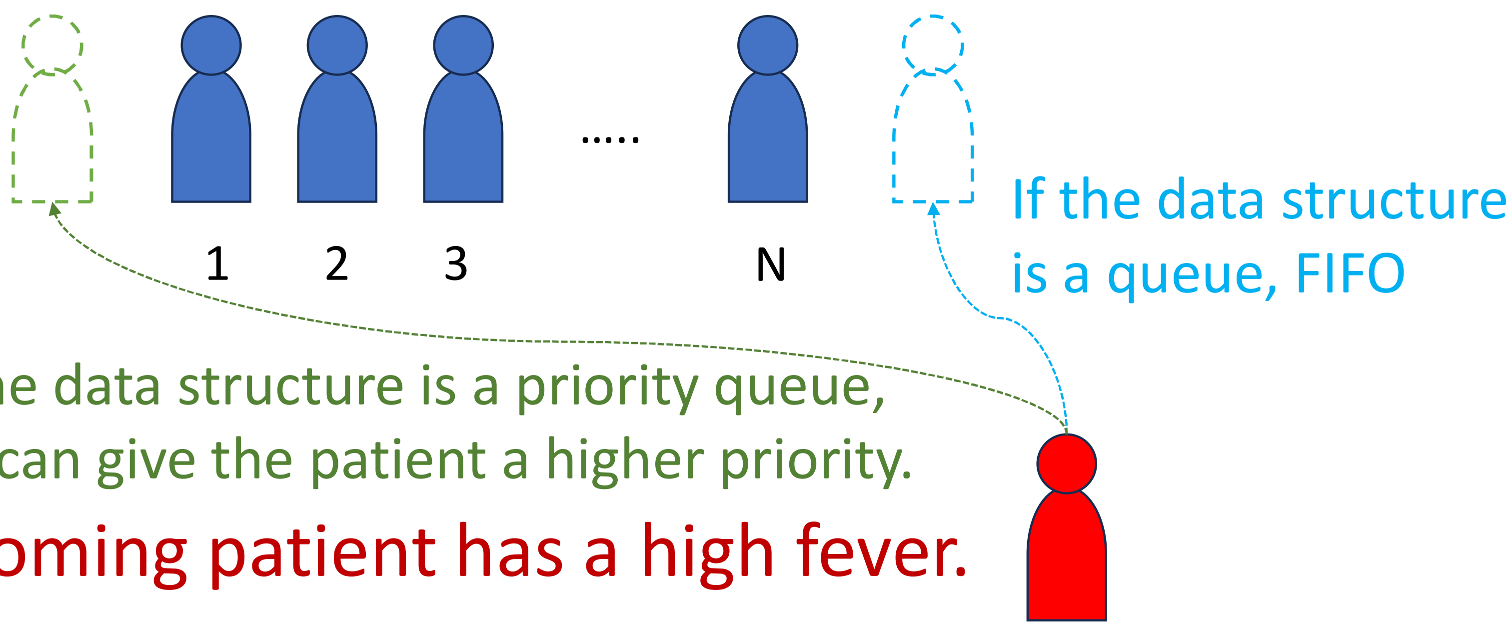
Heaps

Ch. 5.6

Priority queues

- In the hospital, patients are waiting.



- A new coming patient has a high fever.

Priority queues

- Using <u>unordered</u> linear list to represent priority queues.

5	5	5	5	5	5	5

10

- Operations:

Operation	**Unordered list** *Time complexity*	**Min Heap** *Time complexity*
IsEmpty()	O(*1*)	O(*1*)
Top()	O(*n*)	O(1)
Push()	O(*1*)	O(*log n*)
Pop()	O(*n*)	O(*log n*)

Min tree

- Each node has a key.
- Key value in any node is the minimum value in the subtree.

The root has the smallest key.

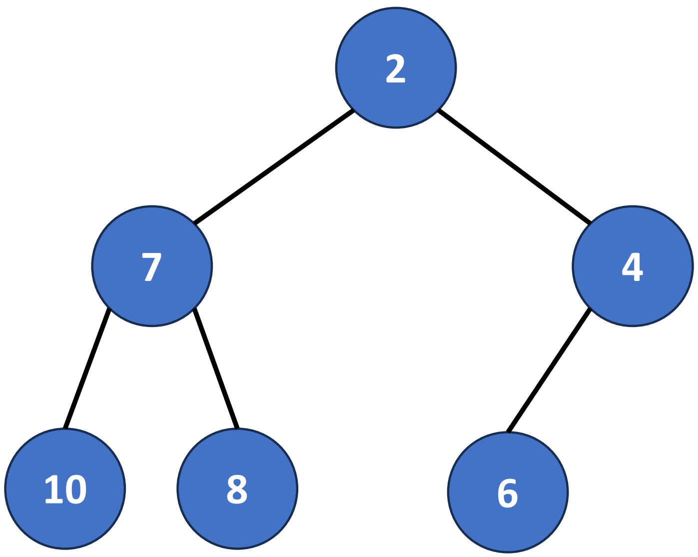
Max tree

- Each node has a key.
- Key value in any node is the maximum value in the subtree.

The root has the largest key.

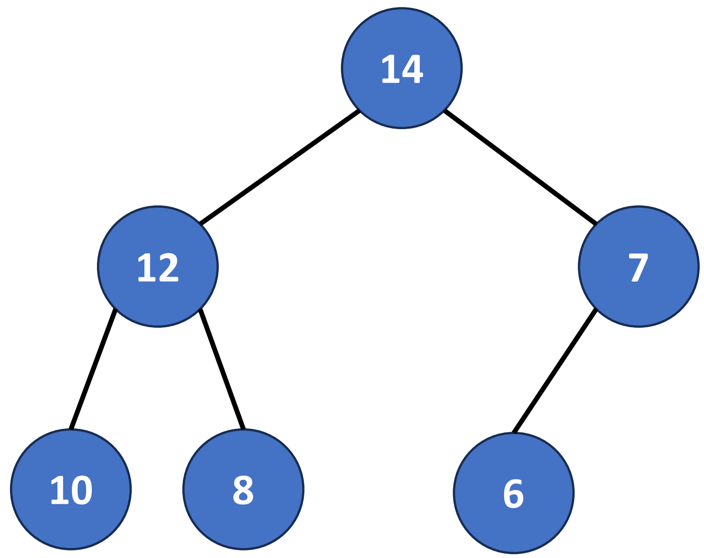
Min Heap

- A complete binary tree
- A min tree.

