Priority Queues

Two kinds of priority queues:

- Min priority queue.
- Max priority queue.

Min Priority Queue

- Collection of elements.
- Each element has a priority or key.
- Supports following operations:
  - empty
  - size
  - insert an element into the priority queue (push)
  - get element with min priority (top)
  - remove element with min priority (pop)

Max Priority Queue

- Collection of elements.
- Each element has a priority or key.
- Supports following operations:
  - empty
  - size
  - insert an element into the priority queue (push)
  - get element with max priority (top)
  - remove element with max priority (pop)

Complexity Of Operations

Use a heap or a leftist tree (both are defined later).

empty, size, and top => O(1) time

insert (push) and remove (pop) => O(log n) time where n is the size of the priority queue

Applications

Sorting

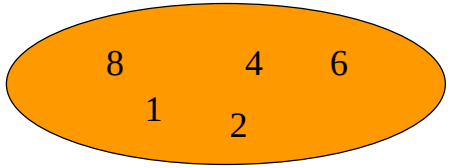
- use element key as priority
- insert elements to be sorted into a priority queue
- remove/pop elements in priority order
  - if a min priority queue is used, elements are extracted in ascending order of priority (or key)
  - if a max priority queue is used, elements are extracted in descending order of priority (or key)

Sorting Example

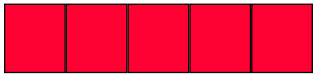
Sort five elements whose keys are 6, 8, 2, 4, 1 using a max priority queue.

- Insert the five elements into a max priority queue.
- Do five remove max operations placing removed elements into the sorted array from right to left.

After Inserting Into Max Priority Queue

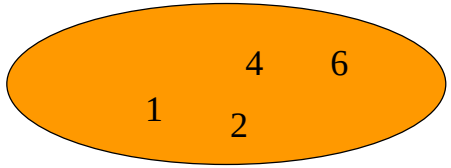


Max Priority Queue

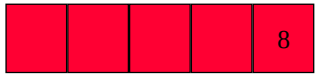


Sorted Array

After First Remove Max Operation

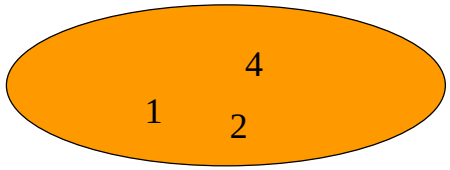


Max Priority Queue

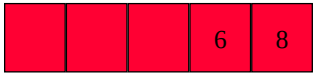


Sorted Array

After Second Remove Max Operation

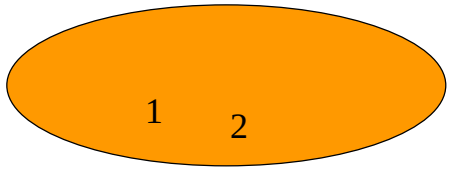


Max Priority Queue

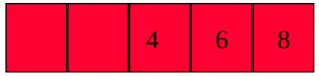


Sorted Array

After Third Remove Max Operation

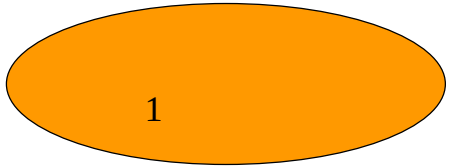


Max Priority Queue

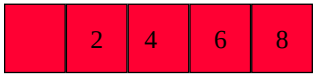


Sorted Array

After Fourth Remove Max Operation

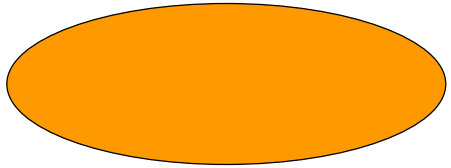


Max Priority Queue

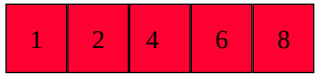


Sorted Array

After Fifth Remove Max Operation



Max Priority Queue



Sorted Array

Complexity Of Sorting

Sort n elements.

- n insert operations => O(n log n) time.
- n remove max operations => O(n log n) time.
- total time is O(n log n).
- compare with $O(n^2)$ for insertion sort of Lecture 1.

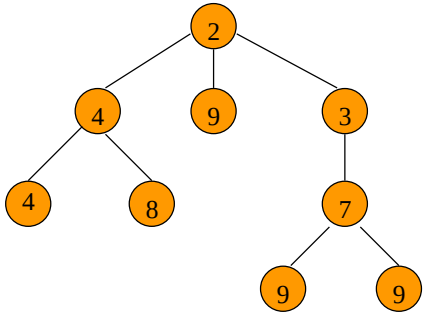
Min Tree Definition

Each tree node has a value.

Value in any node is the minimum value in the subtree for which that node is the root.

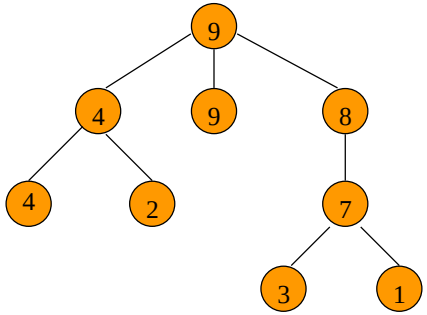
Equivalently, no descendent has a smaller value.

Min Tree Example



Root has minimum element.

Max Tree Example

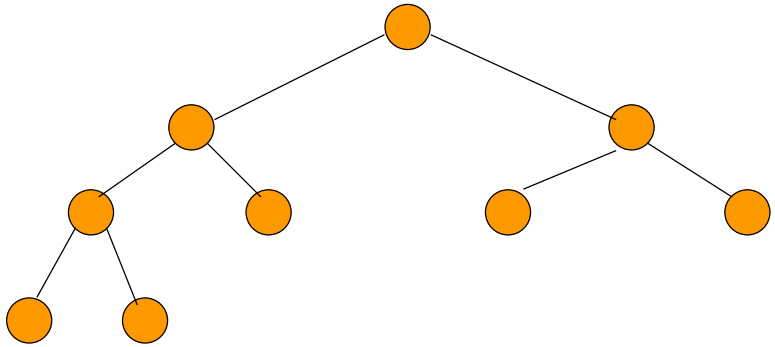


Root has maximum element.

