Technische Universität München
Institut für Informatik I-16
Dr. Tobias Lasser
Jakob Vogel
Matthias Wieczorek

SoSe 2014
Lösungsblatt 5
28. Mai 2014

Algorithmen und Datenstrukturen

Aufgabe 1 Repräsentation von Graphen (Beispiellösung)

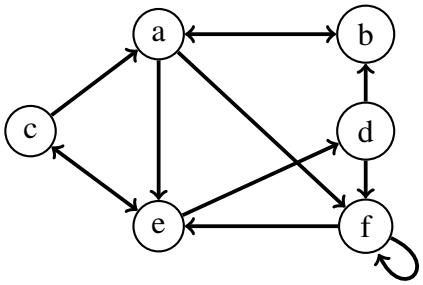
a) Adjazenz-Matrix:

	a	b	c	d	e
a	0	1	0	1	0
b	1	0	1	0	1
c	0	1	0	0	1
d	1	0	0	0	1
e	0	1	1	1	0

Adjazenz-Liste:

$$
\begin{array}{lllllll}
a & \longrightarrow & b & \rightarrow & d & & \\
\downarrow & & & & & & \\
b & \longrightarrow & a & \rightarrow & c & \rightarrow & e \\
\downarrow & & & & & & \\
c & \longrightarrow & b & \rightarrow & e & & \\
\downarrow & & & & & & \\
d & \longrightarrow & a & \rightarrow & e & & \\
\downarrow & & & & & & \\
e & \longrightarrow & b & \rightarrow & c & \rightarrow & d
\end{array}
$$

b) Gerichteter (!) Graph:



Adjazenz-Liste:

$$
\begin{array}{lllllll}
a & \longrightarrow & b & \rightarrow & e & \rightarrow & f \\
\downarrow & & & & & & \\
b & \longrightarrow & a & & & & \\
\downarrow & & & & & & \\
c & \longrightarrow & a & \rightarrow & e & & \\
\downarrow & & & & & & \\
d & \longrightarrow & b & \rightarrow & f & & \\
\downarrow & & & & & & \\
e & \longrightarrow & c & \rightarrow & d & & \\
\downarrow & & & & & & \\
f & \longrightarrow & e & \rightarrow & f & &
\end{array}
$$

Aufgabe 2 Arrays und Heaps (Beispiellösung)

a) Kein Heap; fast Min-Heap bis auf $7 \leftrightarrow 6$.

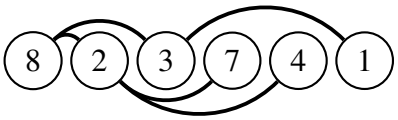
b) Min-Heap; Vaterknoten stets kleiner als Kindknoten.

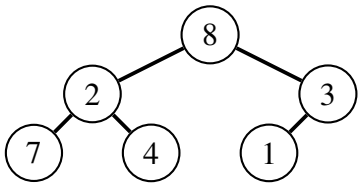
c) Kein Heap; beispielsweise Wurzel $4 < 5$ (linkes Kind), aber $4 > 3$ (rechtes Kind).

d) Max-Heap; Vaterknoten stets größer als Kindknoten.

Aufgabe 3 Heap-Sort (Beispiellösung)

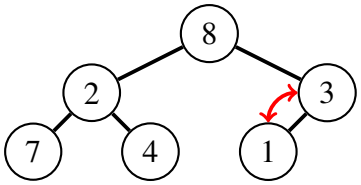
Array als fast vollständiger Binärbaum:

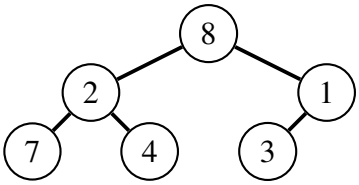




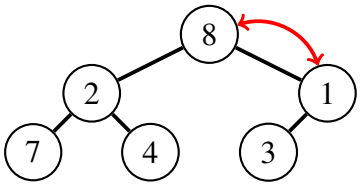
buildMinHeap: (**minHeapify** für alle Werte oder Positionen von hinten nach vorne)

