PRIORITY QUEUES & HEAPS

Heaps more fun than a barrel of monkeys

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

Abstract data type (ADT) that is like a queue, but each element has an associated priority number

Elements with smaller priority values are higher priority and are dequeued before lower priority elements *(VIPs skip to the front of the queue)*

How can you implement this ADT? *(many ways)*

PRIORITY QUEUES

Almost always implemented with a heap

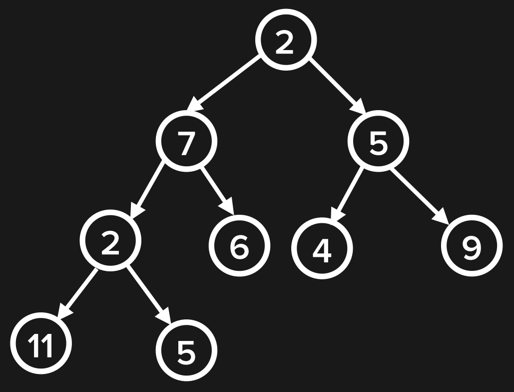
Elements are inserted in O(log n) time instead of O(n) time for a sorted array or linked list

Partial ordering happens with each insertion, so the cost of ordering is distributed across insertion and deletion instead of all at once

COMPLETE BINARY TREE

Every level is completely filled except lowest level, and nodes on lowest level are as far left as possible

Height is always $O(\log_2 n)$



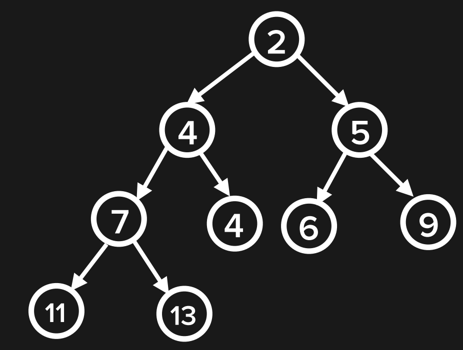
BINARY HEAP DEFINITION

A complete binary tree

Satisfies heap ordering property:

min-heap - each node is *less* than or equal to its children (*min* value is root)

max-heap - each node is *greater* than or equal to its children (*max* value is root)



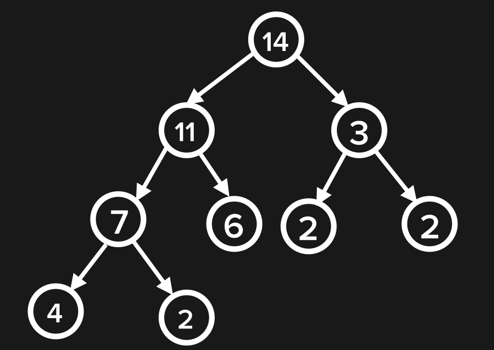
min-heap

- 2 flavors of heap ordering --- just a different convention
- in heaps we use both, and care about being able to do both
- min-heap:
    - min at the root
    - most accessible
    - how a priority queue works!
    - smallest # comes out first
-

CAREFUL

Heaps are *not* completely sorted (like a binary search tree)

Heaps are only “partially ordered”



