In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array.
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

Input:

32	15	2	17	19	26	41	17	17

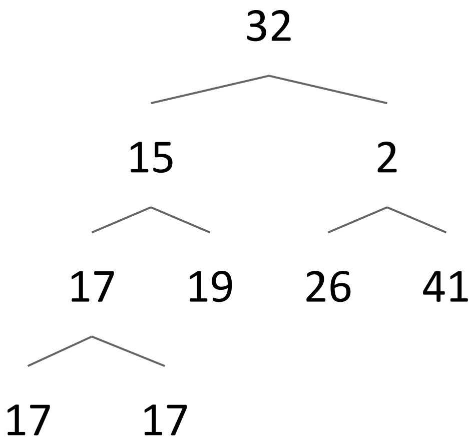
In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- **Bottom-up heapify input array:**
  - Sink nodes in reverse level order: sink(k)
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:

32	15	2	17	19	26	41	17	17



Note: This is not a heap yet!
That's why we're heapifying.

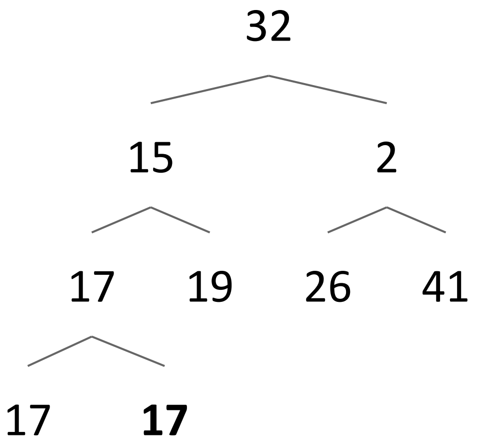
In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:

32	15	2	17	19	26	41	17	17



Sinking 17 has no effect.

In-place Heap Sort: Phase 1: Heapification

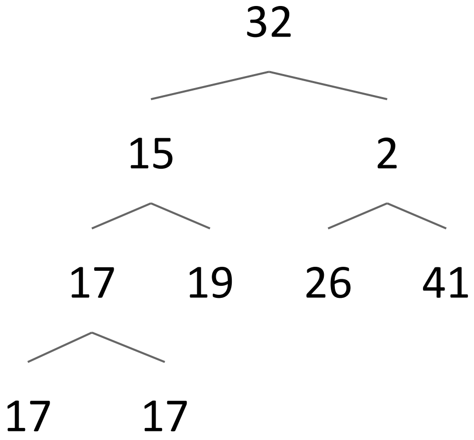
Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**

root of a heap

Input:

32	15	2	17	19	26	41	17	17



In-place Heap Sort: Phase 1: Heapification

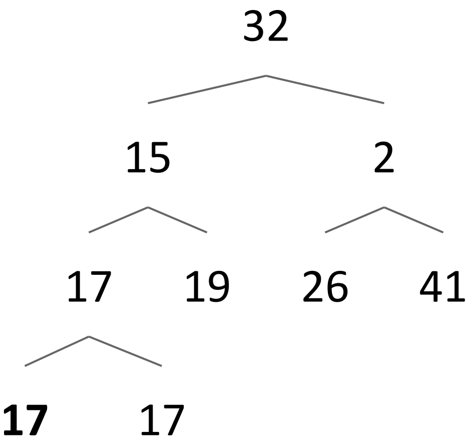
Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

root of a heap

Input:

32	15	2	17	19	26	41	17	17



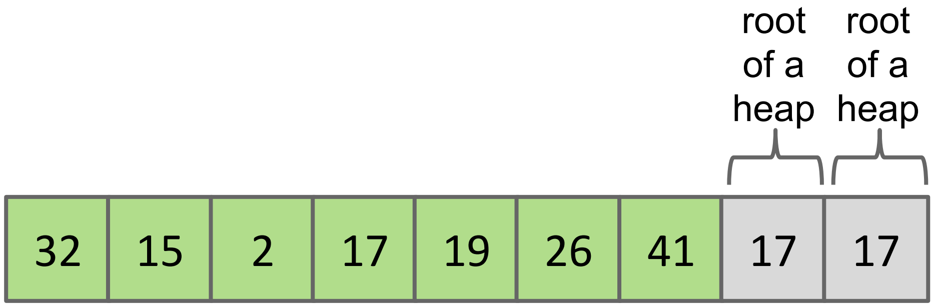
Sinking 17 has no effect.

In-place Heap Sort: Phase 1: Heapification

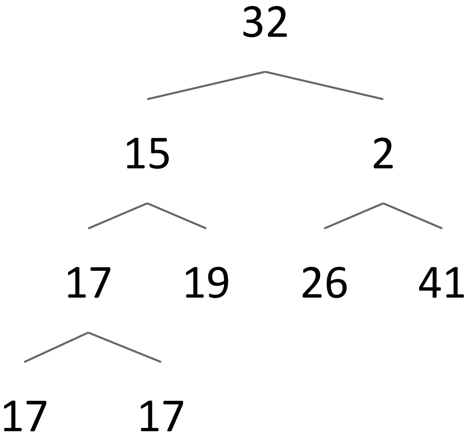
Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**

Input:



