CS-2001
DATA STRUCTURE

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

HEAP

Heap

- A Heap is a special Tree-based data structure in which the **tree is a complete binary tree.**
- Heap is a special case of **balanced binary tree** data structure where the *root-node key is compared with its children* and arranged accordingly.
- Generally, Heaps can be of two types:
  - □ **Max-Heap:**
  - □ **Min-Heap:**

Heap

Max-Heap:

- In a Max-Heap, the key present at the root node must be maximum among the keys present at all of its children.
- The same property must be recursively true for all sub-trees in that Binary Tree.
- If α has child node β then,

$$\mathbf{key(\alpha) \geq key(\beta)}$$

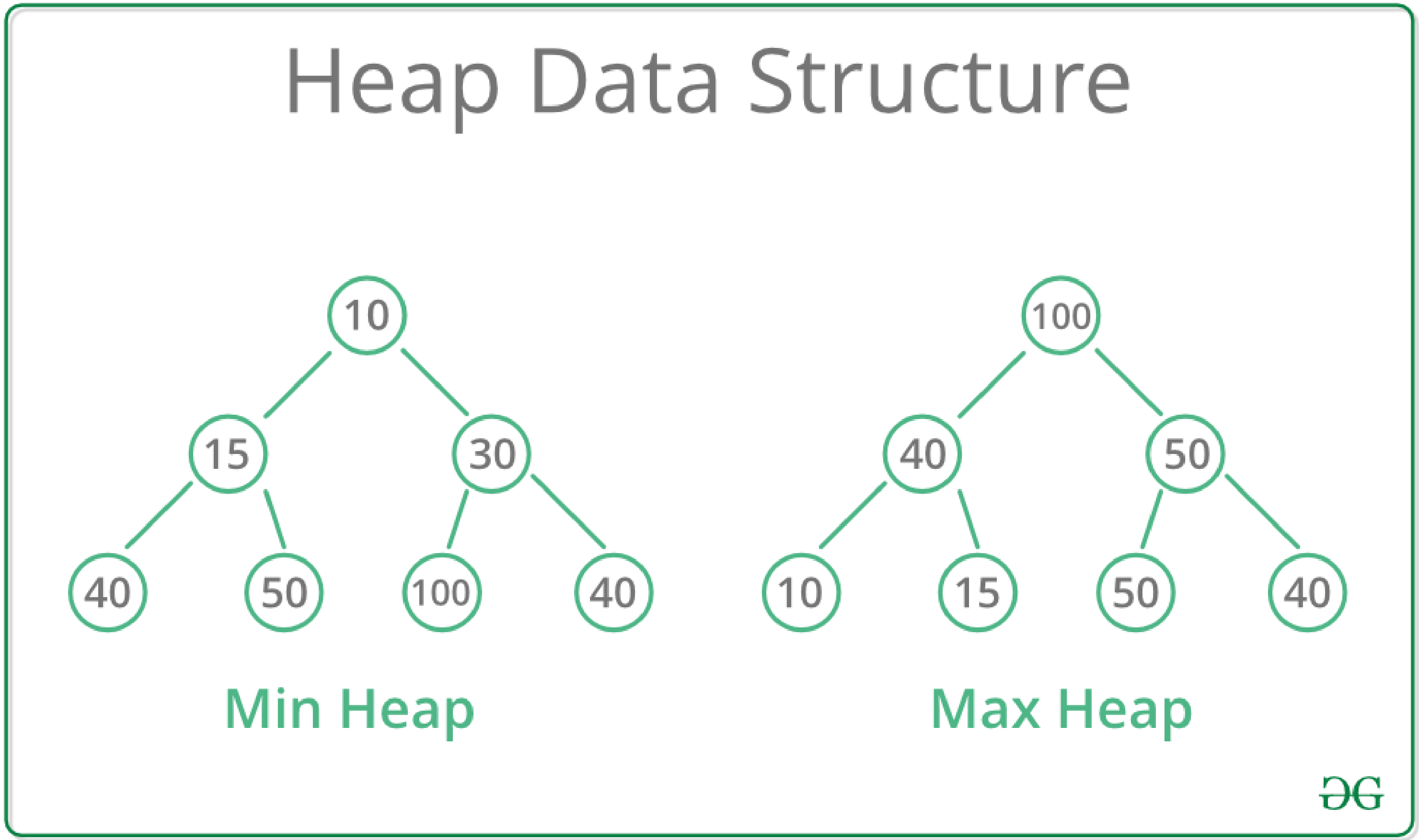
Heap

Min-Heap:

- In a Min-Heap, the key present at the root node must be minimum among the keys present at all of its children.
- The same property must be recursively true for all sub-trees in that Binary Tree.
- If **α** has child node **β** then,

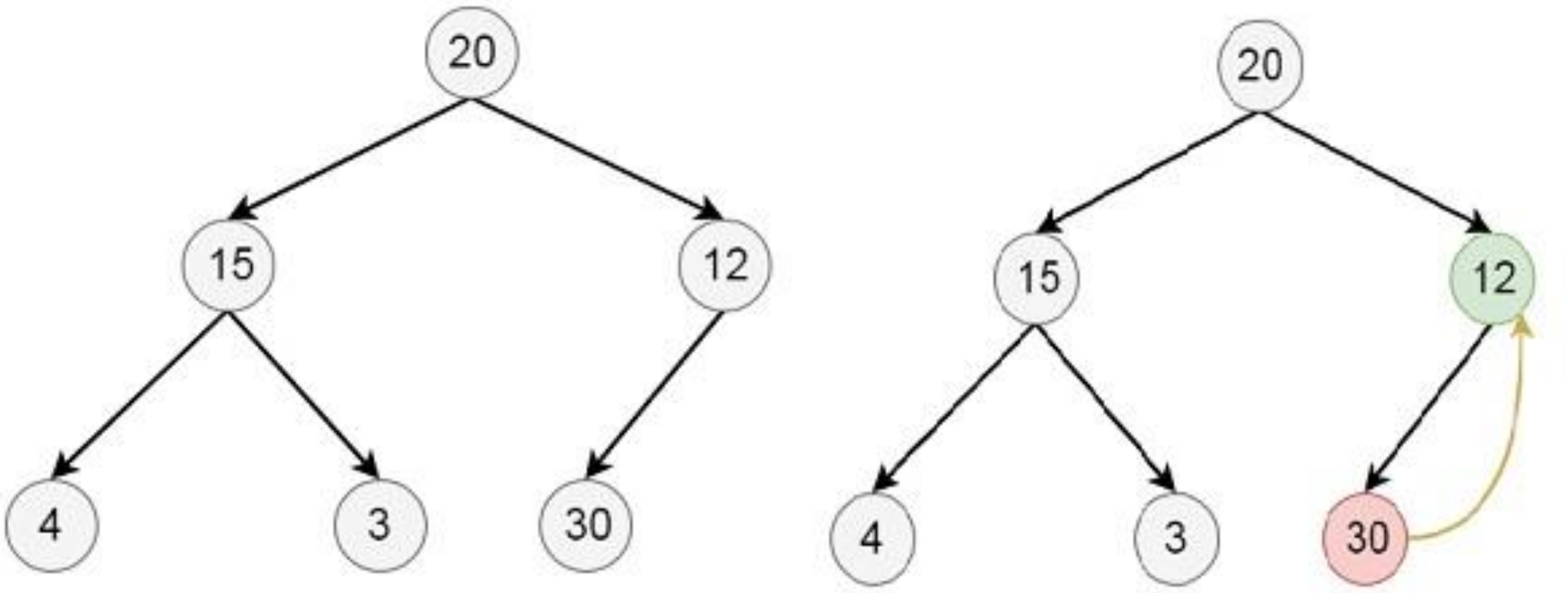
$$\mathbf{key(\alpha) \leq key(\beta)}$$

Examples


Heap Data Structure
10
15
30
40
50
100
40
Min Heap
100
40
50
10
15
50
40
Max Heap

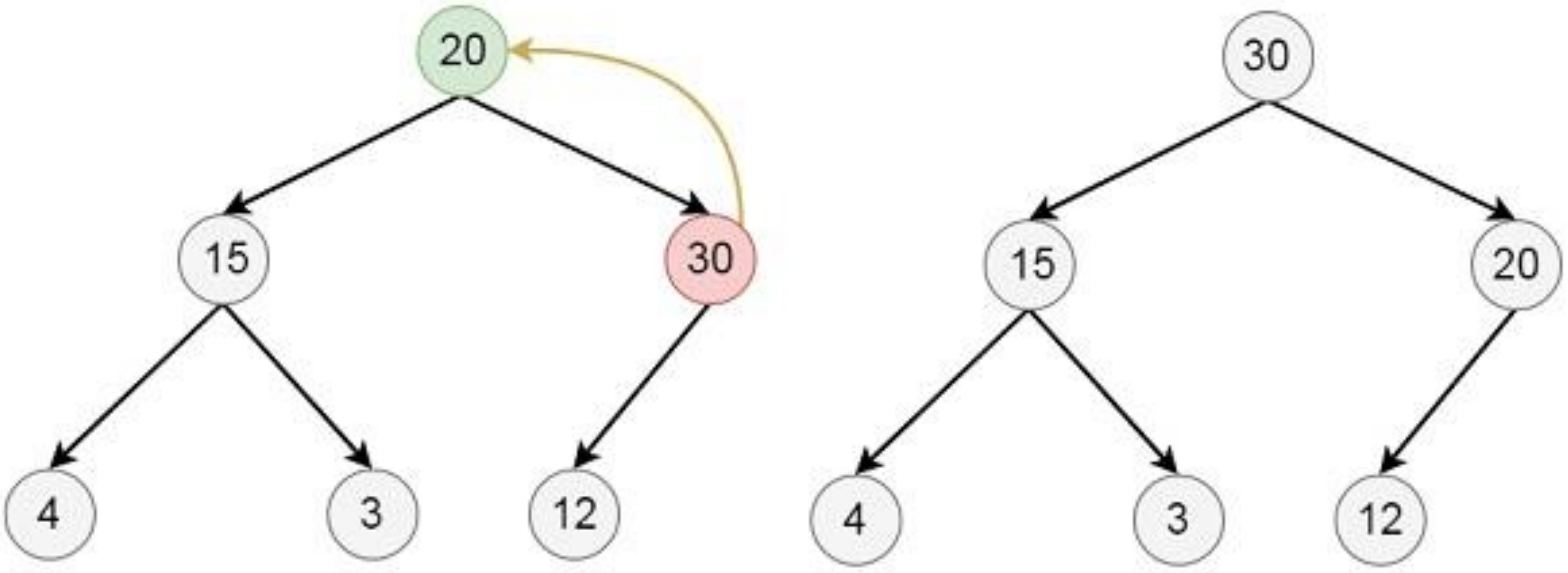
Heap Insertion & Deletion

Suppose we want to insert 30 into the heap –



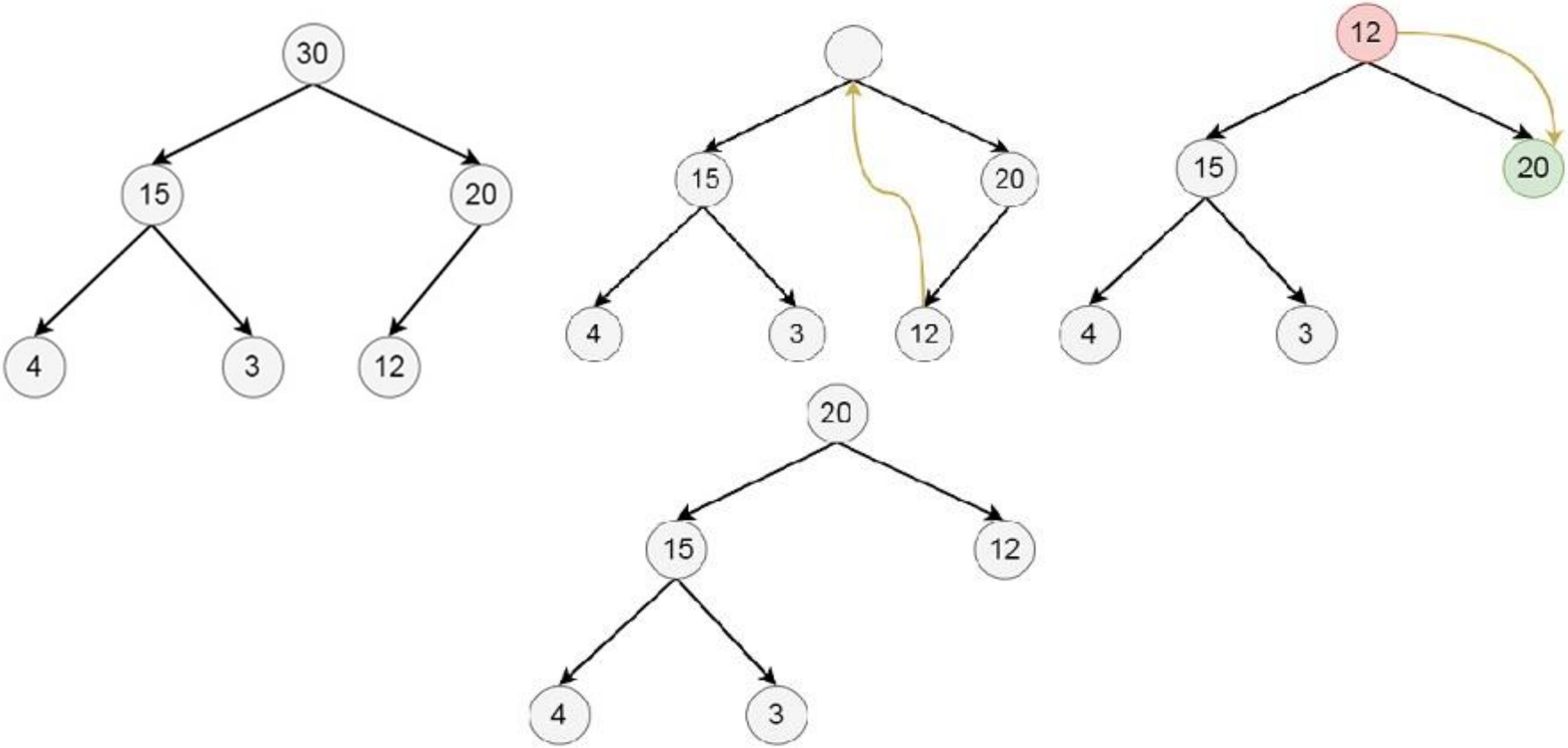
Heap Insertion & Deletion

Suppose we want to insert 30 into the heap –



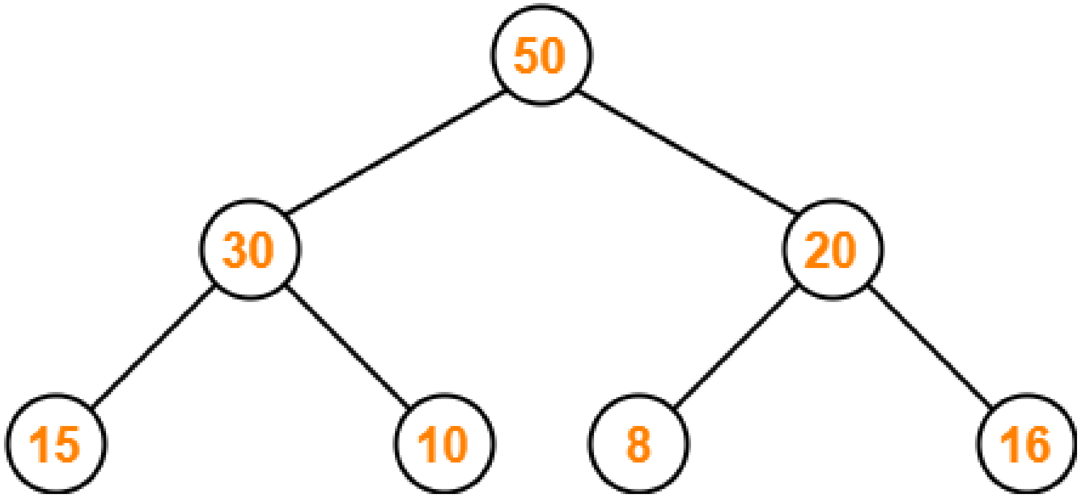
Heap Insertion & Deletion

Suppose we want to Delete 30 into the heap –



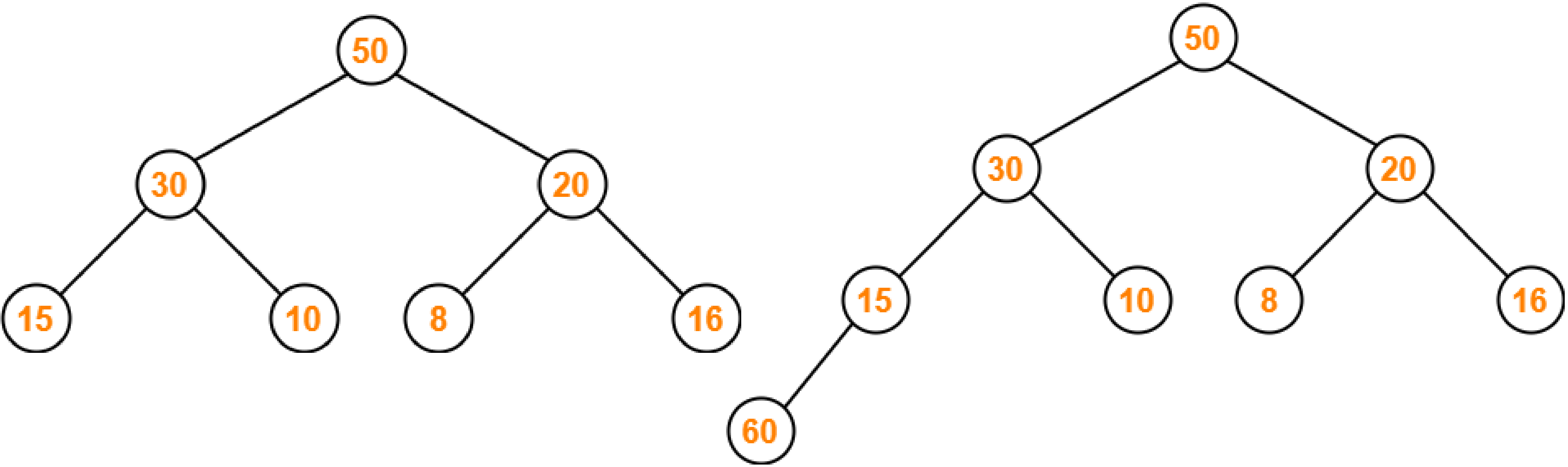
Heap Insertion & Deletion

Suppose we want to Insert 60 into the heap –



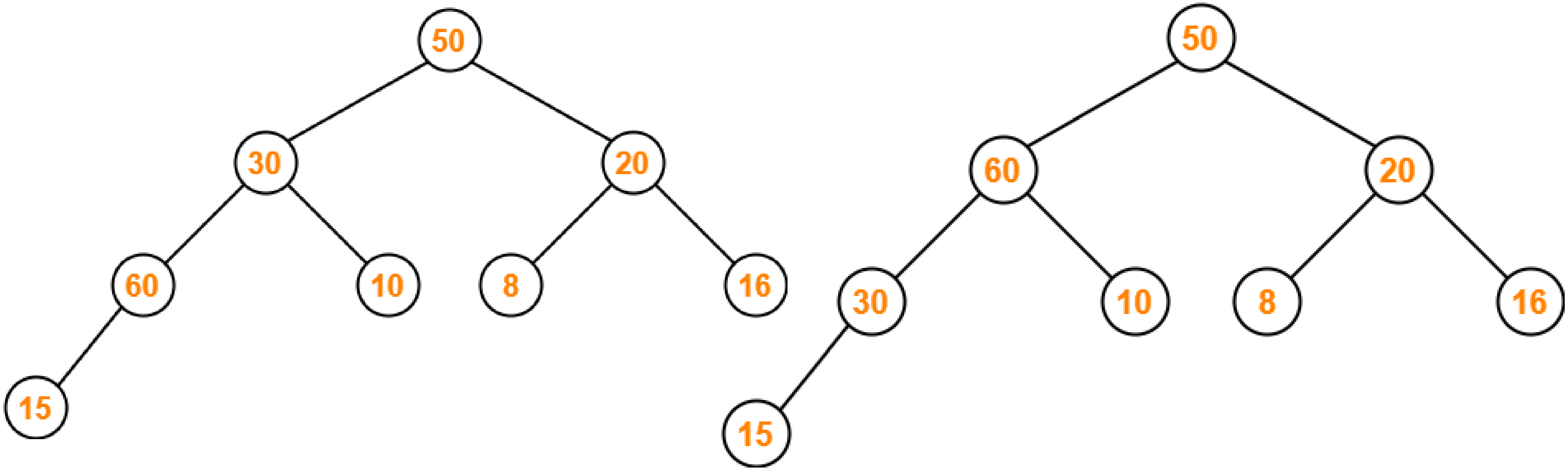
Heap Insertion & Deletion

Suppose we want to Insert 60 into the heap –



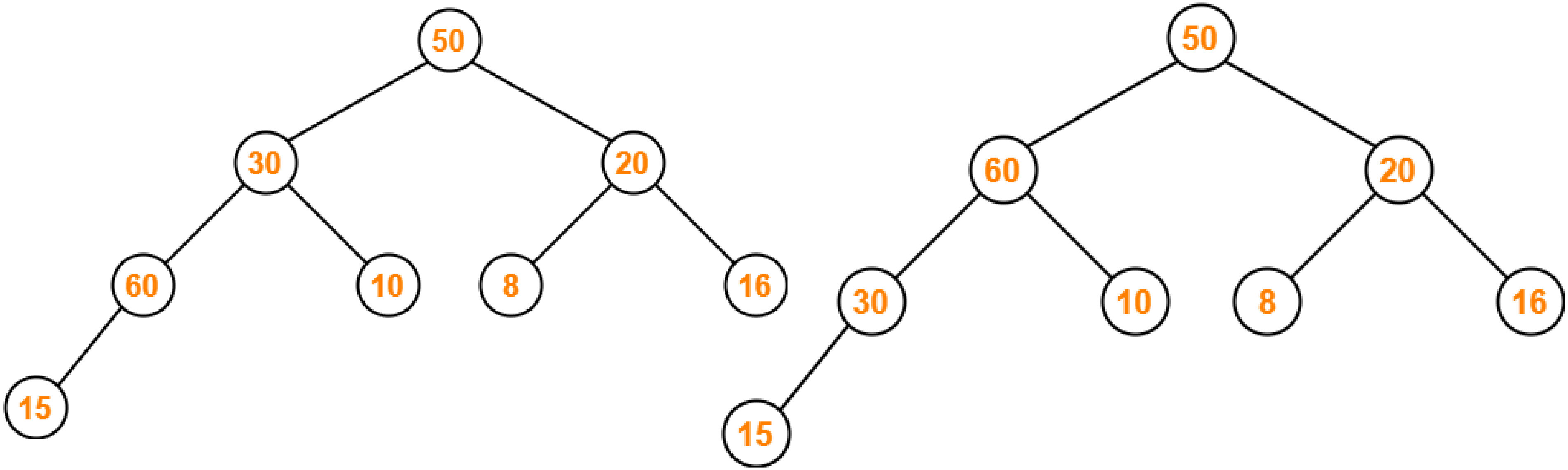
Heap Insertion & Deletion

Suppose we want to Insert 60 into the heap –



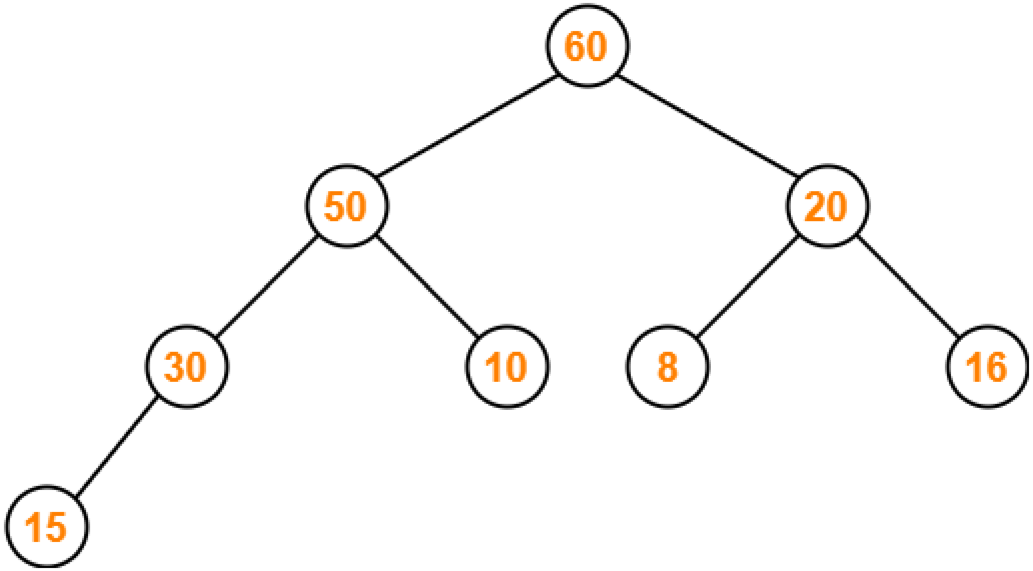
Heap Insertion & Deletion

Suppose we want to Insert 60 into the heap –



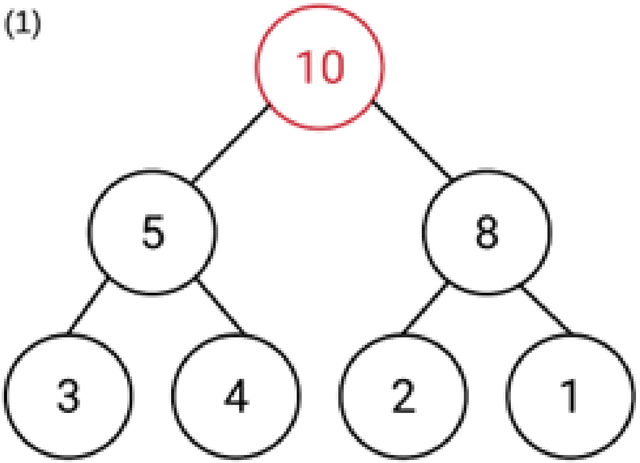
Heap Insertion & Deletion

Suppose we want to Insert 60 into the heap –



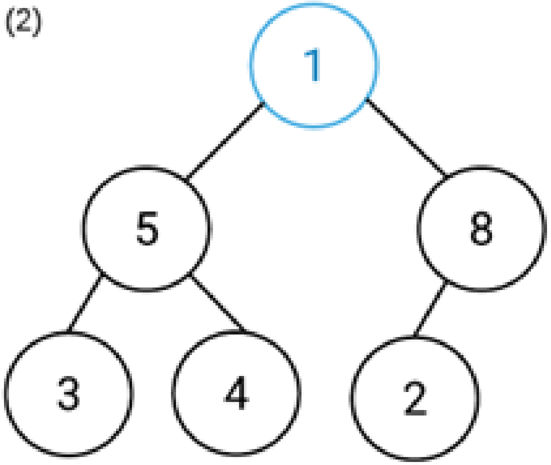
Heap Insertion & Deletion

(1)



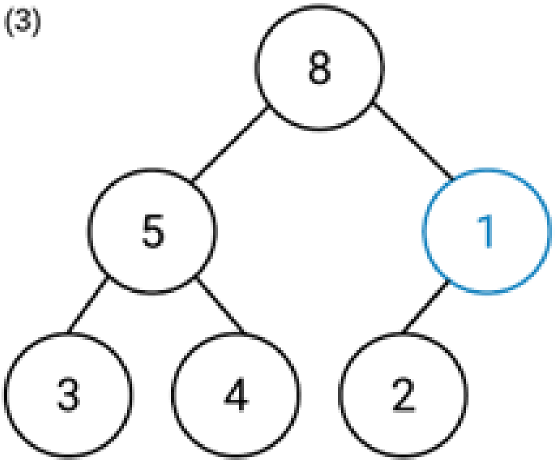
Starting with this max heap

(2)



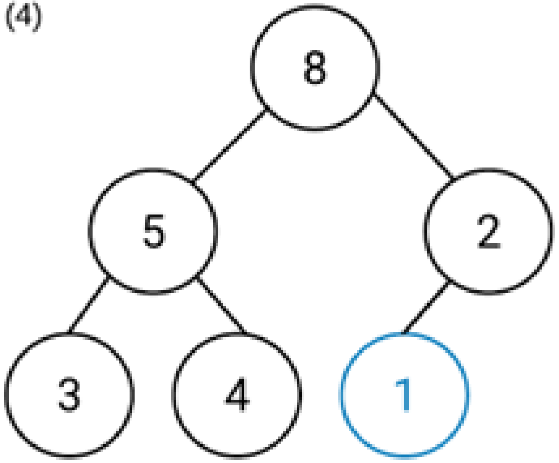
Step 1: the bottom most, left most node, the 1 node, gets placed at the root

(3)



Step 2: Because 1 is less than both of its children, it swaps with the larger element, the 8 node

(4)



Step 3: Once again, 7 is bigger than its parent, the 6 node, so it gets swapped

HEAP SORT

Heap Sort

- □ The Heap sort algorithm to arrange a list of elements in ascending order is performed using the following step,

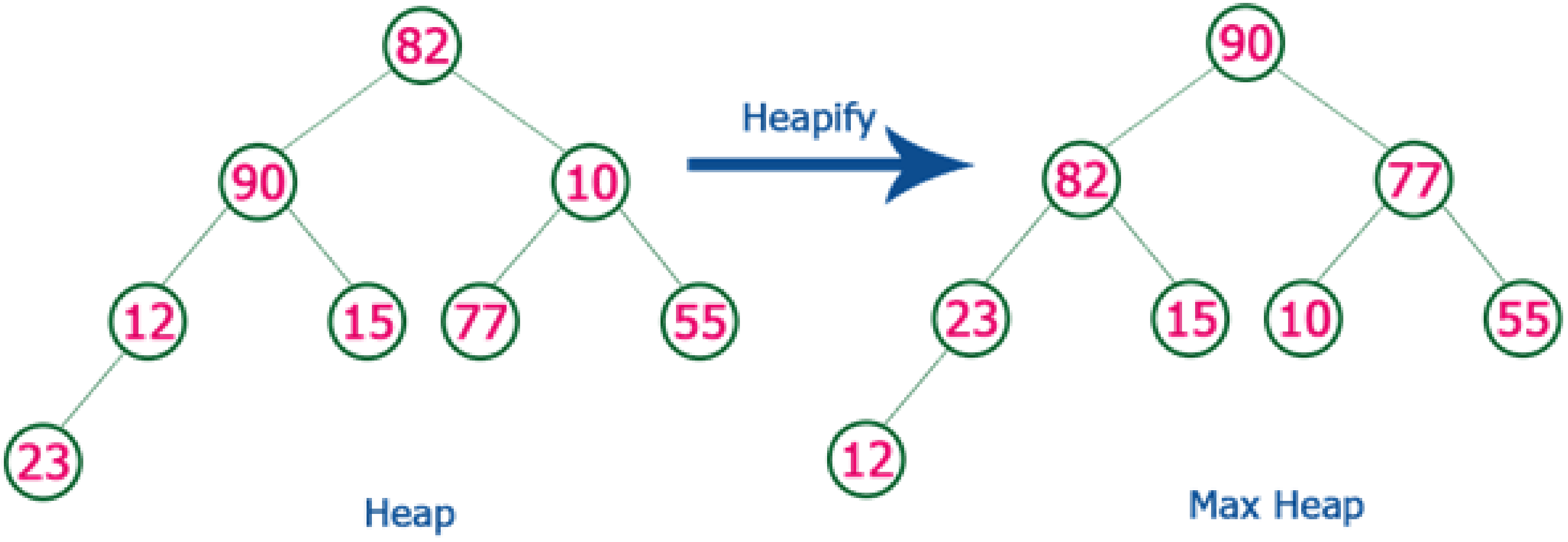
  1. Construct a **Binary Tree.**
  2. Transform the Binary Tree into **Max Heap.**
  3. Delete the root element from Max Heap using the **Heapify** method.
  4. *Put the deleted element into the Sorted list.*
  5. Repeat the same until Max Heap becomes empty.
  6. Display the sorted list.

Heap Sort

Consider the following list of unsorted numbers which are to be sort using Heap Sort

82, 90, 10, 12, 15, 77, 55, 23

Step 1 - Construct a Heap with given list of unsorted numbers and convert to Max Heap

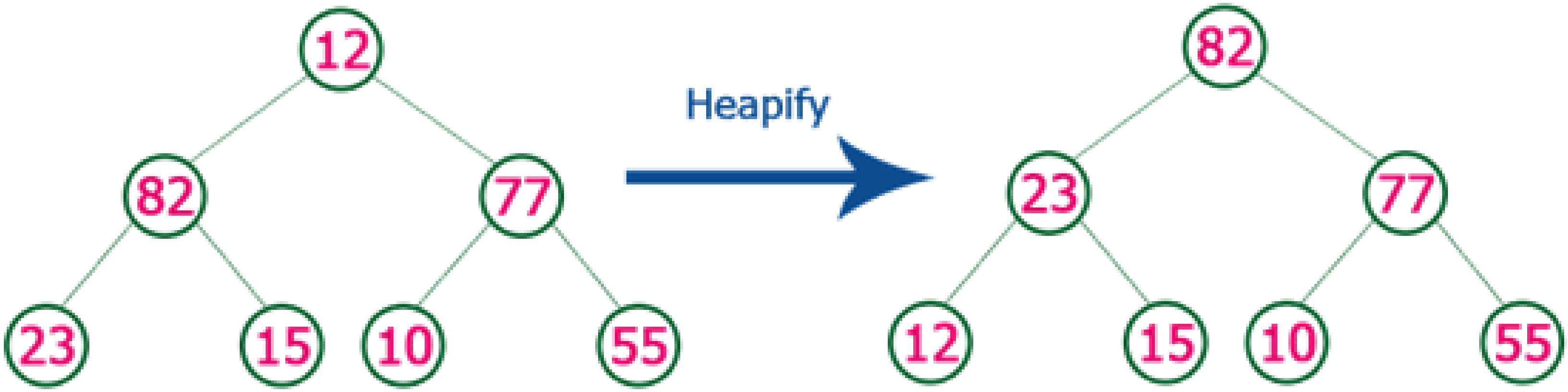


list of numbers after heap converted to Max Heap

90, 82, 77, 23, 15, 10, 55, 12

Heap Sort

Step 2 - Delete root (**90**) from the Max Heap. To delete root node it needs to be swapped with last node (**12**). After delete tree needs to be heapify to make it Max Heap.



Heap after 90 deleted

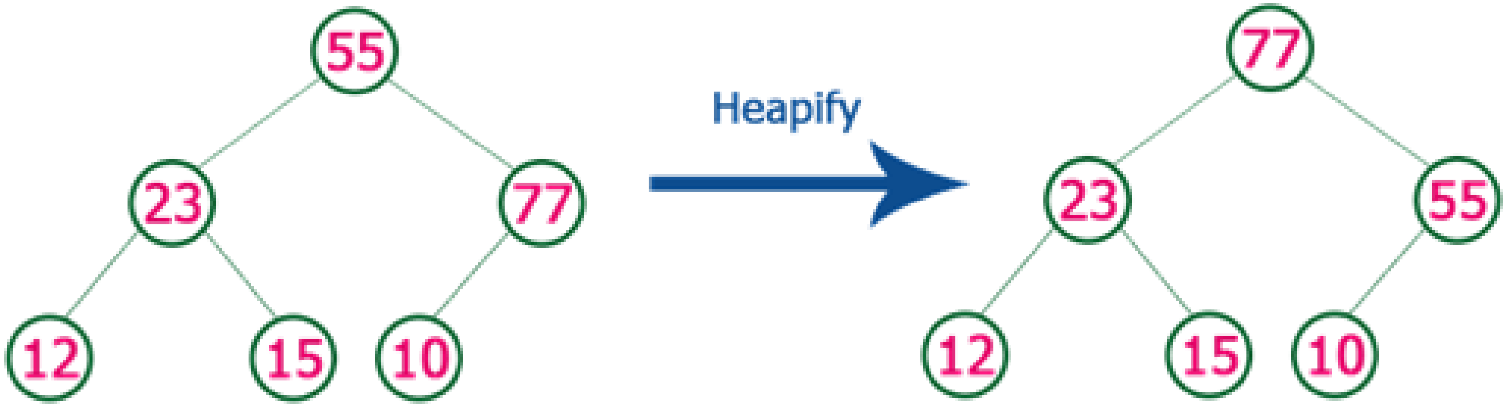
Max Heap

list of numbers after swapping 90 with 12.

12, 82, 77, 23, 15, 10, 55, **90**

Heap Sort

Step 3 - Delete root (**82**) from the Max Heap. To delete root node it needs to be swapped with last node (**55**). After delete tree needs to be heapify to make it Max Heap.



Heap after 82 deleted

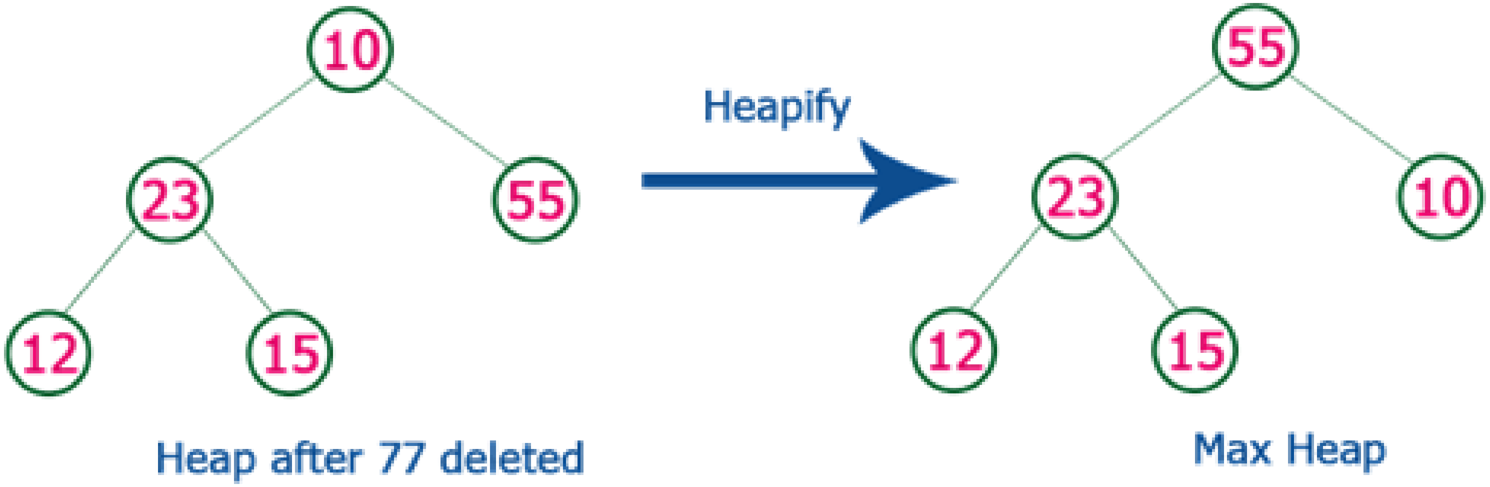
Max Heap

list of numbers after swapping 82 with 55.

12, 55, 77, 23, 15, 10, **82, 90**

Heap Sort

Step 4 - Delete root (**77**) from the Max Heap. To delete root node it needs to be swapped with last node (**10**). After delete tree needs to be heapify to make it Max Heap.



Heap after 77 deleted

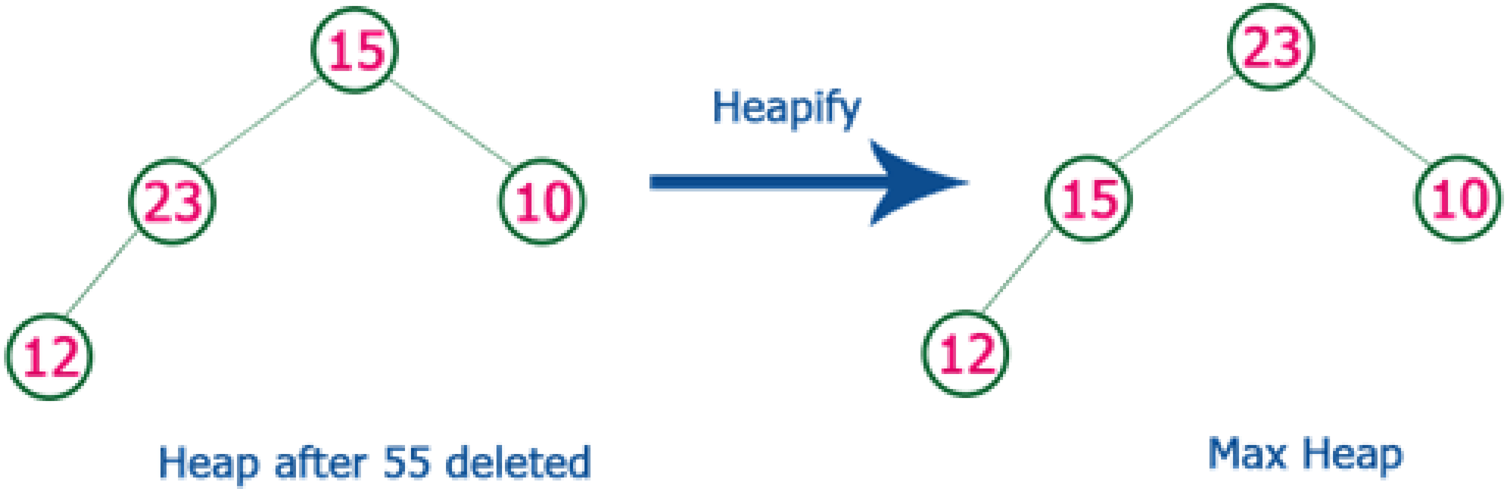
Max Heap

list of numbers after swapping 77 with 10.

12, 55, 10, 23, 15, **77, 82, 90**

Heap Sort

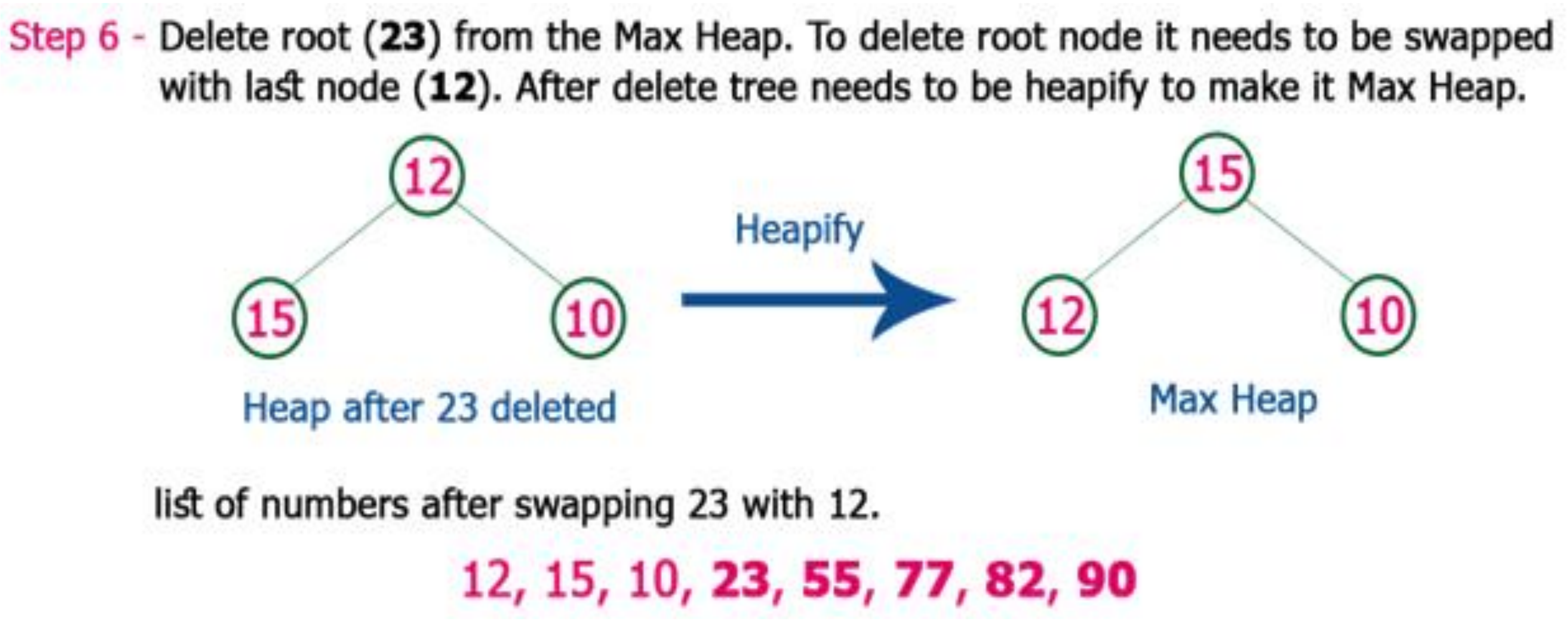
Step 5 - Delete root (**55**) from the Max Heap. To delete root node it needs to be swapped with last node (**15**). After delete tree needs to be heapify to make it Max Heap.



list of numbers after swapping 55 with 15.

12, 15, 10, 23, **55, 77, 82, 90**

Heap Sort



Step 6 - Delete root (**23**) from the Max Heap. To delete root node it needs to be swapped with last node (**12**). After delete tree needs to be heapify to make it Max Heap.

Heap after 23 deleted

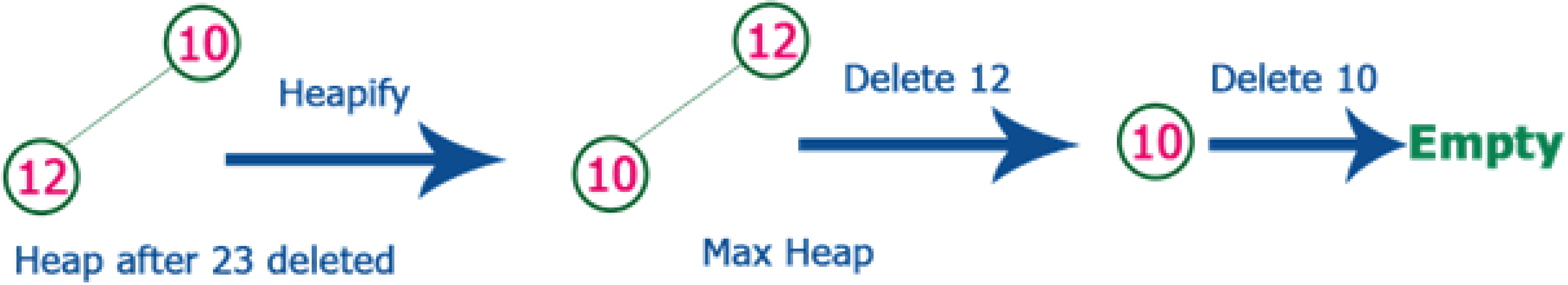
Max Heap

list of numbers after swapping 23 with 12.

12, 15, 10, **23, 55, 77, 82, 90**

Heap Sort

Step 7 - Delete root (**15**) from the Max Heap. To delete root node it needs to be swapped with last node (**10**). After delete tree needs to be heapify to make it Max Heap.



list of numbers after Deleting 15, 12 & 10 from the Max Heap.

10, 12, 15, 23, 55, 77, 82, 90

Whenever Max Heap becomes Empty, the list get sorted in Ascending order

Reading Materials

- Schaum's Outlines: Chapter # 7
- D. S. Malik: Chapter # 11
- Nell Dale: Chapter # 8