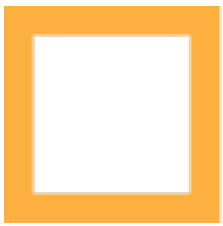
Searching Lists
Queues

Marjahan Begum and Anders Kalhauge



cphbusiness

Spring 2017

Outline

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Queues
- LIFO - Stacks
- FIFO
- Priority

Queues

- LIFO - Stacks
- FIFO
- Priority

Interface of LIFO Queues - Stacks

```
interface Stack<T> {
  void push(T item);
  T pop() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Interface of FIFO Queues

```
interface Queue<T> {
  void enqueue(T item);
  T dequeue() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Exercise 1 - Array queue

Implement the Queue inferface using an **array** data structure.

```
interface Queue<T> {
  void enqueue(T item);
  T dequeue() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Exercise 2 - Linked queue

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Implement the Queue inferface using a **linked list** data structure.

```
interface Queue<T> {
  void enqueue(T item);
  T dequeue() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Interface of Priority Queues

```
interface PriorityQueue<T extends Comparable<T>> {
  void enqueue(T item);
  T dequeue() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Priority Queue Implementations

cphbusiness
COPENHAGEN BUSINESS ACADEMY

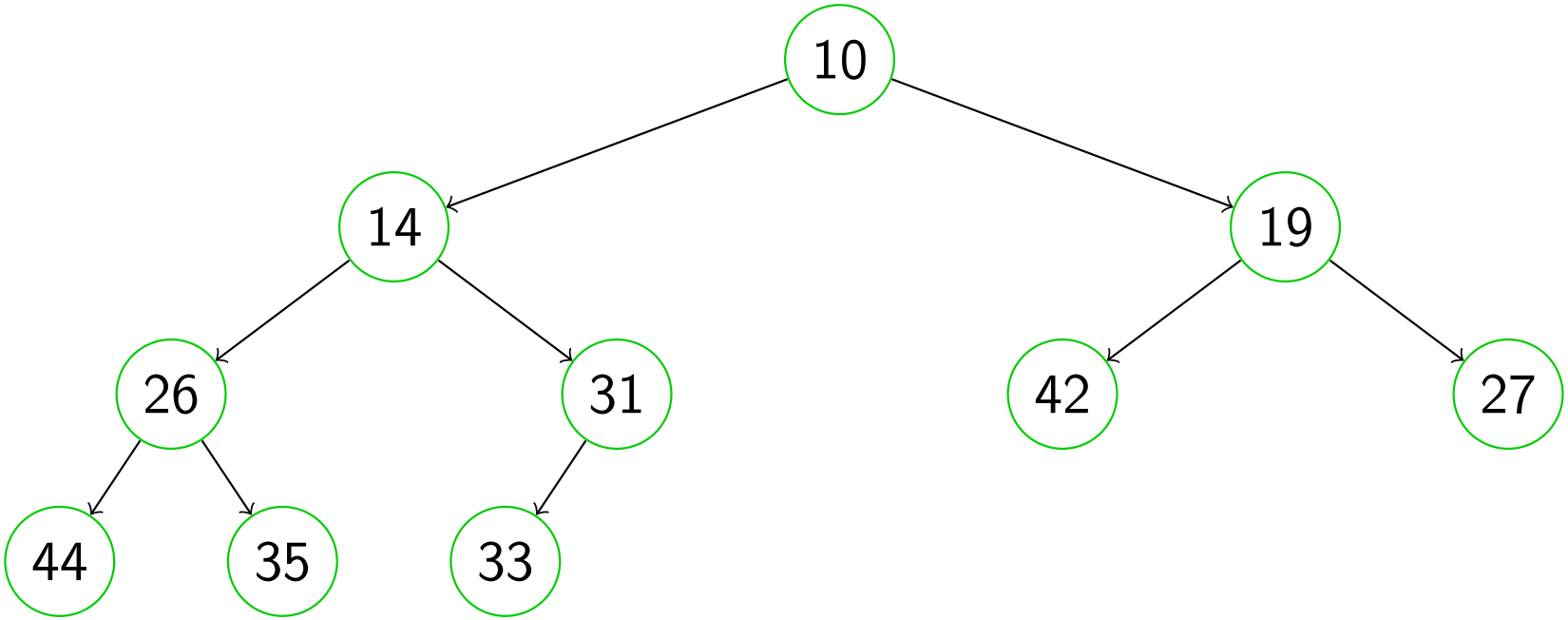
- Search for top item every time.
