Concurrent Access of Priority Queues

Andrew Milner

February 7 2011

Overview

- Review of terms
- Issues with concurrent access
- Paper/Solutions
- Future work and discussion

What is a Priority Queue?

- A data structure for maintaining a set S of nodes, each with an associated value called a key. (Cormen et al)

- Priority queue operations include:
  - Enqueue: Add an node to the queue
  - Dequeue: Remove the top node from the queue
  - SetPriority: Set the priority of an node
  - Top: Look at the top node of the queue

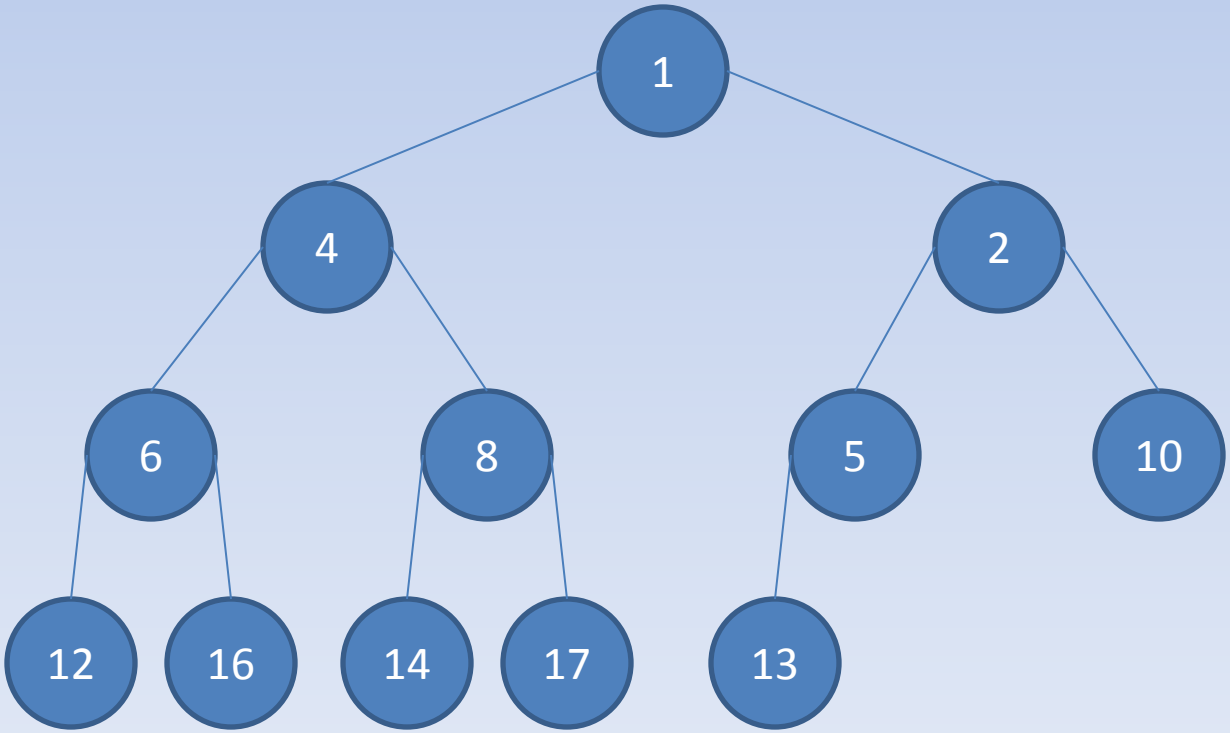
What is a Binary Heap?

- A heap is a complete binary tree of depth *d,* with the property that the value of the key at any node is less than the value of the keys at its children (if they exist). (Nageshwara and Kumar)

- The heap is an important data structure used as a priority queue in a wide variety of parallel algorithms.

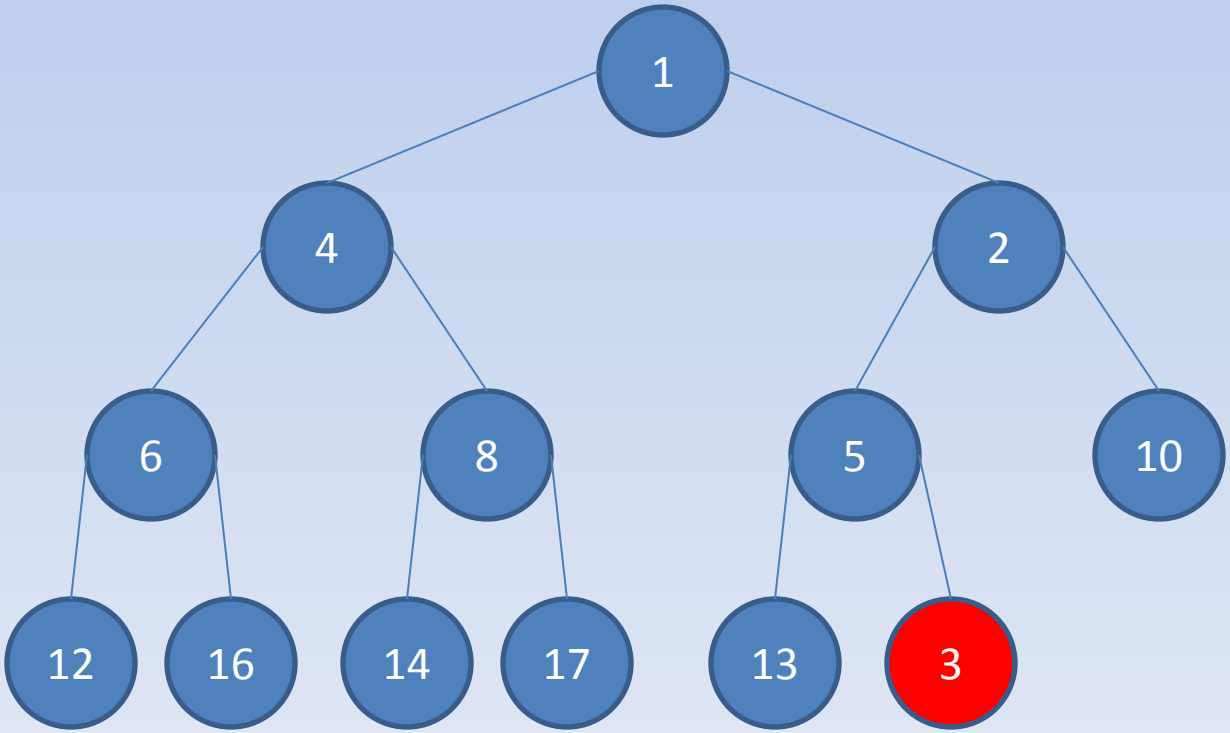
Inserting into a Heap

- Insertions into a binary heap are preformed from the bottom up.



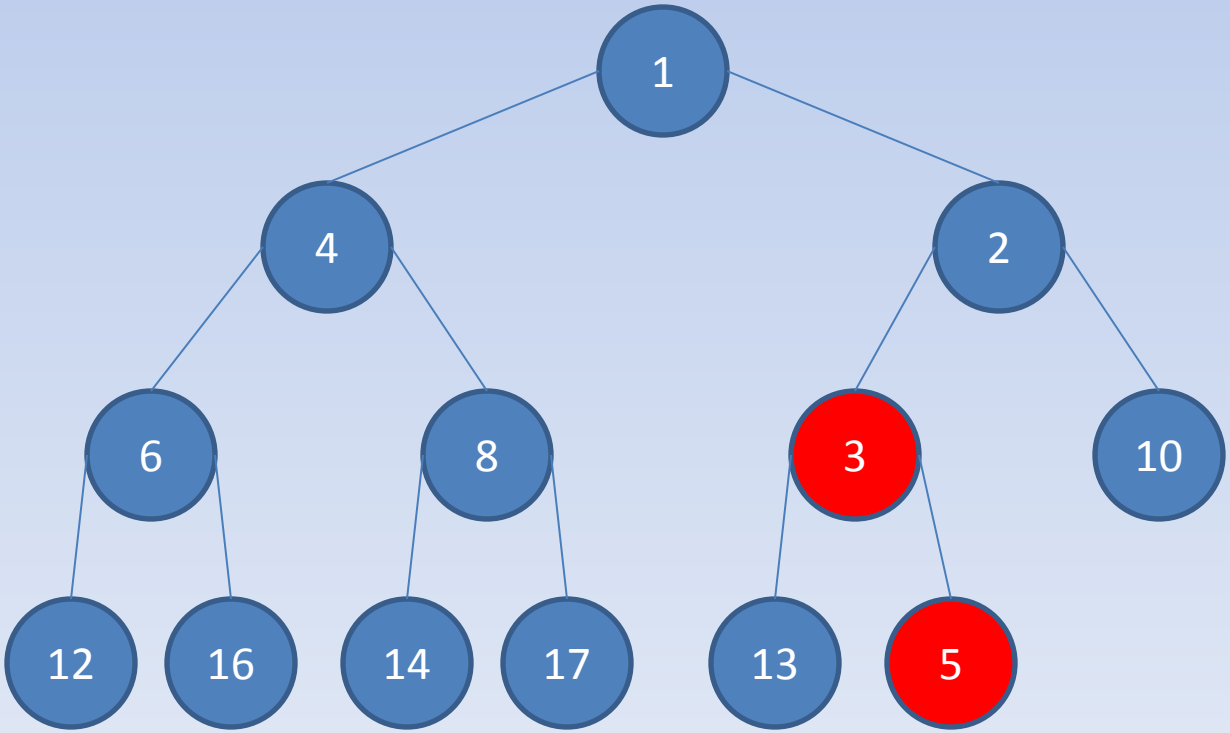
Inserting into a Heap

- The newest node is inserted at the next available node in the heap.



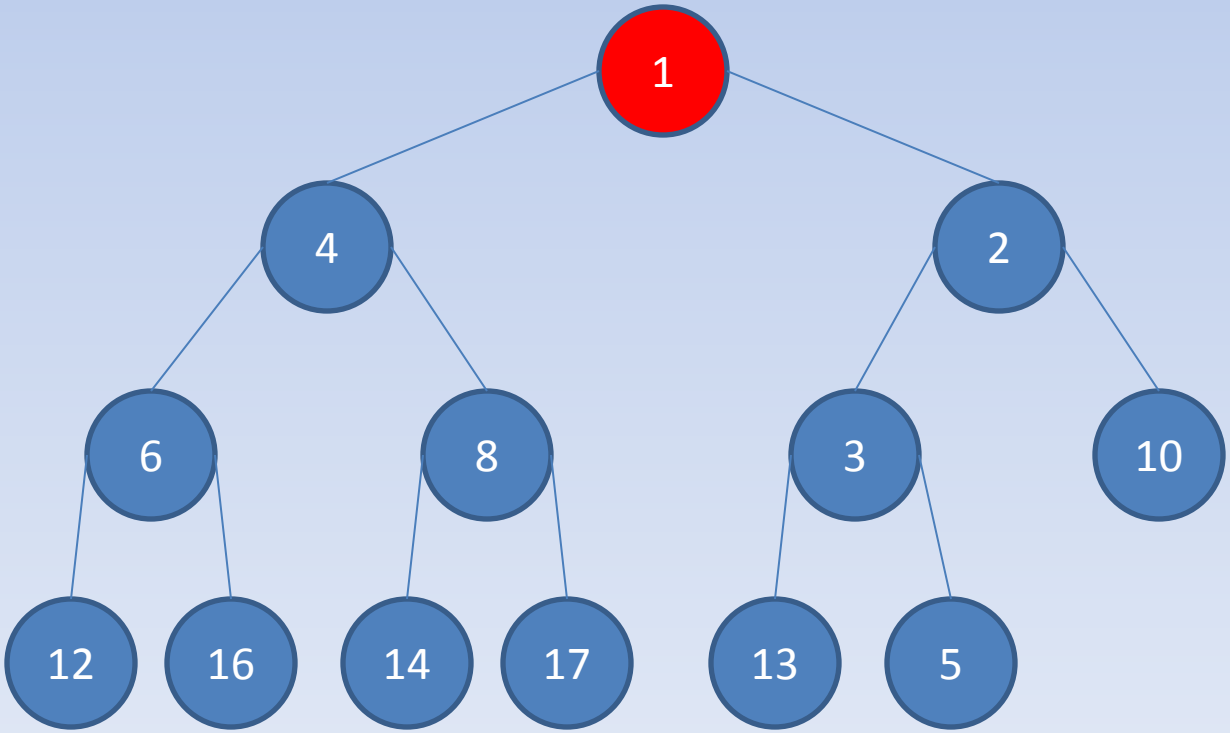
Inserting into a Heap

- If the newly inserted node does not satisfy the heap property, it is moved up through the heap until it does.



