**Heaps**

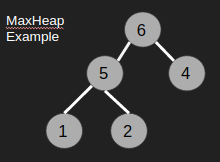
* Heaps are binary tree based data structures.
* Heaps are not necessarily BST (Binary Search Tree)

Heaps are of two types

1. MaxHeap
2. MinHeap

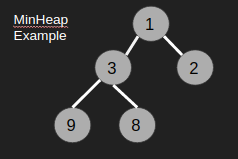
**MaxHeap**

MaxHeap is a heap in which element present at any node is greater than all the elements present in its subtree.



**MinHeap**

MinHeap is a heap in which element present at any node is smaller than all the elements present in its subtree.



**Converting Array into a MaxHeap**

Given an array,



Steps

1. Iterate over each element and **insert** it into the MaxHeap.

Now we will see how to insert elements into a MaxHeap

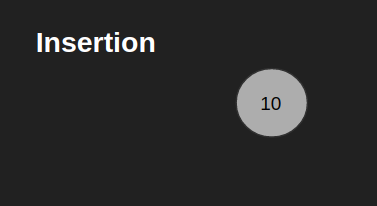
Insertion into a MaxHeap

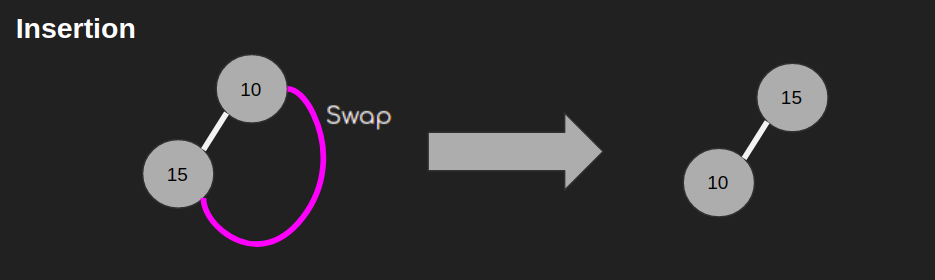
Initially there were no elements in the MaxHeap, so make a newNode in the MaxHeap. Then for each incoming element, connect it to its position and then heapify.

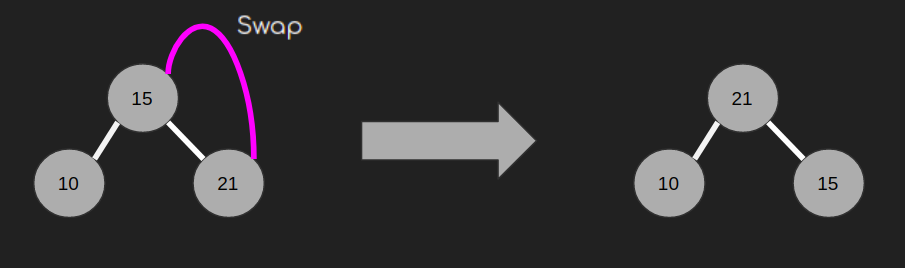
Heapification: Compare the currently inserted element with its parent, there will be 2 cases

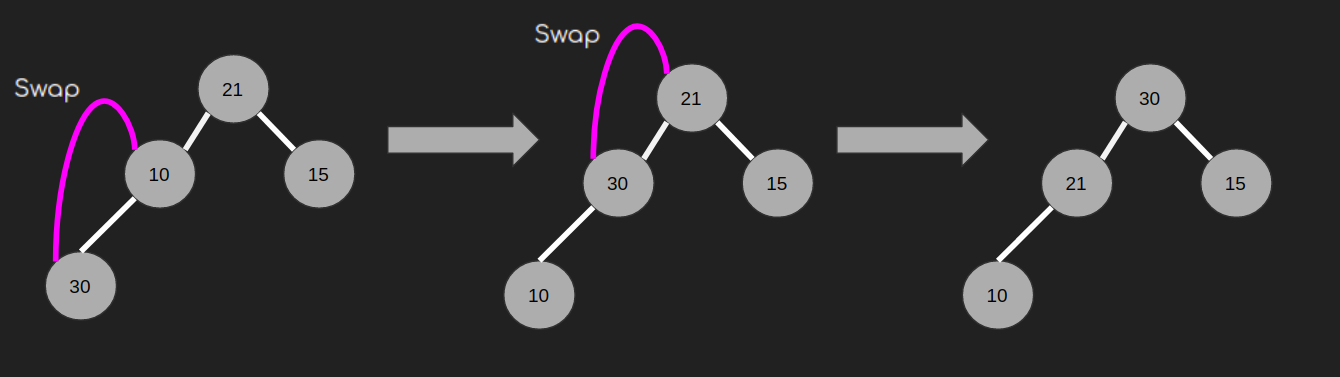
1. Current element > Parent element : Swap the current element and the parent element.
2. Current element <= Parent element: Keep it as it is.