  - insert: $O(1)$
  - dequeue: $O(n)$
- Sort data structure, keep sorted at inserts
  - insert: $O(n)$
  - dequeue: $O(1)$
- Use a semisorted structure, a heap
  - insert: $O(\log n)$
  - dequeue: $O(\log n)$

Heaps

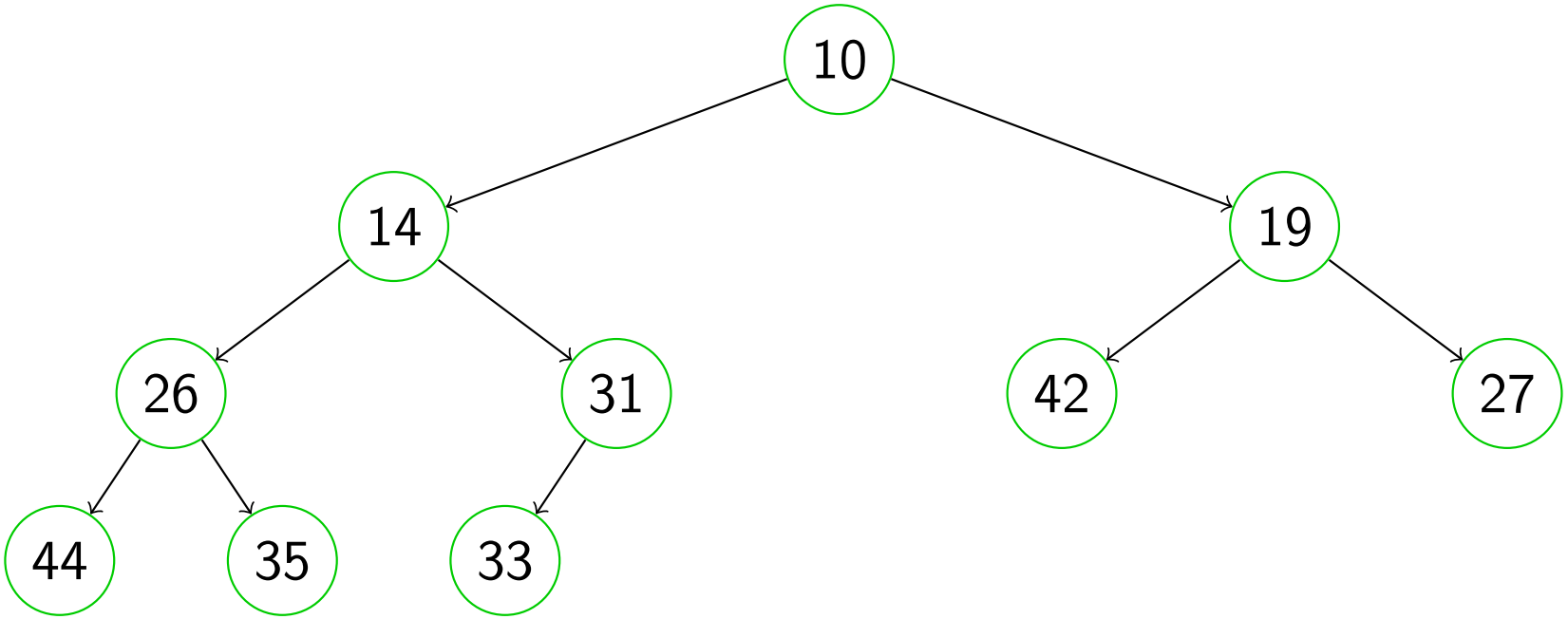
Heaps are semisorted binary trees:

- The root of a heap holds the extreme element (maximum/minimum)
