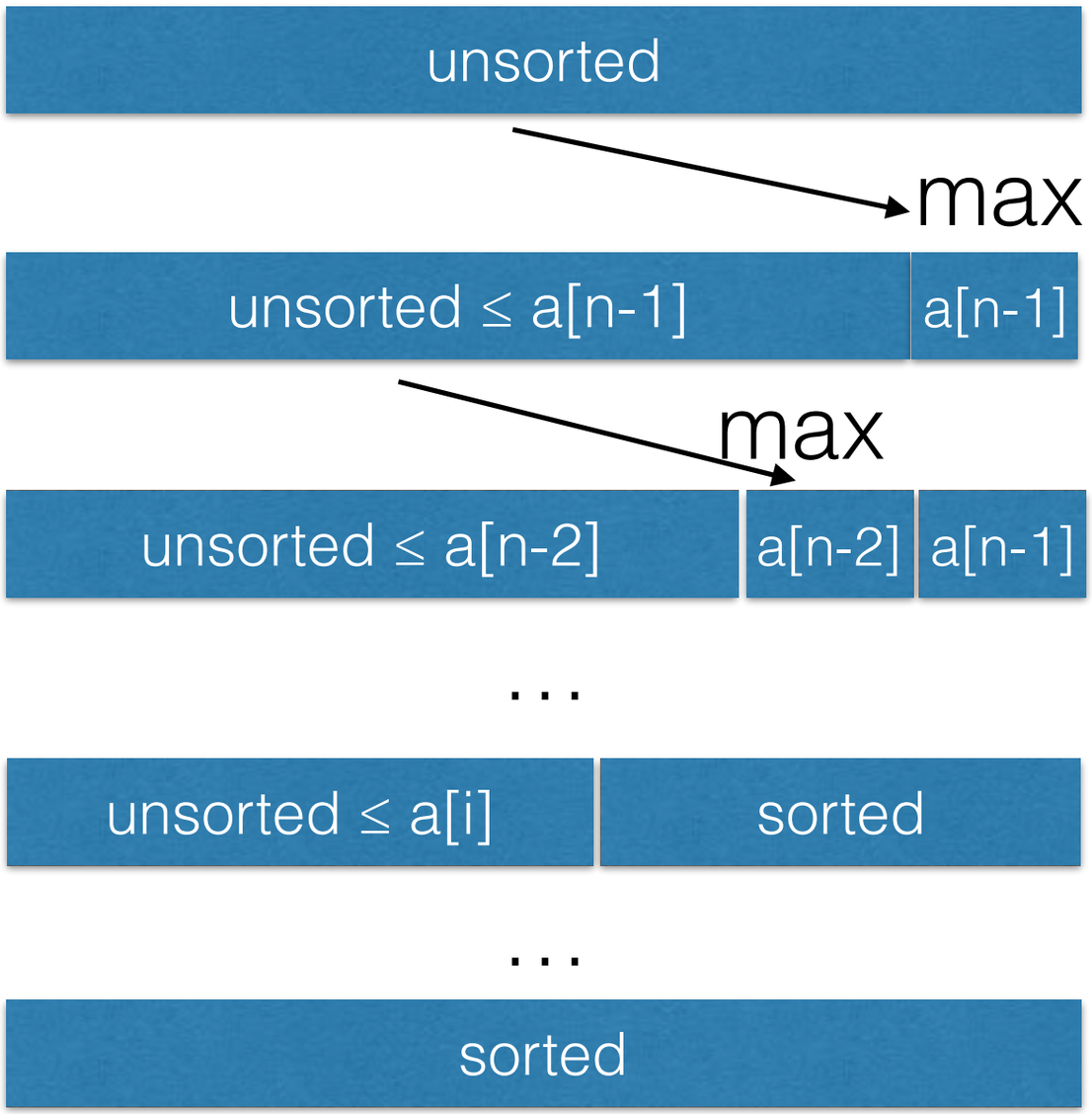
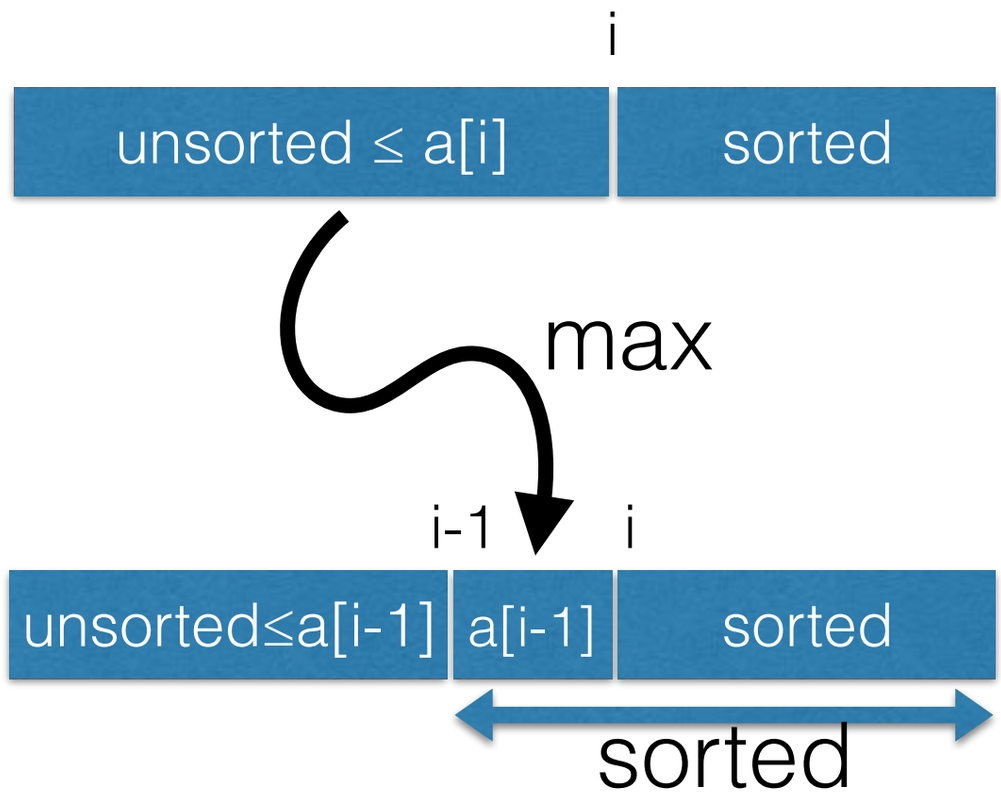
Data structures

