Algorithmen und Datenstrukturen
3. Übungsserie

Florian Ludewig (185722)

25. Mai 2020

Aufgabe 1

1.a)

`STOOGE-SORT` funktioniert für Felder mit 0, 1 oder 2 Elementen:

- **0 Elemente**
  Das Feld ist bereits sortiert.
- **1 Element**
  Das Feld ist bereits sortiert.
- **2 Elemente**
  Sei `A=[x,y]`, dann ist `i=0` und `j=1`. Getauscht wird, wenn $y < x$, wodurch das Feld in jedem Fall korrekt sortiert ist.
  Außerdem ist $l = j - i + 1 = 1 - 0 + 1 = 2$ und $2 \not> 2$ wodurch der Algorithmus terminiert.

Angenommen `STOOGE-SORT` funktioniert für alle Felder der Länge $n - 1$ oder kleiner.
Betrachten eines Feldes `A` der Länge n. Der Algorithmus arbeitet drei Schritte ab:

1. `STOOGE-SORT(i,j-k)`
   Hier wird `STOOGE-SORT` rekursiv mit $i = 0$ und $j = (n-1) - \lfloor \frac{n}{3} \rfloor$ aufgerufen. Weil wir davon ausgehen, dass der Algorithmus für Felder mit kleineren Längen funktioniert, sind nach diesem rekursiven Aufruf alle Zahlen im ersten Drittel des Feldes $[0, ..., \lfloor \frac{n}{3} \rfloor]$ kleiner als im zweiten Drittel $[\lceil \frac{n}{3} \rceil, ..., n - \lfloor \frac{n}{3} \rfloor]$.
2. `STOOGE-SORT(i+k,j)`
   Nun wird `STOOGE-SORT` rekursiv mit $i = \lfloor \frac{n}{3} \rfloor$ und $j = n - 1$ aufgerufen. Danach sind mit Sicherheit alle Zahlen im zweiten Drittel des Feldes $[\lfloor \frac{n}{3} \rfloor, ..., n - \lfloor \frac{n}{3} \rfloor]$ kleiner als die Zahlen im letzten Drittel des Feldes $[n - \lceil \frac{n}{3} \rceil, ..., n-1]$.

Demnach befinden sich an diesem Punkt die größten Zahlen im letzten Drittel des Feldes.

3. `STOOGE-SORT(i,j-k)`
   Zum Schluss wird `STOOGE-SORT` nochmal rekursiv mit $i = 0$ und $j = (n - 1) - \lfloor \frac{n}{3} \rfloor$ aufgerufen, wodurch das Feld in den ersten beiden Dritteln sortiert wird. Weil die größten Zahlen sich bereits im letzten Drittel befinden ist das Feld nach diesem Schritt vollständig sortiert.

1.b)

`STOOGE-SORT` ist ein rekursiver Algorithmus und seine Laufzeit kann durch folgendes rekursives Schema beschrieben werden:

$$T(0) = 1, T(1) = 1, T(2) = 1$$

$$T(n) = 3 \cdot T(\frac{2}{3}n) + O(1) = 3 \cdot T(\frac{n}{1.5}) + O(1)$$

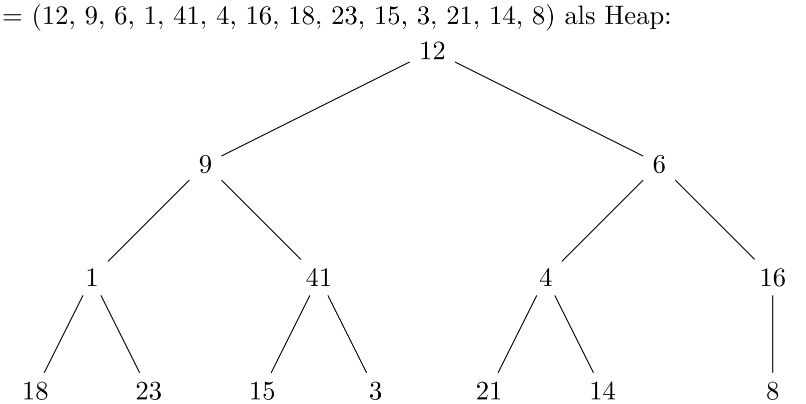
Weil der Algorithmus sich selbst **drei** mal Aufruft und jedes mal **zwei Drittel** des Feldes sortiert. Mithilfe des ersten Falles des Mastertheorems lässt sich dann die Laufzeit bestimmen:

$$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_{\frac{3}{2}} 3}) \approx \Theta(n^{2.7})$$

