

Grundlagen: Algorithmen und Datenstrukturen

Woche 8

Tobias Eppacher

School of Computation, Information and Technology

16. Juni 2025

Inhalt

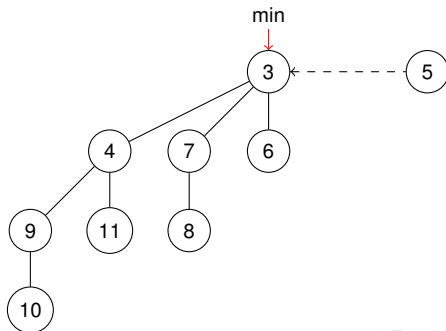
Aufgaben

E-Aufgaben

Hausaufgaben

Aufgabe 8.1 - Binomialheaps

Führen Sie auf dem folgenden Binomial-Heap nacheinander drei deleteMin-Operationen aus. Fügen Sie anschließend die drei entfernten Elemente in der Reihenfolge, in der sie entfernt wurden, wieder hinzu. Zeichnen Sie nach jeder deleteMin- und insert-Operation den entstandenen Binomial-Heap. Sind nach allen Operationen die Werte an derselben Stelle im Heap? Hat der Heap nach allen Operationen dieselbe Struktur? Warum?



Aufgabe 8.1 - Binomialheaps

Erste Operation:

Aufgabe 8.1 - Binomialheaps

Zweite Operation:

Aufgabe 8.1 - Binomialheaps

Dritte Operation:

Aufgabe 8.1 - Binomialheaps

Vierte Operation (ab hier Variante 1):

Aufgabe 8.1 - Binomialheaps

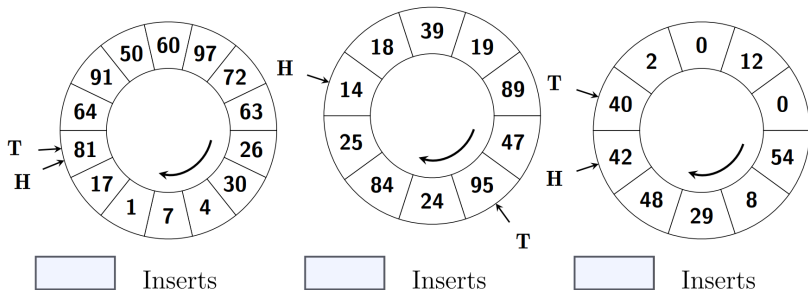
Fünfte Operation:

Aufgabe 8.1 - Binomialheaps

Sechste Operation:

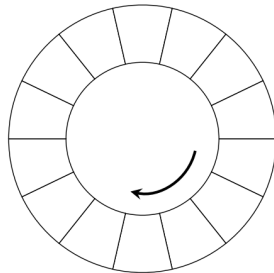
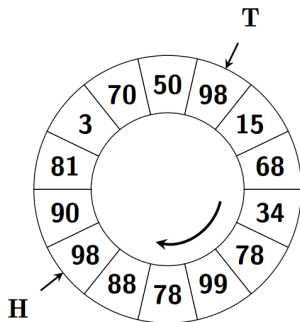
Aufgabe 8.2 - Rückblick: Circular Queues (a)

Gegeben sei ein Zirkulärer Ringspeicher als FIFO-Queue. In folgenden Darstellungen wird der Head mit **H** und der Tail mit **T** markiert. Geben Sie an, wie viele Inserts in folgenden Queues noch möglich sind. Dabei sollen Elemente, die aktuell in der Queue sind, weder überschrieben noch entfernt werden:



Aufgabe 8.2 - Rückblick: Circular Queues (b)

Der folgende Ringspeicher wird nun als **Deque** genutzt. Geben Sie den Ringspeicher an, nachdem folgende Operationen ausgeführt wurden: **pushFront(23)**, **pushBack(42)**, **popFront()**



Aufgabe 8.3 - Rückblick: Landausymbole (a)

Gegeben seien die Funktionen f, g mit $f(n) = \log_2 n * g(n)$.
Begründen Sie folgende Aussagen oder widerlegen Sie sie mit einem Gegenbeispiel:

1. Aus $g = \mathcal{O}(n)$ folgt $f = \mathcal{O}(\log_2 n * n)$

2. Aus $g = o(n)$ folgt $f = \omega(\log_2 n)$

Aufgabe 8.3 - Rückblick: Landausymbole (a)

Gegeben seien die Funktionen f, g mit $f(n) = \log_2 n * g(n)$.
Begründen Sie folgende Aussagen oder widerlegen Sie sie mit einem Gegenbeispiel:

3. Aus $f = \Theta(n^2)$ folgt $g = \mathcal{O}(n^2)$

4. Aus $f * g = \omega(n^4)$ folgt $f + g = \Omega(n^2)$

Aufgabe 8.3 - Rückblick: Landausymbole (b)

Geben Sie eine Funktion $f : \mathbb{N}_0 \rightarrow \mathbb{R}^+$ an, die $f = o(\log_2 n)$ und $\lim_{n \rightarrow \infty} f(n) = \infty$ erfüllt.