- The branches of a heap are:
  - Heaps themselves
  - Empty nodes
- Heaps are balanced
- Filled from “left” to “right”

Min-Heap



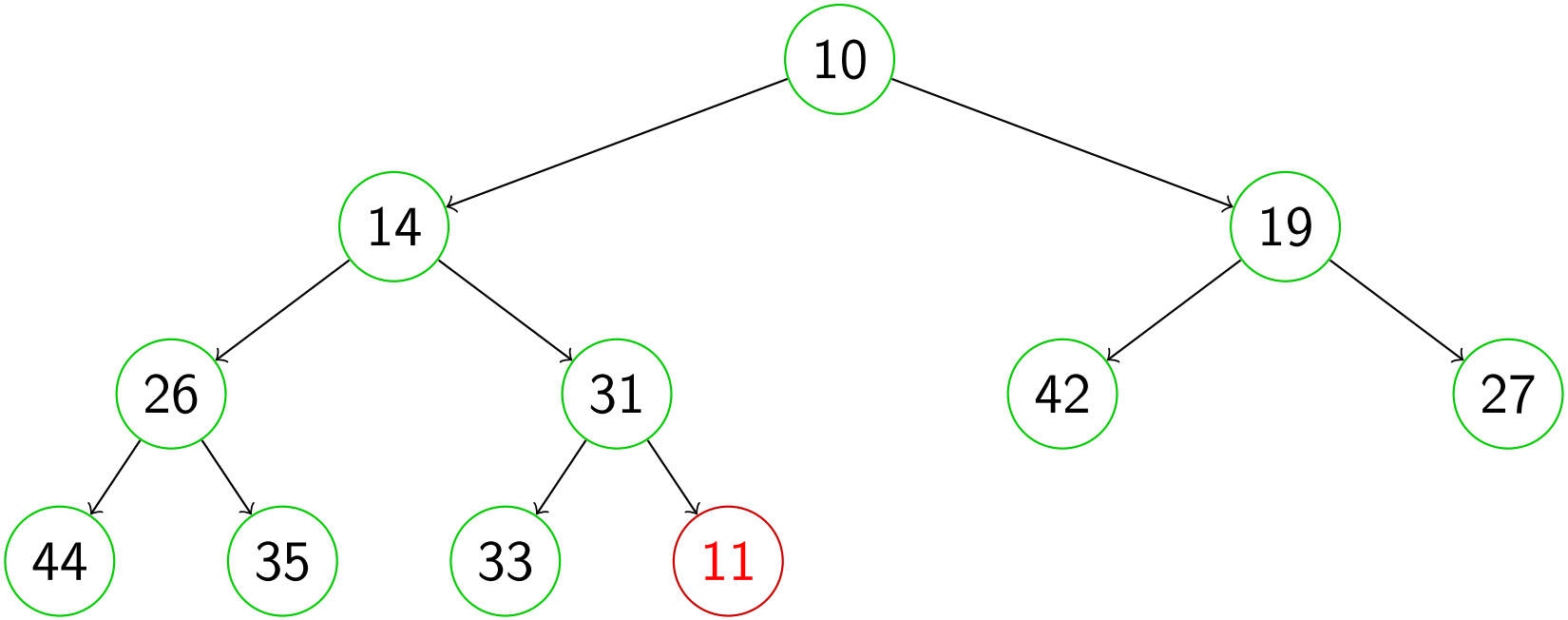
Min-Heap



