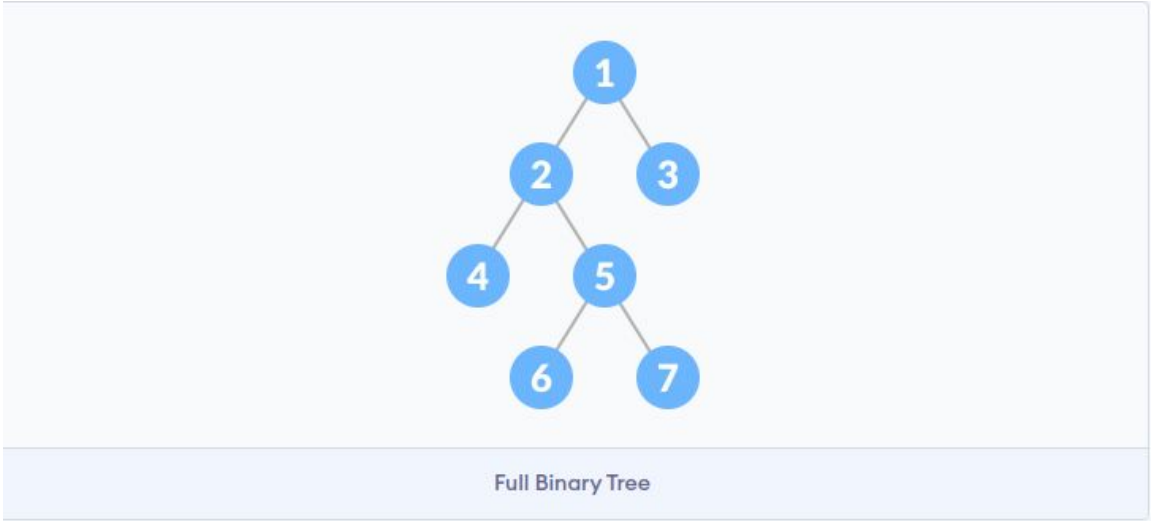
Data Structures

Binary tree, Heap, Priority Queues

Full binary tree

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.

It is also known as a **proper binary tree**.



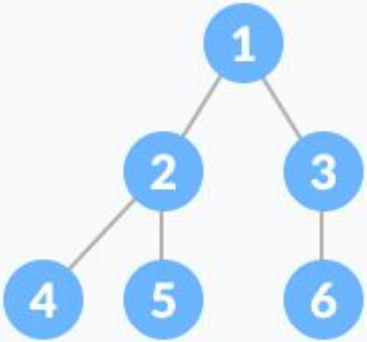
Complete binary tree

A **complete binary tree** is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left.

A complete binary tree is just like a full binary tree, but with two major differences:

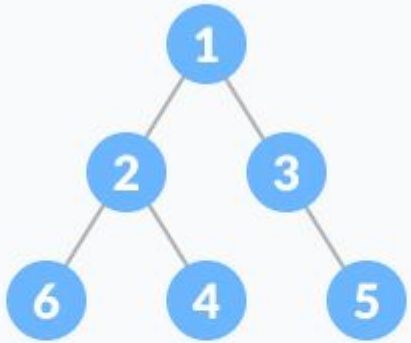
- All the leaf elements must lean towards the left.
- The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.

Complete binary tree



Complete Binary Tree

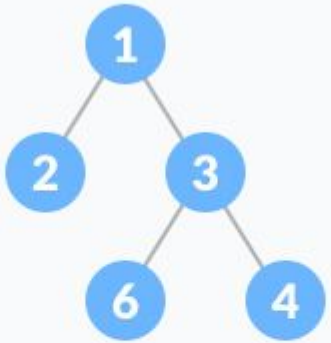
Full Binary Tree vs Complete Binary Tree



✖ Full Binary Tree
✖ Complete Binary Tree

Comparison between full binary tree and complete binary tree

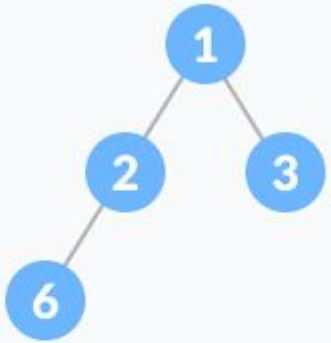
Full Binary Tree vs Complete Binary Tree