Aufgabe 8.4 - Baumtraversierung

Ein nichtleerer Binärbaum kann (unter anderem) in PreOrder, InOrder und PostOrder traversiert werden. Diese Traversierungen sind wie folgt rekursiv definiert:

1. PreOrder:

- a Besuche die Wurzel
- b Traversiere linken Teilbaum PreOrder (falls nicht leer)
- c Traversiere rechten Teilbaum PreOrder (falls nicht leer)

2. InOrder:

- a Traversiere linken Teilbaum InOrder (falls nicht leer)
- b Besuche die Wurzel
- c Traversiere rechten Teilbaum InOrder (falls nicht leer)

3. PostOrder:

- a Traversiere linken Teilbaum PostOrder (falls nicht leer)
- b Traversiere rechten Teilbaum PostOrder (falls nicht leer)
- c Besuche die Wurzel

Aufgabe 8.4 - Baumtraversierung

Anmerkung: PreOrder entspricht dfs-Nummer, PostOrder entspricht (dfs-)finish-Nummer wenn links vor rechts besucht wird.

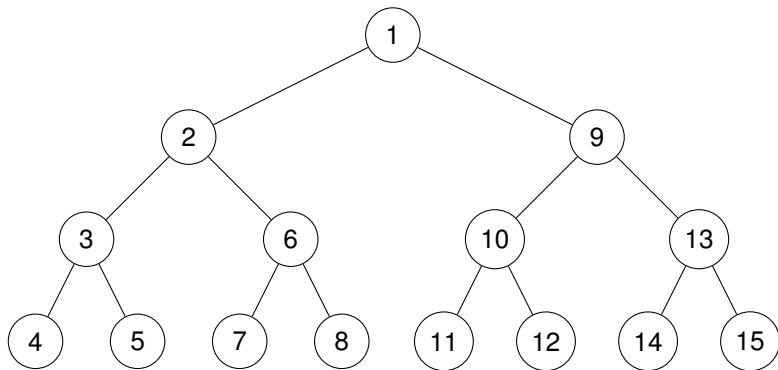
Geben Sie Algorithmen **preNext(v)** und **postNext(v)** an, die zu einem Knoten v in einem Binärbaum den in der PreOrder bzw. PostOrder folgenden Knoten w berechnet. Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Pseudocodes.

Nutzen Sie die folgenden Bäume, um die jeweiligen Traversierungsalgorithmen zu visualisieren.

Berechnen Sie außerdem die asymptotische Laufzeit, wenn mittels der Operationen **preNext(v)** und **postNext(v)** die vollständige PreOrder bzw. PostOrder berechnet wird (also n -maliges Anwenden der Funktion).

Erlaubte Operationen: $parent(v)$, $leftChild(v)$, $rightChild(v)$, $hasleftChild(v)$, $hasrightChild(v)$, $isRoot(v)$, $isInternal(v)$

Aufgabe 8.4 - Baumtraversierung (PreOrder)

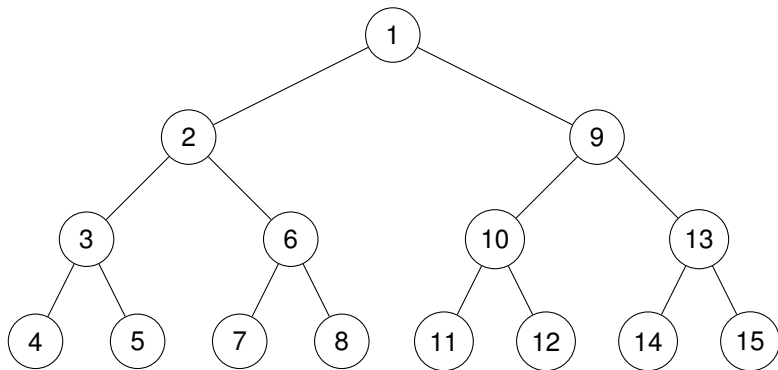


Aufgabe 8.4 - Baumtraversierung (PreOrder)

```
public Node preNext (Node v) {
```

```
}
```

Aufgabe 8.4 - Baumtraversierung (PostOrder)



Aufgabe 8.4 - Baumtraversierung (PostOrder)

```
public Node postNext (Node v) {
```

```
}
```

Aufgabe 8.4 - Baumtraversierung (Asymptotische Laufzeit)

E-Aufgaben

- ▶ Aufgabe 8.5 - Rückblick: Sortierverfahren
 - ▶ Laufzeitenvergleich von Sortierverfahren

Hausaufgaben

- ▶ Hausaufgabe 9 - AB-Baum
(Deadline: 09.07.2025)

Fragen?

- ▶ Nach Übung gerne bei mir melden
- ▶ Tutoriumschannel oder DM an mich auf Zulip
- ▶ Vorlesungschannels von GAD auf Zulip (insbesondere bei Hausaufgaben)

Feedback oder Verbesserungsvorschläge?

Gerne nach dem Tutorium mit mir quatschen oder DM auf Zulip

Bis nächste Woche!