CS 61BL Lab 14

Ryan Purpura

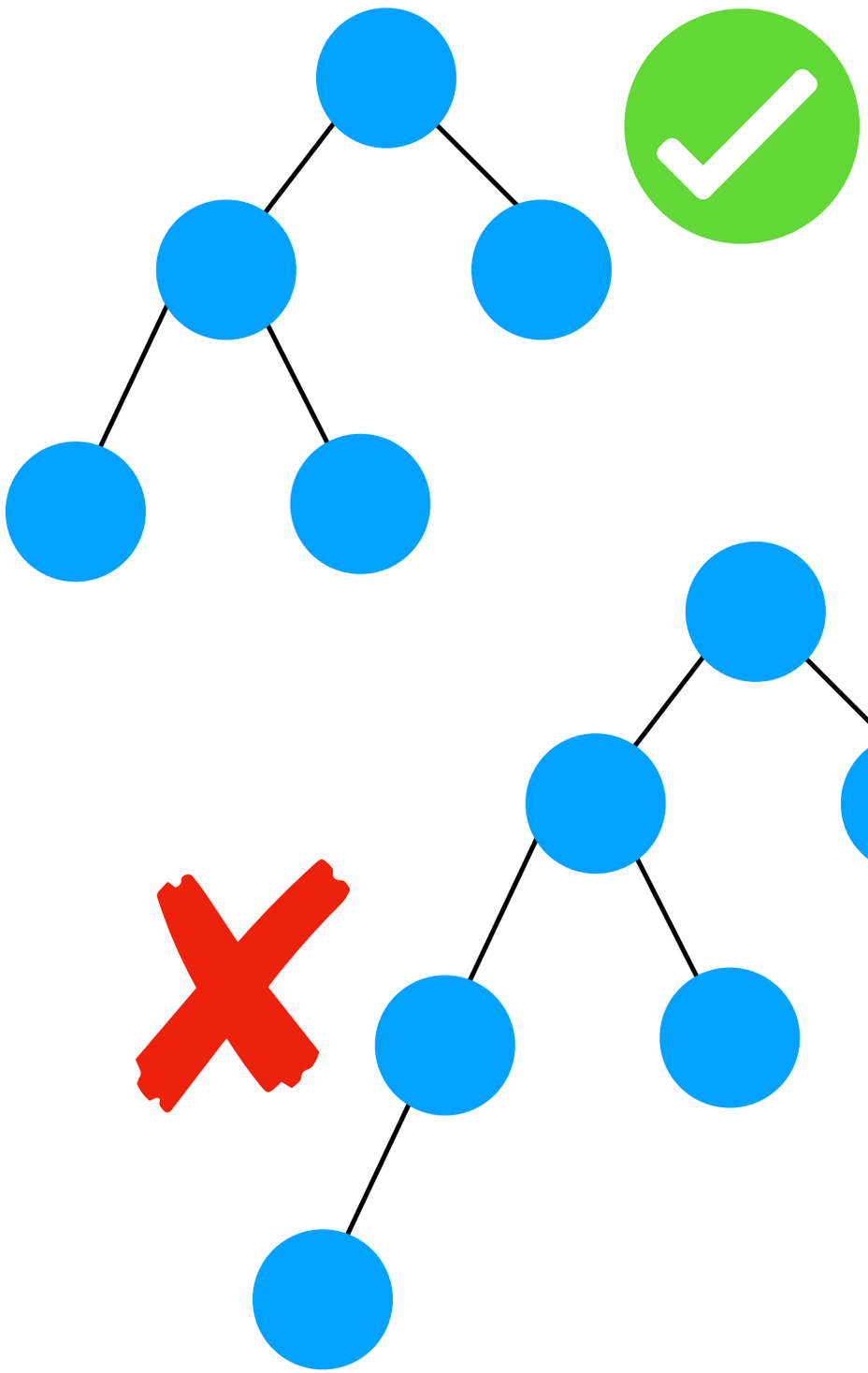
Announcements

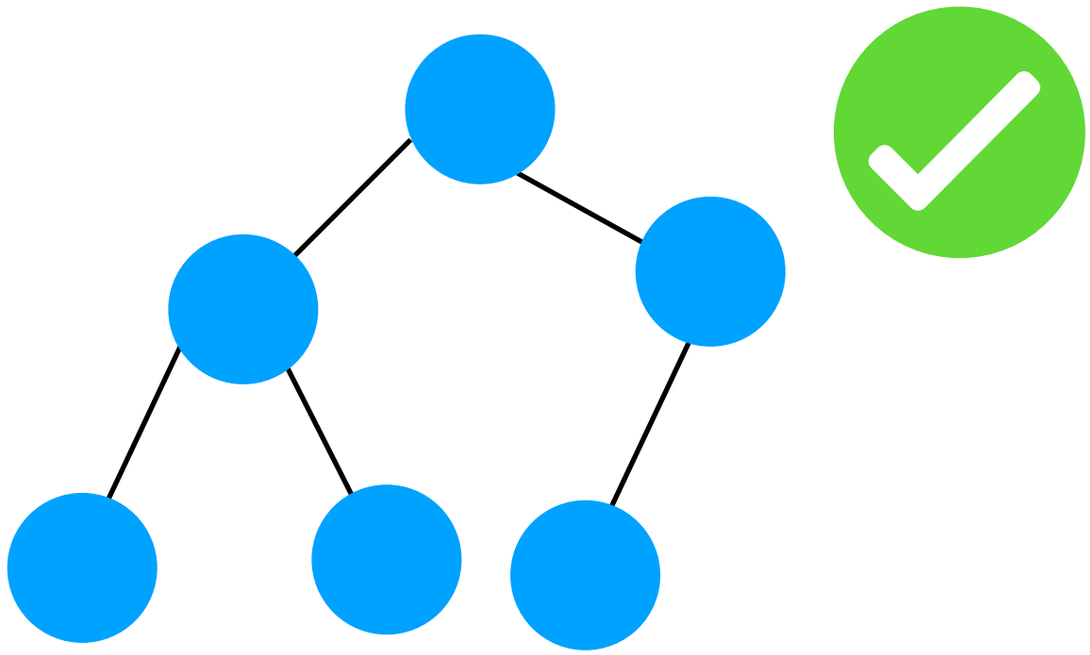
- Thursday has been converted into a project workday. Tries are no longer in scope for Midterm 2.
- Fill out the Mid-semester Survey!
- Midterm next Monday! Study a little bit every day.

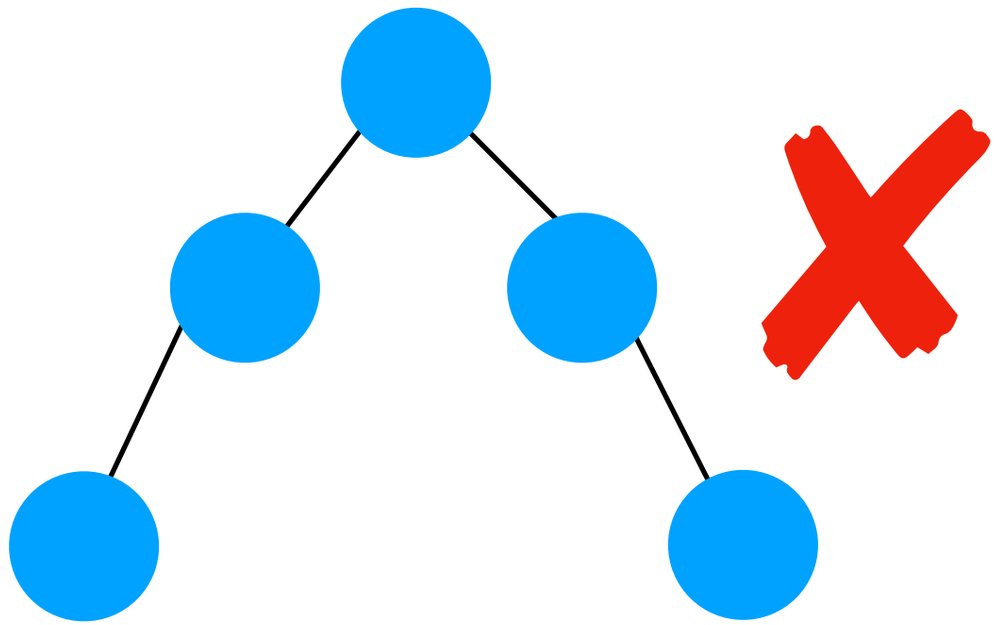
Heaps

- Heaps are trees that have two properties:
  - Completeness: all levels of the trees are filled except possibly the last level
    - If the last level is not completely filled, then it must be filled from left-to-right
  - All parents must be greater than their children (a "max-heap") or all parents must be less than their children (a "min-heap")
    - Known as the **heap property**