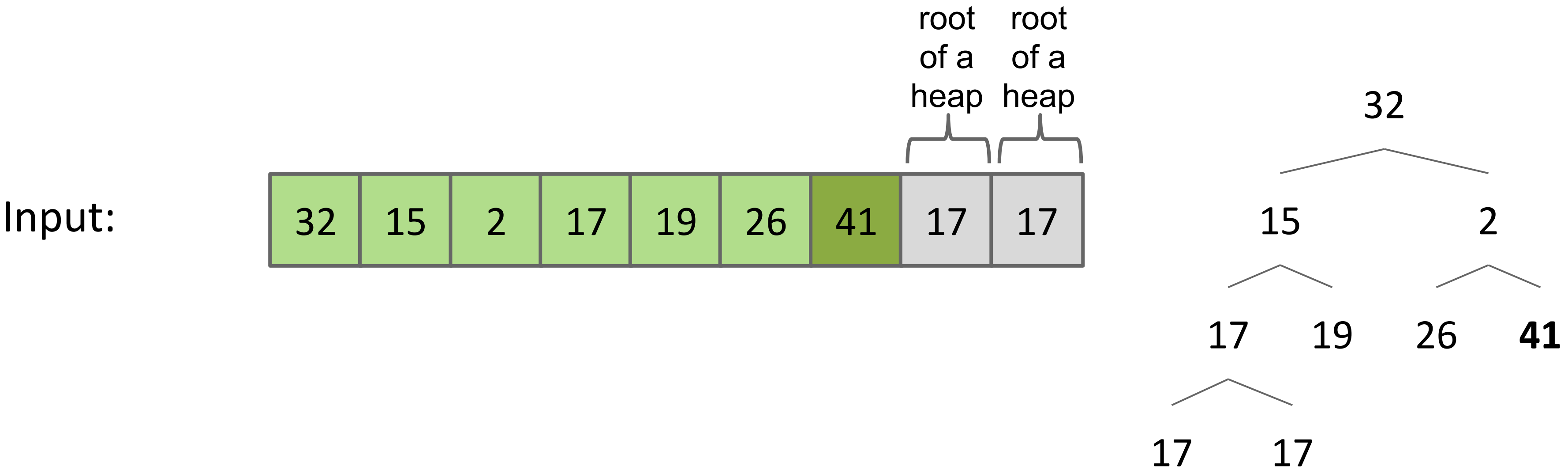
32	15	2	17	19	26	41	17	17



In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.



Input:

32	15	2	17	19	26	41	17	17

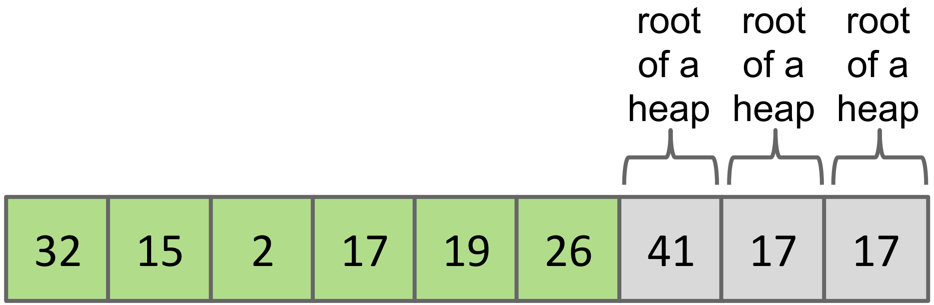
Sinking 41 has no effect.

In-place Heap Sort: Phase 1: Heapification

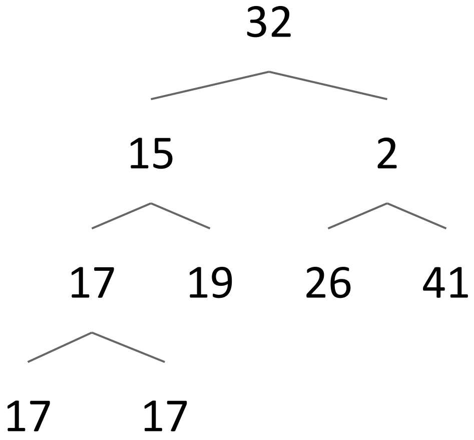
Heap sorting N items:

- Bottom-up heapify input array:
    - Sink nodes in reverse level order: sink(k)
    - **After sinking, guaranteed that tree rooted at position k is a heap.**

Input:



						root of a heap	root of a heap	root of a heap
32	15	2	17	19	26	41	17	17

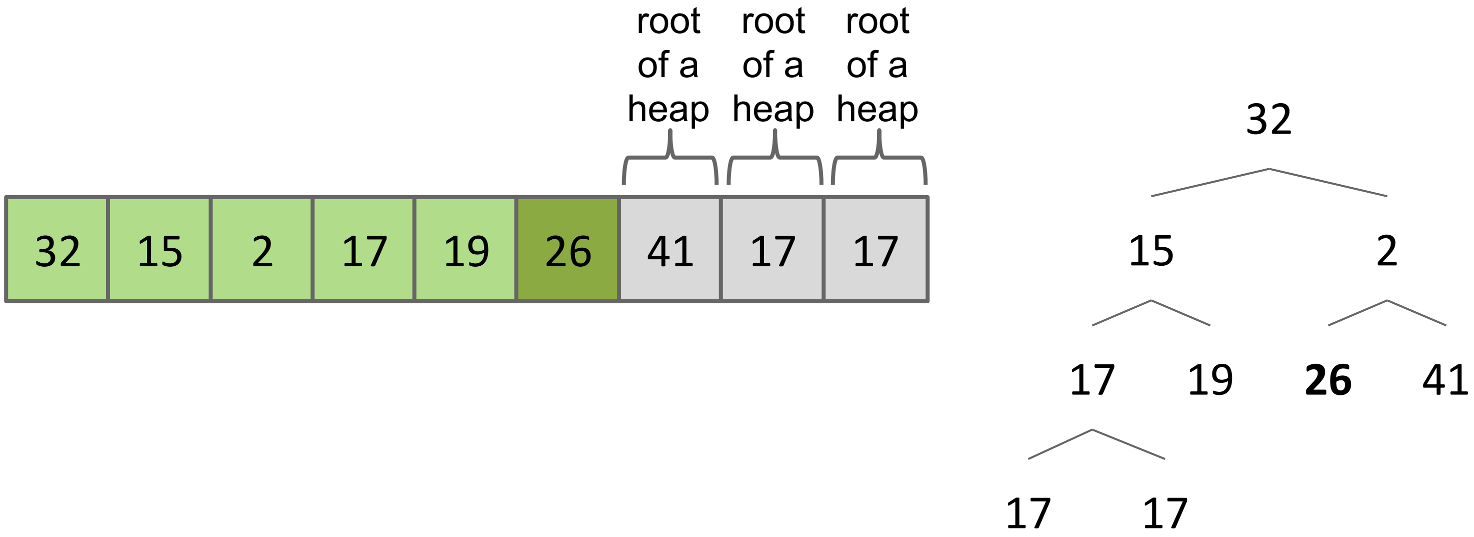


In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:



32	15	2	17	19	26	41	17	17

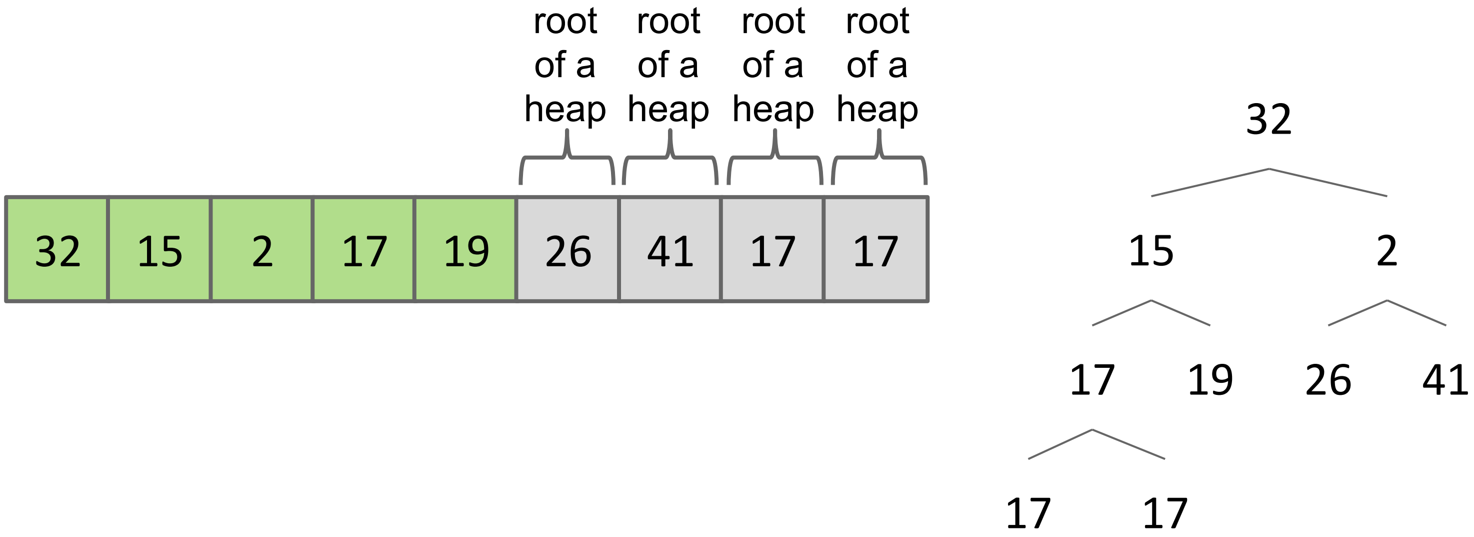
Sinking 26 has no effect.

In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
    - Sink nodes in reverse level order: sink(k)
    - **After sinking, guaranteed that tree rooted at position k is a heap.**

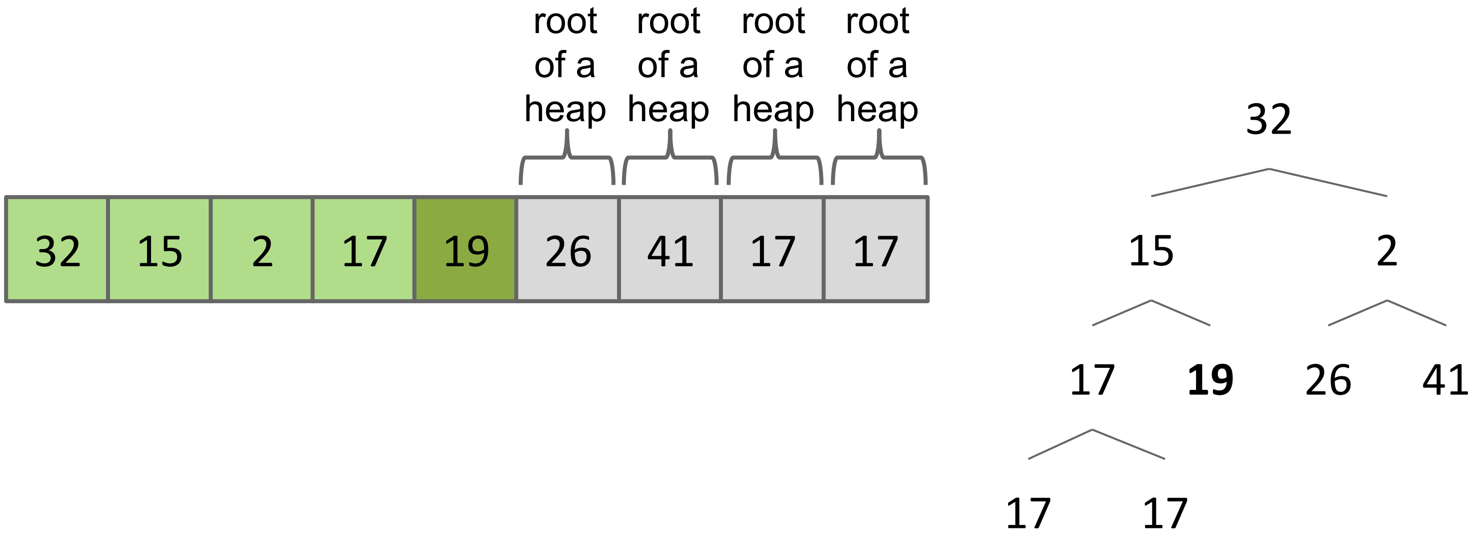
Input:



In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.