10	14	19	26	31	42	27	44	35	33					

cphbusiness
COPENHAGEN BUSINESS ACADEMY

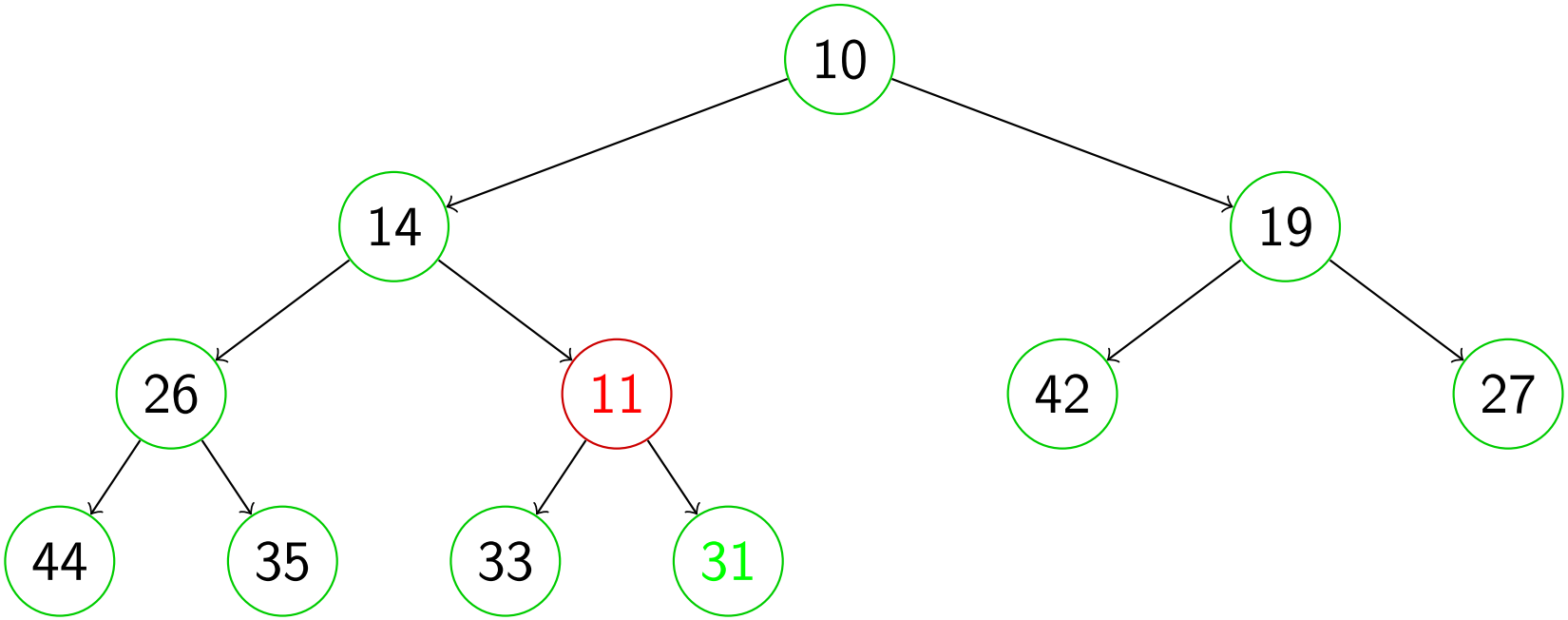
enqueue



10	14	19	26	31	42	27	44	35	33	11				

cphbusiness
