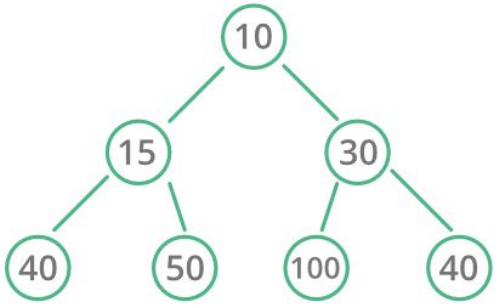
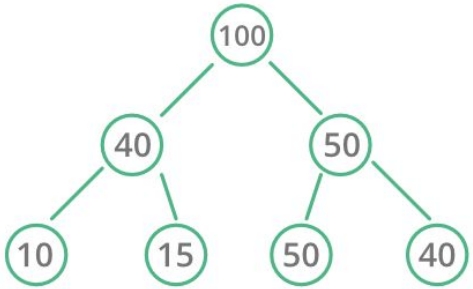
Heaps

Heaps

- **Heap:** a tree-based data structure that satisfies the heap property
- **Heap Property:**
  - **For a max heap:** the key of each node is greater than or equal to the keys of its children
  - **For a min heap:** the key of each node is less than or equal to the keys of its children



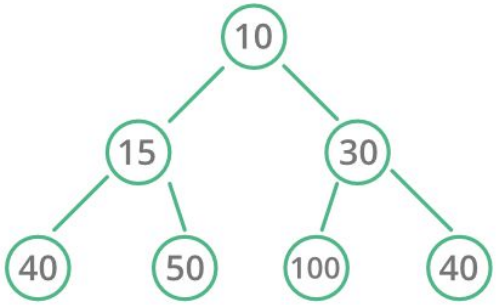
Min Heap



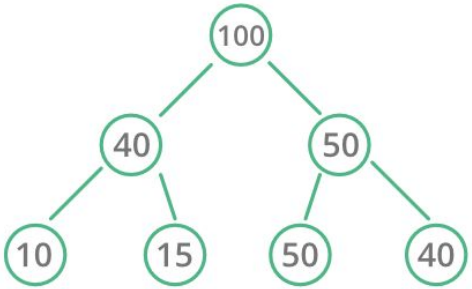
Max Heap

Heaps

- Typically implemented as **complete binary trees**
  - Each node has at most two children
  - Each level of the tree is filled up before a new level is started
- Usually stored as an **array**
- Two main operations:
  - **Insert:** Add new element, maintain heap property, time complexity O(log(n))
  - **Extract:** Remove root element (max/min) and maintain heap property, time complexity O(log(n))



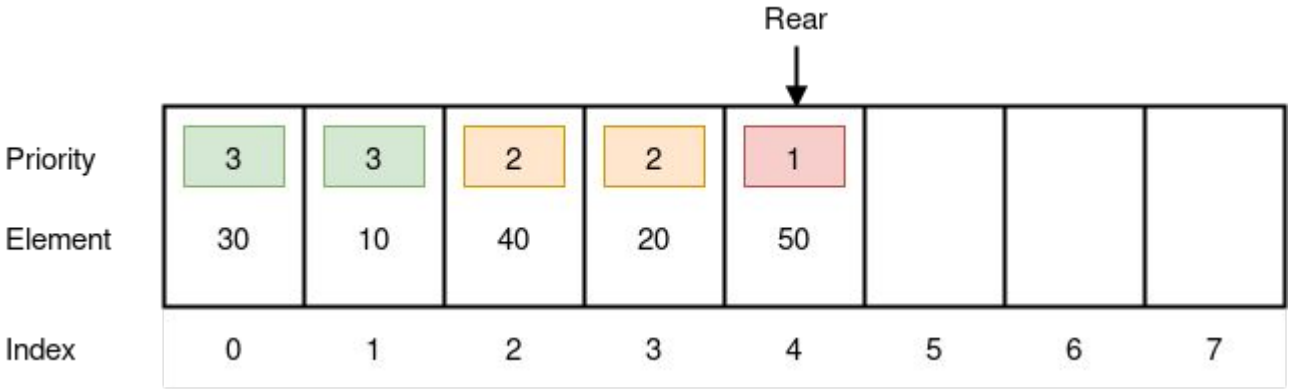
Min Heap



Max Heap

Priority Queues

- **Priority Queue:** application of a (min) heap where items are organized based on their priority
  - allows for easy access to highest priority item
- **LOWER VALUE** usually means **HIGHER PRIORITY**!



When to use Heaps/Priority Queues

- When you want quick access to the top value in a list
- When you want quick access to an item with the highest value in a particular field
- Task scheduling
- Dijkstra's algorithm
- Minimum Spanning Tree
- Huffman Coding

Using Heaps and Queues in Practice

Min Heap In Java

```
import java.util.PriorityQueue;
...
PriorityQueue<Integer> minHeap = new PriorityQueue<>();

maxHeap.add(10);  // add some elements
...
// get the smallest
System.out.println("Smallest element: " + maxHeap.peek());

// Remove the smallest
int smallest = maxHeap.pop();
```

Using Heaps and Queues in Practice

Max Heap In Java is the same, but you pass in `Collections.reverseOrder()`

```
import java.util.PriorityQueue;
import java.util.Collections;
...
PriorityQueue<Integer> maxHeap
        = new PriorityQueue<>(Collections.reverseOrder());

maxHeap.add(10);  // add elements
...
// get the largest
System.out.println("Largest element: " + maxHeap.peek());

// Remove the largest
int largest = maxHeap.pop();
```

Using Heaps and Queues in Practice

Min Heap In Python

```
import heapq

heap = []  # a heap is just a list

for i in (10, 1, 5):  # add some ints
    heapq.heappush(heap, i)

# the smallest is at index 0
print("Smallest element:", heap[0])

# remove the smallest and reheap
smallest = heapq.heappop(heap)
```

- A python heap is just a list you manipulate with the module `heapq`
- As long as you only add/remove from the list with
  - `heapq.heappop(heap)`
  - `heapq.heappush(heap)`

  `heap[0]` will always be smallest

Using Heaps and Queues in Practice

More on heaps In Python

- If you push a **tuple**, you can give items **priority**. (The first elem in the tuple is the priority)

```
for i in [(i, “Jan”), (3, “Mar”), (2, “Feb”)]:
    heapq.heappush(heap, i)
```

- To make a **max heap**, just make the priority **negative**

```
for i in (10, 1, 5):
    heapq.heappush(heap, -i)
print("Largest element:", -heap[0])
largest = -heapq.heappop(heap)
```