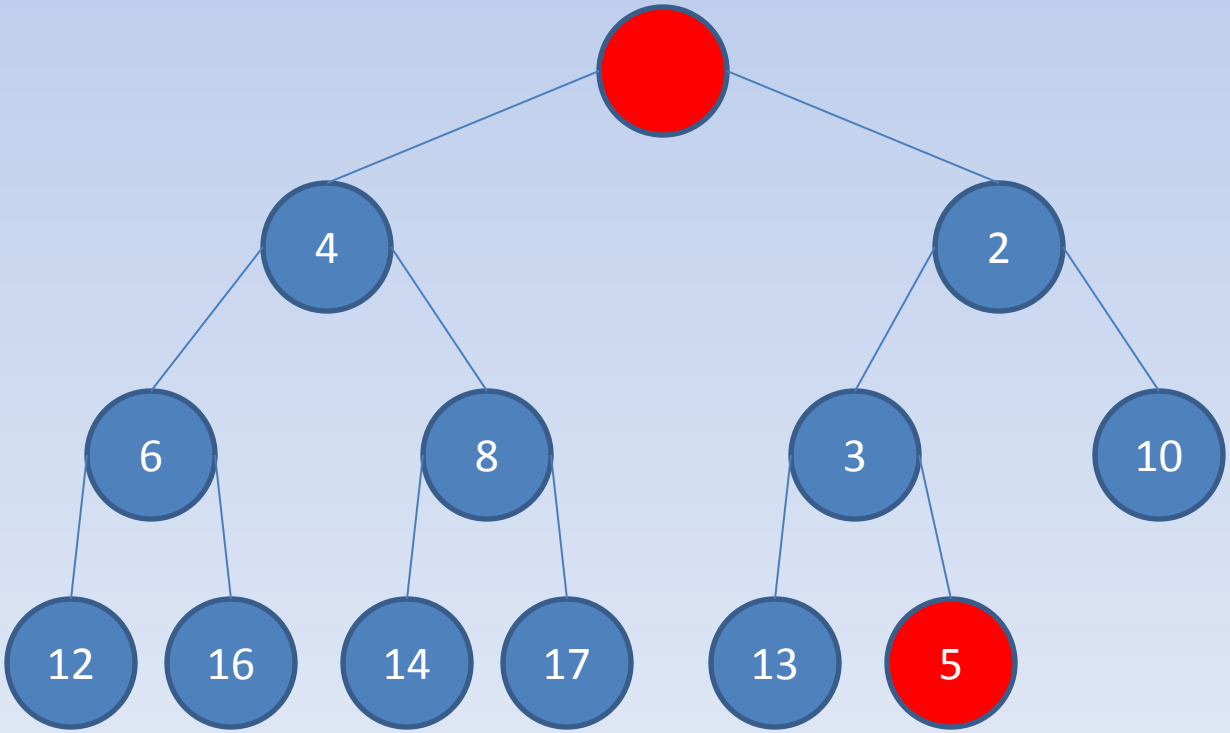
Deleting from a Heap

- Deletions from a heap occur from the top down.



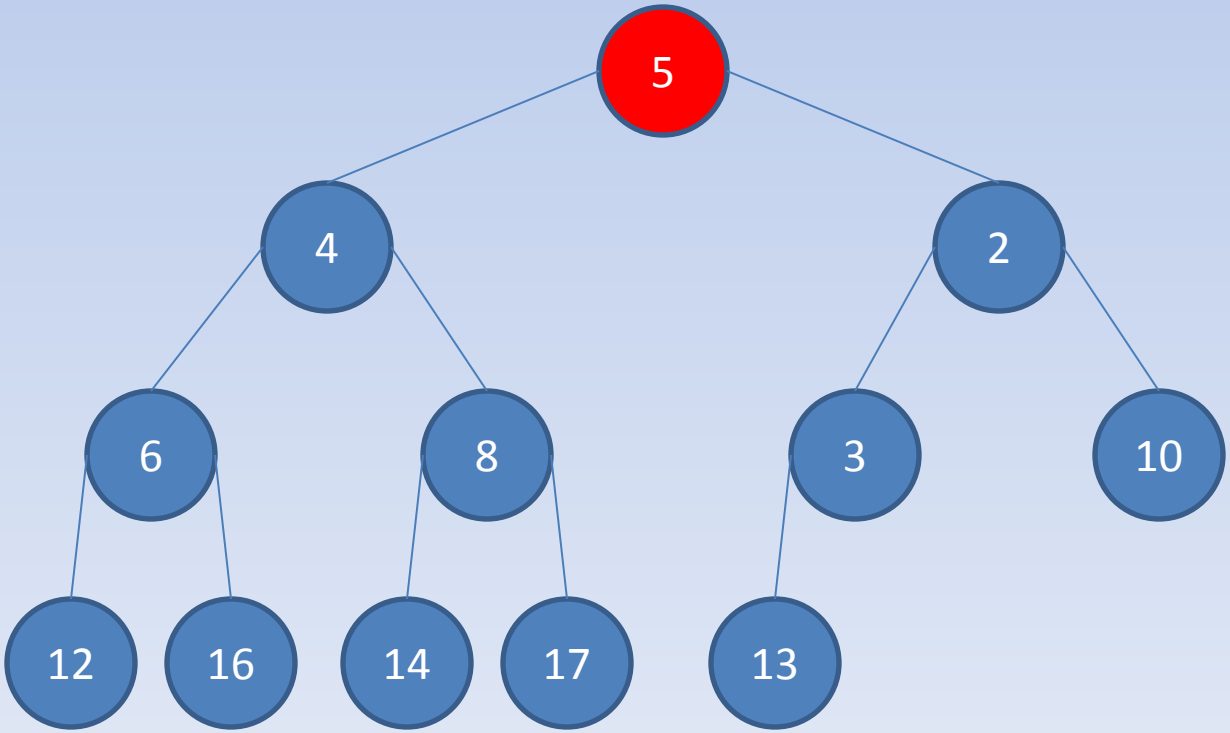
Deleting from a Heap

- The top node is removed from the heap and the most recently inserted node is moved to the top.



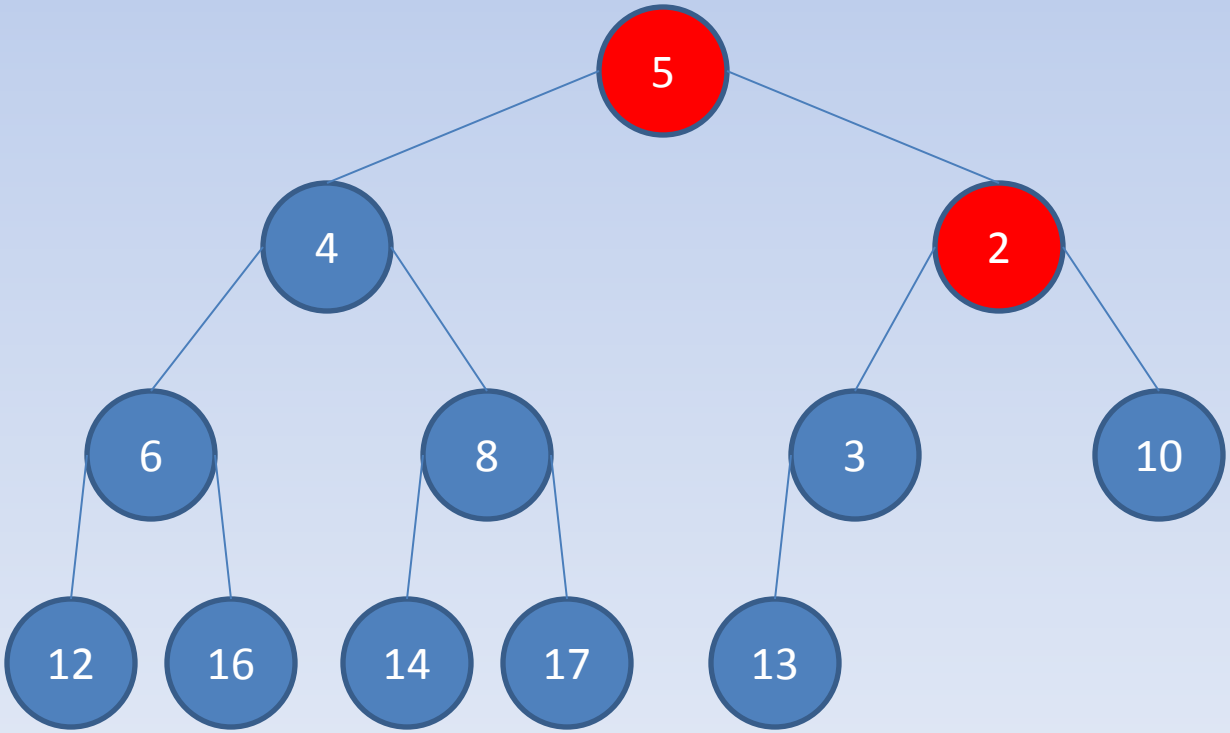
Deleting from a Heap

- The top node is removed from the heap and the most recently inserted node is moved to the top.



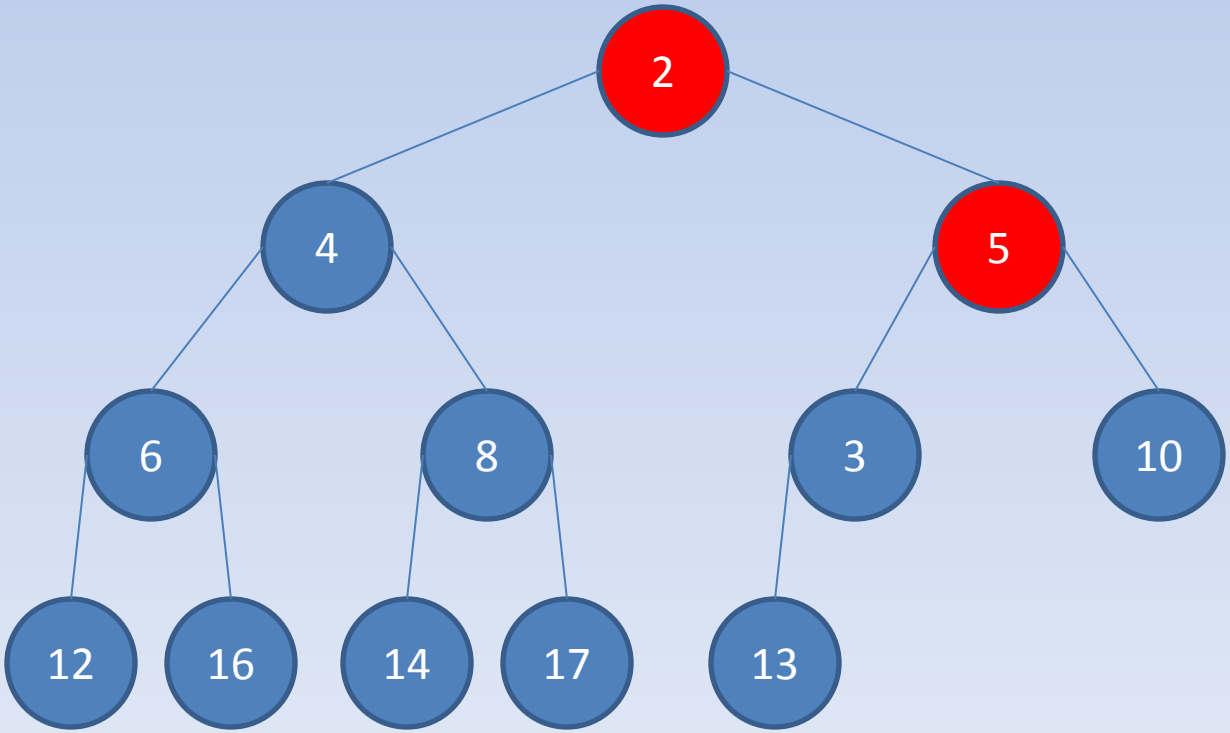
Deleting from a Heap

- If the new top node satisfies the heap property then the deletion call has finished. Otherwise the node is moved down through the heap to maintain the heap property.



