

Serie 3**1. Min-Heap: HEAP-EXTRACT-MIN**

Min-Heap: $A[\text{Parent}(i)] \leq A[i]$ (Parent-Element ist immer kleiner oder gleich)

2	11	7	15	22	9	18	15	27	31
---	----	---	----	----	---	----	----	----	----

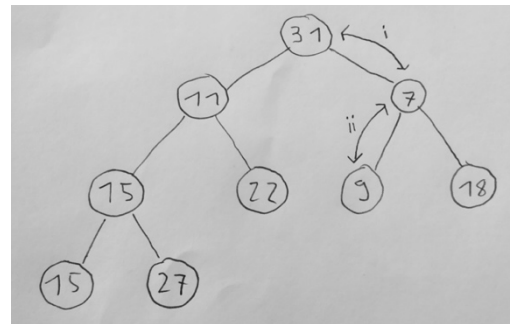
Wurzel-Element wird mit letztem Element vertauscht und der Heap um 1 Element verkleinert

31	11	7	15	22	9	18	15	27
----	----	---	----	----	---	----	----	----

Min-Heap Eigenschaft wiederherstellen

ergibt:

7	11	9	15	22	31	18	15	27
---	----	---	----	----	----	----	----	----

**2. Teilbäume von Max-Heaps**

Sei n = Höhe eines Max-Heap Baums

Für $n = 1$: $A[\text{parent}(1)] \geq A[1]$ stimmt.

Annahme: $A[\text{parent}(n)] \geq A[n]$.

Schritt: $n \rightarrow n + 1$: $A[\text{parent}(n+1)] \geq A[n]$

Da die maximale tiefe eines Unterbaumes jedoch immer um 1 kleiner ist, gilt $A[\text{parent}(n)] \geq A[n-1]$ und da $A[n] \geq A[n-1]$ folgt die Behauptung aus der Annahme.

3. Max-Heap

Max-Heap: $A[\text{Parent}(i)] \geq A[i]$ (Parent-Element ist immer grösser oder gleich)