Min Heap Definition

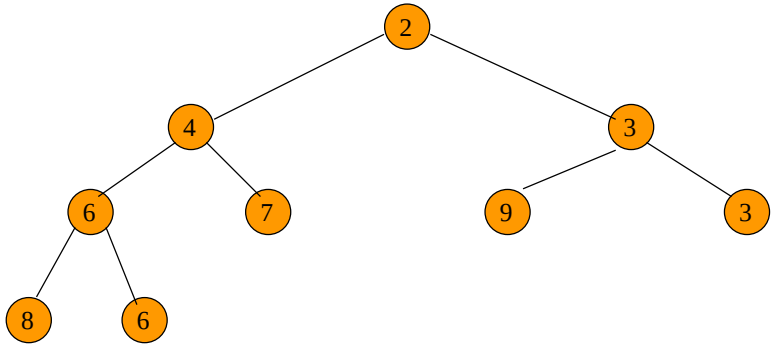
- complete binary tree
- min tree

Min Heap With 9 Nodes



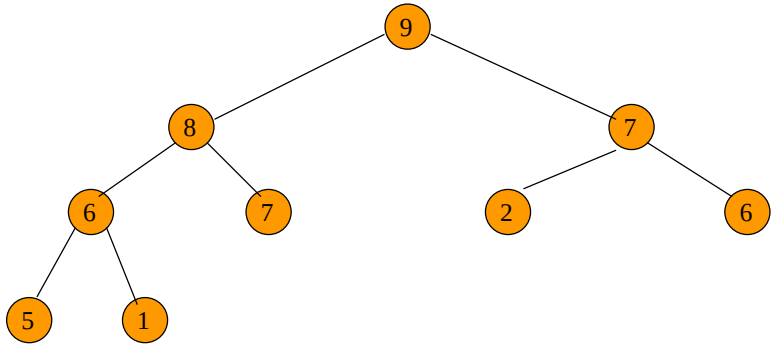
Complete binary tree with 9 nodes.

Min Heap With 9 Nodes



Complete binary tree with 9 nodes
that is also a min tree.

Max Heap With 9 Nodes

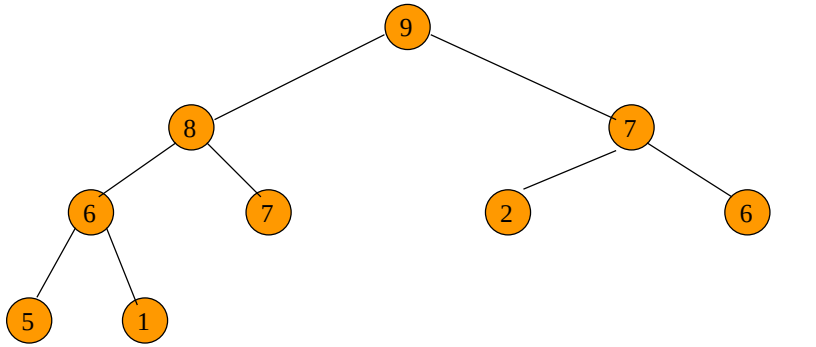


Complete binary tree with 9 nodes
that is also a max tree.

Heap Height

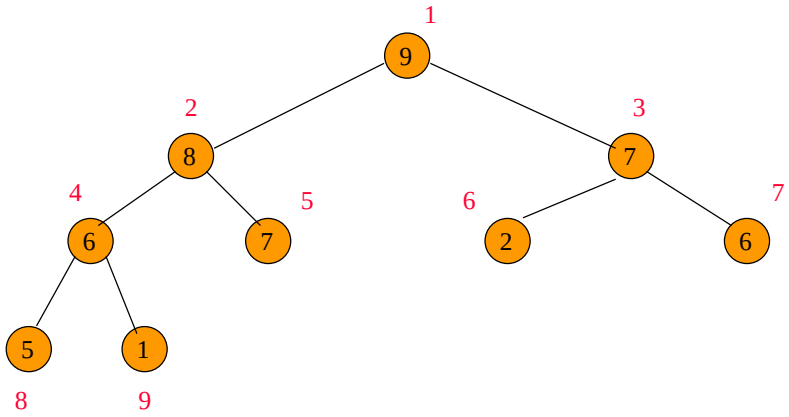
Since a heap is a complete binary tree, the height of an n node heap is $\log_2 (n+1)$.

A Heap Is Efficiently Represented As An Array

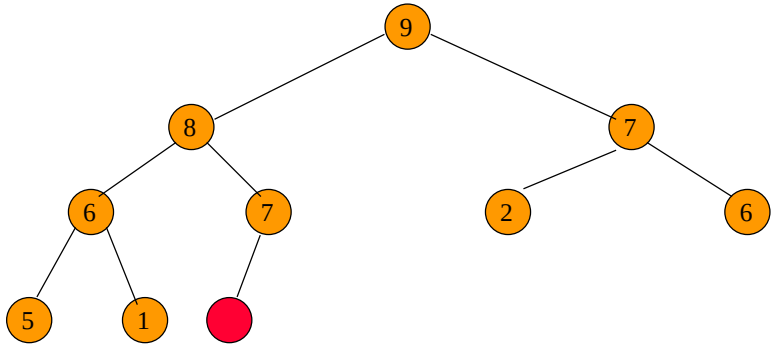


index 0 1 2 3 4 5 6 7 8 9 10

Moving Up And Down A Heap

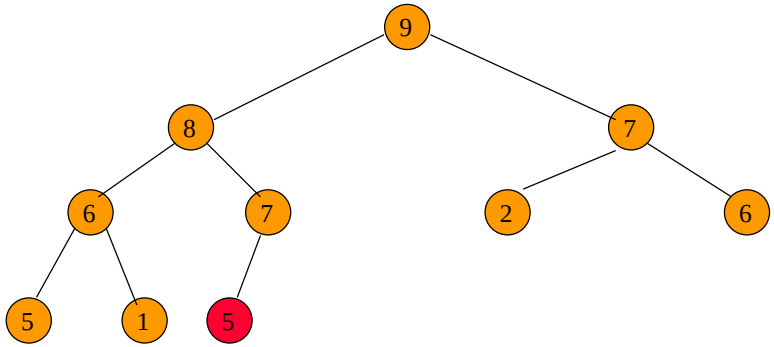


Inserting An Element Into A Max Heap



Complete binary tree with 10 nodes.