Completeness

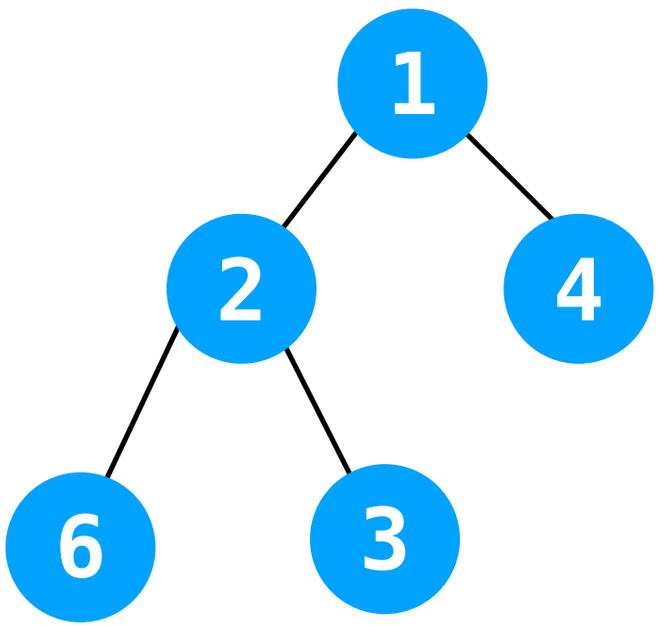




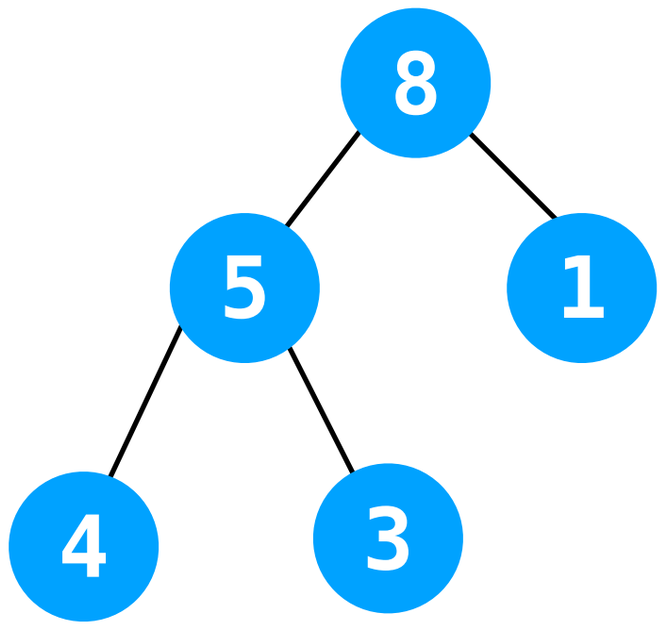


Heap Property

Min Heap

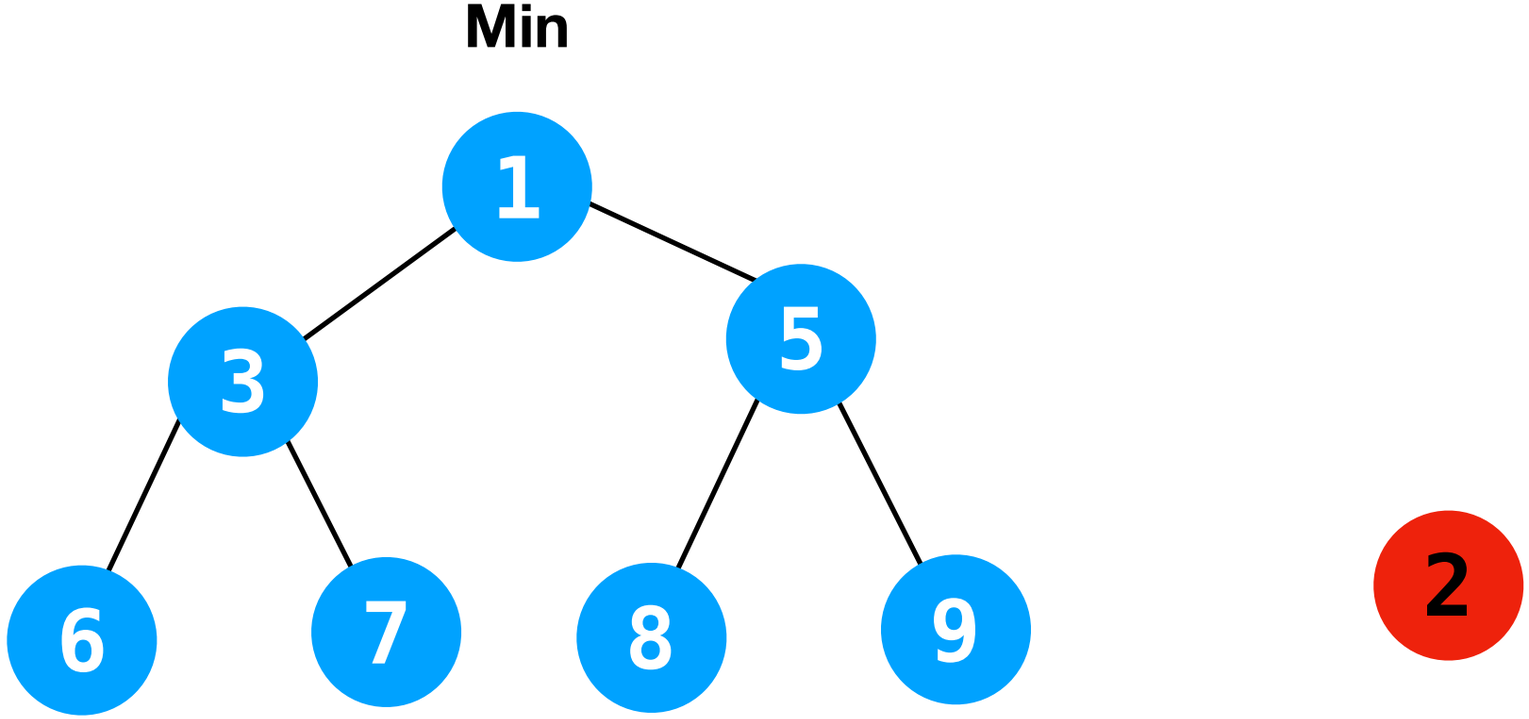


Max Heap



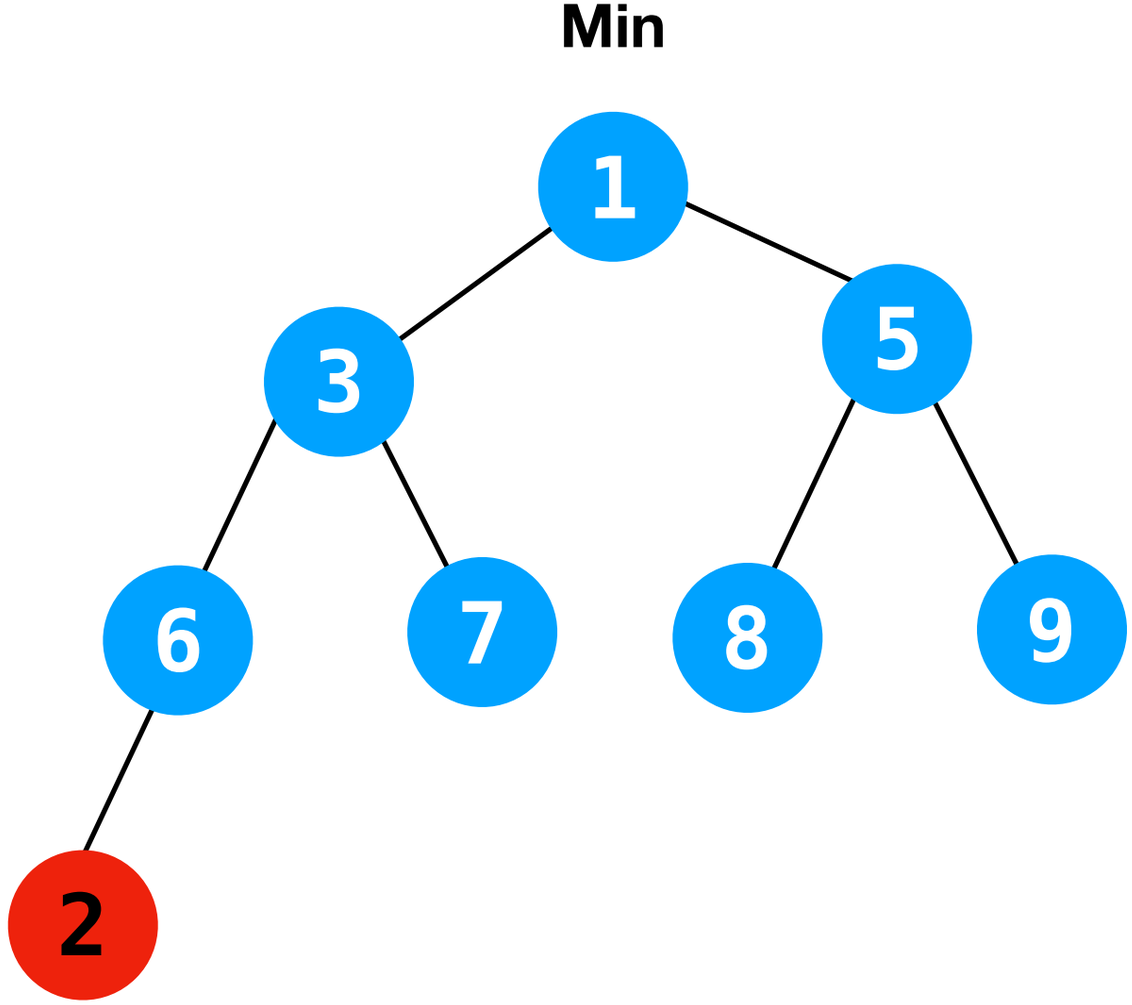
Insertion

- Add new element to bottom-right-most position of tree that's still valid.
- We need to maintain the heap property of our heap, so swap the element with its parent until the heap property is satisfied.