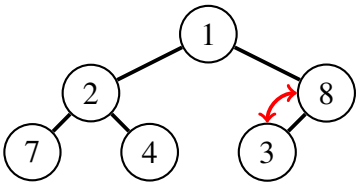
- **minHeapify**(1), **minHeapify**(4), **minHeapify**(7):
  $\rightsquigarrow$ trivial bzw. überflüssig, da Blätter
- **minHeapify**(3):

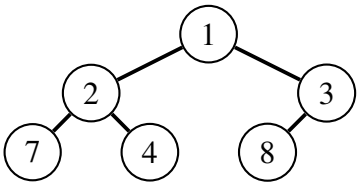




- **minHeapify**(2):
  $\rightsquigarrow$ nichts zu tun
- **minHeapify**(8):



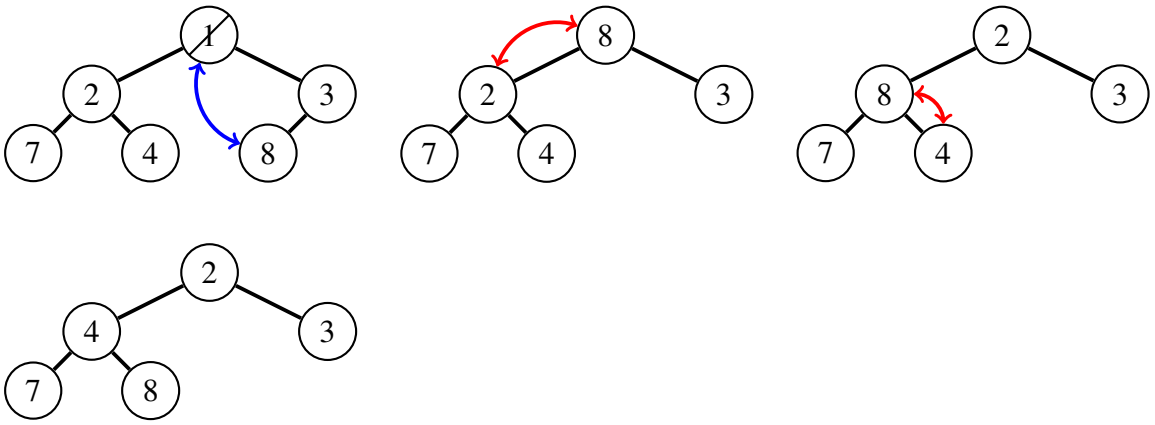




$\Rightarrow$ Heap nach **buildMinHeap**: $(1,2,3,7,4,8)$

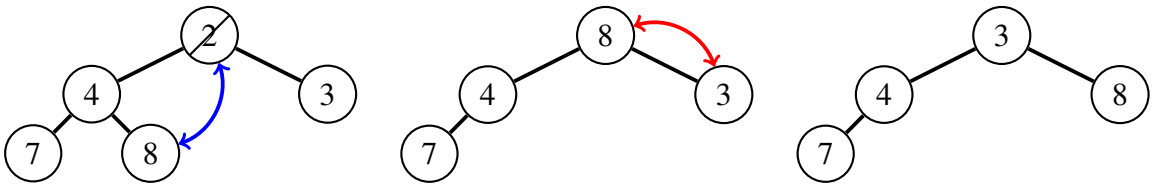
Sortierung: (**extractMin** bis Heap leer)

- **extractMin** $= 1$, dadurch 8 temporär neue Wurzel, folglich **minHeapify**(8)



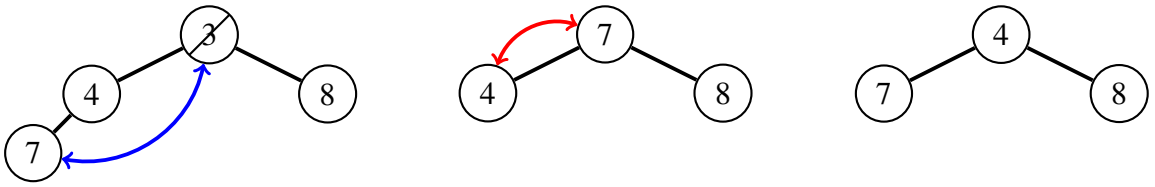
  $\Rightarrow$ Heap nach **extractMin**: $(2,4,3,7,8)$

- **extractMin** $= 2$, dadurch 8 temporär neue Wurzel, folglich **minHeapify**(8)



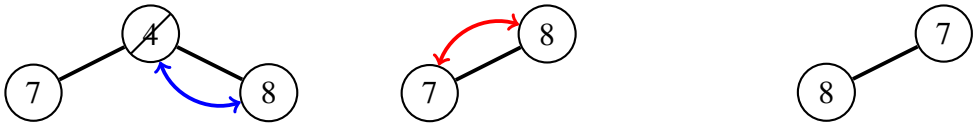
  $\Rightarrow$ Heap nach **extractMin**: $(3,4,8,7)$

- **extractMin** $= 3$, dadurch 7 temporär neue Wurzel, folglich **minHeapify**(7)



  $\Rightarrow$ Heap nach **extractMin**: $(4,7,8)$

- **extractMin** $= 4$, dadurch 8 temporär neue Wurzel, folglich **minHeapify**(8)



  $\Rightarrow$ Heap nach **extractMin**: $(7,8)$

- **extractMin** $= 7$, dadurch 8 neue Wurzel

  $\Rightarrow$ Heap nach **extractMin**: (8)

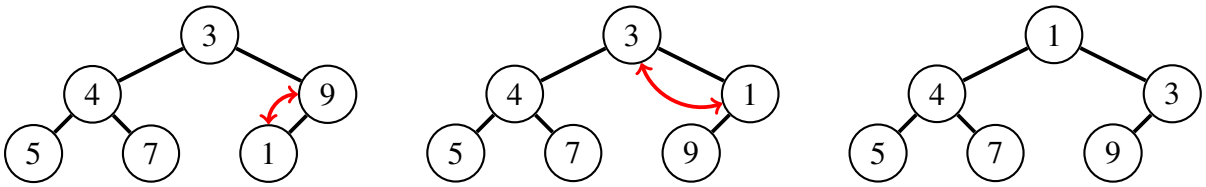
- **extractMin** $= 8$

  $\Rightarrow$ Heap nach **extractMin**: $\emptyset$

Aufgabe 4 Min-Priority Queue (Beispiellösung)

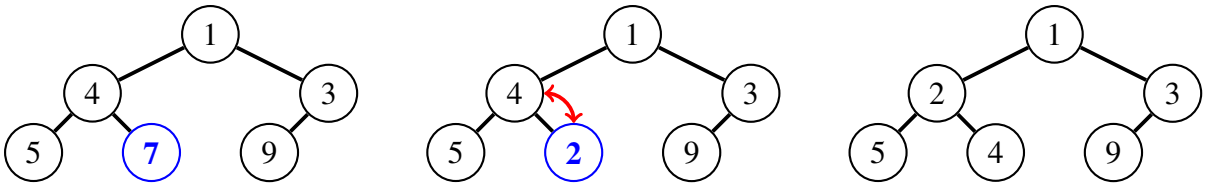
Array als fast vollständiger Binärbaum:

insert(1): (Einfügen am Ende, und Propagation aufwärts)



$\Rightarrow$ Heap nach **insert**(1): $(1,4,3,5,7,9)$

decreaseKey(7, 2): (Propagation aufwärts)



$\Rightarrow$ Heap nach **insert**(1) und **decreaseKey**(7, 2): $(1,2,3,5,4,9)$

Aufgabe 5 Baum-Traversierung (Beispiellösung)

a) Pre-Order $(1,2,4,7,5,8,3,6,9)$
In-Order $(7,4,2,8,5,1,6,9,3)$
Post-Order $(7,4,8,5,2,9,6,3,1)$

b) Pre-Order $(1,2,4,5,8,9,3,6,7)$
In-Order $(4,2,8,5,9,1,6,3,7)$
Post-Order $(4,8,9,5,2,6,7,3,1)$