Tutorium
Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 11

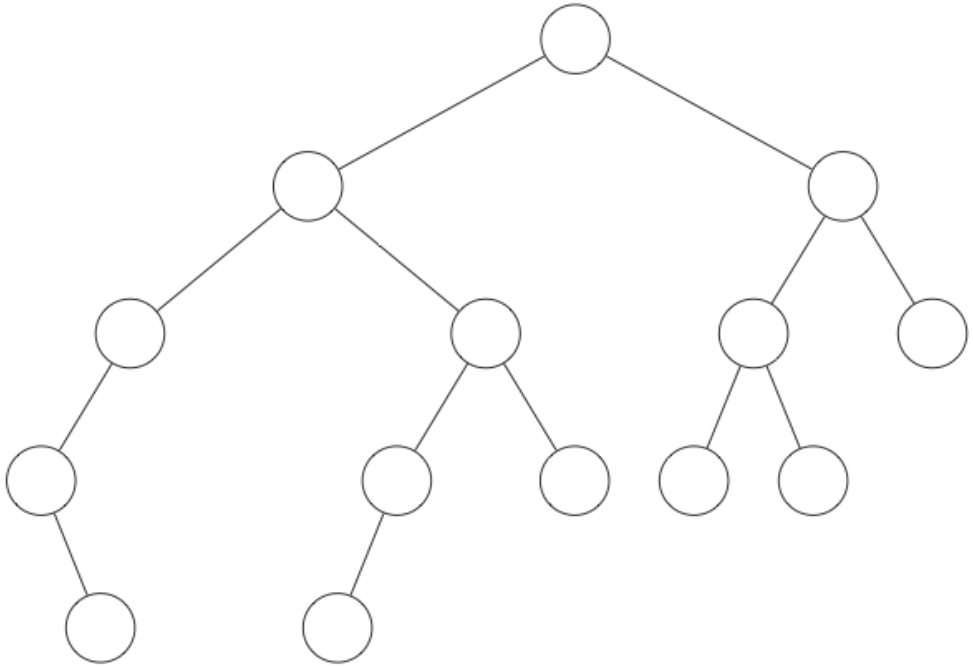
Aufgabe 11.1 – Baumtraversierung

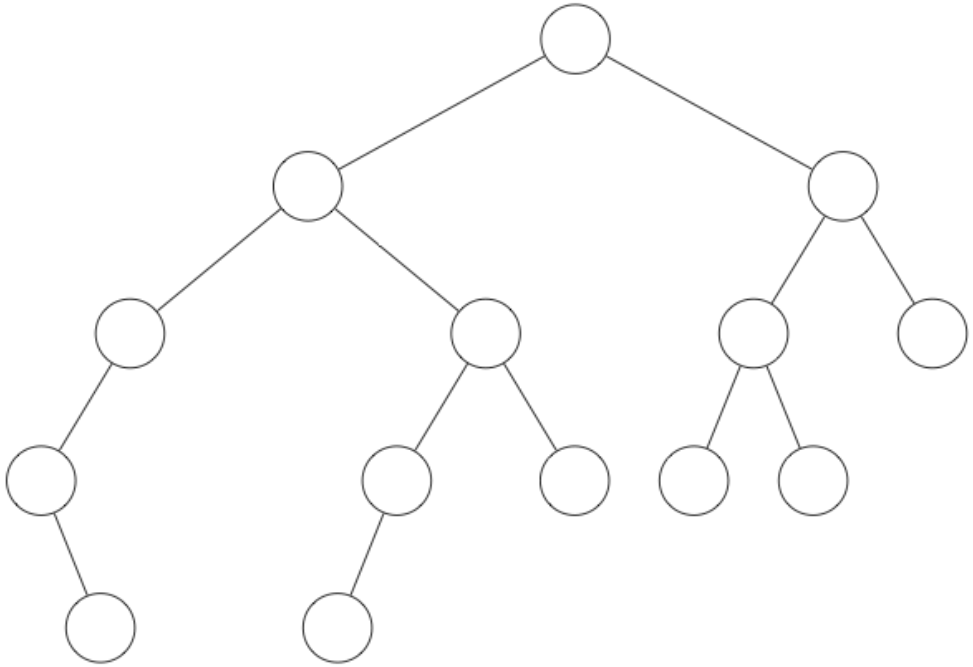
Ein nichtleerer Binärbaum kann (unter anderem) in PreOrder, InOrder und PostOrder traversiert werden. Diese Traversierungen sind wie folgt rekursiv definiert:

- PreOrder:
  a) Besuche die Wurzel.
  b) Traversiere den linken Teilbaum in PreOrder, falls dieser nichtleer ist.
  c) Traversiere rechten Teilbaum in PreOrder, falls dieser nichtleer ist.
- InOrder:
  a) Traversiere den linken Teilbaum in InOrder, falls dieser nichtleer ist.
  b) Besuche die Wurzel.
  c) Traversiere den rechten Teilbaum in InOrder, falls dieser nichtleer ist.
- PostOrder:
  a) Traversiere den linken Teilbaum in PostOrder, falls dieser nichtleer ist.
  b) Traversiere den rechten Teilbaum in PostOrder, falls dieser nichtleer ist.
  c) Besuche die Wurzel.