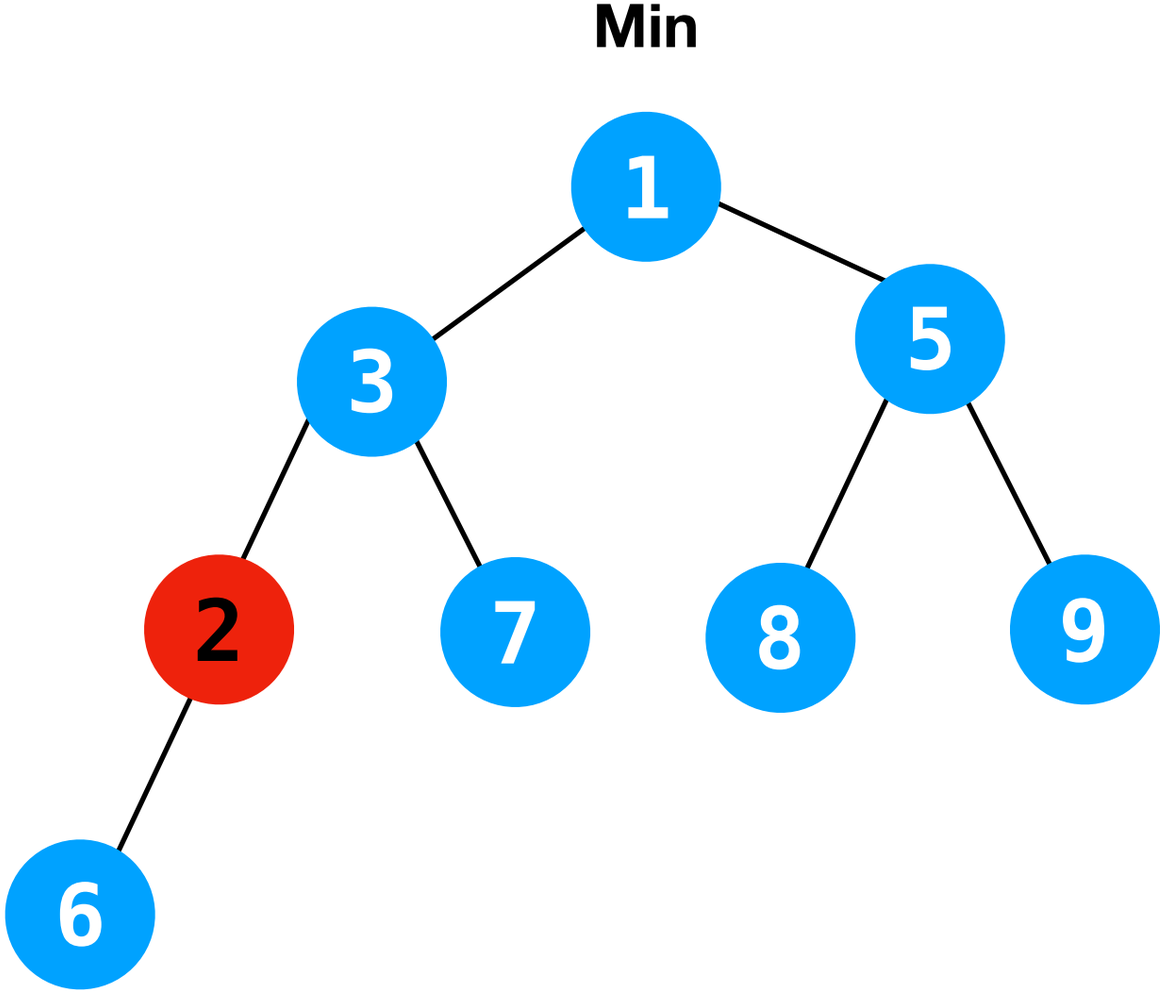
Insertion

- Add new element to bottom-right-most position of tree that's still valid.
- We need to maintain the heap property of our heap, so swap the element with its parent until the heap property is satisfied.



