MAKE
SCHOOL

PRIORITY QUEUES & HEAPS

Heaps more fun than a barrel of monkeys

MAKE SCHOOL

PRIORITY QUEUE
APPLICATIONS

Prioritizing data packets in routers

Tracking unexplored routes in path-finding

Bayesian spam filtering

Data compression

OS: load balancing, interrupt handling

PRIORITY QUEUES

Almost always implemented with a heap

Elements with smaller numbers are higher priority

Elements are inserted in O(log n) time with a heap instead of O(n) time with a sorted array or linked list

Ordering happens during each insertion and deletion, so the cost of ordering is distributed across insertion and deletion operations instead of in one big chunk

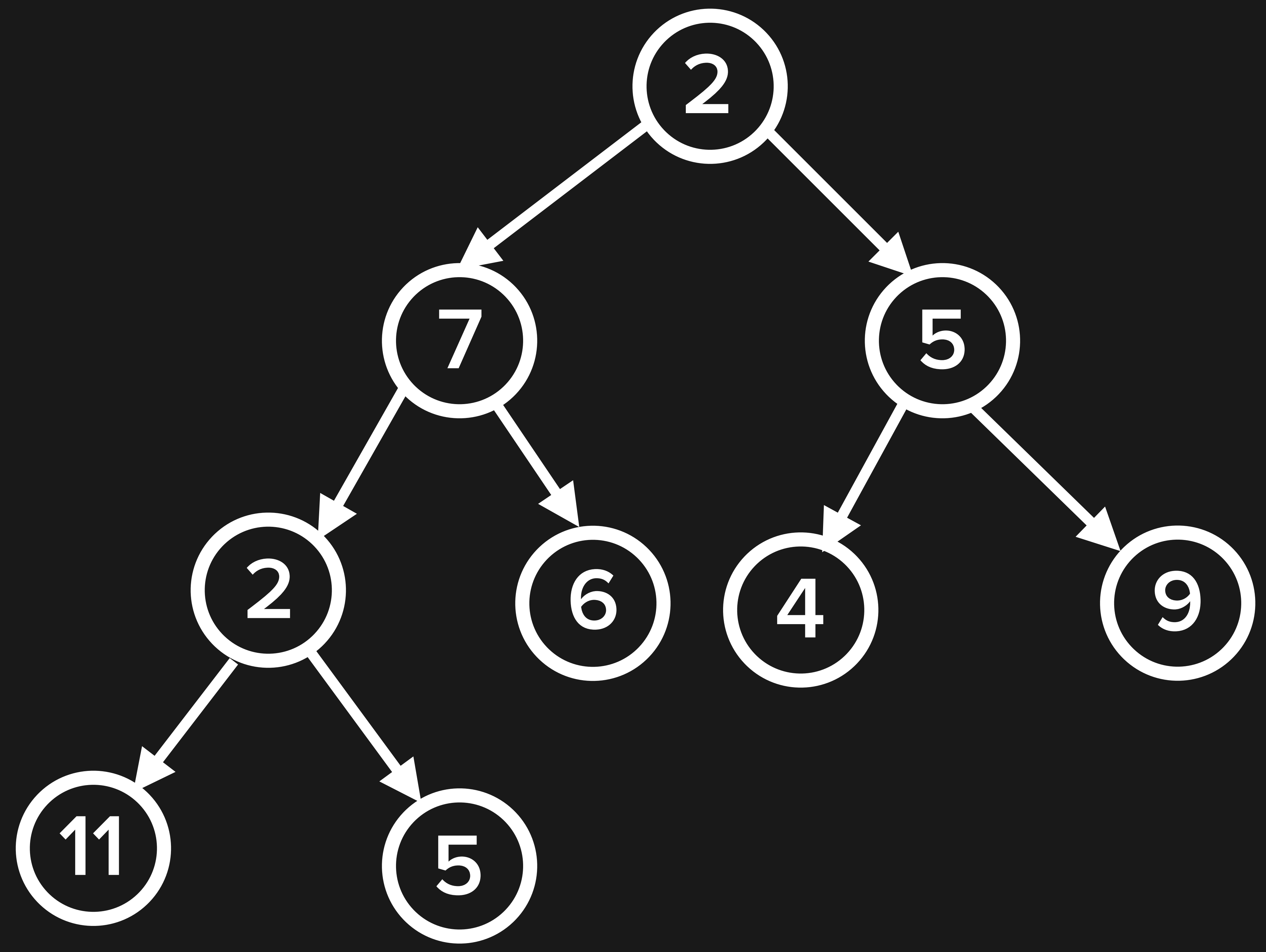
COMPLETE BINARY TREE

(REVIEW)

Every level except possibly last is completely filled and nodes are as far left as possible