Aufgabe 11.1 – Baumtraversierung

- Pre-Order entspricht DFS-Nummerierung
- Post-Order entspricht (DFS-)Finish-Nummerierung



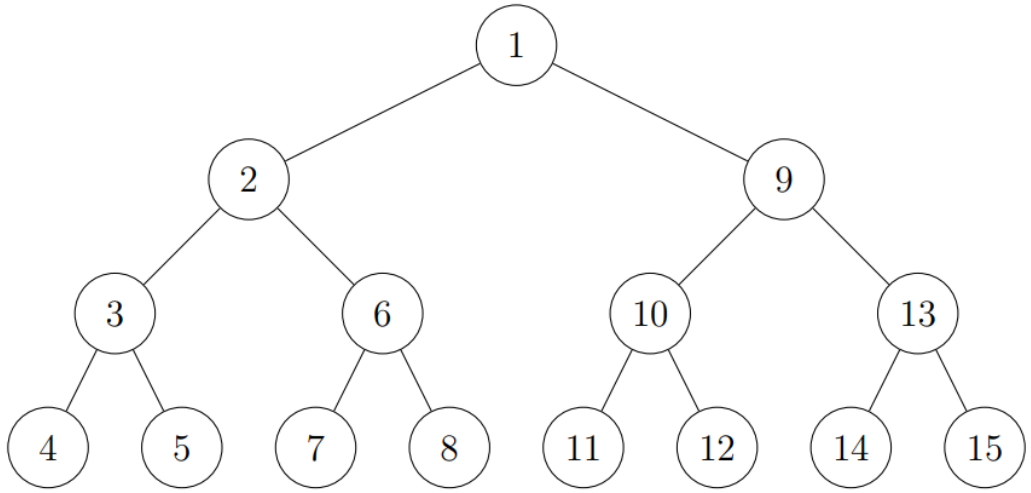


Aufgabe 11.1 – Baumtraversierung

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).

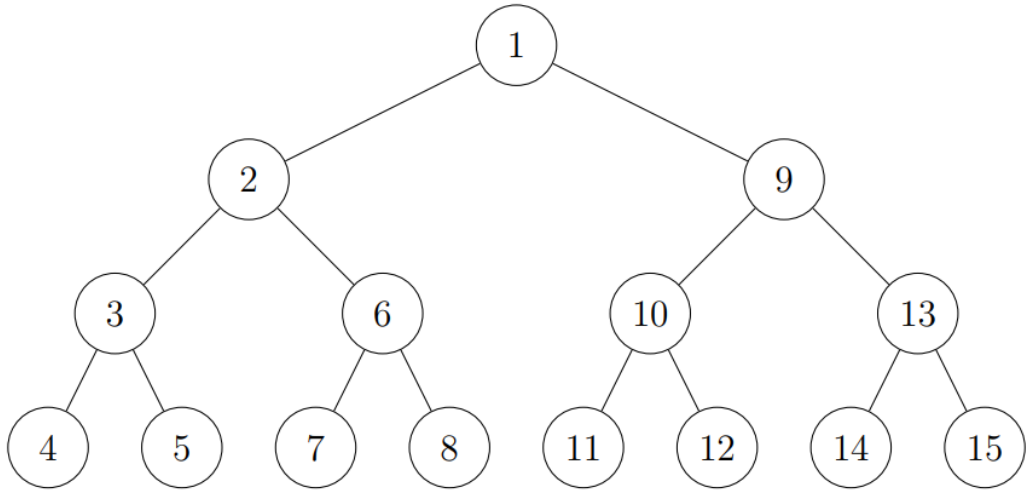


Aufgabe 11.1 – Baumtraversierung

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).



Aufgabe 11.1 – Baumtraversierung

```
1  public Node preNext(Node v){
2    if (hasleftChild(v))
3      return leftChild(v);
4    else if (hasrightChild(v))
5      return rightChild(v);
6    else{
7      Node p = v;
8      Node q;
9      while (!isRoot(p)){
10       q = p;
11       p = parent(p);
12       if (hasrightChild(p) && rightChild(p)!=q)
13         return rightChild(p);
14     }
15     return null;
16   }
17 }
```

Aufgabe 11.1 – Baumtraversierung

```
1  public Node postNext(Node v){
2    if (!isRoot(v)){
3      Node p = parent(v);
4      if (hasrightChild(p)){
5        if (v==rightChild(p))
6          return p;
7        else{
8          Node c = rightChild(p);
9          while (isInternal(c)){
10           if (hasleftChild(c))
11             c = leftChild(c);
12           else
13             c = rightChild(c);
14         }
15         return c;
16       }
17     }
18     else return p;
19   }
20   else return null;
21 }
```

Aufgabe 11.1 – Baumtraversierung

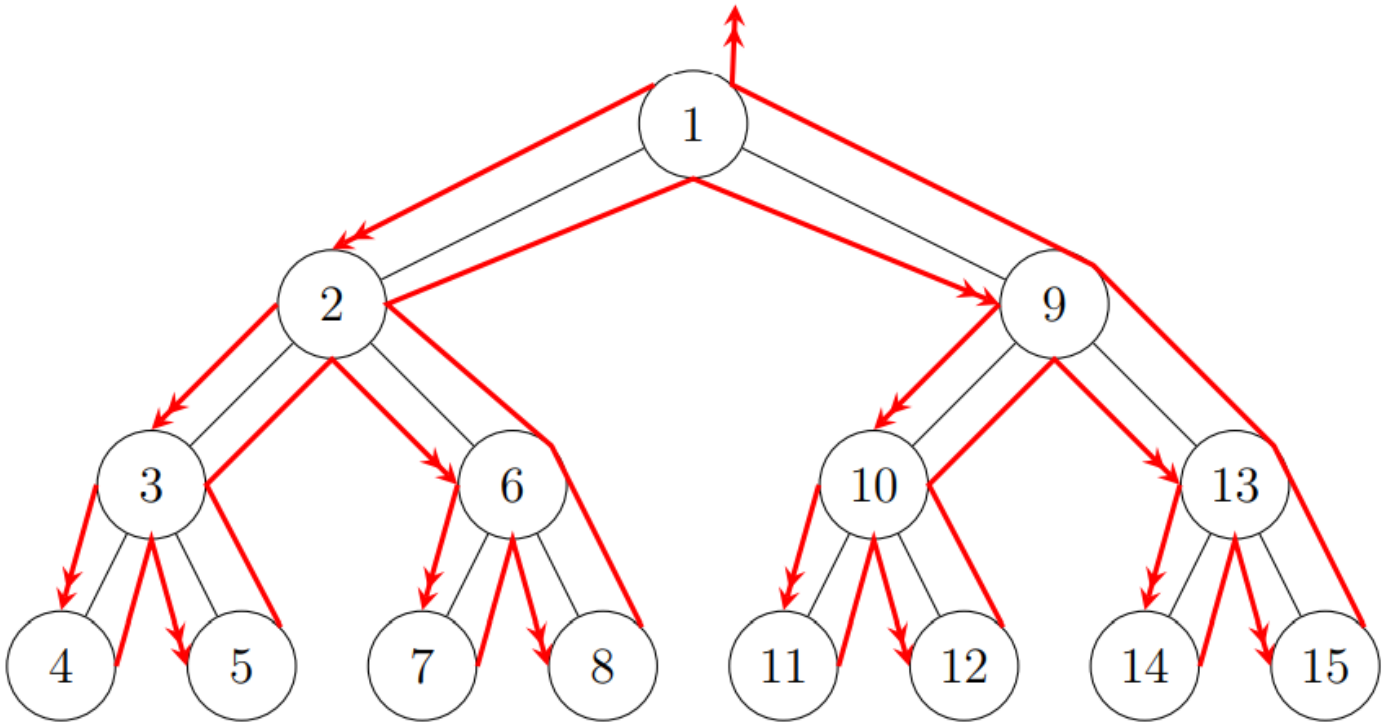
- **Laufzeit?**
  - preNext(v):
  - postNext(v):

