Übung Algorithmen und Datenstrukturen

Sommersemester 2017

Patrick Schäfer, Humboldt-Universität zu Berlin

Agenda: Blatt 5

- Diese Woche: Vorrechnen Blatt 5 (first-come first-served)
  - Gruppe 5 13-15 Uhr https://dudle.inf.tu-dresden.de/AlgoDatGr5U5/
  - Gruppe 6 15-17 Uhr https://dudle.inf.tu-dresden.de/AlgoDatGr6U5/

- Nächste Woche: Vorrechnen Blatt 6 (first-come first-served)
  - Gruppe 5 13-15 Uhr https://dudle.inf.tu-dresden.de/AlgoDatGr5U6/
  - Gruppe 6 15-17 Uhr https://dudle.inf.tu-dresden.de/AlgoDatGr6U6/

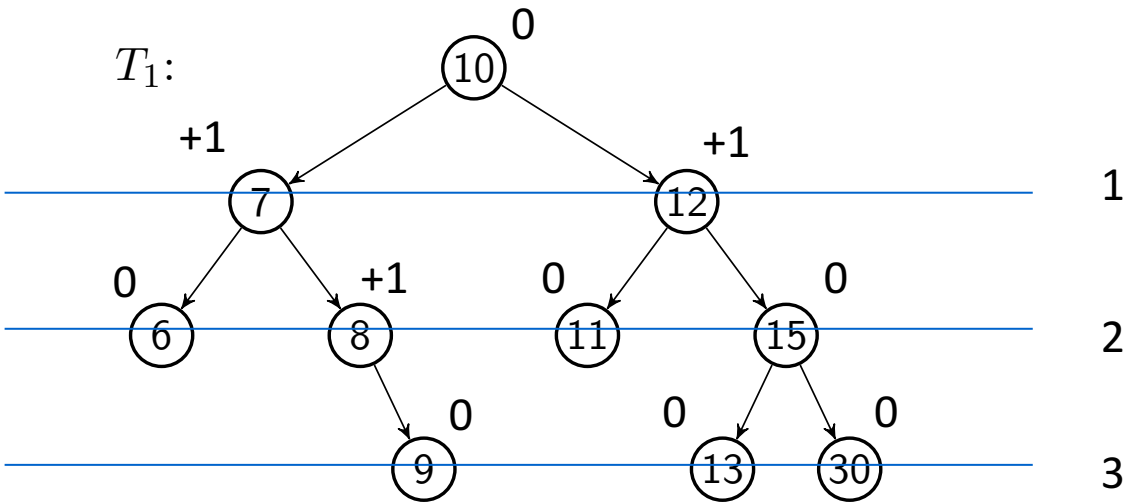
- Montag in der Vorlesung: Austeilen der Probeklausur

Übung: https://hu.berlin/algodat17

Vorlesung: https://hu.berlin/vl_algodat17

Balancefaktoren im AVL-Baum

- Geben Sie die Balancefaktoren für den Baum T1 an:
  - bal(*u*) : Differenz zwischen Höhe des rechten Teilbaums von *u* und Höhe des linkes Teilbaums von *u*



Lambda Ausrücke

```
public <T> List<T> inorder(Function<E, T> lambda) {
  // TODO: Code hier einfügen
  return new ArrayList<>();
}

inorder(p -> p.toLowerCase());
```

- Function Interface
  - It has only one method apply with the following signature:
    public R apply(T t){ }
  - It takes a generic class T and returns a generic class R.

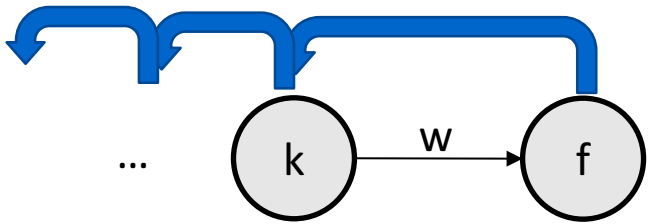
http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html#section4

Aufgabe 1.2

```
1.  G =(V,E);
2.  x : start_node;    # x∈V
3.  A : array_of_distances;
4.  ∀i: A[i]:= ∞;
5.  L := V;
6.  A[x] := 0;
7.  while L ≠ ∅
8.    k := L.get_closest_node();
9.    L:=L\k;
10.   forall (k,f,w) ∈ E do
11.     if f ∈ L then
12.       new_dist := A[k]+w;
13.       if new_dist < A[f] then
14.         A[f] := new_dist;
15.       end if
16.     end if
17.   end for
18. end while;
```

- Idee: Wir merken uns beim Aktualisieren der Distanzen in Zeile 14 zusätzlich den Vorgänger k des Knotens f.