✔ Full Binary Tree
✖ Complete Binary Tree

Comparison between full binary tree and complete binary tree

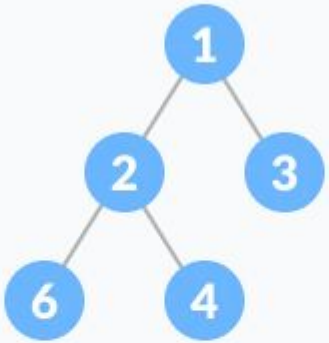
Full Binary Tree vs Complete Binary Tree



✗ Full Binary Tree
✓ Complete Binary Tree

Comparison between full binary tree and complete binary tree

Full Binary Tree vs Complete Binary Tree



✔ Full Binary Tree
✔ Complete Binary Tree

Comparison between full binary tree and complete binary tree

How a complete binary tree is created

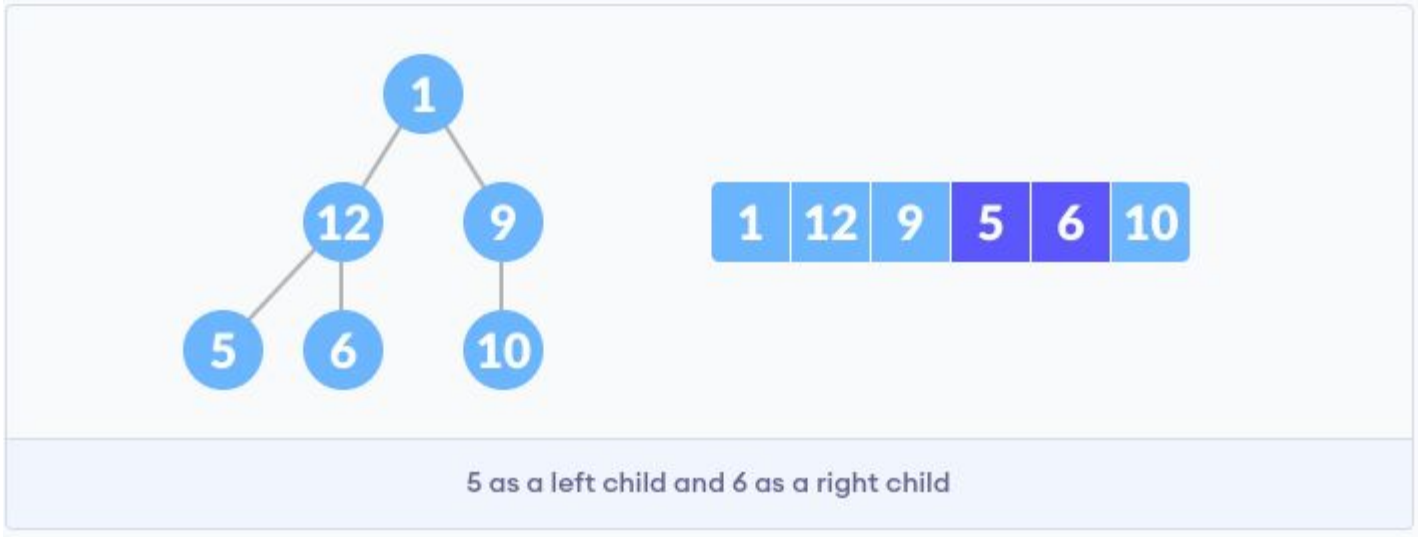
1. Select the first element of the list to be the root node. (no. of elements on level-I: 1)

How a complete binary tree is created

1. Put the second element as a left child of the root node and the third element as the right child. (no. of elements on level-II: 2)

How a complete binary tree is created

1. Put the next two elements as children of the left node of the second level. Again, put the next two elements as children of the right node of the second level (no. of elements on level-III: 4) elements).
2. Keep repeating until you reach the last element.



Relationship between array indexes and tree element

A complete binary tree has an interesting property that we can use to find the children and parents of any node.

If the index of any element in the array is **i**, the element in the index **2i+1** will become the left child and element in **2i+2** index will become the right child. Also, the parent of any element at index **i** is given by the lower bound of **(i-1)/2**.