Aufgabe 11.1 – Baumtraversierung

- **Laufzeit voller Traversierung?**

Aufgabe 11.1 – Baumtraversierung

- **Laufzeit voller Traversierung?**



Volle Traversierung mit preNext(v)

Wiederholung: AB-Bäume

Aufgabe 11.2 – ABBaummaßnahmen I

Führen Sie auf einem anfangs leeren $(2, 3)$-Baum die folgenden Operationen aus:

`insert`: [19, 11, 28, 38, 37, 30, 7, 59 ,41]

gefolgt von:

`remove`: [7, 37, 59, 41, 11, 19, 30, 38]

Hinweis: Zeichnen Sie den Baum nach jedem Schritt. Sie dürfen in Ihrer Zeichnung auf Blattknoten verzichten.

Beachten Sie außerdem das Folgende:

- Beim Aufspalten von Knoten während dem Einfügen wandert das Element am Index $\lfloor b/2 \rfloor$ nach oben.
- Beim Löschen von Elementen aus inneren Knoten wird üblicherweise versucht, entweder den symmetrischen Vorgänger oder symmetrischen Nachfolger intelligent zu wählen. Für diese Aufgabe soll darauf verzichtet werden. Stattdessen wird stets der symmetrische Vorgänger verwendet. Beim *Stehlen* von Elementen wird zunächst der linke Nachbar betrachtet. Beim *Verschmelzen* werden Knoten sofern möglich mit ihrem linken Nachbarn, ansonsten mit dem rechten Nachbarn, vereinigt.

Aufgabe 11.2 – ABBaummaßnahmen I

Führen Sie auf einem anfangs leeren $(2, 3)$-Baum die folgenden Operationen aus:

`insert`: [19, 11, 28, 38, 37, 30, 7, 59 ,41]

gefolgt von:

`remove`: [7, 37, 59, 41, 11, 19, 30, 38]

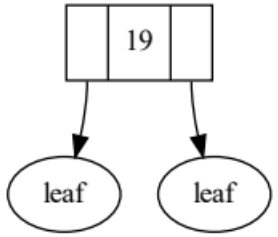
Hinweis: Zeichnen Sie den Baum nach jedem Schritt. Sie dürfen in Ihrer Zeichnung auf Blattknoten verzichten.

Beachten Sie außerdem das Folgende:

- Beim Aufspalten von Knoten während dem Einfügen wandert das Element am Index $\lfloor b/2 \rfloor$ nach oben.
- Beim Löschen von Elementen aus inneren Knoten wird üblicherweise versucht, entweder den symmetrischen Vorgänger oder symmetrischen Nachfolger intelligent zu wählen. Für diese Aufgabe soll darauf verzichtet werden. Stattdessen wird stets der symmetrische Vorgänger verwendet. Beim *Stehlen* von Elementen wird zunächst der linke Nachbar betrachtet. Beim *Verschmelzen* werden Knoten sofern möglich mit ihrem linken Nachbarn, ansonsten mit dem rechten Nachbarn, vereinigt.

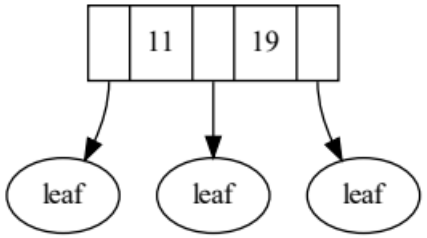
Aufgabe 11.2 – ABBaummaßnahmen I

a) insert(19):



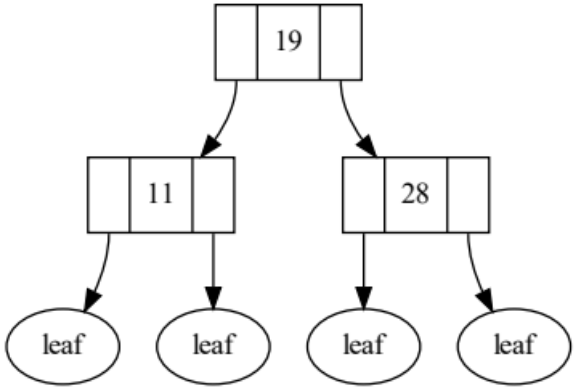
Aufgabe 11.2 – ABBaummaßnahmen I

b) insert(11):



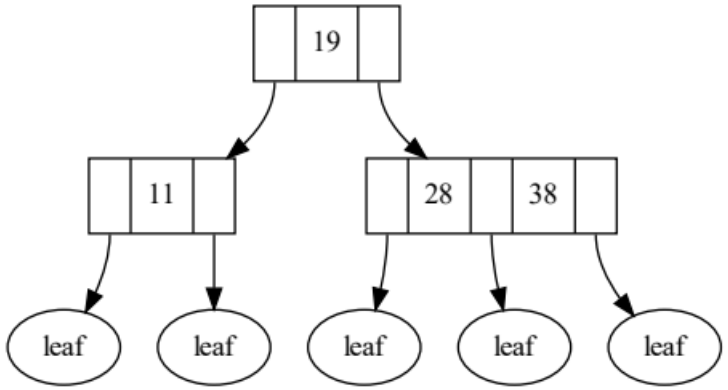
Aufgabe 11.2 – ABBaummaßnahmen I

c) insert(28):



