



openHPI  
Einführung in die Mathematik der Algorithmik  
– Binäre Min-Heaps

**Timo Kötzing und Pascal Lenzner**

Hasso-Plattner-Institut

Universität Potsdam

Wie verwalten wir clever die Randknoten und deren Distanz zum Startknoten?

Aufgaben:

ExtractMin

Insert

DecreaseKey

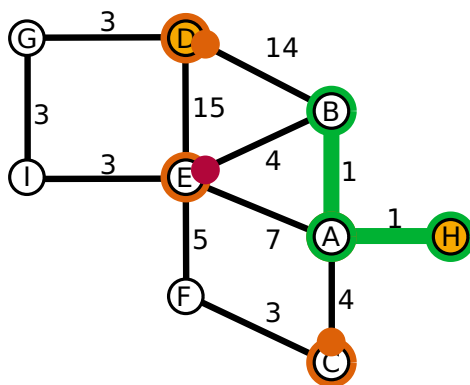
Priority Queue:

- Elemente warten in Warteschlange
- jedes Element hat Prioritätswert
- das Element mit der höchsten Priorität kommt als nächstes an die Reihe

Verwaltung der Randknoten mit einer Priority Queue:

- benutze Distanz zum Startknoten als Priorität
- je kleiner die Distanz, desto höher die Priorität

1. Idee: Verwalte Randknoten als nach Priorität sortierte Liste



Priority Queue:  $(B, 2) \rightarrow (C, 5) \rightarrow (E, 8)$

ExtractMin

$(C, 5) \rightarrow (E, 8)$   $(B, 2)$  wird ausgegeben

Insert

$(C, 5) \rightarrow (E, 8) \rightarrow (D, 16)$   $(D, 16)$  wurde eingefügt

DecreaseKey

$(C, 5) \rightarrow (E, 6) \rightarrow (D, 16)$

Priorität von  $E$  wurde von 8 auf 6 gesenkt

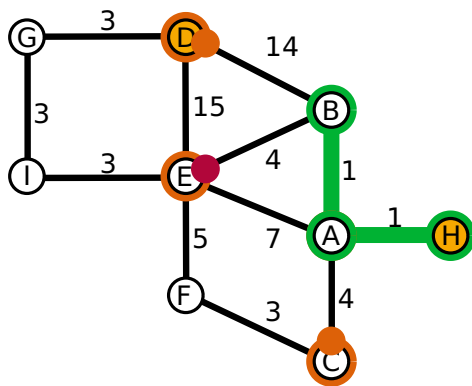
## Problem:

Müssen viel Aufwand betreiben,  
um die Liste sortiert zu halten!