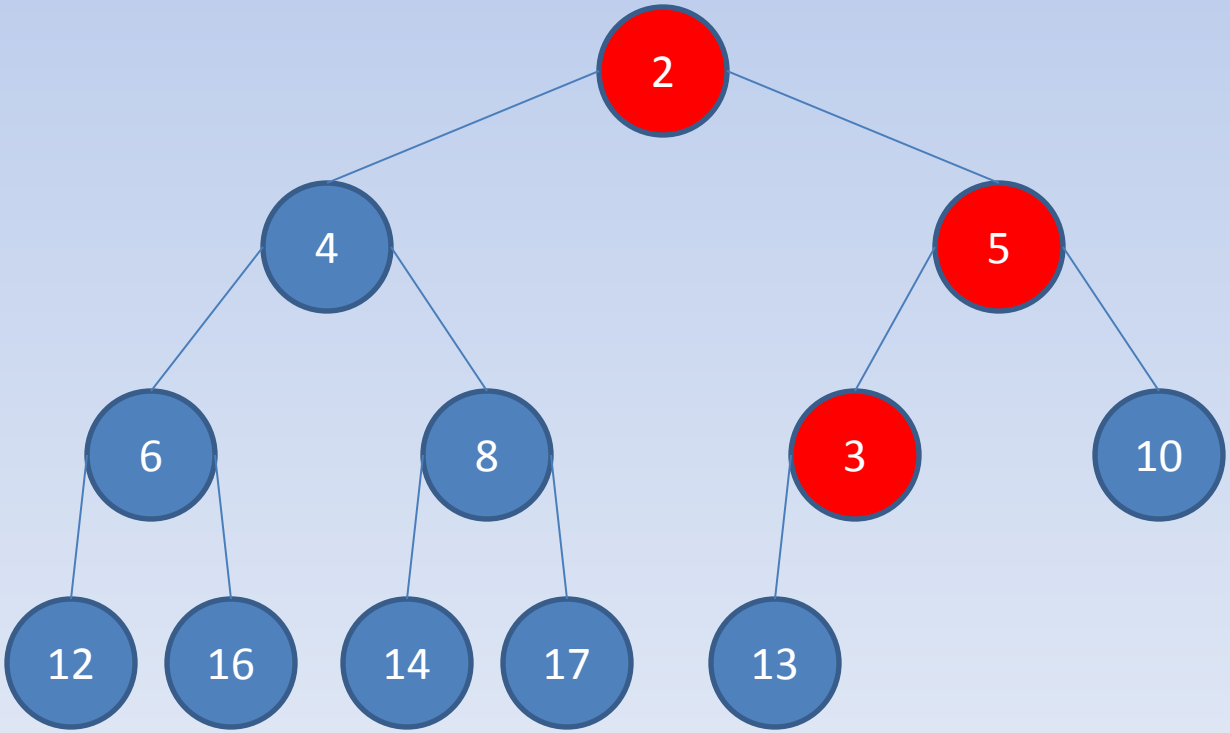
Deleting from a Heap

- If the new top node satisfies the heap property then the deletion call has finished. Otherwise the node is moved down through the heap to maintain the heap property.



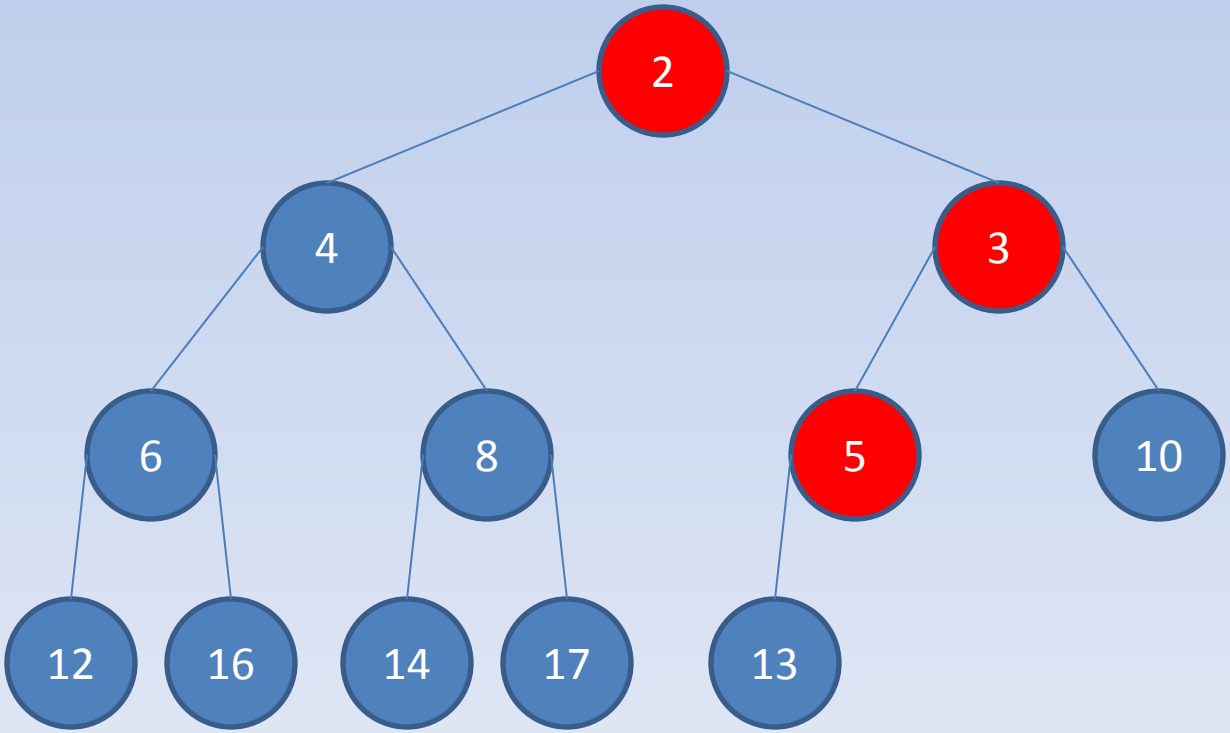
Deleting from a Heap

- If the new top node satisfies the heap property then the deletion call has finished. Otherwise the node is moved down through the heap to maintain the heap property.



Deleting from a Heap

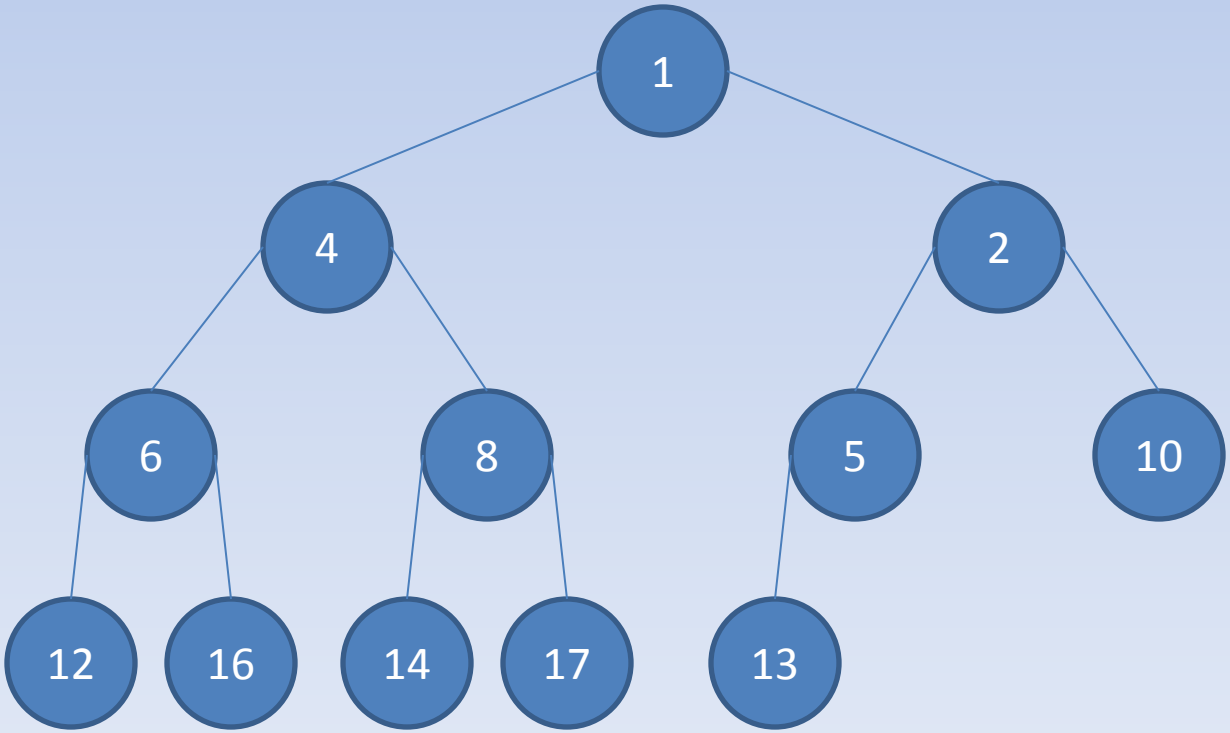
- If the new top node satisfies the heap property then the deletion call has finished. Otherwise the node is moved down through the heap to maintain the heap property.



Inserting and Deleting

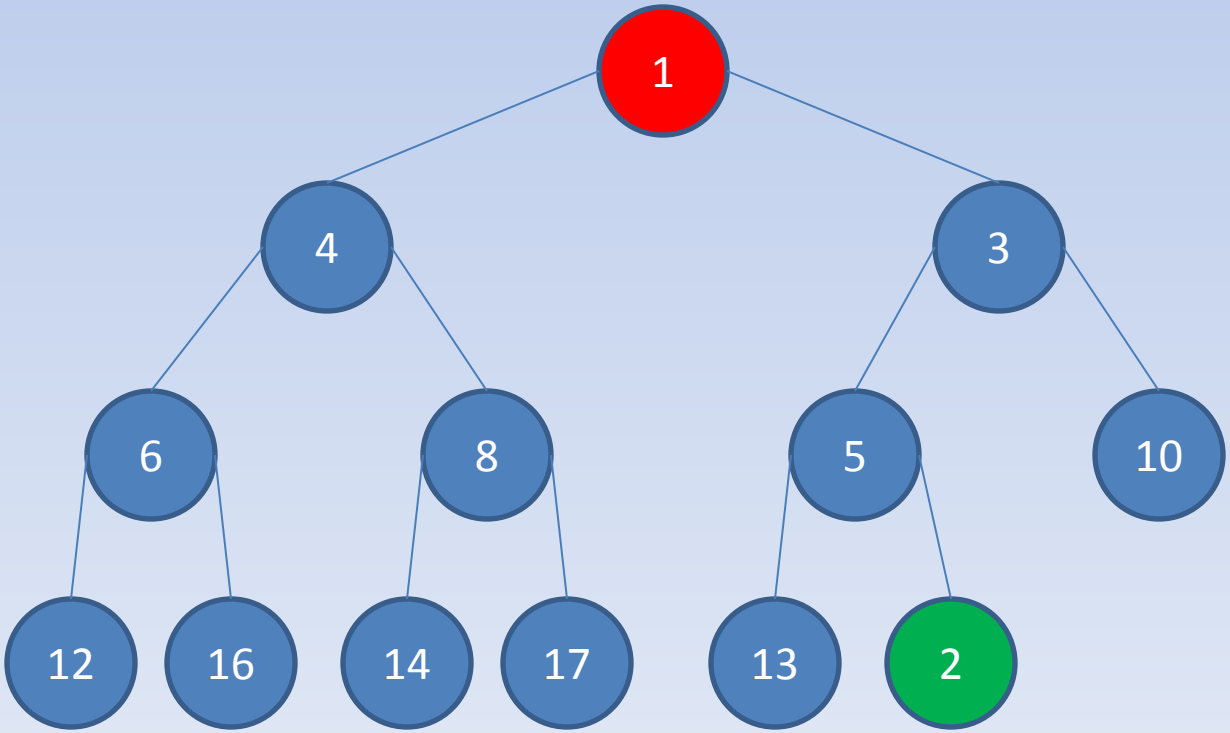
- Processes access the heap through mutual exclusion.
- Multiple insertions or deletions can be performed at the same time.
- An issue arises when attempting to perform an insertion and deletion at the same time.
- This is due to insertions and deletions operations being performed in opposite directions.