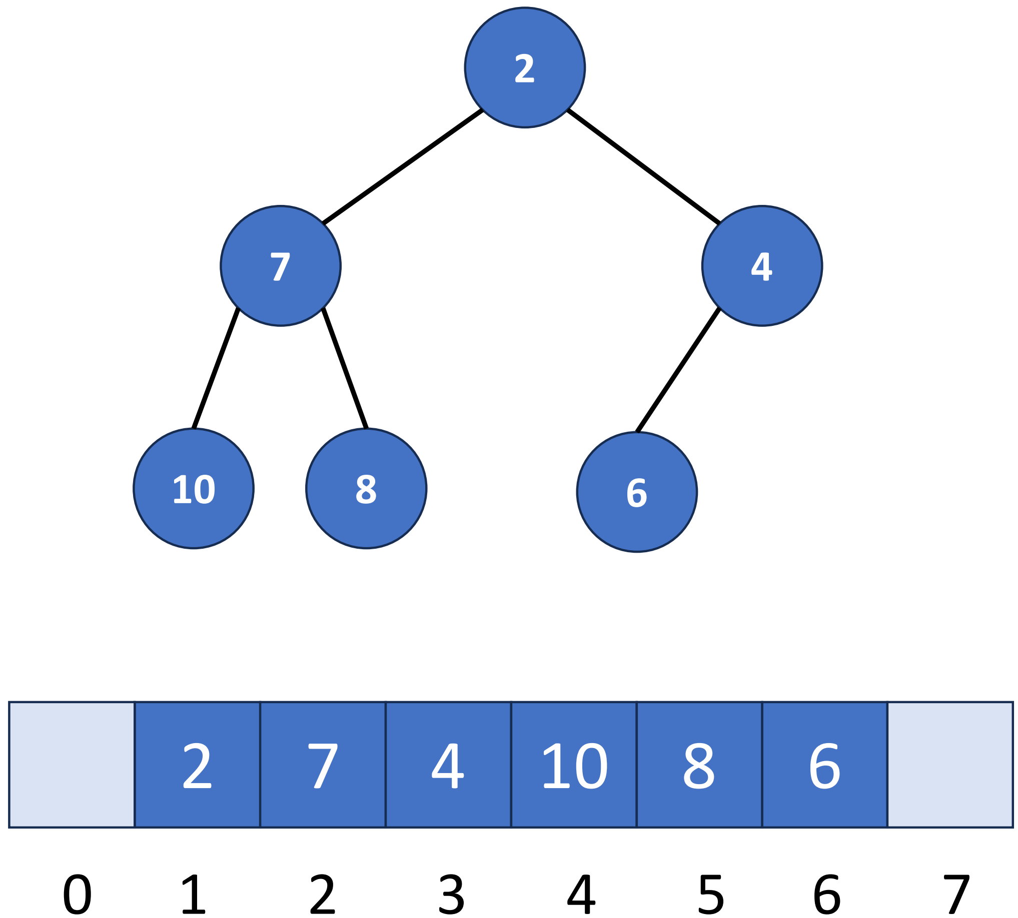


Max Heap

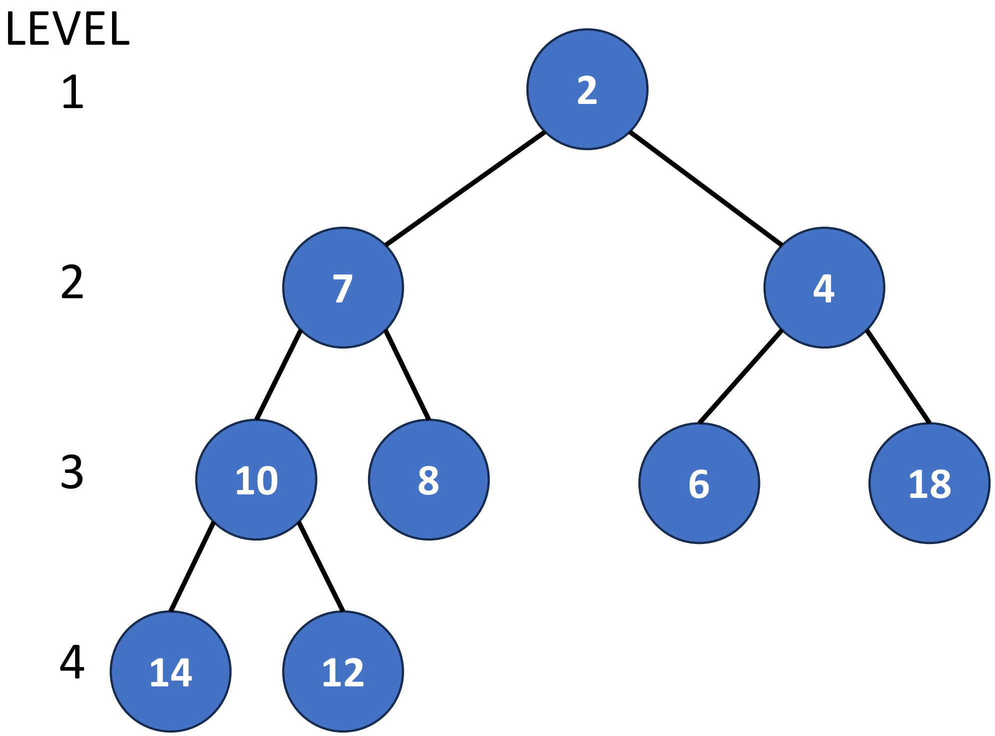
- A complete binary tree
- A max tree.