Almost perfect binary tree

Height: $O(\log_2 n)$



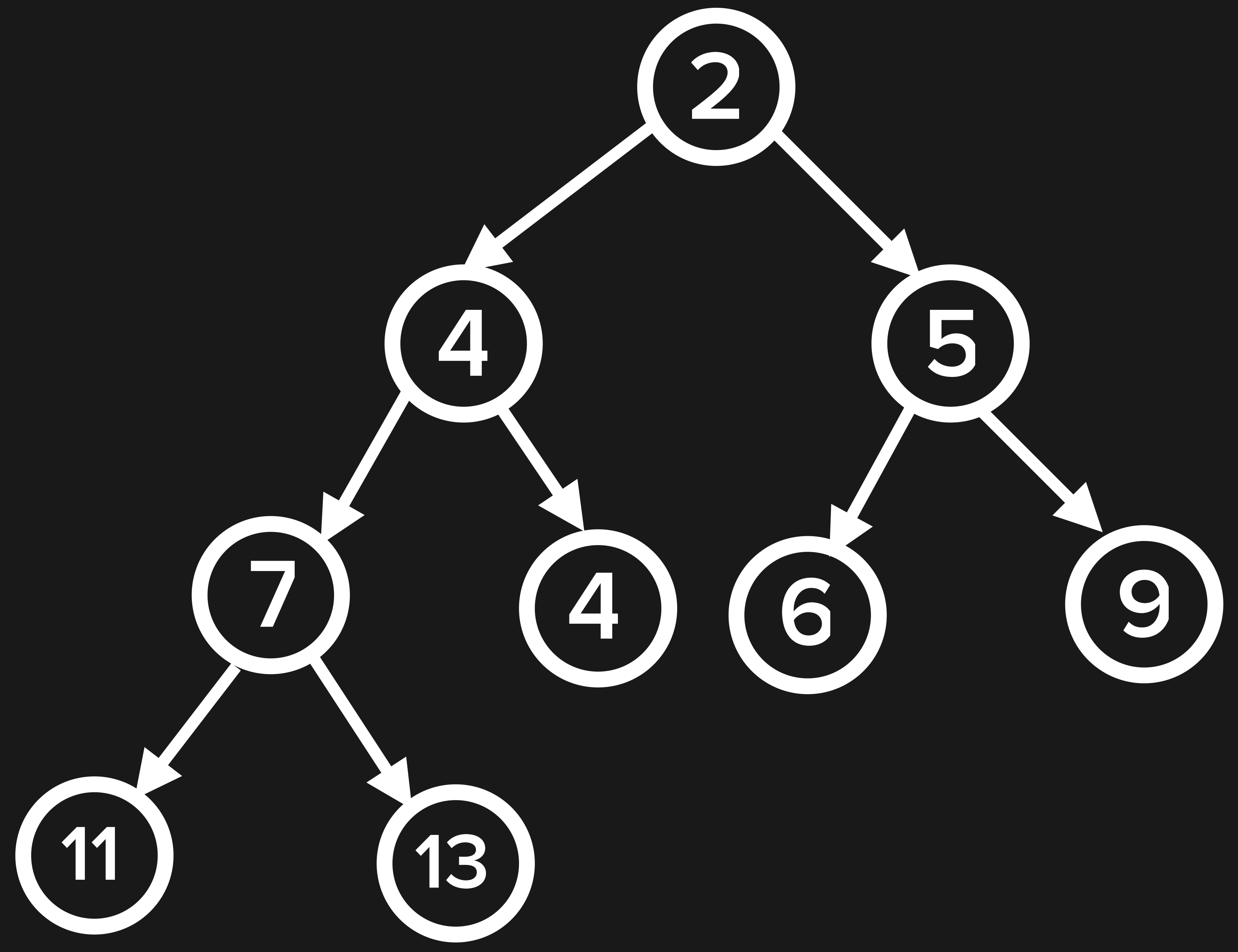
BINARY HEAP DEFINITION

A complete binary tree

Satisfies heap ordering property

min-heap - each node is greater than or equal to its parent (min value is root)

max-heap - each node is less than or equal to its parent (max value is root)

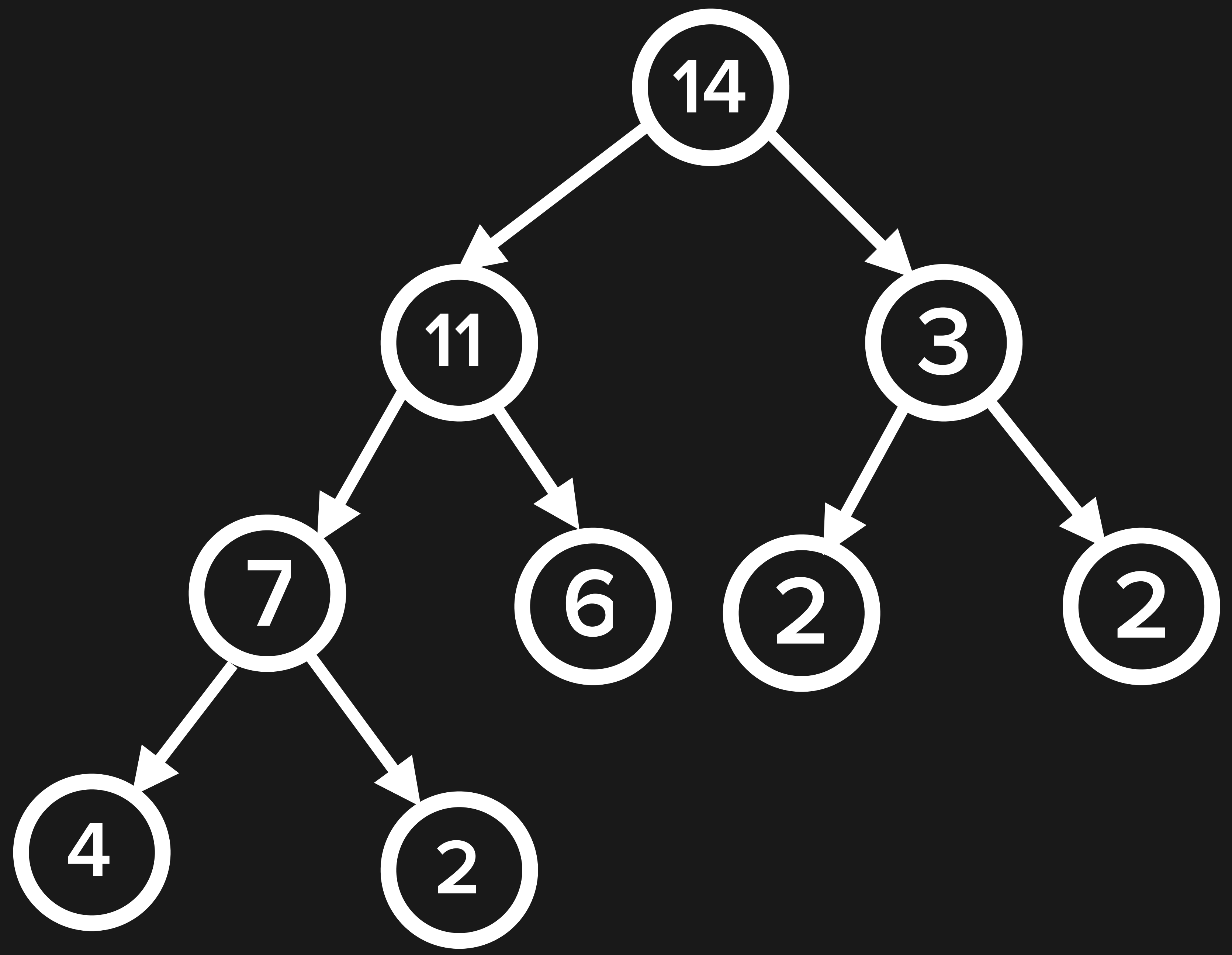


min-heap

NOTE ABOUT ORDERING

Heaps are *not* sorted, instead they are considered "partially ordered"

