Algorithms and Data Structures

Heap Sort, Queues (revisited)

Jacob Trier Frederiksen & Anders Kalhauge

Spring 2019

Outline

Warm Up

- Www.menti.com, check.
- Hand-in Assignments, check.

Queues

- Priority Queues
- Heap Sort
- Some Pretty Animations

Hand-in Assignment #3

- Check on Content and Scope.
- Work together in Classroom (if time permits)

Warm Up

- Www.menti.com, check.
- Hand-in Assignments, check.

Queues

- Priority Queues
- Heap Sort
- Some Pretty Animations

Hand-in Assignment #3

- Check on Content and Scope.
- Work together in Classroom (if time permits)

Warm Up; a little house keeping

Weekly quiz

- Go to www.menti.com, and participate. Try to sub mit your answer *before* looking at the whiteboard.
- How did we do?

Hand-in assignment #3 (Airport Queue)

- Handed in? - Yes? - No?
- Conducted peer review on Hand-in #1? Yes? No?
- Are the assignments doable in finite time?

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Interface of Priority Queues

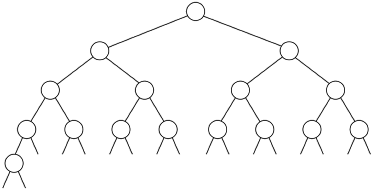
```
interface PriorityQueue<T extends Comparable<T>> {
  void enqueue(T item);
  T dequeue() throws NoSuchElementException;
  T peek() throws NoSuchElementException;
  int size();
  default boolean isEmpty() { return size() = 0; }
  }
```

Priority Queue Implementations

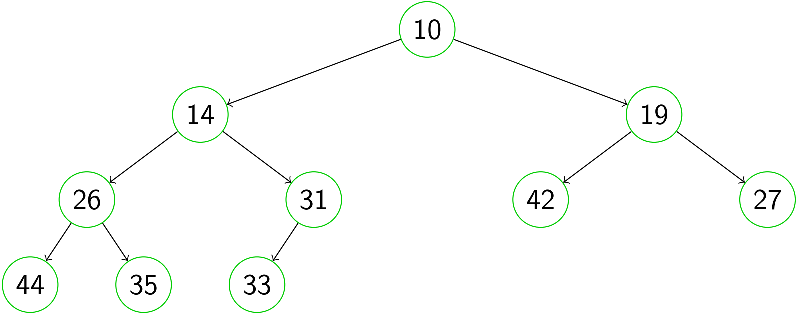
- Search for top item every time.
  - insert: $O(1)$
  - dequeue: $O(n)$
- Sort data structure, keep sorted at inserts
  - insert: $O(n)$
  - dequeue: $O(1)$
- **Use a semisorted structure, a heap**
  - **insert:** $O(\log n)$
  - **dequeue:** $O(\log n)$

Heaps are semisorted binary trees:

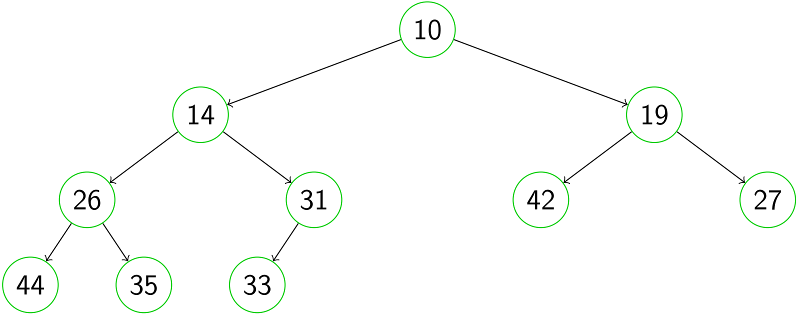
- Heap Root holds the extreme element (max/min)
- The branches of a heap are:
  - Heaps themselves
  - Empty nodes
- Heaps are balanced, they are *Complete Binary Trees*
- Filled from "left" to "right"