A heap is efficiently represented as an array



Heap height

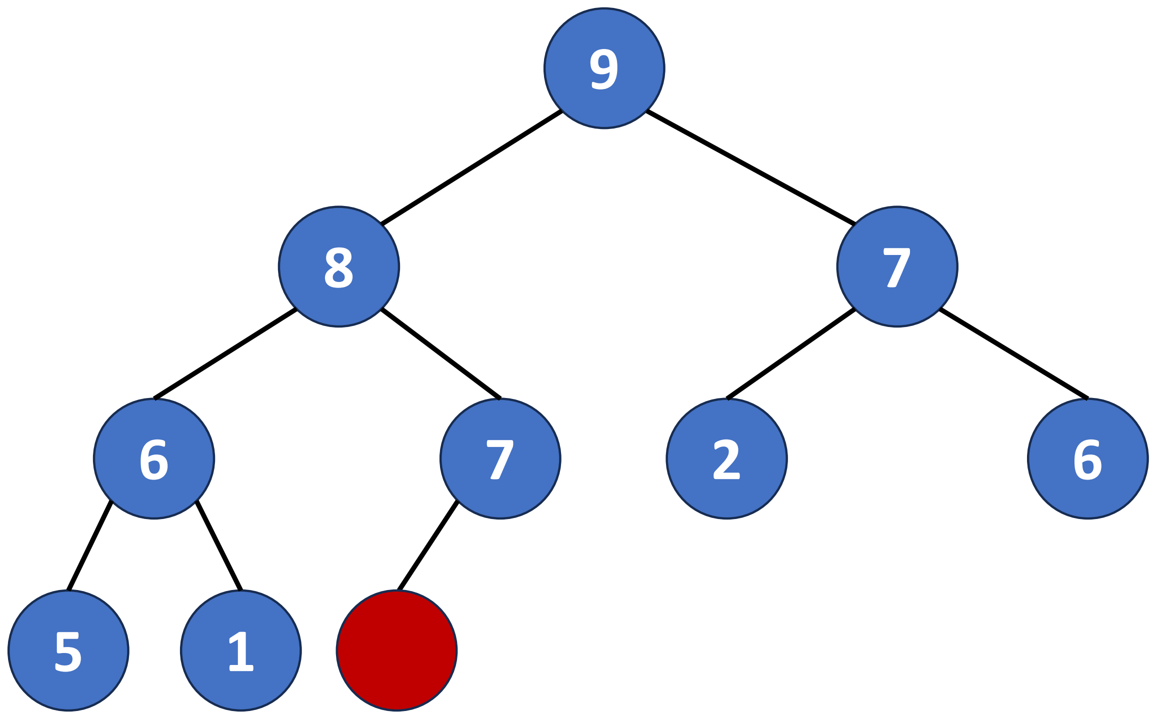


9 nodes, height = 4

- A heap is a complete binary tree.
- The height of an n-node heap is

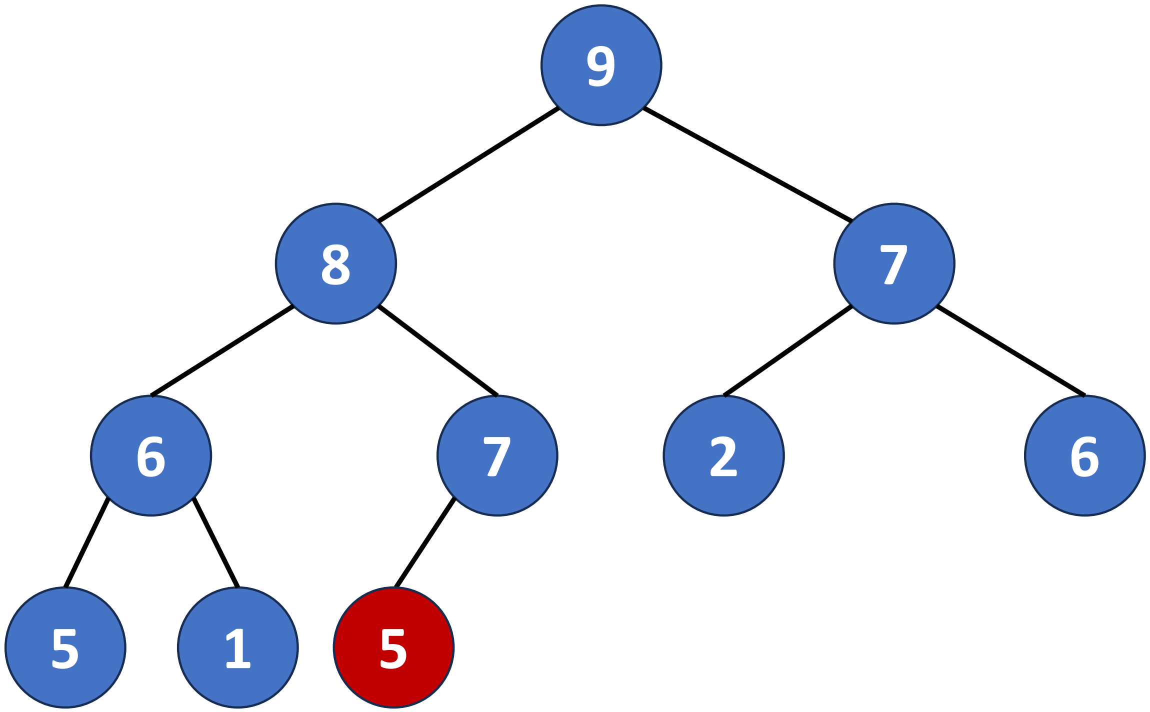
  $O(\log_2 n)$ or $O(\log n)$.

Operation: Insert(*key*) in a max heap



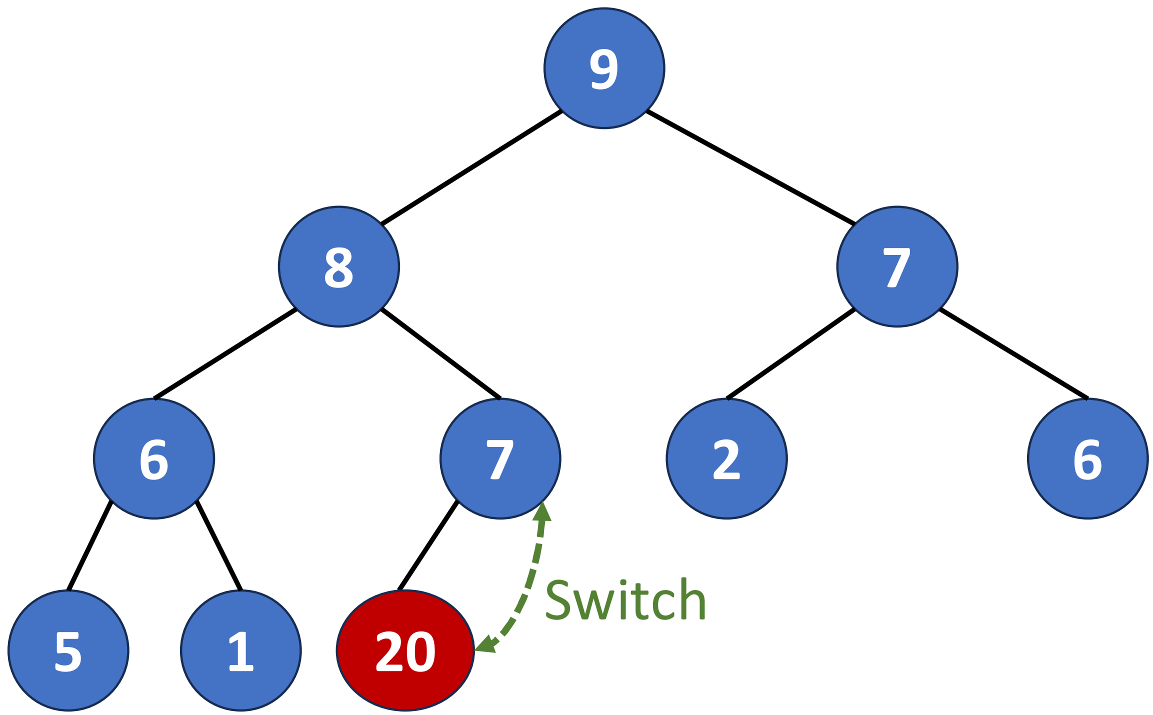
Become a complete binary tree with 10 nodes.