Max element is at the root, but we don't know where min element is, only that it must be a leaf



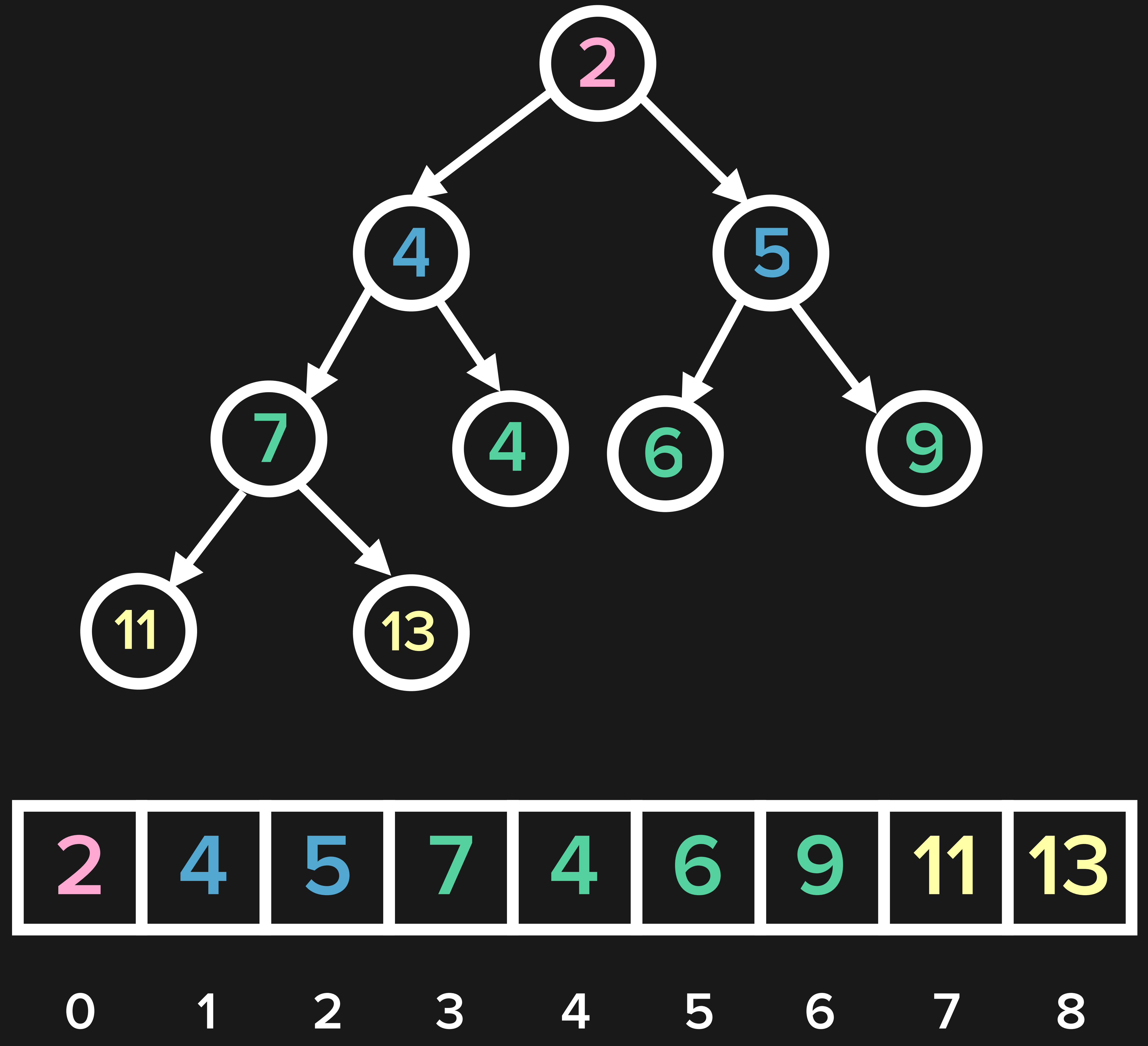
max-heap

ARRAY REPRESENTATION

Items stored in (dynamic) array following level-order traversal

Calculate parent-child index relationships with arithmetic

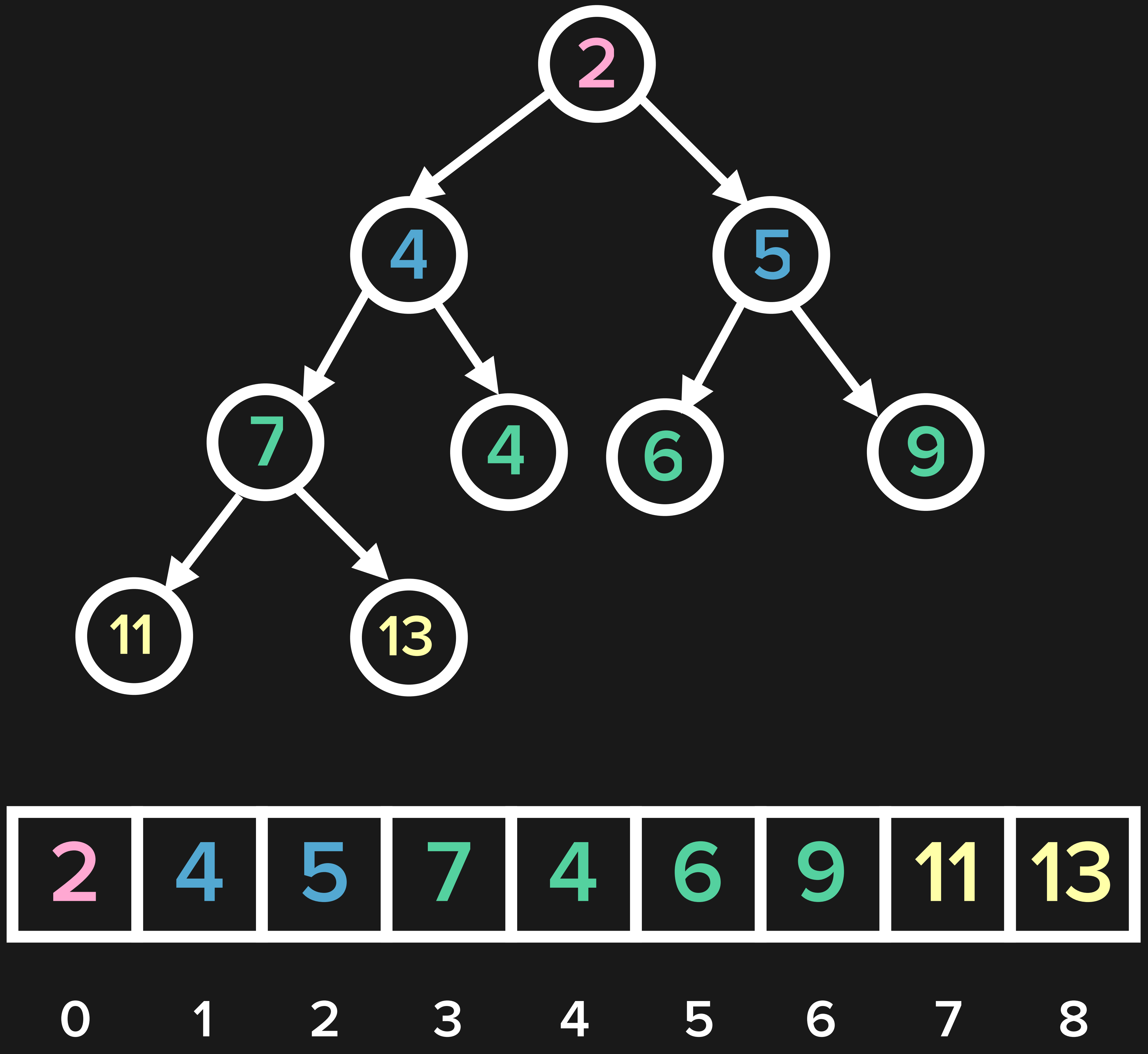
- Left child index: `2*i + 1`
- Right child index: `2*i + 2`
- Parent index: `(i – 1)/2`



ADVANTAGES

Uses less memory than binary tree represented with nodes (avoids node objects containing 3 pointers: data, left, right child)

Allows sorting an array in-place (*heap sort*)