Input:

32	15	2	17	19	26	41	17	17

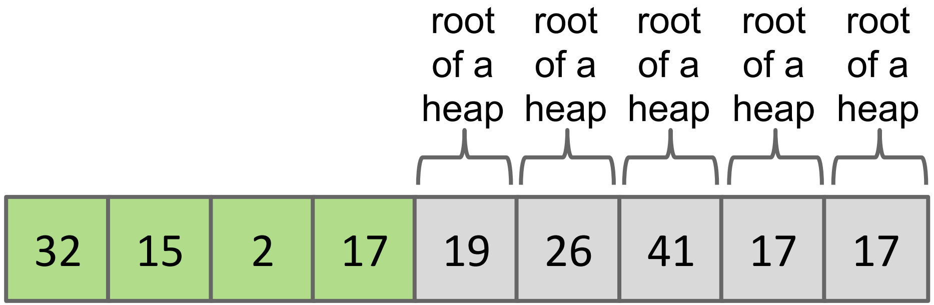
Sinking 19 has no effect.

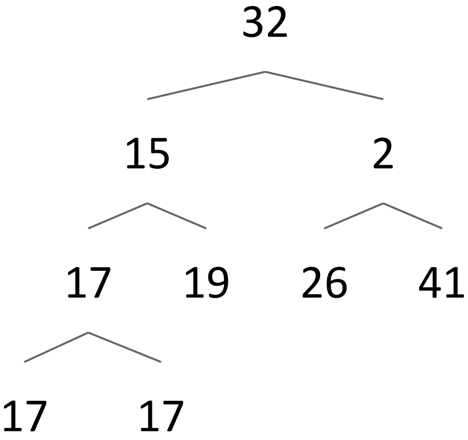
In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
    - Sink nodes in reverse level order: sink(k)
    - **After sinking, guaranteed that tree rooted at position k is a heap.**

Input:





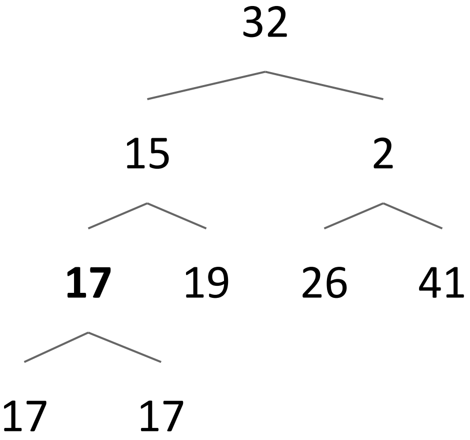
In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
    - **Sink nodes in reverse level order: sink(k)**
    - After sinking, guaranteed that tree rooted at position k is a heap.

Input:

				root of a heap	root of a heap	root of a heap	root of a heap	root of a heap
32	15	2	17	19	26	41	17	17

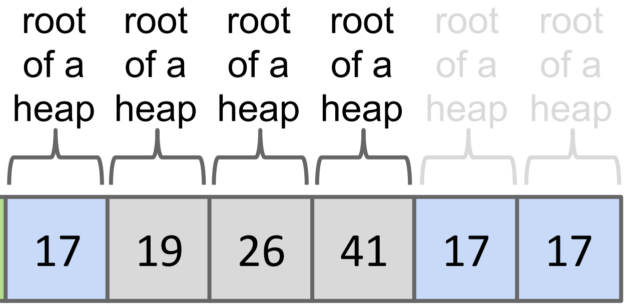


Sinking 17 has no effect.

In-place Heap Sort: Phase 1: Heapification

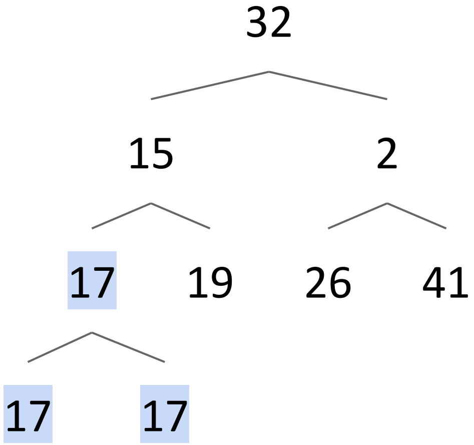
Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**



Input:

32	15	2	17	19	26	41	17	17



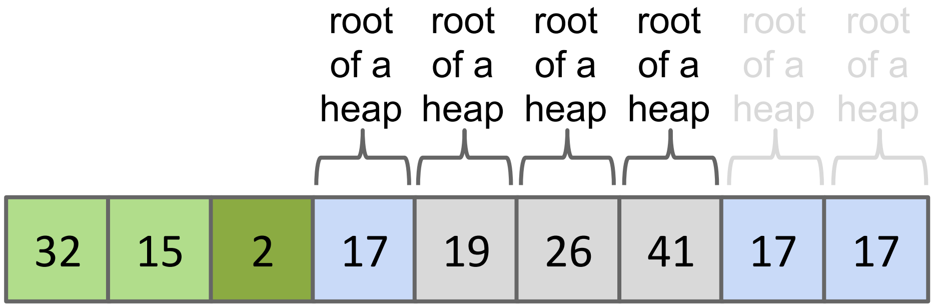
The blue coloring is to make it clear that the three 17s are all part of the same heap. I've also grayed out the "root of a heap" statement about the last two 17s since this is redundant information (all subheap nodes are also roots of that subheap).