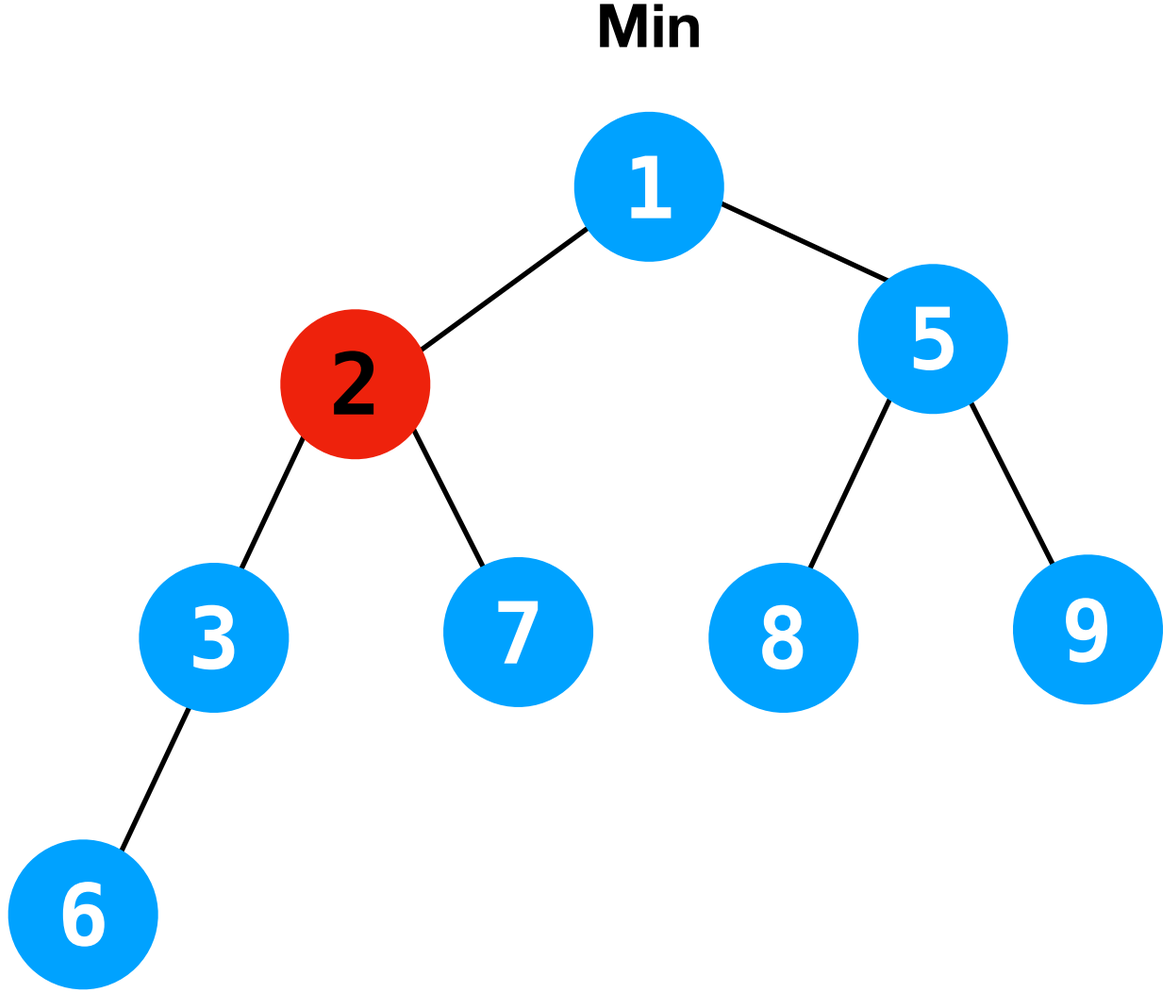
Insertion

- Add new element to bottom-right-most position of tree that's still valid.
- We need to maintain the heap property of our heap, so swap the element with its parent until the heap property is satisfied.



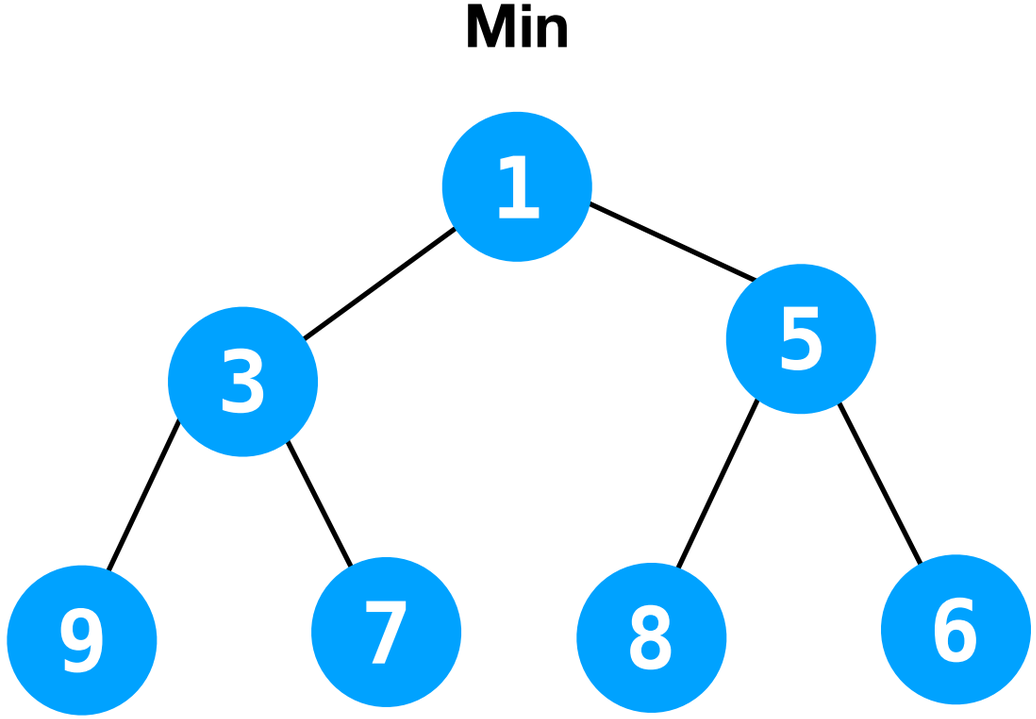
Insertion

- Add new element to bottom-right-most position of tree that's still valid.