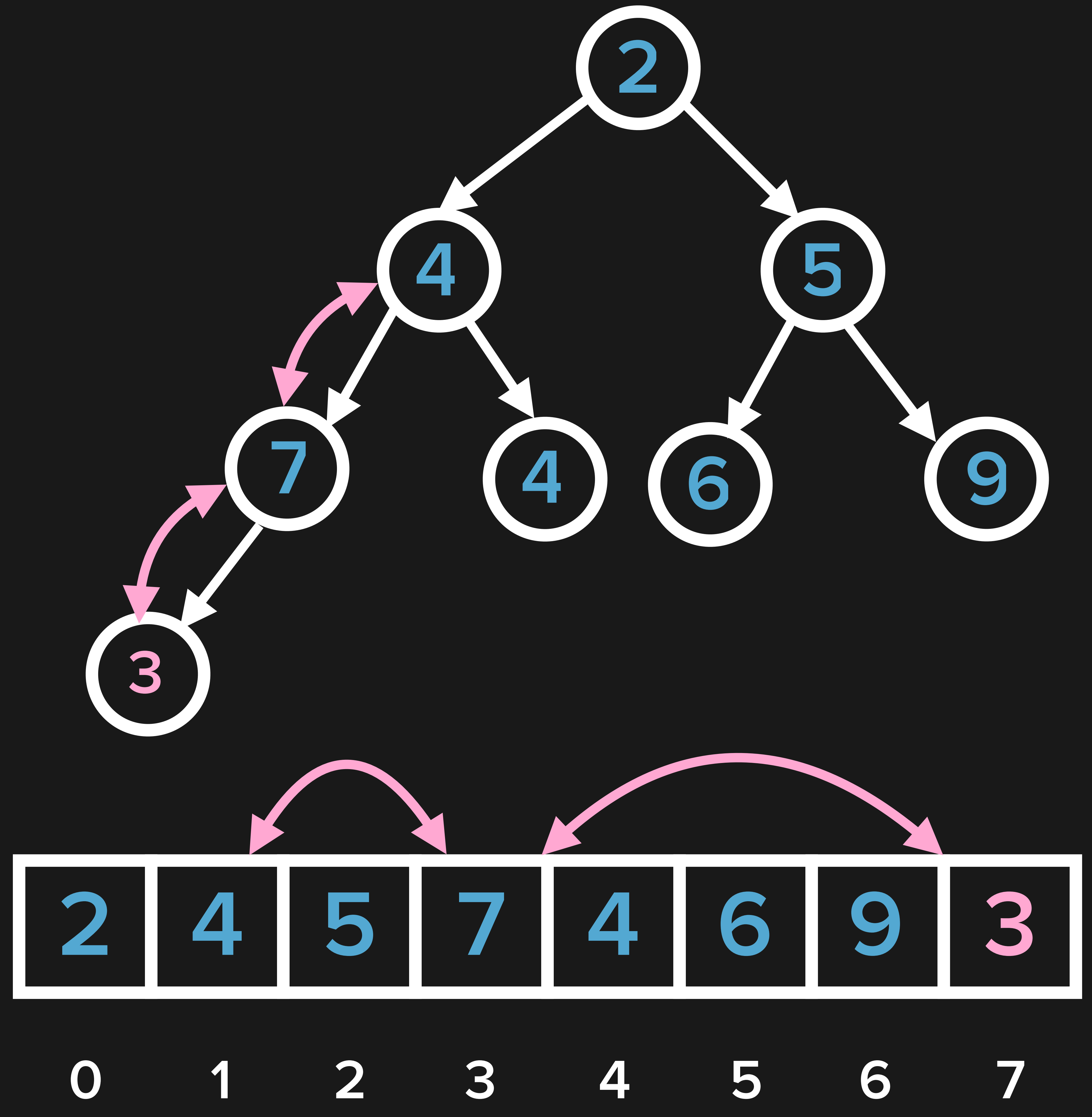
MAKE SCHOOL

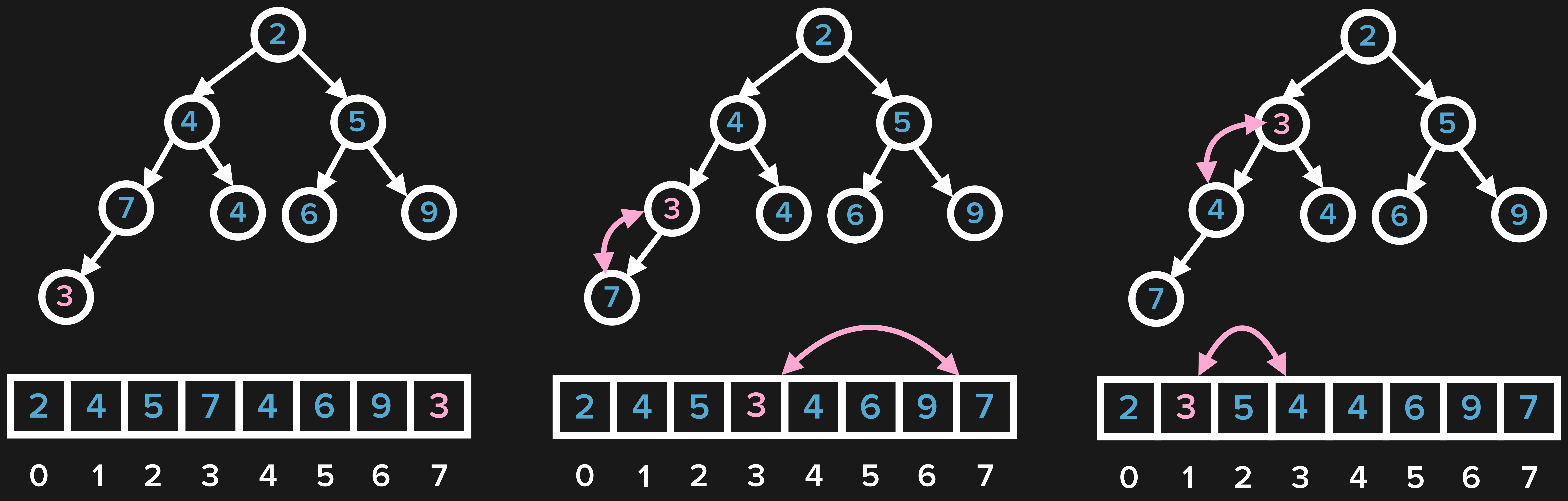
INSERT

Add element to end

Sift up (aka *bubble up*, *percolate up*, *trickle up*)