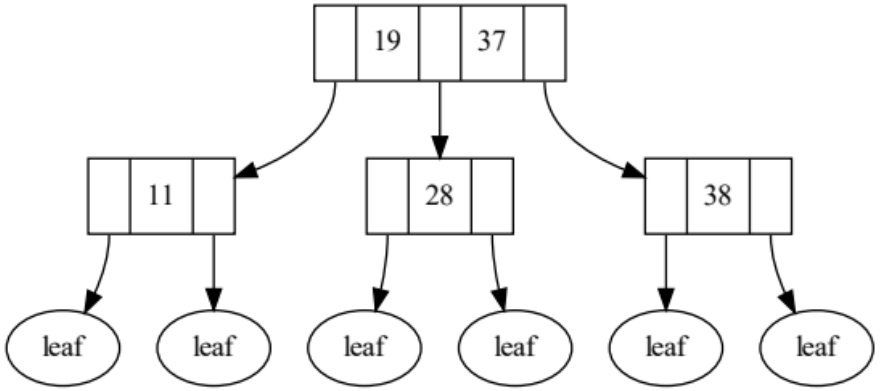
Aufgabe 11.2 – ABBaummaßnahmen I

d) insert(38):



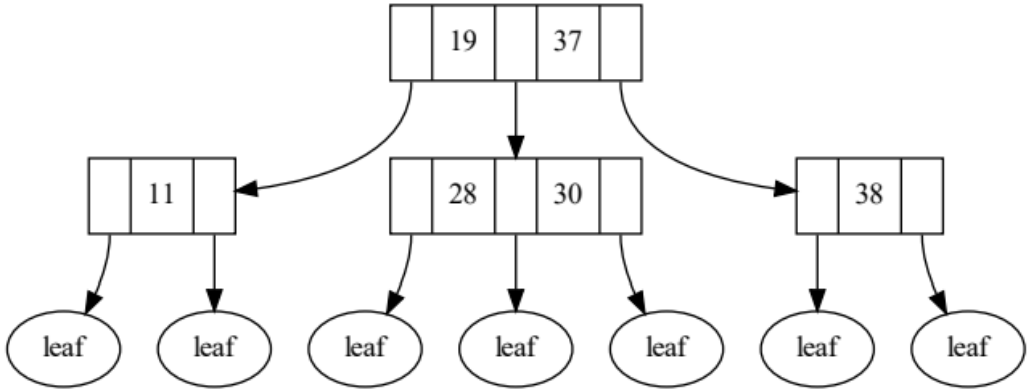
Aufgabe 11.2 – ABBaummaßnahmen I

e) insert(37):



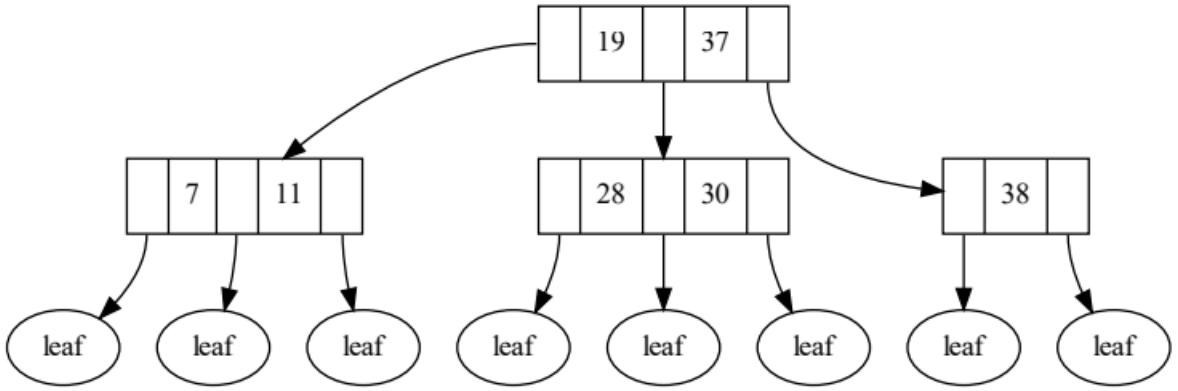
Aufgabe 11.2 – ABBaummaßnahmen I

f) insert(30):



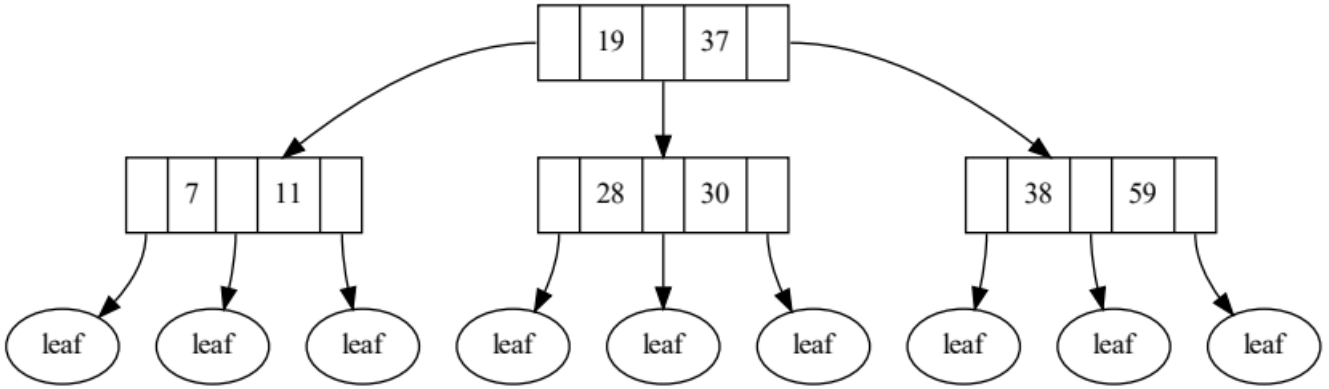
Aufgabe 11.2 – ABBaummaßnahmen I

g) insert(7):