CS 146 - Spring 2017

Recall selection sort



loop invariant



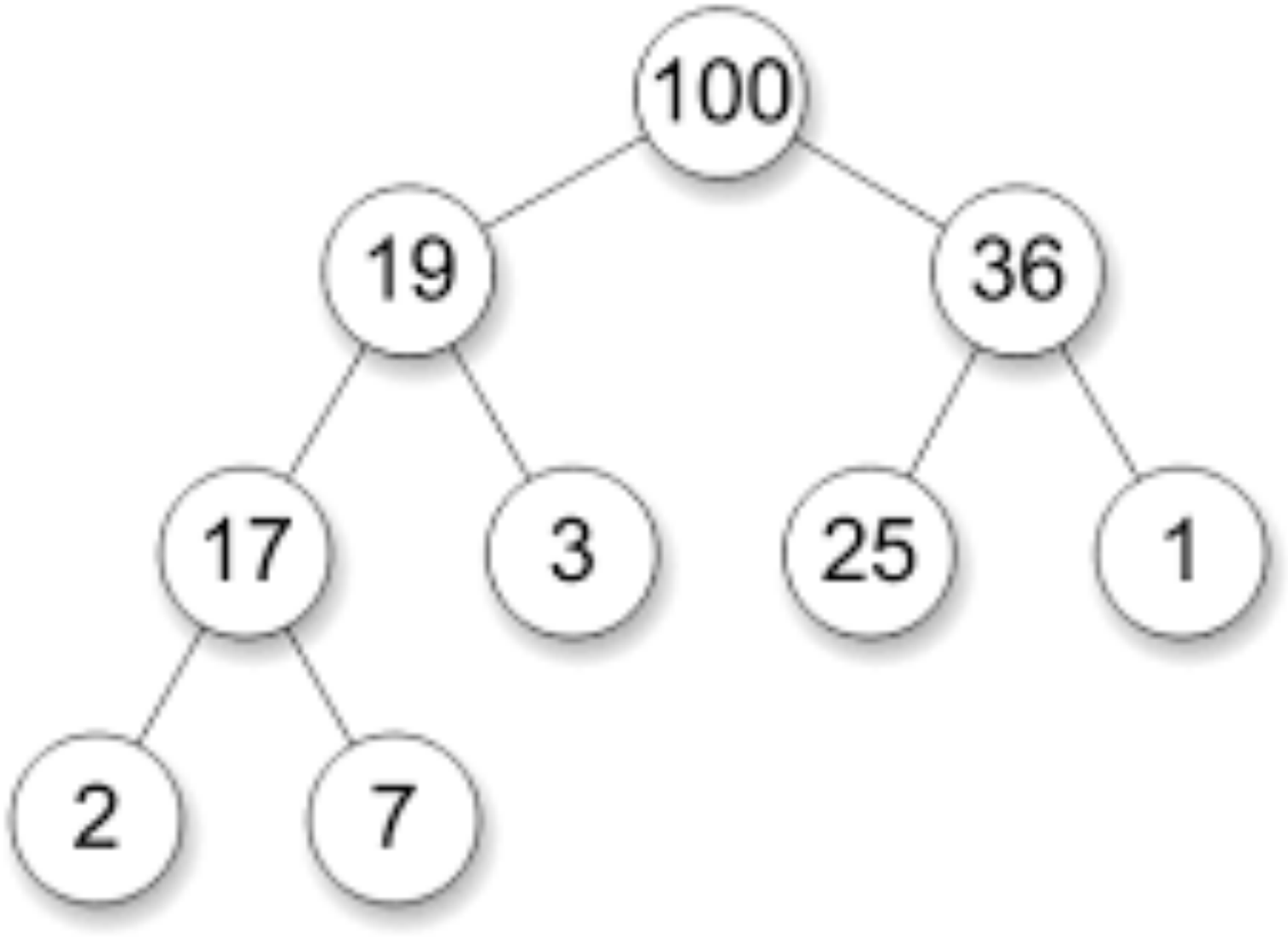
Today

- heapsort
- heap

Heapsort is selection-sort…

- using a heap as to find the min
- hopefully, the running time is n log n:
  - n iterations to select min and put it in right spot
  - **select min op in O(log n)**

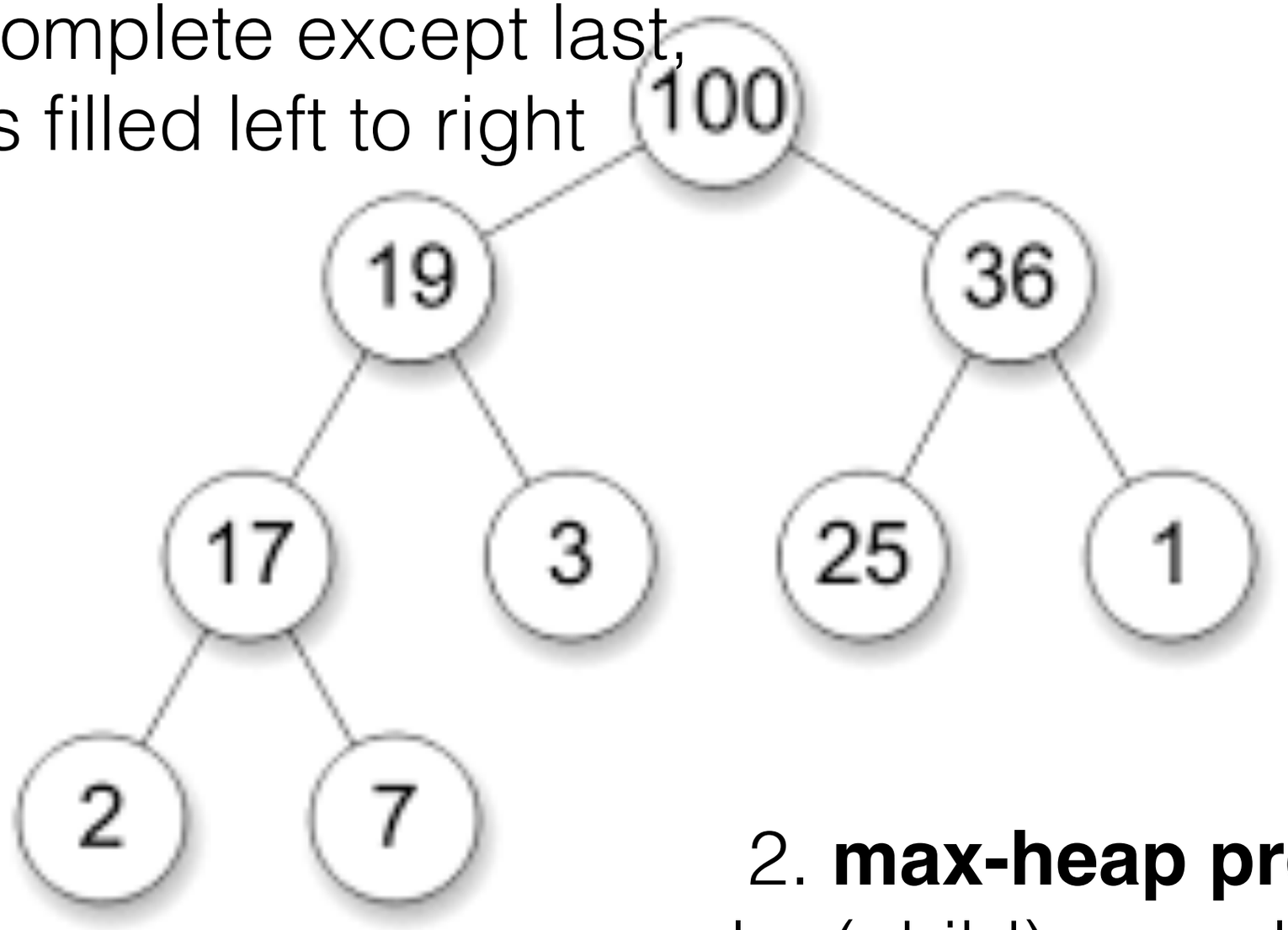
This is a max-heap



http://www.brianstorti.com/implementing-a-priority-queue-in-ruby/

max-heap definition

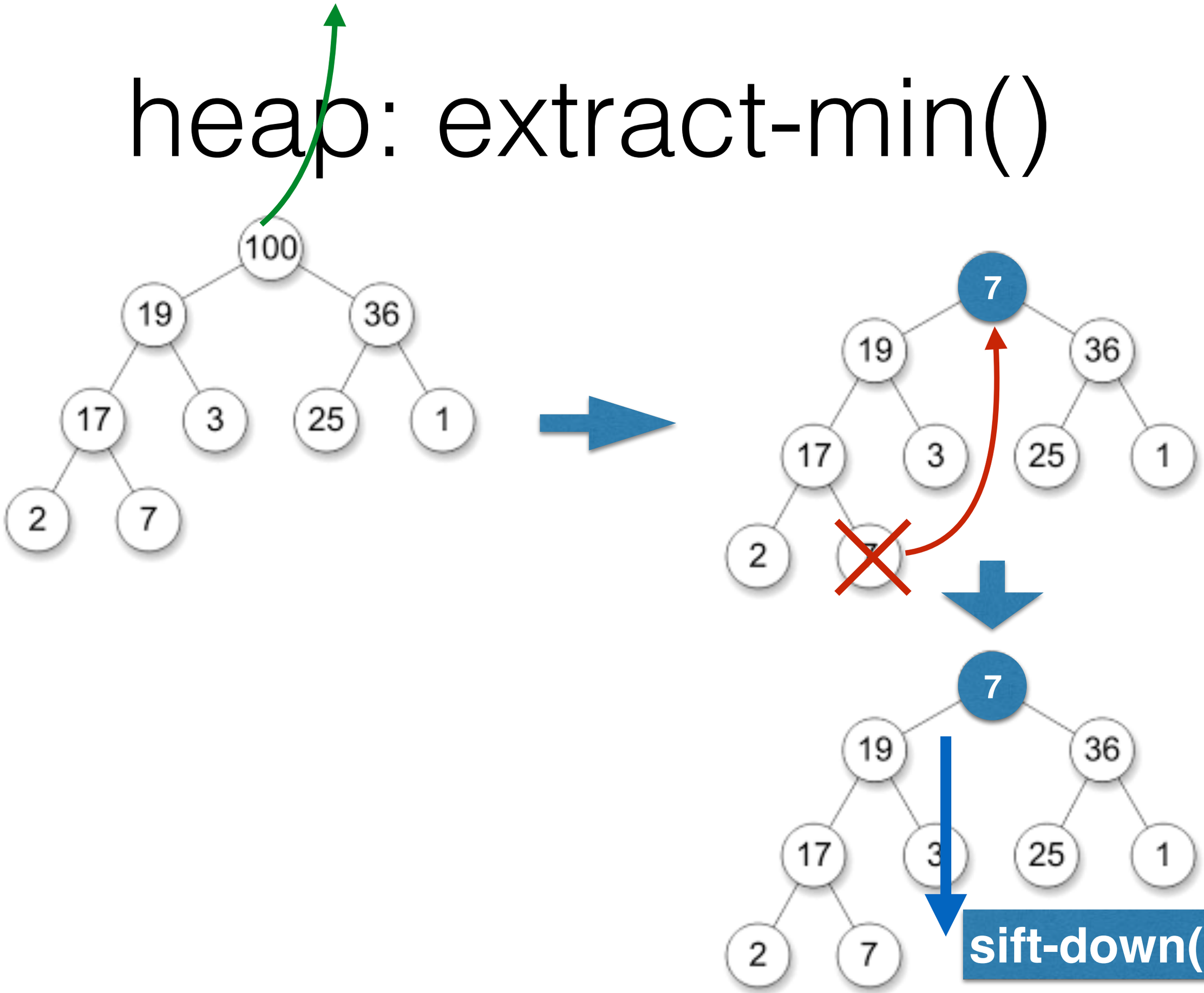
1. **complete binary tree**:
all levels complete except last,
which is filled left to right