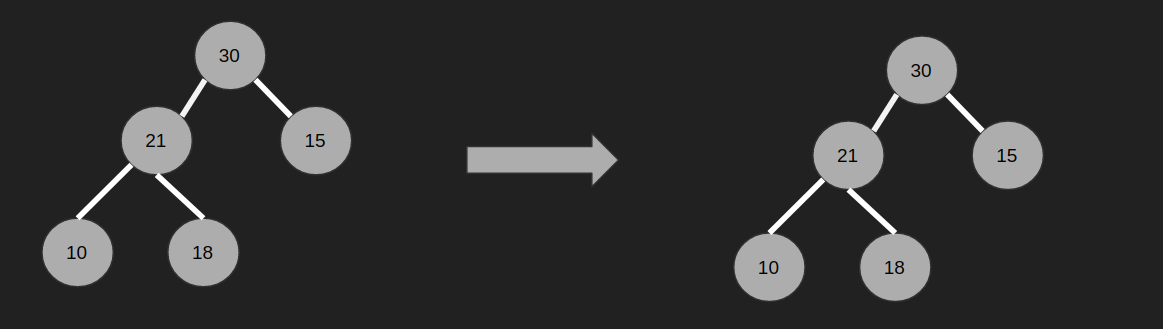
Iterations(insertions)

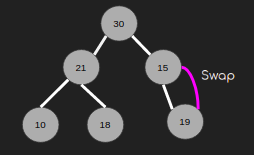
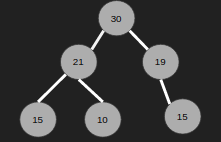






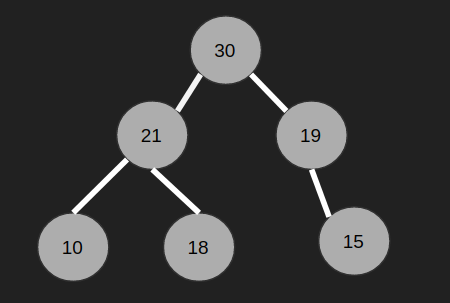




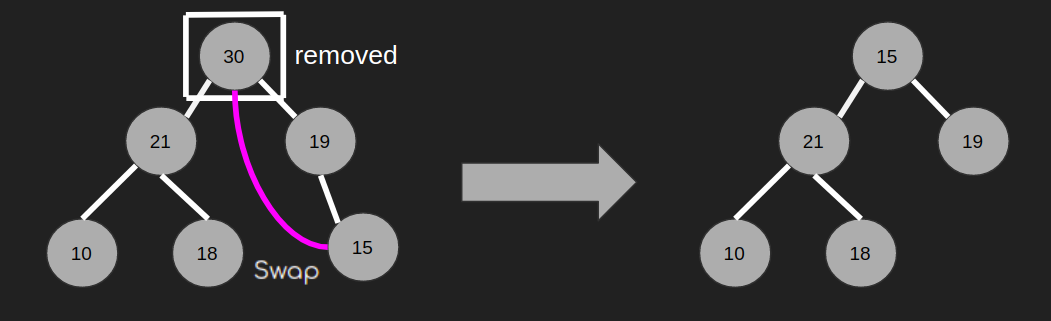
 

**Popping Elements from the MaxHeap**

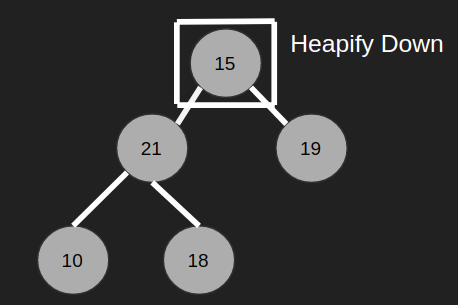
1. Given the MaxHeap



1. Remove the top element and bring the last element to the position of top element.



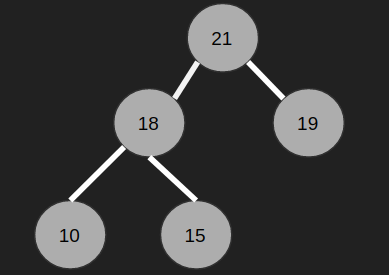
1. Apply heapify down on the top element.



* 1. Heapify Down: In case of MaxHeap, compare the current element with the greater child, there will be 2 cases
     1. Current element < Greater child : Swap current element with greater child.
     2. Current element >= Greater child: Keep the heap as it is.

Do this step until we reach our (ii) case.

1. After performing step 3, we will get our MaxHeap after popping the element.



Code