Operation: Insert(*key*) in a max heap



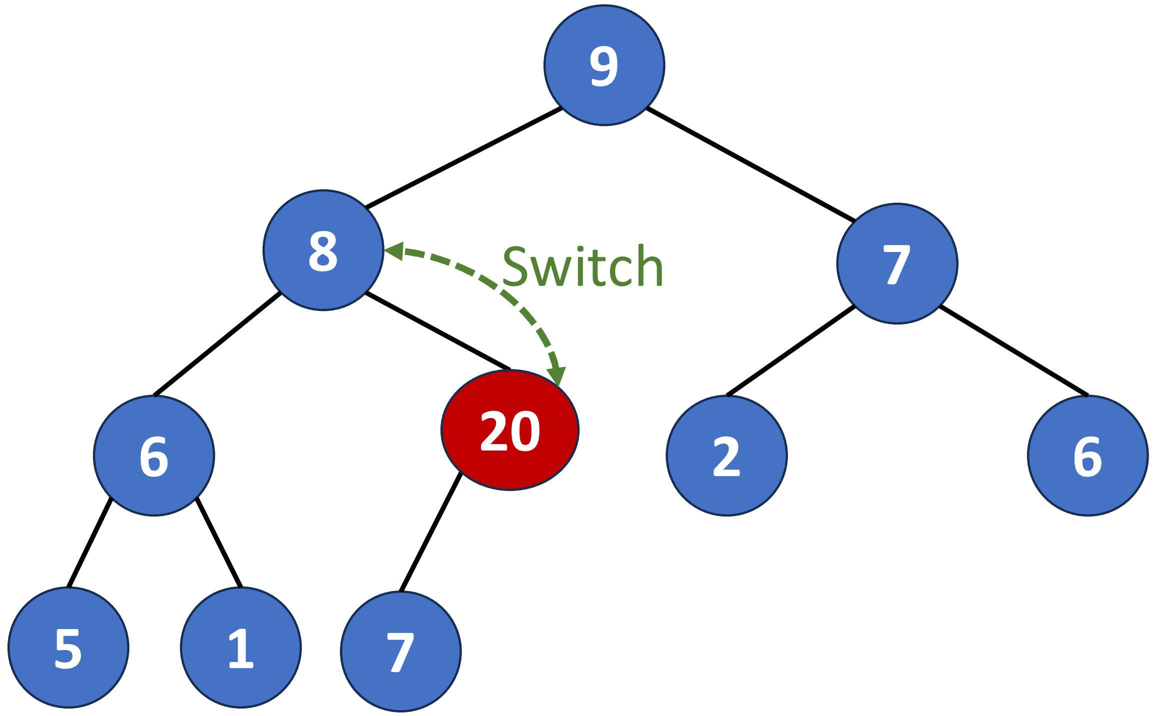
The new element containing key 5.
It can be the left child of node 7.

Operation: Insert(*key*) in a max heap



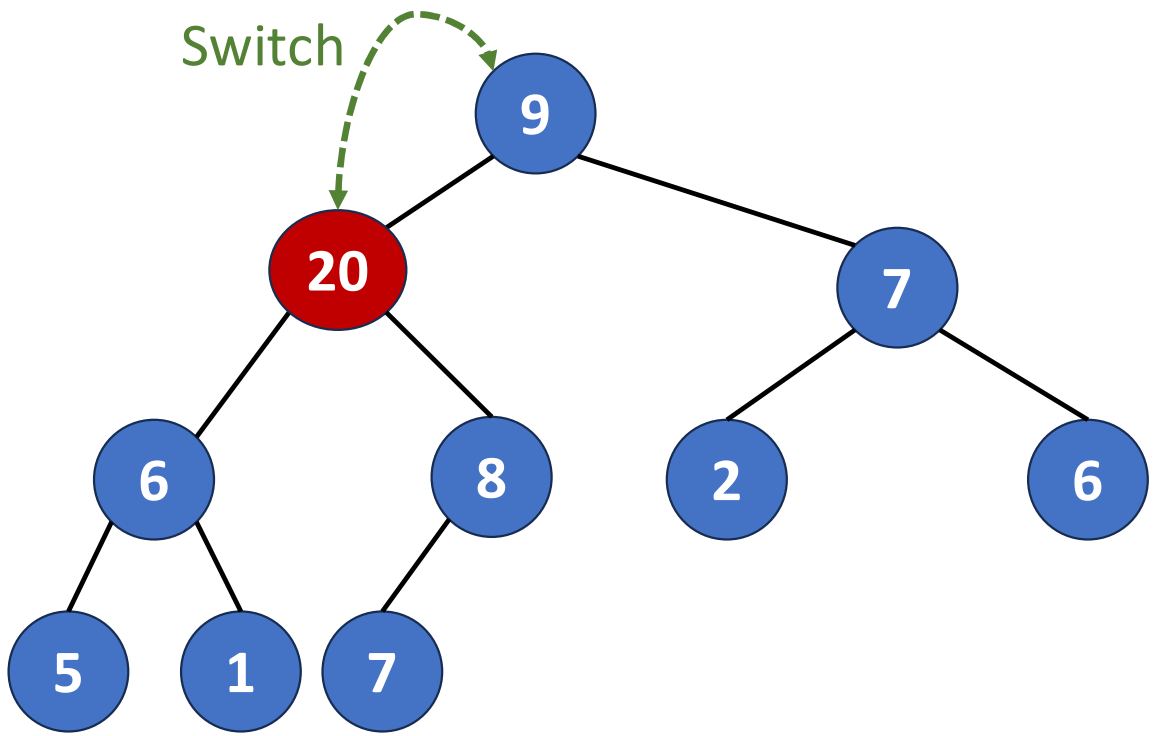
The new element containing key 20.
It cannot be the left child of node 7. ➔ Switch 7 and 20

Operation: Insert(*key*) in a max heap



The new element containing key 20.
It cannot be the right child of node 8.➔ Switch 8 and 20