- We need to maintain the heap property of our heap, so swap the element with its parent until the heap property is satisfied.



Deleting the root

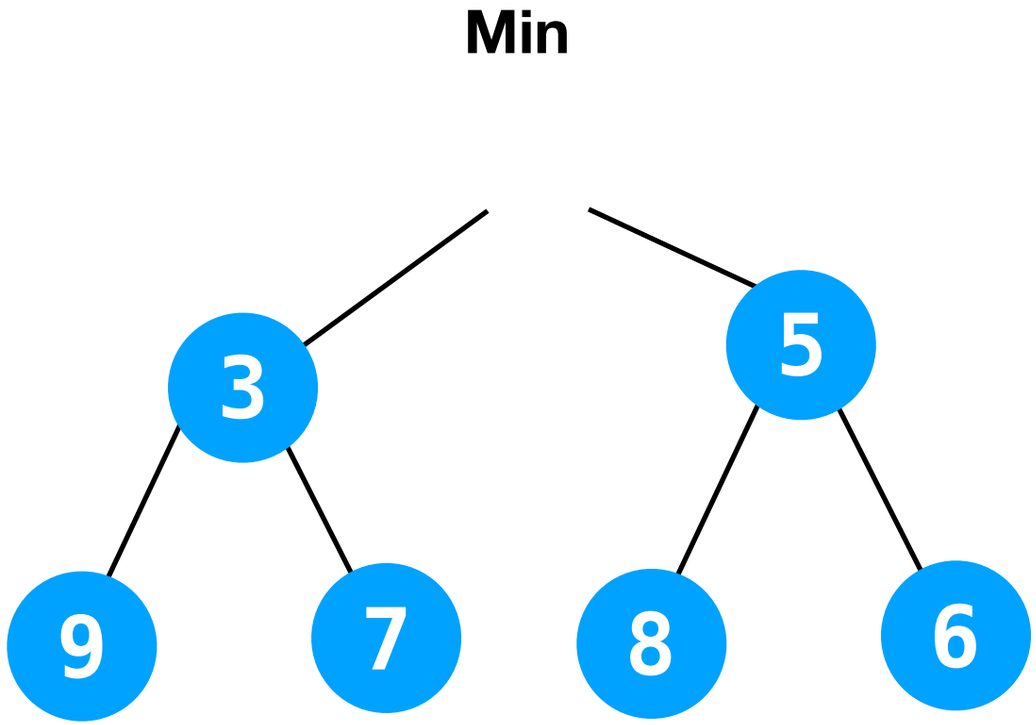
- Remove the root, and move the bottom-right-most node into the root position.
- Swap with the child that best restores the heap property.