Min-Heap



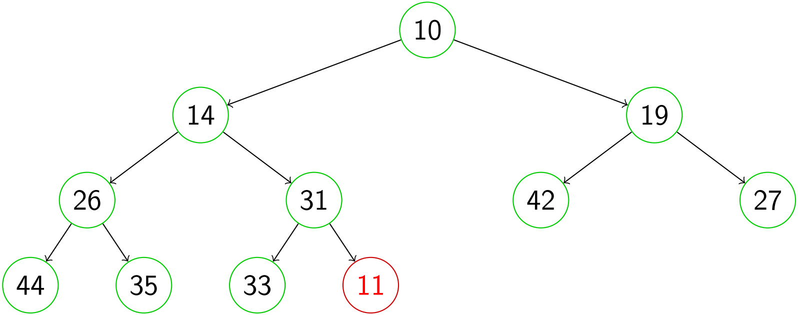
Min-Heap



10	14	19	26	31	42	27	44	35	33					

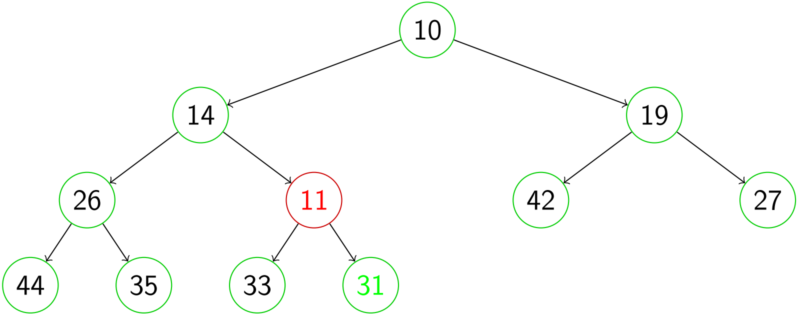
Heap

enqueue



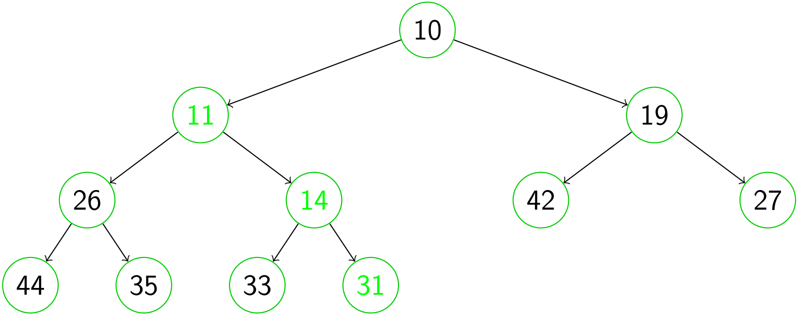
10	14	19	26	31	42	27	44	35	33	11				

enqueue



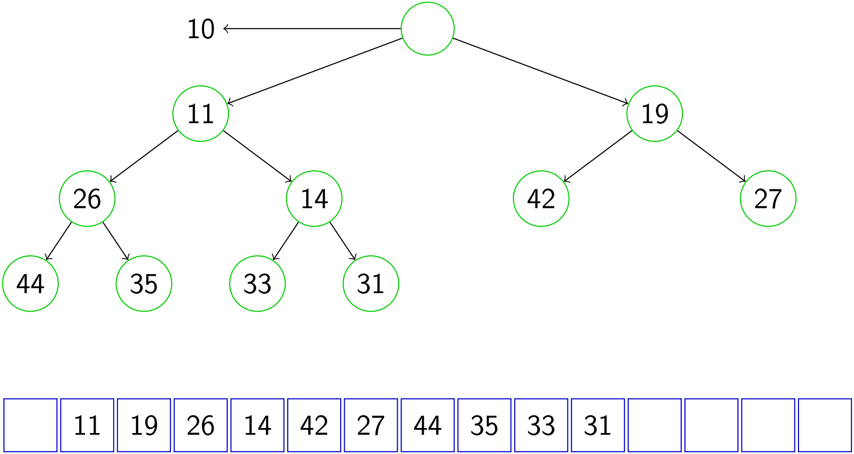
10	14	19	26	11	42	27	44	35	33	31				

enqueue

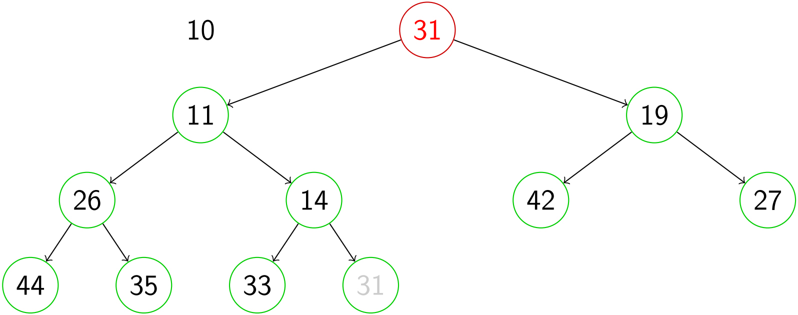


10	11	19	26	14	42	27	44	35	33	31				

dequeue

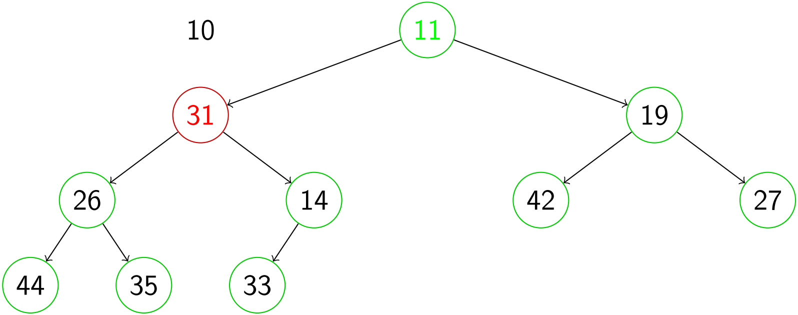


dequeue