23	17	14	6	13	10	1	5	7	12
----	----	----	---	----	----	---	---	---	----

$23 \geq 17$ und $23 \geq 14$ true

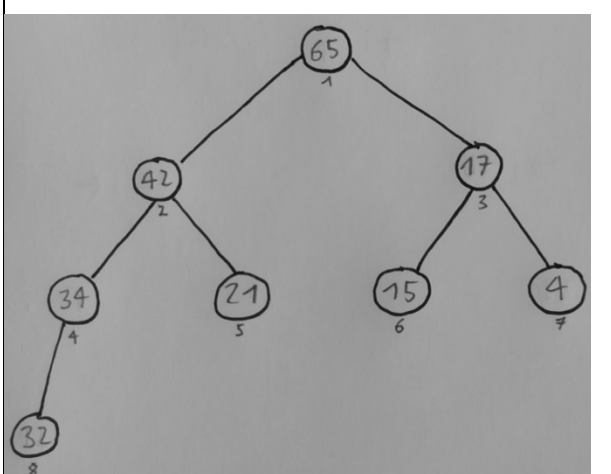
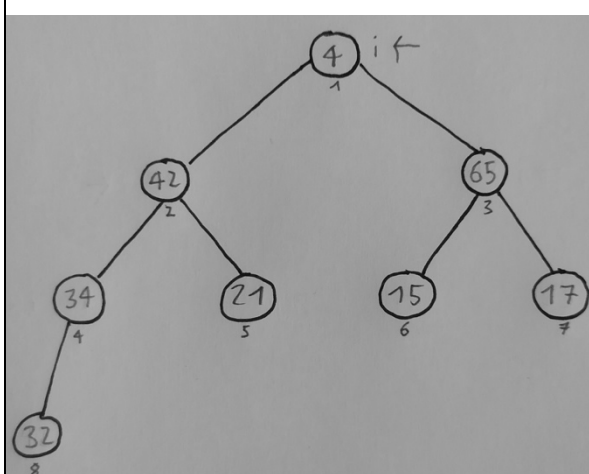
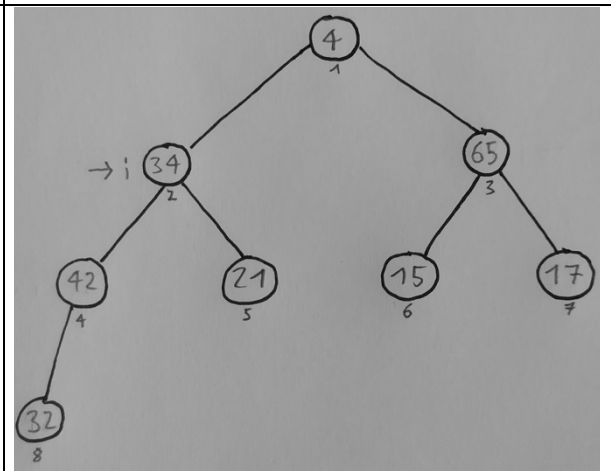
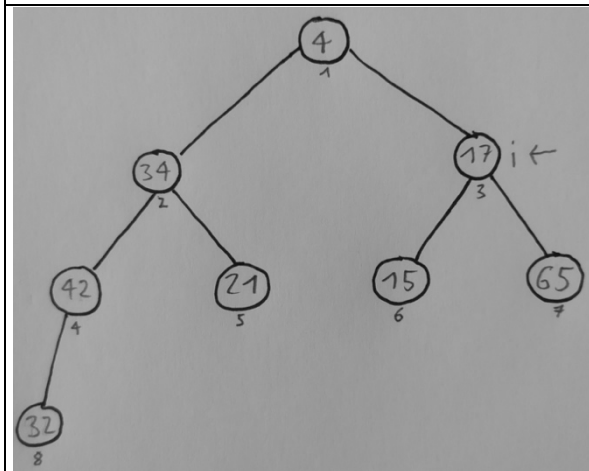
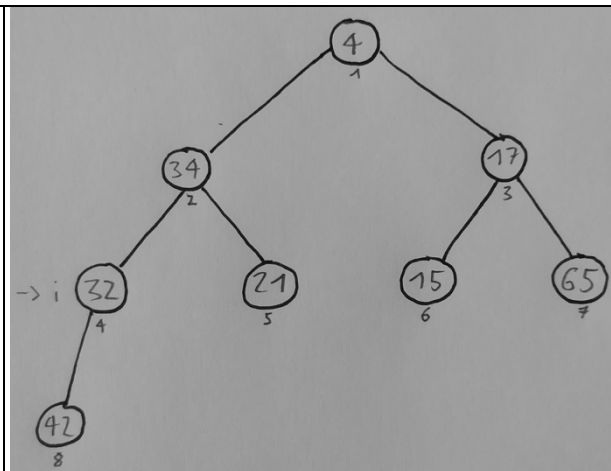
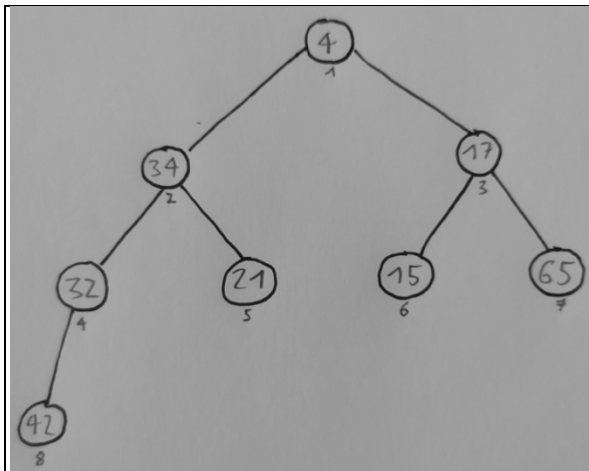
$17 \geq 6$ und $17 \geq 13$ true

$14 \geq 10$ und $14 \geq 1$ true

$6 \geq 5$ aber nicht $6 \geq 7 \rightarrow$ Widerspruch, es handelt sich nicht um ein Max-Heap

4. BUILD-MAX-HEAP

4	34	17	32	21	15	65	42
---	----	----	----	----	----	----	----



➔

65	42	17	34	21	15	4	32
----	----	----	----	----	----	---	----

5. Laufzeit von HEAPSORT

In aufsteigender Reihenfolge sortiert

Heap Konstruktion: Build Max-Heap $\rightarrow O(n)$

Aufrechterhaltung der Heap Eigenschaft:

- das Element an Stelle 1 wird mit dem letzten Element des Heaps vertauscht
 - der Heap wird um 1 Element gekürzt
 - das kleinste Element, dass nun anstelle 1 ist «rutscht» wieder herunter im Baum (Max-Heapify(A,1))
- ➔ das ganze wird (n-1) mal wiederholt: $O(n \lg n)$