COPENHAGEN BUSINESS ACADEMY

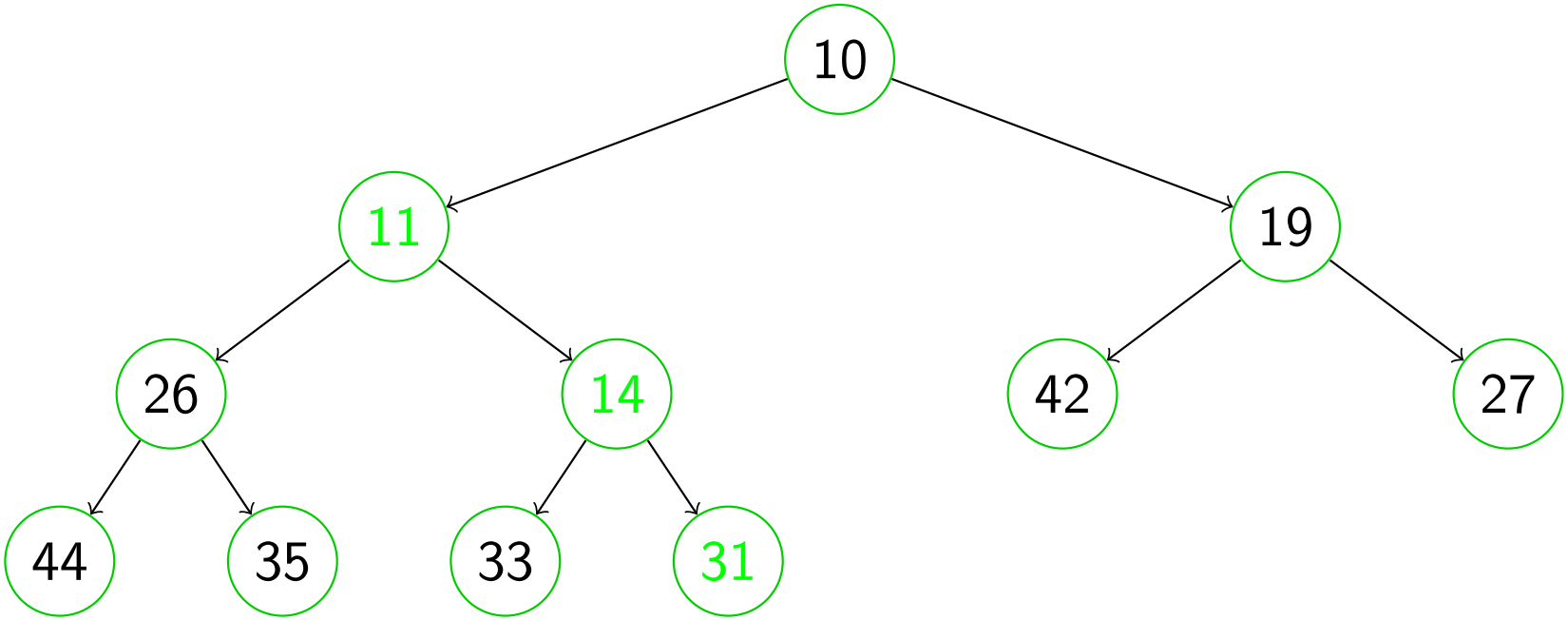
enqueue



10	14	19	26	11	42	27	44	35	33	31				

cphbusiness
COPENHAGEN BUSINESS ACADEMY

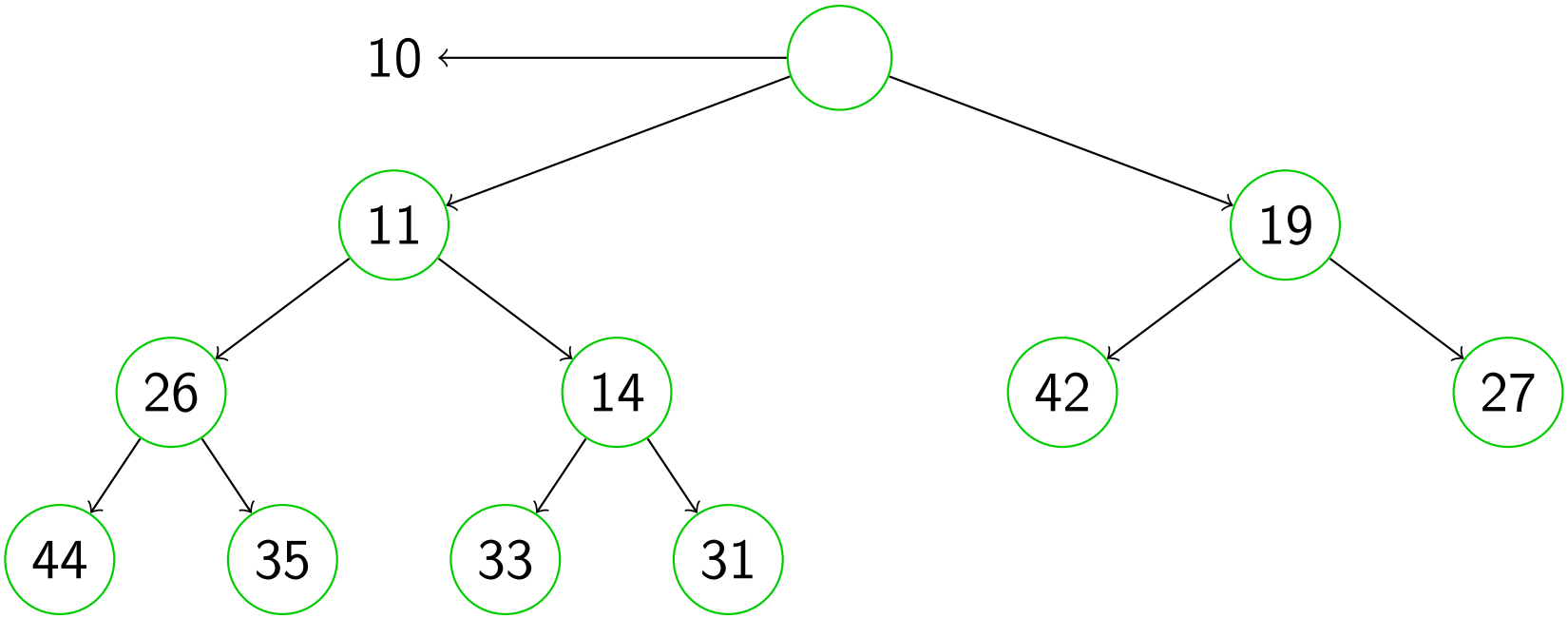