## Lösung:

Benutze spezialisierte Daten-  
struktur: **Binäre Min-Heaps!**

1. Idee: Verwalte Randknoten als nach Priorität sortierte Liste



Priority Queue:  $(B, 2) \rightarrow (C, 5) \rightarrow (E, 8)$

**ExtractMin**

$(C, 5) \rightarrow (E, 8)$   $(B, 2)$  wird ausgegeben

**Insert**

$(C, 5) \rightarrow (E, 8) \rightarrow (D, 16)$   $(D, 16)$  wurde  
eingefügt

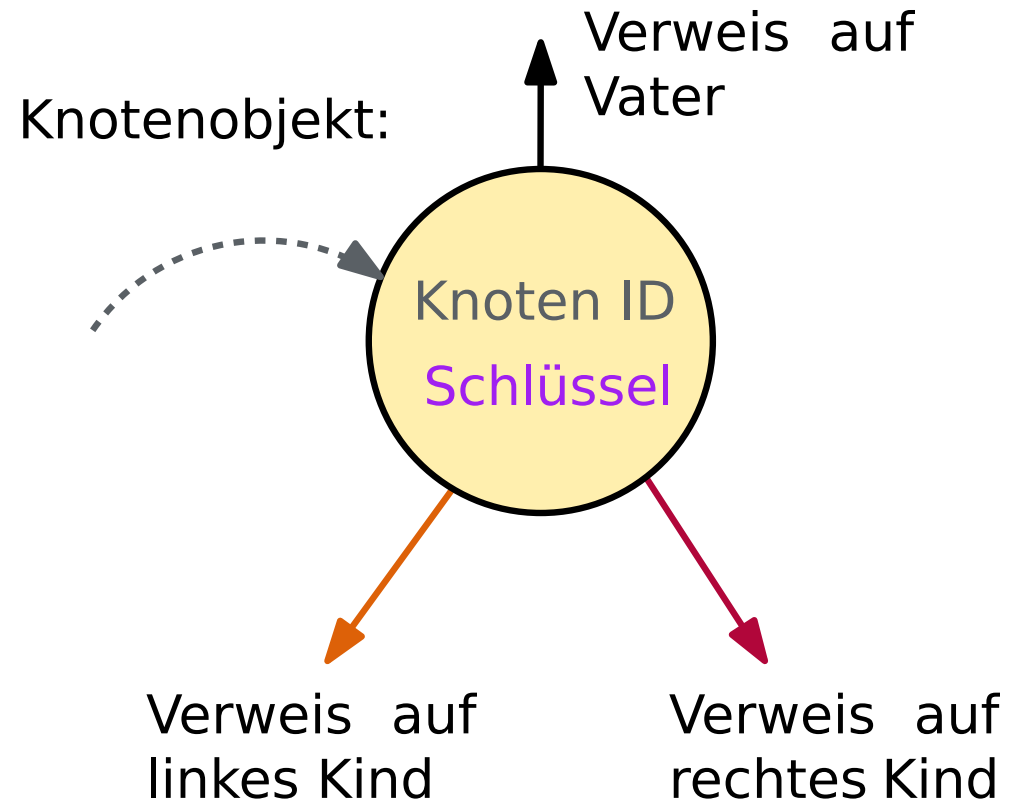
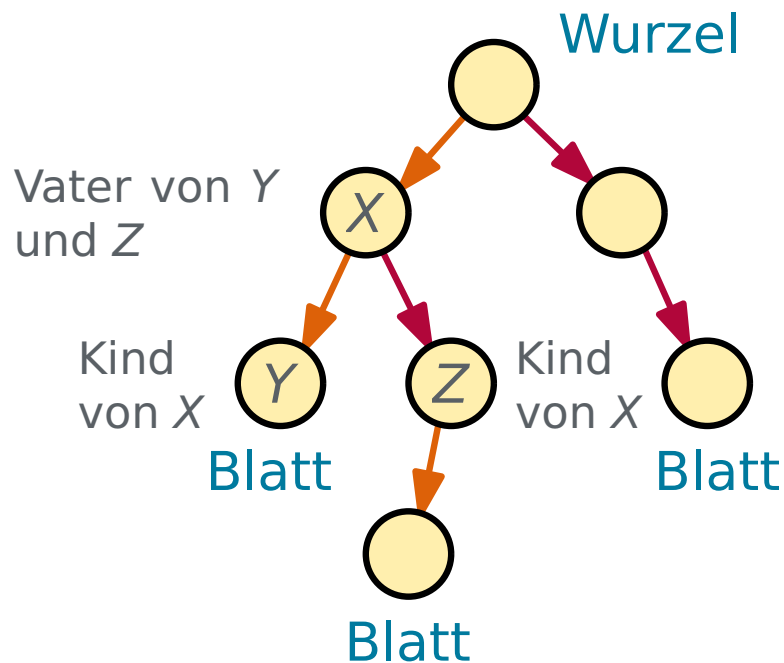
**DecreaseKey**

$(C, 5) \rightarrow (E, 6) \rightarrow (D, 16)$

Priorität von  $E$   
wurde von 8  
auf 6 gesenkt

## 2. Idee: Verwalte Randknoten in einer Baumstruktur!

Betrachten Binärbäume:

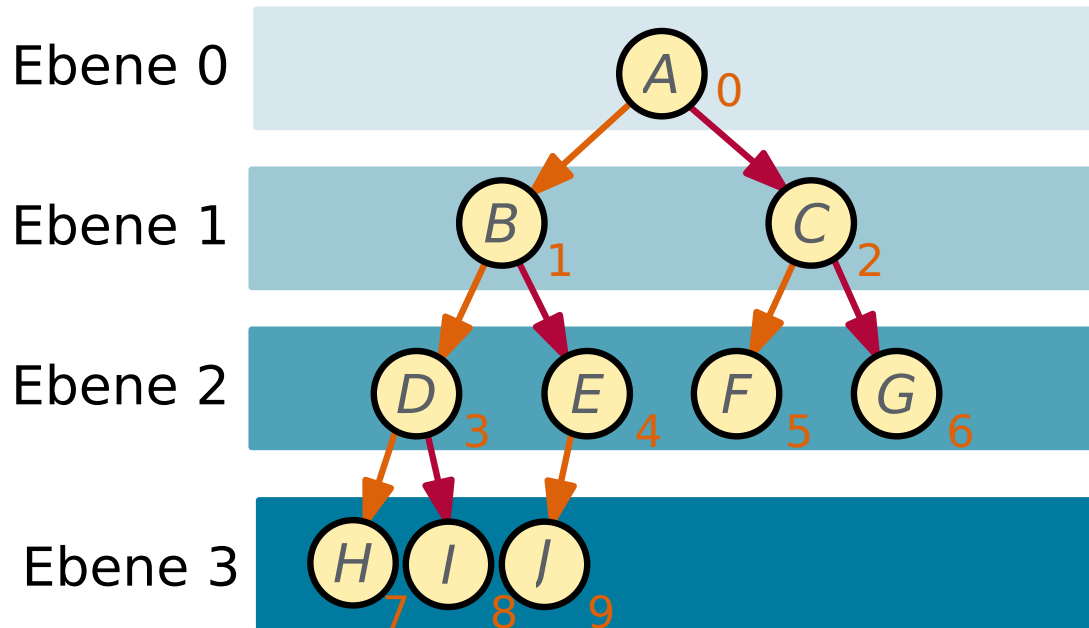


- Benötigen für Struktur nur Knotenobjekte und Verweise
- Haben zusätzlich Verweis von jedem Knoten im Graph auf entsprechenden Knoten im Baum

## 2. Idee: Verwalte Randknoten in einer Baumstruktur!

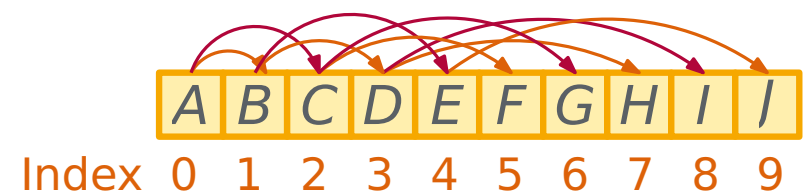
Betrachten **vollständige** Binärbäume:

- alle Ebenen sind gefüllt, letzte Ebene ist von links her gefüllt



Vorteil:

Vollständiger Binärbaum kann leicht in einem Array gespeichert werden!



Wurzel:  
Index 0

Linkes Kind von Knoten  
mit Index  $i$ : Index  $2 \cdot i + 1$

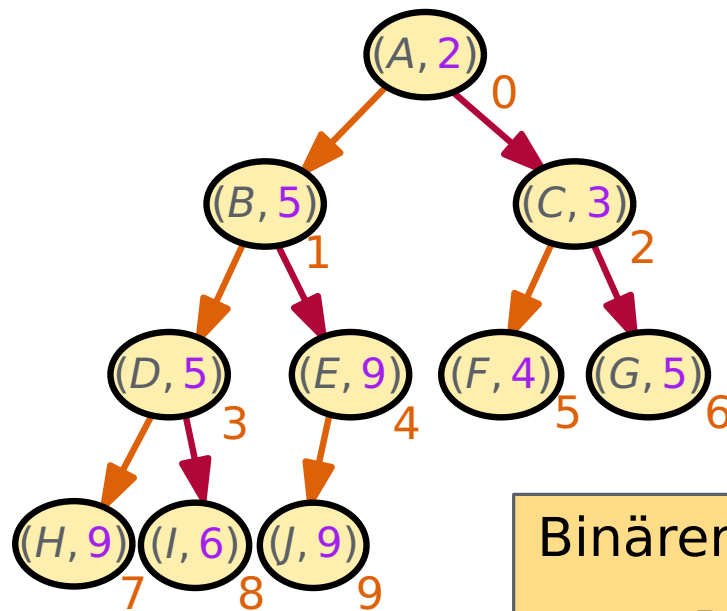
Rechtes Kind von Knoten  
mit Index  $i$ : Index  $2 \cdot i + 2$



