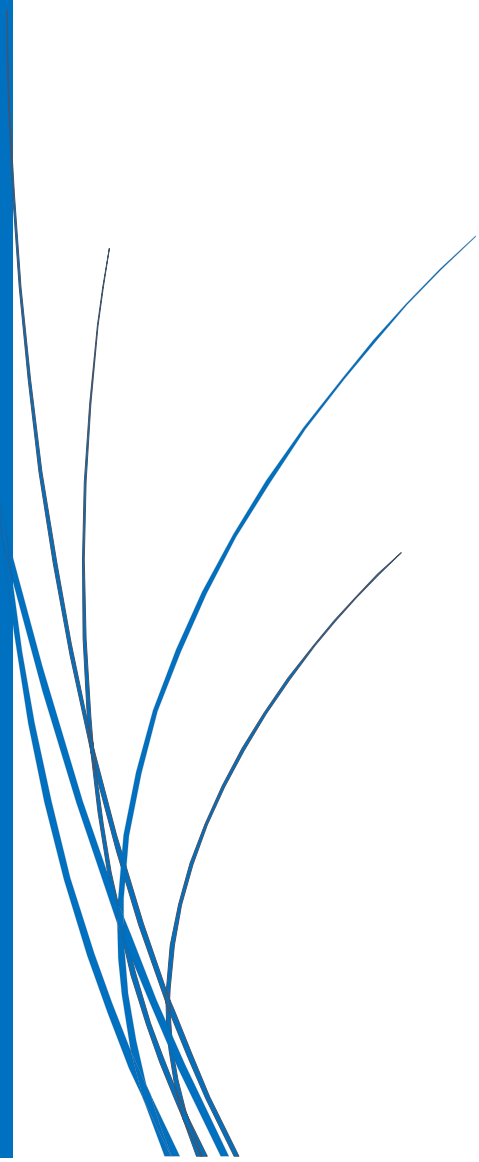
Made By:
Kenan Gazwan

Heaps

ICS202-Summary

King Fahd University of
Petroleum and Minerals



Telegram: @KenanGazwan

(Heaps)

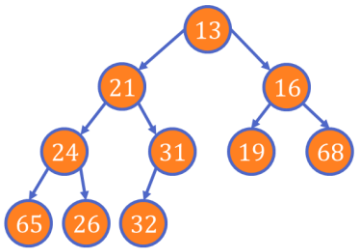
✓ Binary Heaps

A binary heap is a **complete** binary tree with one of the following heap order properties:

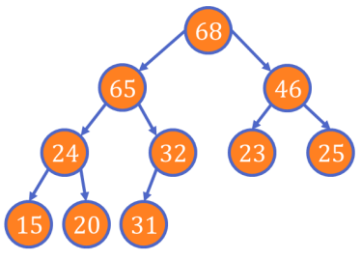
- MinHeap property: Each node key is less than or equal to the children keys.
- MaxHeap property: Each node key is greater than or equal to children keys.
  (Duplication is allowed)

Recall: A complete binary tree may have missing nodes only on the right side of the lowest level.

Illustration



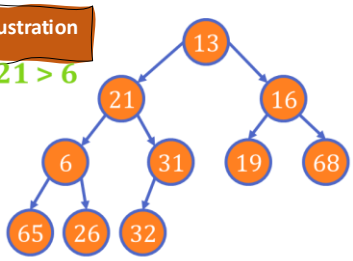
A MinHeap



A MaxHeap

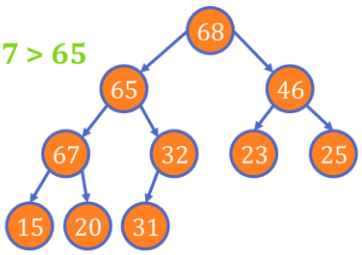
Illustration

21 > 6

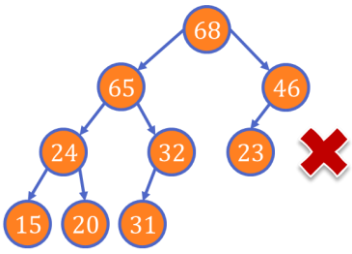


Not a MinHeap

67 > 65



Not a MaxHeap



Not A Heap

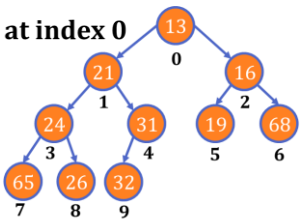
Violates heap structural property

A heap is manipulated more efficiently using an array.