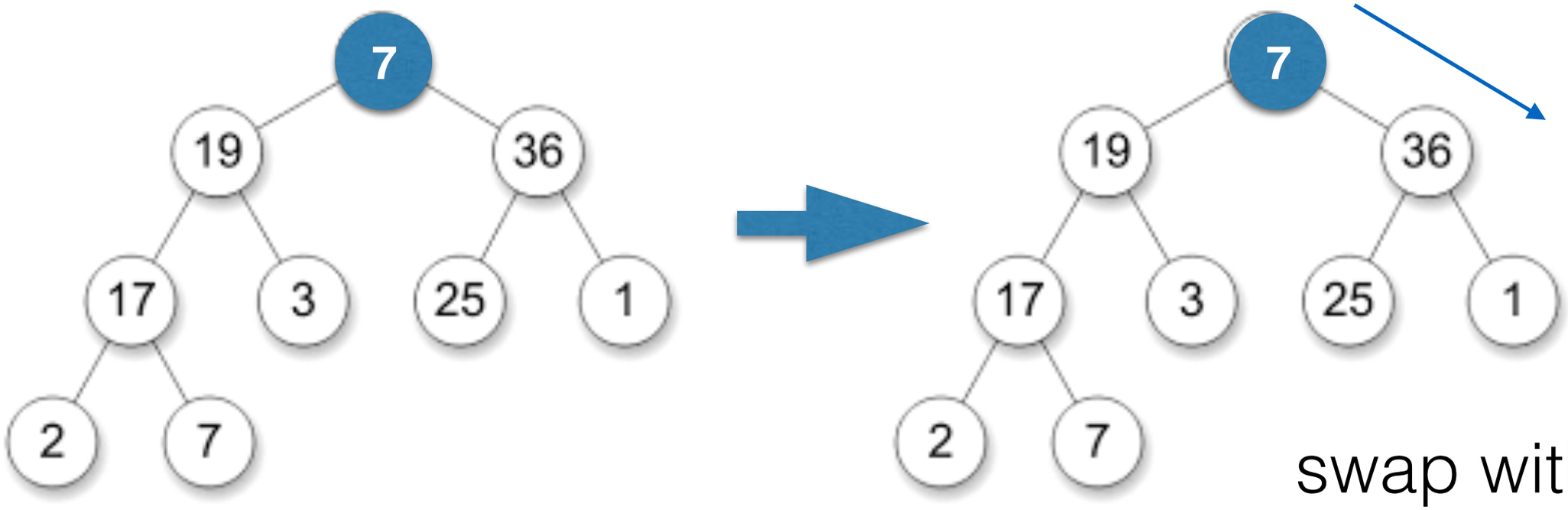


2. **max-heap property:**
value(child) <= value(parent)

http://www.brianstorti.com/implementing-a-priority-queue-in-ruby/

heap: extract-min()

