**Heap**

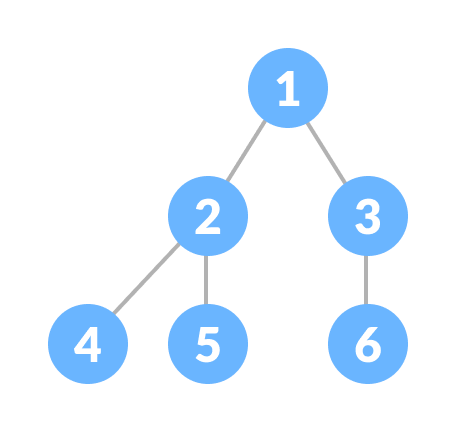
A [Heap](http://www.geeksforgeeks.org/binary-heap/) is a special [Tree-based data structure](https://www.geeksforgeeks.org/binary-tree-data-structure/) in which the tree is a [complete binary tree](http://www.geeksforgeeks.org/binary-tree-set-3-types-of-binary-tree/). Since a heap is a complete binary tree, a heap with **N** nodes has **log N** height. It is useful to remove the highest or lowest priority element. It is typically represented as an [array](https://www.geeksforgeeks.org/array-representation-of-binary-heap/). There are two types of Heaps in the [data structure](https://www.geeksforgeeks.org/data-structures/).

[**Complete binary tree**](http://www.geeksforgeeks.org/binary-tree-set-3-types-of-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

1. All the leaf elements must lean towards the left.
2. 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

**Min-Heap**

In a [Min-Heap](https://www.geeksforgeeks.org/min-heap-in-java/) the key present at the root node must be less than or equal among the keys present at all of its children. The same property must be recursively true for all sub-trees in that Binary Tree. In a Min-Heap the minimum key element present at the root. Below is the Binary Tree that satisfies all the property of Min Heap.

[Timeline

Description automatically generated](https://media.geeksforgeeks.org/wp-content/uploads/20201106115157/MinHeap.jpg)

**Step 1** − Remove root node.

**Step 2** − Move the last element of last level to root.

**Step 3** − Compare the value of this child node with its parent.

**Step 4** − If value of parent is less than child, then swap them.

**Step 5** − Repeat step 3 & 4 until Heap property holds.

**Max Heap**

In a [Max-Heap](https://www.geeksforgeeks.org/max-heap-in-java/) the key present at the root node must be greater than or equal among the keys present at all of its children. The same property must be [recursively](https://www.geeksforgeeks.org/recursion/) **true** for all sub-trees in that Binary Tree. In a Max-Heap the maximum key element present at the root. Below is the Binary Tree that satisfies all the property of Min Heap.

[Timeline

Description automatically generated](https://media.geeksforgeeks.org/wp-content/uploads/20201106115254/MaxHeap.jpg)

**Step 1** − Create a new node at the end of heap.

**Step 2** − Assign new value to the node.

**Step 3** − Compare the value of this child node with its parent.

**Step 4** − If value of parent is less than child, then swap them.

**Step 5** − Repeat step 3 & 4 until Heap property holds.

**Difference between Min Heap and Max Heap**

|  | **Min Heap** | **Max Heap** |
| --- | --- | --- |
| 1. | In a Min-Heap the key present at the root node must be less than or equal to among the keys present at all of its children. | In a Max-Heap the key present at the root node must be greater than or equal to among the keys present at all of its children. |
| 2. | In a Min-Heap the minimum key element present at the root. | In a Max-Heap the maximum key element present at the root. |
| 3. | A Min-Heap uses the ascending priority. | A Max-Heap uses the descending priority. |
| 4. | In the construction of a Min-Heap, the smallest element has priority. | In the construction of a Max-Heap, the largest element has priority. |
| 5. | In a Min-Heap, the smallest element is the first to be popped from the heap. | In a Max-Heap, the largest element is the first to be popped from the heap. |