MAKE
SCHOOL

HEAPS

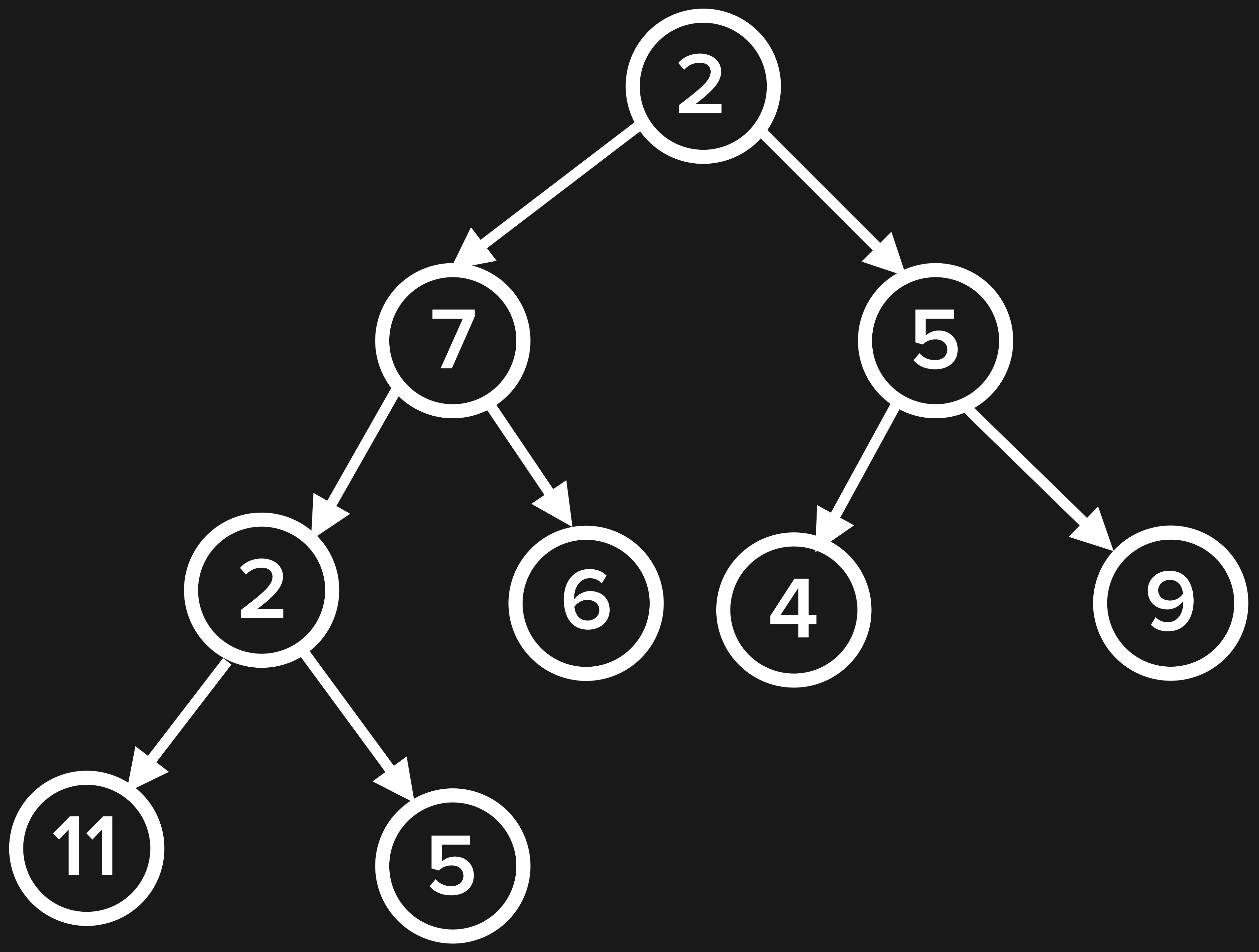
Heaps more fun than a barrel of monkeys

COMPLETE BINARY TREE

(REVIEW)