Total (Konstruktion + Aufrechterhaltung): $O(n) + O(n \lg n) = O(n \lg n)$

In absteigender Reihenfolge sortiert

Heap Konstruktion: ist bereits ein Max-Heap

Aufrechterhaltung der Heap Eigenschaft:

- ➔ Gleicher Ablauf wie oben: $O(n \lg n)$

Total (Konstruktion + Aufrechterhaltung): $O(n \lg n)$

6. Quicksort

```

Quicksort(A, 1, 9)
  ( Quicksort(A, 1, 0) )
  Quicksort(A, 2, 9)
    Quicksort(A, 2, 3)
      ( Quicksort (A,2, 1) )
      Quicksort(A, 3, 3)
    Quicksort(A, 5, 9)
      Quicksort(A, 5, 5)
      Quicksort(A, 7, 9)
        Quicksort(A, 7, 7)
        Quicksort(A, 9, 9)

```

	1	2	3	4	5	6	7	8	9	
A =	19	17	3	28	60	33	20	30	2	Partition
Partition (A,1,9)	19	17	3	28	60	33	20	30	2	pivot Element x
	19	17	3	28	60	33	20	30	2	i Block < x
	2	17	3	28	60	33	20	30	19	j Block > x
Partition (A,2,9)	2	17	3	28	60	33	20	30	19	fixes Element
	2	17	3	28	60	33	20	30	19	
	2	17	3	19	60	33	20	30	28	
Partition (A,2,3)	2	17	3	19	60	33	20	30	28	
	2	17	3	19	60	33	20	30	28	
	2	3	17	19	60	33	20	30	28	
Partition (A,3,3)	2	3	17	19	60	33	20	30	28	
	2	3	17	19	60	33	20	30	28	
	2	3	17	19	60	33	20	30	28	
Partition (A,5,9)	2	3	17	19	60	33	20	30	28	
	2	3	17	19	20	33	60	30	28	
	2	3	17	19	20	28	60	30	33	
Partition (A,5,5)	2	3	17	19	20	28	60	30	33	
	2	3	17	19	20	28	60	30	33	
	2	3	17	19	20	28	60	30	33	
Partition (A,7,9)	2	3	17	19	20	28	60	30	33	
	2	3	17	19	20	28	30	60	33	
	2	3	17	19	20	28	30	33	60	
Partition (A,7,7)	2	3	17	19	20	28	30	33	60	
	2	3	17	19	20	28	30	33	60	
	2	3	17	19	20	28	30	33	60	
Partition (A,9,9)	2	3	17	19	20	28	30	33	60	
	2	3	17	19	20	28	30	33	60	
	2	3	17	19	20	28	30	33	60	

Praktische Aufgaben

Output

MatrikelNr Name Vorname

94419832 Briod Jean

99323462 Fischer Hans

96419642 Gans Gustav

92987654 Habegger Pascal

89345675 Moser Käthy

99876532 Müller Anton

90588921 Müller Kurt

98345632 Schneider Anna

98222634 Stucki Daniel

92123456 Wenger Thomas

StackOverflowError

Beim Quicksort wird ein Pivot Element ausgewählt und das Array dann bei diesem Pivot Element in zwei Arrays aufgeteilt. Alle Elemente im Array welche grösser sind als das Pivot Element kommen dann in das rechte und alle kleineren in das linke Array. Dieser Prozess wird rekursiv auf den Teil-Arrays wiederholt. Bei einem sortierten Array ist der beste Fall, wenn das Pivot Element genau in der Mitte des Arrays gewählt wird. Der schlechteste Fall bei einem sortierten Array ist, wenn es beim grössten oder kleinsten gewählt wird. Bei dieser Implementierung wird das rechteste Element im Array als erstes Pivot Element gewählt. Da das Array bereits sortiert ist, ist das auch das grösste Element und es tritt der zeitliche Worst-Case des Algorithmus ein. Der StackOverflowError entsteht, da bei jeder Rekursion die benötigten Variablen im Stack gespeichert werden und dieser nach genügend Rekursionen voll ist. Das kann auch eintreten, wenn eine Rekursive Methode keine Abbruchbedingung hat. Hier liegt es an der grossen Input Menge und der Wahl des Pivots.