Operation: Insert(*key*) in a max heap

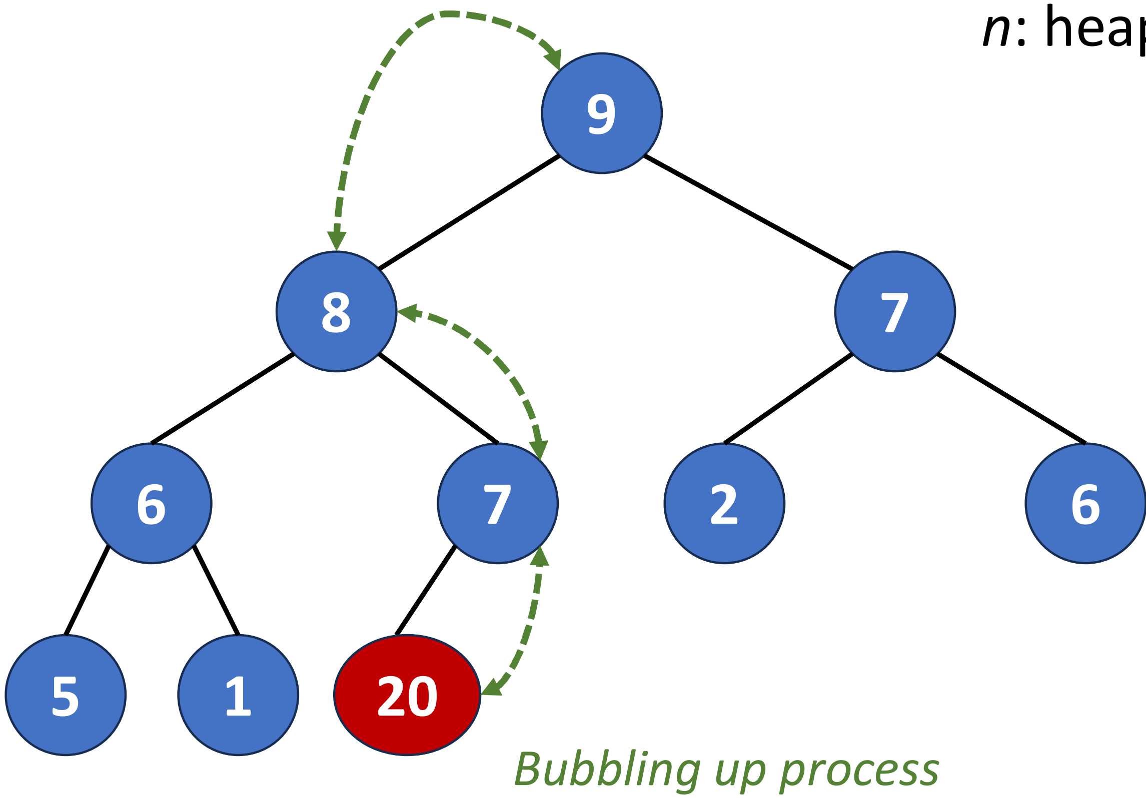


The new element containing key 20.
It cannot be the left child of node 9. ➔ Switch 9 and 20

Time complexity of insertion

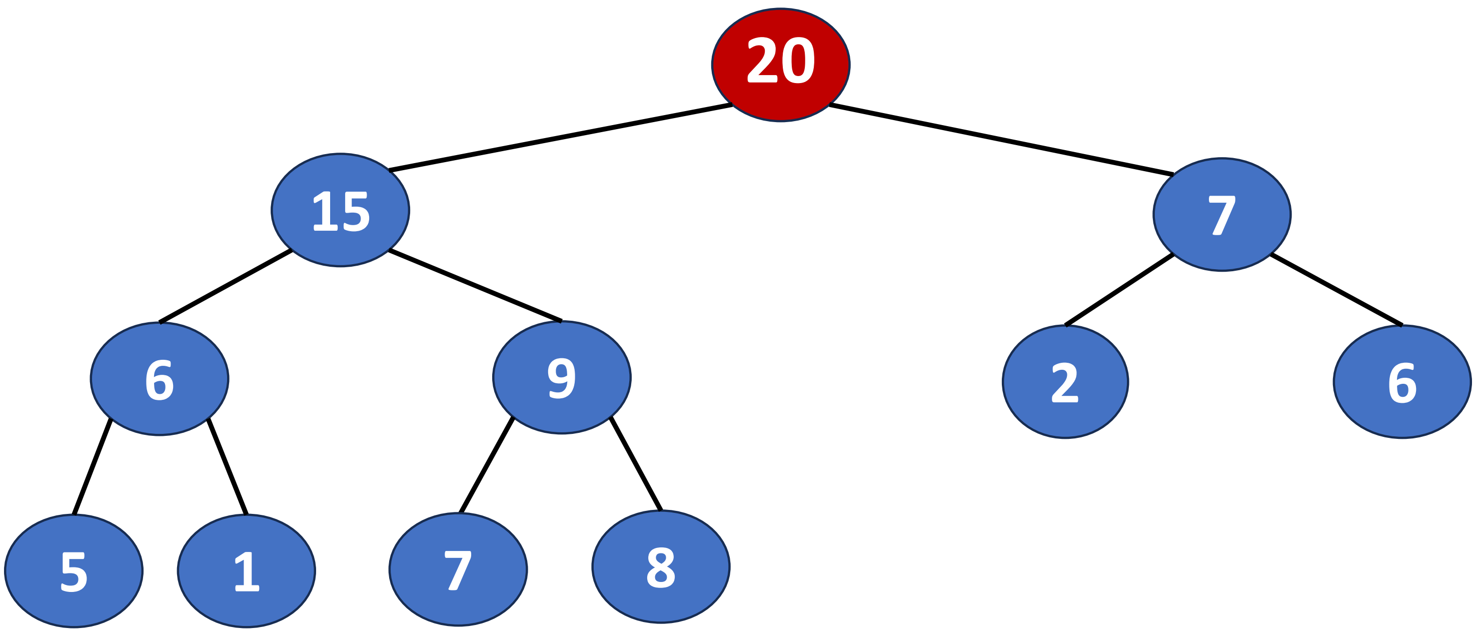
- Depending on the height of heap. ➔ O(log *n*)

n: heap size



Operation: Delete() or Pop()

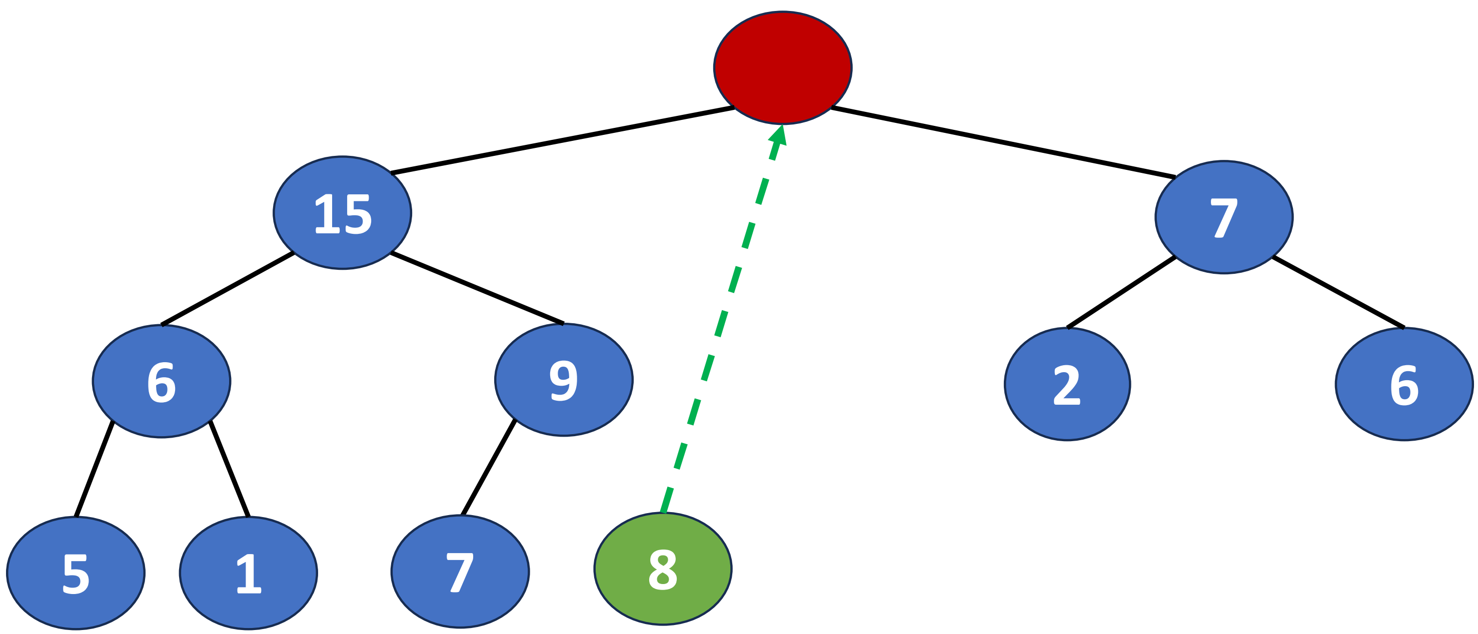
- Removing the root of the heap.
  - Root is the min element in a min heap.
  - Root is the max element in a max heap.