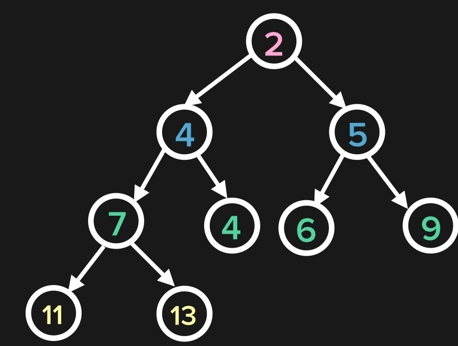
max-heap

ARRAY REPRESENTATION

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

Calculate parent-child index relationships with arithmetic

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



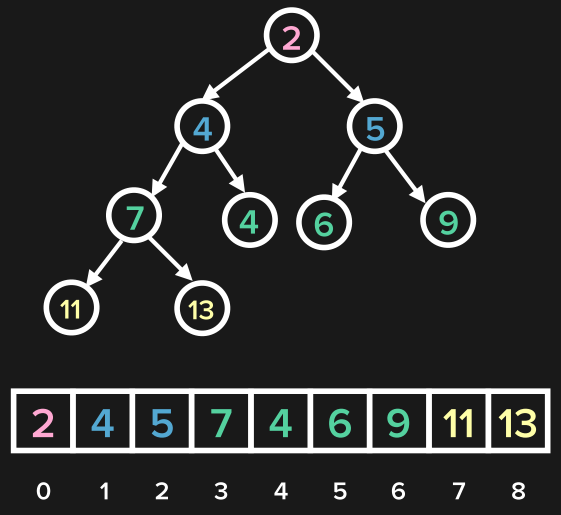
2	4	5	7	4	6	9	11	13
0	1	2	3	4	5	6	7	8

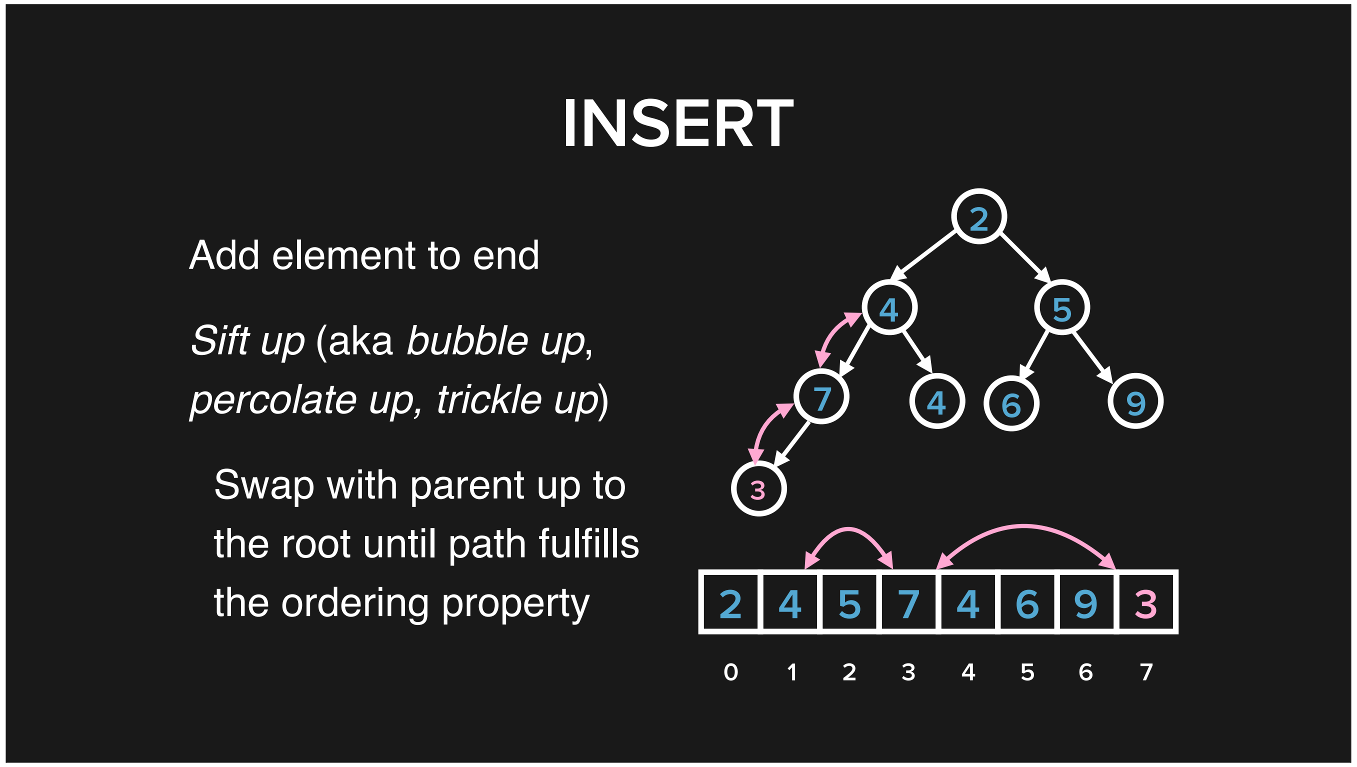
integer math always truncates a decimal value

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*)





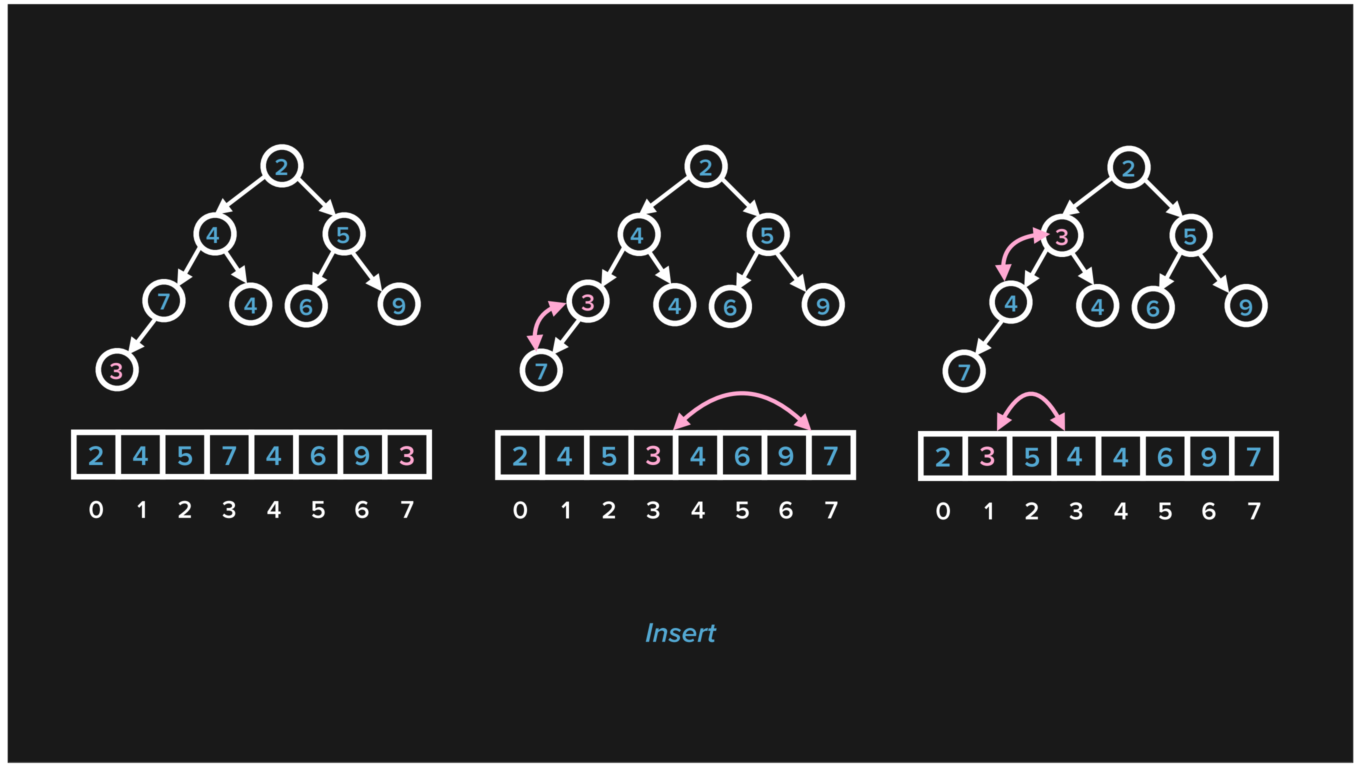
insert = enqueue in priority queue

just talking about # bc it's simple --- all that is important is that the objects are orderable / comparable

tree doesn't exist in memory, just a way to perceive / view it

insert on far left side of last level, next spot in reading order, must stay a complete tree

then we do a new kind of op called "sift up"



use the math trick to go to parent index

Left child index: $2i + 1$

Right child index: $2i + 2$

Parent index: $(i-1) / 2$

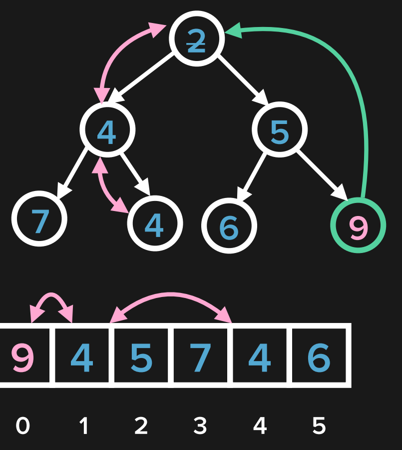
worst case: log n

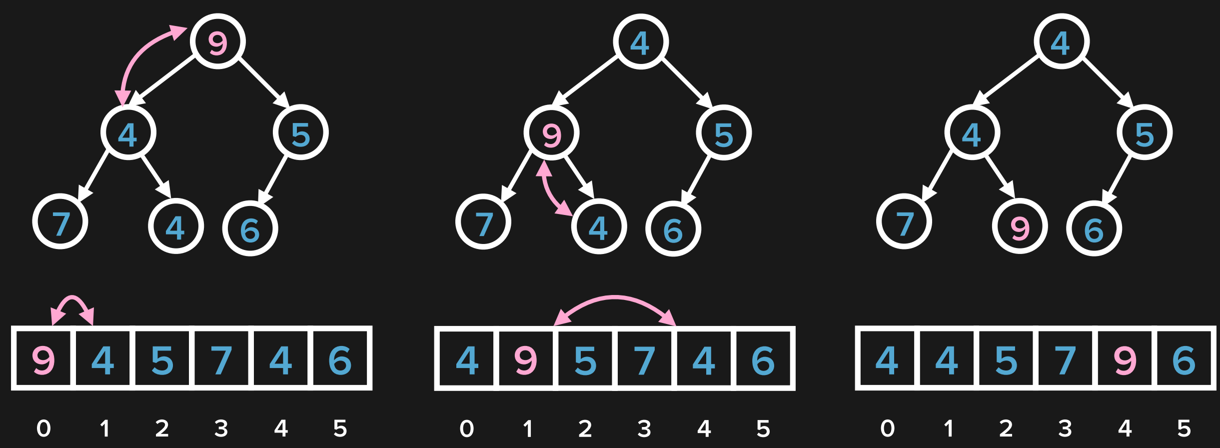
DELETE MIN / MAX

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



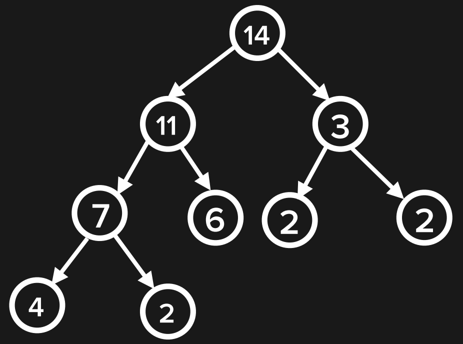


Delete min

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

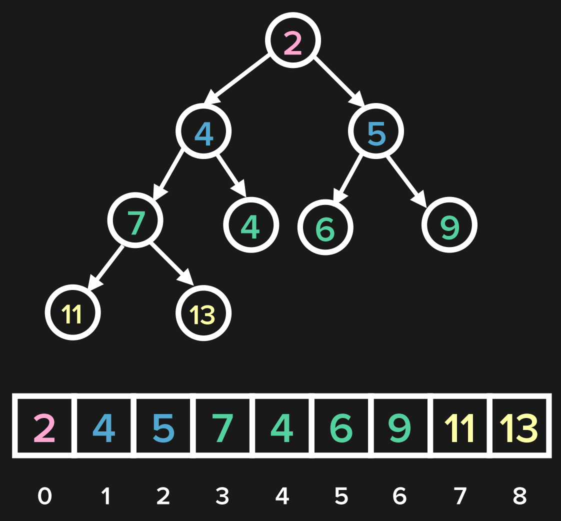
Start at last parent node

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

```
while index >= 0:
```

  Sift down element at `index`

```
  index -= 1
```



2
4
5
7
4
6
9
11
13
2 4 5 7 4 6 9 11 13
0 1 2 3 4 5 6 7 8

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)

RESOURCES

http://www.cs.cmu.edu/~adamchik/15-121/lectures/Binary%20Heaps/heaps.html

http://en.wikipedia.org/wiki/Binary_heap

http://en.wikipedia.org/wiki/Heap_(data_structure)

http://en.wikipedia.org/wiki/Priority_queue