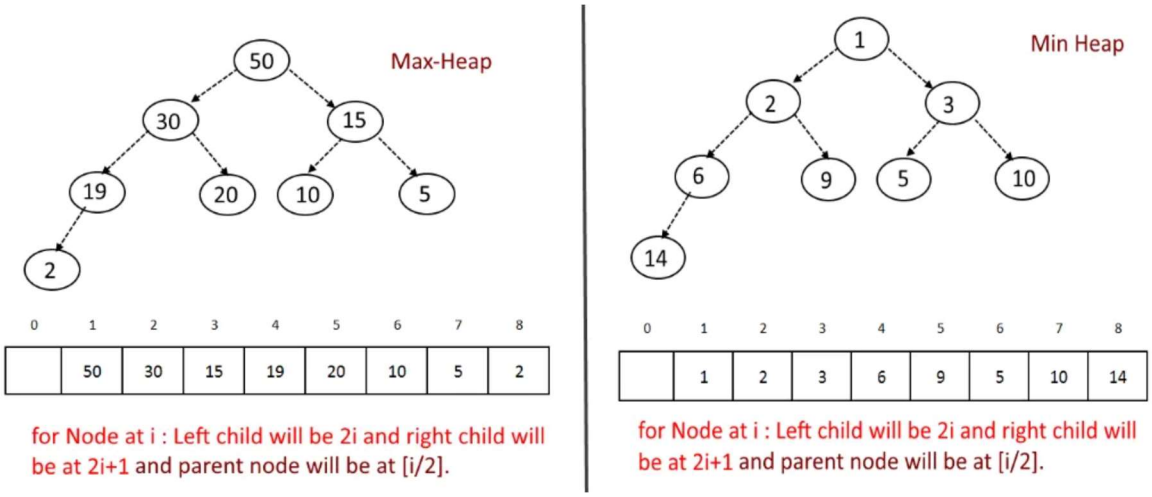
[A02] HEAP



Was ist ein Heap?
Ist eine Datenstruktur, welche sich für das Sortieren von Daten oder für eine Prioritätswarteschlange eignet. Es können Elemente gesammelt, abgelegt und auch wieder entnommen werden.

Heap Baum
Ein Heap lässt sich in Form eines Baumes oder eines Arrays darstellen. Es gibt z.B. einen binären Heap der sich als Binärbaum darstellen lässt. Jeder Knoten darf höchstens aus zwei Kindern bestehen. Wie bei einem Baum wächst es von der Wurzel (ganz oben) nach unten (links und/oder rechts).

Min-Heap:
Bei einem Min-Heap ist immer das Objekt mit dem minimalsten „Schlüssel“ als Wurzel eines Baumes anzutreffen. Somit ist der Elternknoten immer kleiner oder gleich dem Kindknoten.

In unserem Heap befindet sich ein Min-Heap. Bei der swim Methode geht es darum, dass die Kinder- mit den Elternknoten verglichen werden und somit nach oben „schwimmen“ oder nicht.
Bei der sink Methode ist es genau umgekehrt, hier wird geprüft, ob der Elternknoten größer ist als der Kinderknoten. Danach wird der Elternknoten mit der kleinsten Kinderknote vertauscht.

Max-Heap:
Im Gegensatz zu einem Min-Heap ist ein Max-Heap genau umgedreht. Hier ist die Wurzel des Baumes die mit dem maximalen „Schlüssel. Dadurch ist der Elternknoten immer größer oder gleich vom Kindknoten sein.

Laufzeit:
- insert() -> log(n)
- remove() -> log(n)

Wichtige Elemente:

```
private void swim(int pos) {

   if (prio(parent(pos)) < prio(pos)) // check if "swim up" is required if not return from
function // termination for recursive call
      return;

   exchange(parent(pos), pos);          // position change with parent element
   swim(parent(pos));             // recursive function call
}

private void sink(int pos) {

   if (exists(left(pos)) && exists(right(pos))) {                   // check if both child's
exist
      if (prio(pos) > prio(left(pos)) || prio(pos) > prio(right(pos))) { // check if parent is
greater than any child
         exchange(pos, minChild(pos));                          // exchange parent with
smallest child
         sink(minChild(pos));                                  // recursive function call
      }
   } else if (exists(left(pos))){             // check if child exists (due heap property only
left) // hasChildren() function another option (=harder to read for me)
      if (prio(pos) > prio(left(pos)))         // check if parent is greater than child
         exchange(pos, left(pos));           // exchange parent with child if it is greater
                                          // recursive function call not needed since "end of
tree" -> due single child -> heap property
   }
}
```