enqueue



10	11	19	26	14	42	27	44	35	33	31				

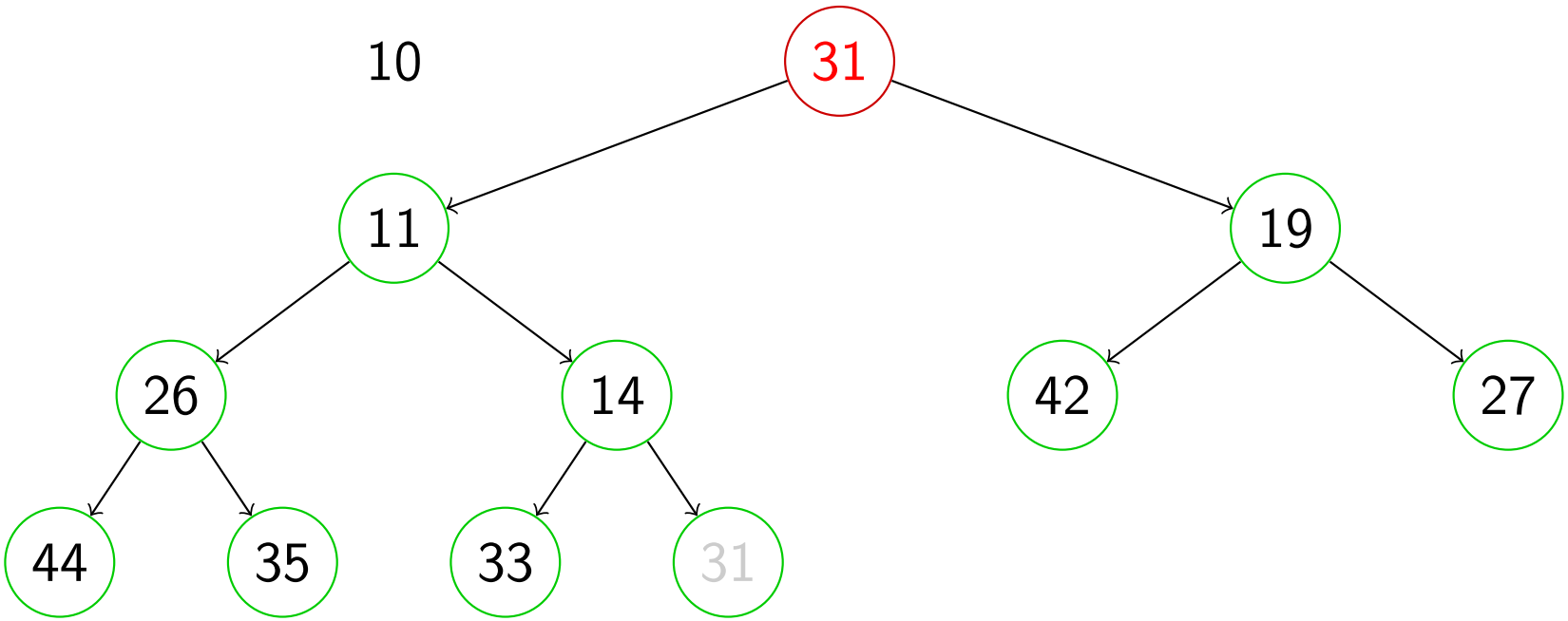
Heap

dequeue



Heap

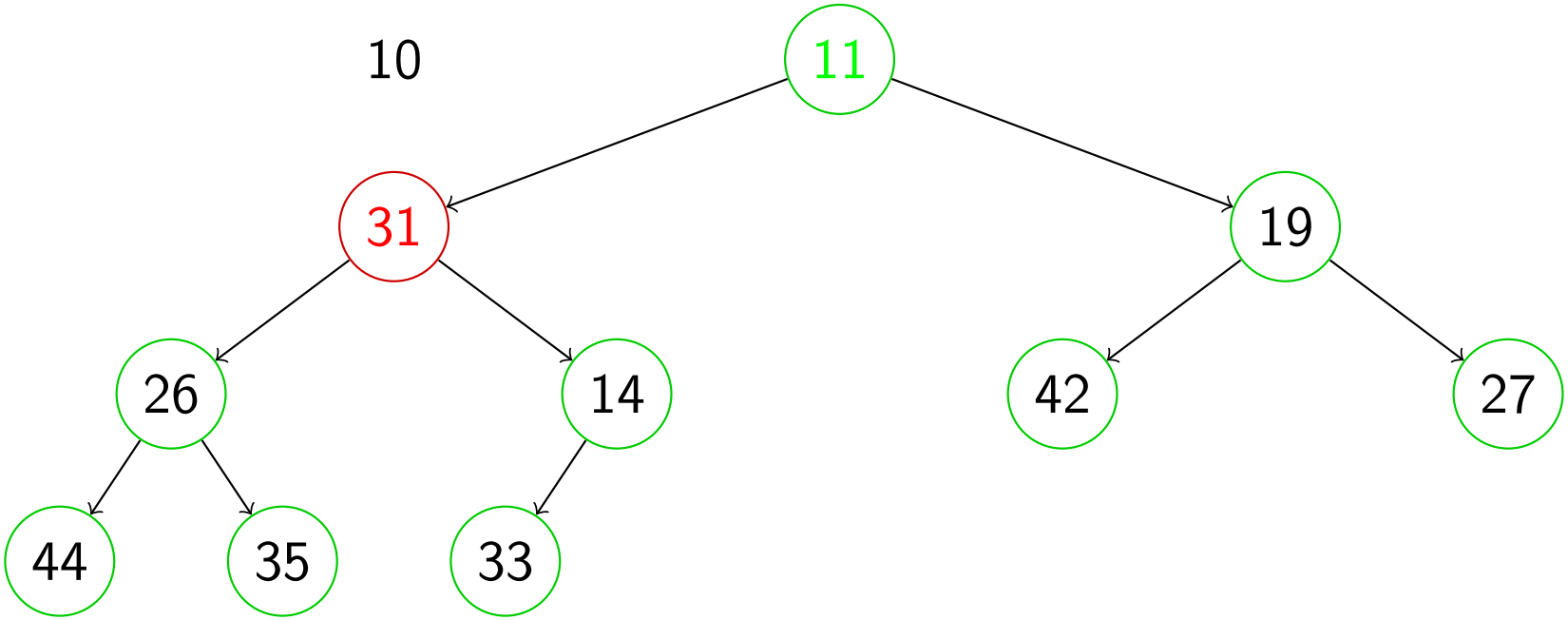
dequeue