In-place Heap Sort: Phase 1: Heapification

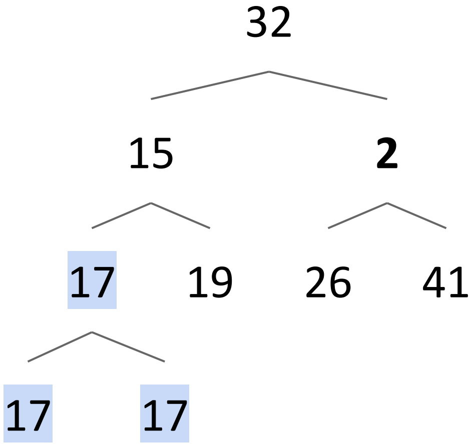
Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:



32	15	2	17	19	26	41	17	17



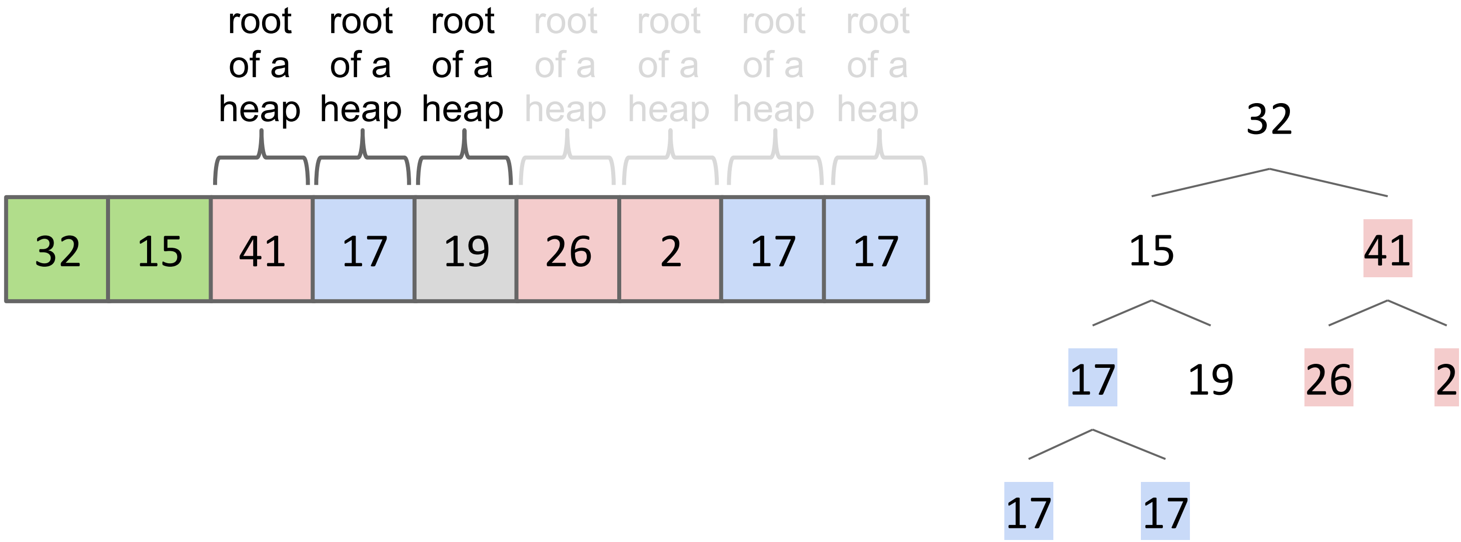
Sinking 2 does something!

In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**

Input:

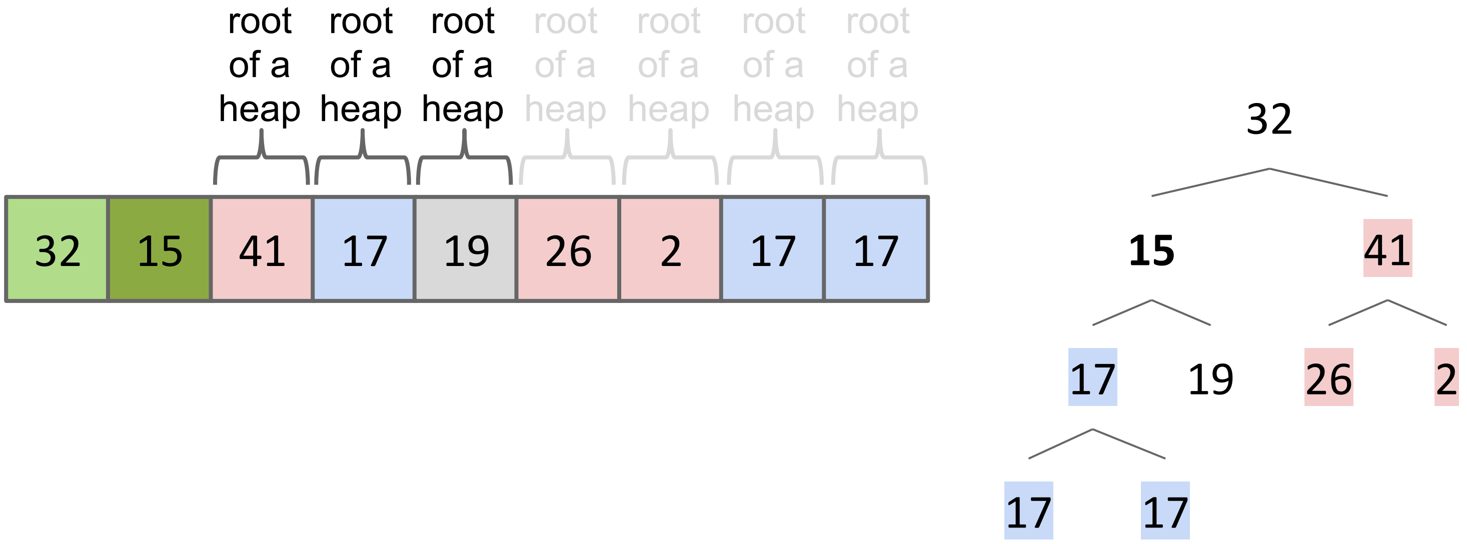


In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:



		root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap
32	15	41	17	19	26	2	17	17