- Damit lässt sich der Weg vom Endknoten aus nachverfolgen.

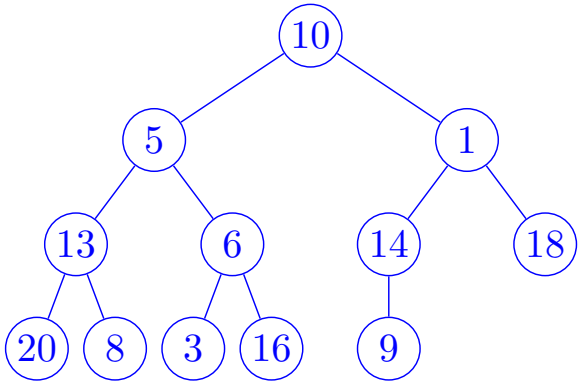


Aufgabe 1.2

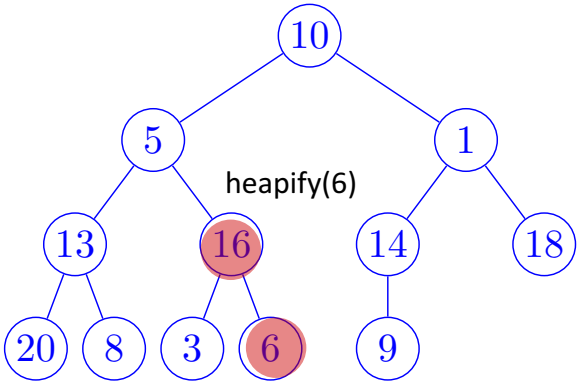
```
1.  G =(V,E);
2.  x : start_node;    # x∈V
3.  A : array_of_distances;
4.  ∀i: A[i]:= ∞;
5.  L := V;
6.  A[x] := 0;
7.  while L ≠ ∅
8.    k := L.get_closest_node();
9.    L:=L\k;
10.   forall (k,f,w) ∈ E do
11.     if f ∈ L then
12.        new_dist := A[k]+w;
13.        if new_dist < A[f] then
14.           A[f] := new_dist;
15.        end if
16.     end if
17.   end for
18. end while;
```

```
1.  G =(V,E);
2.  x : start_node;    # x∈V
3.  A : array_of_distances;
4.  P : array_of_node_pointers;
5.  ∀i: A[i]:= ∞; P[i] = NULL;
6.  L := V;
7.  A[x] := 0;
8.  while L ≠ ∅
9.    k := L.get_closest_node();
10.   L:=L\k;
11.   forall (k,f,w) ∈ E do
12.     if f ∈ L then
13.        new_dist := A[k]+w;
14.        if new_dist < A[f] then
15.           A[f] := new_dist;
16.           P[f] := k;
17.        end if
18.     end if
19.   end for
20. end while;
```