2. Idee: Verwalte Randknoten in einer Baumstruktur!

Betrachten **vollständige** Binärbäume mit **Min-Heap-Eigenschaft**:

- Schlüsselwert des Vaters ist  $\leq$  Schlüsselwert jedes Kindes



- Wurzel hat kleinsten Schlüsselwert
- auf jedem Pfad von der Wurzel zu einem Blatt sind die Schlüsselwerte monoton steigend

Binärer Min-Heap:

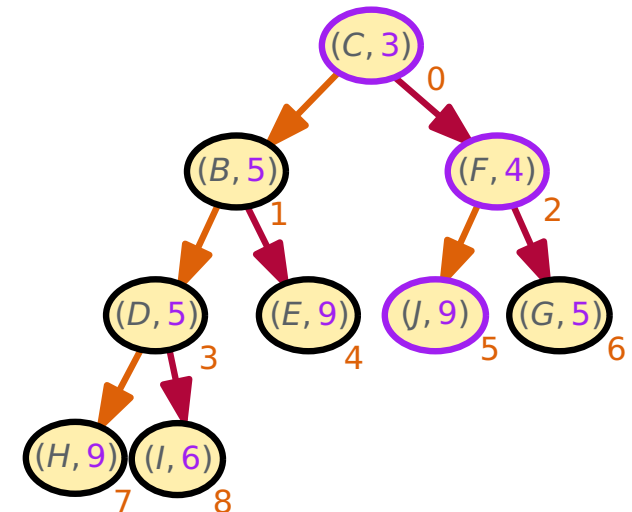
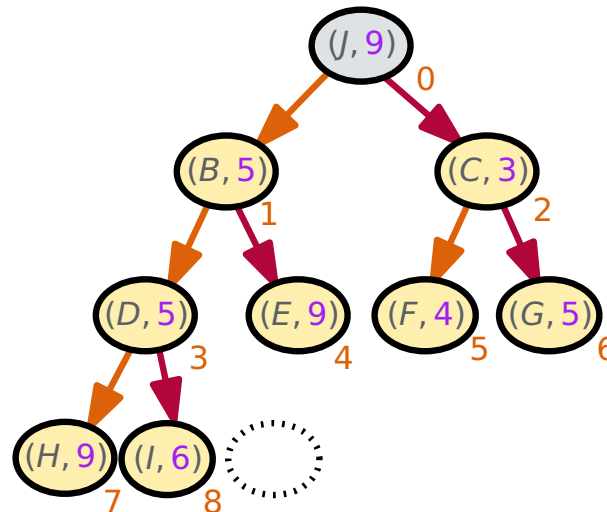
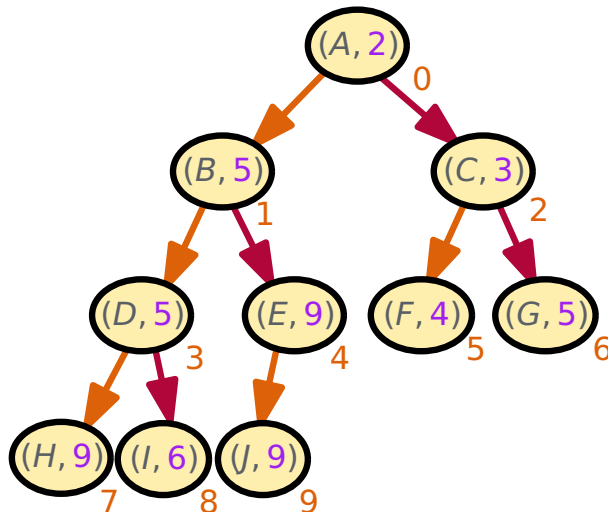
- Daten+Schlüssel werden in vollständigem Binärbaum gespeichert
- Im Baum gilt die Min-Heap-Eigenschaft