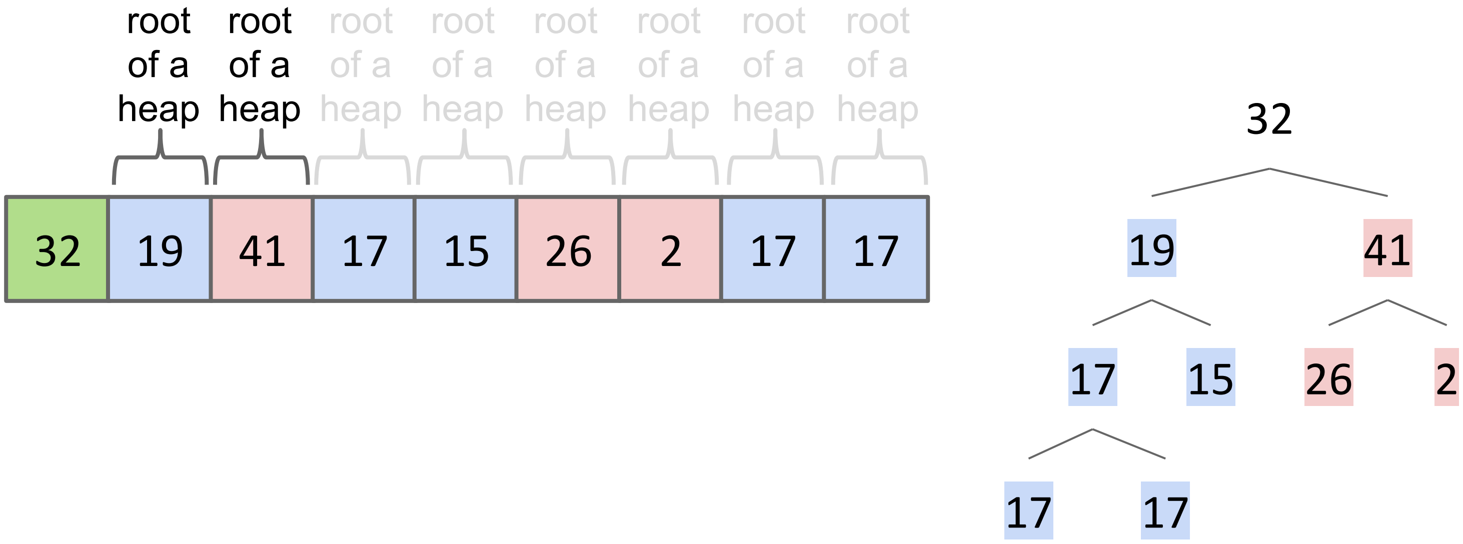
Sinking 15 does something!

In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**

Input:



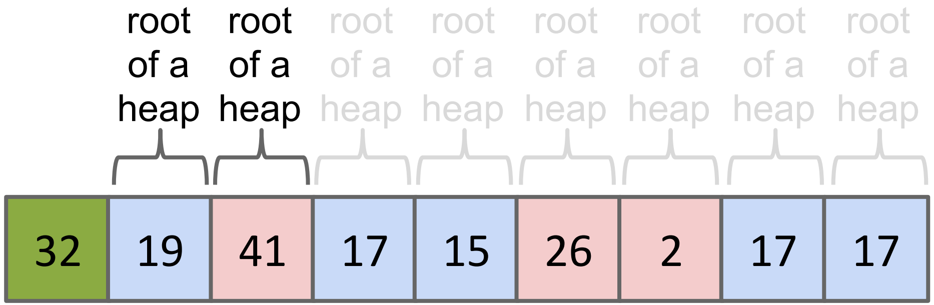
	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap
32	19	41	17	15	26	2	17	17

In-place Heap Sort: Phase 1: Heapification

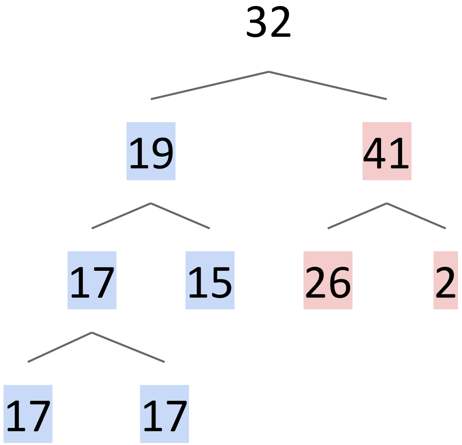
Heap sorting N items:

- Bottom-up heapify input array:
  - **Sink nodes in reverse level order: sink(k)**
  - After sinking, guaranteed that tree rooted at position k is a heap.

Input:



32	19	41	17	15	26	2	17	17



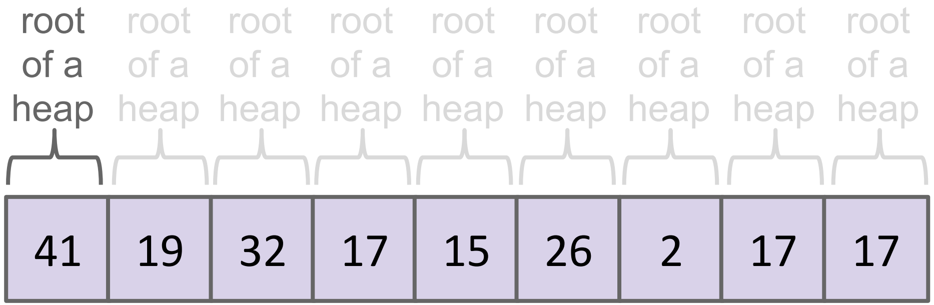
Sinking 32 does something!