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

Height: $O(\log_2 n)$



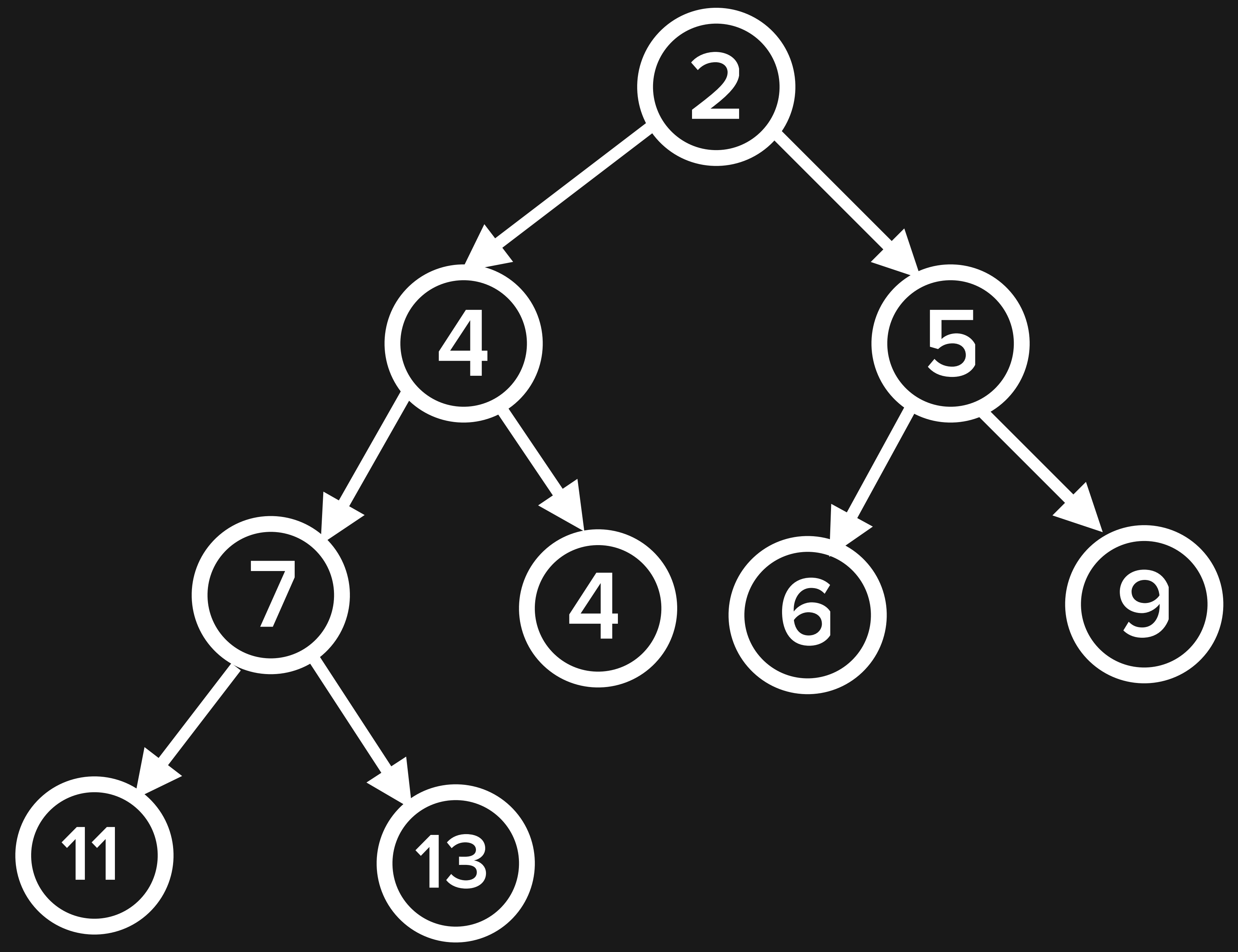
BINARY HEAP DEFINITION

A complete binary tree

Satisfies heap ordering property

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

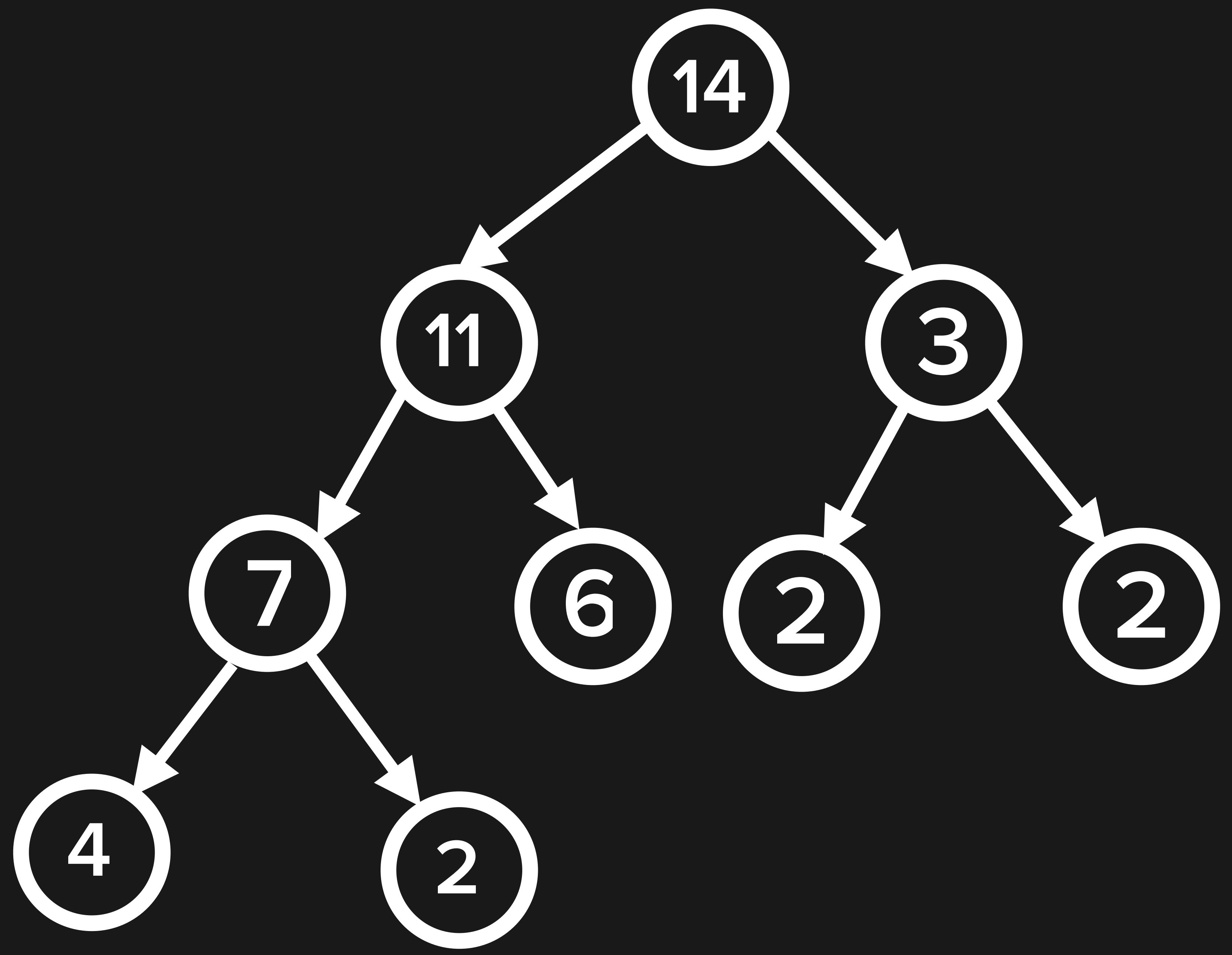
max-heap - each node is less than or equal to its parent (max value is root)