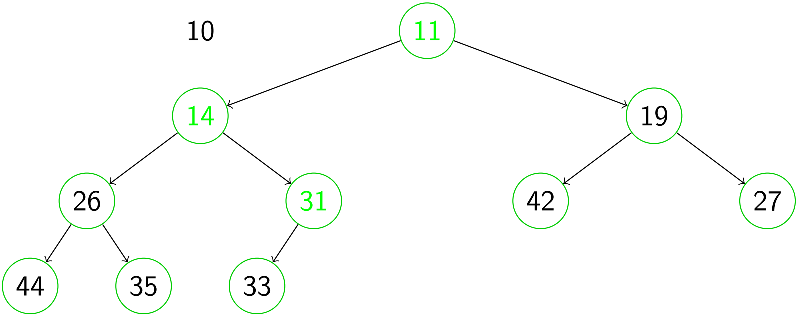
31	11	19	26	14	42	27	44	35	33	31				

dequeue



11	31	19	26	14	42	27	44	35	33					

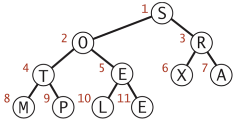
dequeue



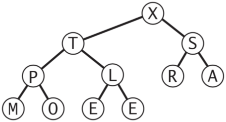
11	14	19	26	31	42	27	44	35	33					

Heap Sort is a natural consequence of the heap data structure. It has two basic parts:

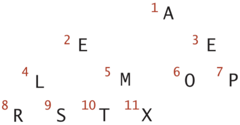
- **Heap Construction** (data are not in heap order), $\mathcal{O}(2N)$
- **Heap Sort-down**, $\mathcal{O}(2N \log N)$.