Swap with parent up to the root until path fulfills heap ordering property





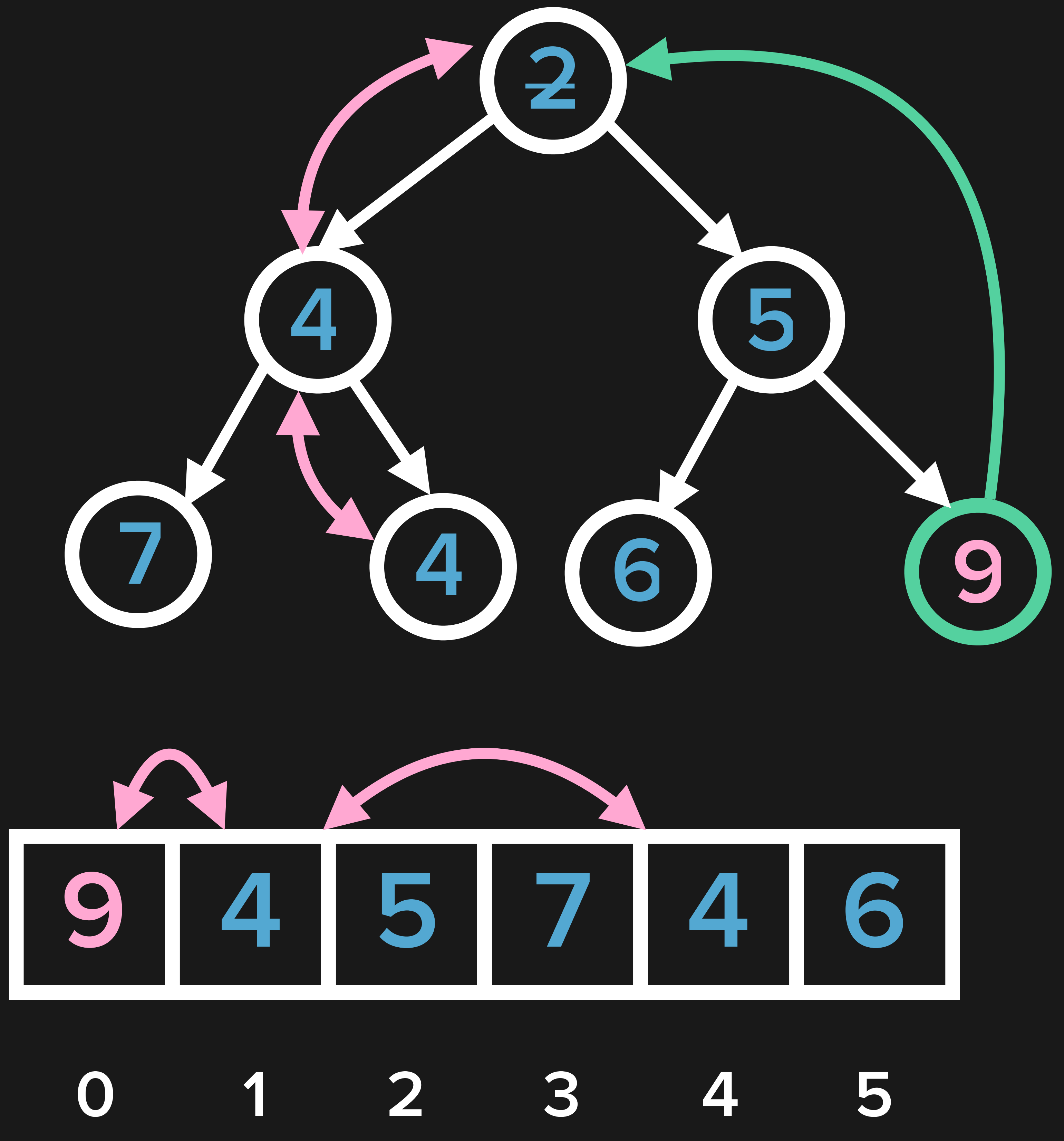
Insert 3 and sift up

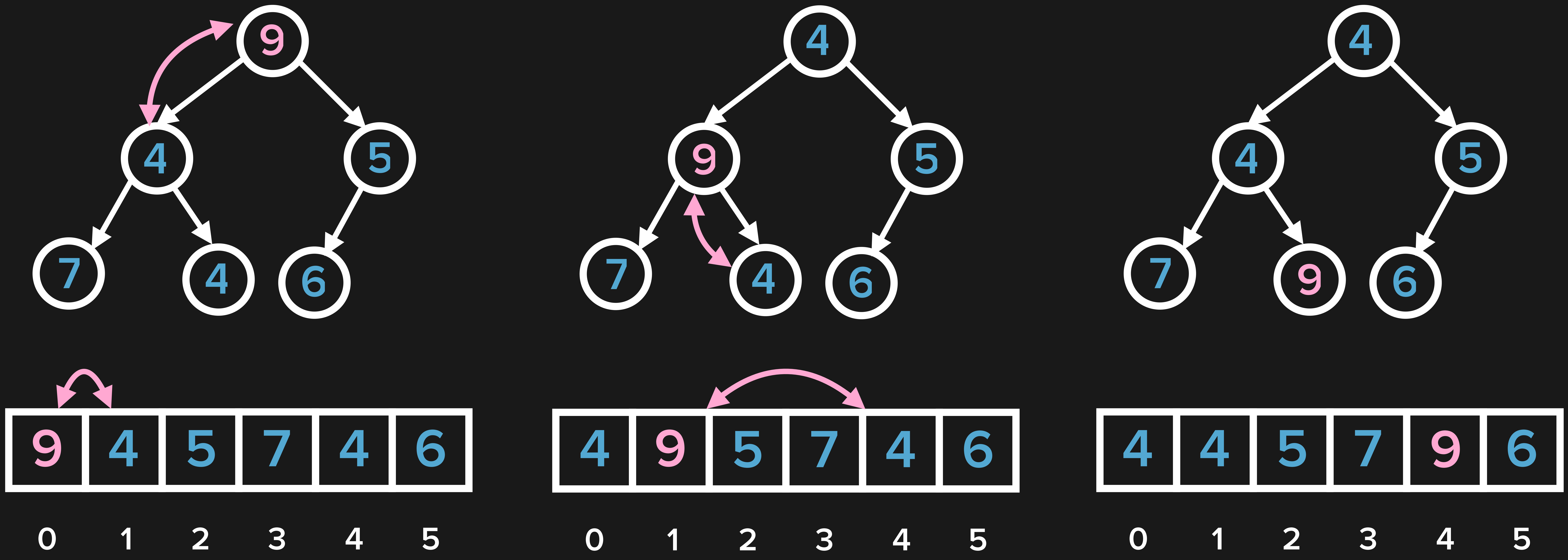
DELETE MIN / MAX

Replace root with last element

Sift down (aka *bubble down, percolate down, trickle down*)

- Swap with smaller child (min) or larger child (max) until trio fulfills heap ordering property