## ExtractMin

- Gib Datenelement der Wurzel zurück
- Entferne Wurzel

## Binärer Min-Heap:

- Daten+Schlüssel werden in vollständigem Binärbaum gespeichert
- Im Baum gilt die Min-Heap-Eigenschaft



## Vorgehen:

1. Gib Element der Wurzel zurück
2. Vertausche Daten der Wurzel mit Daten des rechten Knotens in der letzten Ebene. Lösche danach den letzten Knoten.

3. Stelle Min-Heap-Eigenschaft ausgehend von der Wurzel wieder her.

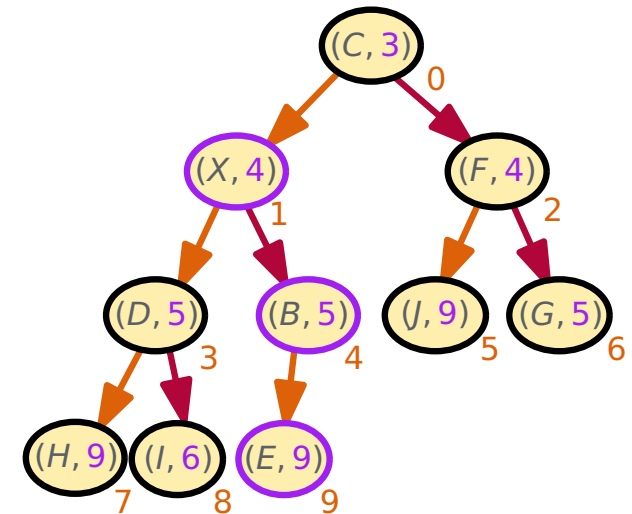
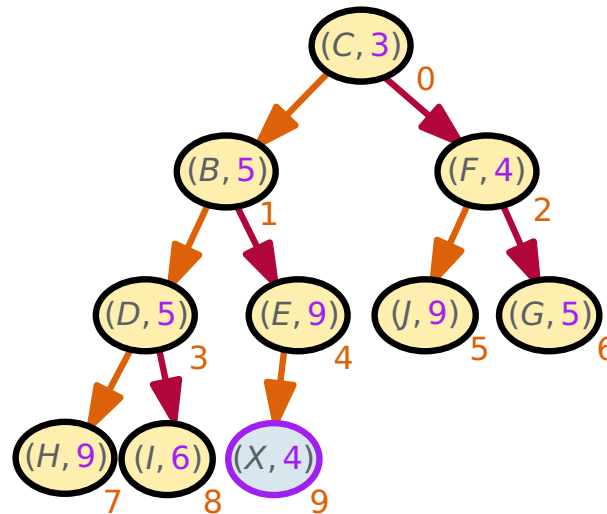
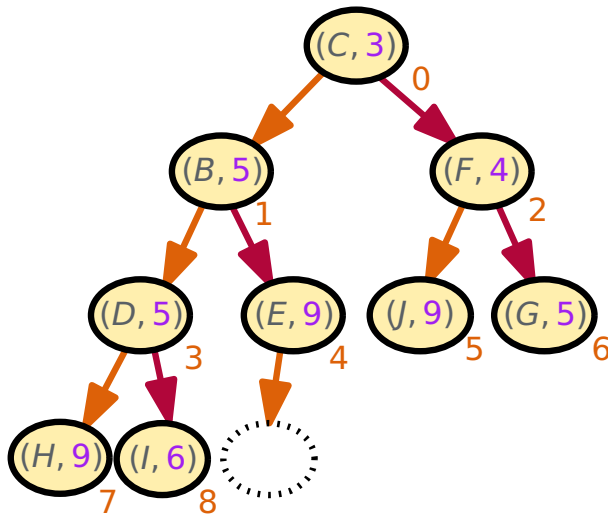
Pro Knoten, falls Eigenschaft verletzt: Tausche mit Kind mit kleinstem Schlüssel!