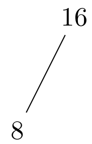
Aufgabe 2

2.a)

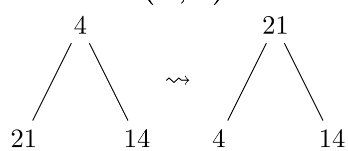
A = (12, 9, 6, 1, 41, 4, 16, 18, 23, 15, 3, 21, 14, 8) als Heap:



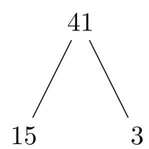
- **HEAPIFY(A, 7)**



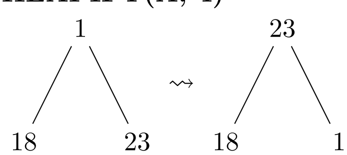
- **HEAPIFY(A, 6)**



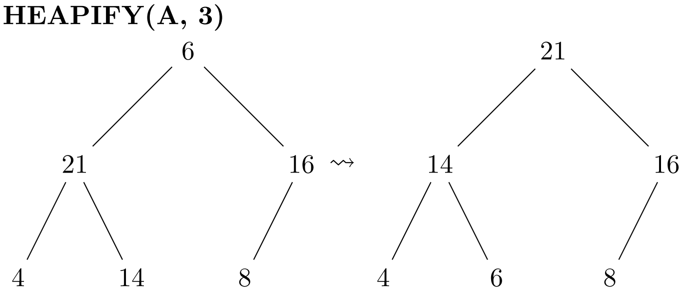
- **HEAPIFY(A, 5)**



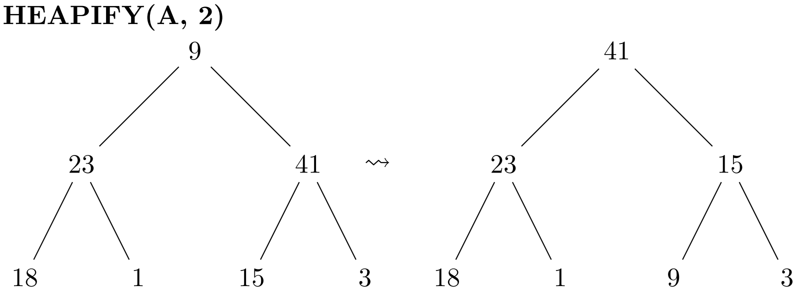
- **HEAPIFY(A, 4)**