Inserting and Deleting



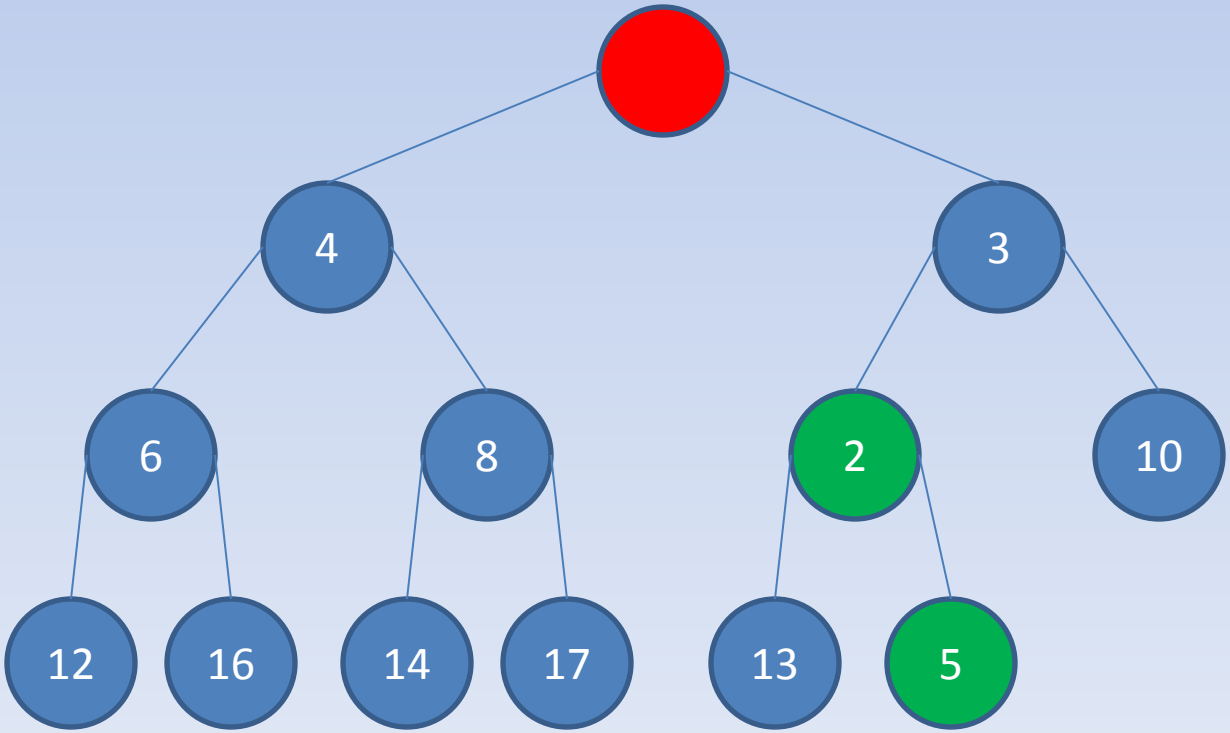
Inserting and Deleting

- The algorithm attempts to perform and insertion and deletion at the same time.



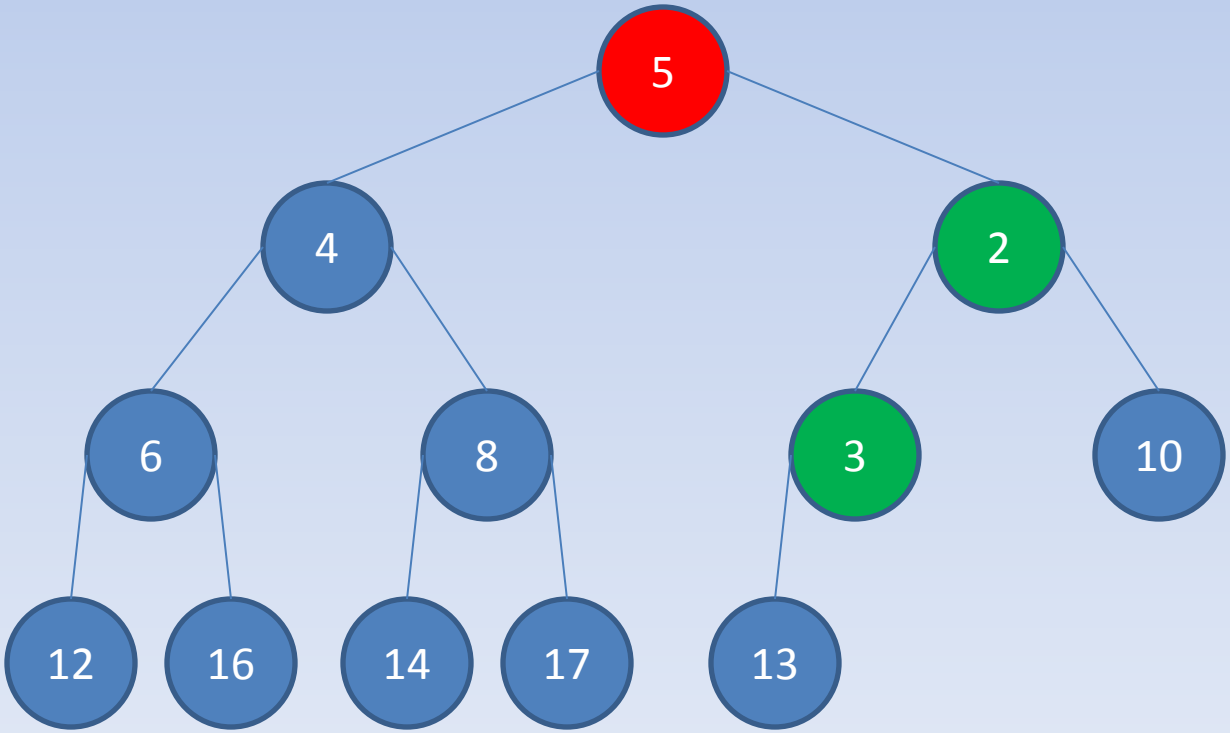
Inserting and Deleting

- The top node is deleted but cannot get the last node until it is unlocked.



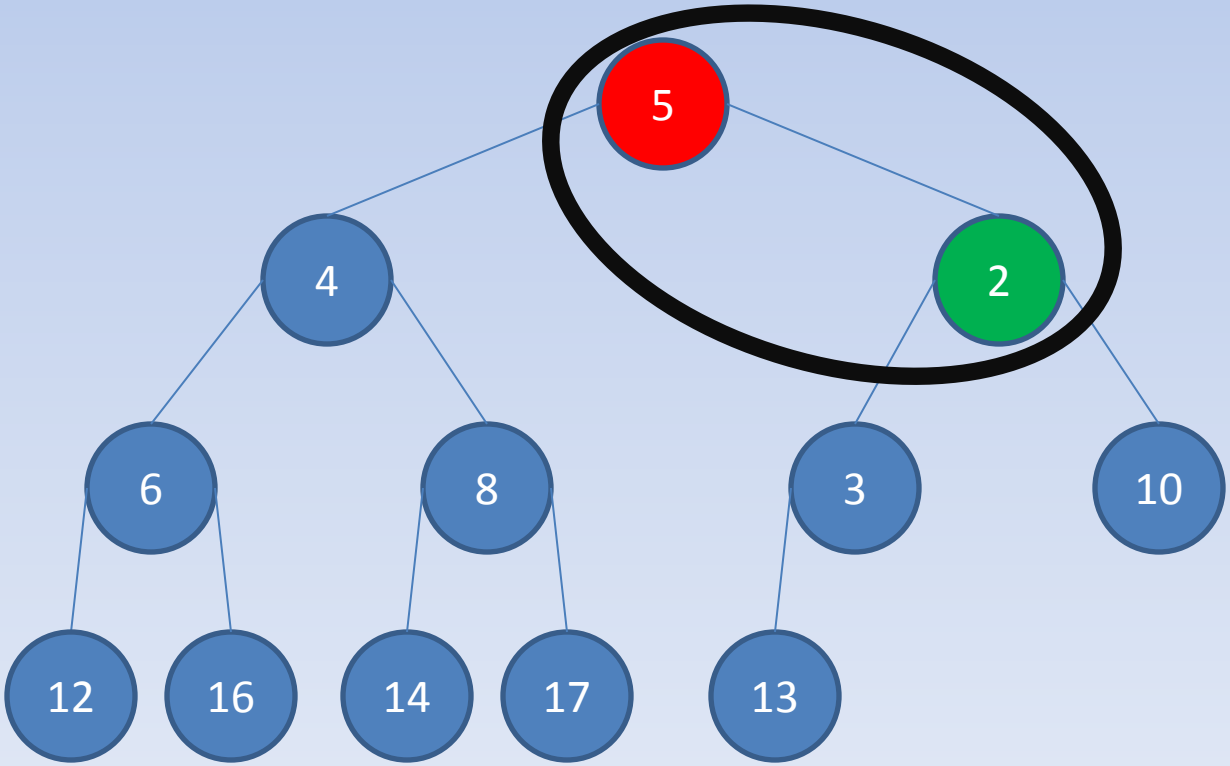
Inserting and Deleting

- The last node is unlocked and moved to the top. The newly inserted node continues to move up through the heap.



Inserting and Deleting

- The two nodes are now deadlocked! Both nodes are waiting for the other to unlock and the algorithm is now stuck.



My Paper

R.V. Nageshwara, V. Kumar, Concurrent Access of Priority Queues. *IEEE Transactions on Computers*, 37(12):1657-1665, Dec. 1988.

Solution

- Nageshwara and Kumar suggest insertions should be completed from the top down.

- Both procedures now perform actions from the top down and eliminate the potential for a deadlock to occur.

Top Down Insertions

- Top down insertions can be performed by predicating the final location of the node to be inserted.

- Two values are required in order to predicate the path the node takes through the heap.
  - LastElem: The node location of the last element + 1.
  - FullLevel: The node location of the first element at the deepest level of the heap.

- The different between the two values gives the path.
  - This is possible because the heap is a binary tree.

Top Down Insertions

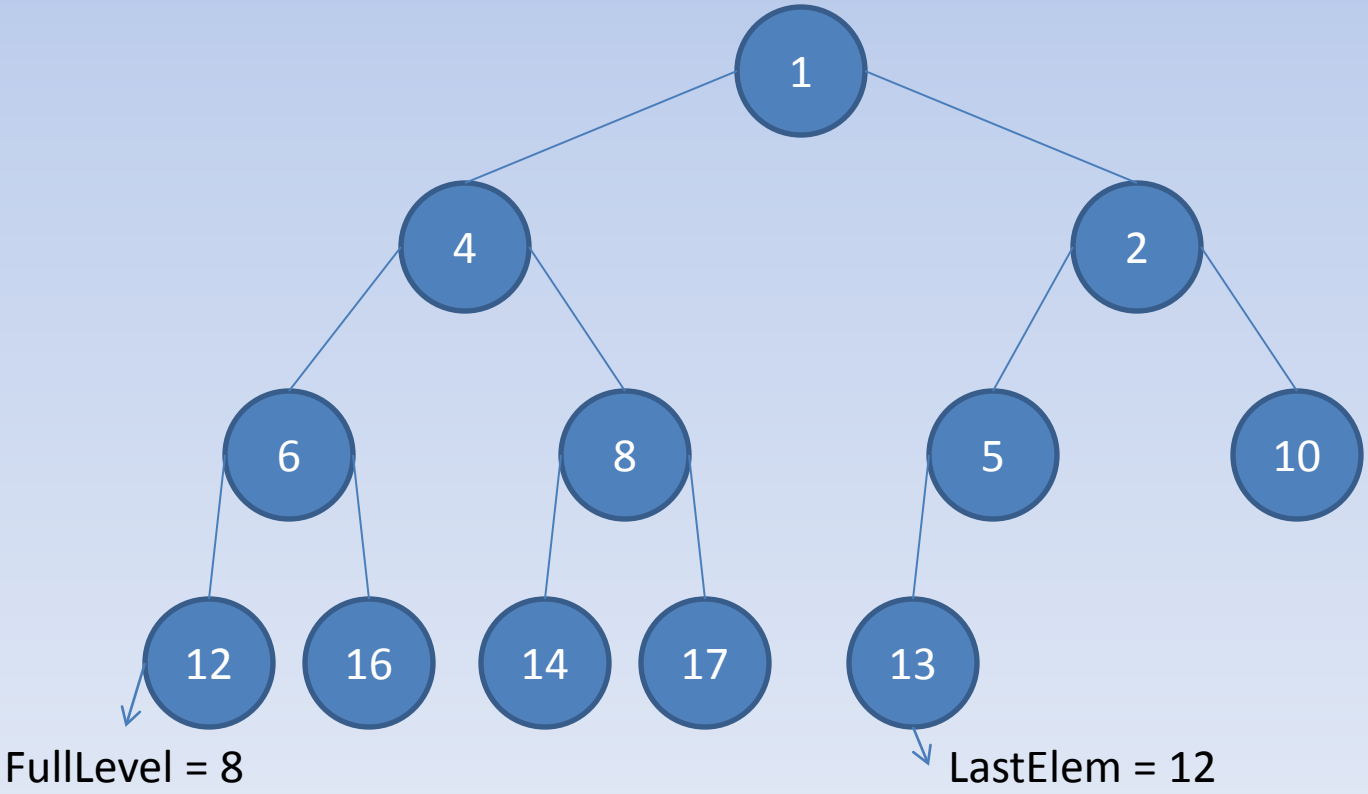
- The value obtained by finding the difference between LastElem and FullLevel is represented as a binary value.
- Each digit in the binary value tells the algorithm which direction to travel through the heap.
- 1 indicates a right movement and 0 indicates a left movement.

Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 5

NodeLoc = 101

The path of the node through the heap is right, left, right.

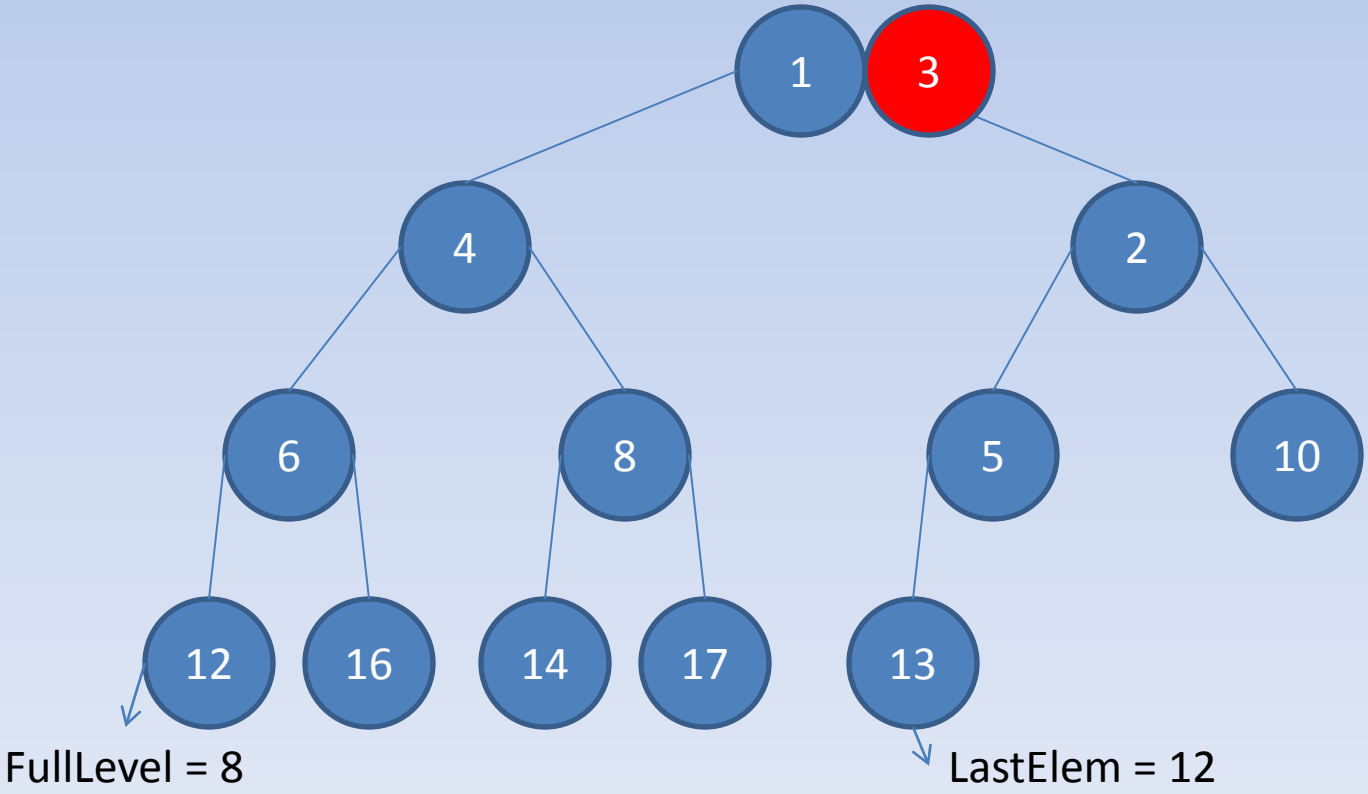


Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 3

NodeLoc = 101

The path of the node through the heap is right, left, right.

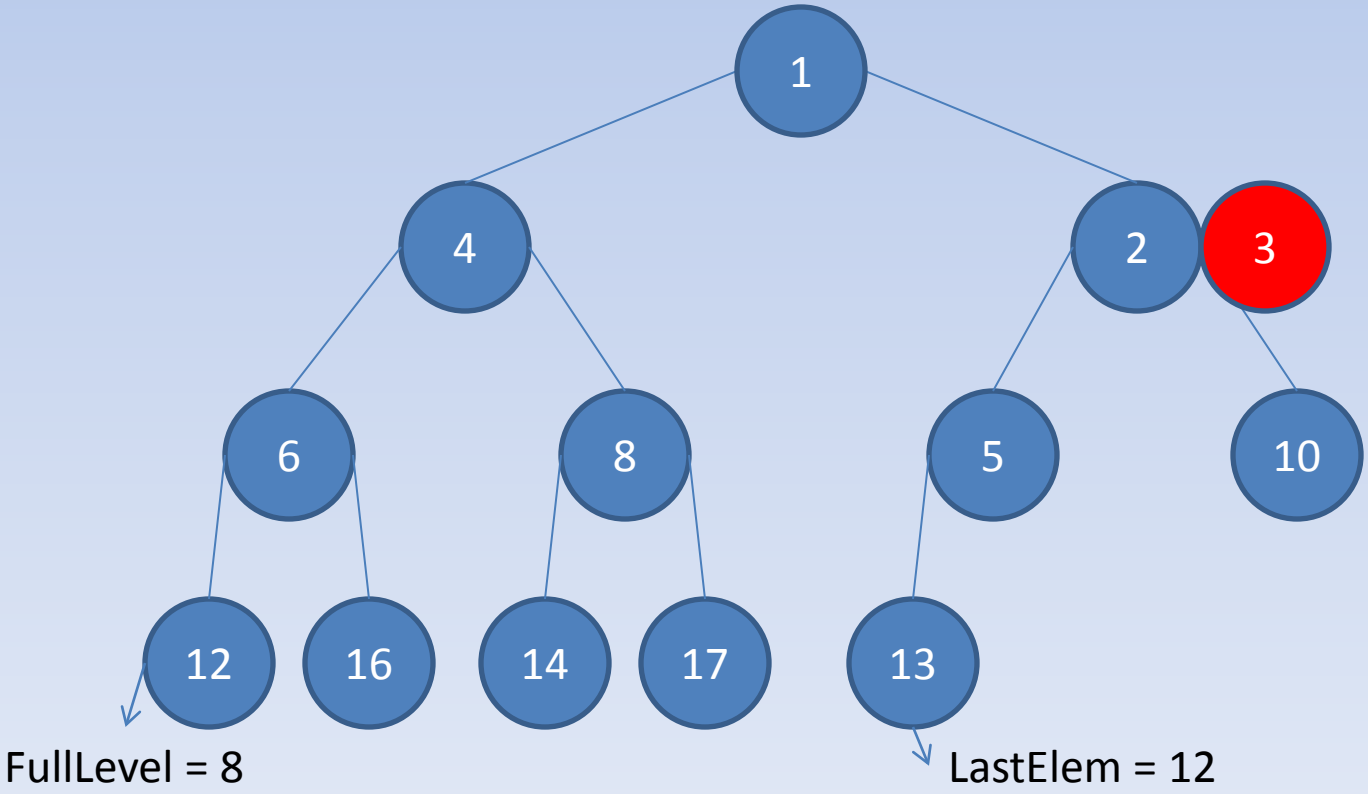


Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 3

NodeLoc = 101

The path of the node through the heap is right, left, right.

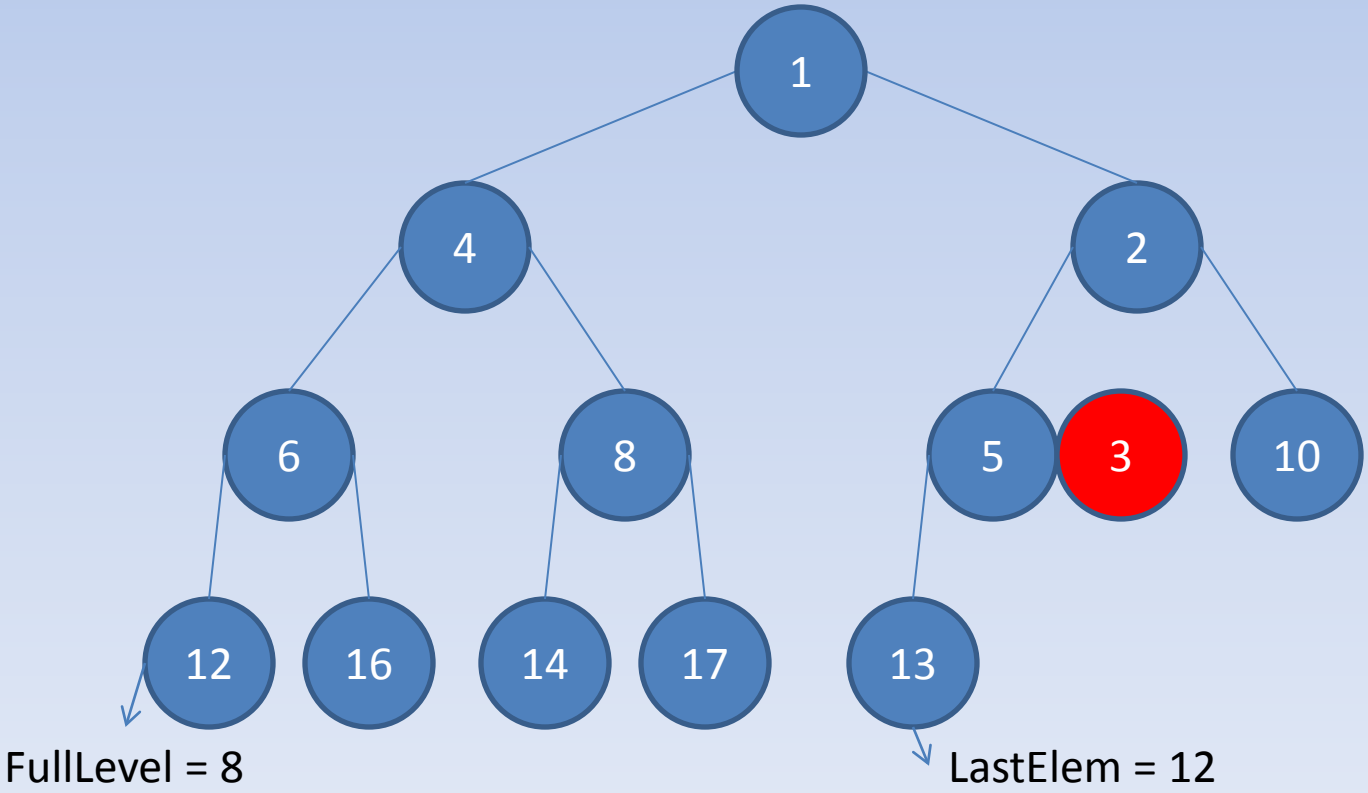


Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 3

NodeLoc = 101

The path of the node through the heap is right, left, right.