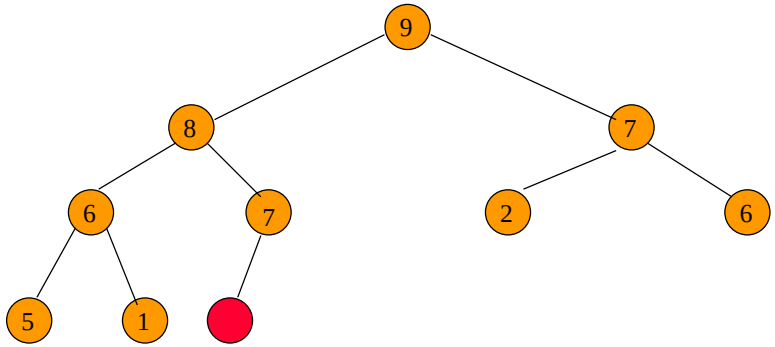
Inserting An Element Into A Max Heap



New element is 5.

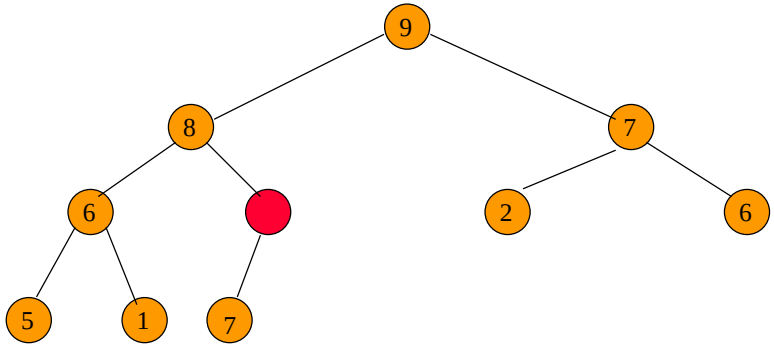
Needless to adjust the tree

Inserting An Element Into A Max Heap



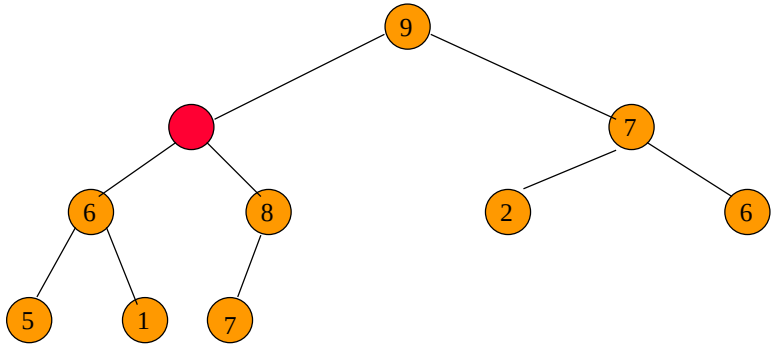
New element is 20.

Inserting An Element Into A Max Heap



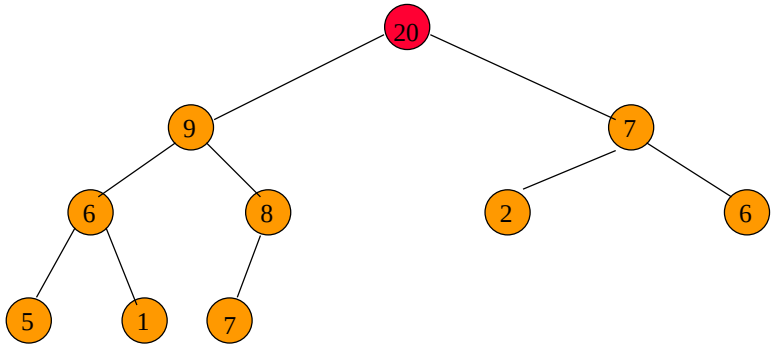
New element is 20.

Inserting An Element Into A Max Heap



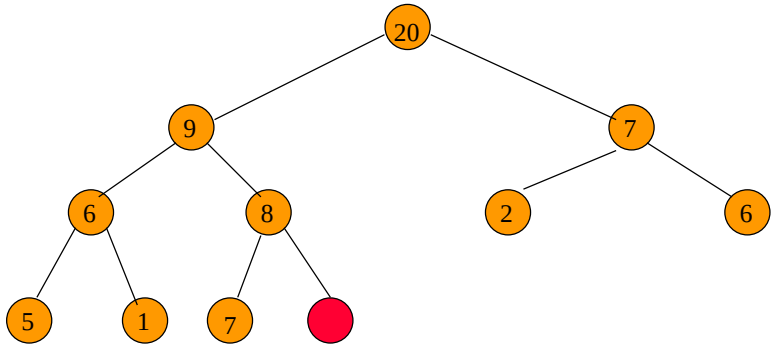
New element is 20.

Inserting An Element Into A Max Heap



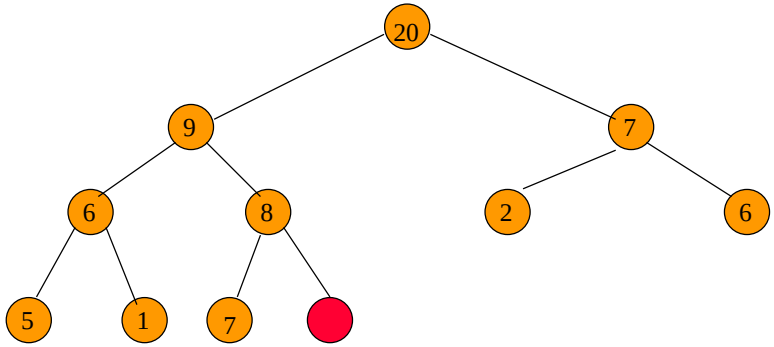
New element is 20.