20 is removed. Heap with 10 nodes.

Operation: Delete() or Pop()

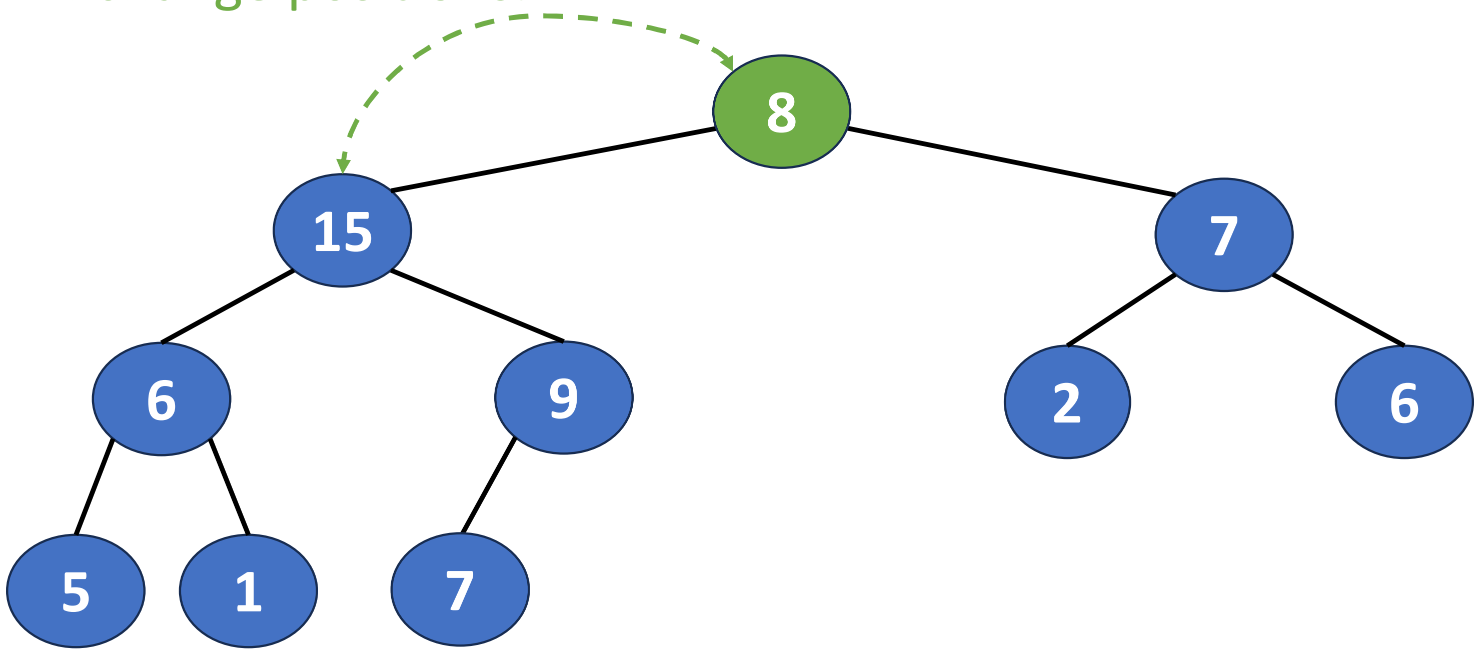
- Removing the last node and insert into the root.



Reinsert 8 into the heap.

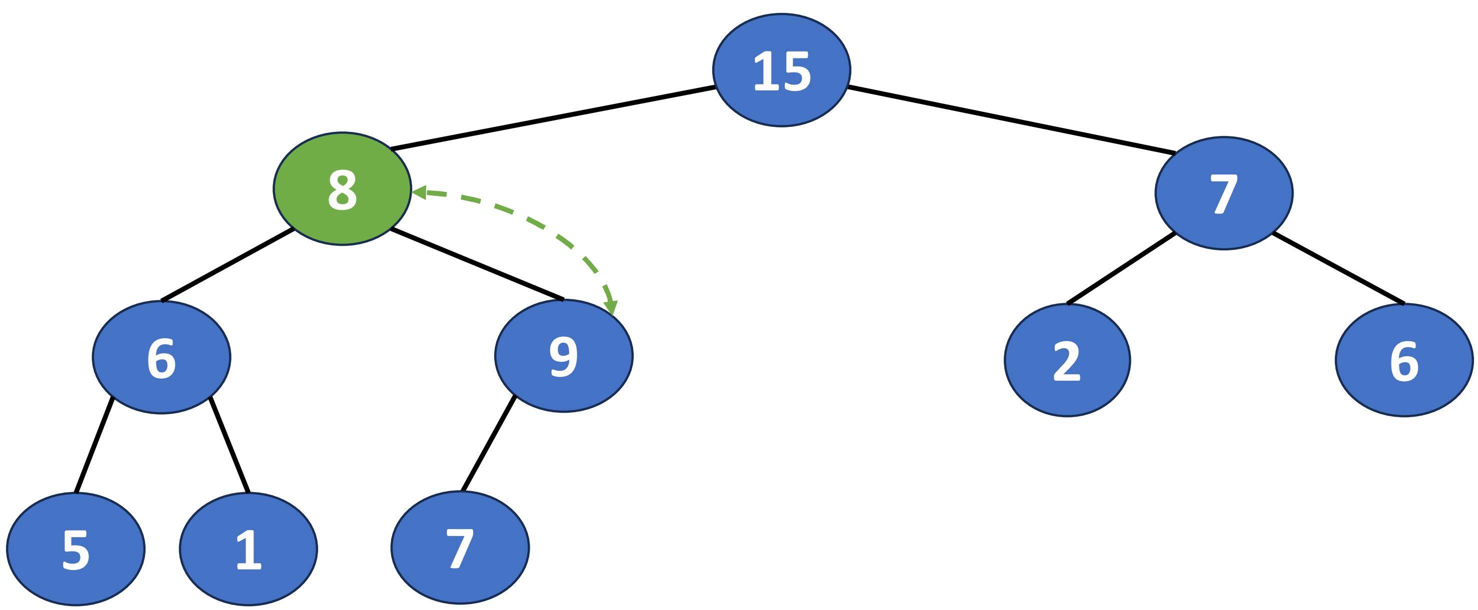
Operation: Delete() or Pop()

- Moving the node to a proper position.
  - Find the child containing max key value.
  - Exchange positions.