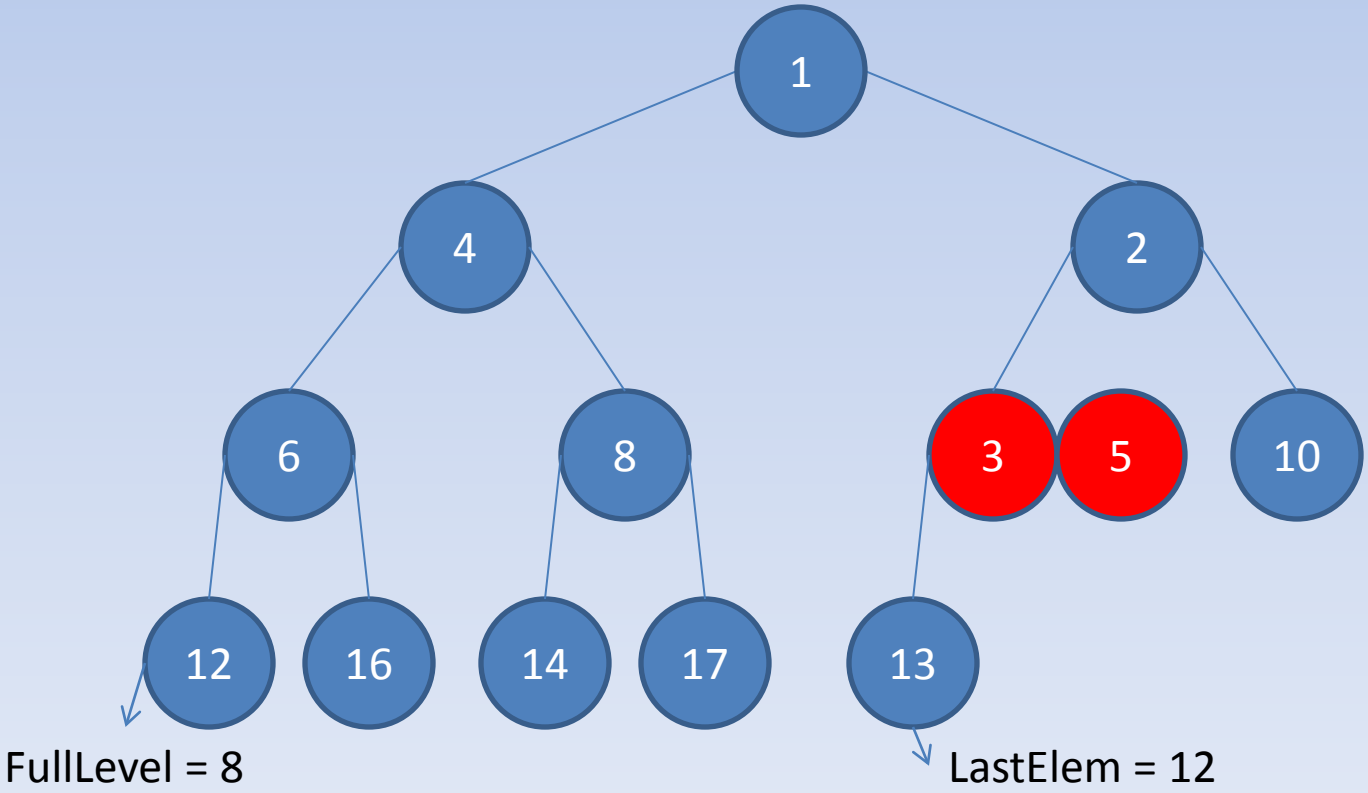


Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 3

NodeLoc = 101

The path of the node through the heap is right, left, right.

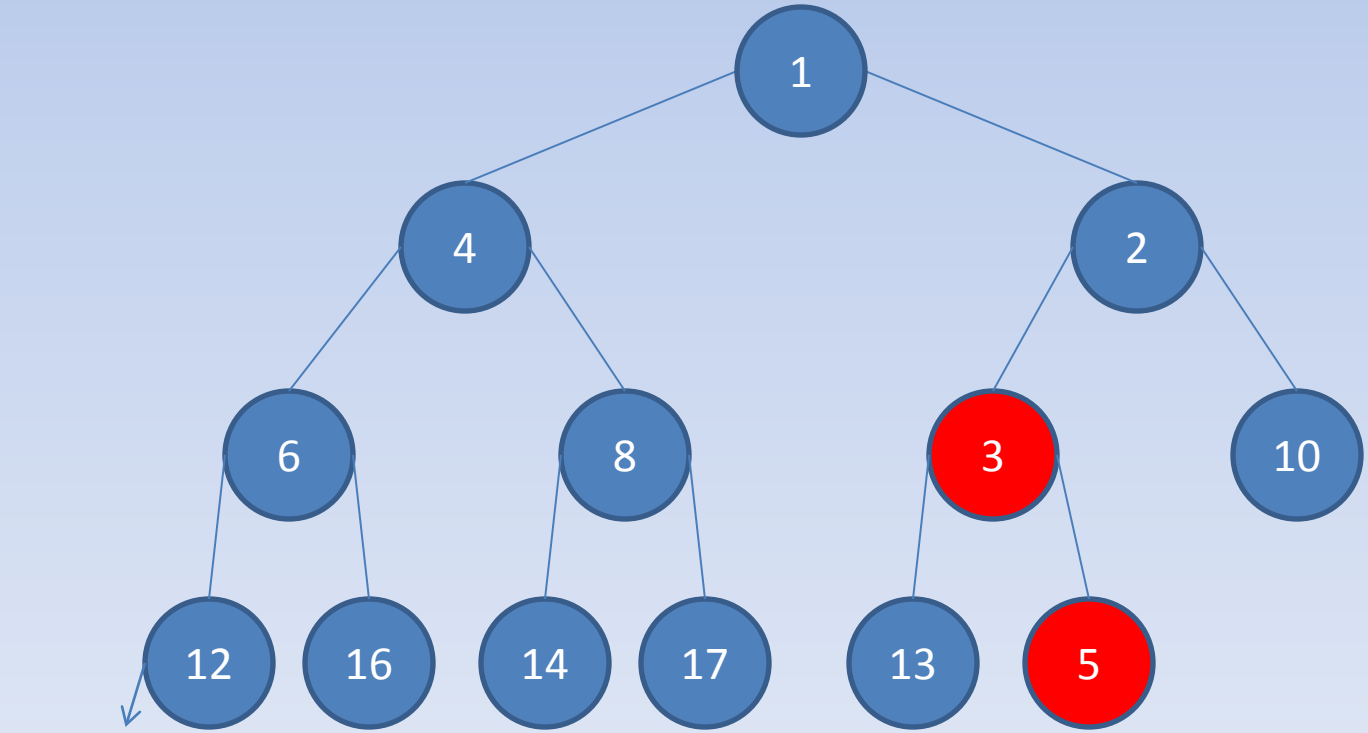


Top Down Insertions

NodeLoc = (LastElem + 1) – FullLevel = 3

NodeLoc = 101

The path of the node through the heap is right, left, right.



Deleting/Inserting

- Instead of locking the entire heap, only a small portion of the heap is locked.
- This locking window consists of three nodes for deletion (the parent and child nodes) and one node for insertions.
- In order to allow for window locking we associate a lock with each node.
- Anytime a node is accessed it is locked in order to maintain mutual exclusion.
- FullLevel and LastElem are modified only during the initialization part of an insertion or deletion call.

Deleting/Inserting

- Although insertion and deletion operation are performed on the heap from top to bottom, one issue still exists which prevents them from working together.

- This occurs when the delete operation requires a node that the insertion operation has not finished placing yet.
  - If an insertion operation is in progress, then this last node does not have a value.
  - If delete picks up the key of any other leaf node, then the resulting heap may become unbalanced.
  - If the delete operation waits for the insertion to complete then concurrency is lost.

Deleting/Inserting

- The problem is solved by associating a status field with each of the nodes.

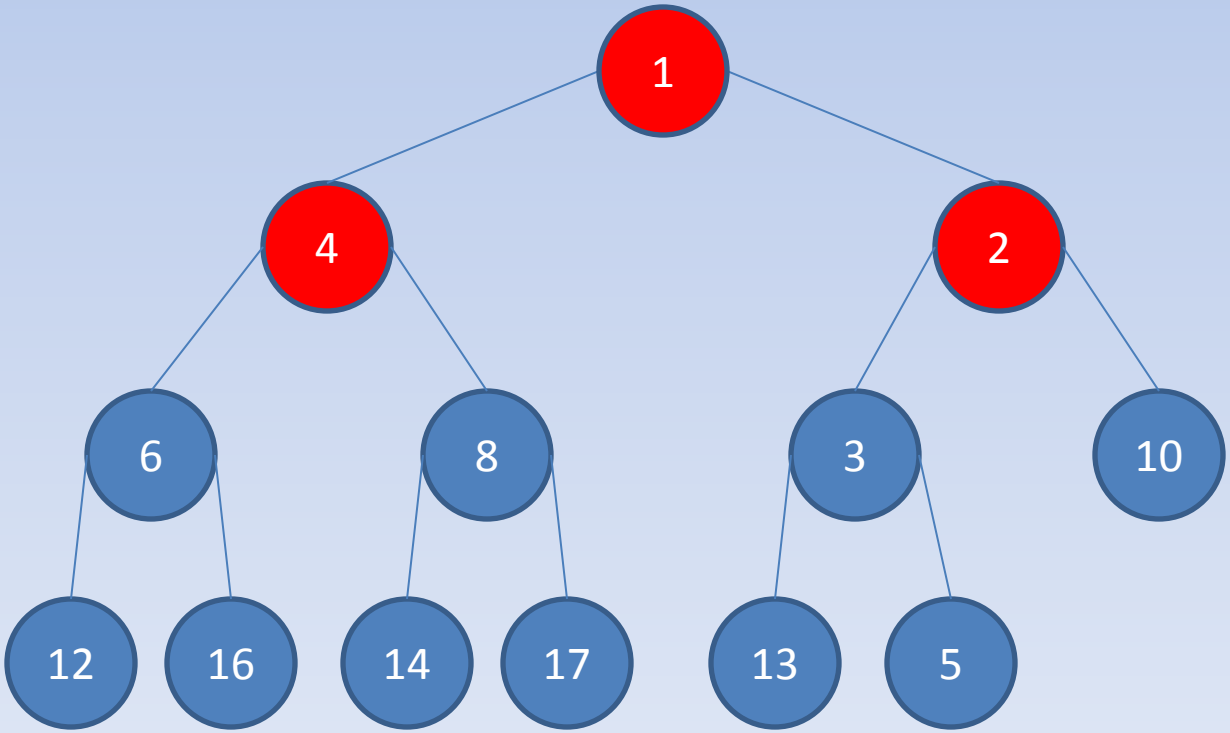
Status Code	**Meaning**
PRESENT	A key exists at the node.
PENDING	An insertion is in progress which will ultimately insert a key at the node.
WANTED	A deleter is waiting for the key.
ABSENT	No key is present at the node.

Deleting/Inserting

- When an insertion operation begins the target status is set to PENDING.
- If the deletion operation is invoked while an insertion operation is still in progress, the status of the target is changed to WANTED.
- During each reheapification loop, the insertion operation checks to see if the target node has changed its status to WANTED. If this is the case, the node is moved to the root of the heap and the insertion operation exits.