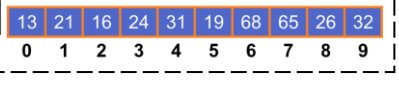
Array Representation of A MinHeap:

Illustration

Starting at index 0

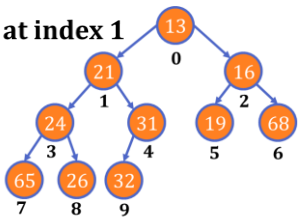


For array[i]	
Root	array[0]
Parent	array[(i – 1) / 2]
Left Child	array[2i + 1]
Right Child	array[2i + 2]

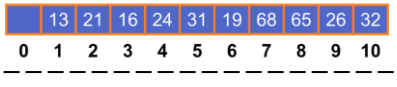


Illustration

Starting at index 1



For array[i]	
Root	array[1]
Parent	array[i/ 2]
Left Child	array[2i]
Right Child	array[2i + 1]



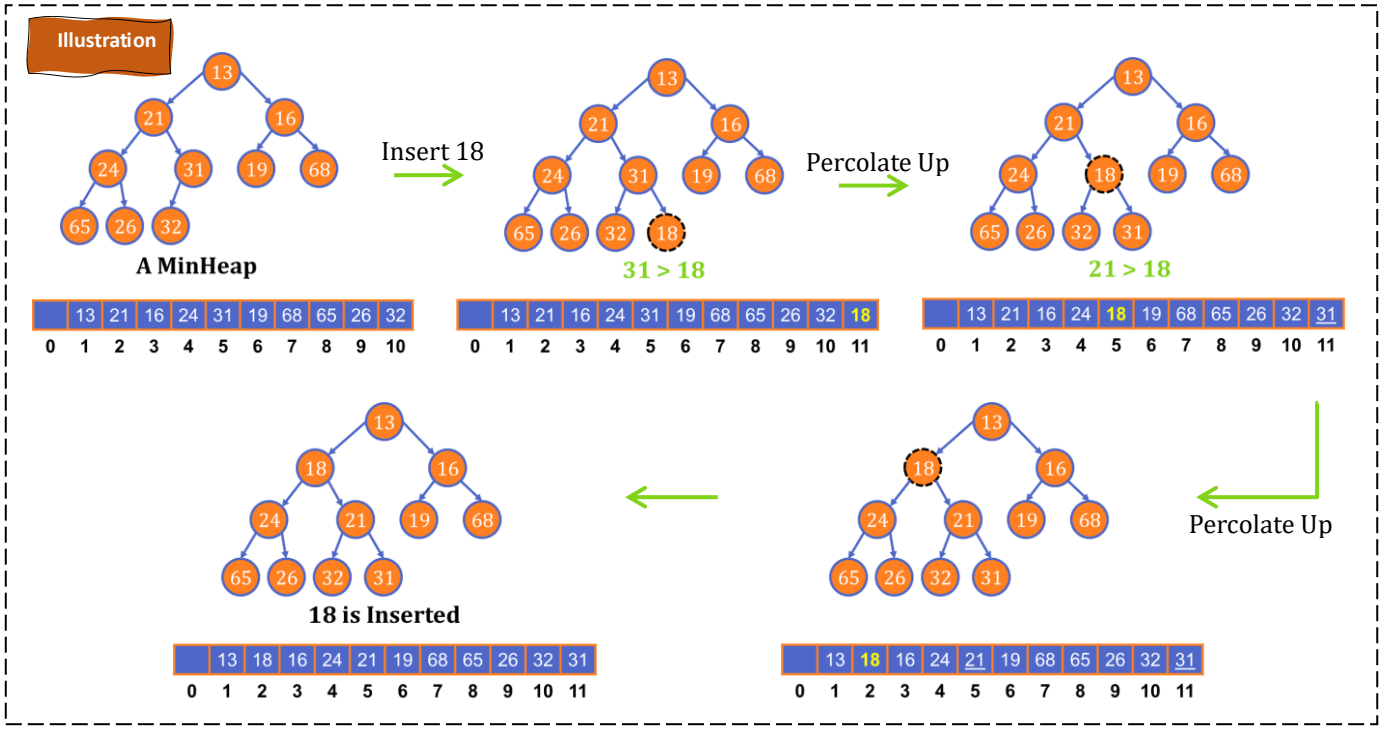
Advantageous of Array Representation:

- The Main Operations are $\boldsymbol{O(1)}$:
  - Add a node at the end of array
  - Find Parent / Child
  - Swap Parent and Child
- A lot of dynamic memory allocation of tree nodes is avoided

Percolate Up/Down: The process of swapping an element with its parent to restore the heap order property.

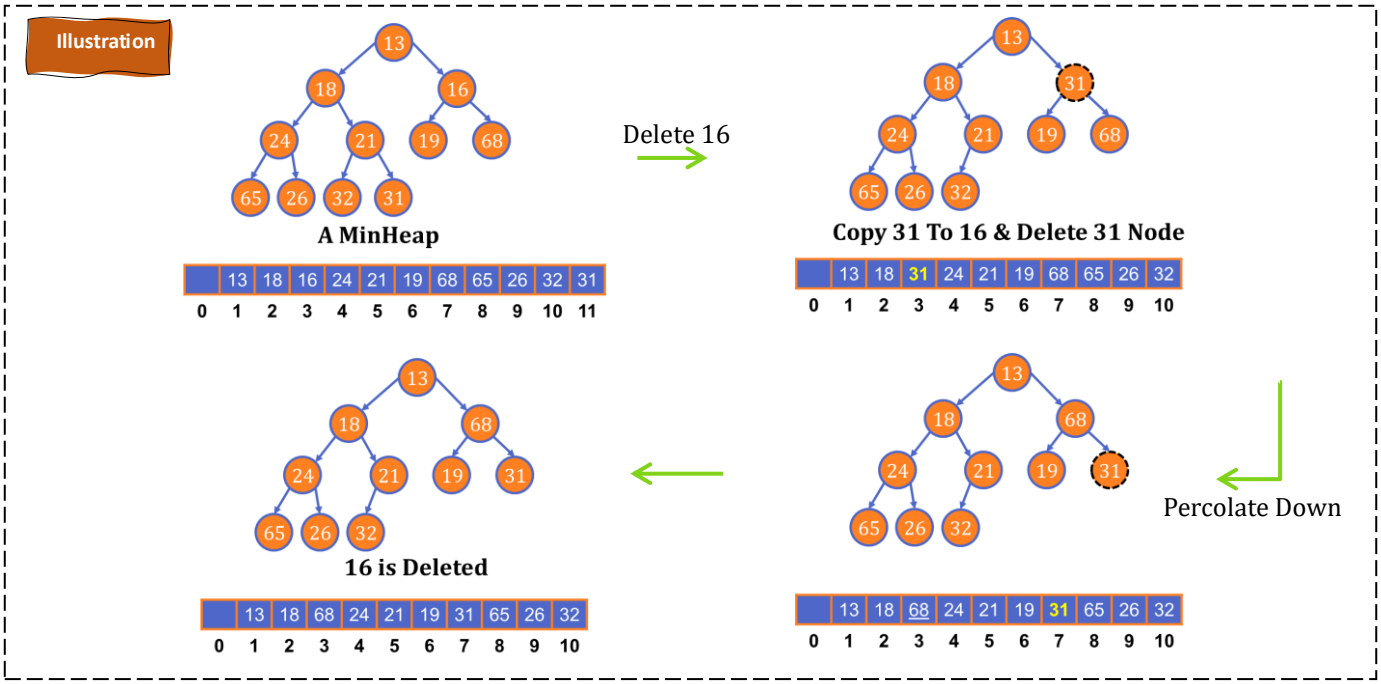
Heap Insertion: $O(log\ n)$

- ✓ Insert the key at the end of the heap.
- ✓ As long as the heap order property is violated: percolate up.