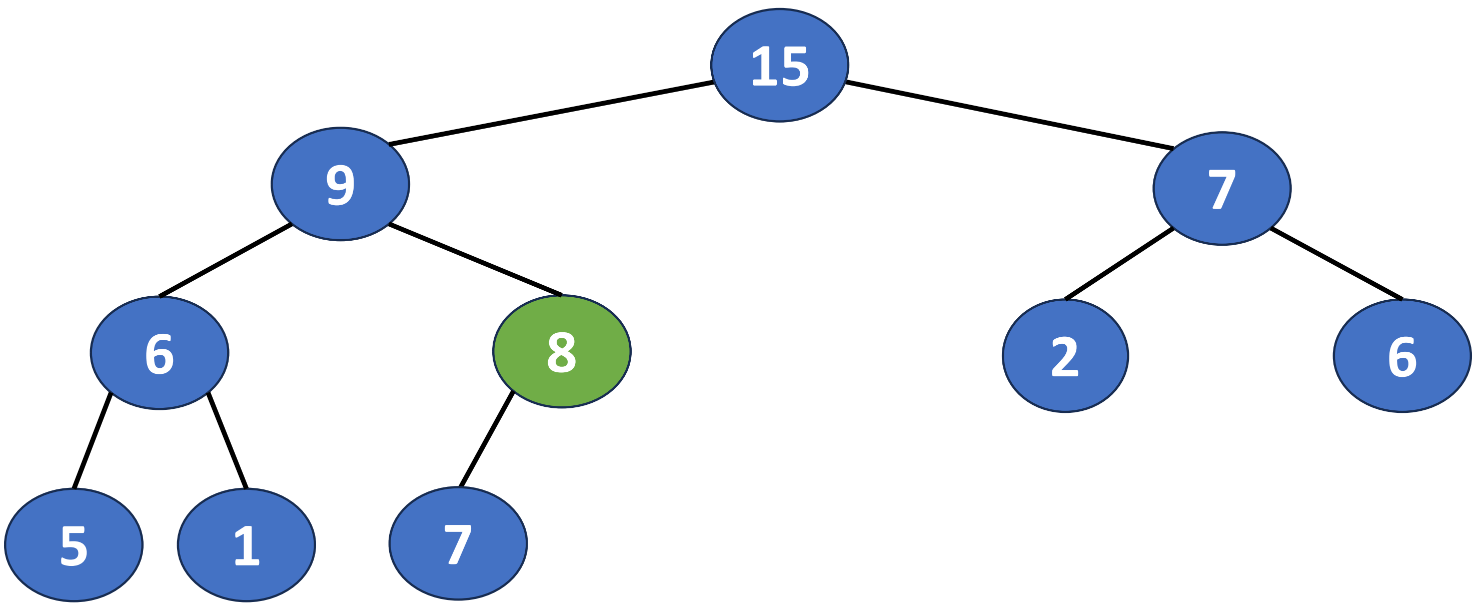
Operation: Delete() or Pop()

- Moving the node to a proper position.
  - Find the child containing max key value.
  - Exchange positions.



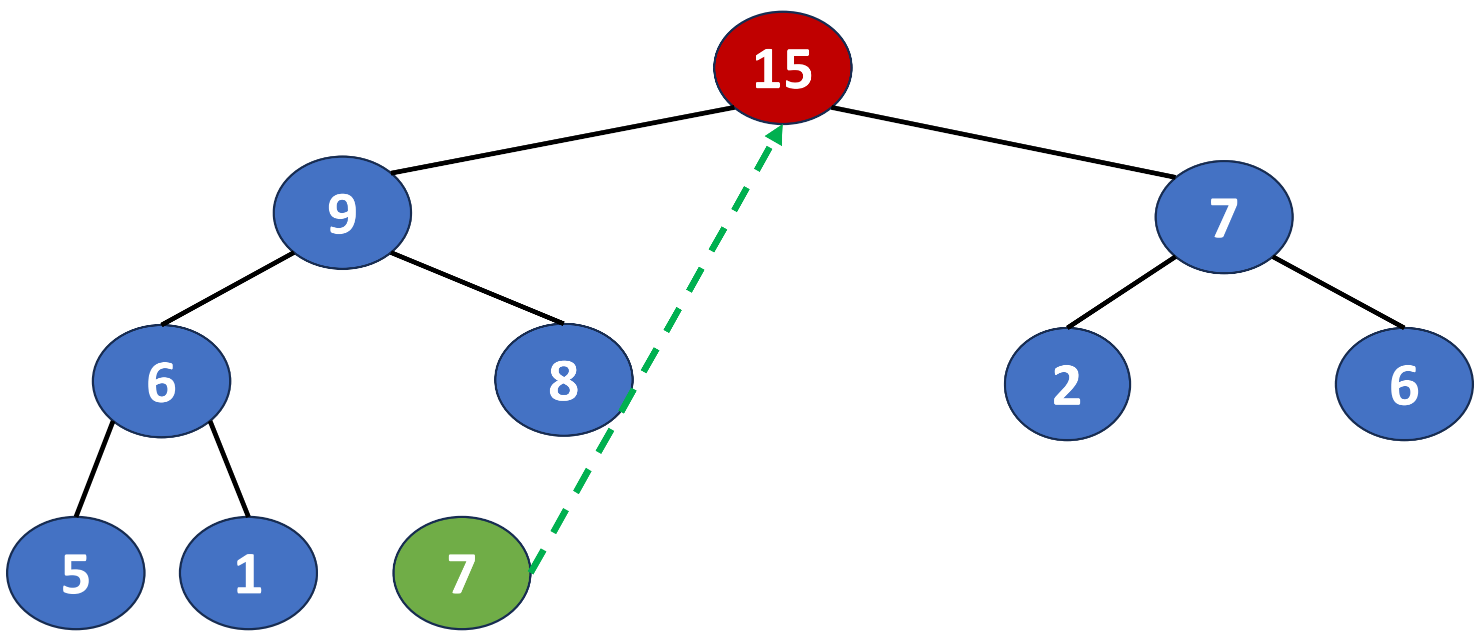
Operation: Delete() or Pop()

- Moving the node to a proper position.
  - Find the child containing max key value.
  - Exchange positions.