Inserting An Element Into A Max Heap



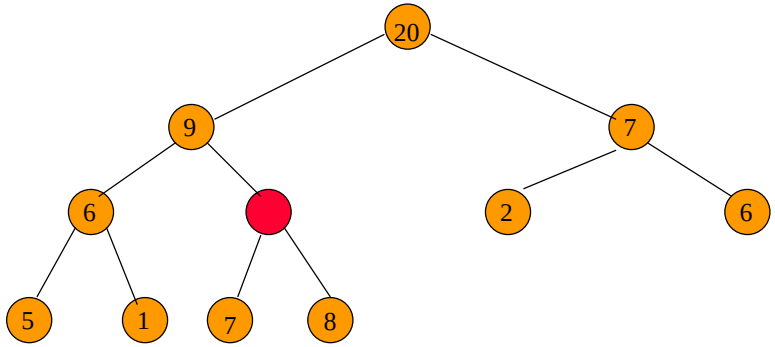
Complete binary tree with 11 nodes.

Inserting An Element Into A Max Heap



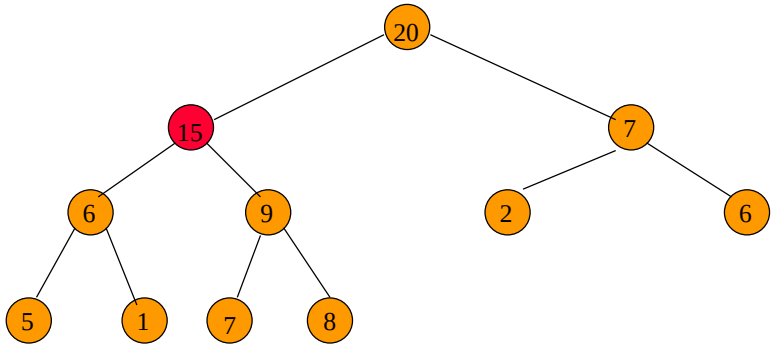
New element is 15.

Inserting An Element Into A Max Heap



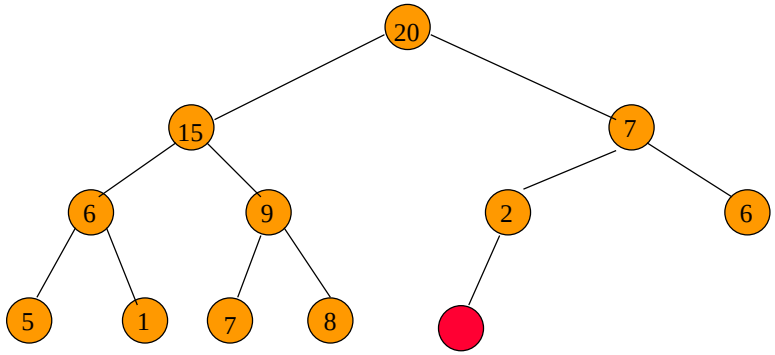
New element is 15.

Inserting An Element Into A Max Heap



New element is 15.

Complexity Of Insert

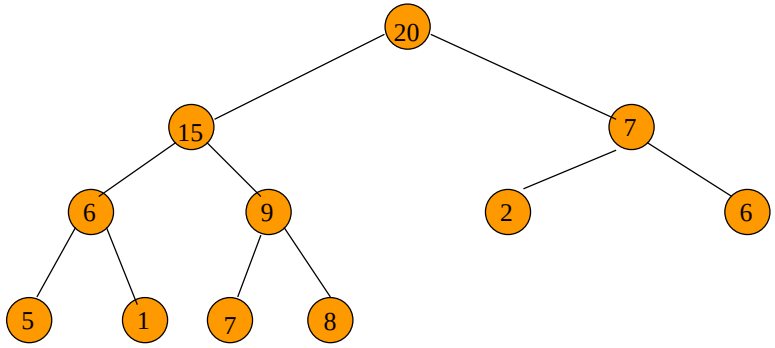


Complexity is O(log n), where n is heap size.

In Class Exercise

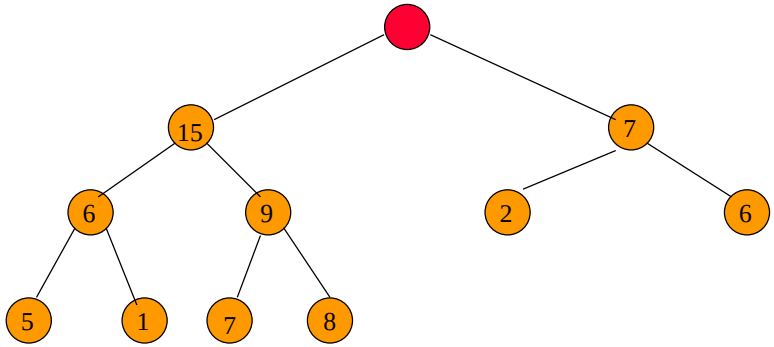
- Insert 17 into the heap illustrated in last slide.
- Prove that above insertion results in a valid max heap.

Removing The Max Element



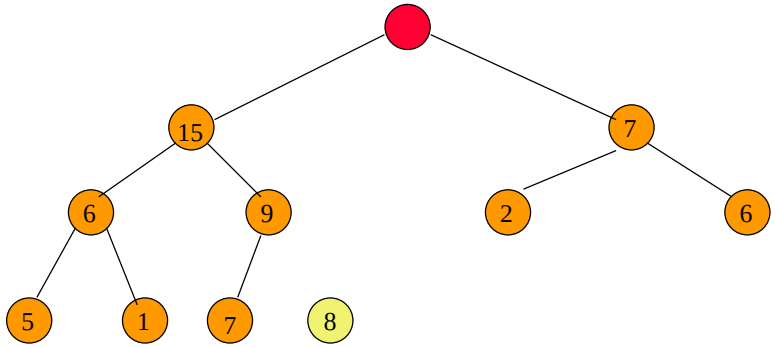
Max element is in the root.