Delete min and sift down

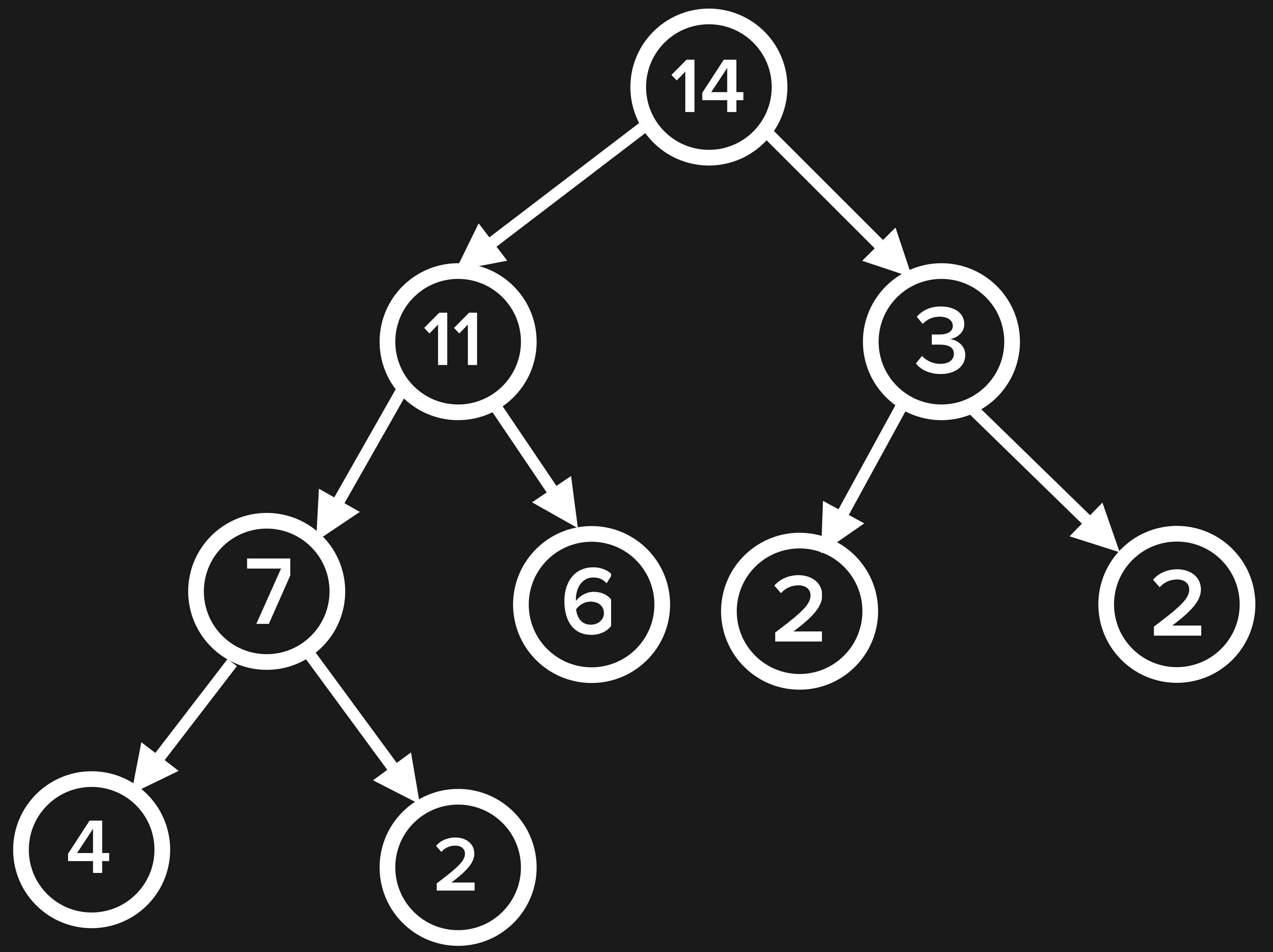
OTHER METHODS

peek (aka *find-min* or *find-max*)

returns the root value

size (aka *count* or *length*)

returns number of elements



max-heap

HEAPIFY

Input is an array (usually unsorted, unordered)

Output is an array that satisfies the binary heap ordering property

HEAPIFY