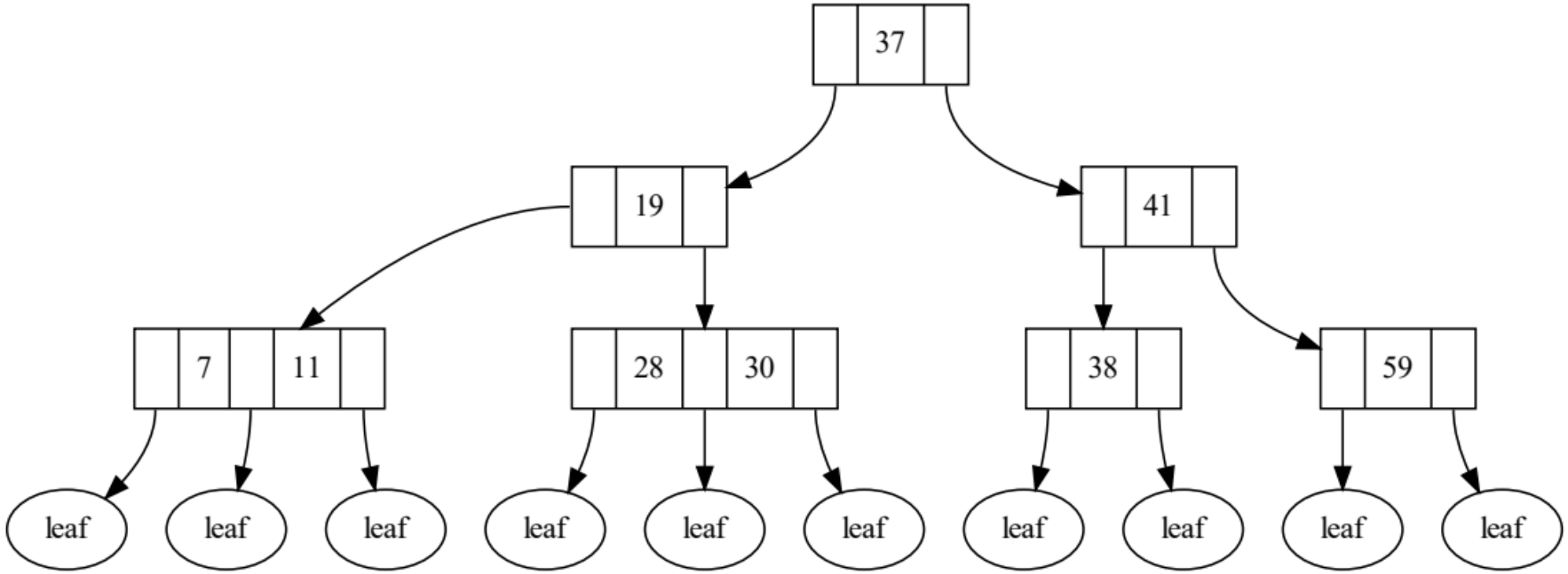
Aufgabe 11.2 – ABBaummaßnahmen I

h) insert(59):



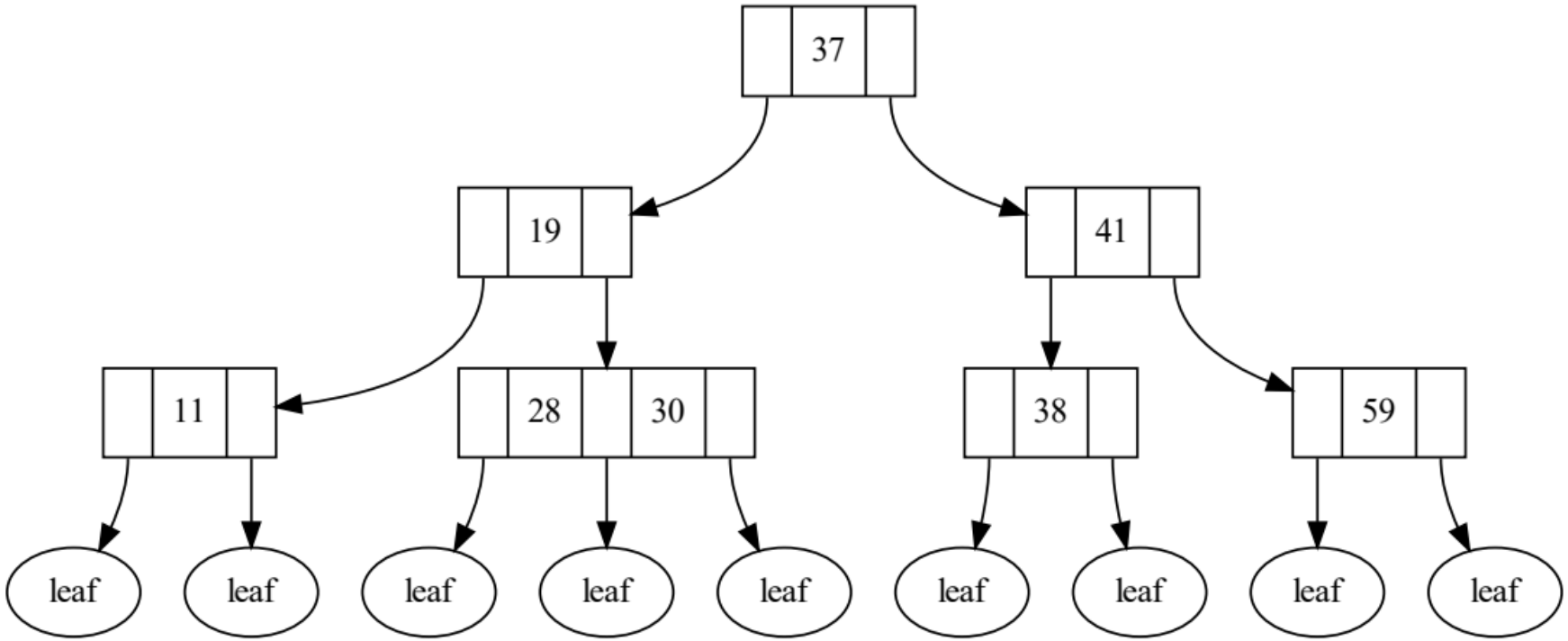
Aufgabe 11.2 – ABBaummaßnahmen I

i) insert(41):