min-heap

CAREFUL

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



max-heap

PRIORITY QUEUES

Almost always implemented with a heap

Elements with smaller numbers are higher priority

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

Ordering happens with each insertion, so the cost of ordering is distributed across insertion instead of in one big chunk

PRIORITY QUEUE
APPLICATIONS

Prioritizing data packets in routers

Tracking unexplored routes in path-finding

Bayesian spam filtering

Data compression

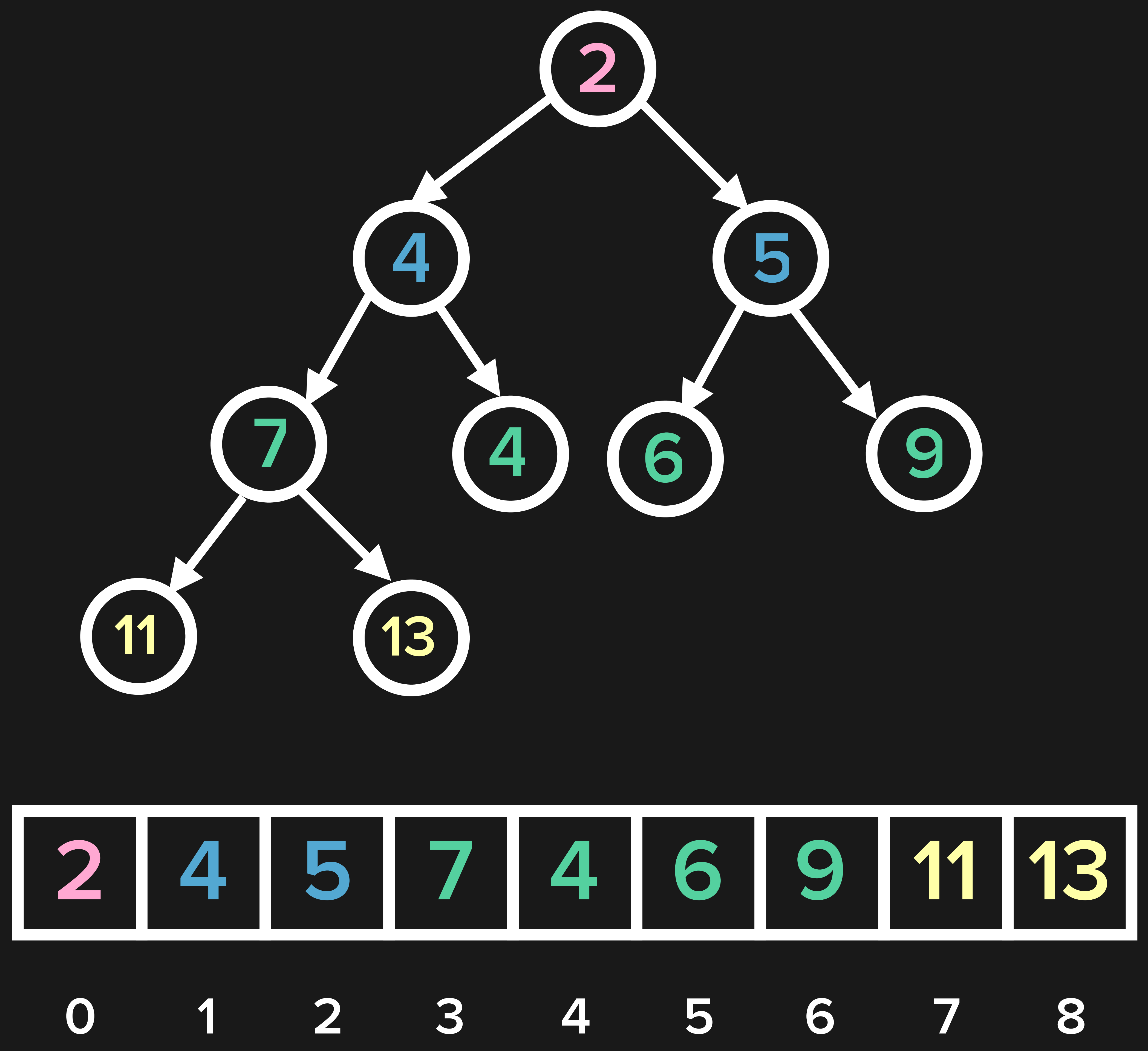
OS: load balancing, interrupt handling

ARRAY REPRESENTATION

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

Calculate parent-child index relationships with arithmetic

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

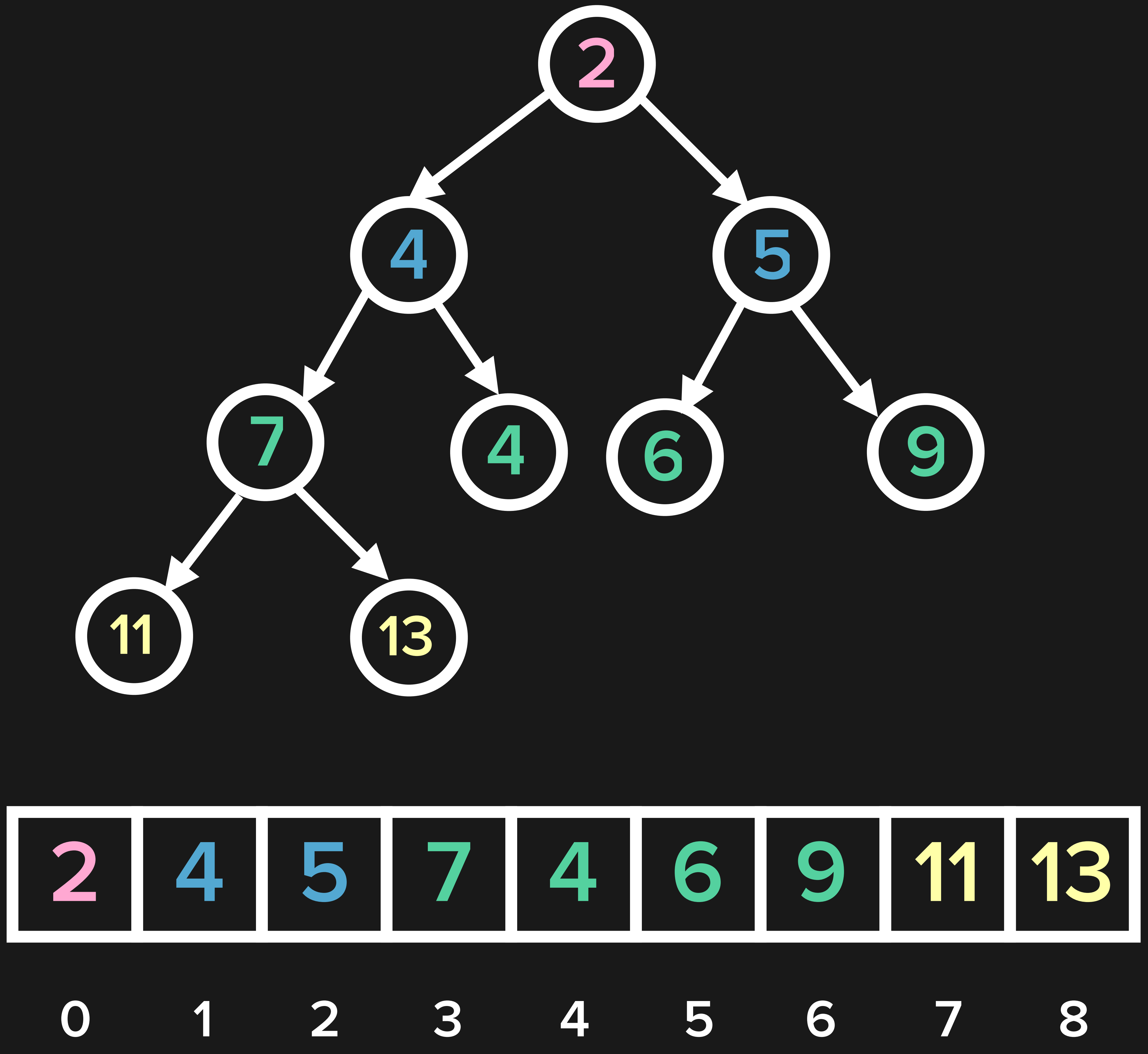


MAKE SCHOOL

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 (*heapsort*)