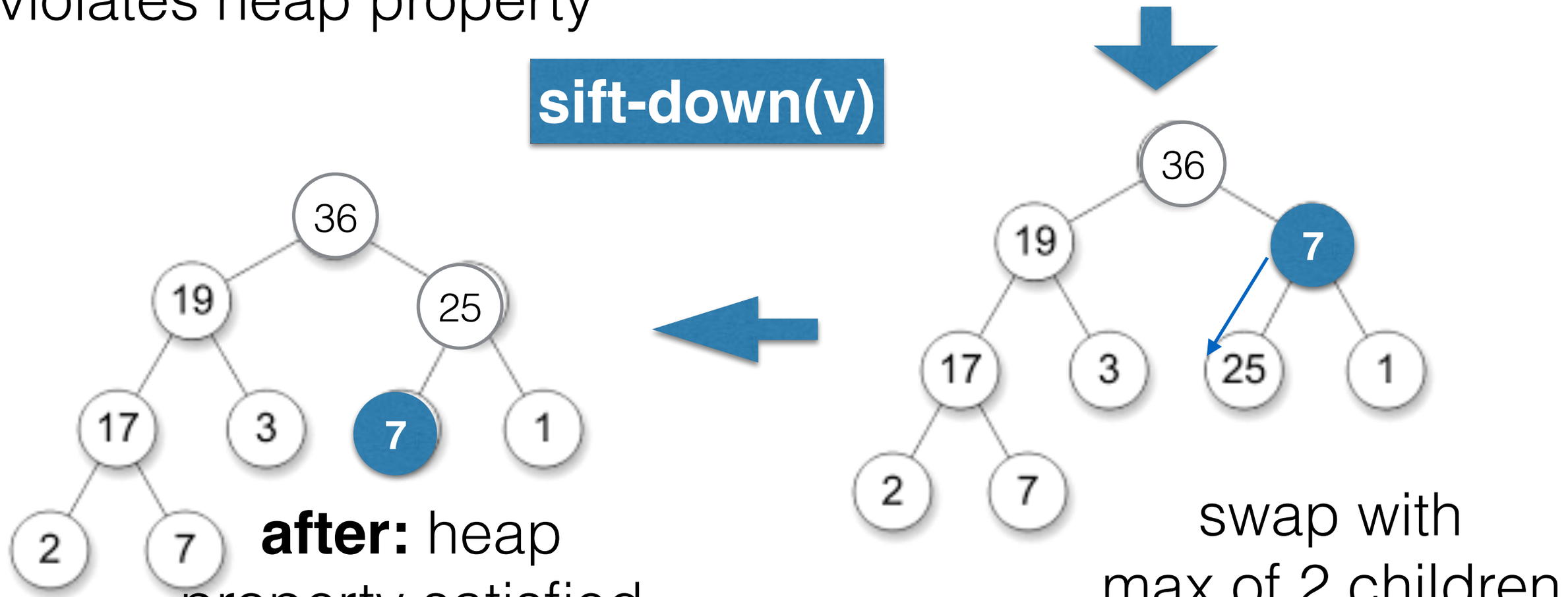




before: only one node v violates heap property

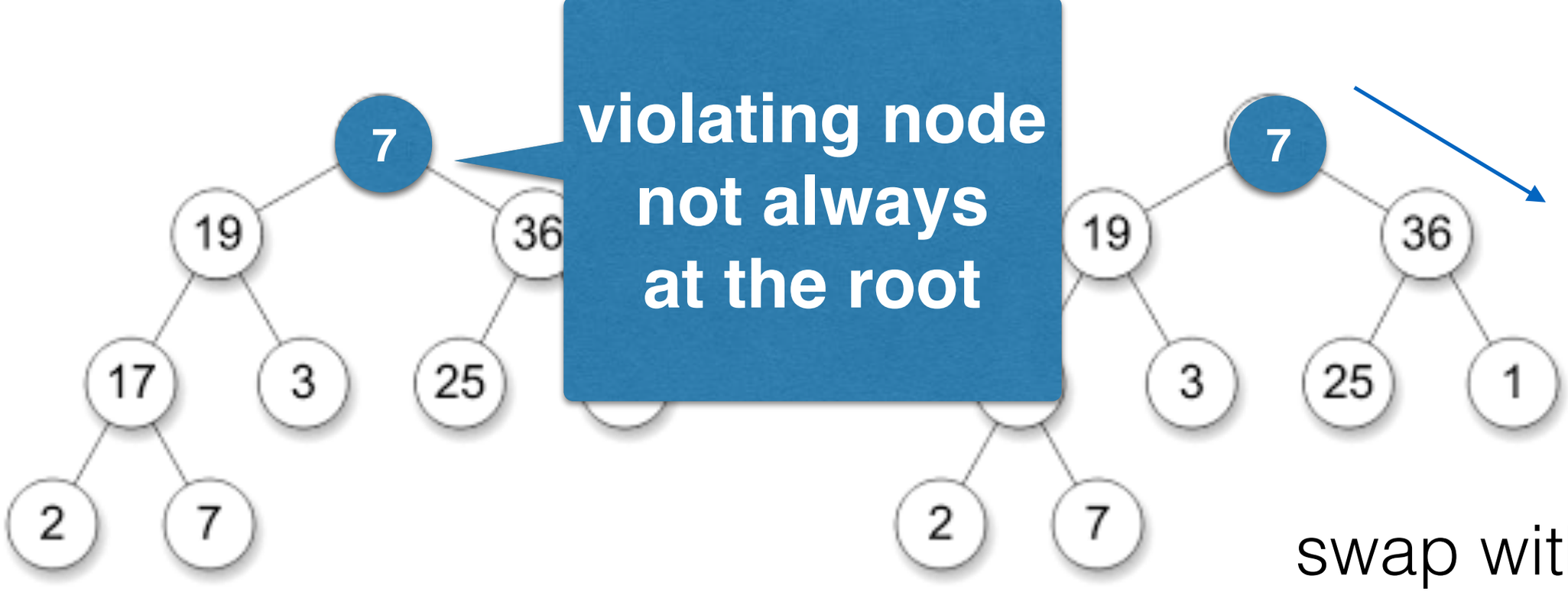
swap with max of 2 children

sift-down(v)