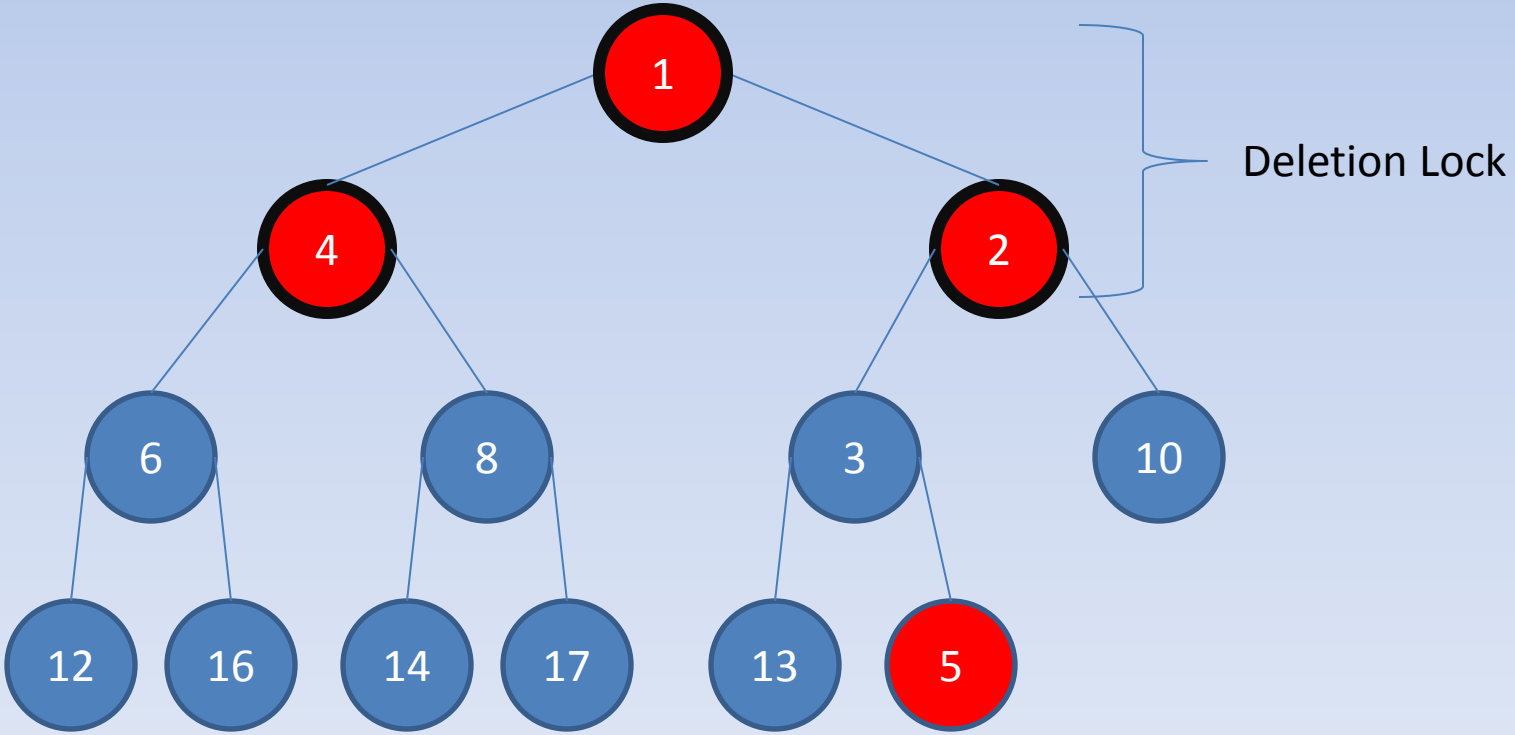
Deleting/Inserting

- Example of concurrent deletion and insertion operations.



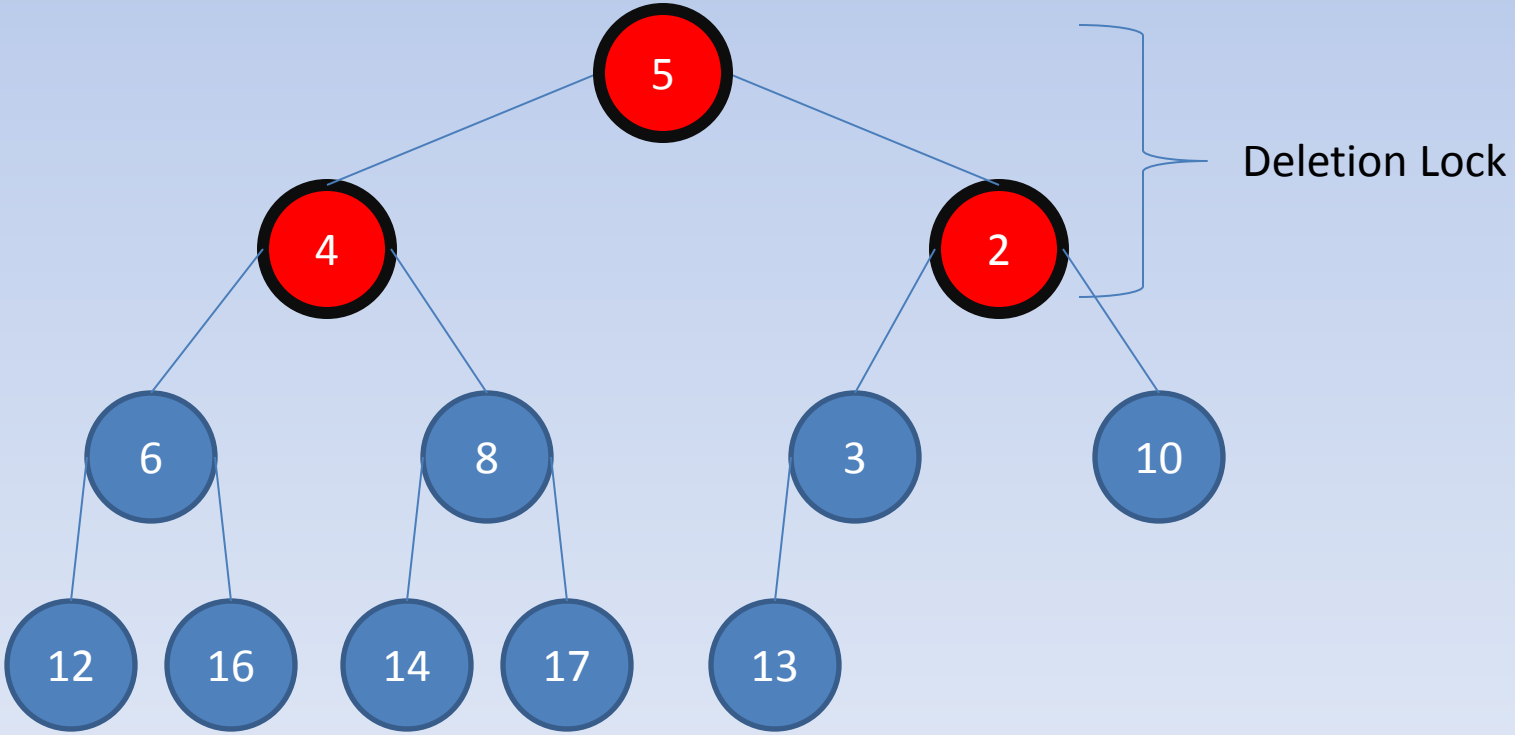
Deleting/Inserting

- The root node and child nodes are locked.



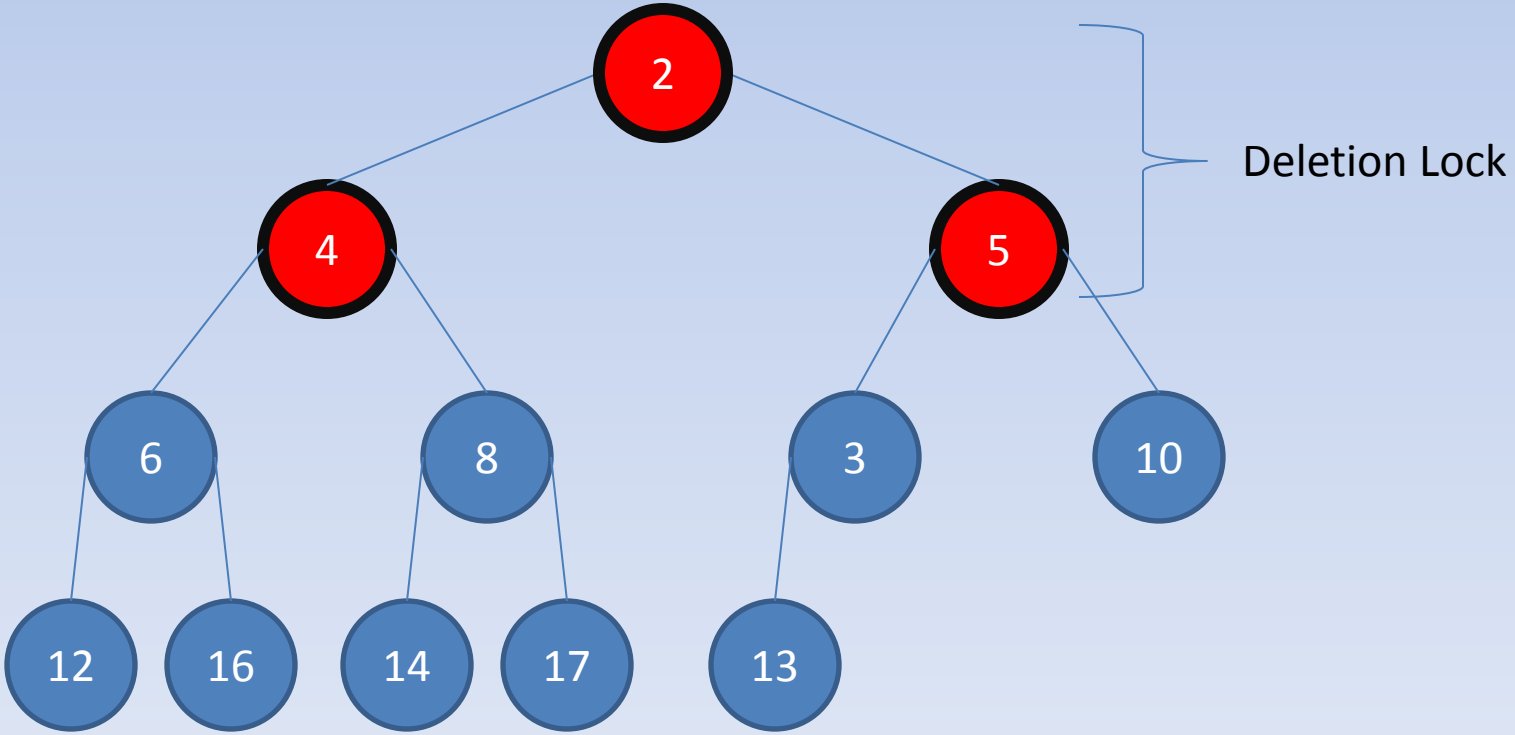
Deleting/Inserting

- The top node is removed and the last element is put at the root of the heap.



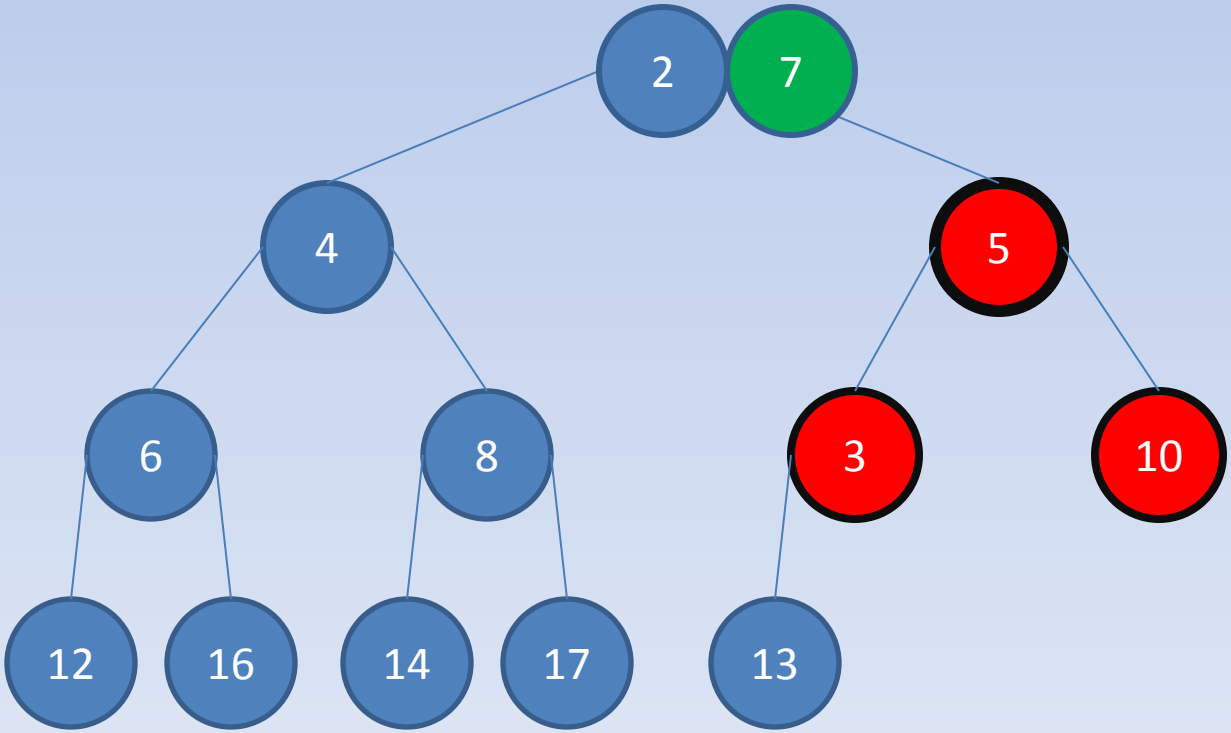
Deleting/Inserting

- Reheapification is performed in order to maintain the heap property.