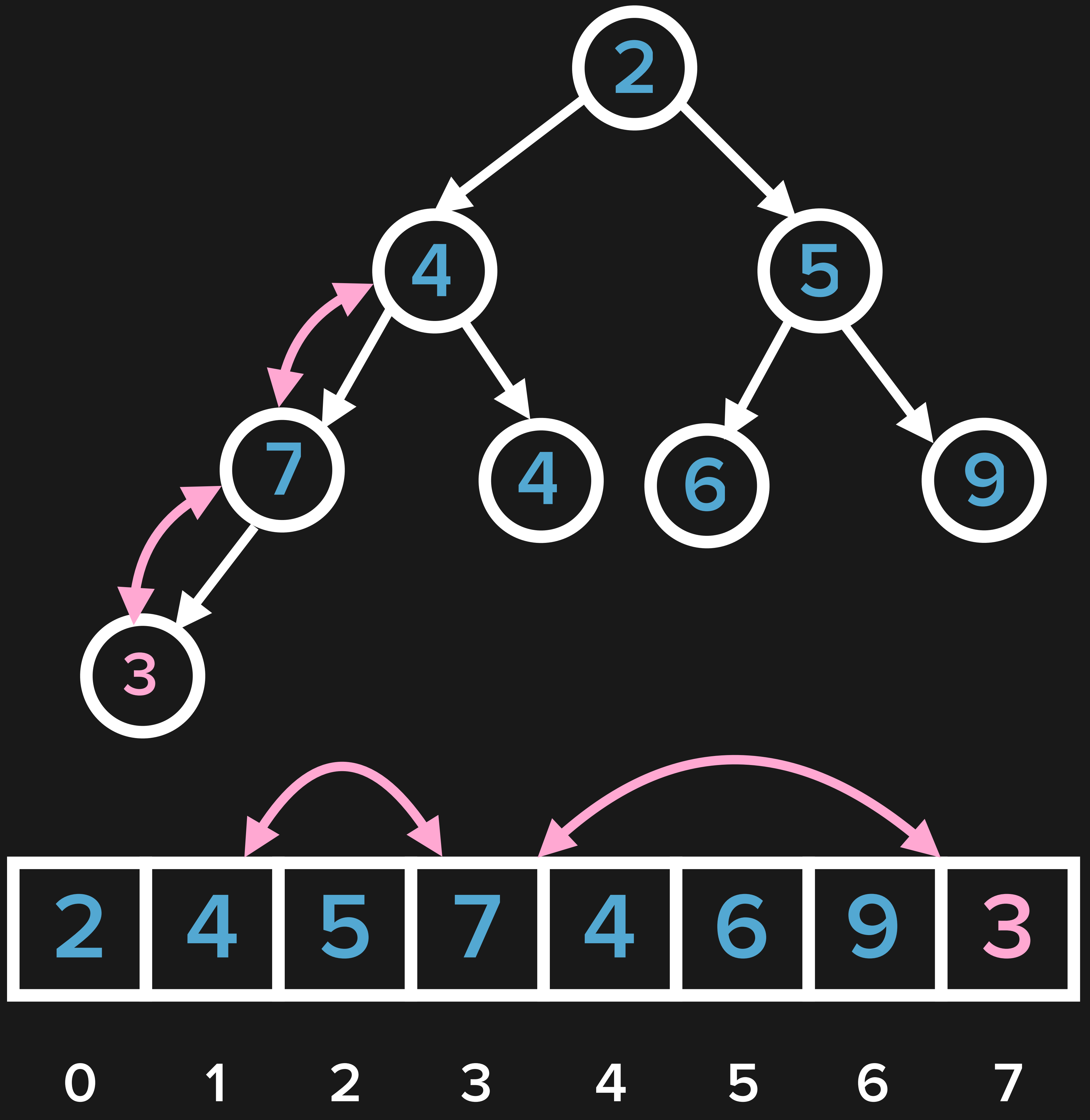


INSERT

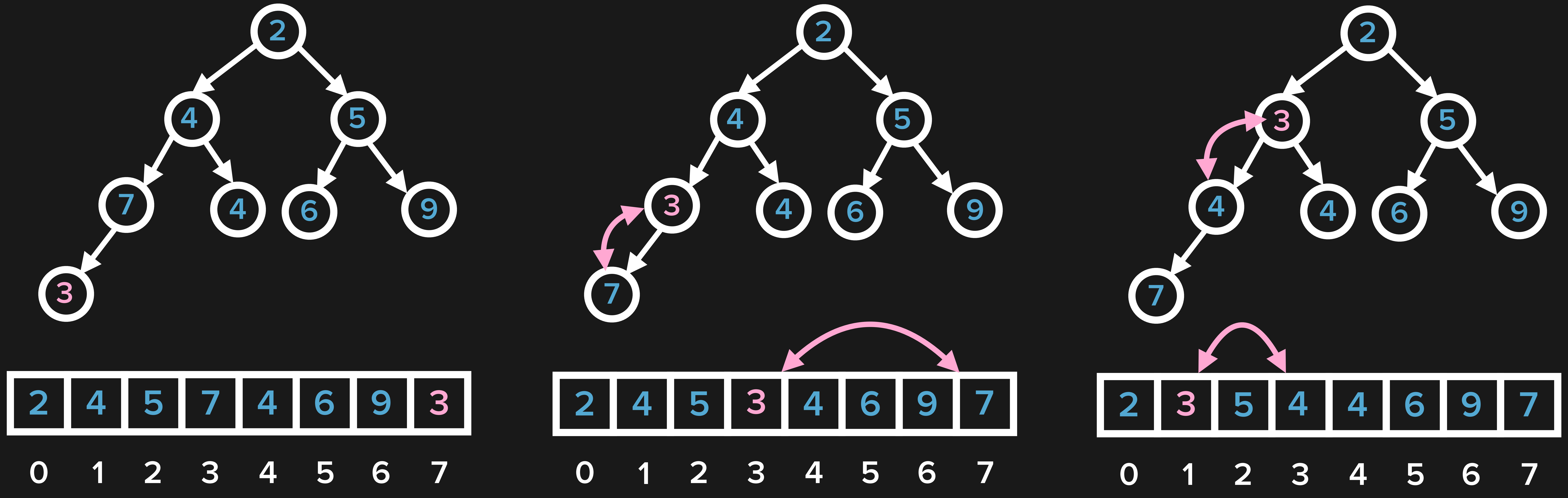
Add element to end

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

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



MAKE SCHOOL



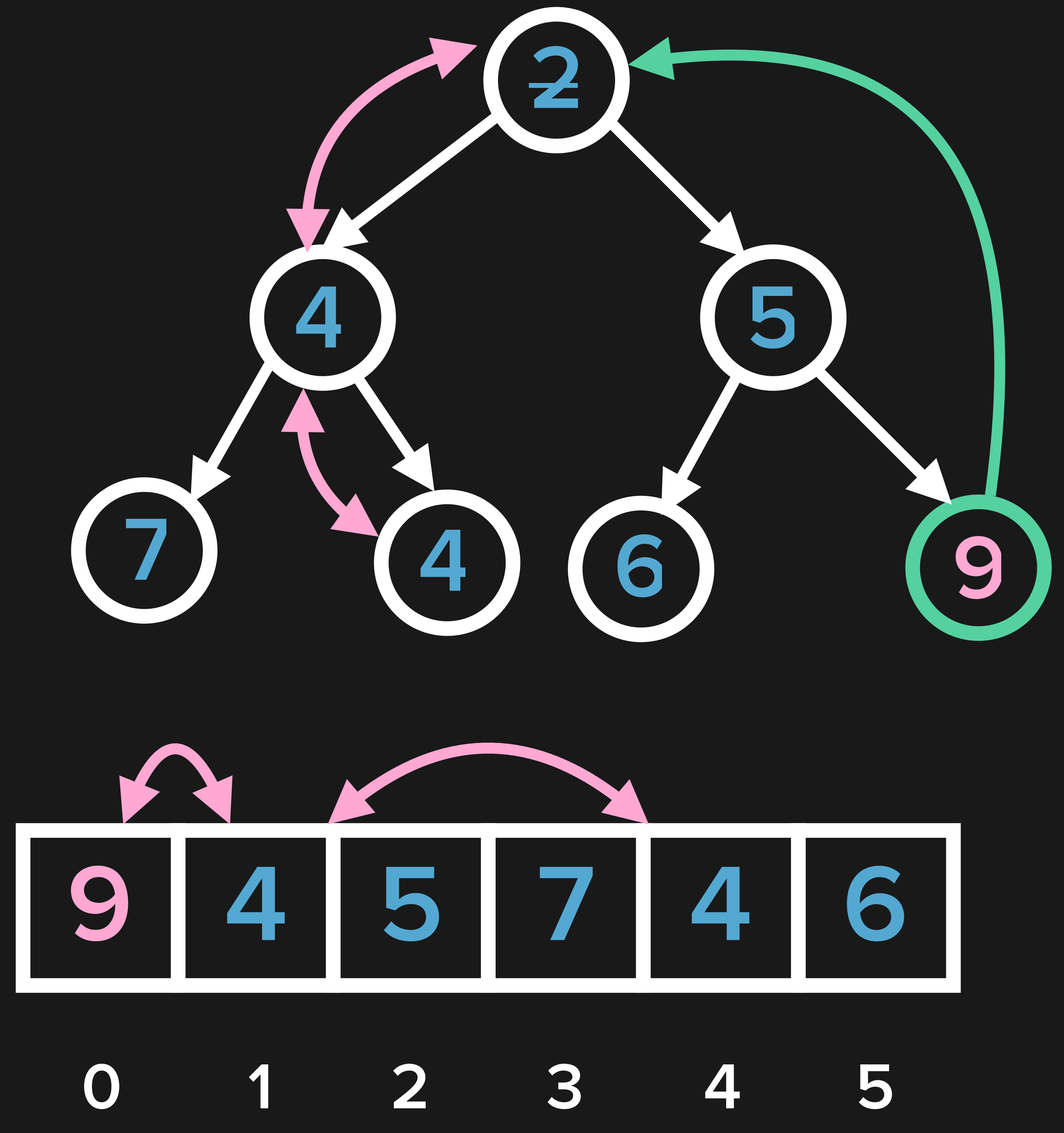
Insert

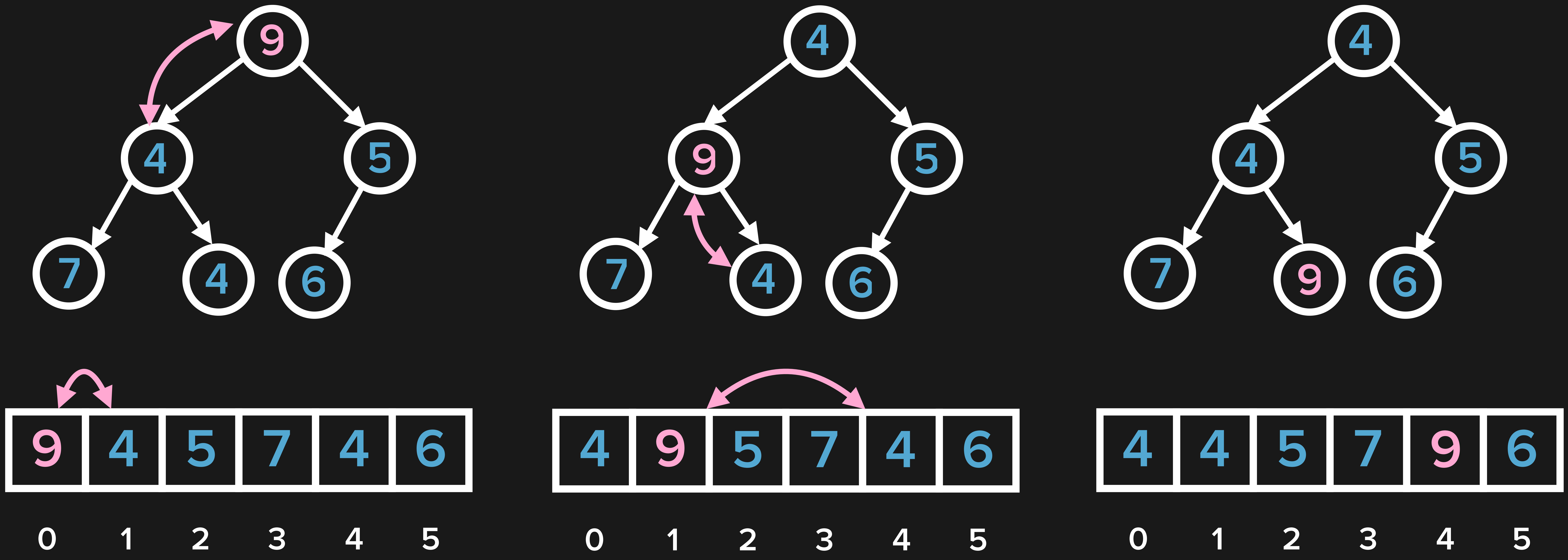
DELETE MIN / MAX

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 the ordering property





Delete min

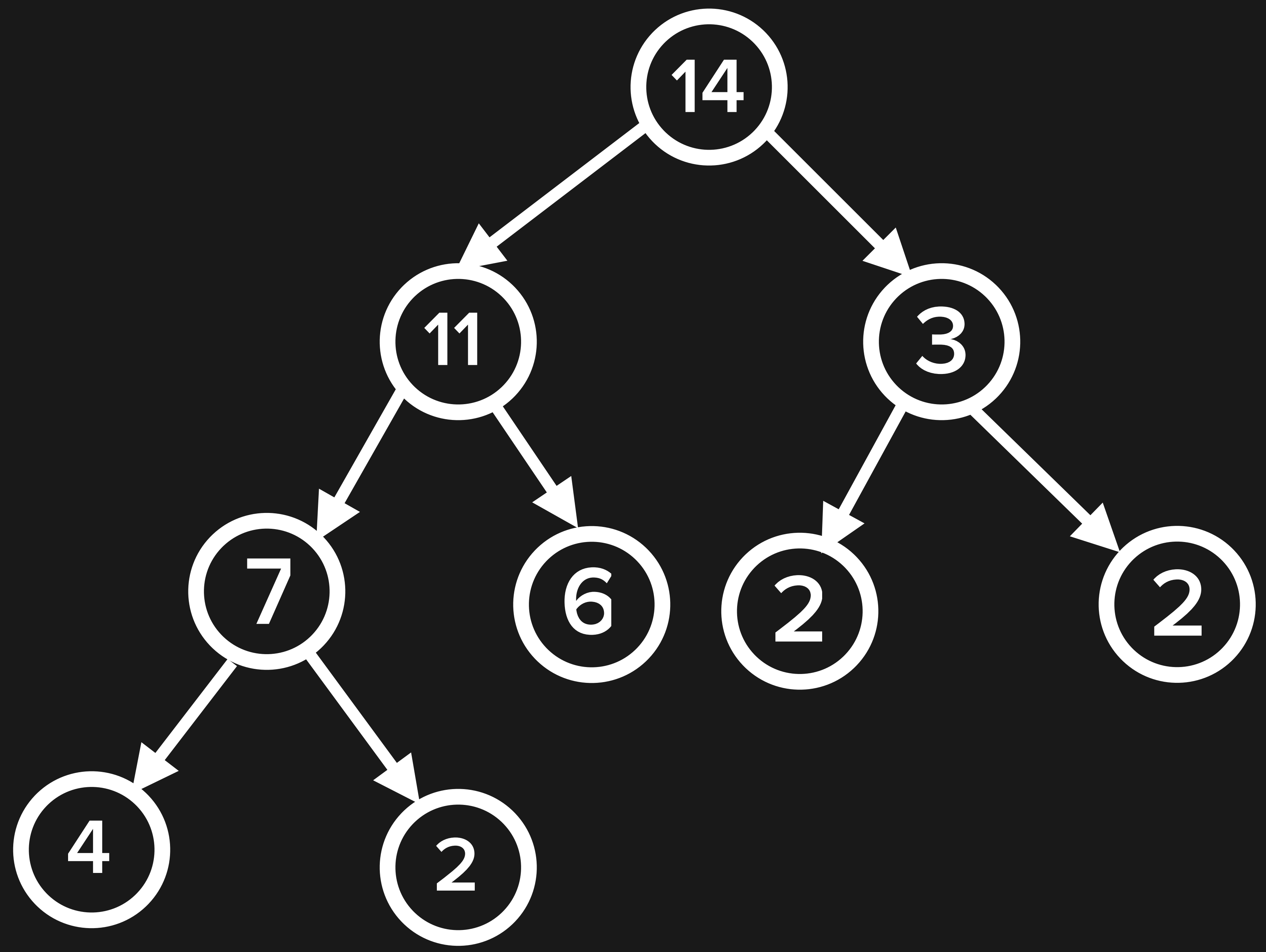
OTHER METHODS

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

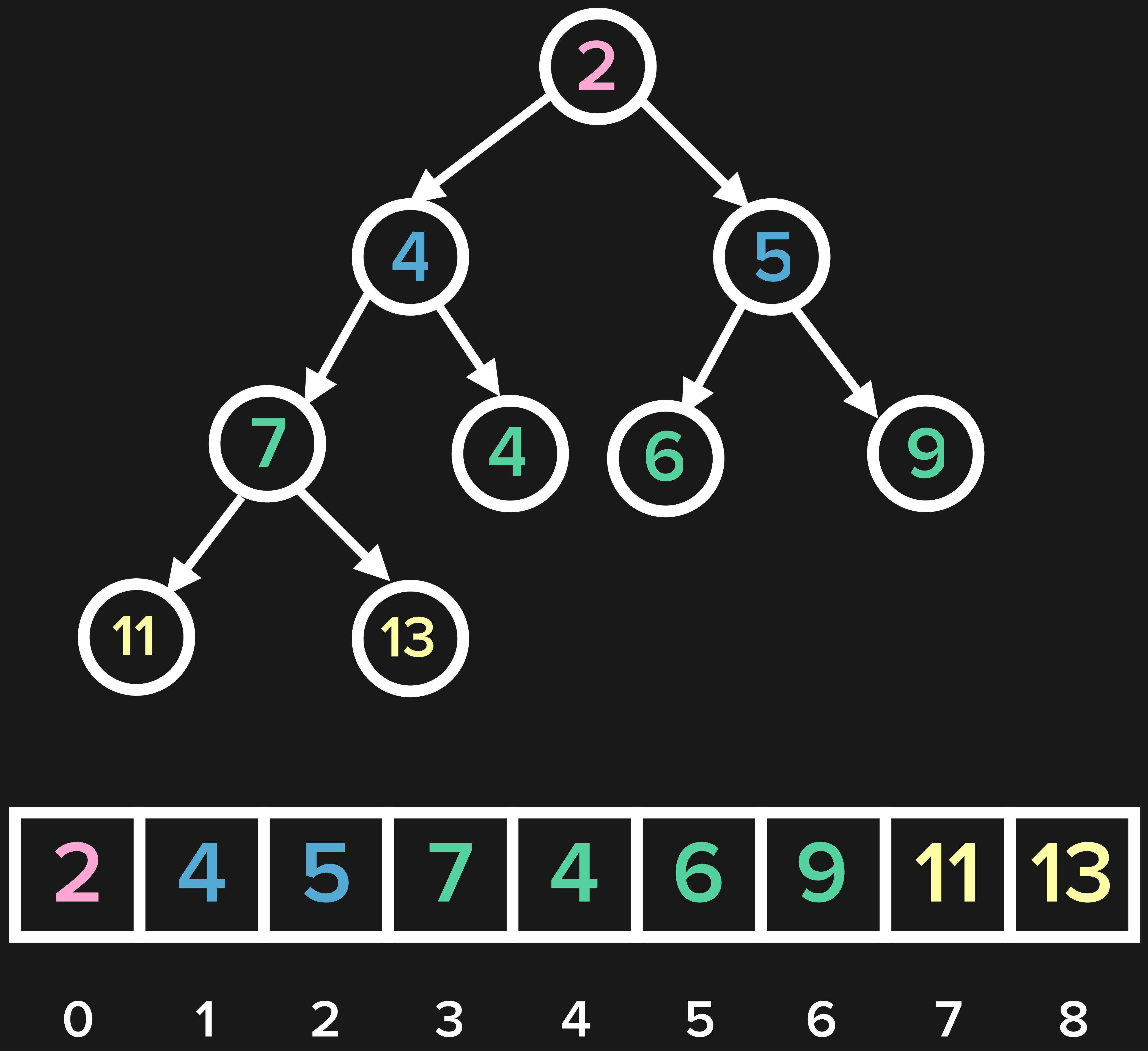
```
index = (count - 2) / 2

while index >= 0:
```

  Sift down element at `index`

```
  index -= 1
```

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



HEAPSORT

`heapify` array

```
while (count > 0):
```

Grab **min** or **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)

HEAPSORT RUNTIME

	Average Case	Worst Case
Space	O(n)	O(n)
Heapify	O(n log n)	O(n log n)
Heapsort	O(n log n)	O(n log n)