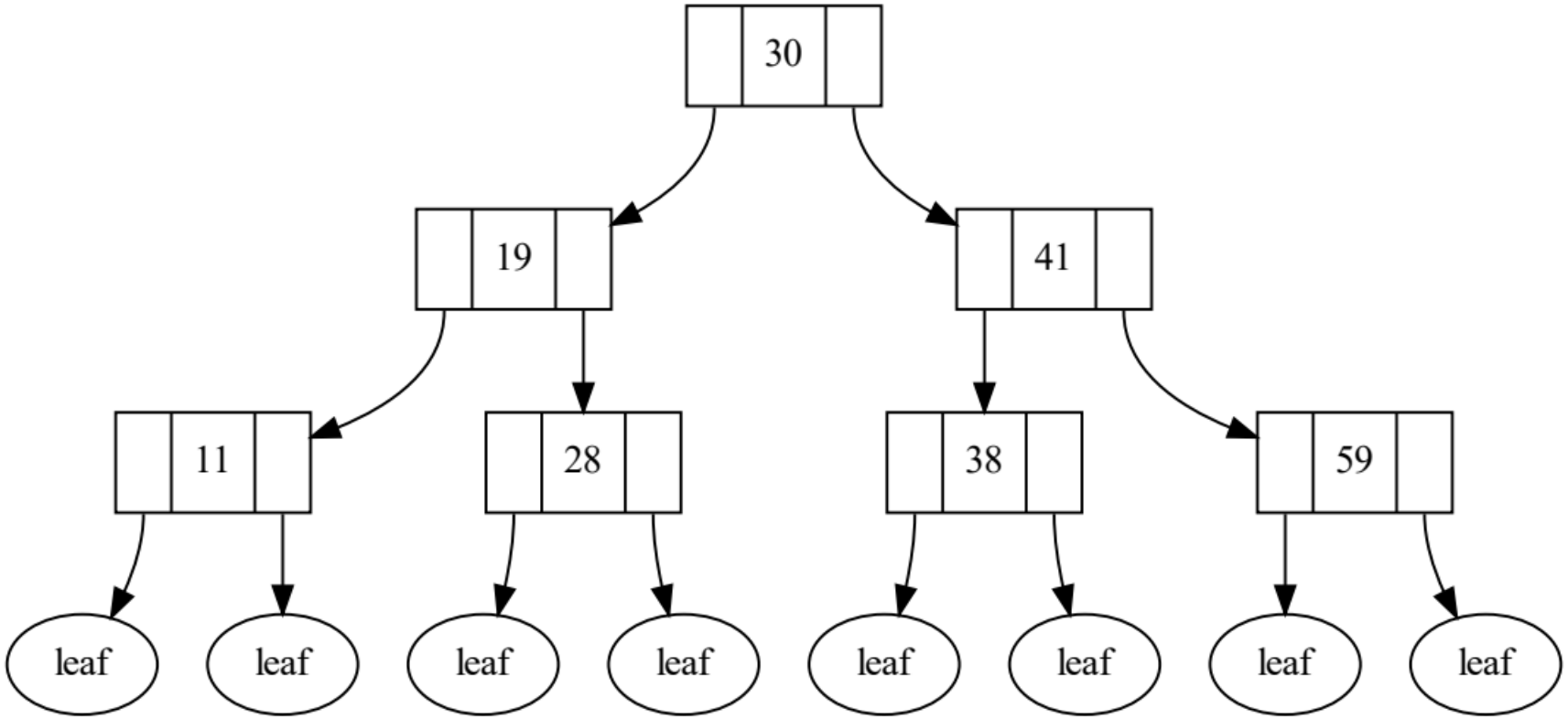
Aufgabe 11.2 – ABBaummaßnahmen I

j) remove(7):



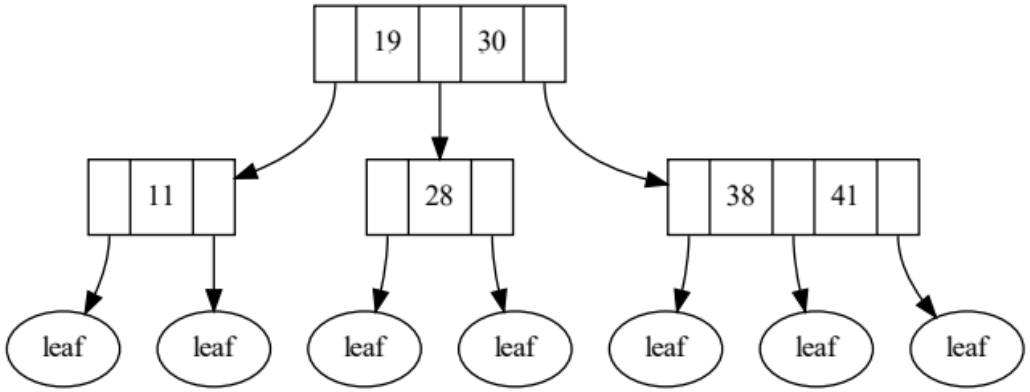
Aufgabe 11.2 – ABBaummaßnahmen I

k) remove(37):



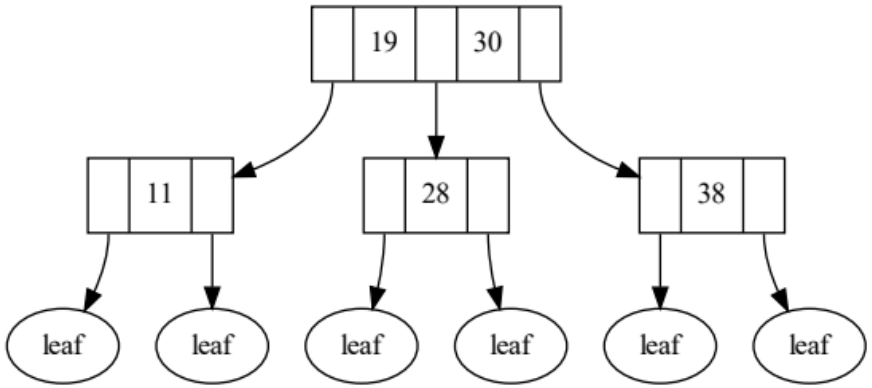
Aufgabe 11.2 – ABBaummaßnahmen I

l) remove(59):