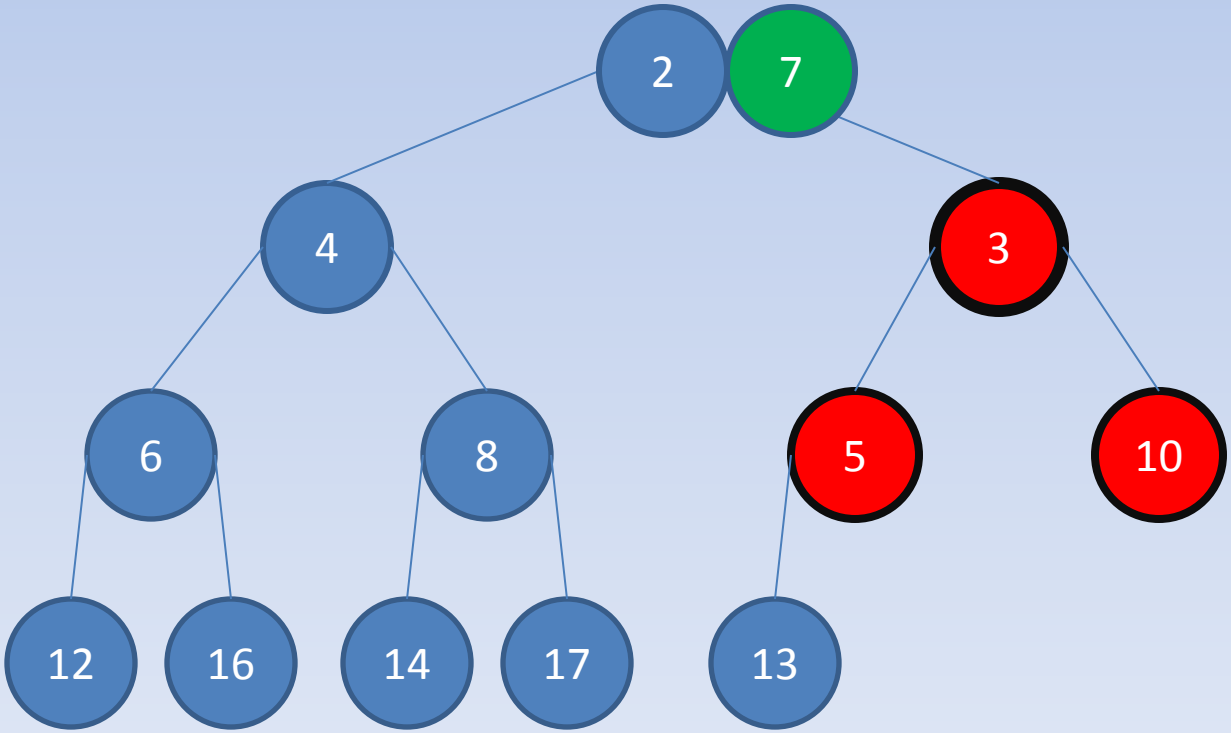
Deleting/Inserting

- An insertion operation has started.



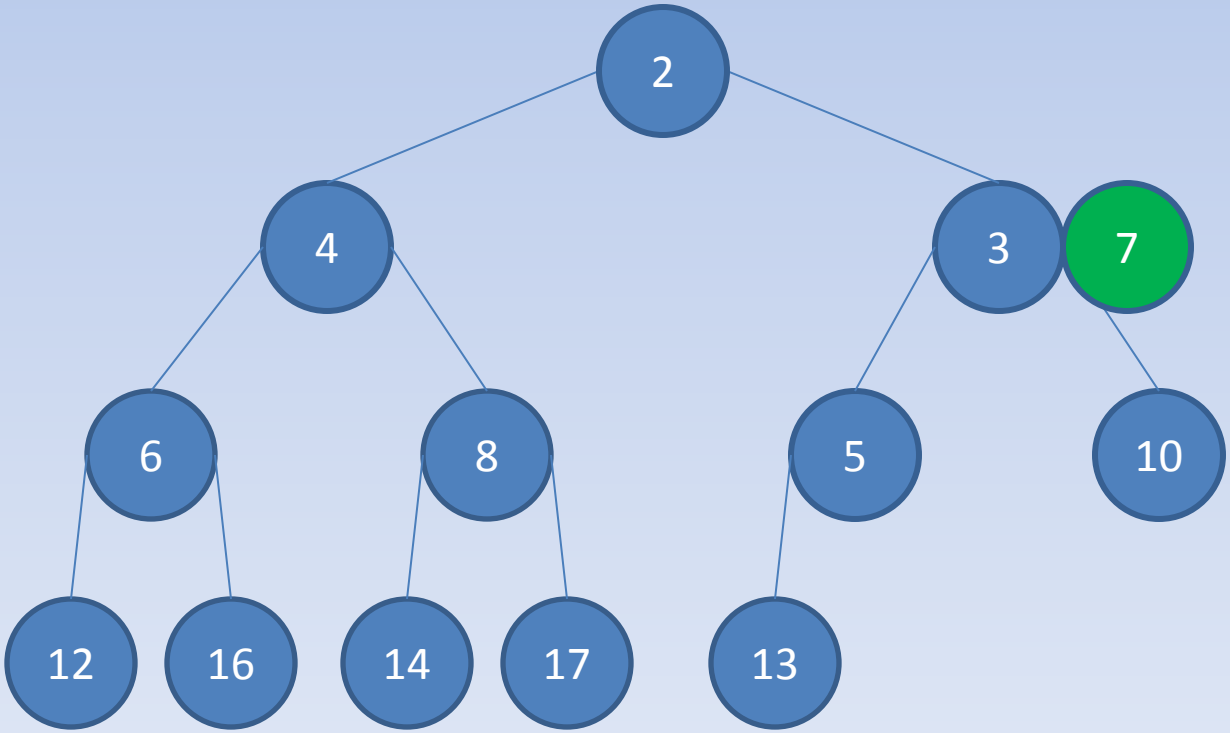
Deleting/Inserting

- The insertion operation cannot proceed until the deletion operation unlocks the required nodes.



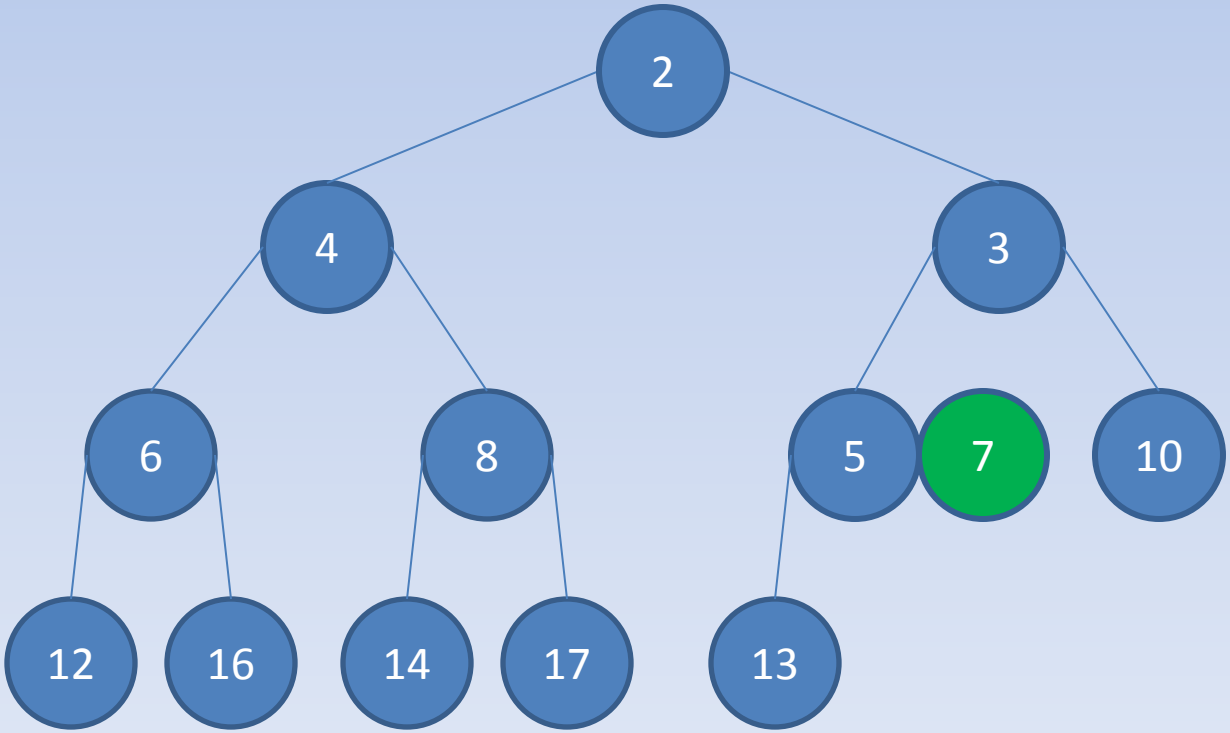
Deleting/Inserting

- The deletion operation has completed and the insertion operation can continue.



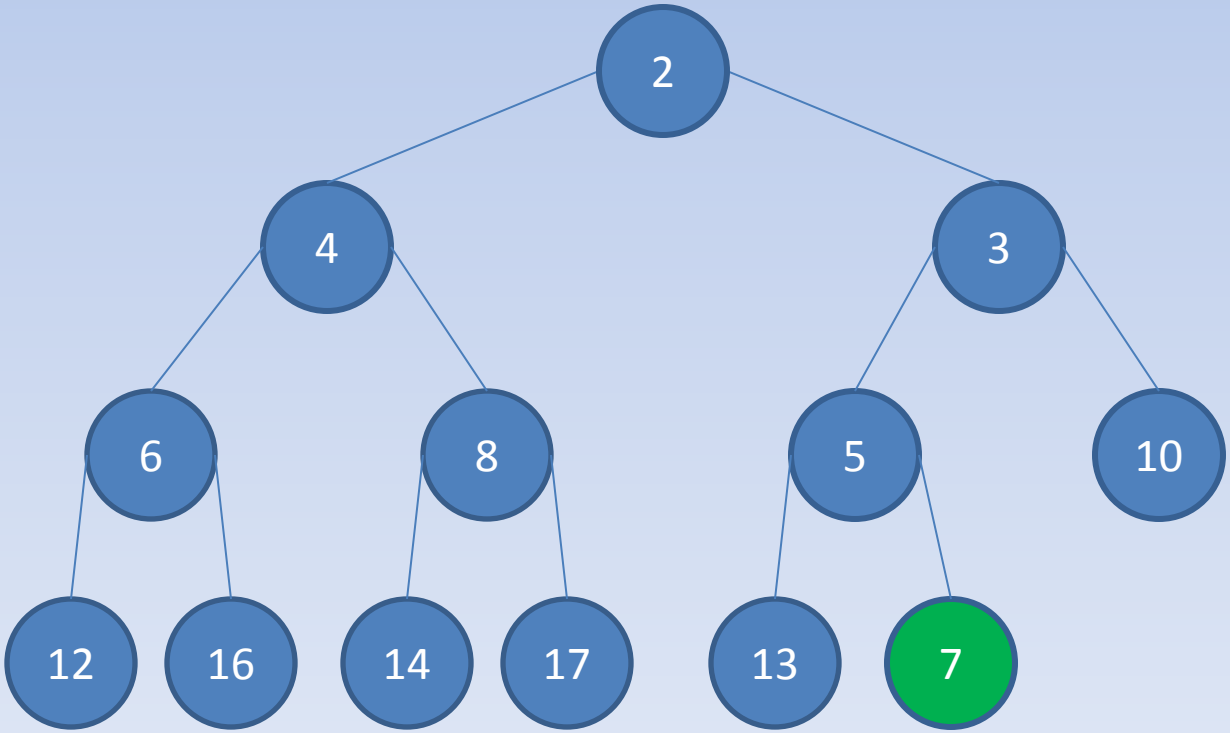
Deleting/Inserting

- The deletion operation has completed and the insertion operation can continue.



Deleting and Inserting

- The deletion operation has completed and the insertion operation can continue.



Future Work

- Implement the concurrent priority queue.
- Test the implementation using a range of insertion and deletion operations.
- Compare performance with a serial priority queue.

Sources

R.V. Nageshwara, V. Kumar. *Concurrent Access of Priority Queues. IEEE Transactions on Computers,* 37(12): 1657-1665, December 1988.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms,* Second Edition. MIT Press and McGraw-Hill, 2001.

Questions?