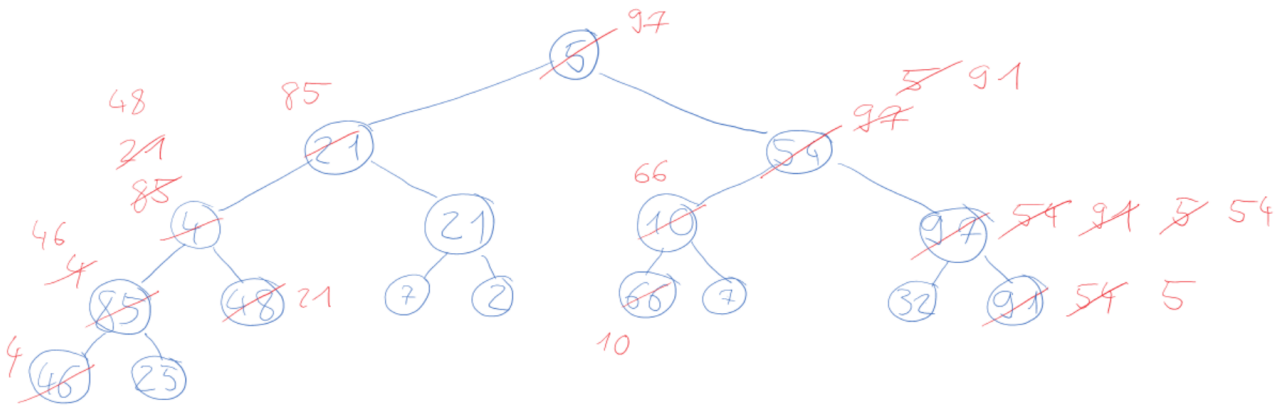
Lösungen Trainingsaufgaben Heap & HeapSort

1. Heap-Aufbau nach Floyd

Das abgebildete Array soll durch Anwendung des „Algorithmus von Floyd“ zu einem gültigen Max-Heap aufgebaut werden. Verwenden Sie das abgebildete Array direkt als Heap-Array und führen Sie darauf den Algorithmus auf. Sie können dies entweder direkt im Array erledigen oder den Heap zur Hilfe als Baum-struktur aufzeichnen. Zeigen Sie nach erfolgreichem Max-Heap Aufbau nochmals, wie die Elemente nun im Heap-Array angeordnet sind.

5	21	54	4	21	10	97	85	48	7	2	66	7	32	91	46	25

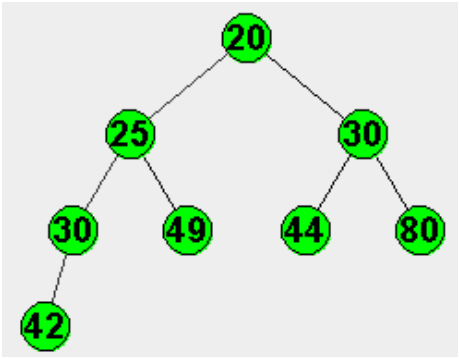


Max-Heap nach erfolgreichem Aufbau mithilfe des „*Algorithmus von Floyd*“:

97	85	91	48	21	66	54	46	21	7	2	10	7	32	5	4	25

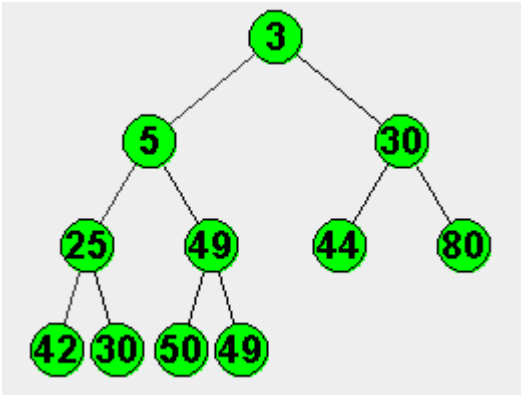
2. Operationen auf einem bestehenden Heap