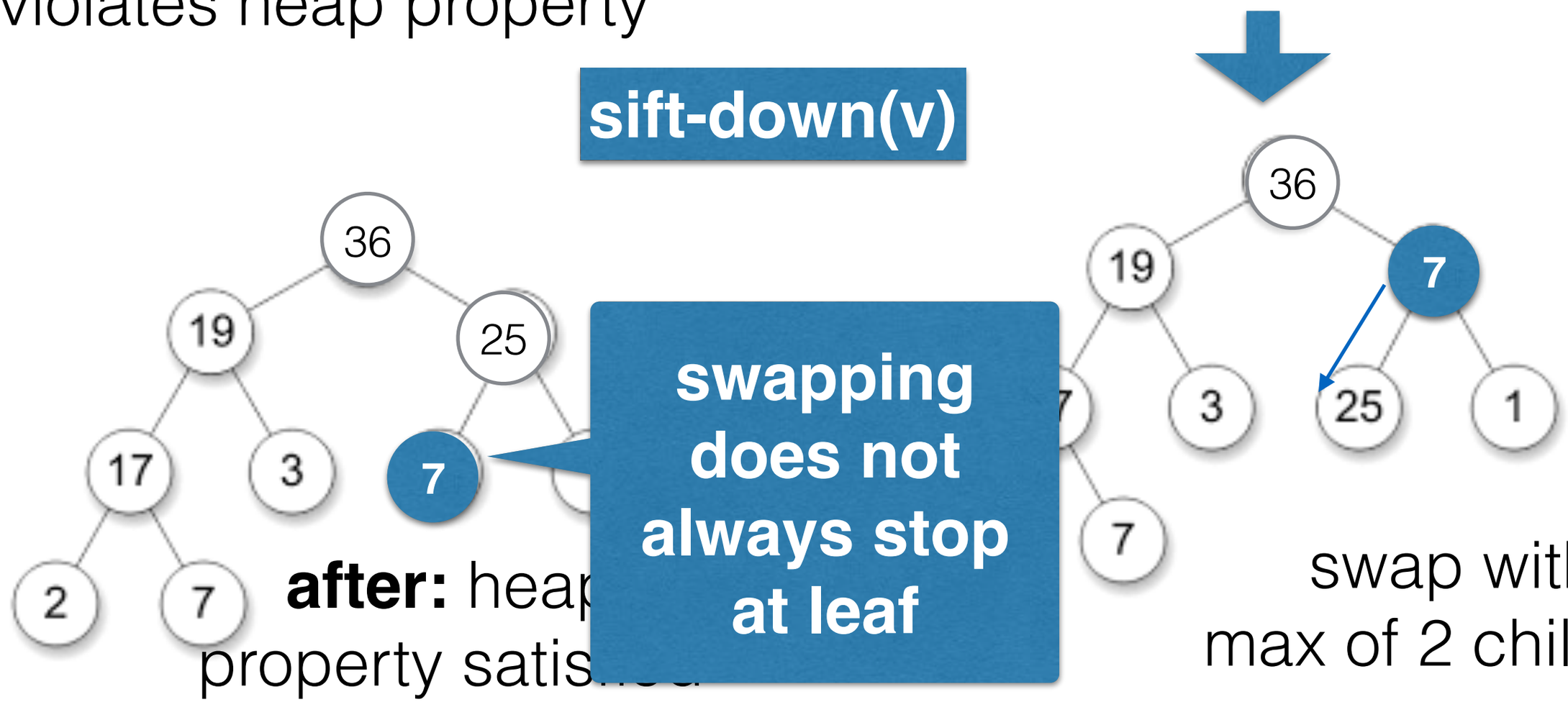
after: heap property satisfied

swap with max of 2 children



before: only one node v
violates heap property

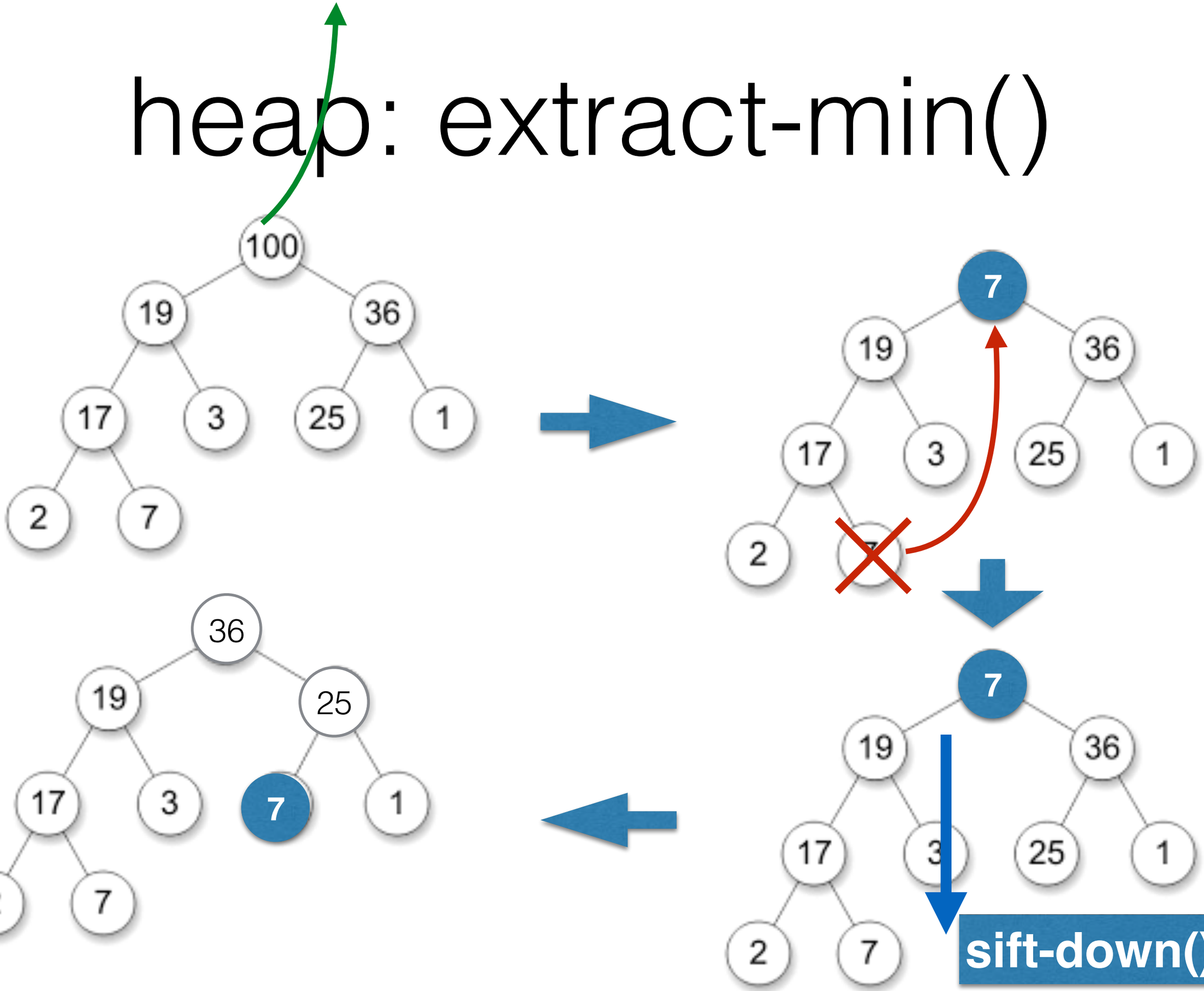
swap with
max of 2 children



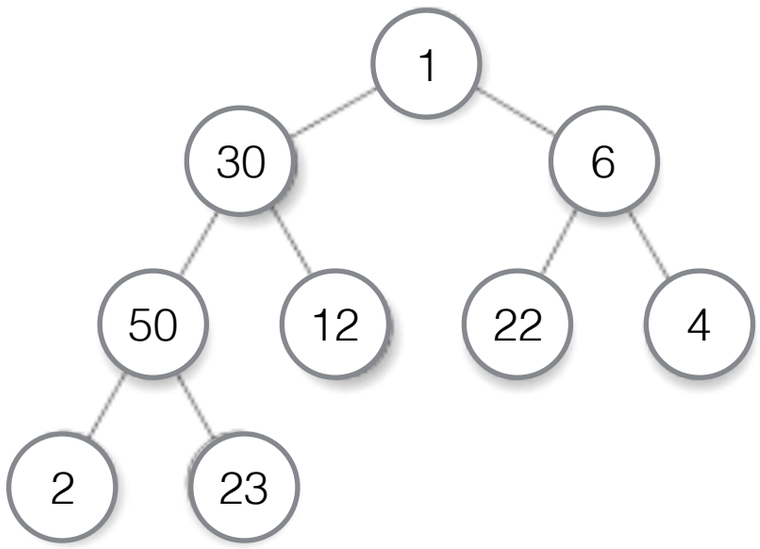
after: heap
property satisfied

swap with
max of 2 children

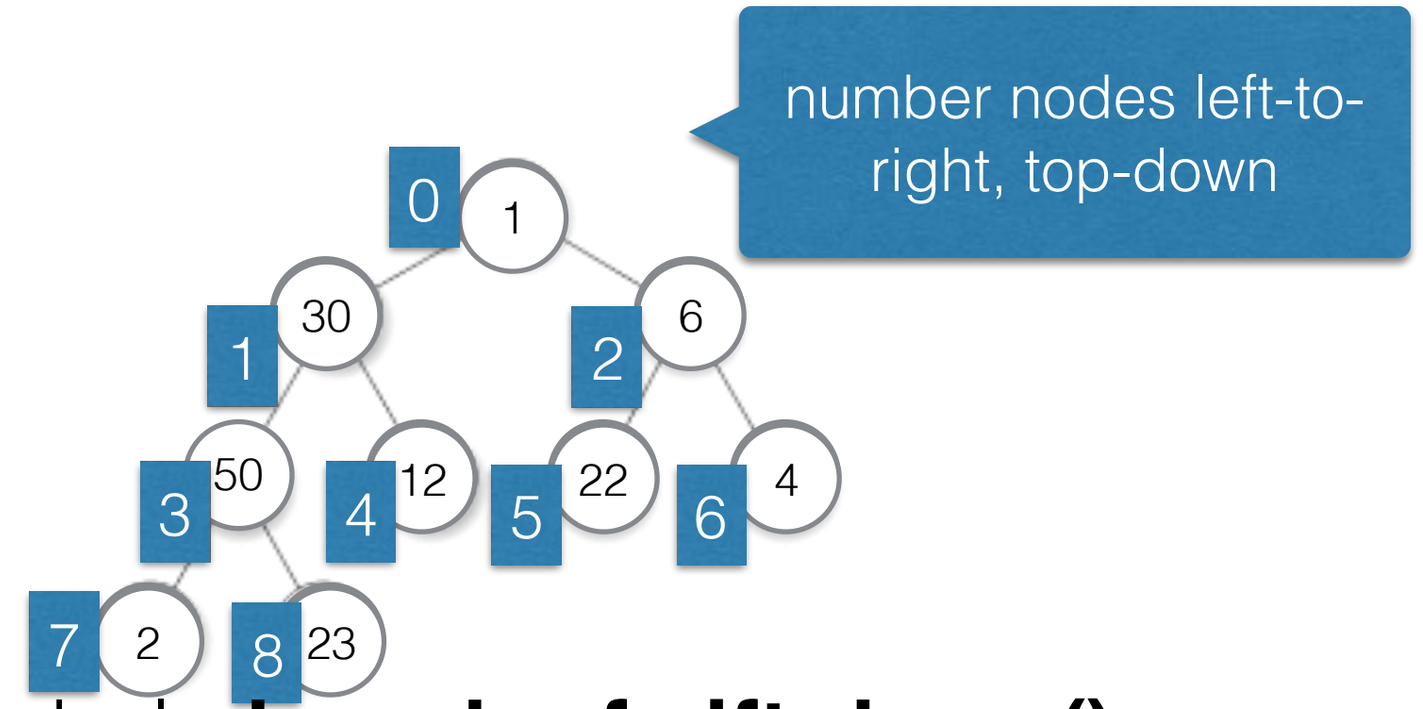
heap: extract-min()



heap: make-queue()



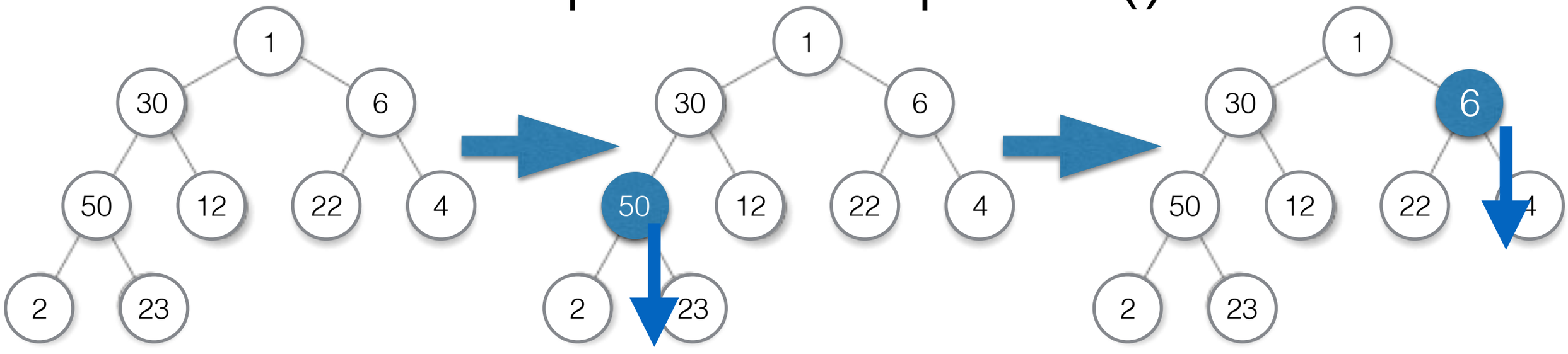
before: random values in complete binary tree



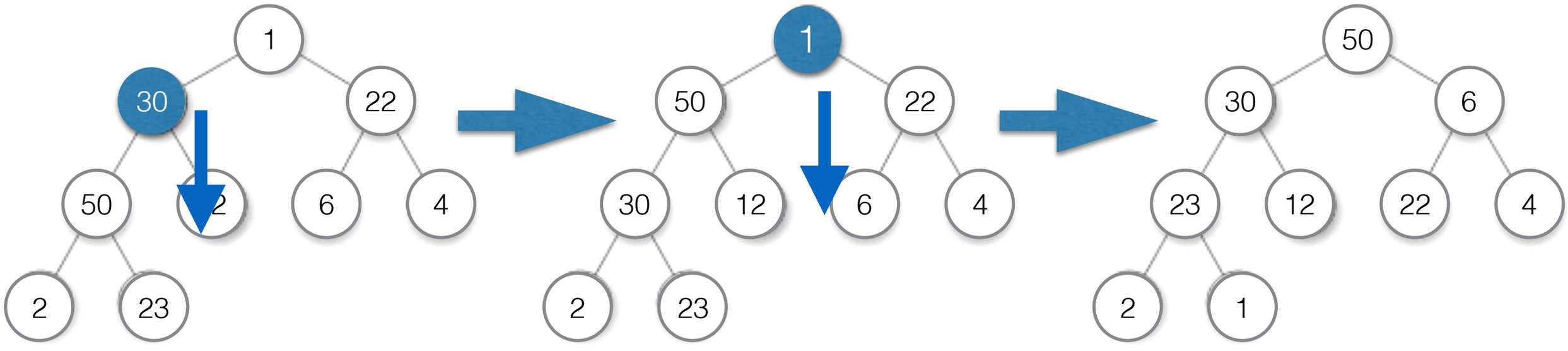
idea: do a whole **bunch of sift-down()** from the bottom level nodes to the top level, ie in reverse order of the numbering

after: hopefully no heap property violations

heap: make-queue()

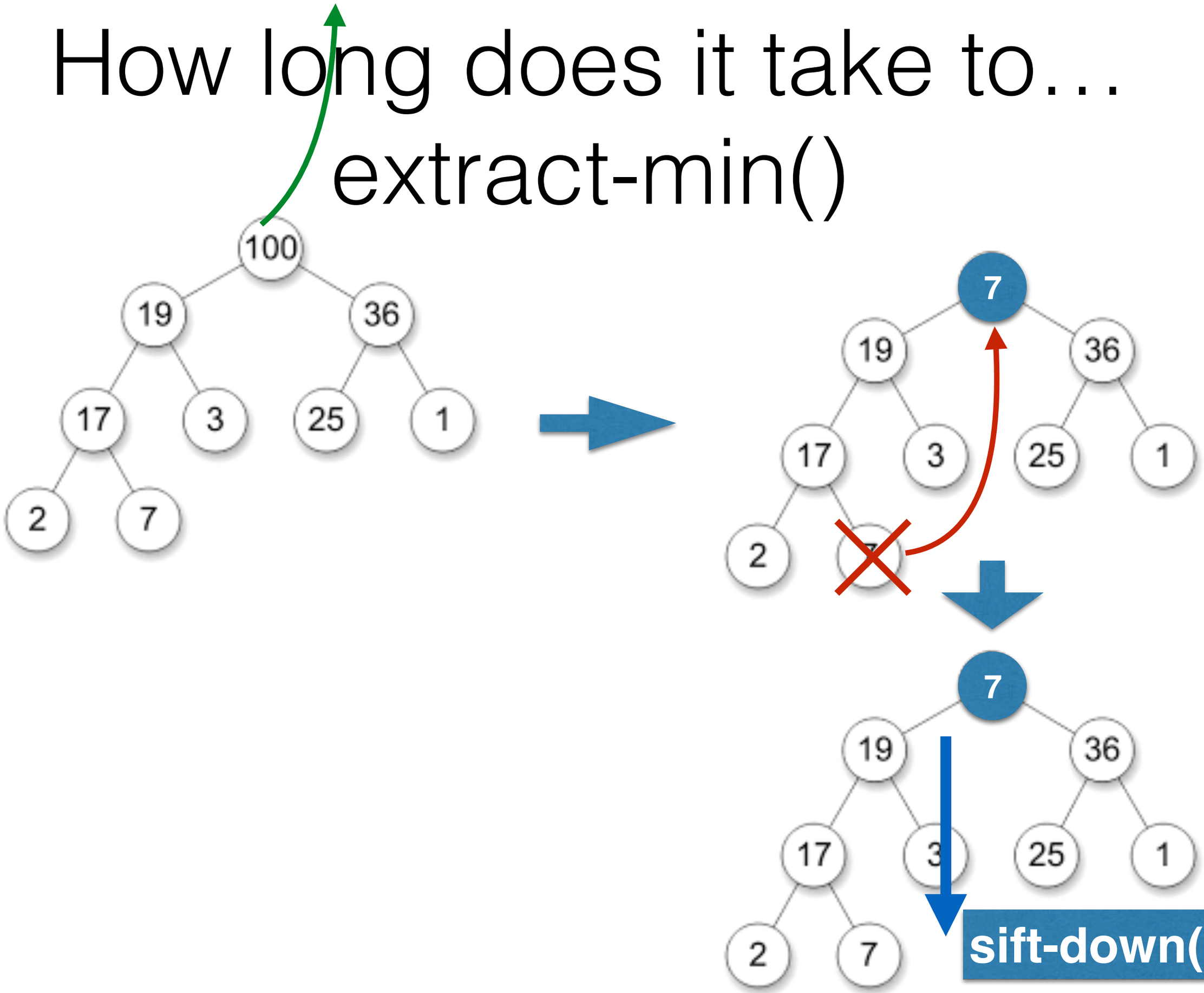


before: random values
in complete binary tree

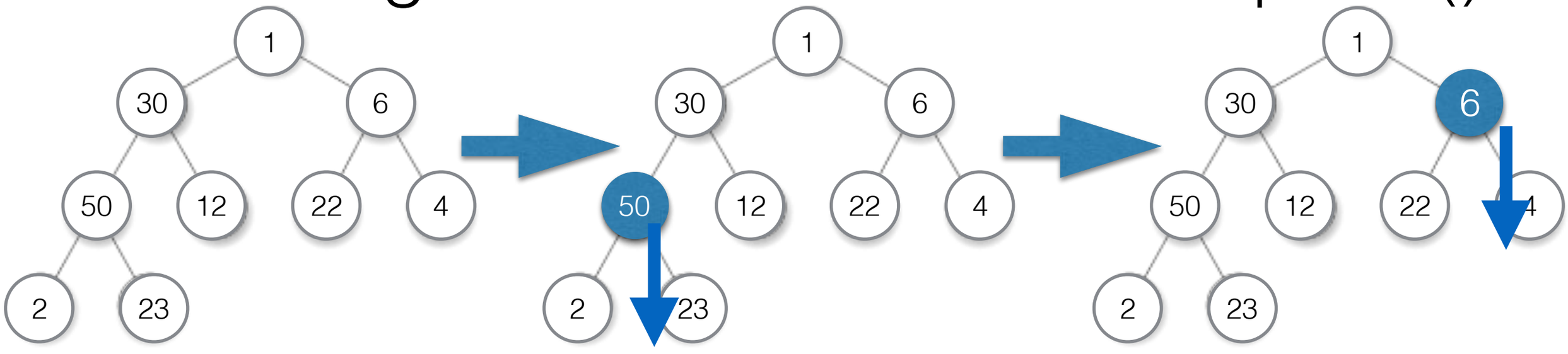


after: got a heap!

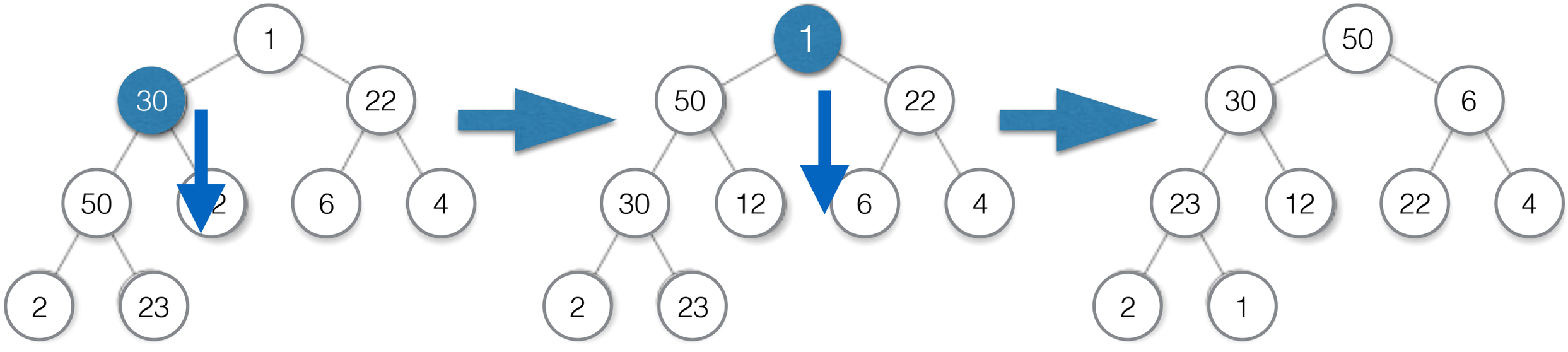
How long does it take to… extract-min()



How long does it take to… make-queue()



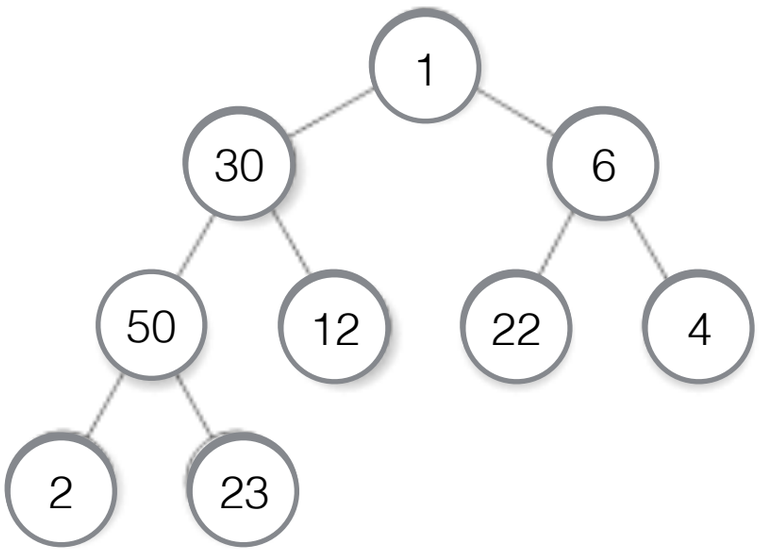
before: random values in complete binary tree