Smallest child for a min-heap, largest child for a max-heap

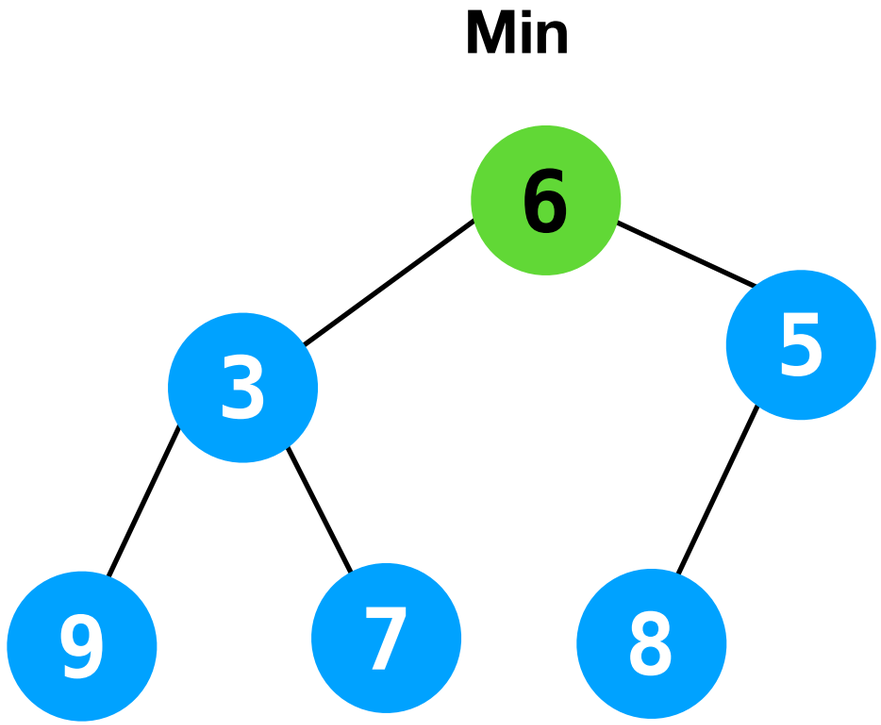
Deleting the root

- Remove the root, and move the bottom-right-most node into the root position.
- Swap with the child that best restores the heap property.



Deleting the root

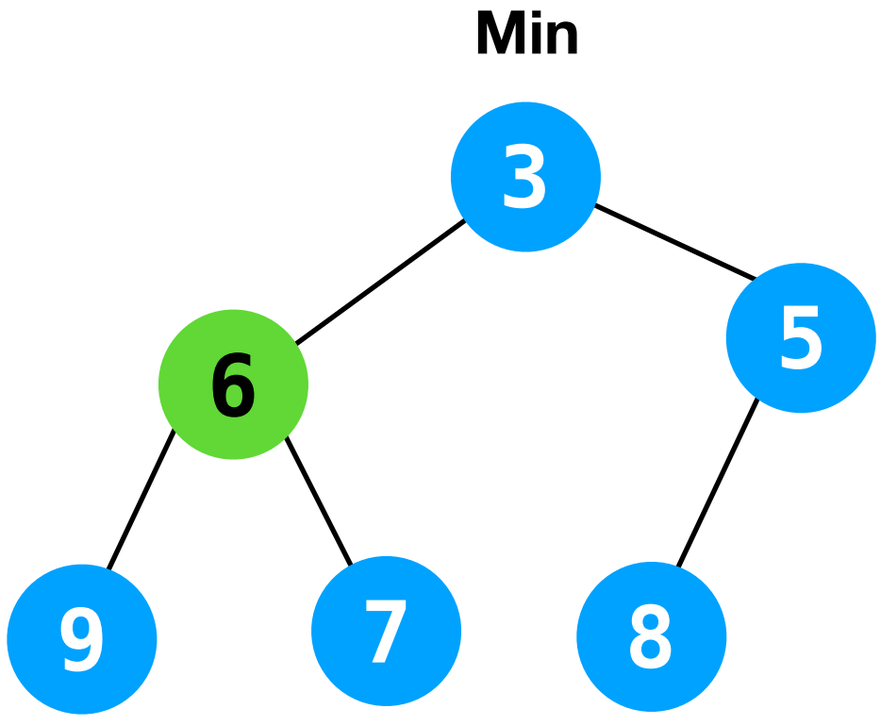
- Remove the root, and move the bottom-right-most node into the root position.
- Swap with the child that best restores the heap property.