"Random" initial condition



Heap Constructed



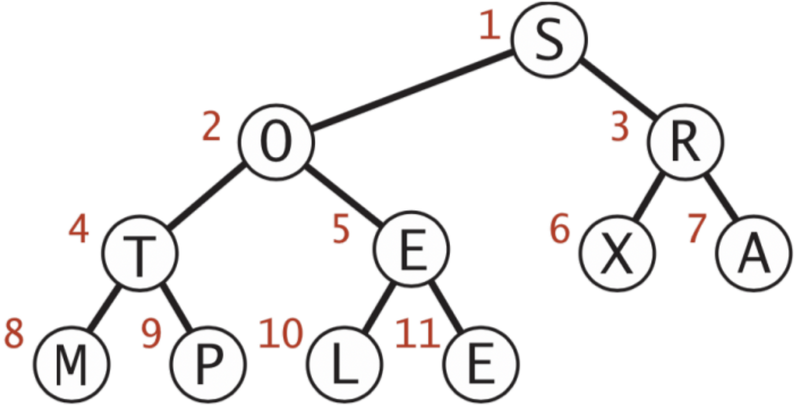
Array sorted

1	2	3	4	5	6	7	8	9	10	11
S	O	R	T	E	X	A	M	P	L	E

1	2	3	4	5	6	7	8	9	10	11
X	T	S	P	L	R	A	M	O	E	E

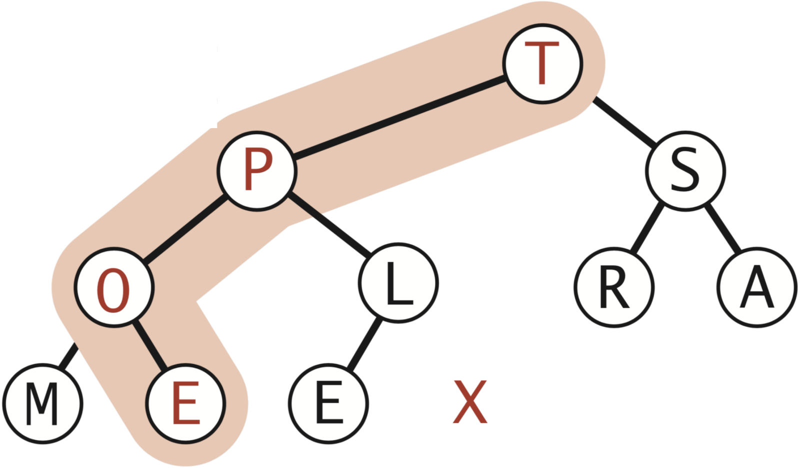
1	2	3	4	5	6	7	8	9	10	11
A	E	E	L	M	O	P	R	S	T	X

Build the heap, bottom-up. Someone do it on the blackboard?



cphbusiness
COPENHAGEN BUSINESS ACADEMY

Sort the array by exchange/deletion + sinking top, to re-heapify.



Exercise 1 - Build A Heap

By your own choice of method – i.e. pen+pencil or whiteboard or computer – build a (max-oriented) heap from the keys:

`A M U C H L O N G R E X P T H N B F.`

- Note that the characters above now are unique.
- There is some code which might help you in the repository *'./cphbusiness/algorithm/examples/....'*.
- If you use the helper code, try to make the code accept chars (instead of integers).

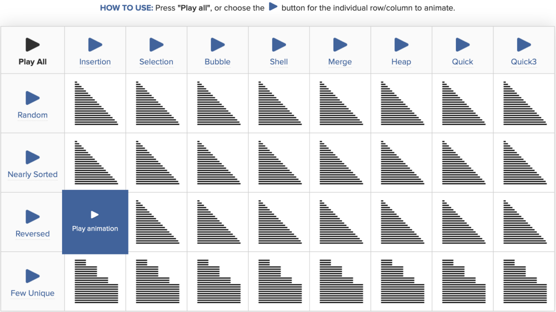
Exercise 2 - Sort a (max) Heap

By your own choice of method – i.e. pen+pencil or whiteboard or computer – sort-down the heap from Exercise 1.

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Here is a nice animation of the trace of fundamental algorithms.

https://www.toptal.com/developers/sorting-algorithms



Warm Up

- Www.menti.com, check.
- Hand-in Assignments, check.

Queues

- Priority Queues
- Heap Sort
- Some Pretty Animations

Hand-in Assignment #3

- Check on Content and Scope.
- Work together in Classroom (if time permits)

Airport Prioritized Queue

You *may* (but must not) use the template provided in the Week 09 folder on GitHub.

In groups:

Implement a priotitized queueing system for an airport. You can use any priority queue algorithm, but you must be able to argue that the time complexity is no worse than $O(\log n)$ for enqueue and dequeue respectively.

You should implement the priority queue in a setup that simulates passengers arriving to an airport, and passengers passing security.

Passenger priority can be derived from the passenger category and arrival time:

1. Late to flight
2. Business class
3. Disabled
4. Family
5. Monkey

Work together in Classroom (if time permits)