Führen Sie auf dem abgebildeten Min-Heap die folgenden Operationen in gegebener Reihenfolge aus:



`removeMin(), add(1), add(5), add(3), add(49), add(50), add(-1), removeMin(), removeMin()`

Heap nach allen ausgeführten Operationen:



3. HeapSort: Baum Zeichnung

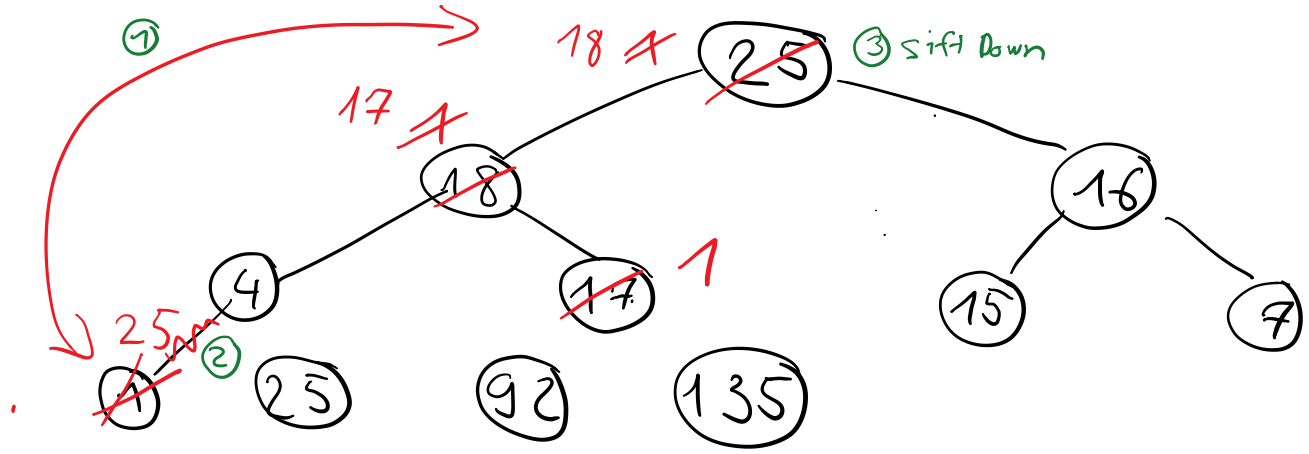
Gegeben sei ein Array, welches in einen „Heap-Bereich“ (Max-Heap) und einen „sortierten Bereich“ unter-teilt ist. Es wurden bereits einige Sortier-Iterationen ausgeführt. Geben Sie an, wie viele Elemente der „sortierte Bereich“ umfasst.

Array:

25	18	16	4	17	15	7	1	25	92	135

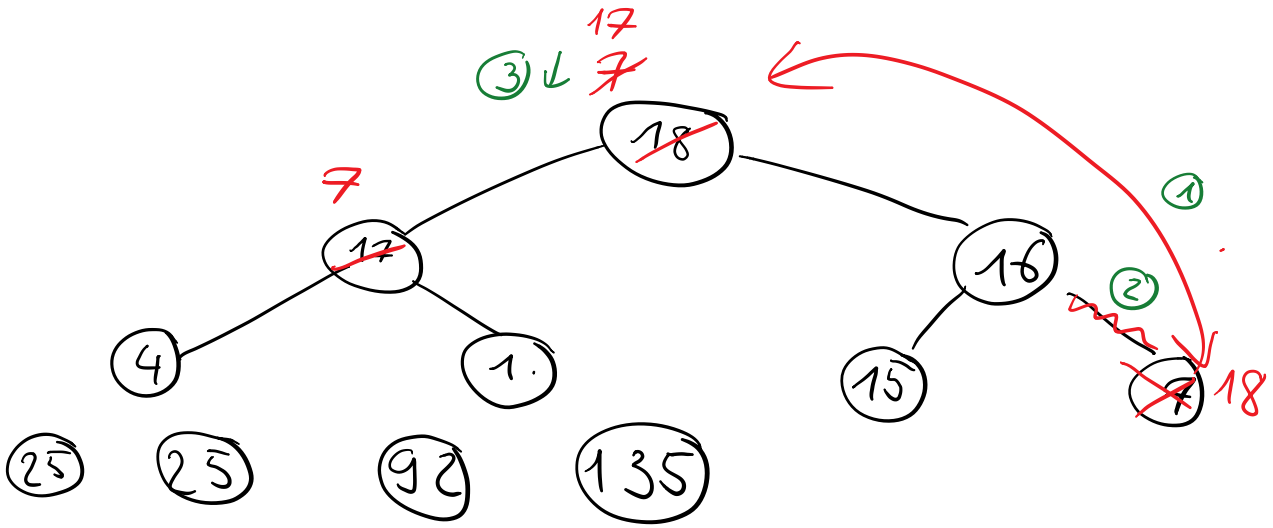
Die letzten drei Elemente sind bereits sortiert.