Start at last parent node

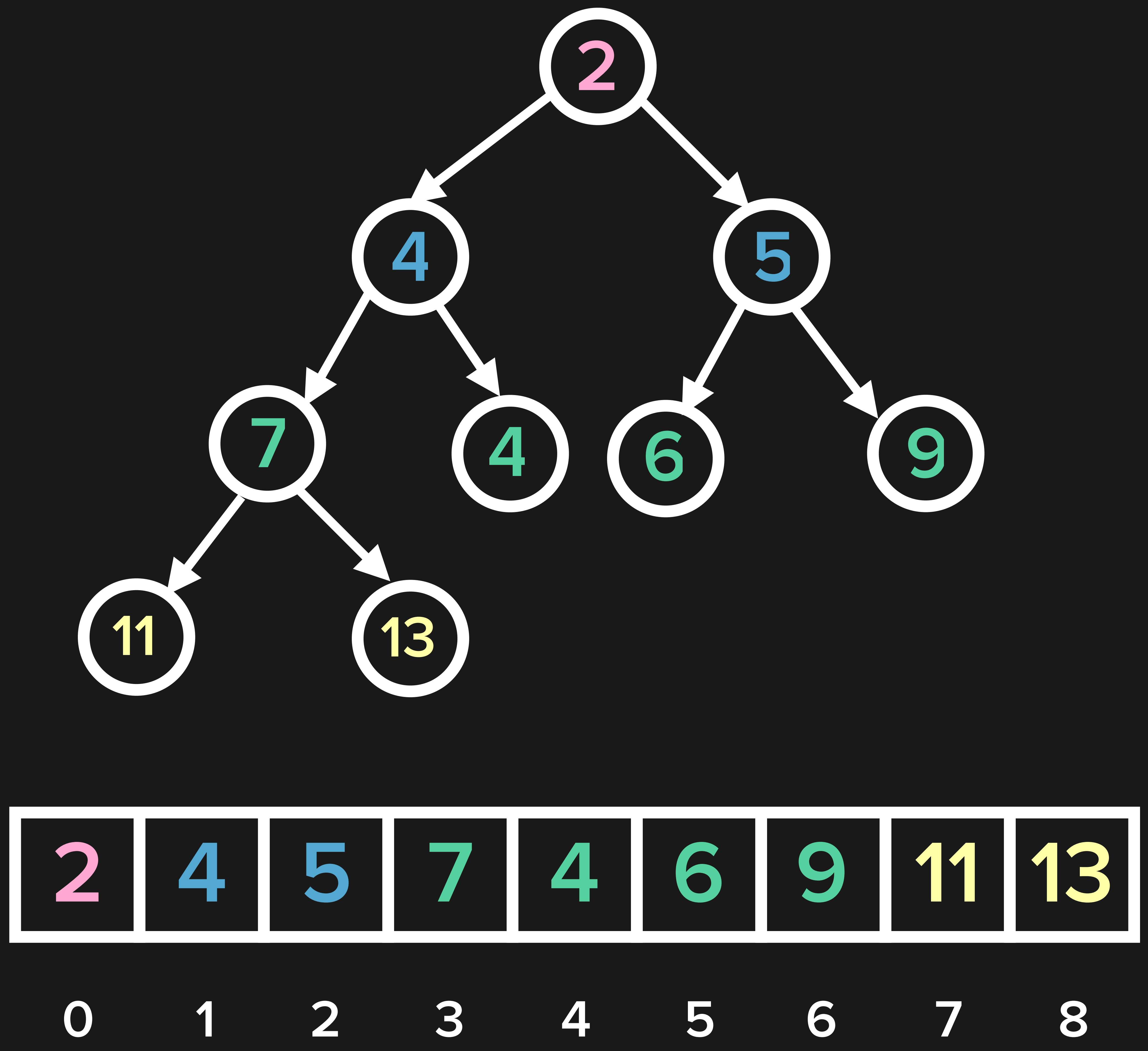
`index = (size - 2) / 2`

`while index >= 0:`

Sift down element at `index`

`index -= 1`

Start at `index = (size - 2) / 2` because that's the last parent node



MAKE SCHOOL

HEAP SORT

`heapify` array

`while (count > 0):`

  `find-min` or `find-max` element (**`peek`**)