In-place Heap Sort: Phase 1: Heapification

Heap sorting N items:

- Bottom-up heapify input array:
  - Sink nodes in reverse level order: sink(k)
  - **After sinking, guaranteed that tree rooted at position k is a heap.**

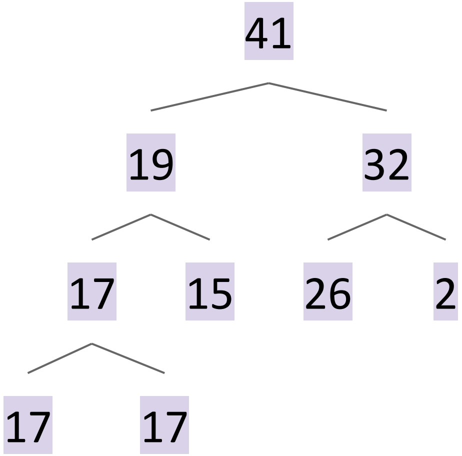
Input:



root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap	root of a heap
41	19	32	17	15	26	2	17	17

(No room to leave an unused, spot, so we will actually use position zero for this algorithm!)

Punchline: Since tree rooted at position 0 is the root of a heap, then entire array is a heap.



In-place Heap Sort

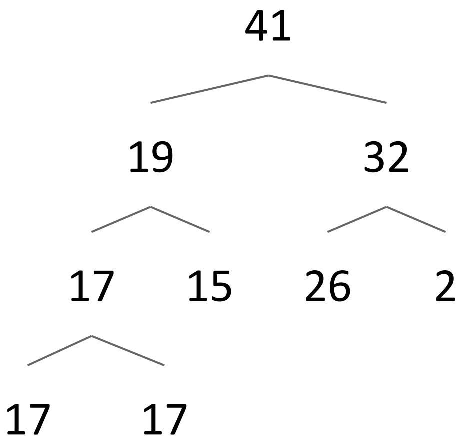
Heap sorting N items:

- **Bottom-up heapify input array (done!).**
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

Input:

41	19	32	17	15	26	2	17	17

Size: 9



In-place Heap Sort

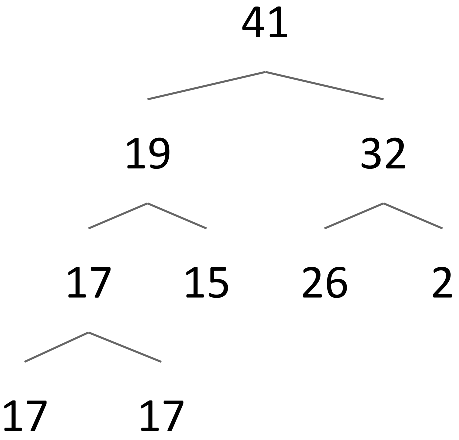
Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

Input:

41	19	32	17	15	26	2	17	17

Size: 9



In-place Heap Sort

Heap sorting N items:

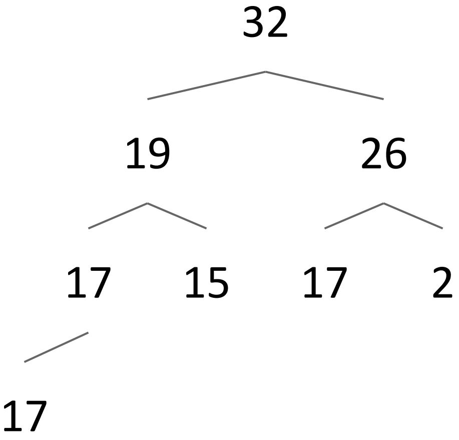
- Bottom-up heapify input array (done!).
- Repeat N times:
  - **Delete largest item from the max heap, swapping root with last item in the heap.**

Input:

sorted

32	19	26	17	15	17	2	17	41

Size: 8

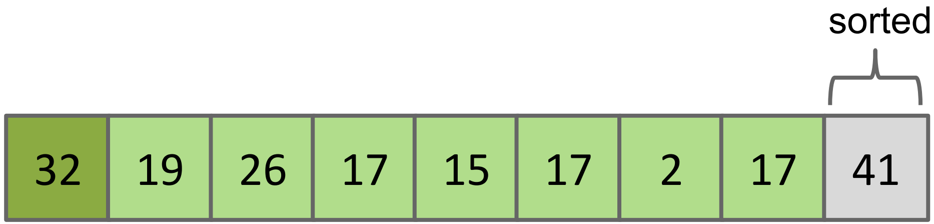


In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

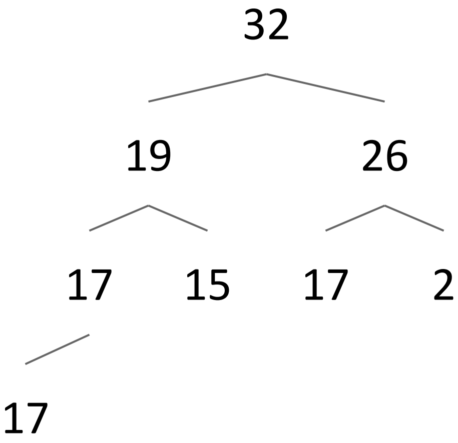
Input:



sorted

32	19	26	17	15	17	2	17	41

Size: 8

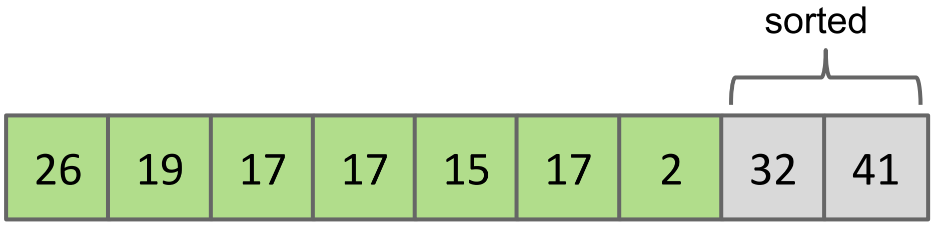


In-place Heap Sort

Heap sorting N items:

