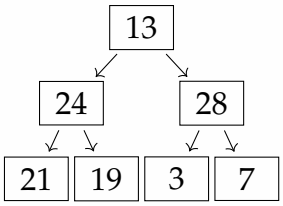
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	19	7	28	24	11	12	42

(e) Löschen Sie den Wurzelknoten mit Wert 42 aus der folgenden Max-Halde und stellen Sie anschließend die Halden-Eigenschaft ausgehend von einer neuen Wurzel wieder her, wobei möglichst viele Knoten der Halde unverändert bleiben und die Halde zu jedem Zeitpunkt links-vollständig sein soll. Geben Sie nur das Endergebnis an.

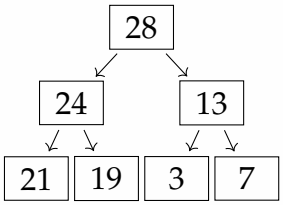


Lösungsvorschlag

Nach dem Ersetzen von „42“ mit „13“:

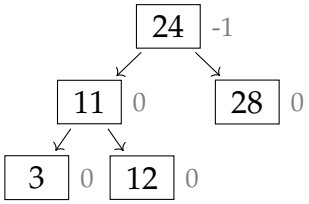


Nach dem Vertauschen von „13“ und „28“:



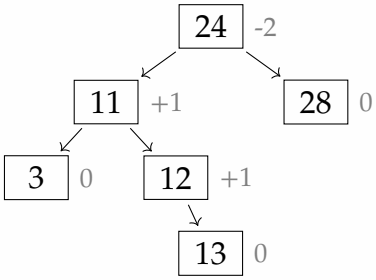
(f) Fügen Sie in jeden der folgenden AVL-Bäume mit aufsteigender Sortierung jeweils einen neuen Knoten mit dem Wert 13 ein und führen Sie anschließend bei Bedarf die erforderliche(n) Rotation(en) durch. Zeichnen Sie den Baum vor und nach den Rotationen.

(i) AVL-Baum A

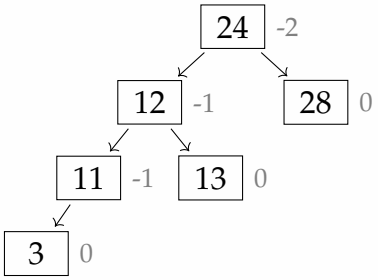


Lösungsvorschlag

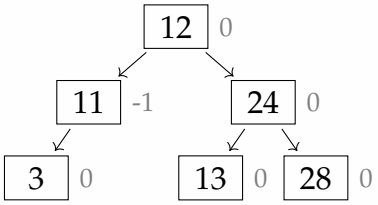
Nach dem Einfügen von „13“:



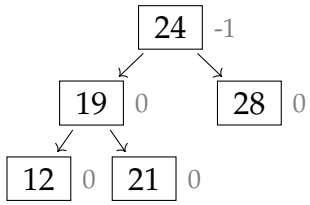
Nach der Linksrotation:



Nach der Rechtsrotation:

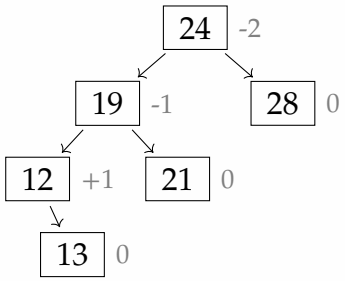


(ii) AVL-Baum B

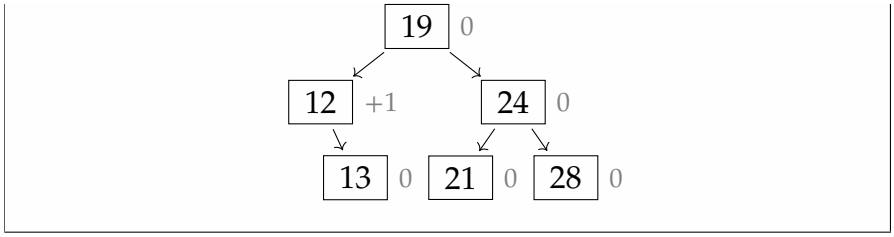


Lösungsvorschlag

Nach dem Einfügen von „13“:



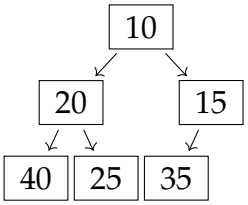
Nach der Rechtsrotation:



Staatsexamen 66115 / 2020 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 2 / Aufgabe Nr. 3

Aufgabe 3 [DeleteMin]

Es sei der folgende Min-Heap gegeben:[16]



(a) Geben Sie obigen Min-Heap in der Darstellung eines Feldes an, wobei die Knoten in Level-Order abgelegt sind.

Lösungsvorschlag

0	**1**	**2**	**3**	**4**	**5**
10	20	15	40	25	35

(b) Führen Sie wiederholt DeleteMin-Operationen auf dem gegebenen Heap aus, bis der Heap leer ist. Zeichnen Sie dafür den aktuellen Zustand des Heaps als Baum und als Feld nach jeder Änderung des Heaps, wobei Sie nur gültige Bäume zeichnen (d. h. solche die keine Lücken haben). Dokumentieren Sie, was in den einzelnen Schritten geschieht.

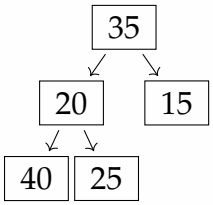
Lösungsvorschlag

Löschen von 10

Nach dem Ersetzen von „10“ durch „35“:

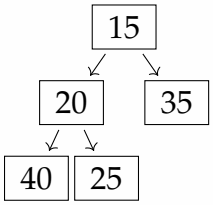
[16] **examen:66115:2020:09**.

0	1	2	3	4
35	20	15	40	25



Nach dem Vertauschen von „35“ und „15“:

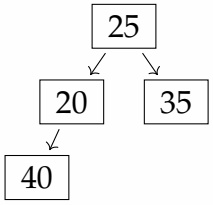
0	1	2	3	4
15	20	35	40	25



Löschen von 15

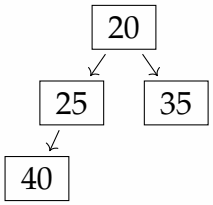
Nach dem Ersetzen von „15“ mit „25“:

0	1	2	3
25	20	35	40



Nach dem Vertauschen von „25“ und „20“:

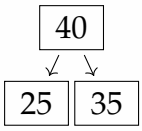
0	1	2	3
20	25	35	40



Löschen von 20

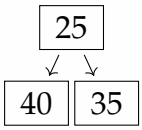
Nach dem Vertauschen von „20“ mit „40“:

0	1	2
40	25	35



Nach dem Vertauschen von „40“ und „25“:

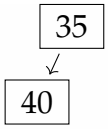
0	1	2
25	40	35



Löschen von 25

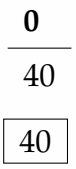
Nach dem Ersetzen von „25“ durch „35“:

0	1
35	40



Löschen von 35

Nach dem Ersetzen von „35“ mit „40“:



Löschen von 40

Nach dem Löschen von „40“: