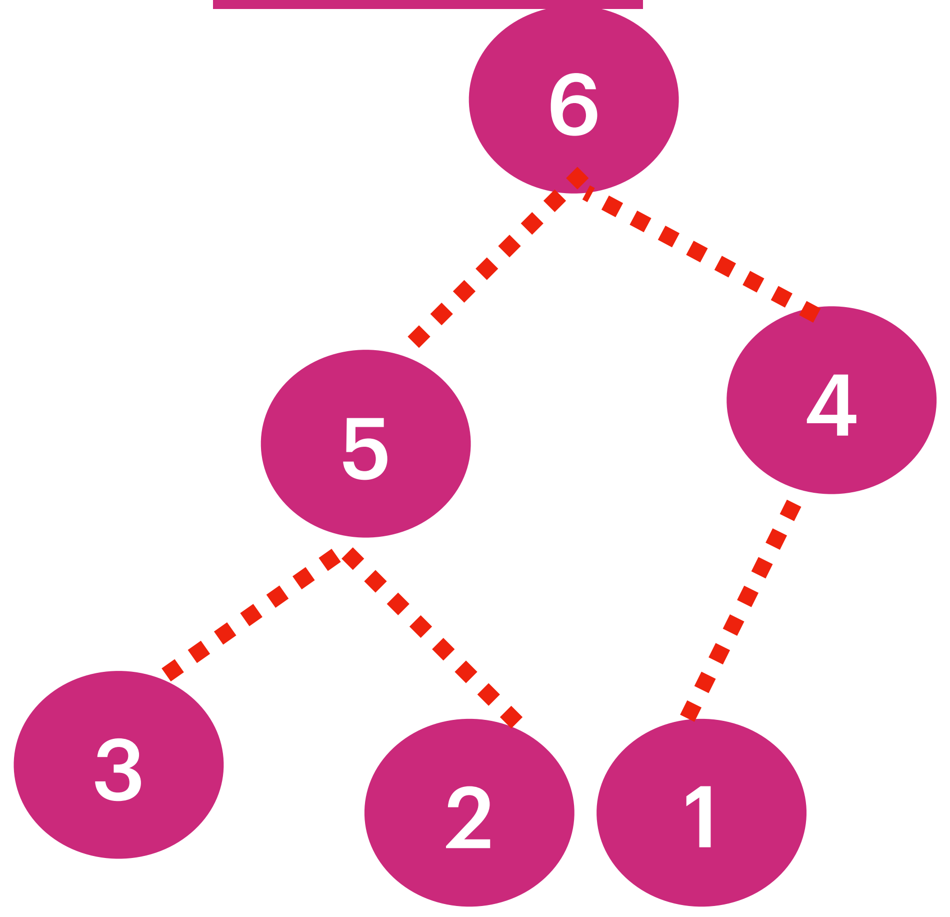
Max Heap

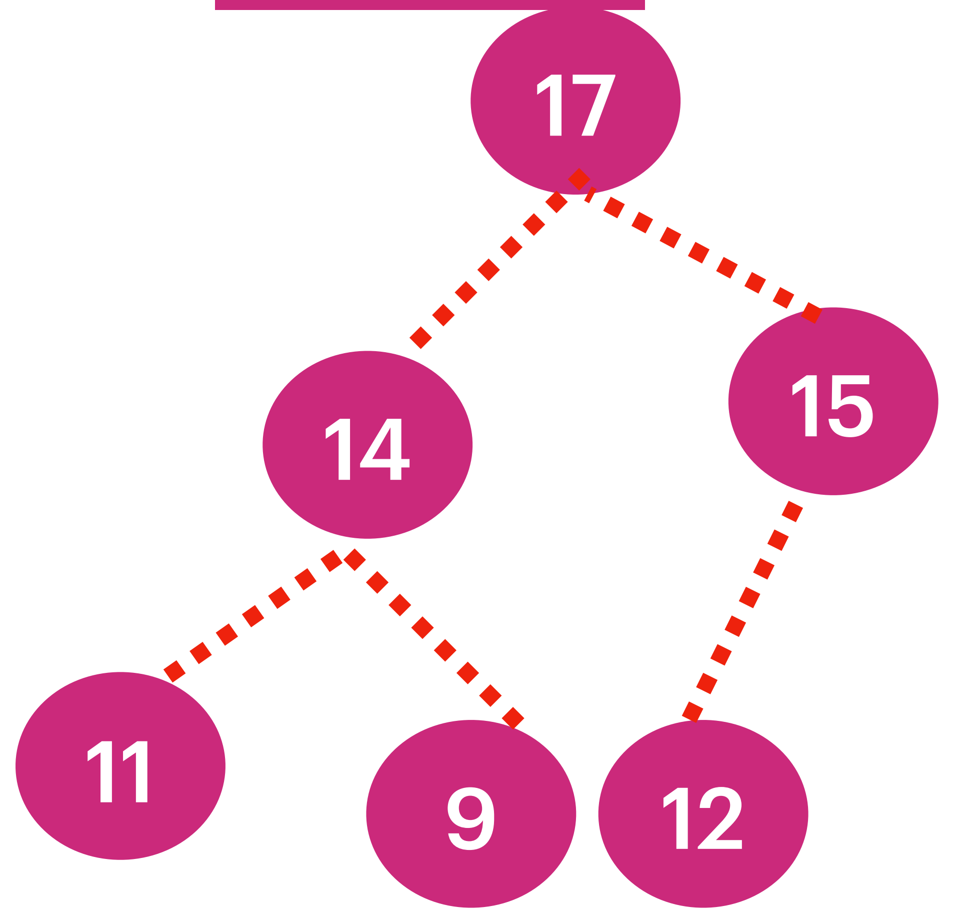
** The Data Structure should Complete Binary Tree

** Each Parent Node should be >= it child node’s

Max Heap



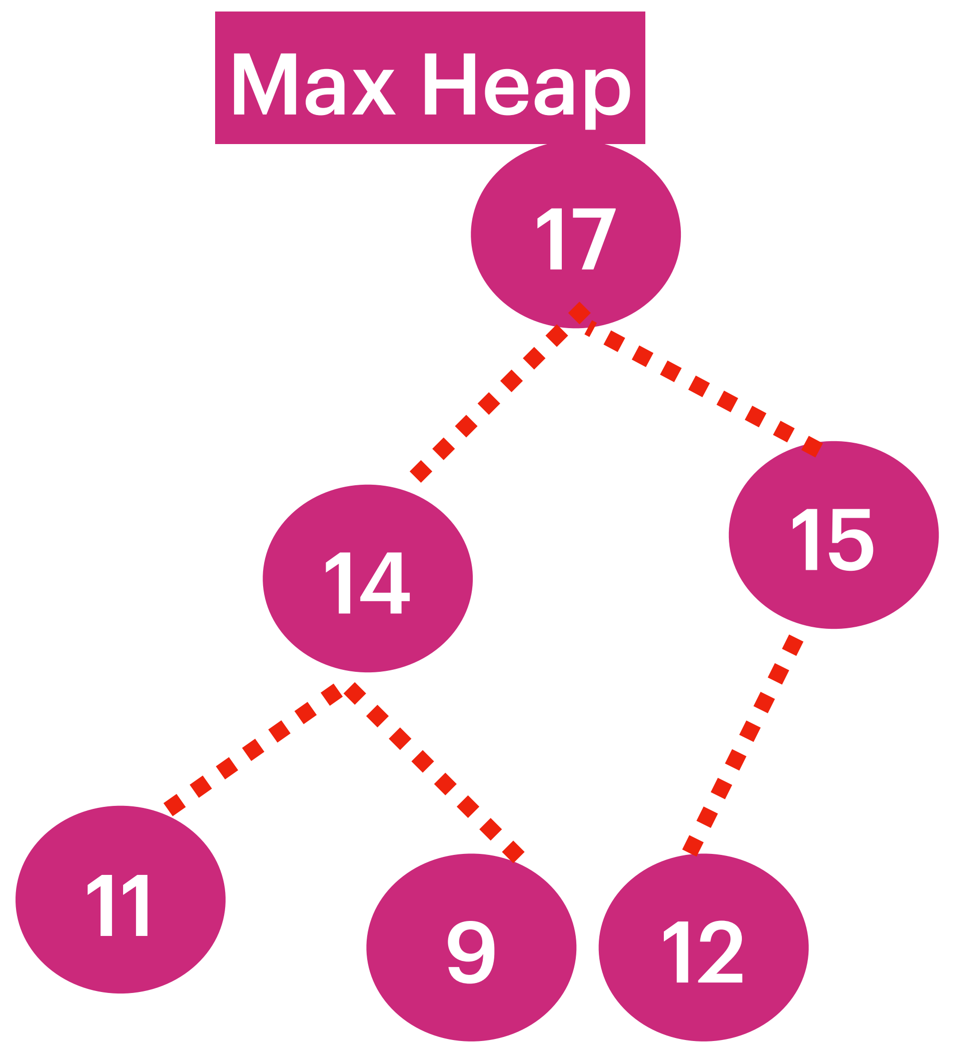
Max Heap

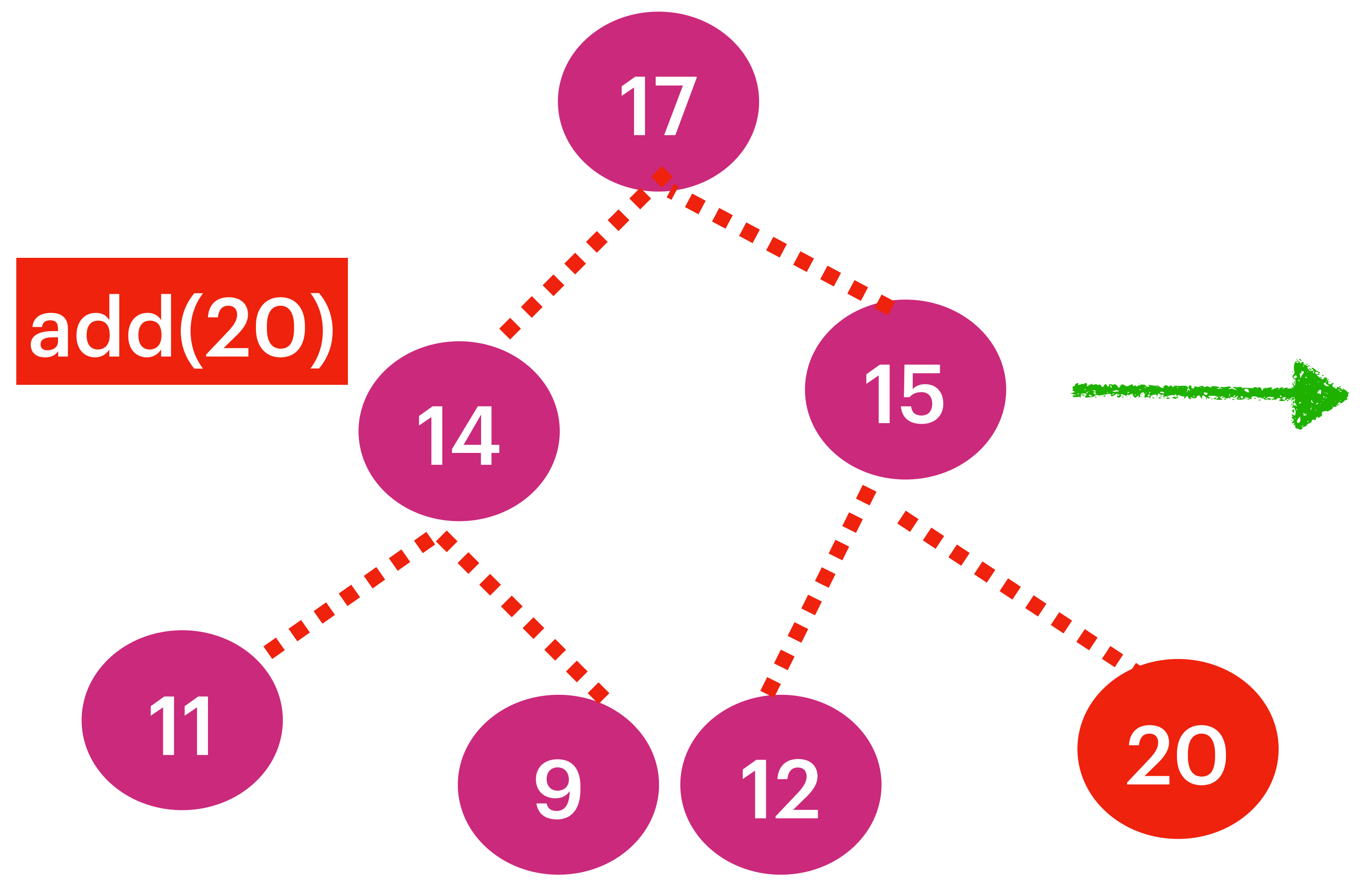


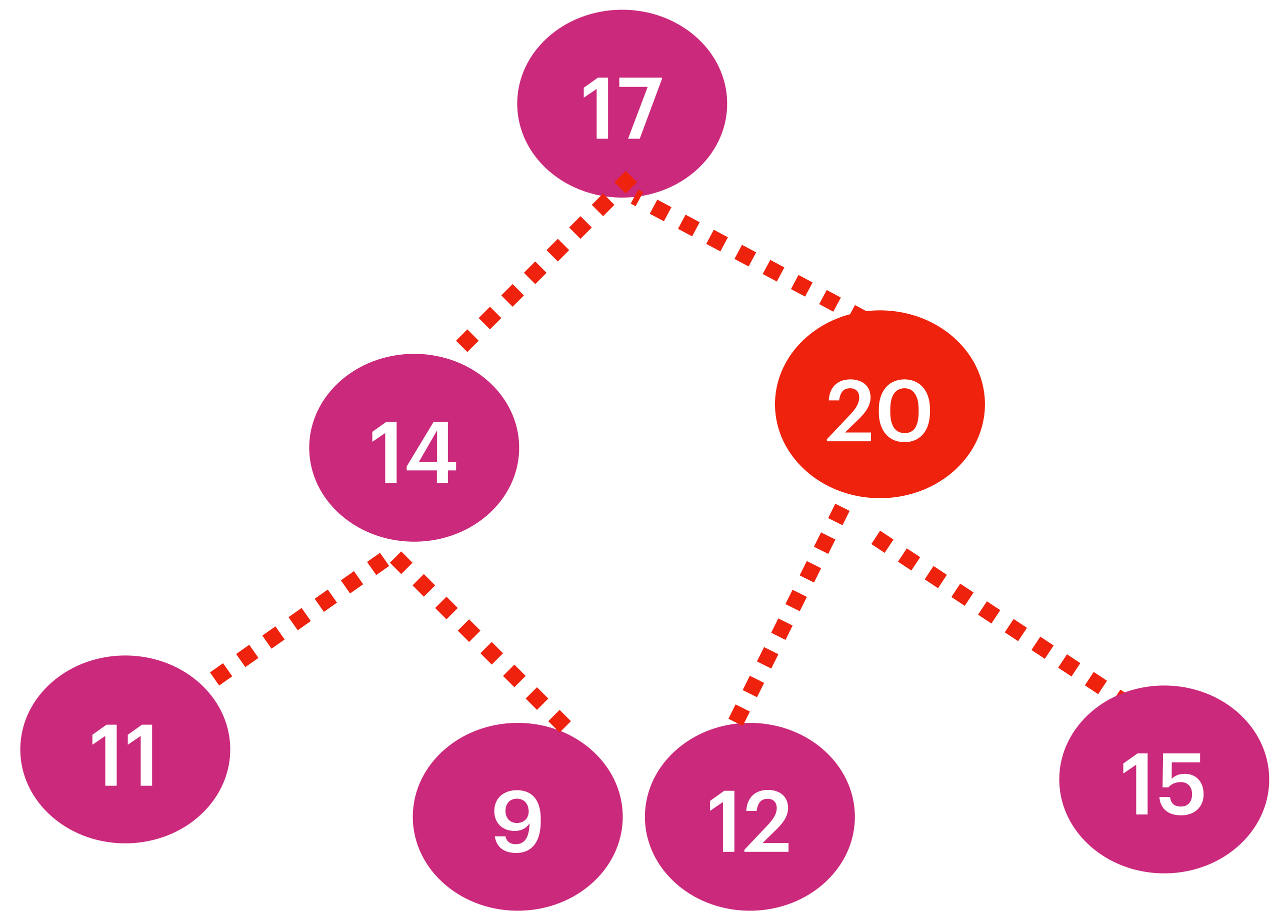
Max Heap

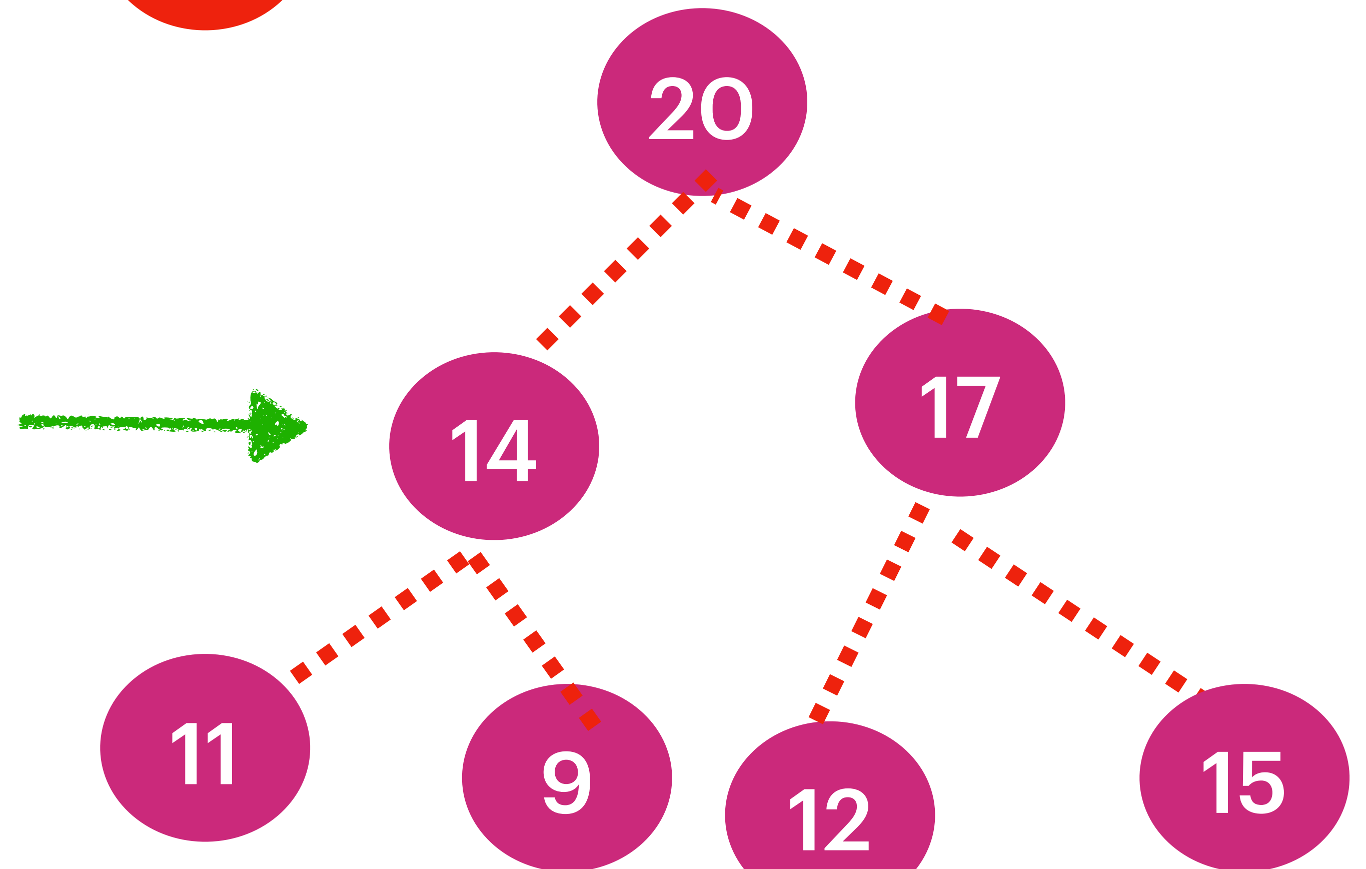
** The Data Structure should Complete Binary Tree

** Each Parent Node should be >= it child node's









Time Complexity : O(logn)
Space Complexity : O(1)

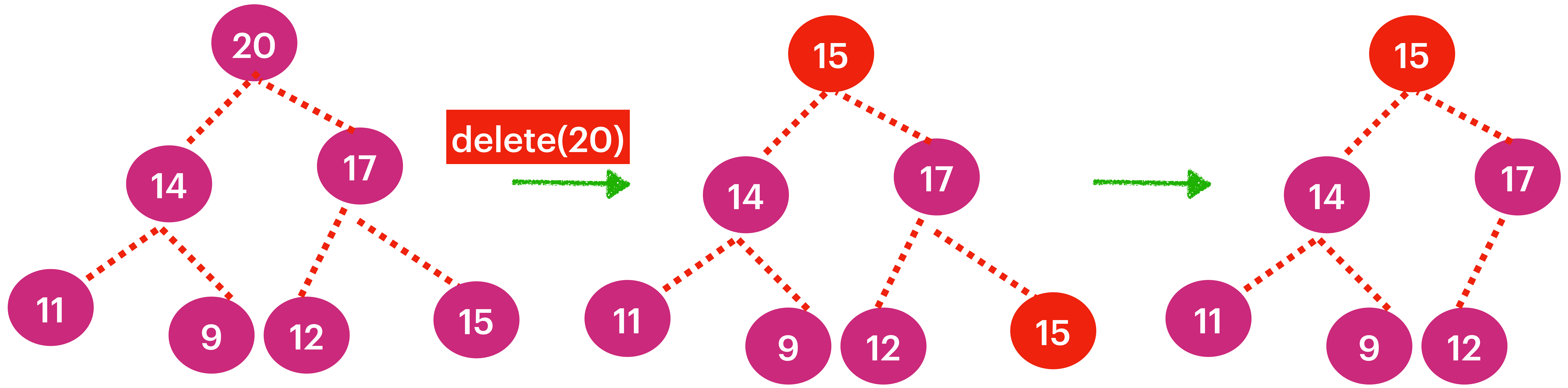
Algorithm :

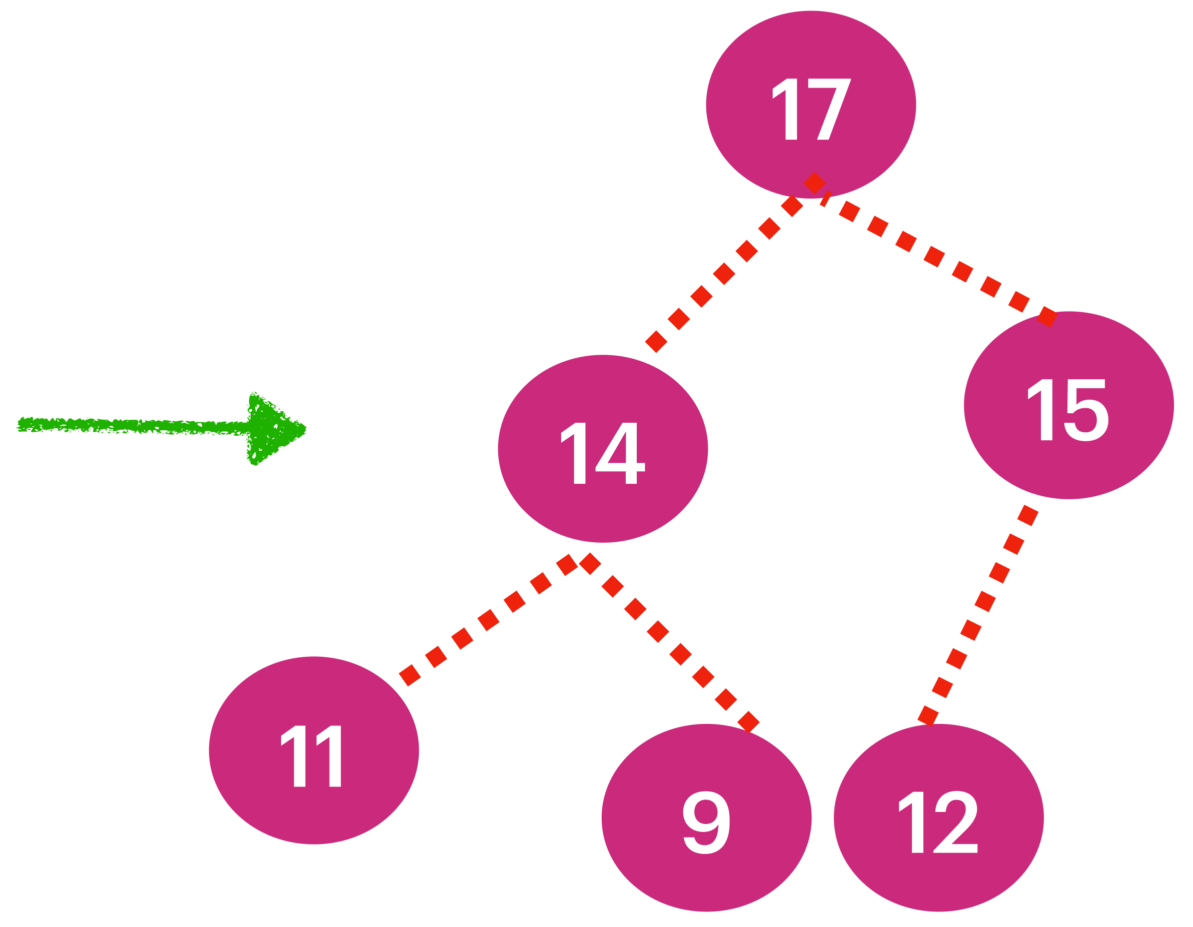
Add to the RightMost position.
If the current element >parent then swap.

Max Heap

** The Data Structure should Complete Binary Tree

** Each Parent Node should be >= it child node's





Time Complexity : O(logn)
Space Complexity : O(1)

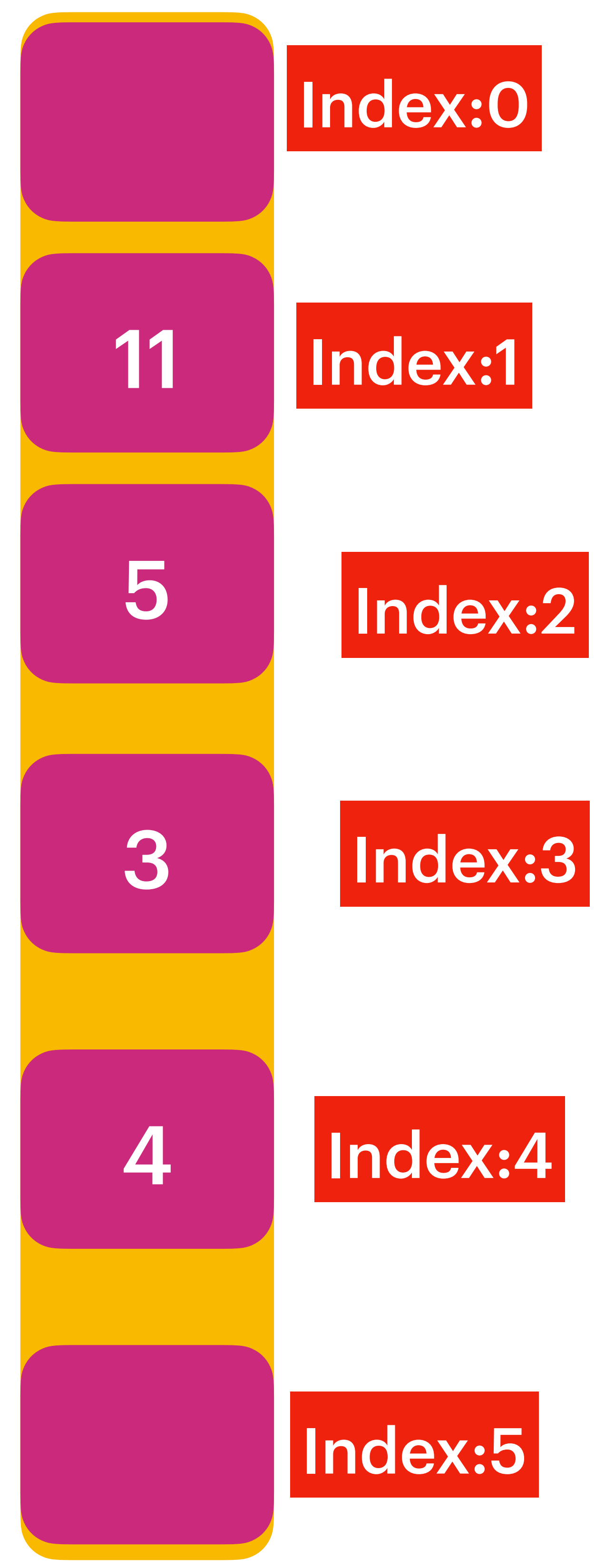
Algorithm :

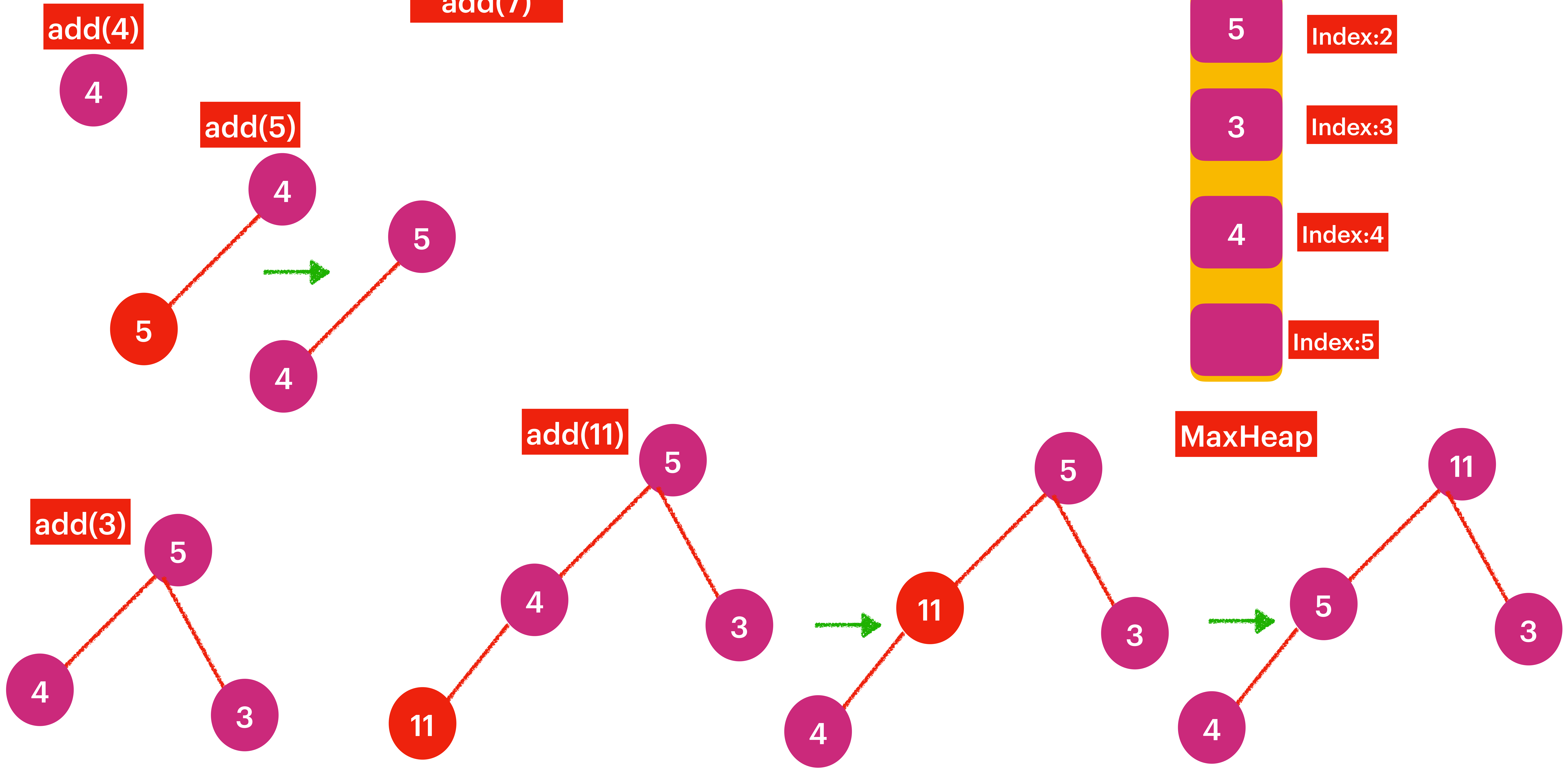
Replace with the right most element ,
Delete the right most element.
If the current Element less than either (leftNode || rightNode) swap accordingly

HeapSize : 5 [4,5,3,11,7]

add(4)
add(5)
add(3)
add(11)
add(7)

Parent = index/2 ;
LeftChild = 2 * index
RightChild = 2 * index + 1
leafNode > realSize/2



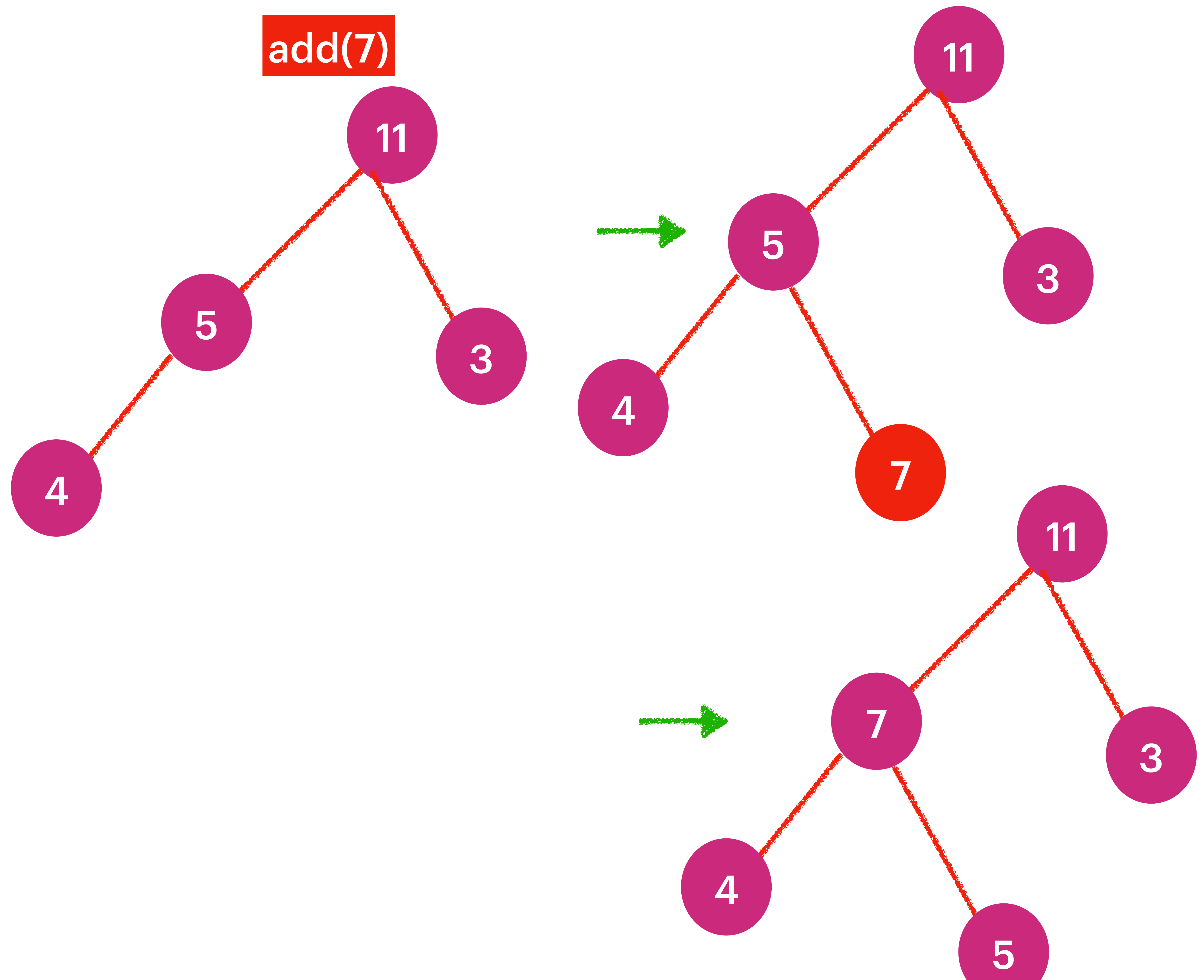


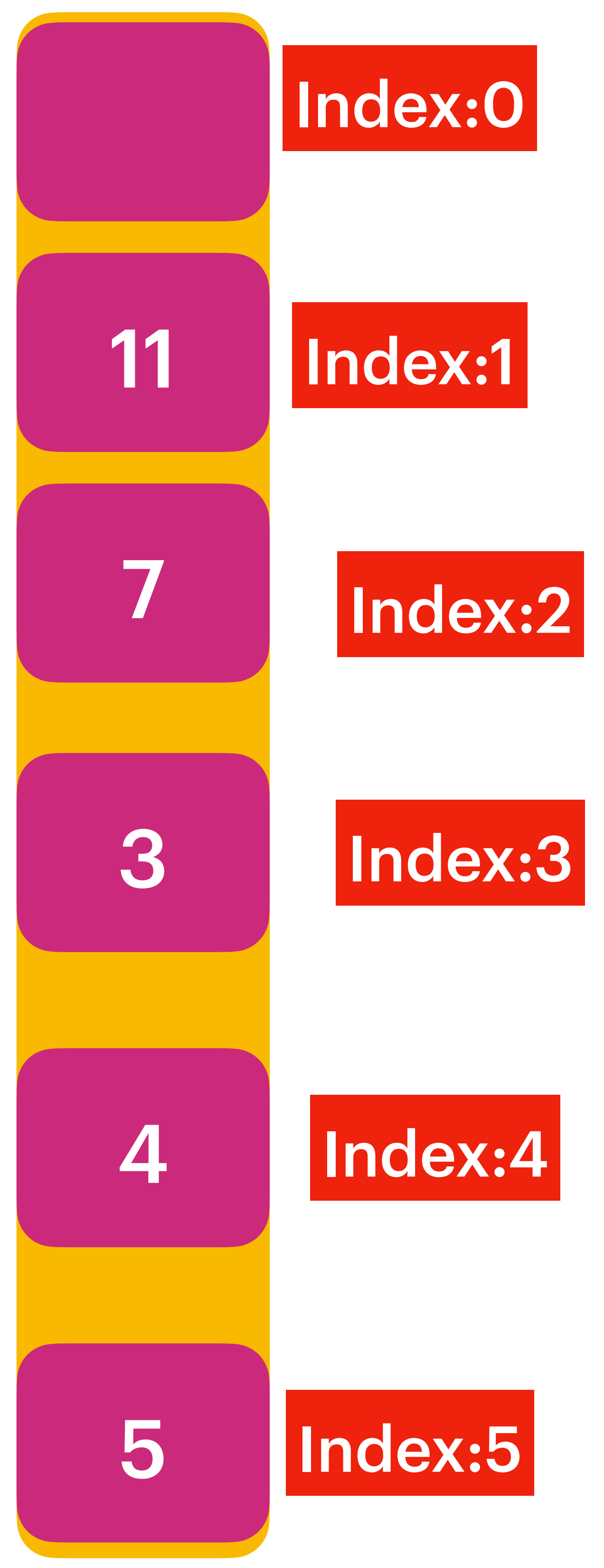
HeapSize : 5 [4,5,3,11,7]

add(4)
add(5)
add(3)
add(11)
add(7)

Parent = index/2 ;
LeftChild = 2 * index
RightChild = 2 * index + 1
leafNode > realSize/2

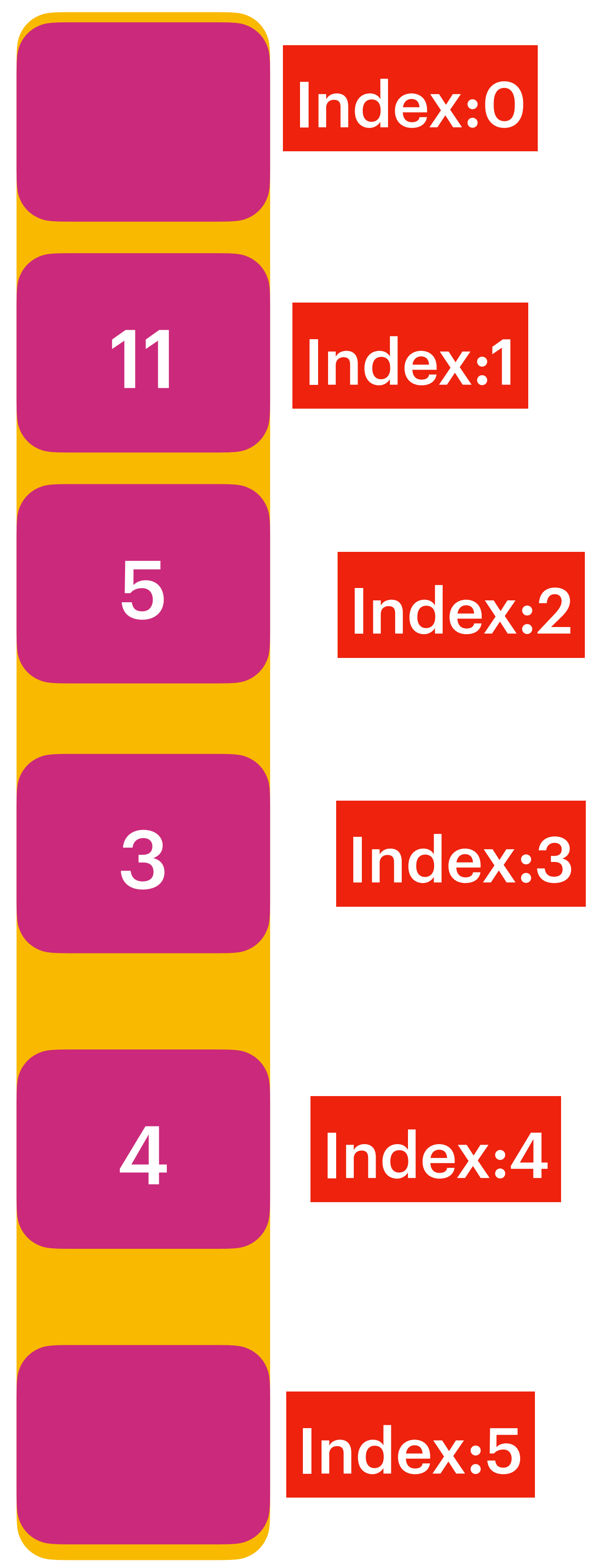
add(7)



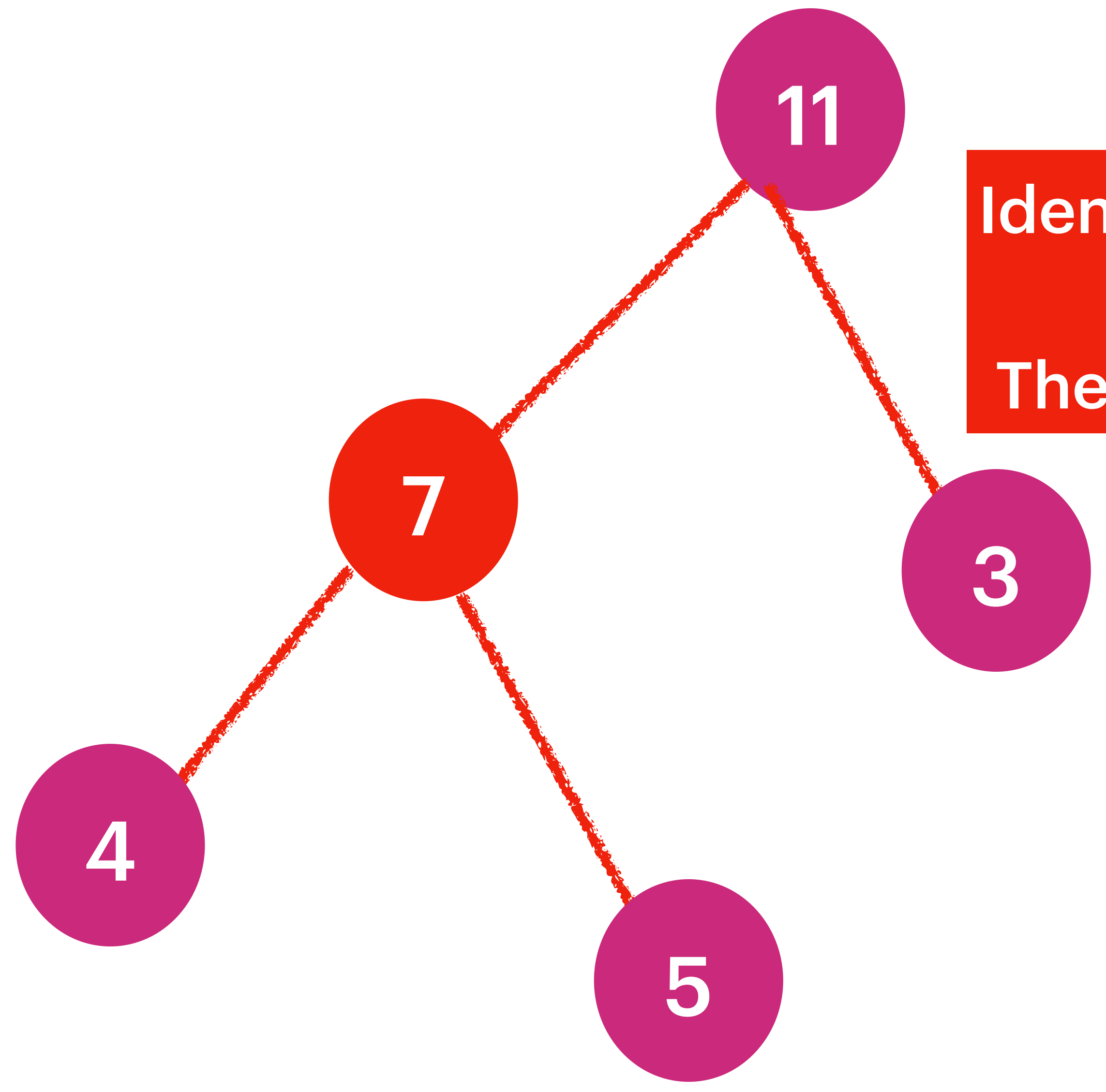


MaxHeap

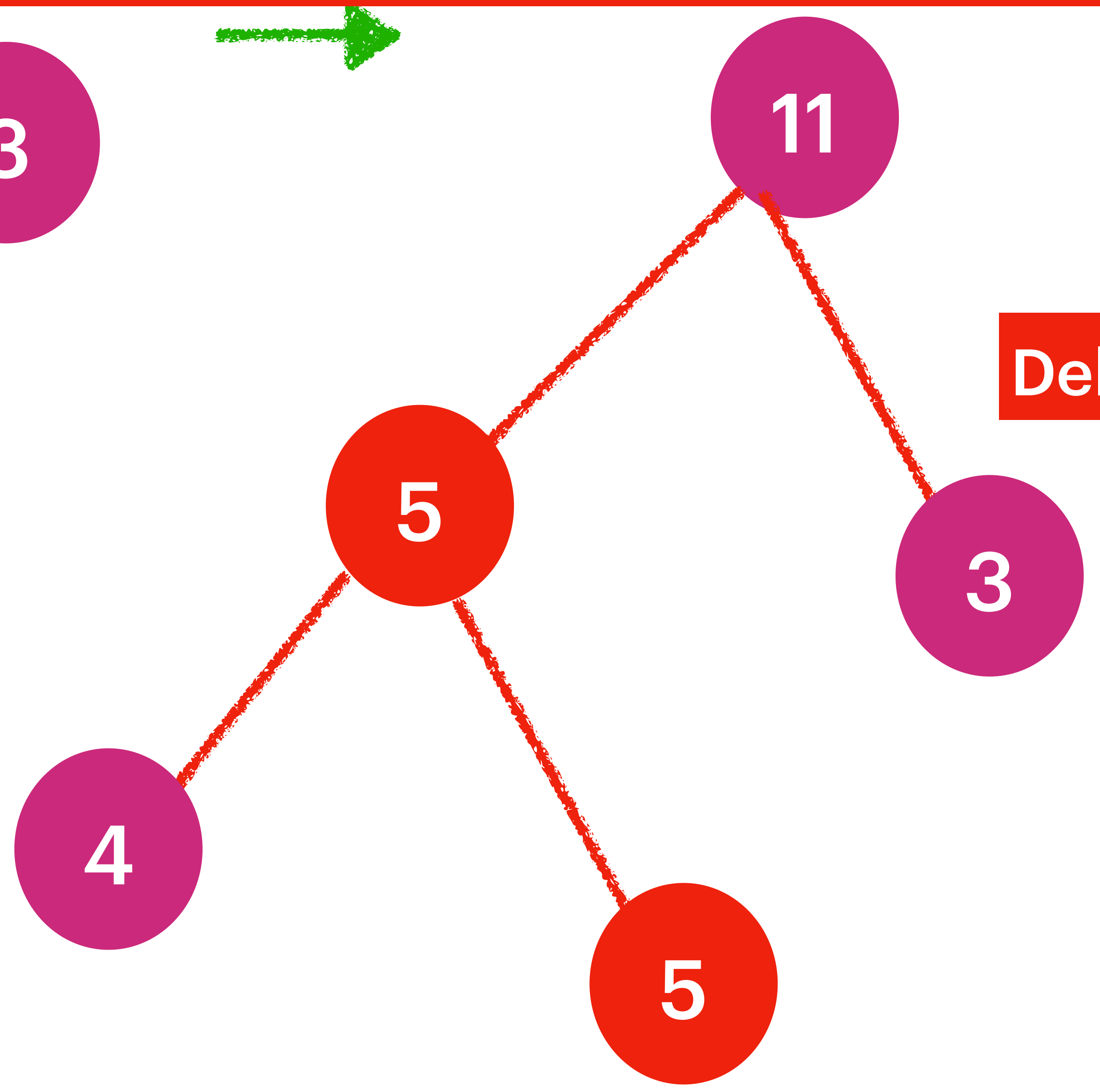
Parent = index/2 ;
LeftChild = 2 * index
RightChild = 2 * index + 1
leafNode > realSize/2



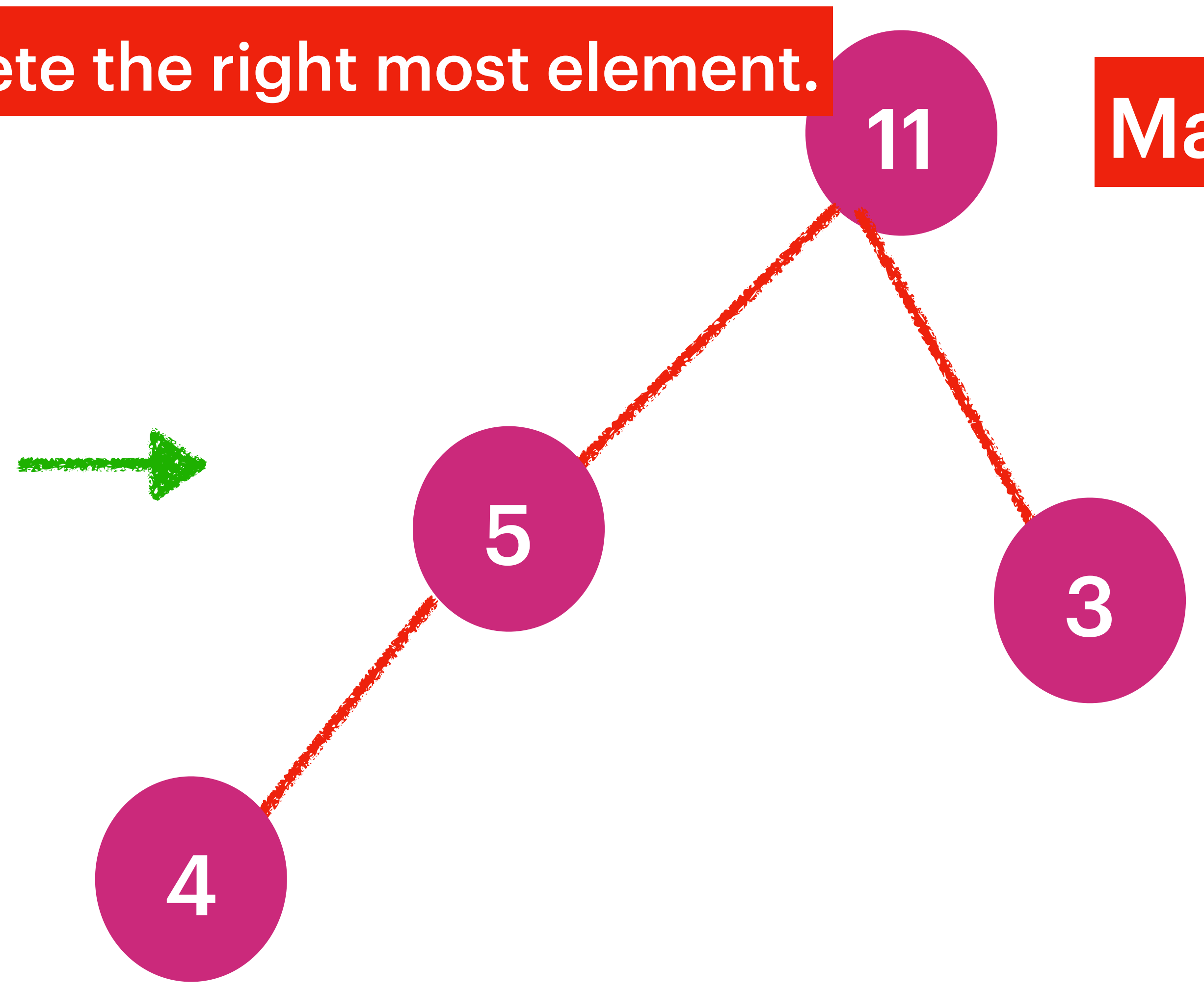
delete(7)



Identify the index of element 7 it is :index:2
O(n) :
Then replace with the right most element.



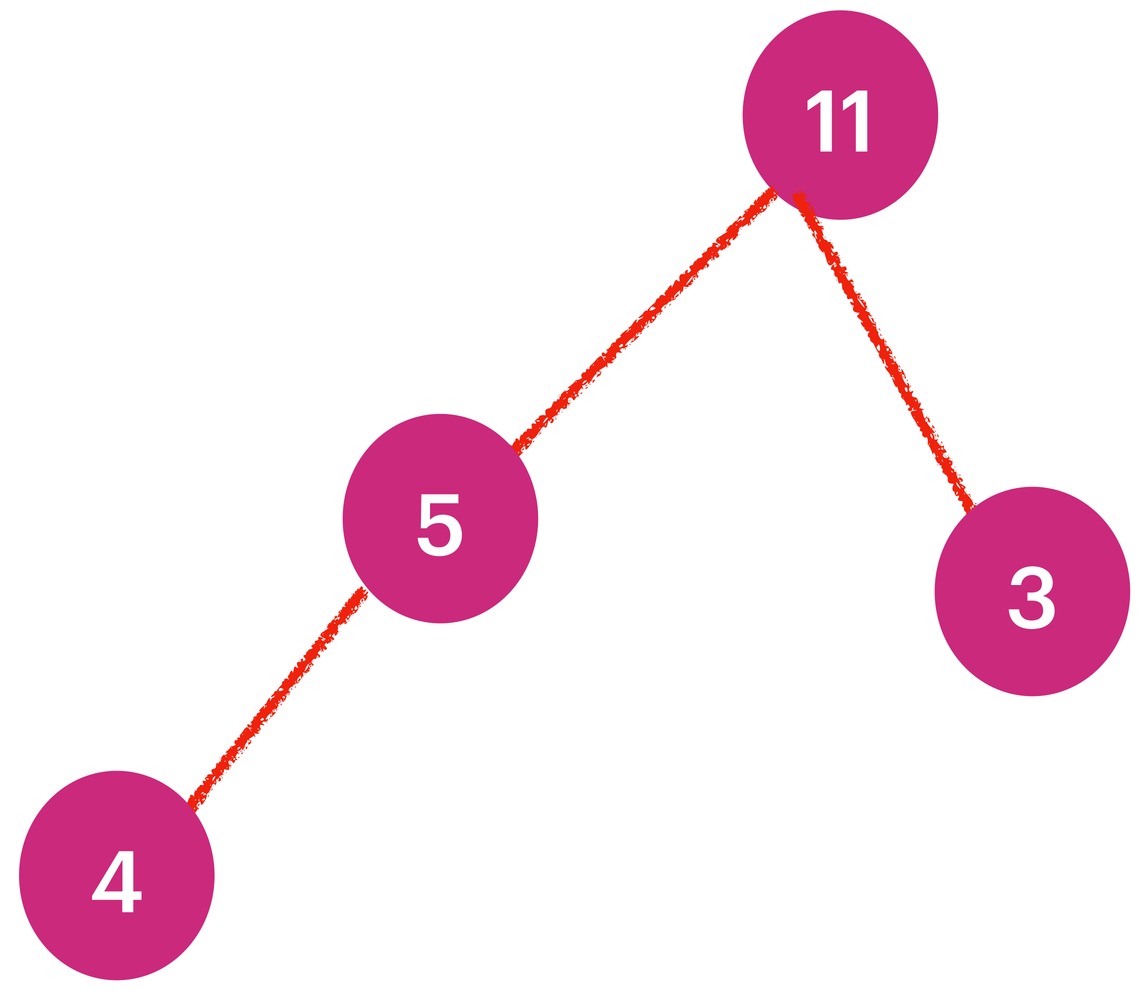
Delete the right most element.



MaxHeap

Parent = index/2 ;
LeftChild = 2 * index
RightChild = 2 * index + 1
leafNode > realSize/2

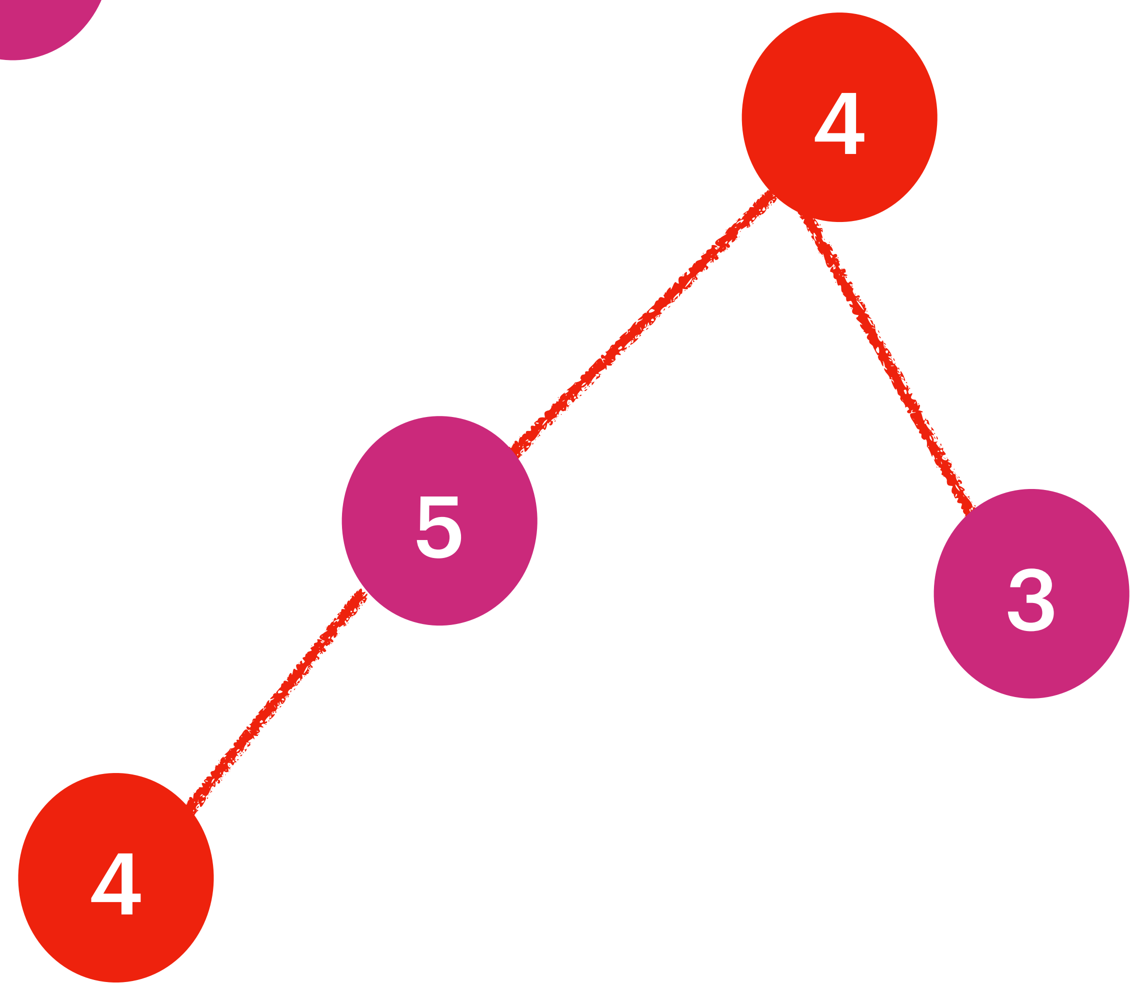
delete(11)

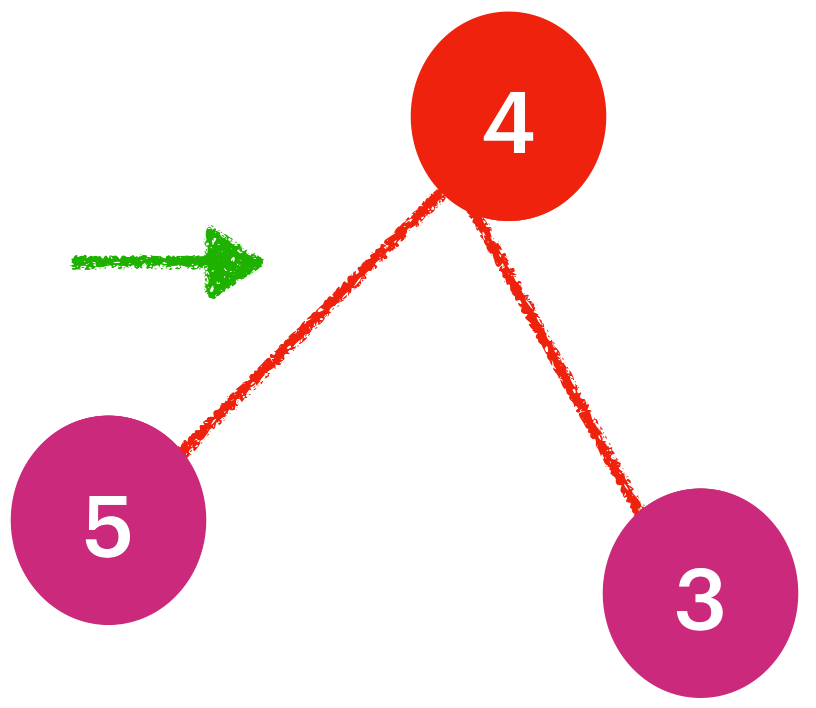


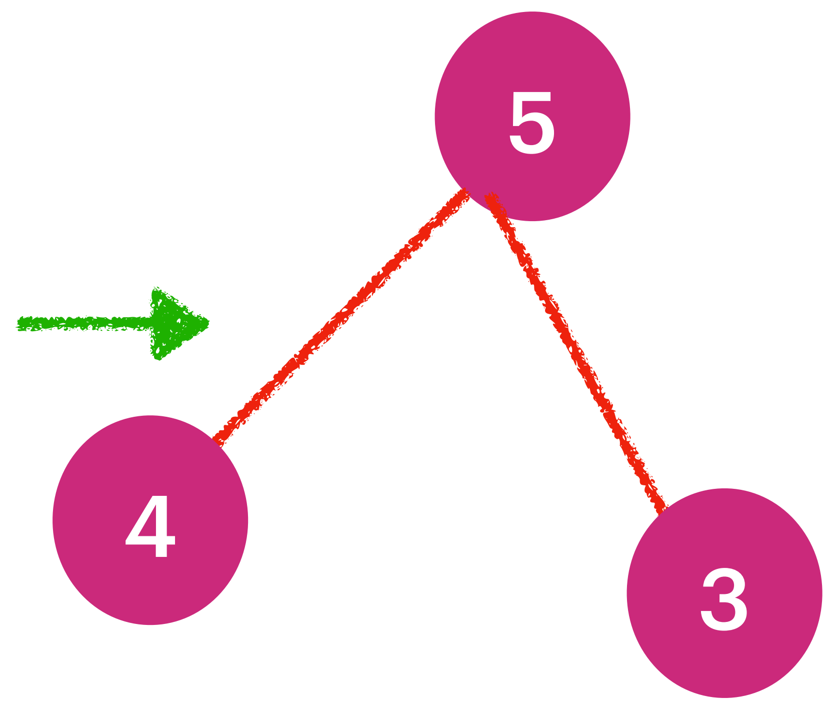
Identify the index of element 11 it is :index:1

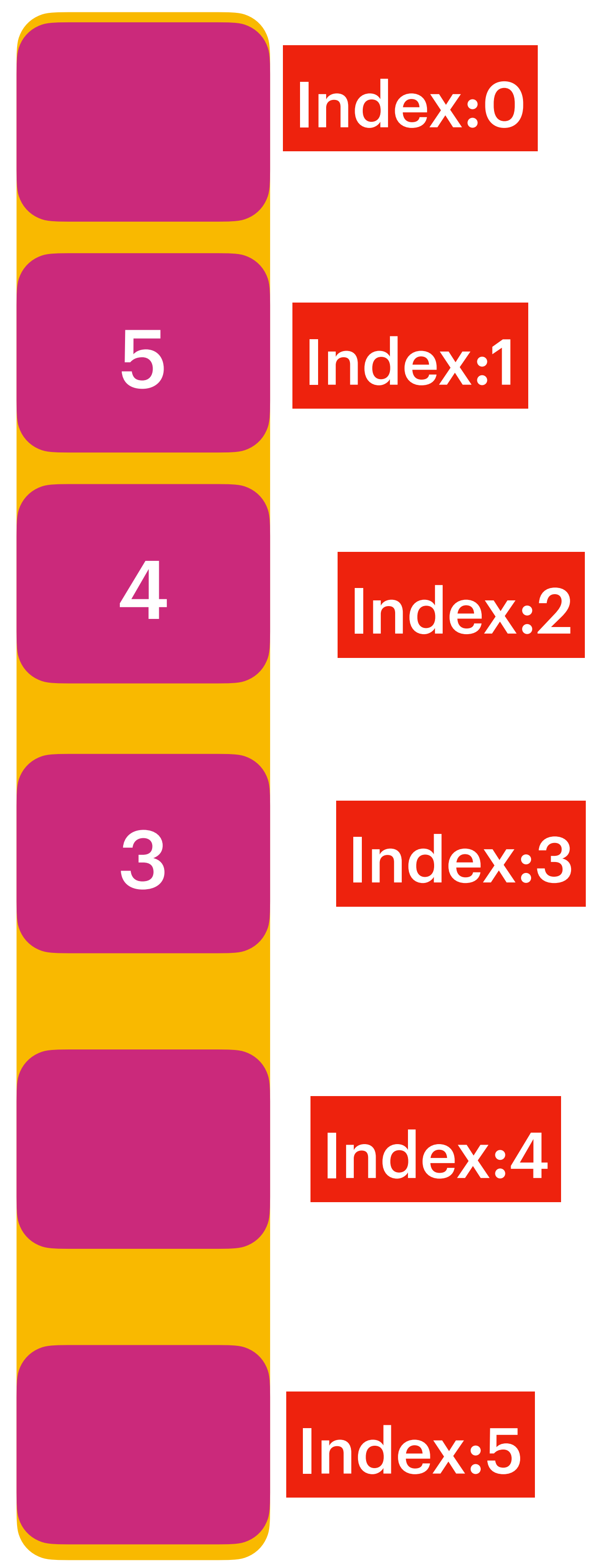
Then replace with the right most element.

delete(11)

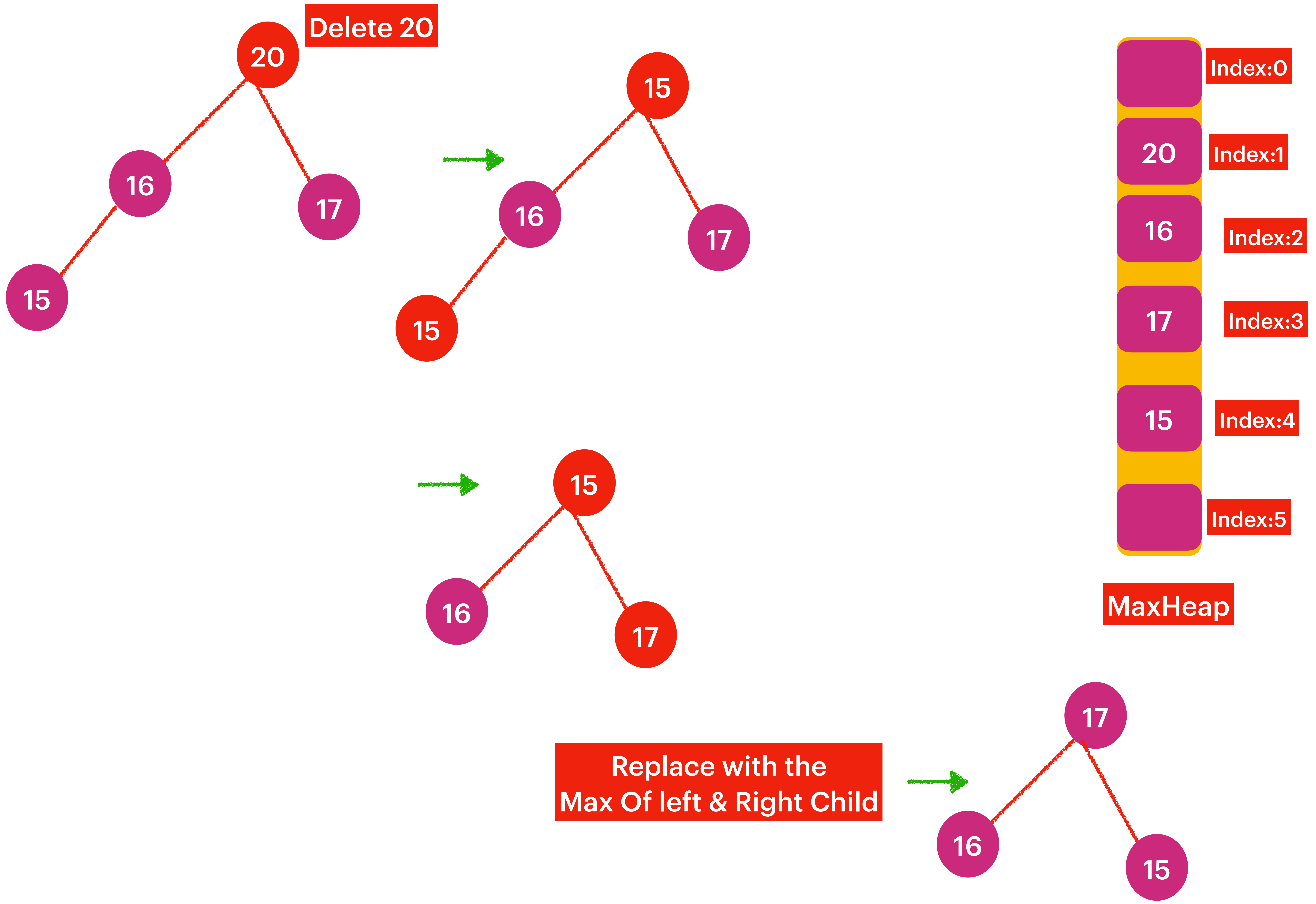




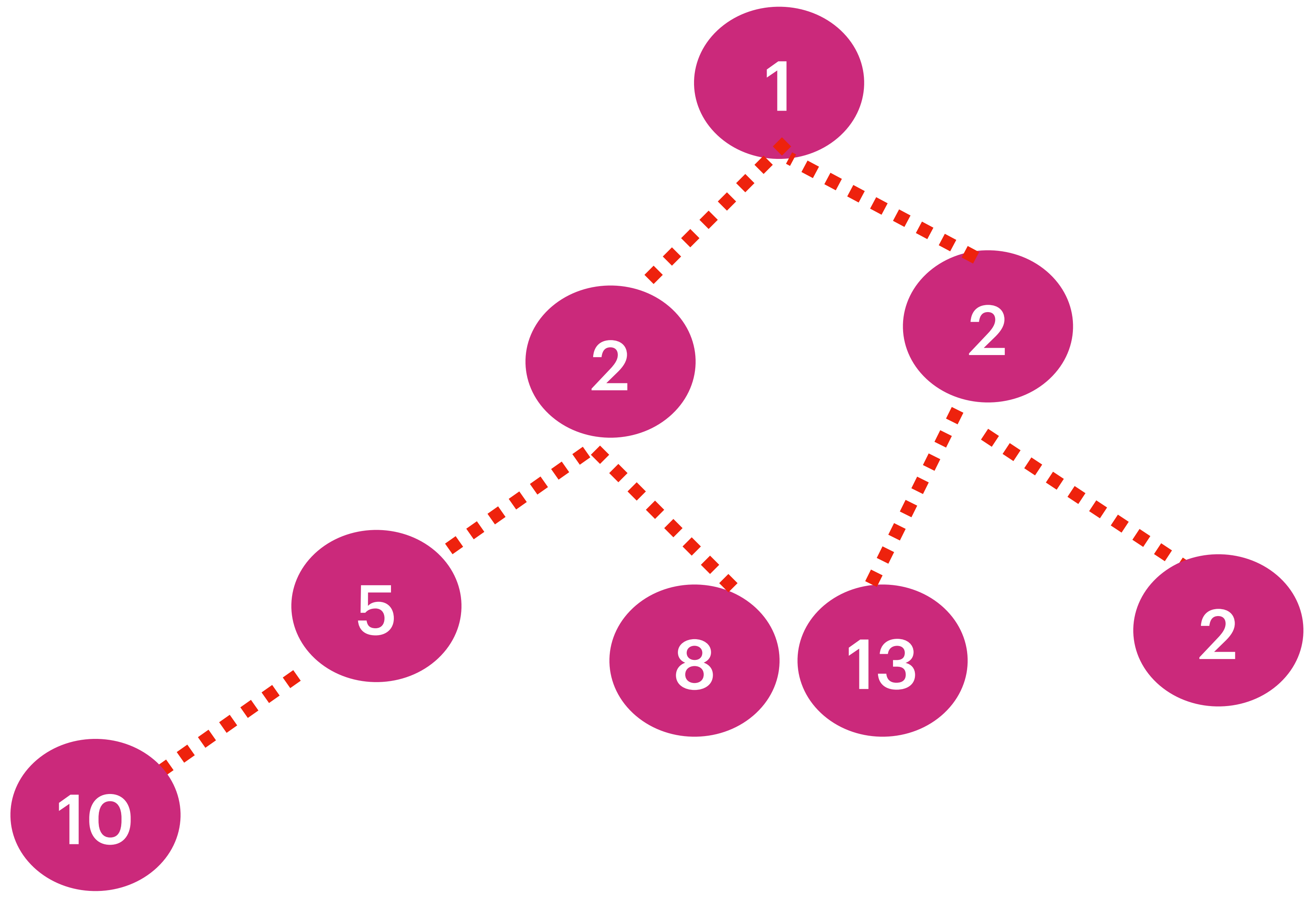


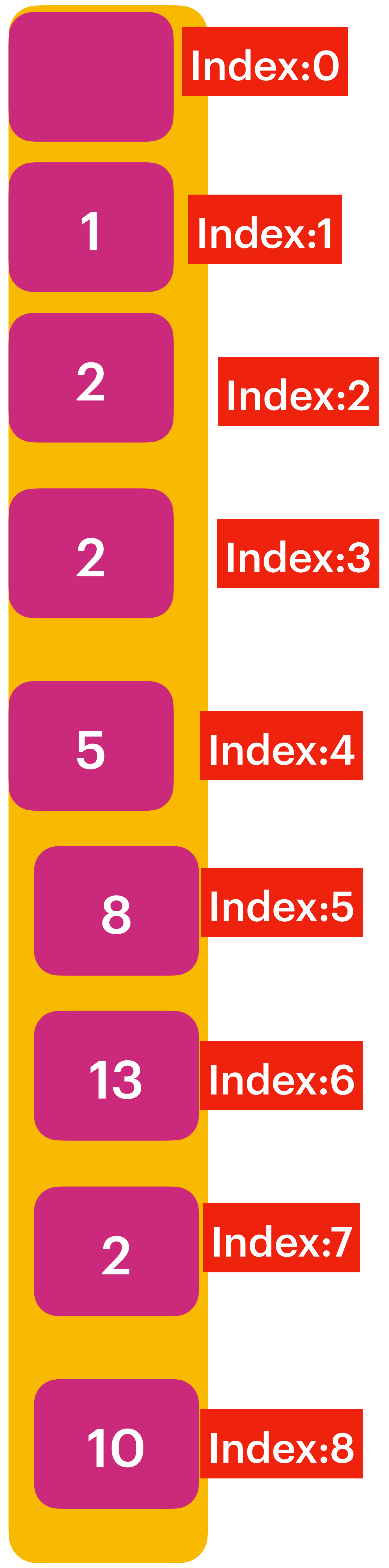


MaxHeap

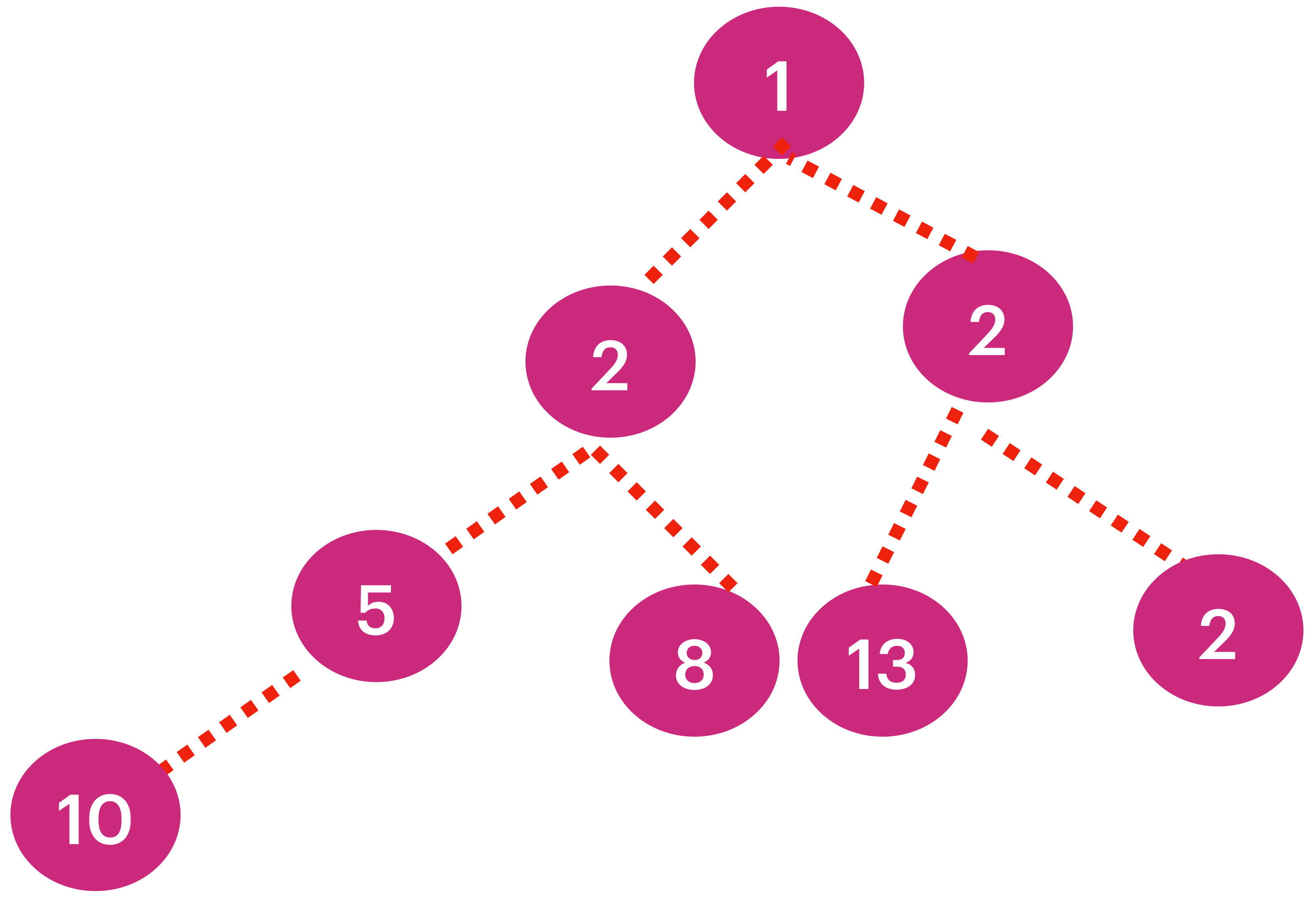

Delete 20
20
16
17
15
15
16
17
15
15
16
17
Index:0
20
Index:1
16
Index:2
17
Index:3
15
Index:4
Index:5
MaxHeap
Replace with the
Max Of left & Right Child
17
16
15

```
queue.add(10);
queue.add(5);
queue.add(13);
queue.add(2);
queue.add(8);
queue.add(2);
queue.add(1);
queue.add(2);
```



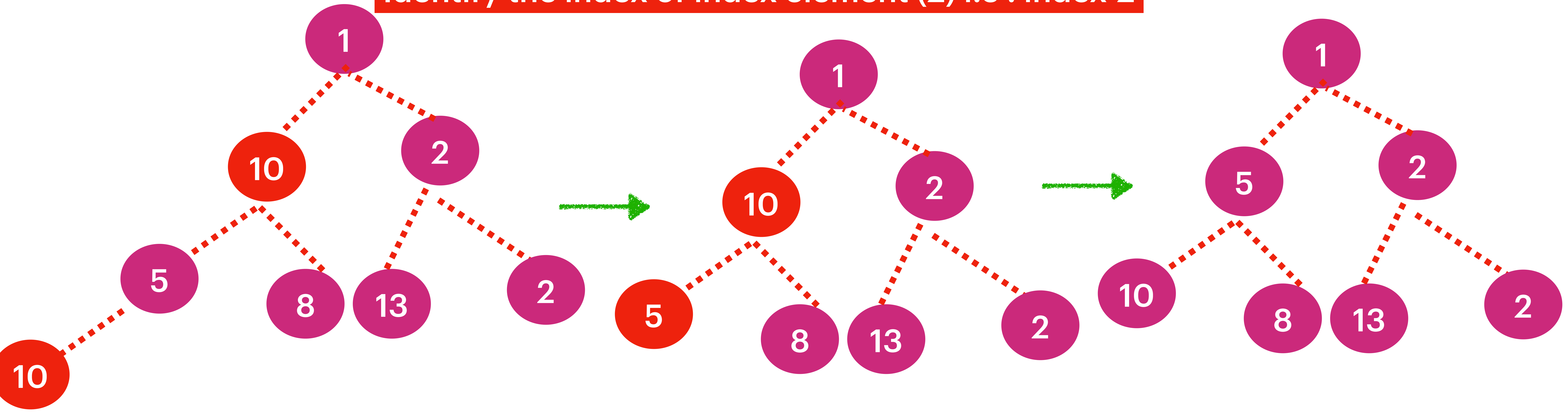


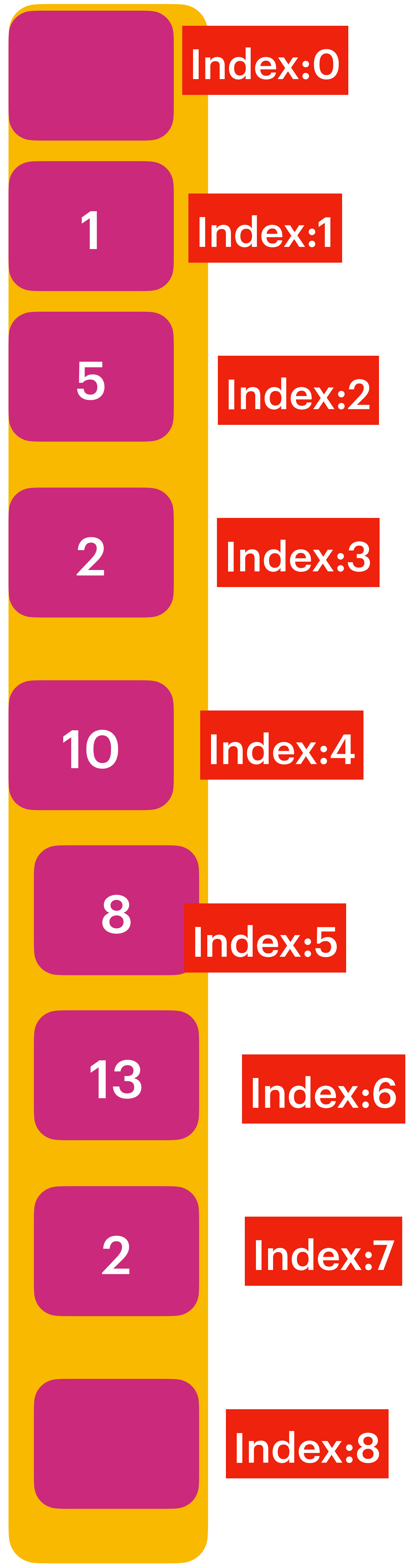
```
queue.add(10);
queue.add(5);
queue.add(13);
queue.add(2);
queue.add(8);
queue.add(2);
queue.add(1);
queue.add(2);
```



Delete (2)

Identify the index of index element (2) i.e : index-2


1
10
2
5
8
13
2
10
1
10
2
5
8
13
2
1
5
2
10
8
13
2


Index:0
1
Index:1
5
Index:2
2
Index:3
10
Index:4
8
Index:5
13
Index:6
2
Index:7
Index:8