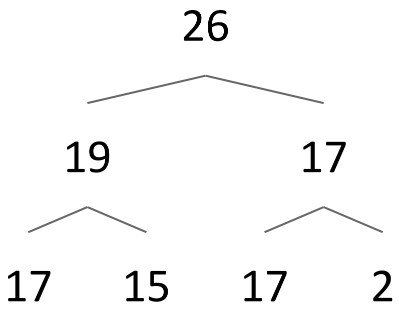
- Bottom-up heapify input array (done!).
- Repeat N times:
    - **Delete largest item from the max heap, swapping root with last item in the heap.**

Input:



26	19	17	17	15	17	2	32	41

Size: 7

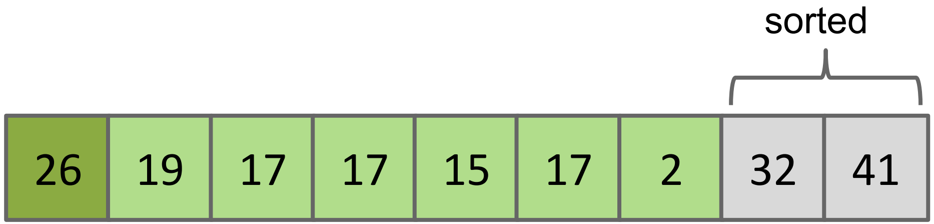


In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

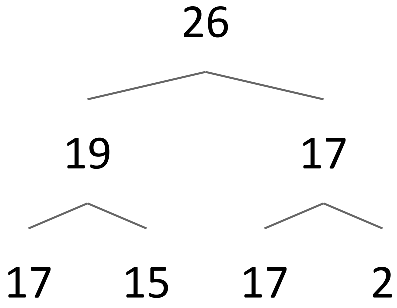
Input:



sorted

26	19	17	17	15	17	2	32	41

Size: 7



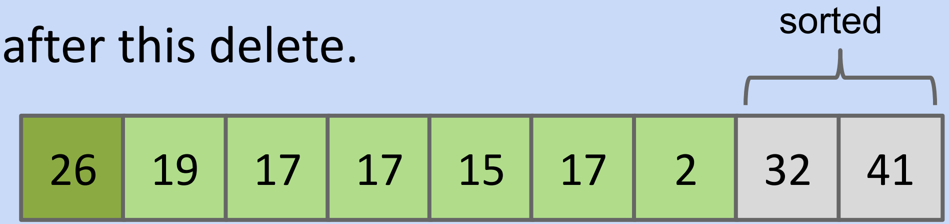
In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
    - **Delete largest item from the max heap, swapping root with last item in the heap.**

Give the array after this delete.

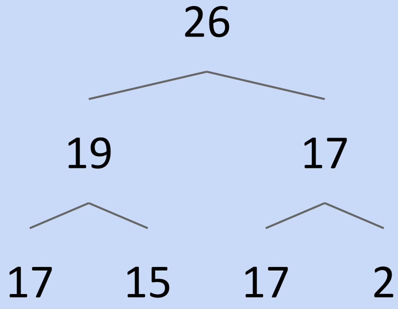
Input:



sorted

26	19	17	17	15	17	2	32	41

Size: 7

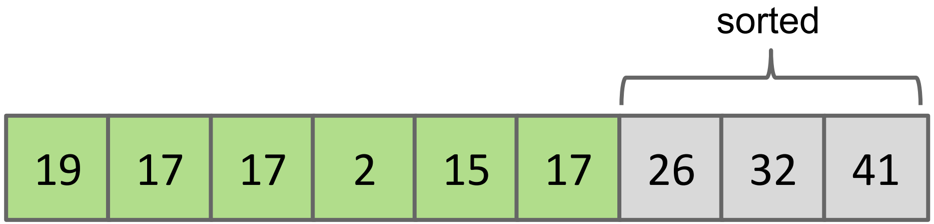


In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
  - Delete largest item from the max heap, swapping root with last item in the heap.

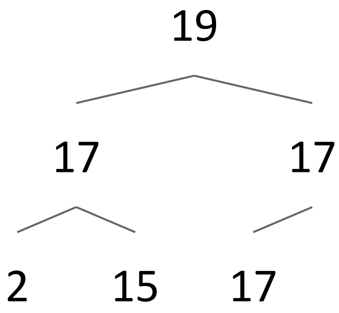
Input:



sorted

19	17	17	2	15	17	26	32	41

Size: 6

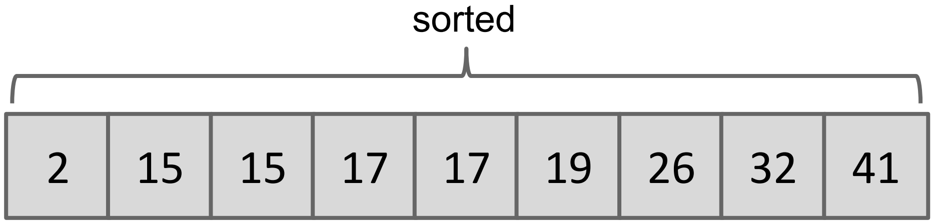


From here on out, the process is just the same, so verbose steps are omitted...

In-place Heap Sort

Heap sorting N items:

- Bottom-up heapify input array (done!).
- Repeat N times:
    - Delete largest item from the max heap, swapping root with last item in the heap.



Input:

2	15	15	17	17	19	26	32	41

Size: 0