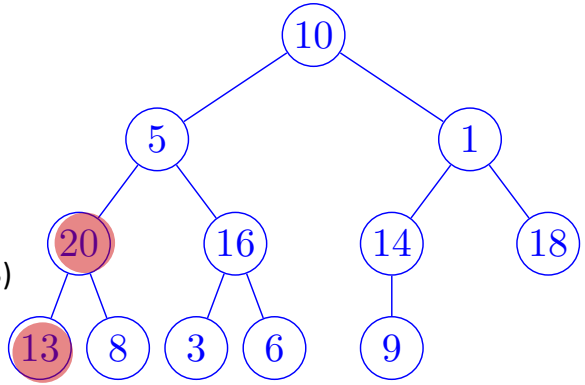
Aufgabe 2.1



initial

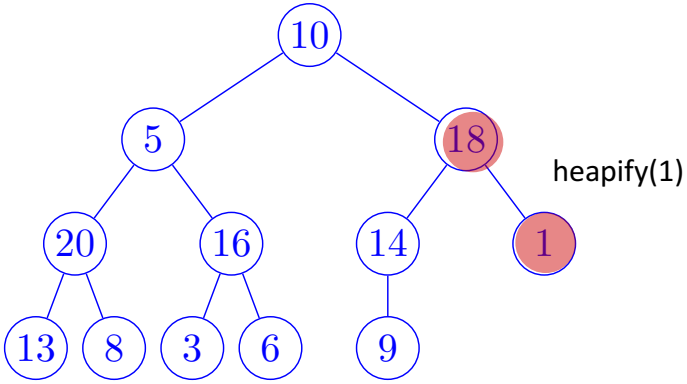


swap 1



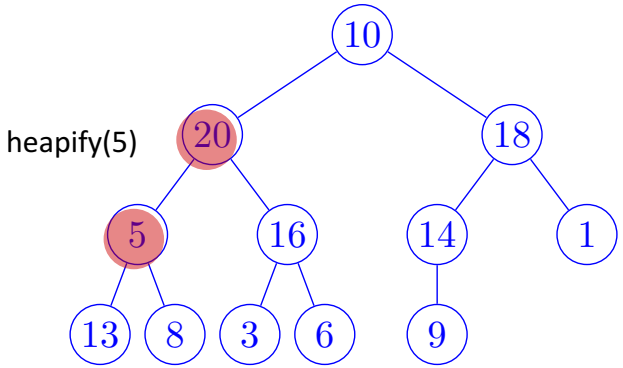
heapify(13)

swap 2

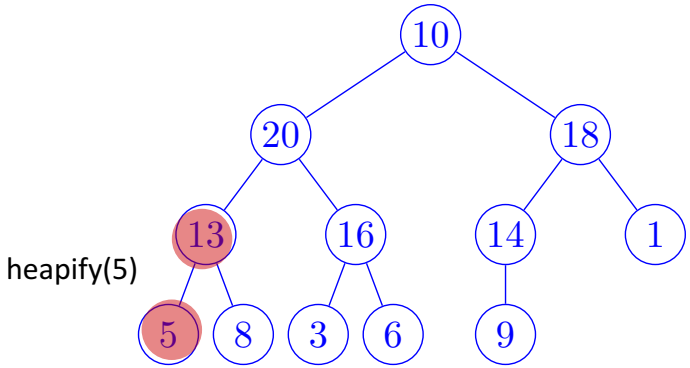


swap 3

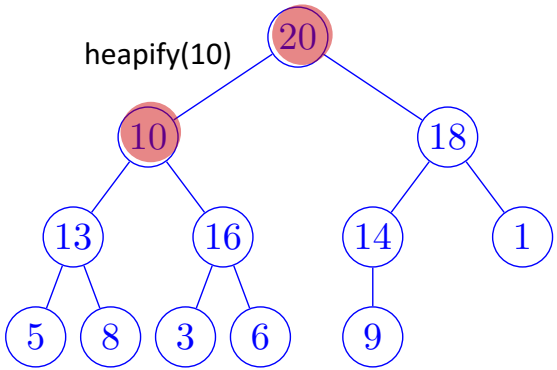
Aufgabe 2.1



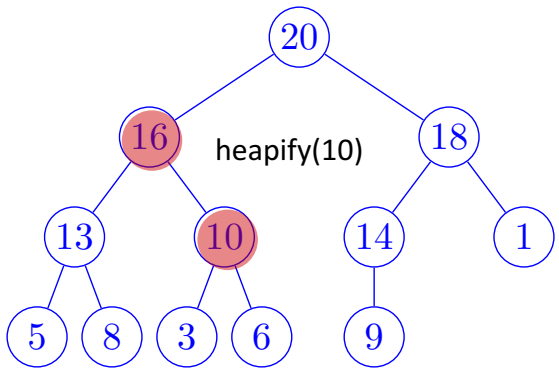
swap 4



swap 5



swap 6



swap 7

$$L(32) = 6, 8, 4, 0, 9, 1, 5, 2, 10, 3, 7 \qquad L(5) = 1, 9, 5, 3, 10, 4, 7, 8, 6, 0, 2$$
$$L(43) = 6, 0, 4, 9, 7, 10, 1, 2, 5, 8, 3 \qquad L(60) = 2, 5, 0, 9, 3, 1, 8, 4, 7, 10, 6$$
$$L(16) = 2, 10, 9, 3, 0, 8, 7, 6, 1, 4, 5 \qquad L(54) = 1, 9, 6, 0, 4, 5, 10, 2, 8, 7, 3$$
$$L(26) = 6, 9, 4, 5, 2, 8, 10, 1, 3, 7, 0$$

- Uniformes Hashing:

0	1	2	3	4	5	6	7	8	9	10	**Hash-Werte**
						32					Einfügen 32
43						32					Einfügen 43 (1x weiter sondiert)
43		16				32					Einfügen 16
43		16				32			26		Einfügen 26 (1x weiter sondiert)
43	5	16				32			26		Einfügen 5
43	5	16			60	32			26		Einfügen 60 (1x weiter sondiert)
43	5	16		54	60	32			26		Einfügen 54 (4x weiter sondiert)

Bei diesem Verfahren wird ein Element immer an der ersten freien Position seiner Permutationsliste in die Hashtabelle eingefügt.