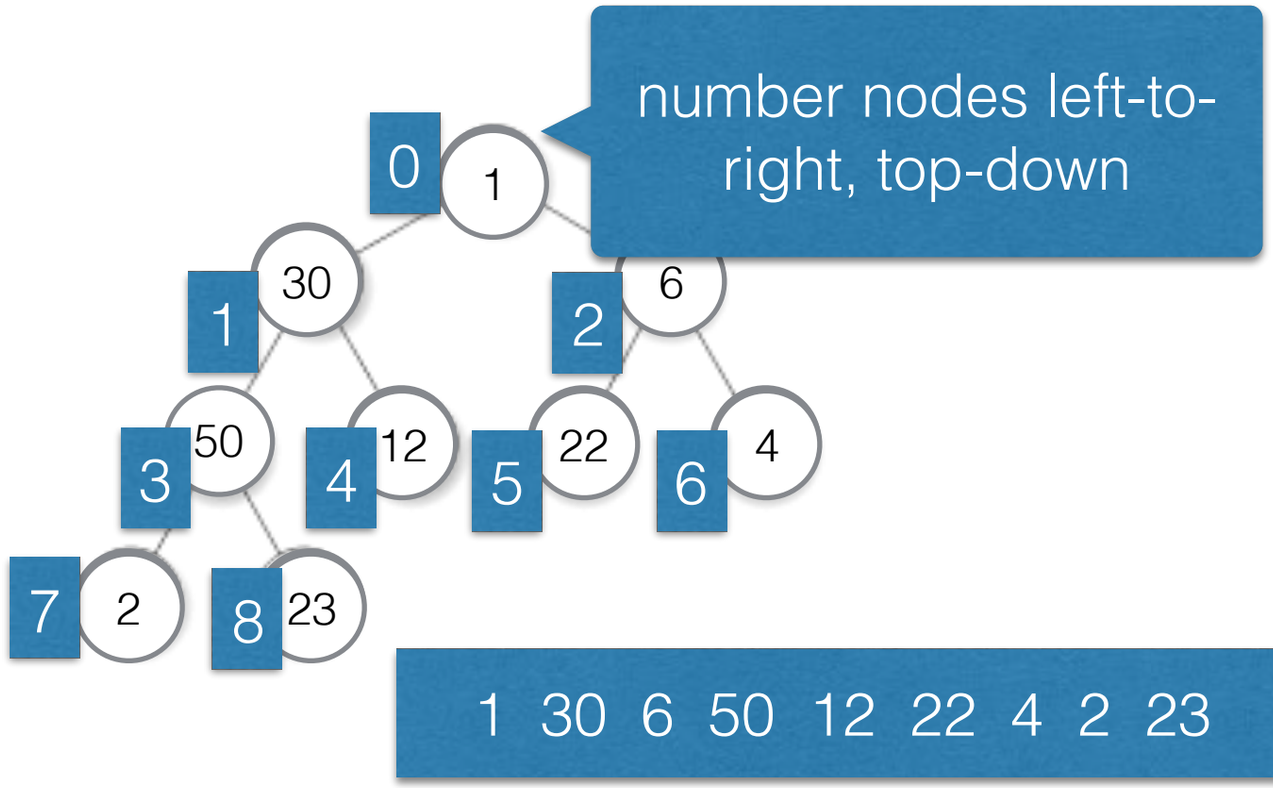
after: got a heap!

heap in memory

as explicit tree
(with tree nodes)

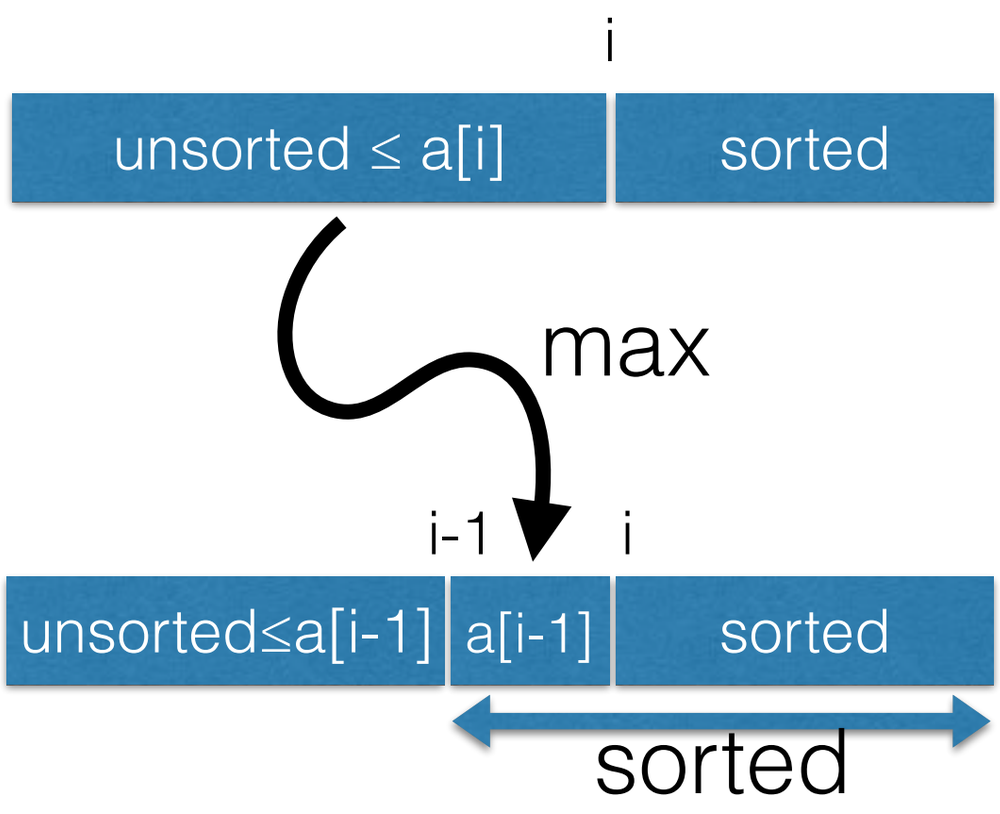


implicitly as an array

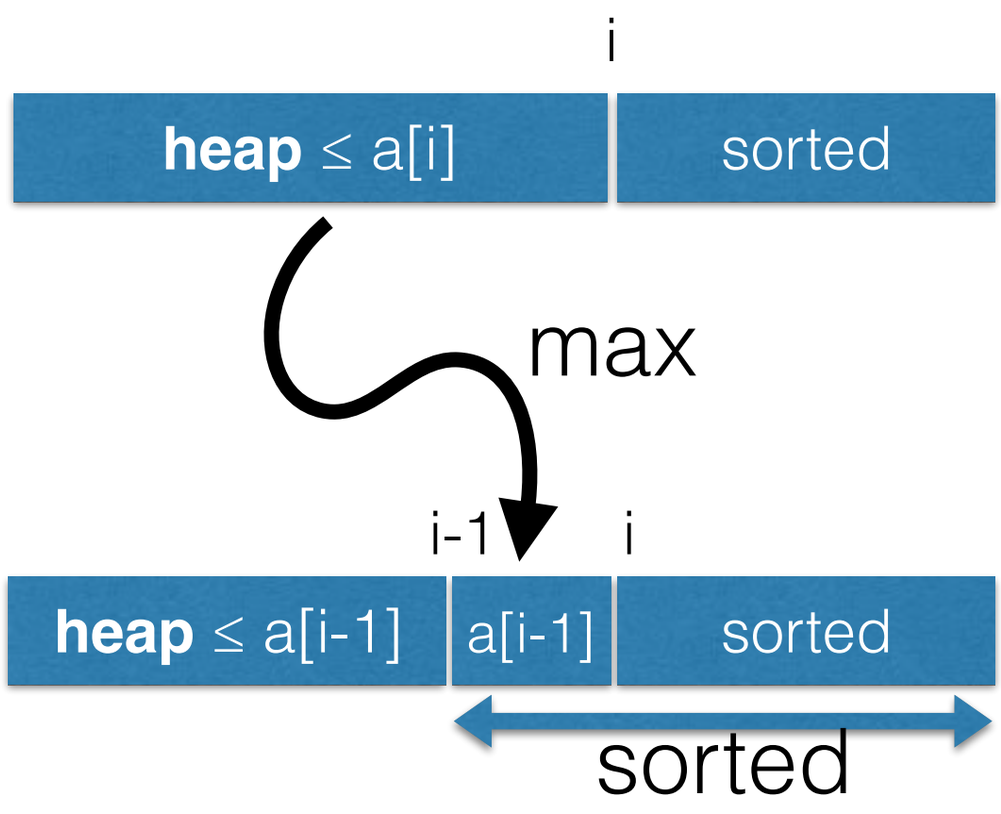


parent(7) = 3
parent(8) = 3

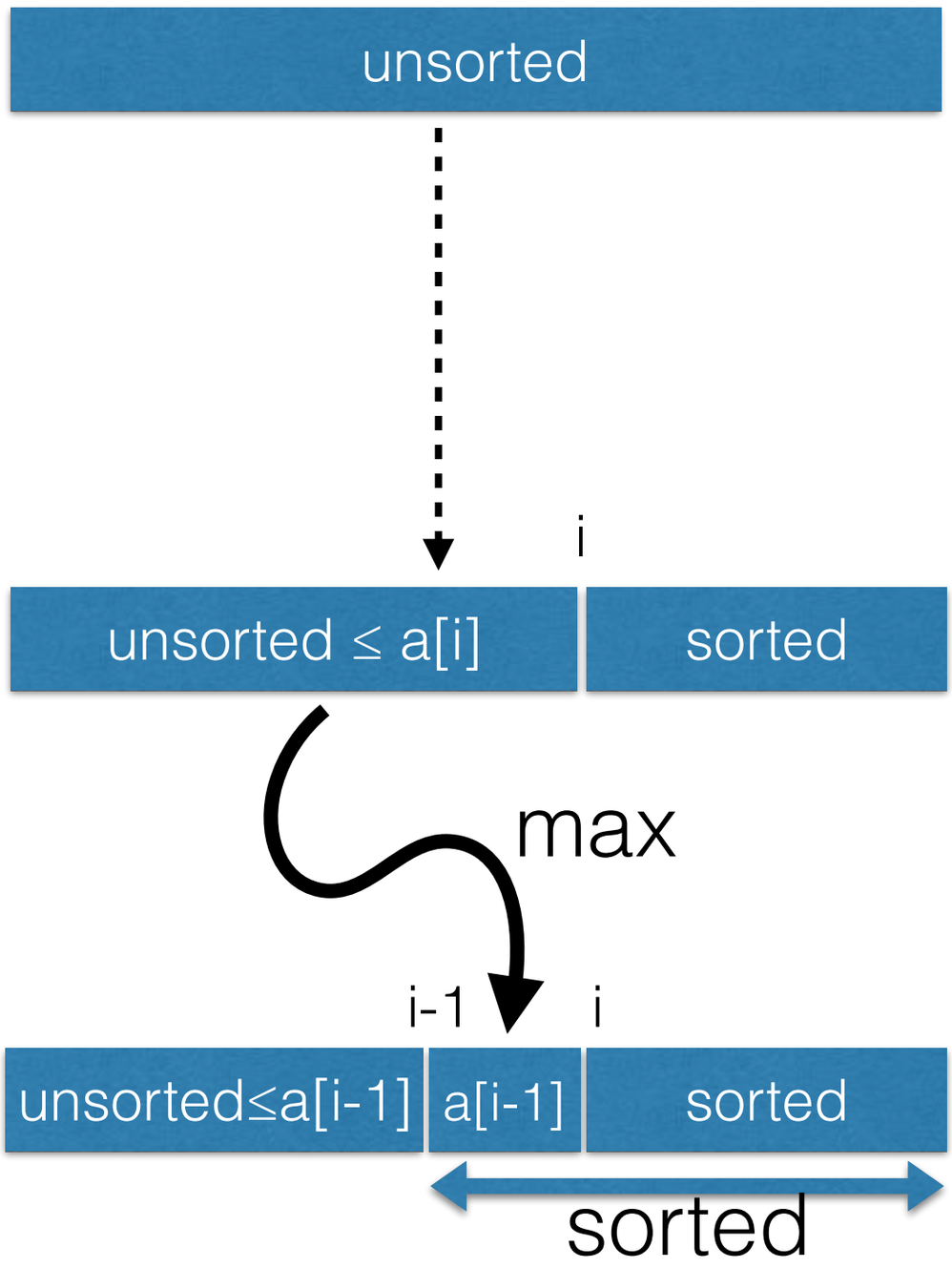
left-child(3) = 7
right-child(3) = 8



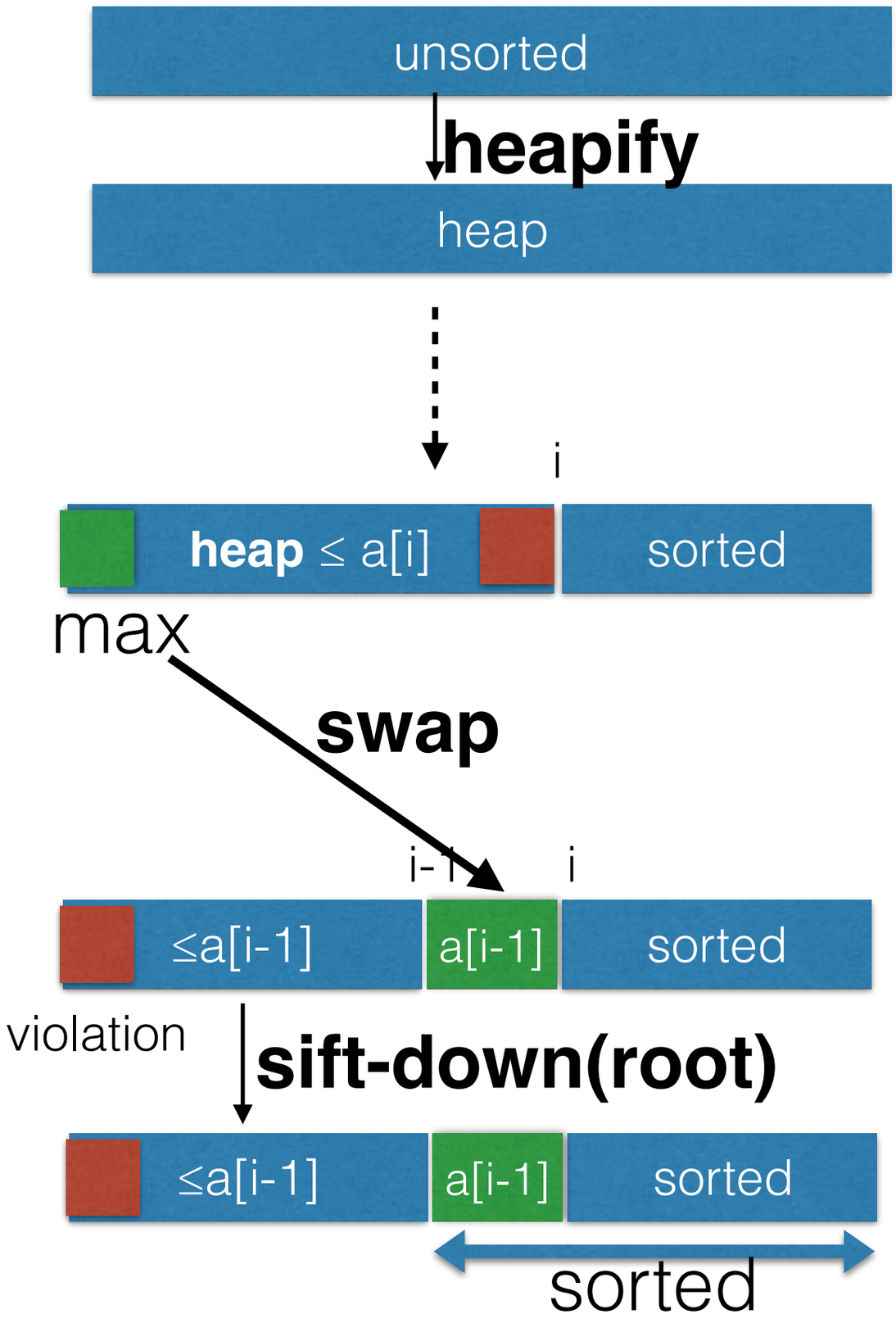
selection sort
loop invariant



heap sort
loop invariant



selection sort
loop invariant



heap sort loop invariant

Recap: sorting

- selection sort: O(n^2)
- heapsort: O(n log n) using a clever data structure, and in-place, with clever use of space
- merge sort: O(n log n) divide-and-conquer
- quicksort: O(n log n) expected: divide-and-conquer + randomization

Recap: heap

- implements the priority queue ADT
- make-queue() : O(n)
- extract-min(): O(log n)
- insert(): O(log n)
- decrease-key()/increase-key(): O(log n)