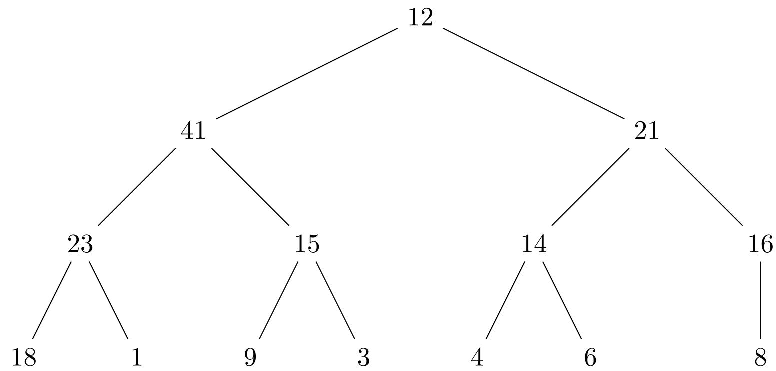
- **HEAPIFY(A, 3)**

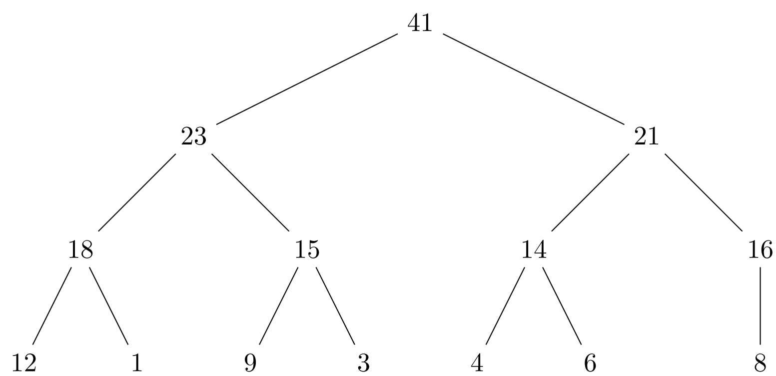


- **HEAPIFY(A, 2)**



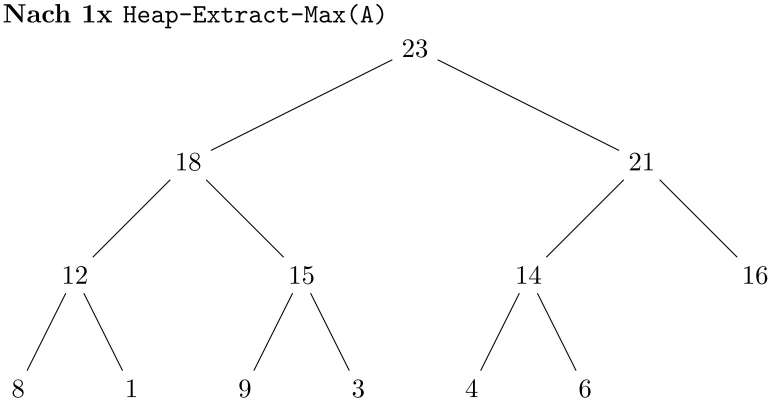
- **HEAPIFY(A, 1)**



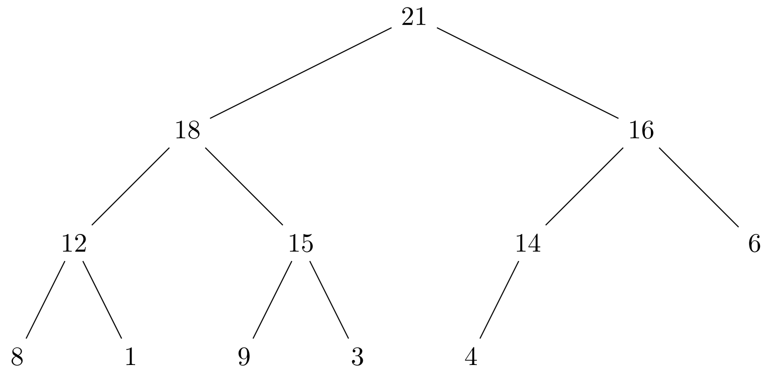


2.b)

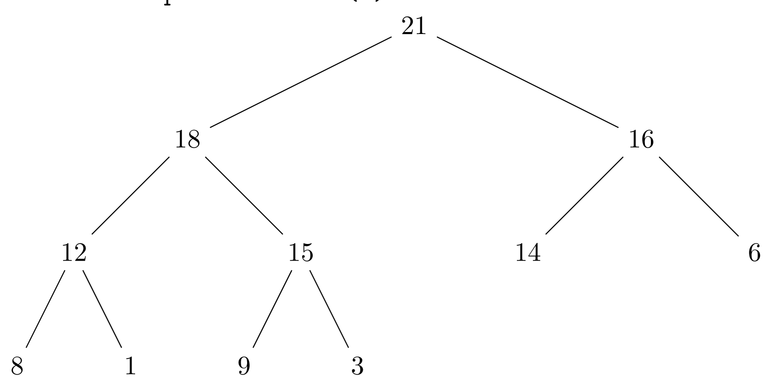
- **Nach 1x** `Heap-Extract-Max(A)`



- **Nach 2x** `Heap-Extract-Max(A)`



- **Nach 3x** `Heap-Extract-Max(A)`



Aufgabe 3

3.a)

3.b)

Aufgabe 4

4.a)

4.b)

4.c)

4.d)

4.e0)