Heap Deletion: $O(log\ n)$

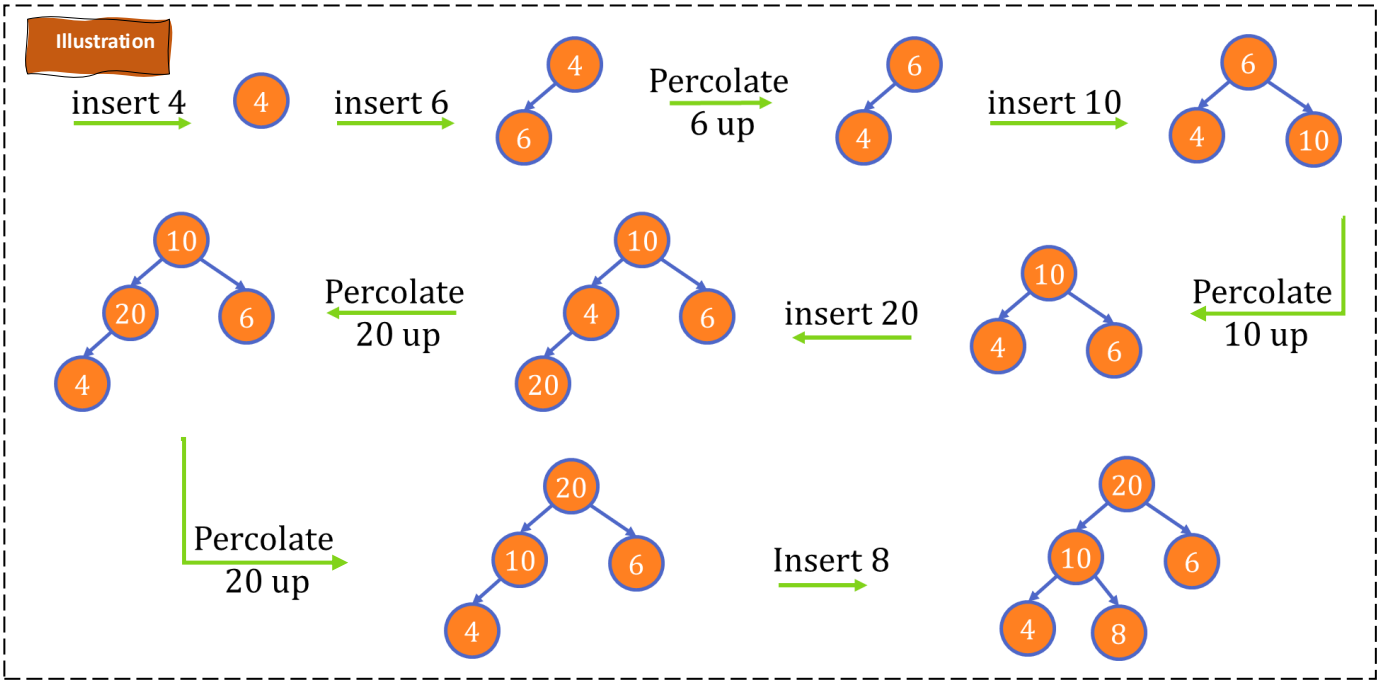
- ✓ Copy the last key node to the deleted node
- ✓ Delete the last node
- ✓ As long as the heap order property is violated: percolate up/down



Building Heaps:

✓ **Building A Heap (Top Down): $O(n\ logn)$**

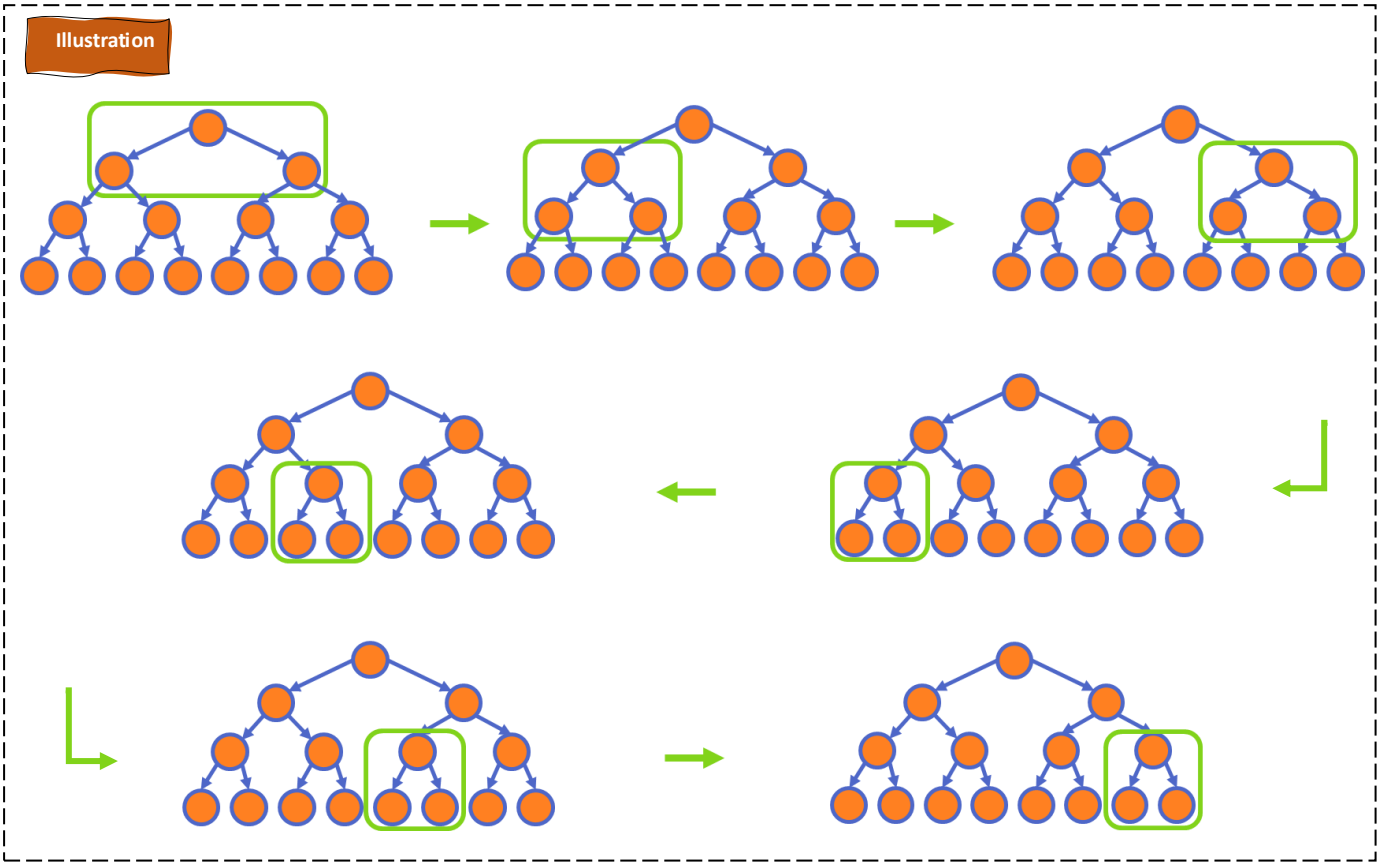
Insert 4, 6, 10, 20, and 8 into MaxHeap:



✓ **Convert An Array Into A Heap (Top Down): $O(n\ logn)$**

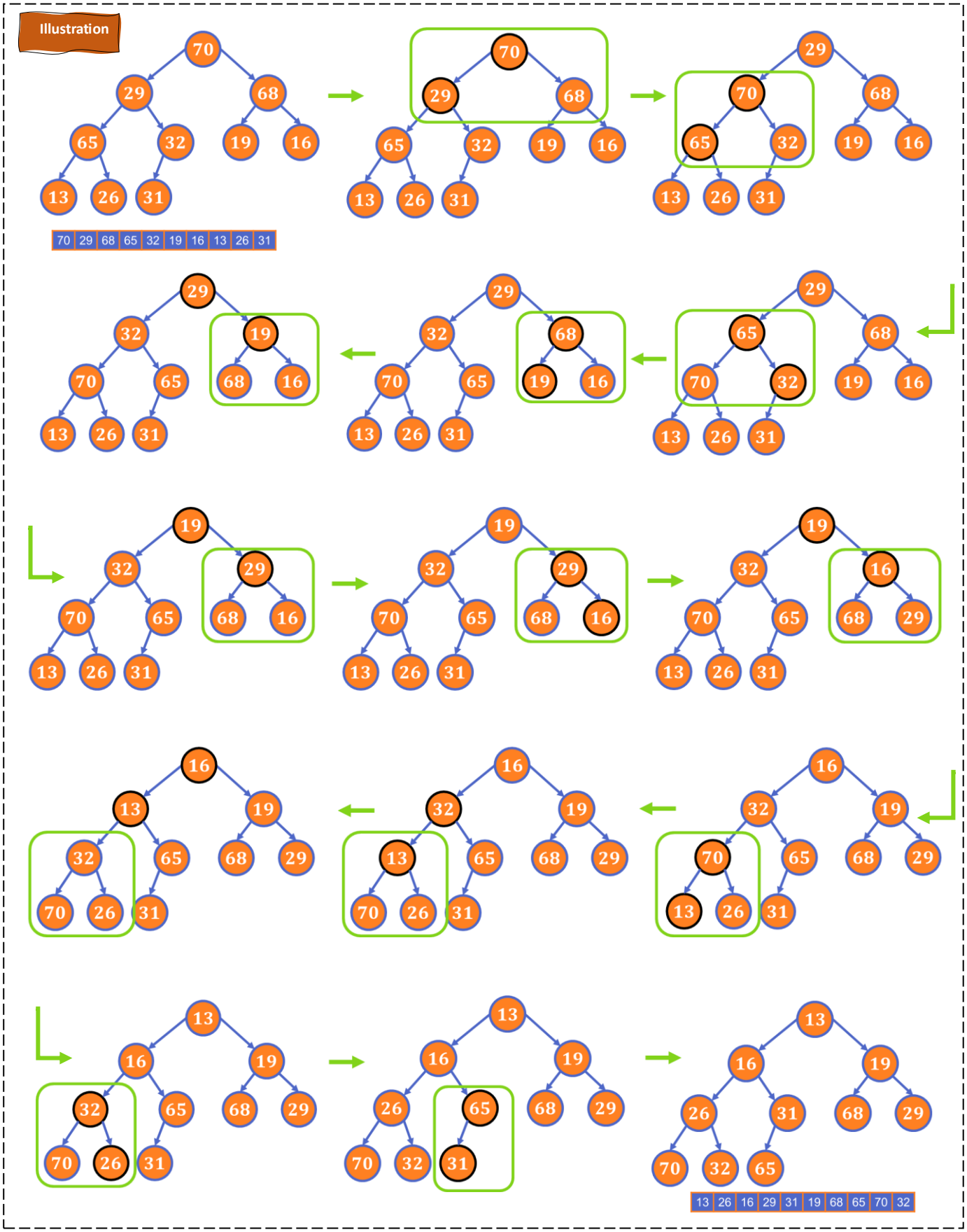
<u>The Procedure of Conversion:</u>

Always solve the nodes inside the current rectangle until it reaches the heap property. You can solve above the current rectangle if needed but not below it.