Smallest child for a min-heap, largest child for a max-heap

Deleting the root

- Remove the root, and move the bottom-right-most node into the root position.
- Swap with the child that best restores the heap property.



Smallest child for a min-heap, largest child for a max-heap

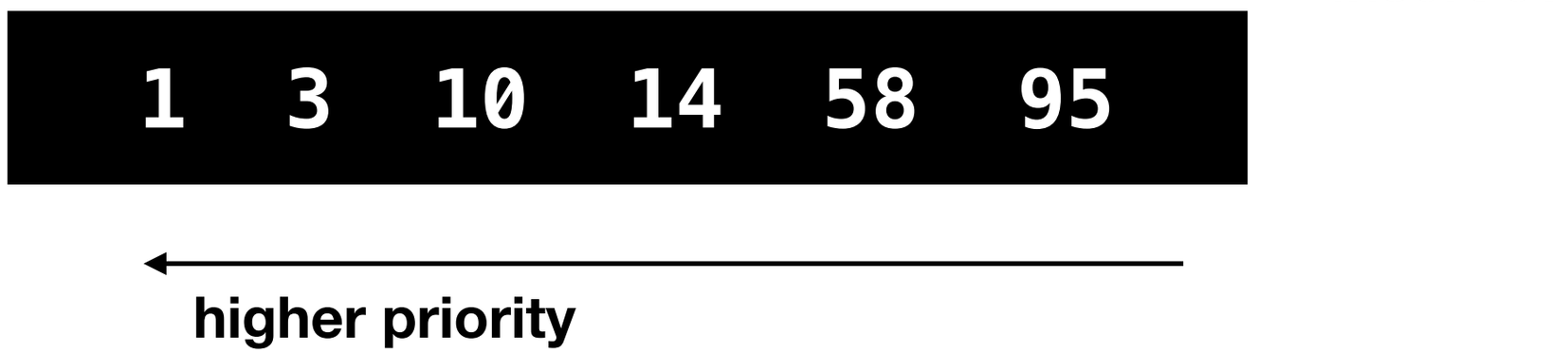
Why Heaps Are Useful

- Removing the root and inserting new elements are both $\log N$ time.
- Furthermore, the root is always the smallest (for min-heap) or largest (for max-heap) element
- This makes it extremely useful for implementing *priority queues*

Priority Queues

- A normal queue processes items in the order that the items are added: the oldest items are processed first.
- This is not always what we want: we might want to process items in order of importance, aka priority.
  - If a high-priority item is inserted, it should be able to cut in front of lower-priority items.
- The data structure that accomplishes this is called a priority queue.

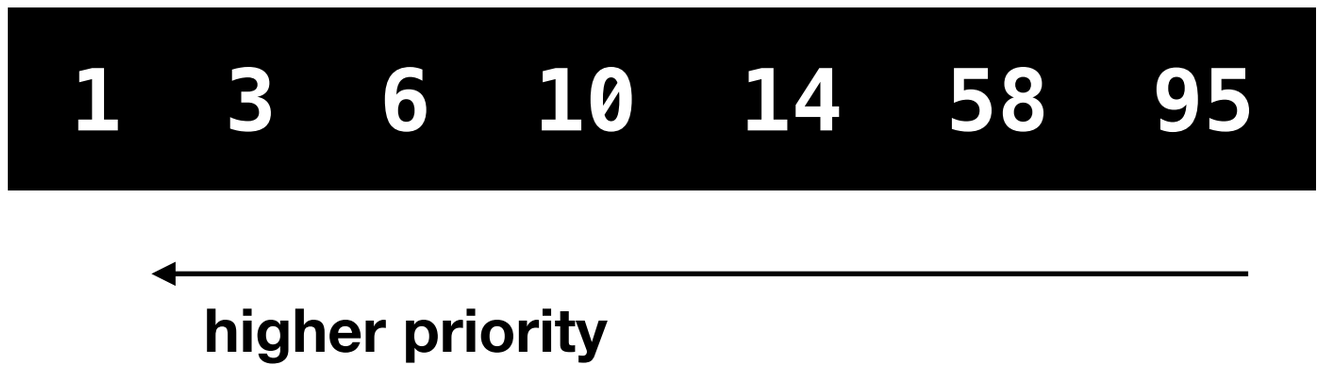
Example of priority queue where smaller numbers have higher priority



Priority Queues

- A normal queue processes items in the order that the items are added: the oldest items are processed first.
- This is not always what we want: we might want to process items in order of importance, aka priority.
  - If a high-priority item is inserted, it should be able to cut in front of lower-priority items.
- The data structure that accomplishes this is called a priority queue.

Example of priority queue where smaller numbers have higher priority



Implementing Priority Queues

- Goals:
  - Removing highest priority element efficiently
  - Inserting new elements efficiently

Use heaps!

- Heaps give us both properties.
- Removing highest priority element efficiently?
  - The root is always the largest/smallest value, and we can remove it and restore the heap properties in logarithmic time.
- Inserting new elements efficiently?
  - We just use the insertion algorithm from earlier, which runs logarithmically.