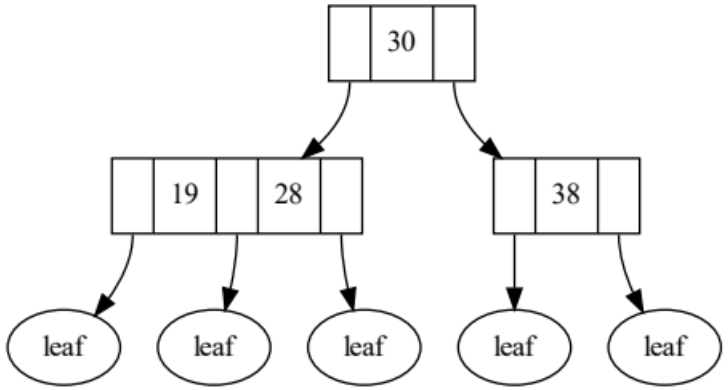
Aufgabe 11.2 – ABBaummaßnahmen I

m) remove(41):



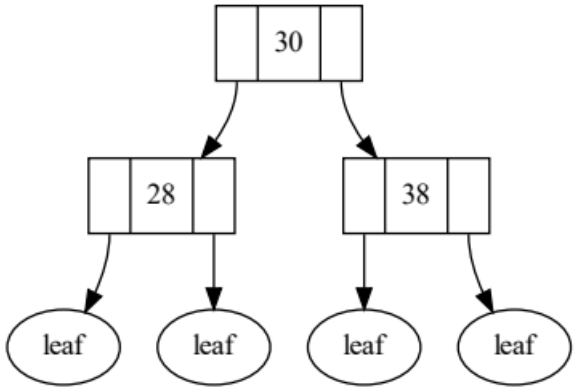
Aufgabe 11.2 – ABBaummaßnahmen I

n) remove(11):



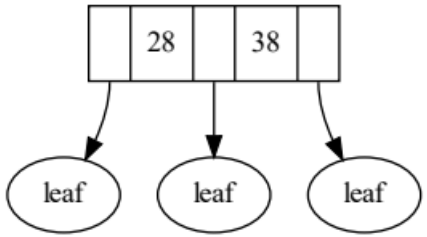
Aufgabe 11.2 – ABBaummaßnahmen I

o) remove(19):



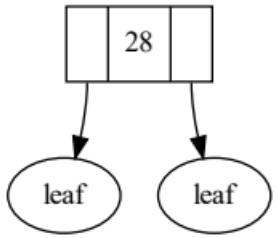
Aufgabe 11.2 – ABBaummaßnahmen I

p) remove(30):



Aufgabe 11.2 – ABBaummaßnahmen I

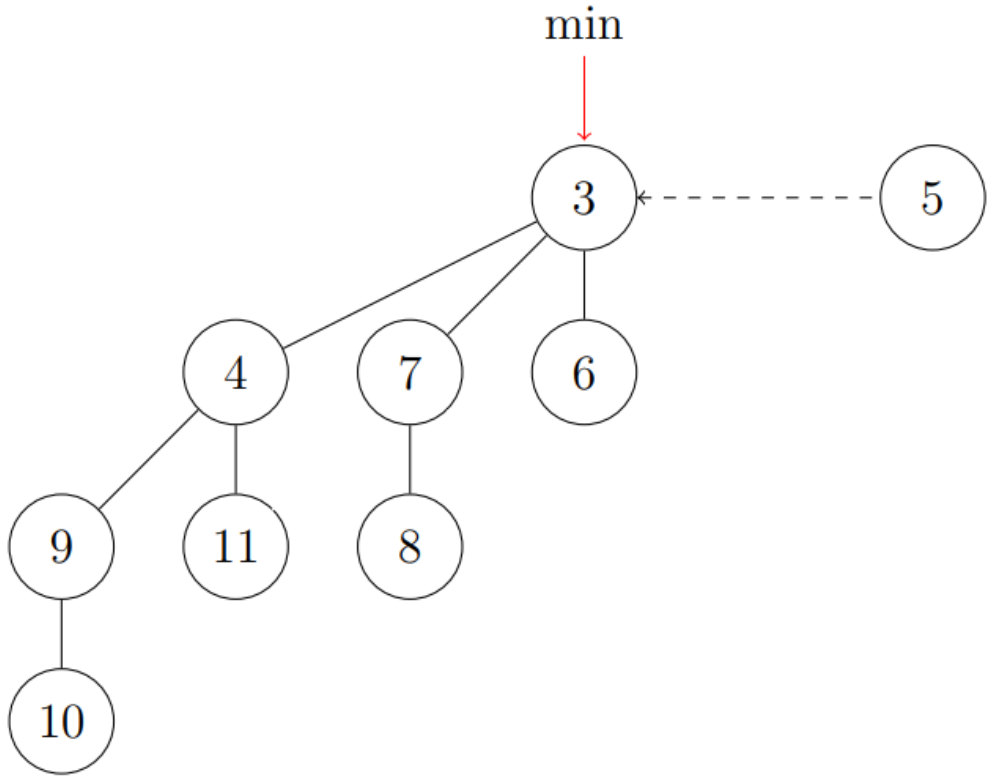
q) remove(38):