## Insert

- füge neues Element + Schlüsselwert ein

### Binärer Min-Heap:

- Daten+Schlüssel werden in vollständigem Binärbaum gespeichert
- Im Baum gilt die Min-Heap-Eigenschaft



### Vorgehen:

1. Erstelle neuen Knoten als rechtesten Knoten in der letzten Ebene (falls voll, dann neue Ebene beginnen)
2. Neuer Knoten erhält Element + Schlüsselwert

3. Stelle Min-Heap-Eigenschaft ausgehend vom neuen Knoten wieder her.

Pro Knoten, falls Eigenschaft verletzt: Tausche mit Vater!

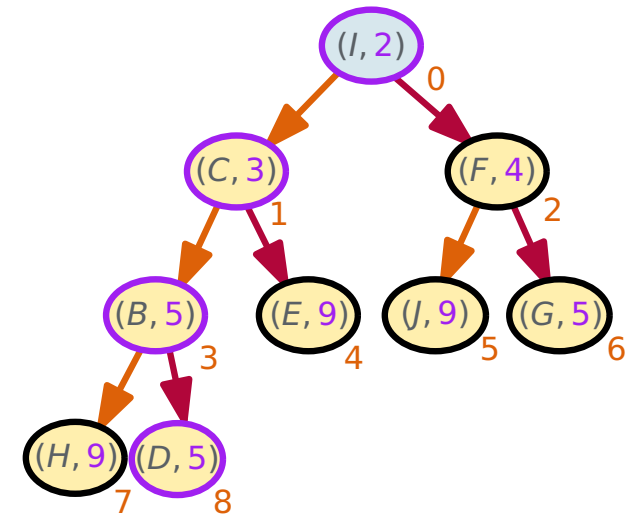
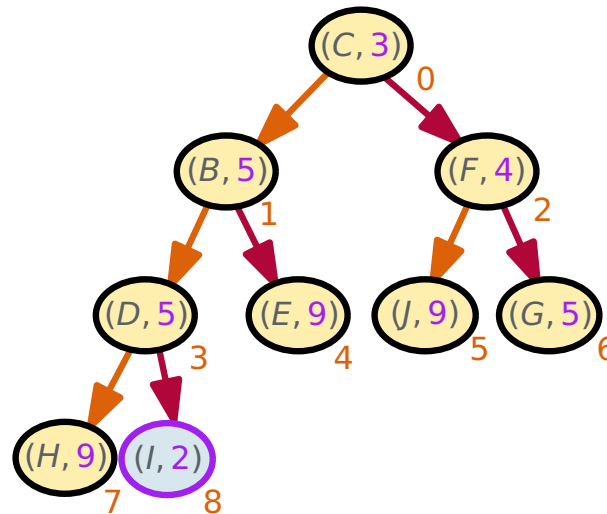
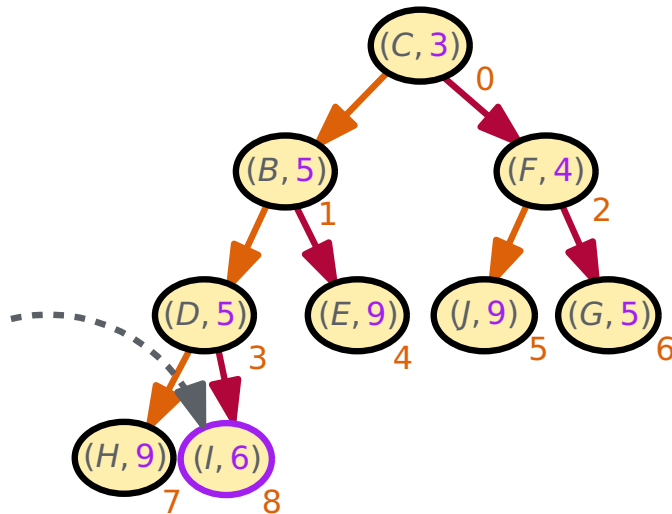


## DecreaseKey

- verringert Schlüsselwert eines vorhandenen Elements

## Binärer Min-Heap:

- Daten+Schlüssel werden in vollständigem Binärbaum gespeichert
- Im Baum gilt die Min-Heap-Eigenschaft



## Vorgehen:

1. Per Direktzugriff auf zu ändernden Knoten zugreifen
2. Schlüsselwert verringern

3. Stelle Min-Heap-Eigenschaft ausgehend vom geänderten Knoten wieder her

Pro Knoten, falls Eigenschaft verletzt: Tausche mit Vater!