  `delete-min` or `delete-max` element

HEAP RUNTIME

	Average Case	Worst Case
Space	O(n)	O(n)
Insert	O(log n)	O(log n)
Delete	O(log n)	O(log n)

HEAP SORT RUNTIME

	Average Case	Worst Case
Space	O(n)	O(n)
Heapify	O(n log n)	O(n log n)
Heap sort	O(n log n)	O(n log n)

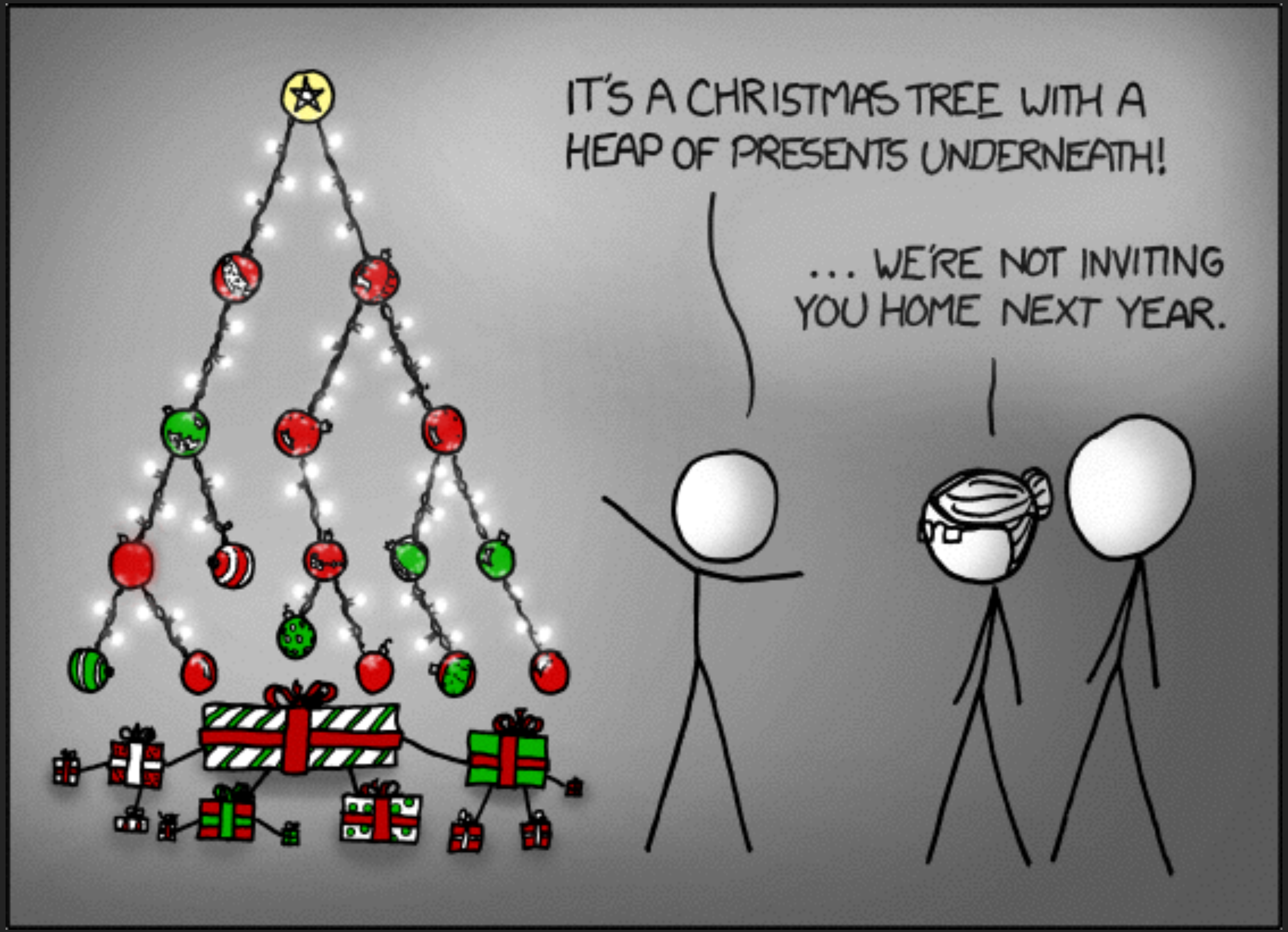
PRIORITY QUEUE SORTING

Name	Priority Queue Implementation	Best	Average	Worst
Selection sort	Unordered Array	n^2	n^2	n^2
Insertion sort	Ordered Array	n	n^2	n^2
Tree sort	Self-balancing binary search tree	$n\log(n)$	$n\log(n)$	$n\log(n)$
Heapsort	Heap	$n\log(n)$	$n\log(n)$	$n\log(n)$
Smoothsort	Leonardo Heap	n	$n\log(n)$	$n\log(n)$

Several sorting algorithms can be generalized by inserting items into and then removing them from a priority queue. Implementing the priority queue with different underlying data structures (including arrays, trees, and heaps) accounts for different sorting algorithm time complexities.

MAKE SCHOOL

RELEVANT XKCD COMIC



Title text: Not only is that terrible in general, but you just KNOW Billy's going to open the root present first, and then everyone will have to wait while the heap is rebuilt.