Zeichnen Sie den Heap als Baumstruktur. Sortieren Sie nun die restlichen Elemente Schritt für Schritt mit dem HeapSort Algorithmus.



Nach diesem Sortier-Schritt sieht das Array wie folgt aus (der sortierte Bereich ist grün hinterlegt):

18	17	16	4	1	15	7	25	25	92	135



Nach diesem Sortier-Schritt sieht das Array wie folgt aus (der sortierte Bereich ist grün hinterlegt):

17	7	16	4	1	15	18	25	25	92	135

Aus Platzgründen werden hier jetzt nicht alle Sortier-Schritte im Detail aufgezeichnet.

Nach allen Sortier-Schritten haben die Elemente im Array die folgende Reihenfolge:

1	4	7	15	16	17	18	25	25	92	135

4. HeapSort: Array

Gegeben ist ein vollständig aufgebauter Min-Heap. Führen Sie den HeapSort Algorithmus direkt auf dem Array aus, so dass die Elemente schlussendlich in absteigender Reihenfolge sortiert sind. Beschriften Sie dazu zuerst die Array-Indizes. Wählen Sie als Index für das erste Element 0 oder 1, je nachdem, wel-che Heap-Implementation Sie bevorzugen.

12	20	35	42	50	37	39

Lösung (grün ist jeweils der bereits sortierte Bereich)

Nach 1. Iteration

20	39	35	42	50	37	12

Nach 2. Iteration

35	39	37	42	50	20	12

Nach 3. Iteration

37	39	50	42	35	20	12

Nach 4. Iteration

39	42	50	37	35	20	12

Nach 5. Iteration

42	50	39	37	35	20	12

Nach 6. Iteration

50	42	39	37	35	20	12

Nach 7. Iteration

50	42	39	37	35	20	12

5. HeapSort: Array

Betrachten Sie Ihre (oder unsere) Lösung der Selbststudiums-Aufgabe zu HeapSort mit dem SortDemo-Framework. Wo müssen Sie überall etwas ändern – und was, damit das Programm absteigend und nicht aufsteigend sortiert?

```
public class HeapSort implements SortAlg {
   @Override
   public void run(SortData data) {
      int size = data.size();
      for (int i = size / 2 - 1; i >= 0; i--) {
         siftDown(data, i, size);
      }
      for (int i = size - 1; i > 0; i--) {
         data.swap(i, 0);
         siftDown(data, 0, i);
      }
```

```
    }

    private void siftDown(SortData data, int i, int size) {
       int j = 2 * i + 1;
       if (j + 1 < size && data.less(j, j + 1)) j++;          data.less(j + 1, j)
       while (j < size && data.less(i, j)) {                  data.less(j, i)
          data.swap(i, j);
          i = j;
          j = 2 * i + 1;
          if (j + 1 < size && data.less(j, j + 1)) j++;       data.less(j + 1, j)
       }
    }
 }
```