Wie verwalten wir clever die Randknoten und deren Distanz zum Startknoten?

Aufgaben:

ExtractMin

Insert

DecreaseKey

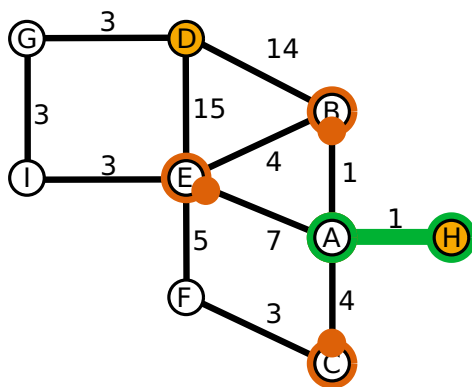
Priority Queue:

- Elemente warten in Warteschlange
- jedes Element hat Prioritätswert
- das Element mit der höchsten Priorität kommt als nächstes an die Reihe

Verwaltung der Randknoten mit einer Priority Queue:

- benutze Distanz zum Startknoten als Priorität
- je kleiner die Distanz, desto höher die Priorität

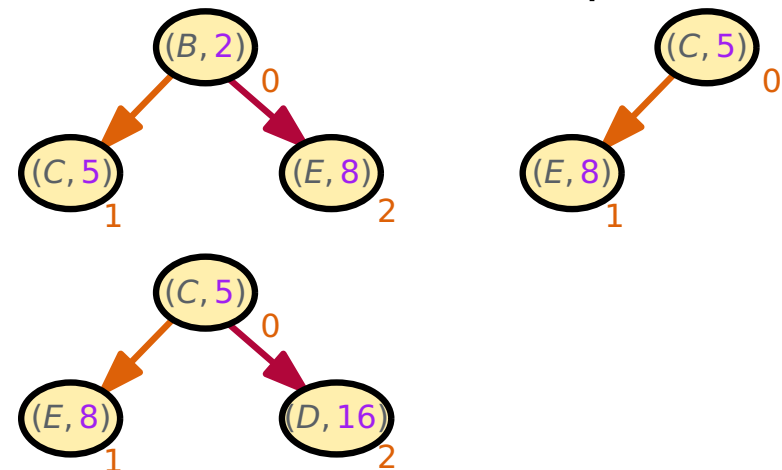
2. Idee: Verwalte Randknoten in einem binären Min-Heap



Priority Queue:

ExtractMin

Insert



Wie verwalten wir clever die Randknoten und deren Distanz zum Startknoten?

Aufgaben:

ExtractMin

Insert

DecreaseKey

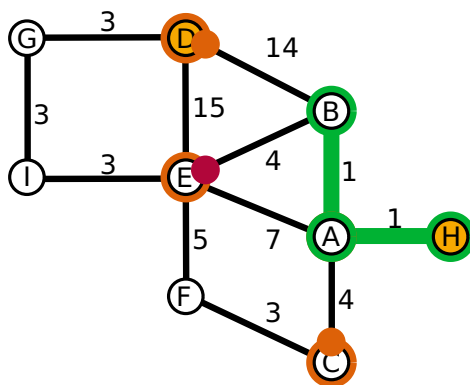
Priority Queue:

- Elemente warten in Warteschlange
- jedes Element hat Prioritätswert
- das Element mit der höchsten Priorität kommt als nächstes an die Reihe

Verwaltung der Randknoten mit einer Priority Queue:

- benutze Distanz zum Startknoten als Priorität
- je kleiner die Distanz, desto höher die Priorität

2. Idee: Verwalte Randknoten in einem binären Min-Heap



Priority Queue:

ExtractMin

Insert

DecreaseKey