31	11	19	26	14	42	27	44	35	33	31				

Heap

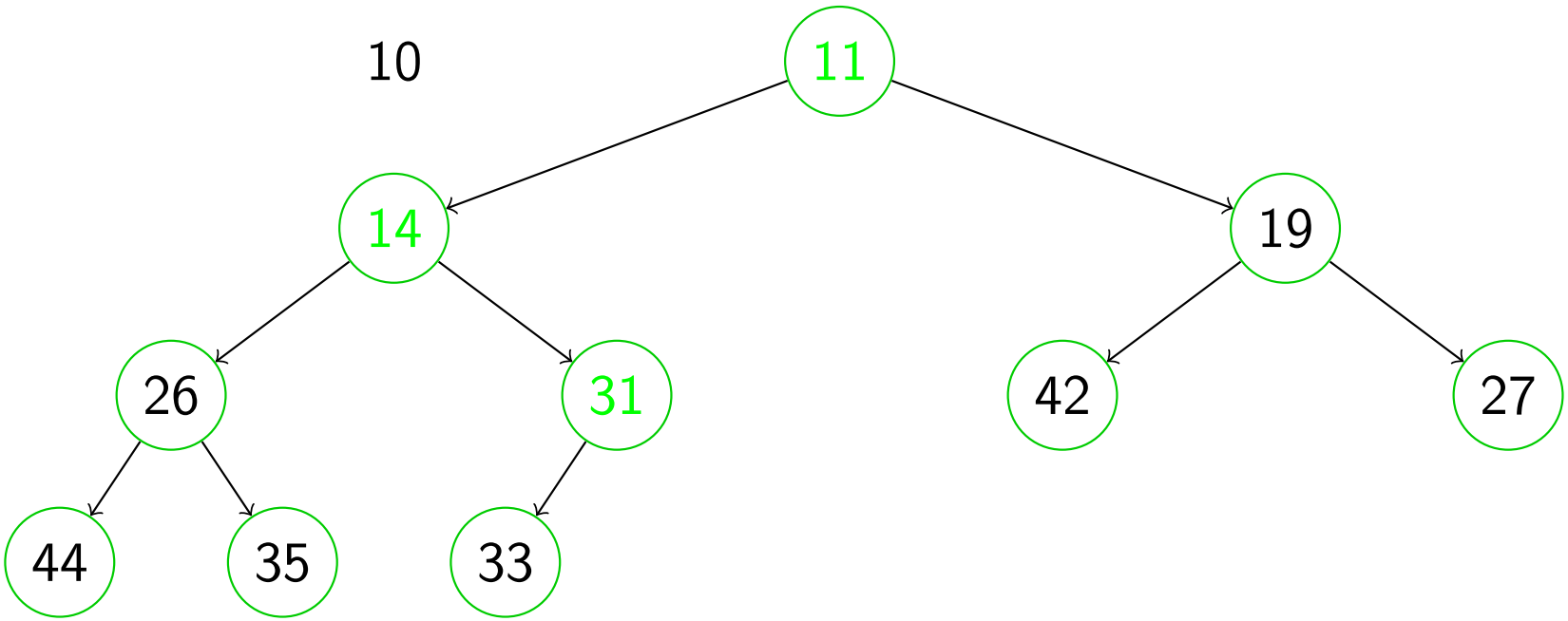
dequeue



11	31	19	26	14	42	27	44	35	33					

Heap

dequeue



11	14	19	26	31	42	27	44	35	33					