Heap

Heap data structure is a complete binary tree that satisfies the **heap property**, where any given node is:

- always greater than its child node/s and the key of the root node is the largest among all other nodes. This property is also called **max heap property**.
- always smaller than the child node/s and the key of the root node is the smallest among all other nodes. This property is also called **min heap property**.

Priority Queue

A priority queue is a special type of queue in which each element is associated with a priority value. And, elements are served on the basis of their priority. That is, higher priority elements are served first.

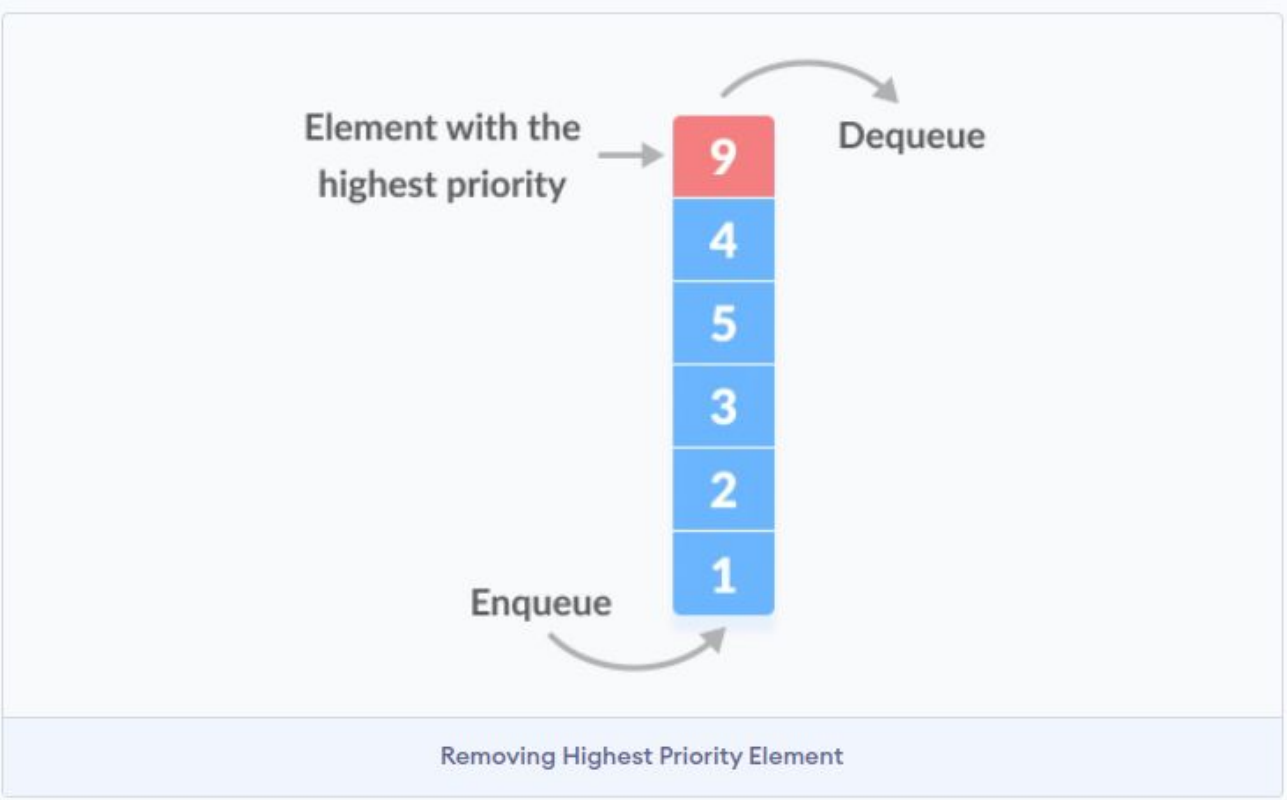
However, if elements with the same priority occur, they are served according to their order in the queue.

Assigning Priority Value

Generally, the value of the element itself is considered for assigning the priority. For example, the element with the highest value is considered the highest priority element. However, in other cases, we can assume the element with the lowest value as the highest priority element.

We can also set priorities according to our needs.

Assigning Priority Value



Removing Highest Priority Element

Difference between Priority Queue and Normal Queue

In a queue, the first-in-first-out rule is implemented whereas, in a priority queue, the values are removed on the basis of priority. The element with the highest priority is removed first.