✓ **Example: Convert An Array Into A Heap (Top Down):**

It is solved in detail. It might look long; however, it is just a systematic procedure 😊

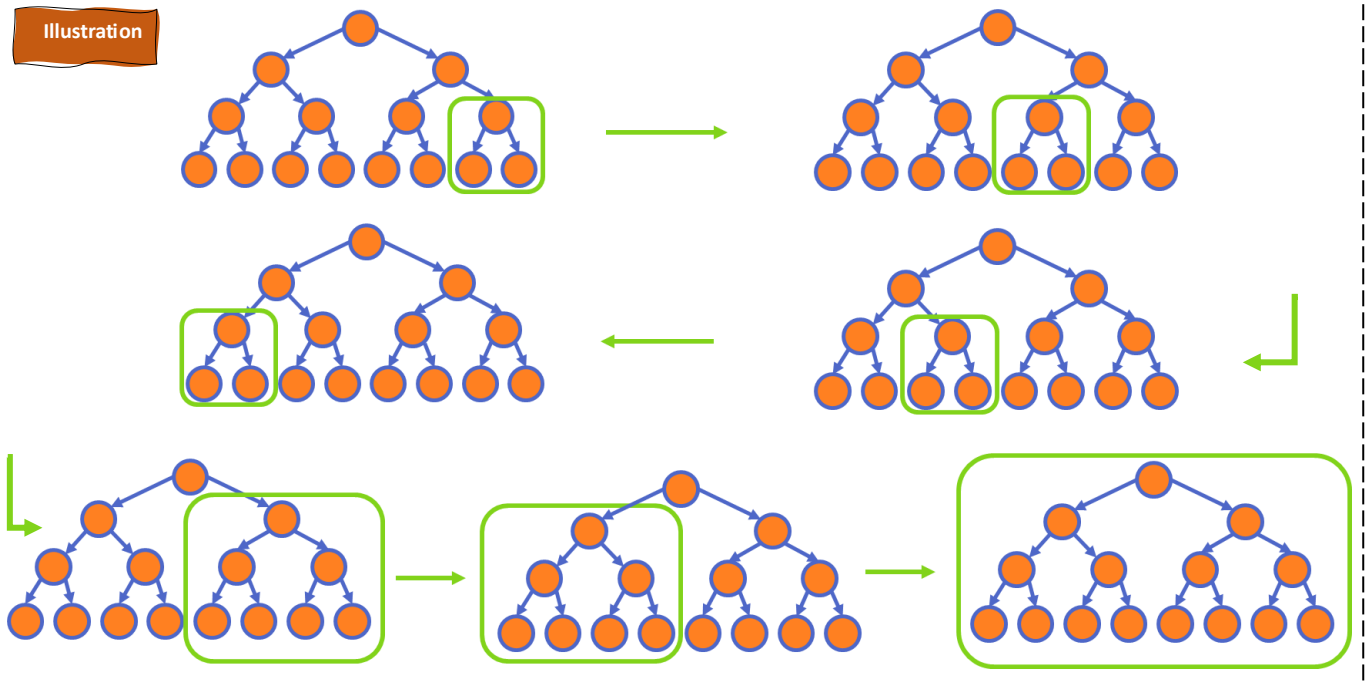


✓ **Convert An Array Into A Heap (Bottom Up): $O(n)$**

The Procedure of Conversion:

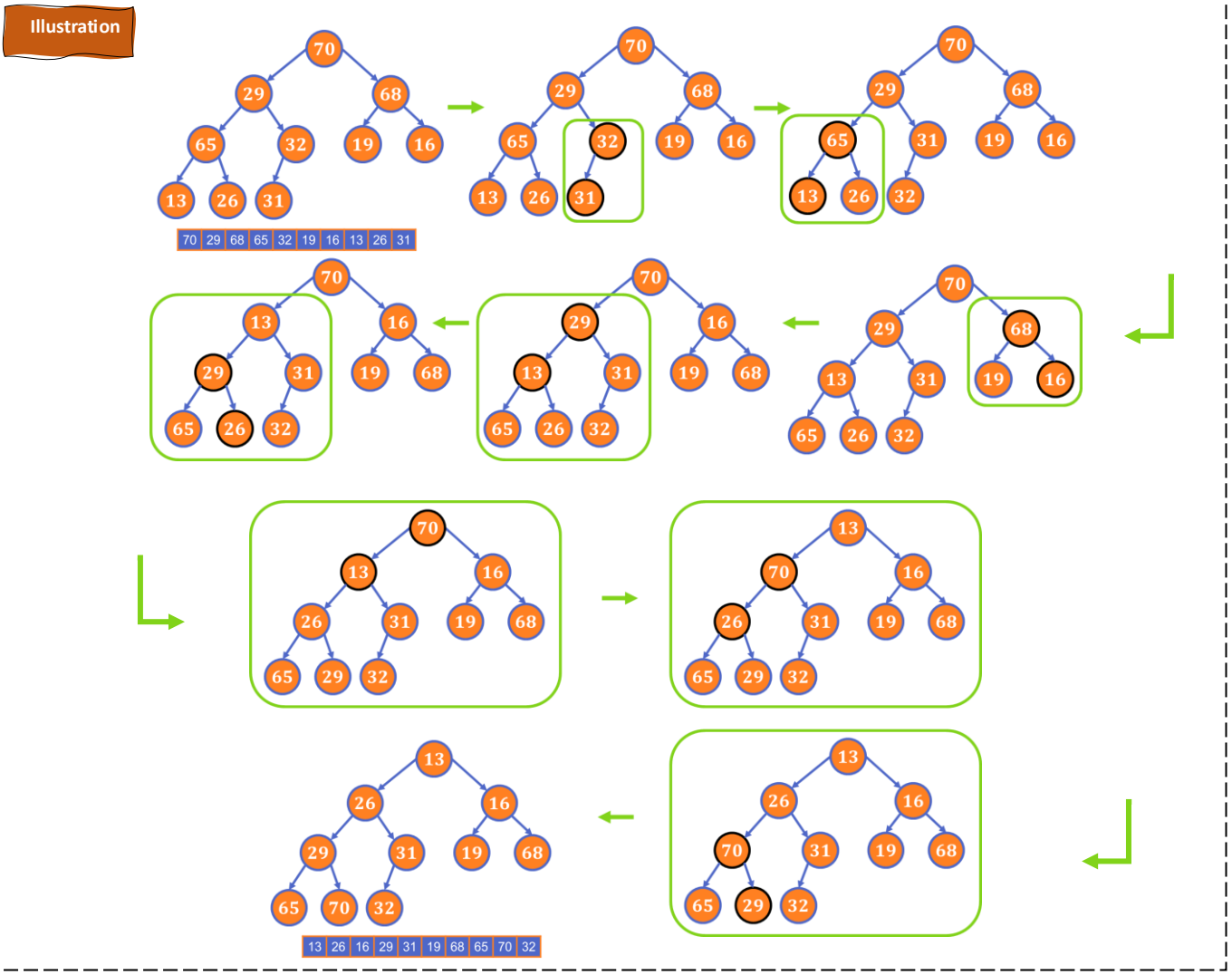
Always solve the nodes inside the current rectangle until it reaches the heap property.

Illustration



✓ **Example: Convert An Array Into A Heap (Bottom Up):**

Illustration



Heap sort steps:

- Build a min / max heap from an unsorted array.
- Remove the current minimums / maximums from the heap *n* times and store in an array.

The overall complexity of Heap Sort = $O(n\ logn)$

Heap Operations Complexity	
Add a node at the end of array	$\boldsymbol{O(1)}$
Find Parent / Child	$\boldsymbol{O(1)}$
Swap Parent and Child	$\boldsymbol{O(1)}$
Heap Insertion	$\boldsymbol{O(log\ n)}$
Heap Deletion	$\boldsymbol{O(log\ n)}$
Top-Down Building / Converting	$\boldsymbol{O(n\ logn)}$
Bottom-Up Converting	$\boldsymbol{O(n)}$
Delete Max/Min	$\boldsymbol{O(log\ n)}$
HeapSort	$\boldsymbol{O(n\ logn)}$