Wiederholung: Binomialheaps

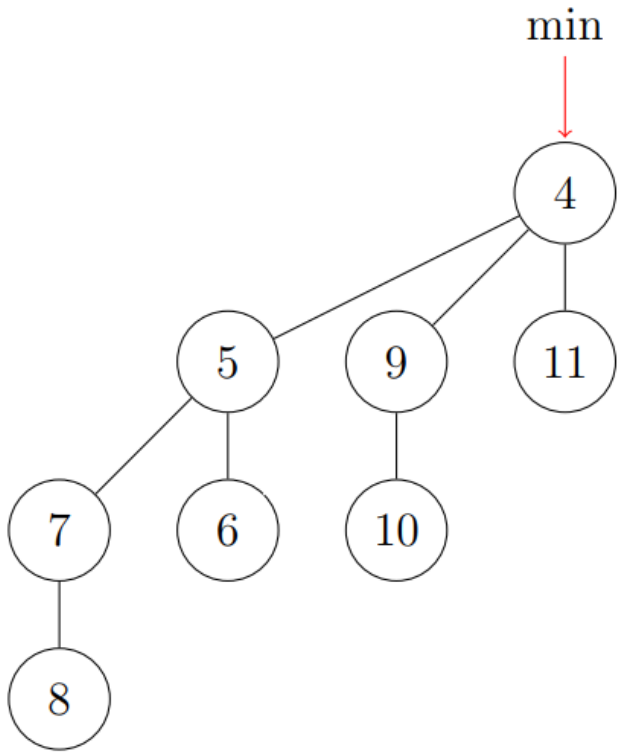
Aufgabe 11.3 – Rückblick I: Binomialheaps

Führen Sie auf dem folgenden Binomial-Heap nacheinander drei deleteMin-Operationen aus.
Zeichnen Sie nach jeder deleteMin-Operation den entstandenen Binomial-Heap.



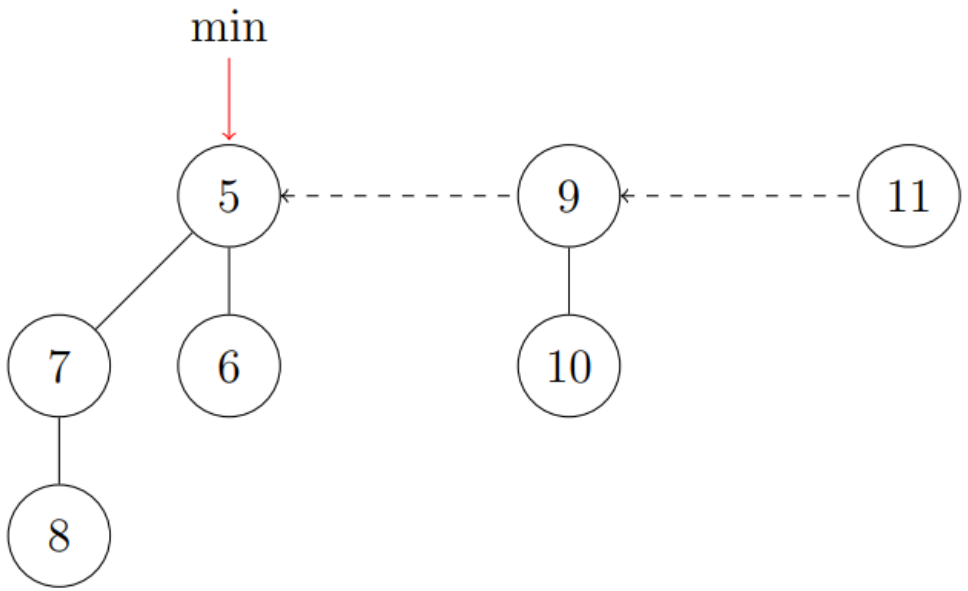
Aufgabe 11.3 – Rückblick I: Binomialheaps

Erste OP:



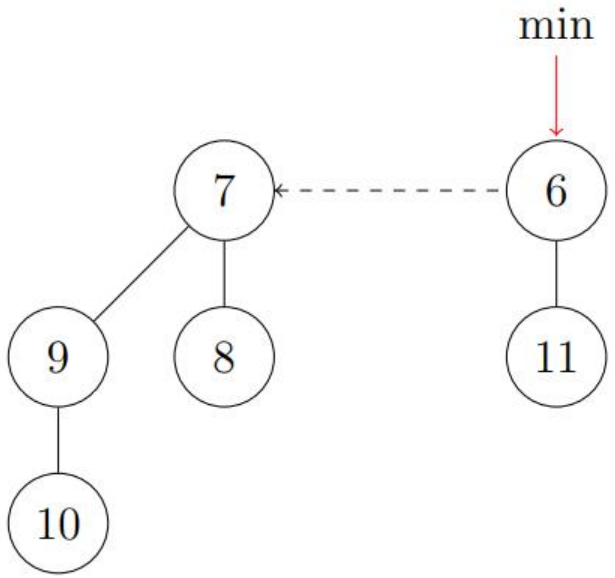
Aufgabe 11.3 – Rückblick I: Binomialheaps

Zweite OP:



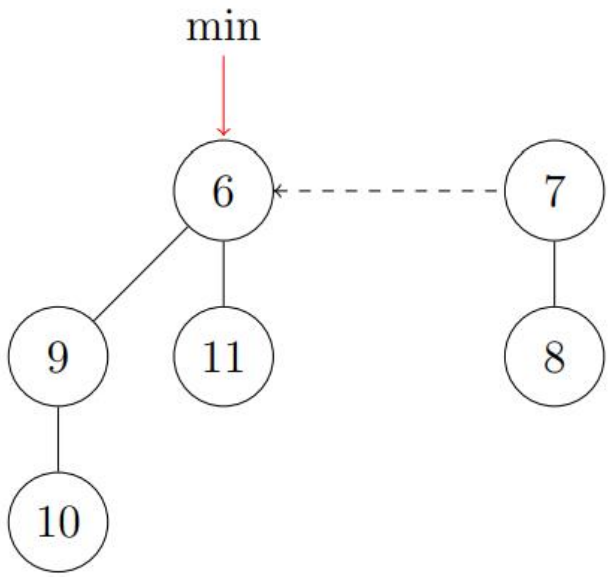
Aufgabe 11.3 – Rückblick I: Binomialheaps

Dritte OP, Variante 1:



Aufgabe 11.3 – Rückblick I: Binomialheaps

Dritte OP, Variante 2:



Aufgabe 11.3 – Rückblick I: Binomialheaps

Dritte OP, Variante 3: