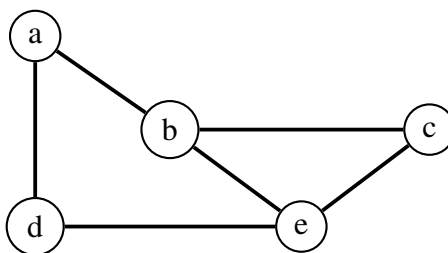


Algorithmen und Datenstrukturen

Aufgabe 1 Repräsentation von Graphen

a) Geben Sie Adjazenz-Matrix und Adjazenz-Liste an!



b) Skizzieren Sie den Graphen, und geben Sie die Adjazenz-Liste an!

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>a</i>	0	1	0	0	1	1
<i>b</i>	1	0	0	0	0	0
<i>c</i>	1	0	0	0	1	0
<i>d</i>	0	1	0	0	0	1
<i>e</i>	0	0	1	1	0	0
<i>f</i>	0	0	0	0	1	1

Aufgabe 2 Arrays und Heaps

Repräsentieren folgende Arrays Heaps? Begründen Sie Ihre Antwort!

- a) 1, 2, 9, 3, 7, 10, 11, 5, 4, 6, 8
- b) 1, 4, 2, 6, 7, 3, 5, 11, 10, 8, 9
- c) 4, 5, 3, 11, 8, 2, 1, 12, 9, 10, 7
- d) 12, 11, 6, 9, 7, 1, 2, 8, 4, 3, 5

Aufgabe 3 Heap Sort

Die sequentielle Liste

8, 2, 3, 7, 4, 1

ist mittels (*Min*-)Heap Sort zu sortieren. Geben Sie den Zustand des Heaps nach der **buildMin-Heap** und nach jedem **extractMin** wieder als sequentielle Liste an.

Der Einfachheit halber können Sie die Abspeicherung des sortierten Teils ignorieren.

Aufgabe 4 **Min-Priority Queue**

Die sequentielle Liste

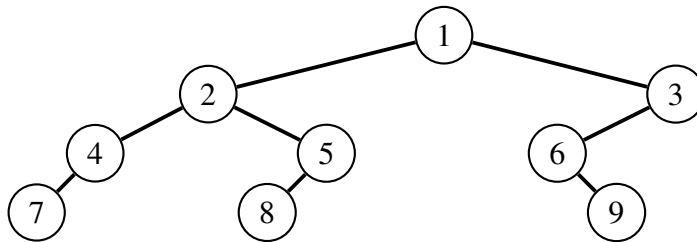
3,4,9,5,7

repräsentiere einen (*Min*-)Heap als Grundlage für eine (*Min*-)Priority Queue. Geben Sie jeweils den Zustand (als sequentielle Liste) an, nachdem zunächst **insert**(1) und dann **decreaseKey**(7, 2) ausgeführt wurde. Dabei bezeichne 7 den zu erhöhenden **Wert**, keinen Index.

Aufgabe 5 **Baum-Traversierung**

Geben Sie für die folgenden Bäume jeweils die sequentiellen Listen an, die bei *Pre-Order*-, *In-Order*- und *Post-Order-Traversierung* entstehen!

a)



b)