Removing The Max Element



After max element is removed.

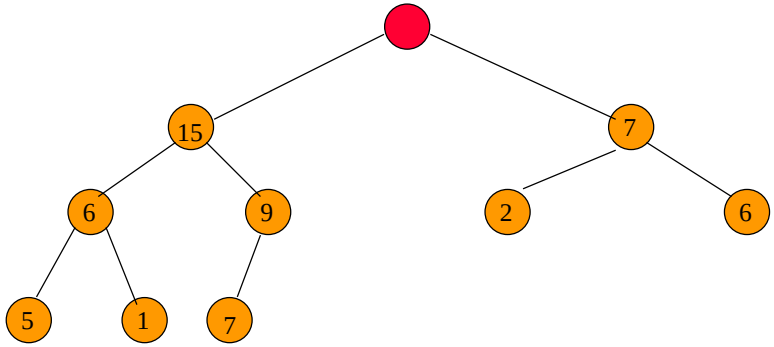
Removing The Max Element



Heap with 10 nodes.

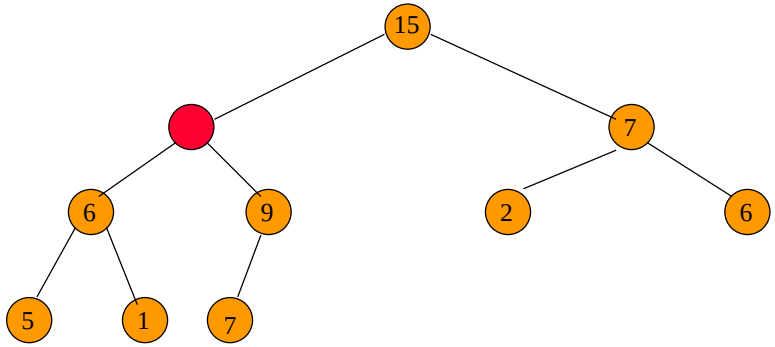
Move 8 to the root of heap.

Removing The Max Element



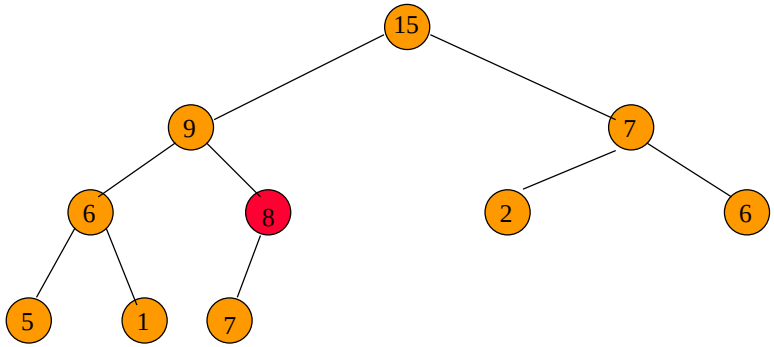
Reinsert 8 into the heap.

Removing The Max Element



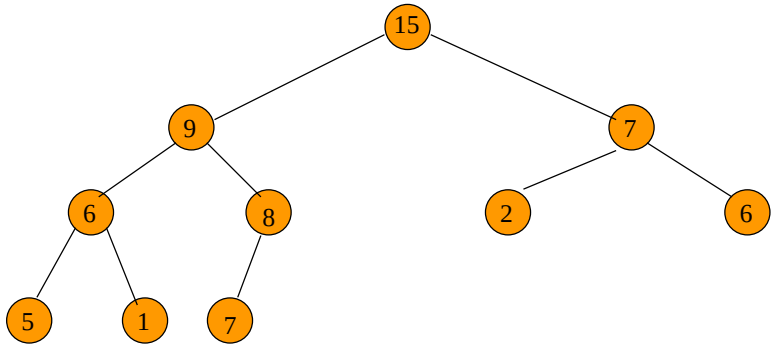
Reinsert 8 into the heap.

Removing The Max Element



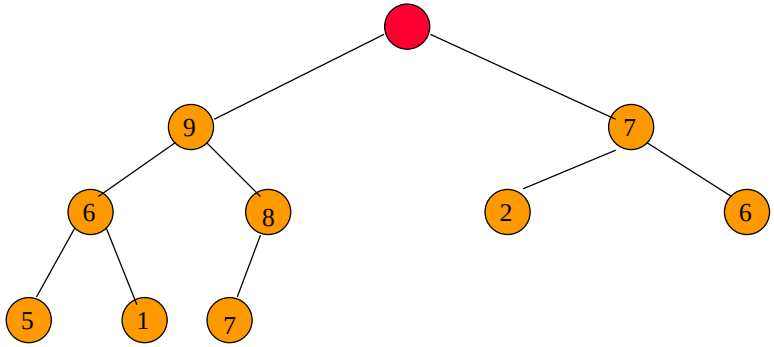
Reinsert 8 into the heap.

Removing The Max Element



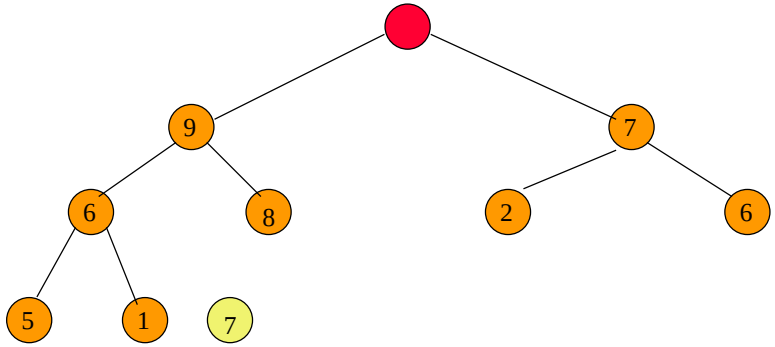
Max element is 15.

Removing The Max Element



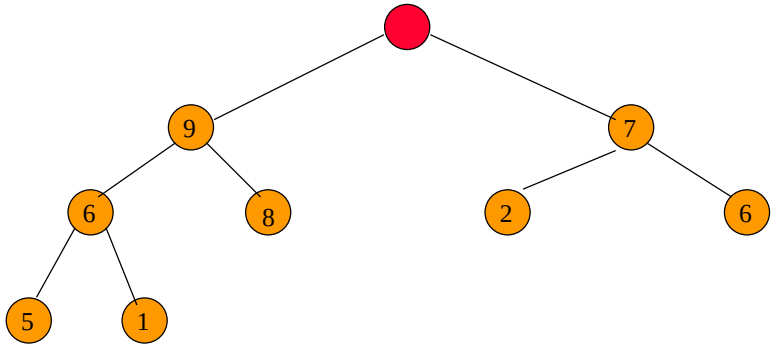
After max element is removed.

Removing The Max Element



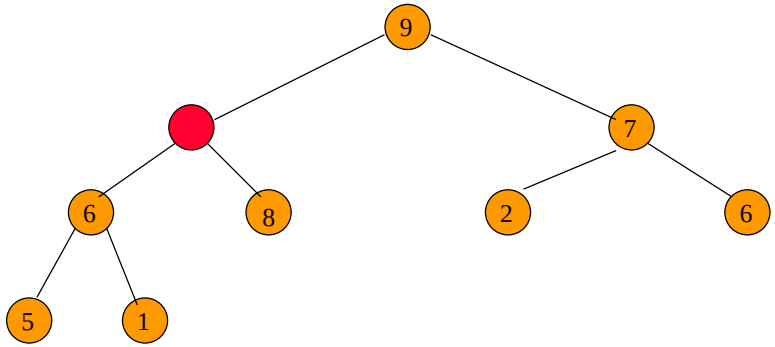
Heap with 9 nodes.

Removing The Max Element



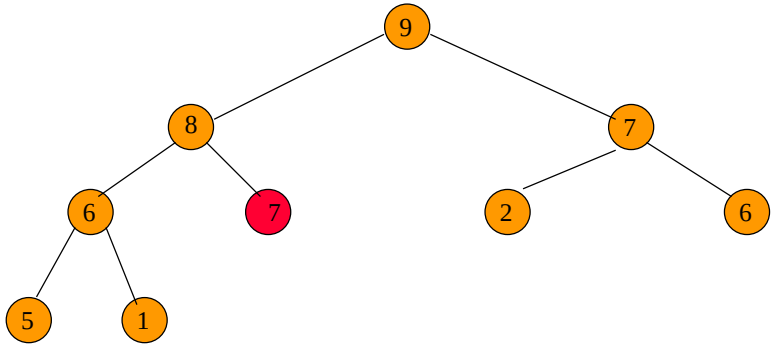
Reinsert 7.

Removing The Max Element



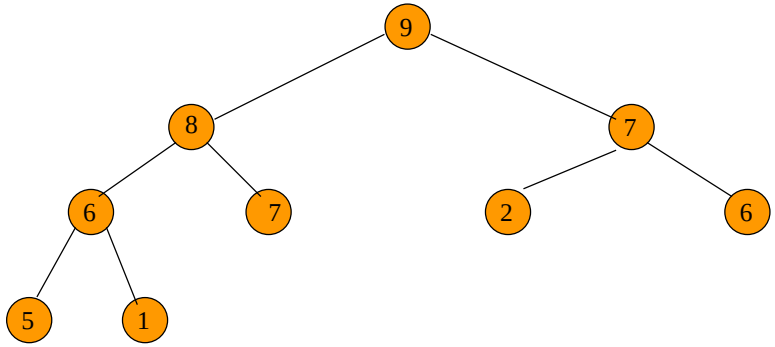
Reinsert 7.

Removing The Max Element



Reinsert 7.

Complexity Of Remove Max Element

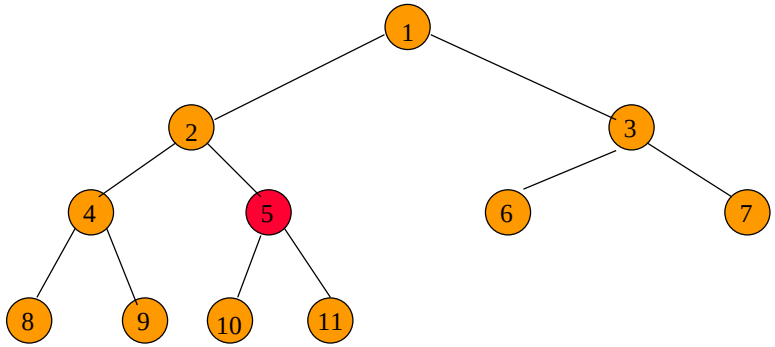


Complexity is O(log n).

In Class Exercise

- Remove the max element 9 from the heap illustrated in last slide.

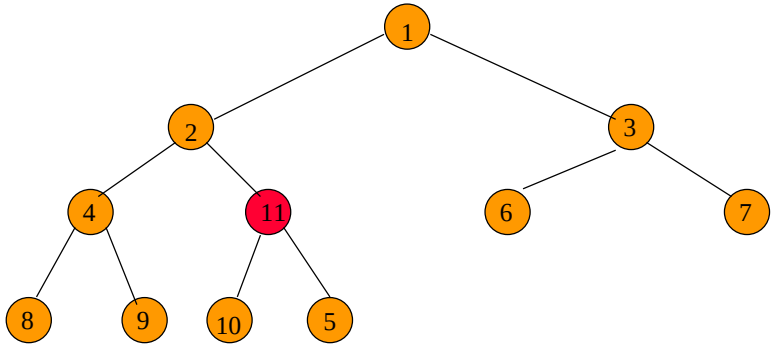
Initializing A Max Heap



Start at rightmost array position that has a child.

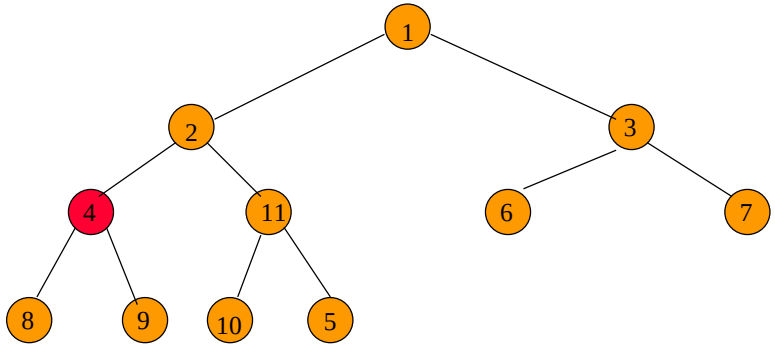
Index is n/2.

Initializing A Max Heap

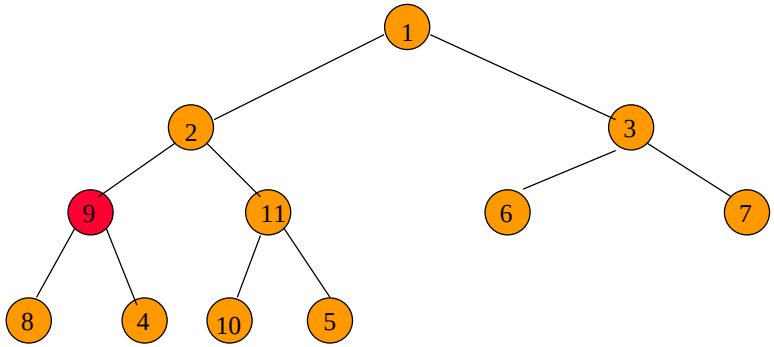


Move to next lower array position.

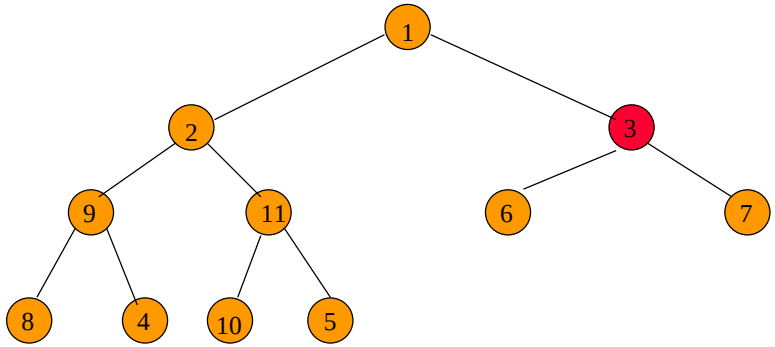
Initializing A Max Heap



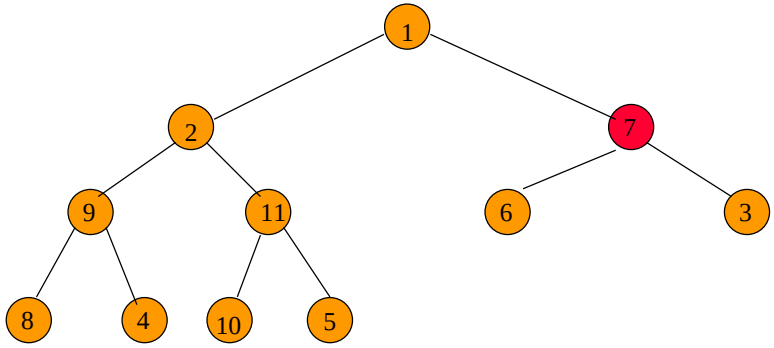
Initializing A Max Heap



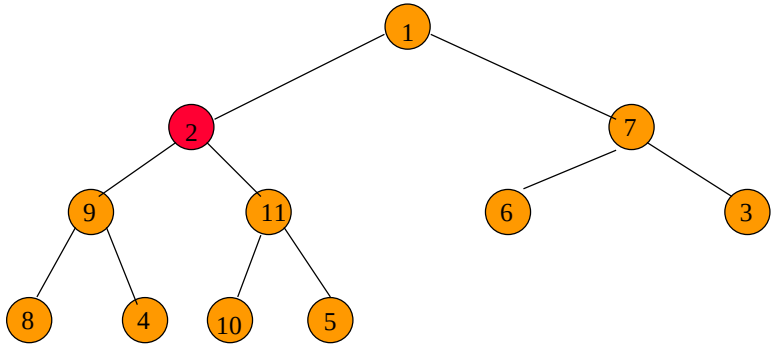
Initializing A Max Heap



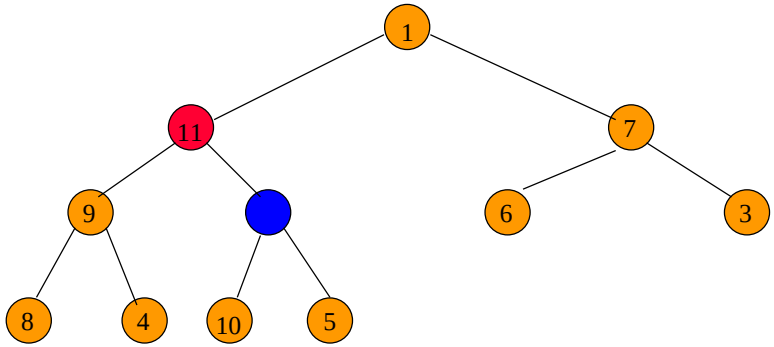
Initializing A Max Heap



Initializing A Max Heap

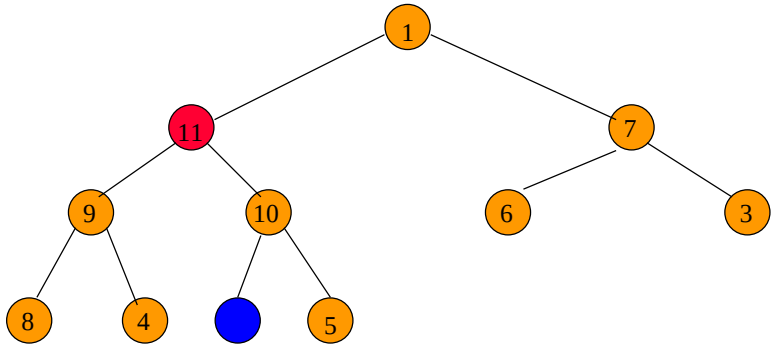


Initializing A Max Heap



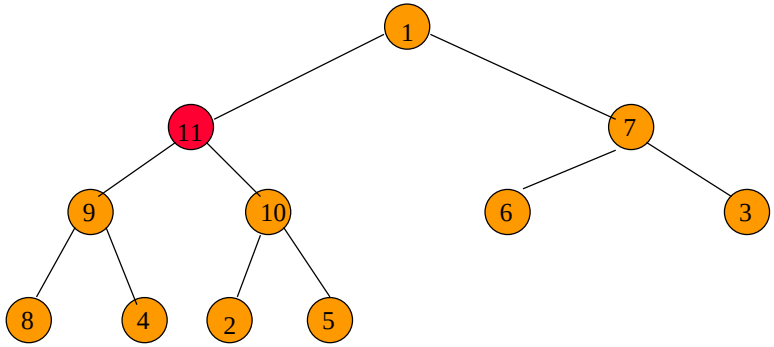
Find a home for 2.

Initializing A Max Heap



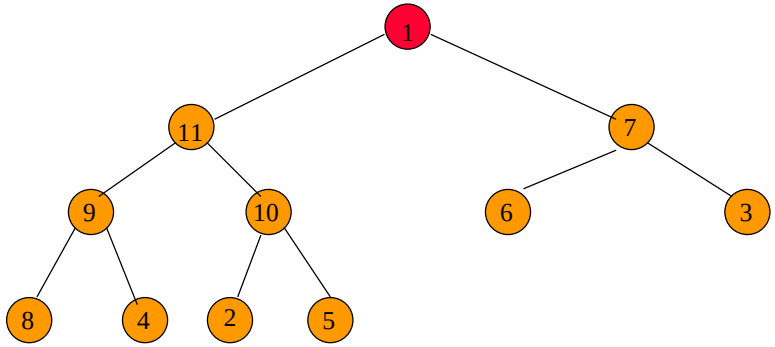
Find a home for 2.

Initializing A Max Heap



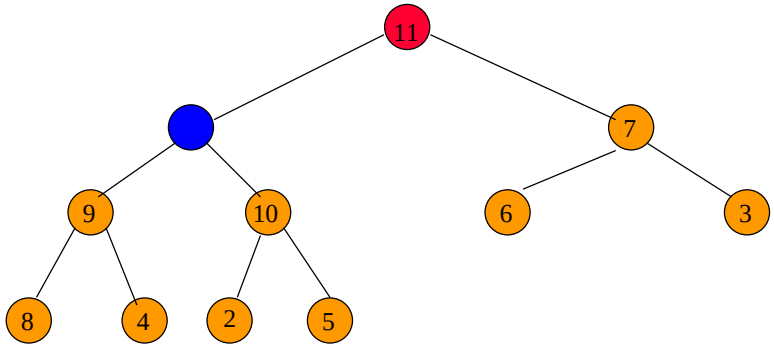
Done, move to next lower array position.

Initializing A Max Heap



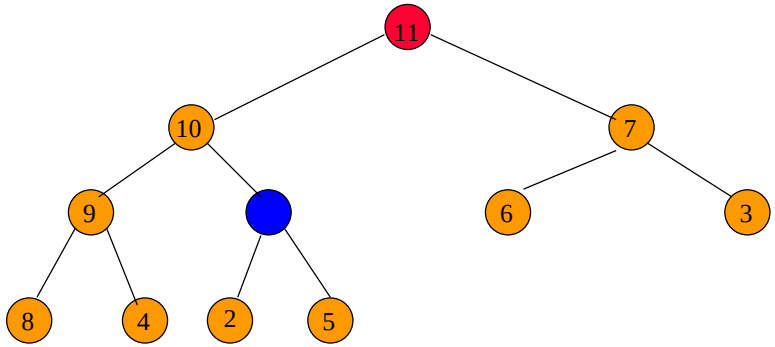
Find home for 1.

Initializing A Max Heap



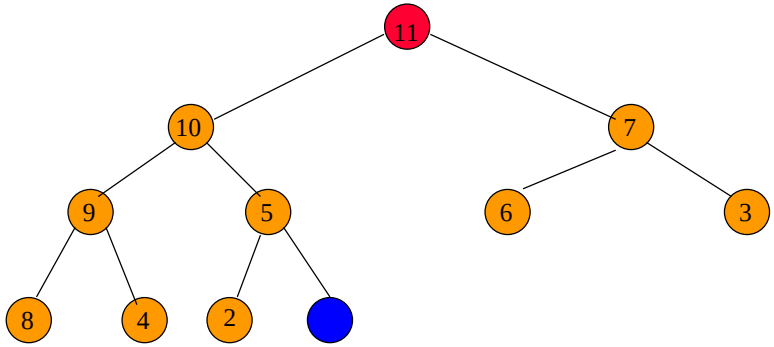
Find home for 1.

Initializing A Max Heap



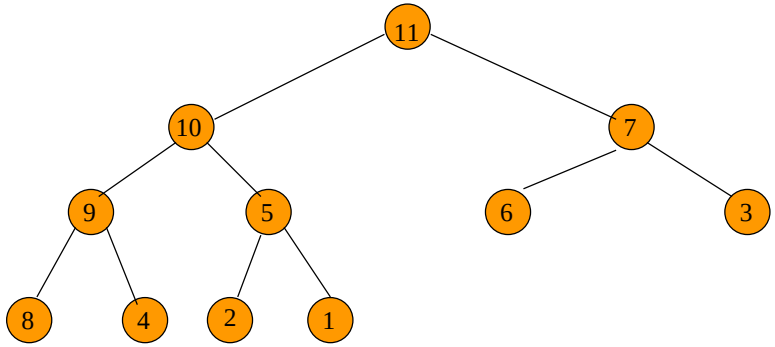
Find home for 1.

Initializing A Max Heap



Find home for 1.

Initializing A Max Heap



Done.

Homework

Implement a Program that Initialize a Max Heap

- Node 1、2、3、4、5、6、7 stored in array
- tip: using recursion