Operation: Delete() or Pop()

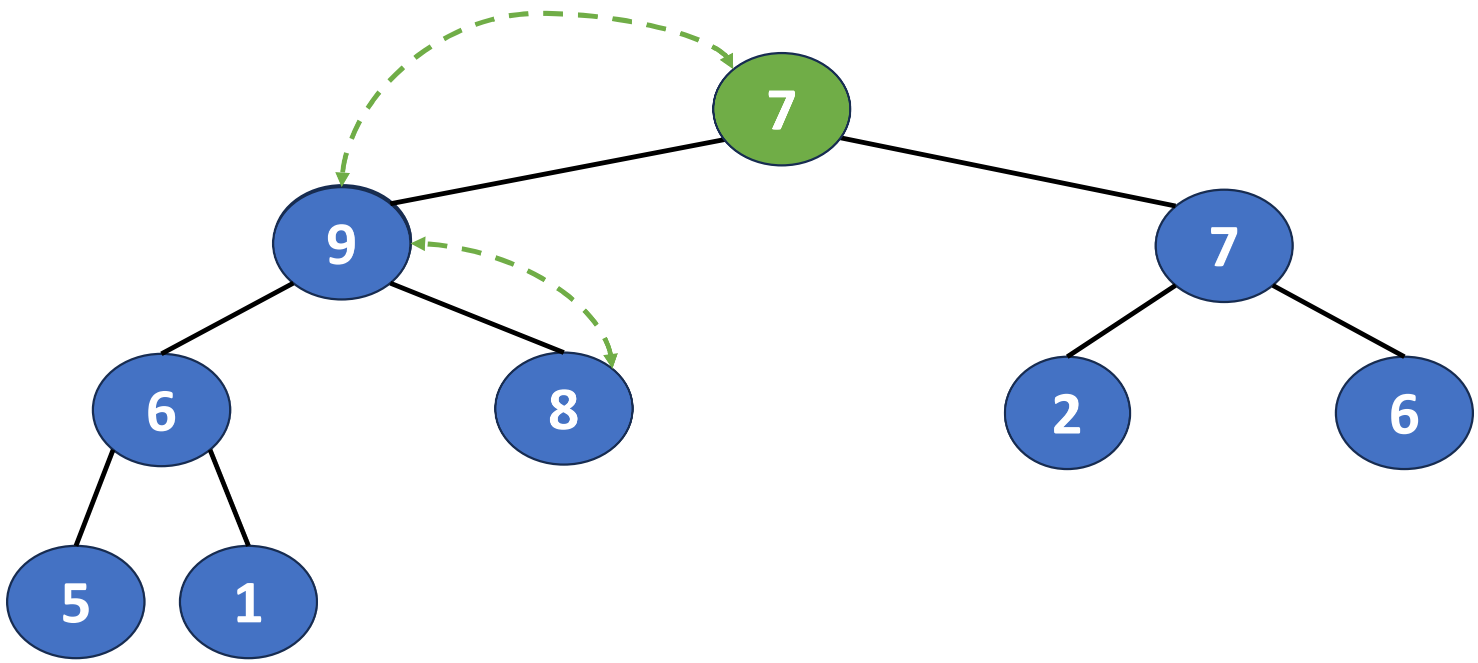
- Removing the root of the heap.
- Removing the last node and insert into the root.



Remove 15 and reinsert 7 into the heap.

Operation: Delete() or Pop()

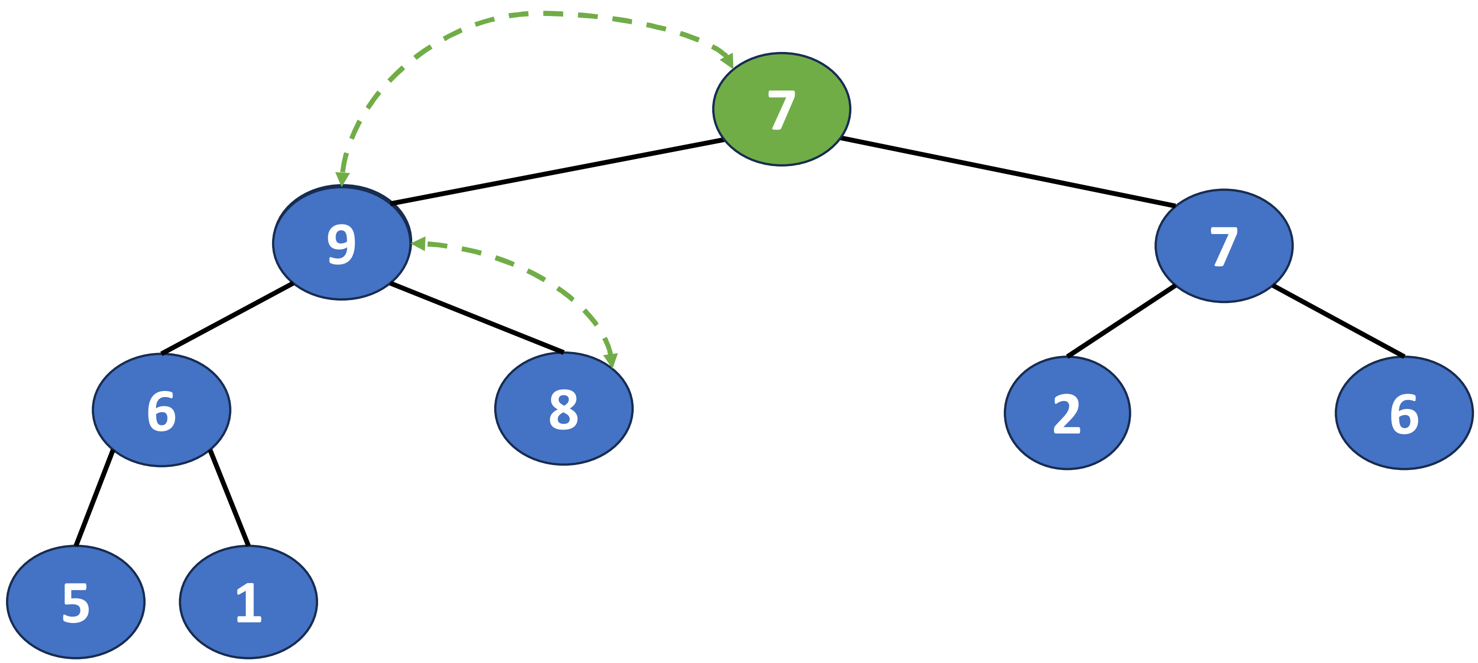
- Moving the node to a proper position.



Move the 7 to position 2 and then to position 5.

Time complexity of deletion

- Depending on the height of heap. ➔ O(log *n*)



Exercise

- Given the following key values: 7, 16, 49, 82, 5, 31, 6, 2, 44
  - Q9: Write out the **max** heap after inserting all elements.
  - Q10: Write out the **min** heap after inserting all elements.

Note: Please use array representation for the max heap.

Please reply your answers of Q9 and Q10 via the following link:

Group members: 1~3 people

Summary

- Priority queue
- Min tree and max tree
- Min heap and max heap
  - Operations: Insertion and Deletion