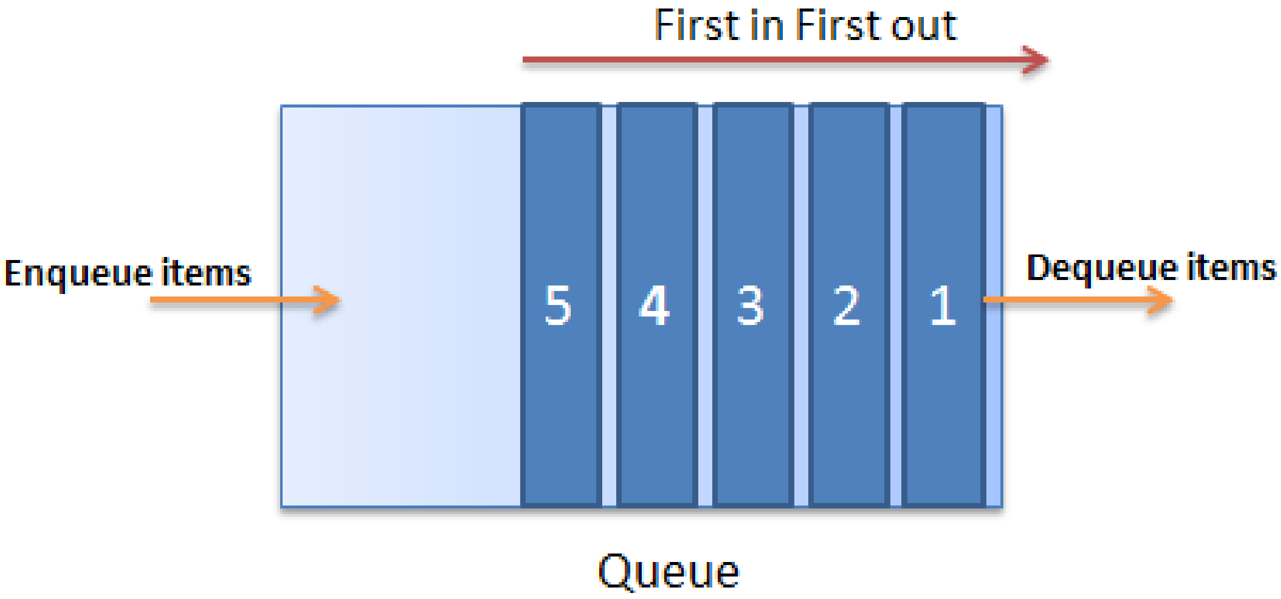
Priority Queue & Heap

Data Structures

Fall 2022

Queue vs Priority Queue

- Recall that the *Queue* data structure follows fair policy for insertion and removal i.e. First In First Out **(FIFO)**



Queue vs Priority Queue

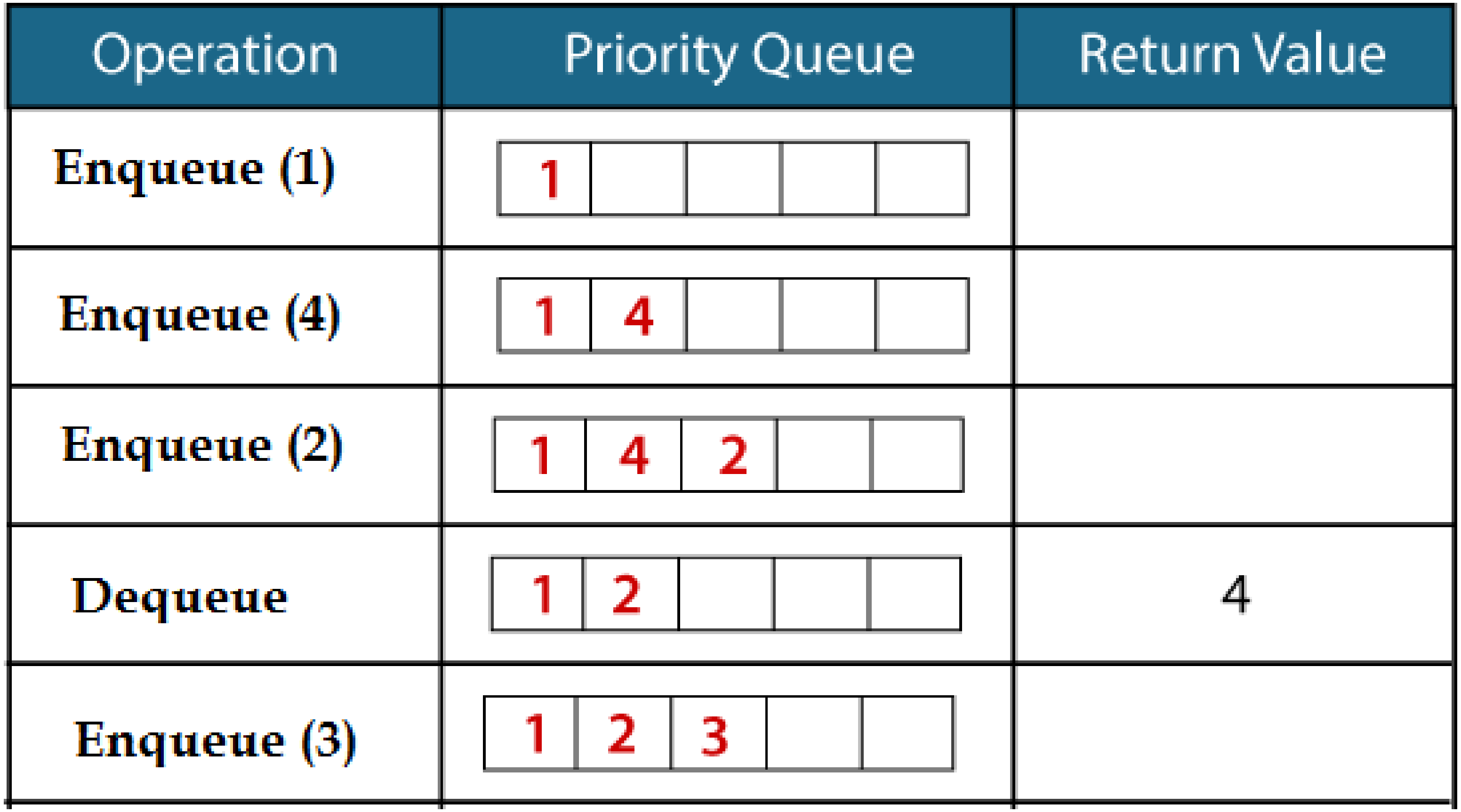
- What if some elements of different priorities?
- What if the highest (or lowest) priority needs to be removed from the queue instead of the element that was inserted first?
- We need a **priority based** (yet unfair) policy for queues!!!

Priority Queue

- A ***priority queue*** is a special type of queue in which each element is associated with a priority and is served/read/removed/outputted according to its priority

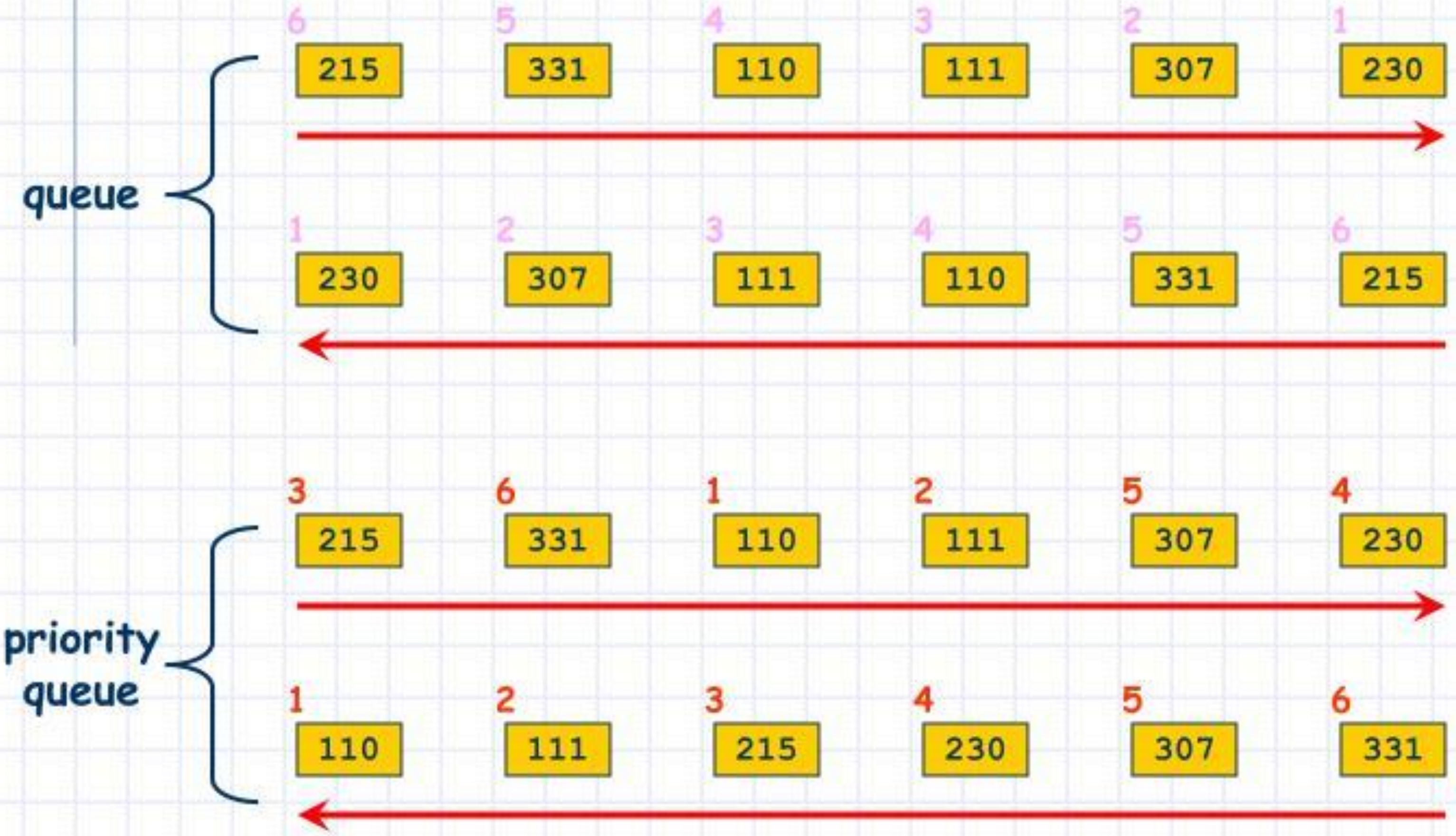
- If elements with the same priority occur, they are served according to their order in the queue

Example: Priority Queue



Operation	Priority Queue	Return Value
Enqueue (1)	1	
Enqueue (4)	1 4	
Enqueue (2)	1 4 2	
Dequeue	1 2	4
Enqueue (3)	1 2 3	

Queue vs Priority Queue



queue

6 215 5 331 4 110 3 111 2 307 1 230

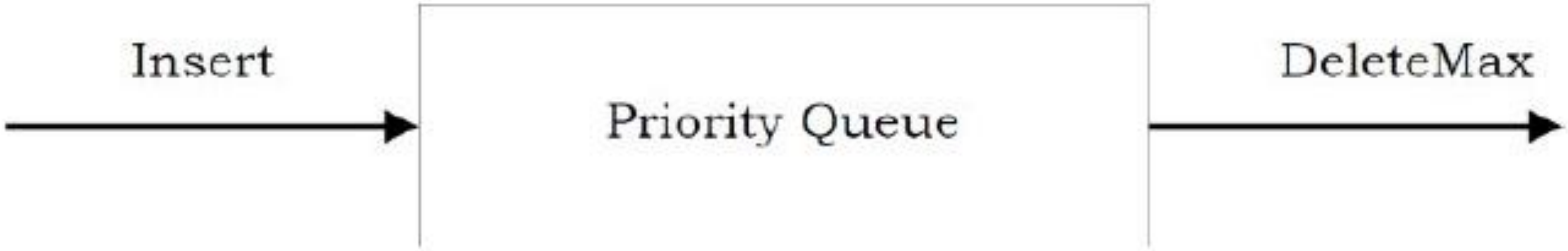
1 230 2 307 3 111 4 110 5 331 6 215

priority queue

3 215 6 331 1 110 2 111 5 307 4 230

1 110 2 111 3 215 4 230 5 307 6 331

Example: Max Priority Queue



Operations

- **Primary operations**
  - Enqueue : Inserting a new element
  - DeleteMin/DeleteMax : Performing deletion (dequeue) based on priority
  - GetMin/GetMax : Read min or max priority value without deleting it

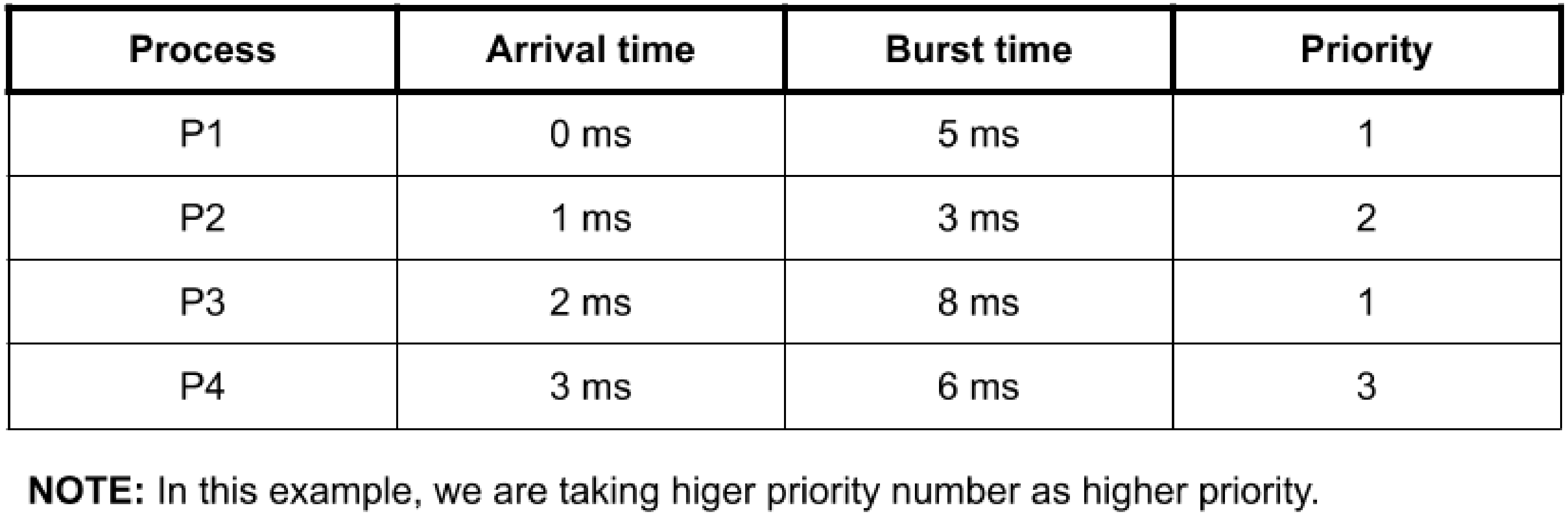
- **Secondary operations**
  - kth smallest / kth largest element
  - Size : Returning size of queue

Applications

- Minimum spanning tree
- Shortest path algorithms
- Operating System scheduling algorithms
- Real-time customer care

... and many more

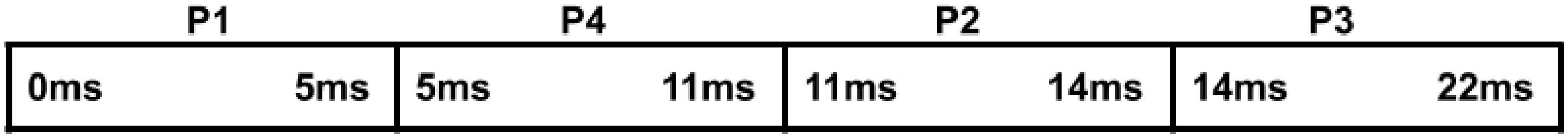
Application: OS Scheduling Algorithm



Process	Arrival time	Burst time	Priority
P1	0 ms	5 ms	1
P2	1 ms	3 ms	2
P3	2 ms	8 ms	1
P4	3 ms	6 ms	3

NOTE: In this example, we are taking higer priority number as higher priority.

Job Schedule based on Priority:



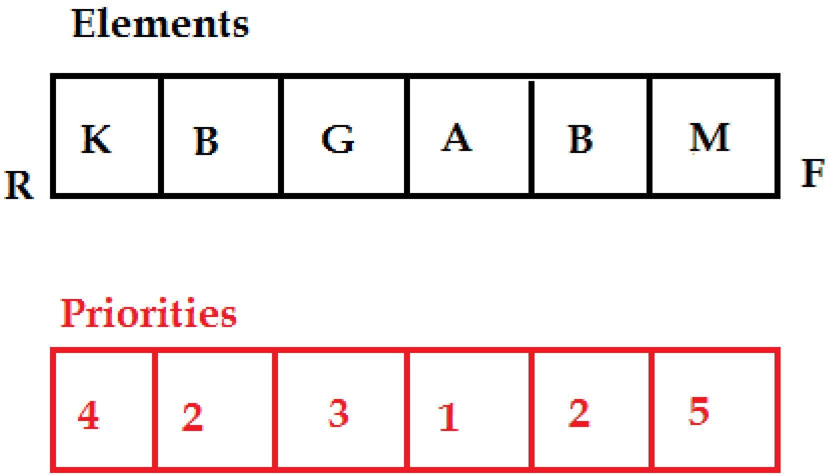
P1	P4	P2	P3
0ms 5ms	5ms 11ms	11ms 14ms	14ms 22ms

Implementation

- Using arrays (shadow array)
- Using Linked List
- Using Heap

Naïve Array Implementation

- Maintain another array of priorities
- Each element in shadow array represents priority at corresponding index of queue



Elements

R

K	B	G	A	B	M

F

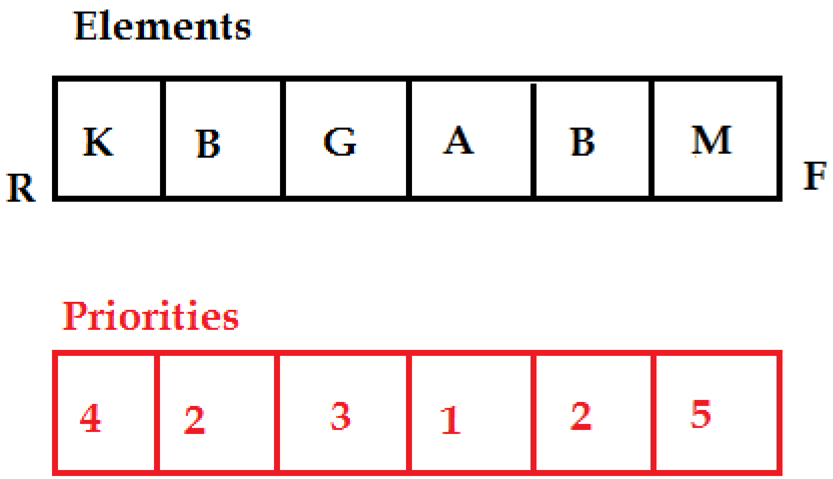
Priorities

4	2	3	1	2	5

Naïve Array Implementation

- Enqueue is same as in queue

- For dequeue a single pass (O(n)) is made over shadow array and index of highest value is recorded.



Elements

R	K	B	G	A	B	M	F

Priorities

4	2	3	1	2	5

- Element at that index is then DeQueued

Binary Heap

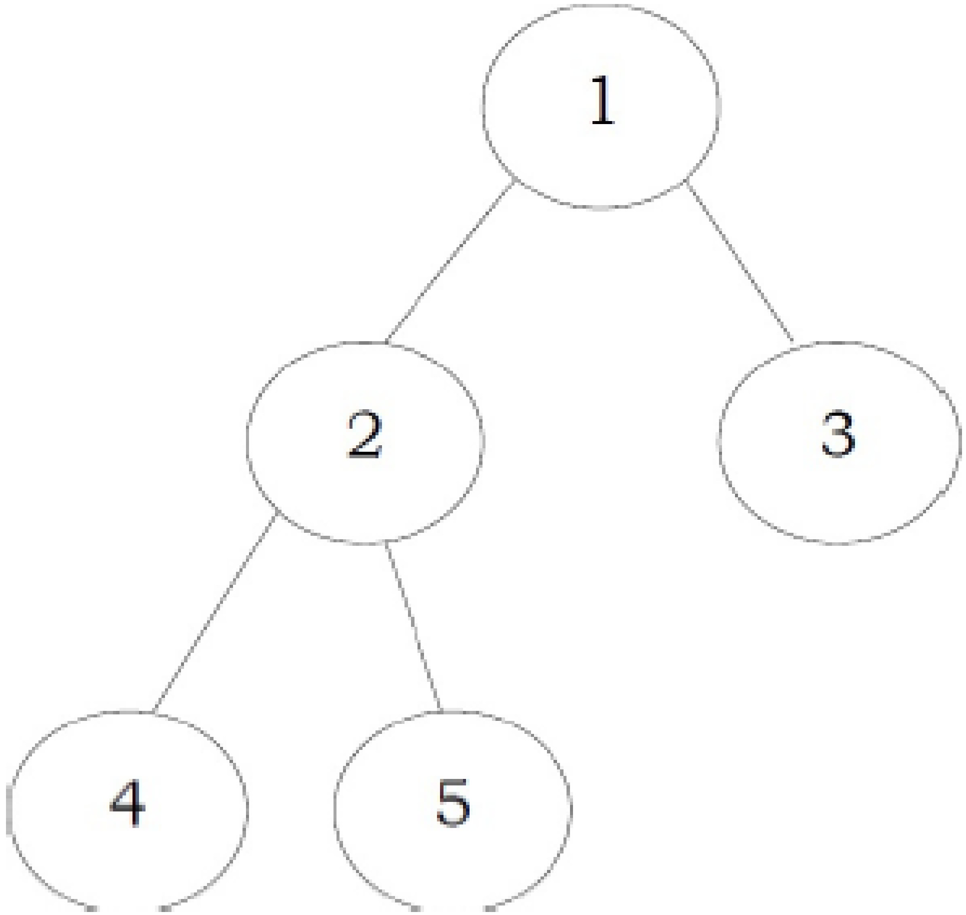
- A Binary Heap is a data structure which has the following properties:
  - It is a complete binary tree
  - For any given node, its value must be ≥ (or ≤) than the values of its children

- This is called Heap Property

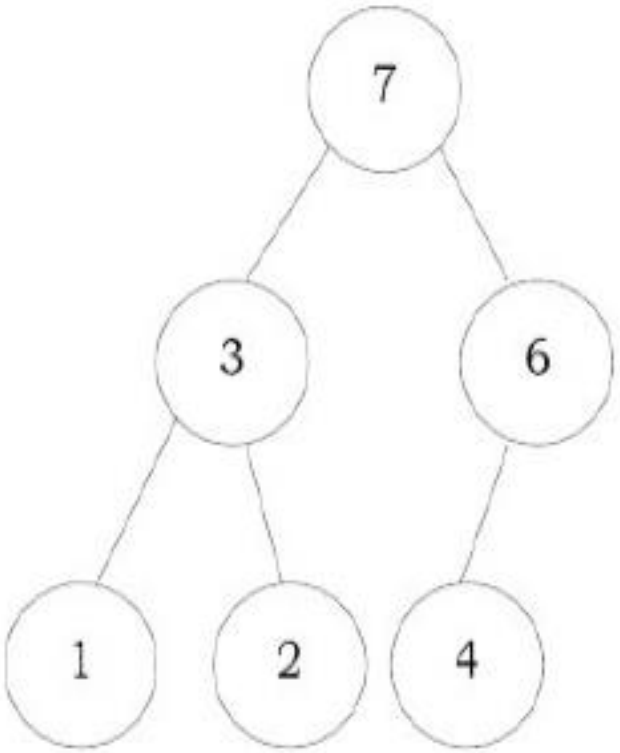
Types of Heap

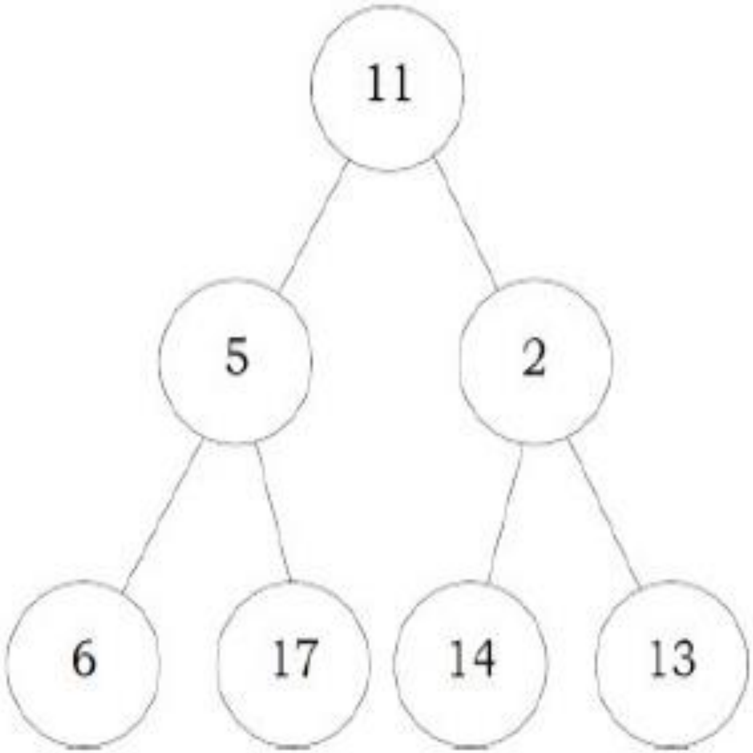
- There are two types of Heaps:
  - Max Heap : The value of a node must be greater than or equal to the values of its children
  - Min Heap : The value of a node must be less than or equal to the values of its children

Example

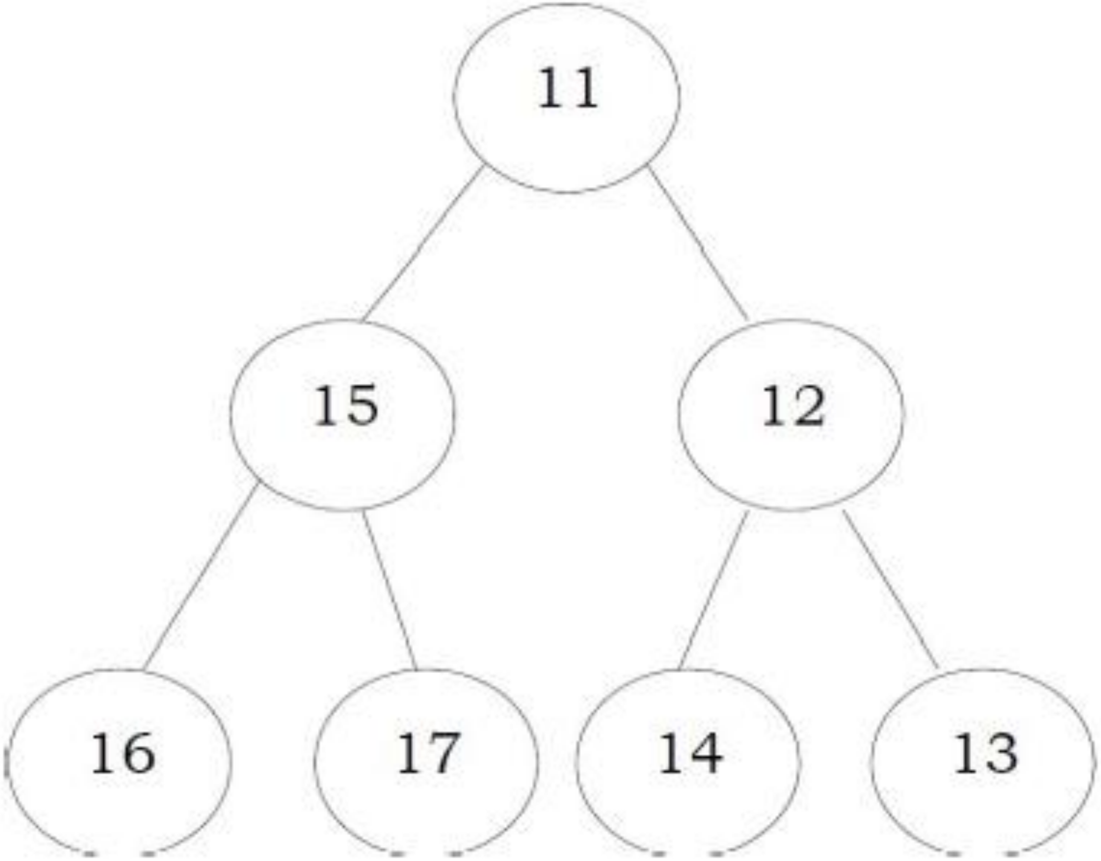


Which of these is a Max Heap?

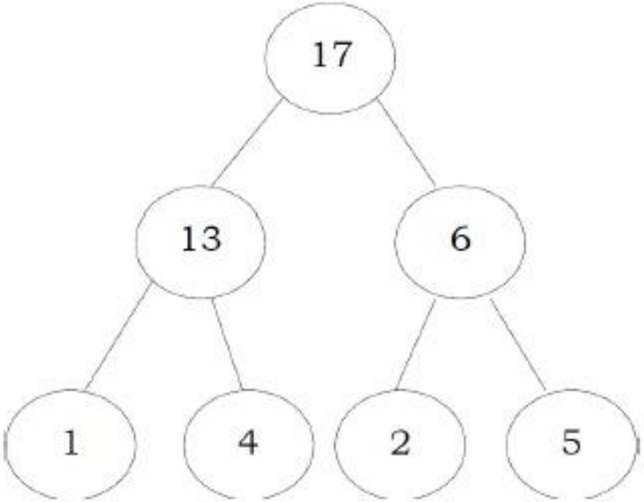




Is this a Max or Min Heap?



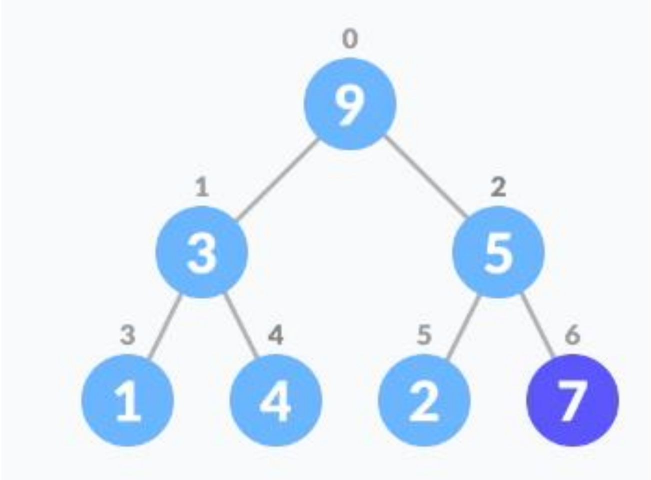
Priority Queue using Heap



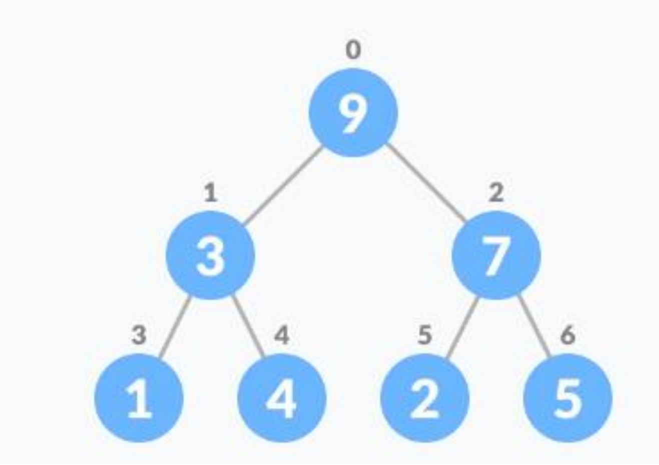
17	13	6	1	4	2	5
0	1	2	3	4	5	6

Insertion in Priority Queue

- Insert new element

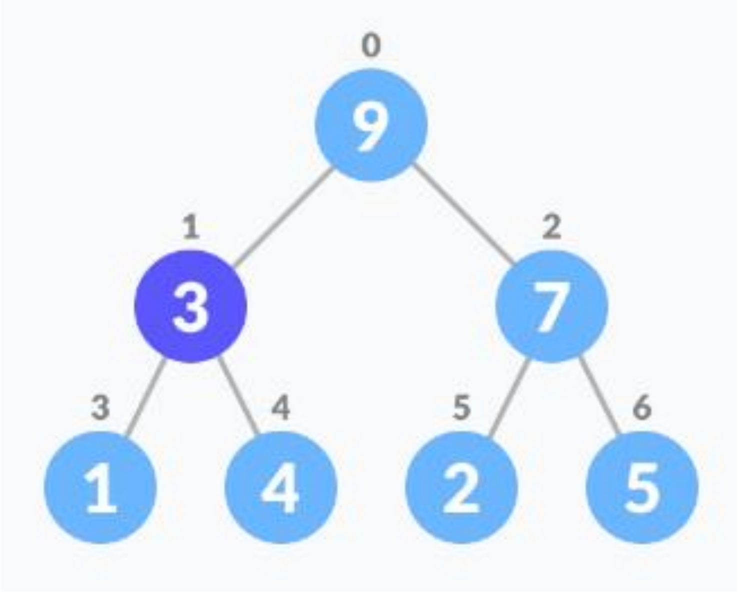


- Heapify



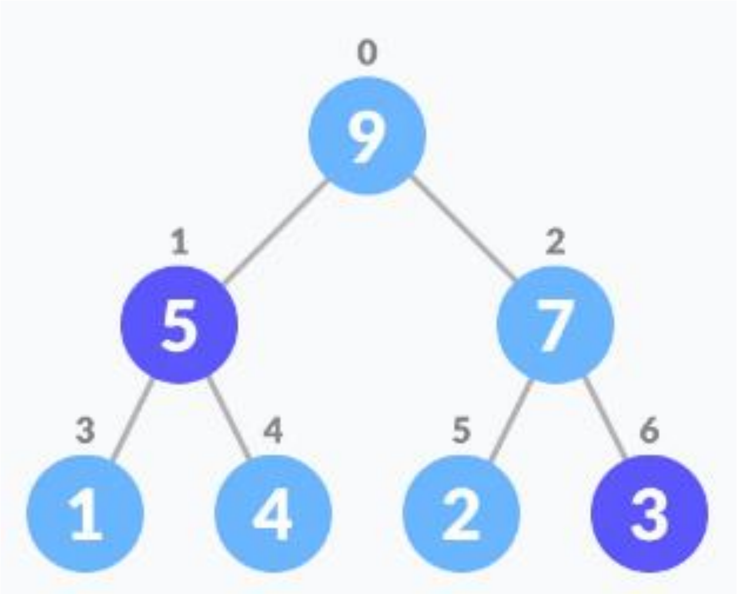
Deletion in Priority Queue

- Select the element to be deleted



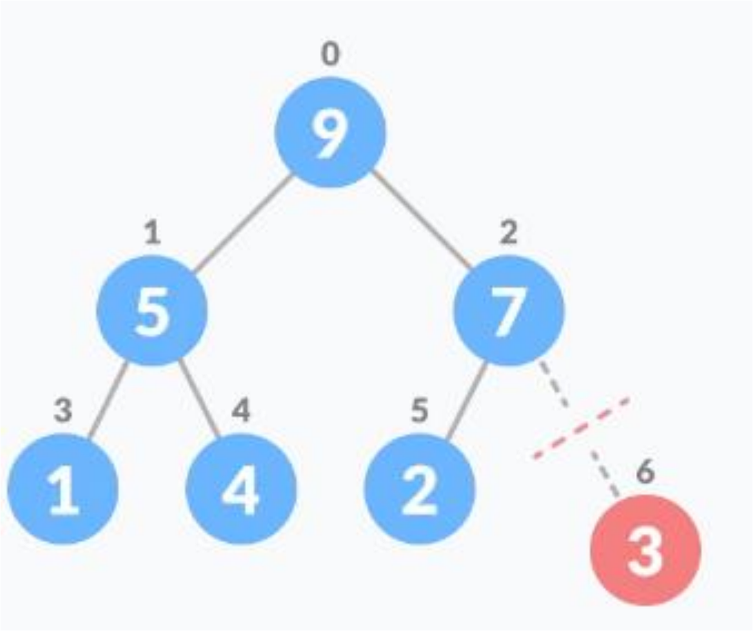
Deletion

- Swap it with the last element



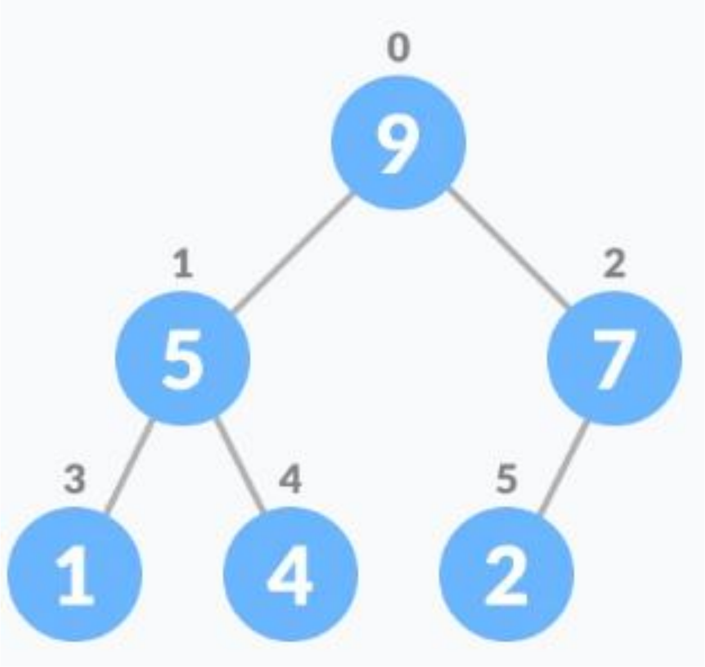
Deletion

- Remove the last element

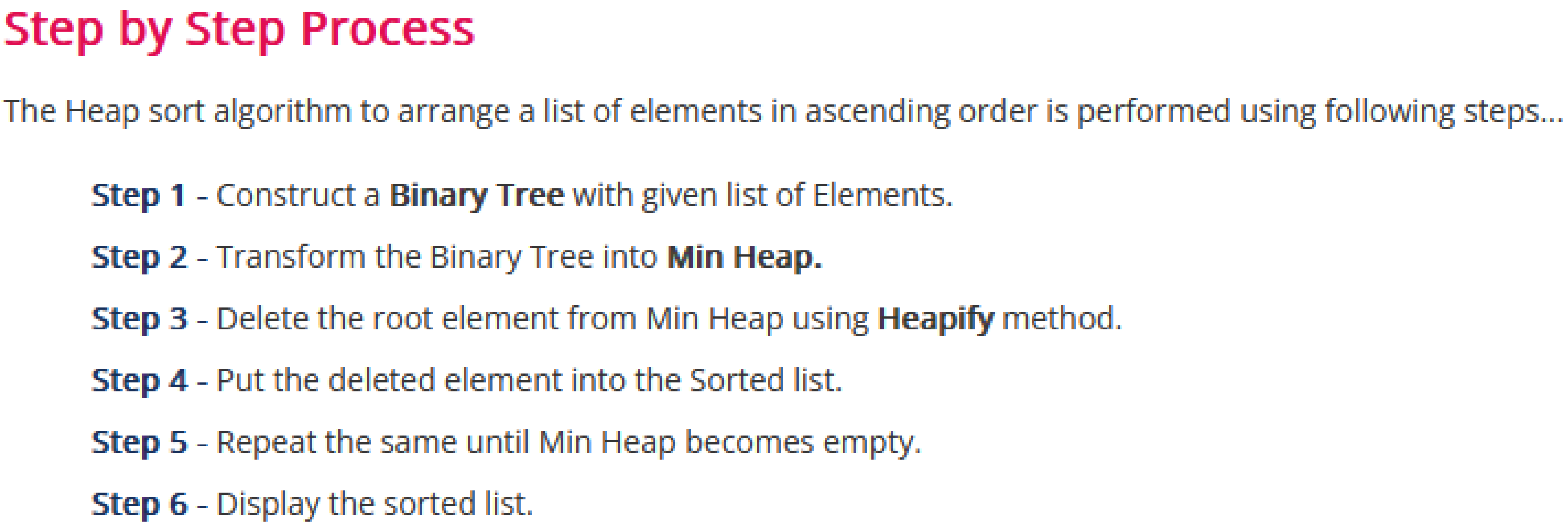


Deletion

- Heapify



Heap Sort Algorithm



Step by Step Process

The Heap sort algorithm to arrange a list of elements in ascending order is performed using following steps...

Step 1 - Construct a **Binary Tree** with given list of Elements.

Step 2 - Transform the Binary Tree into **Min Heap.**

Step 3 - Delete the root element from Min Heap using **Heapify** method.

Step 4 - Put the deleted element into the Sorted list.

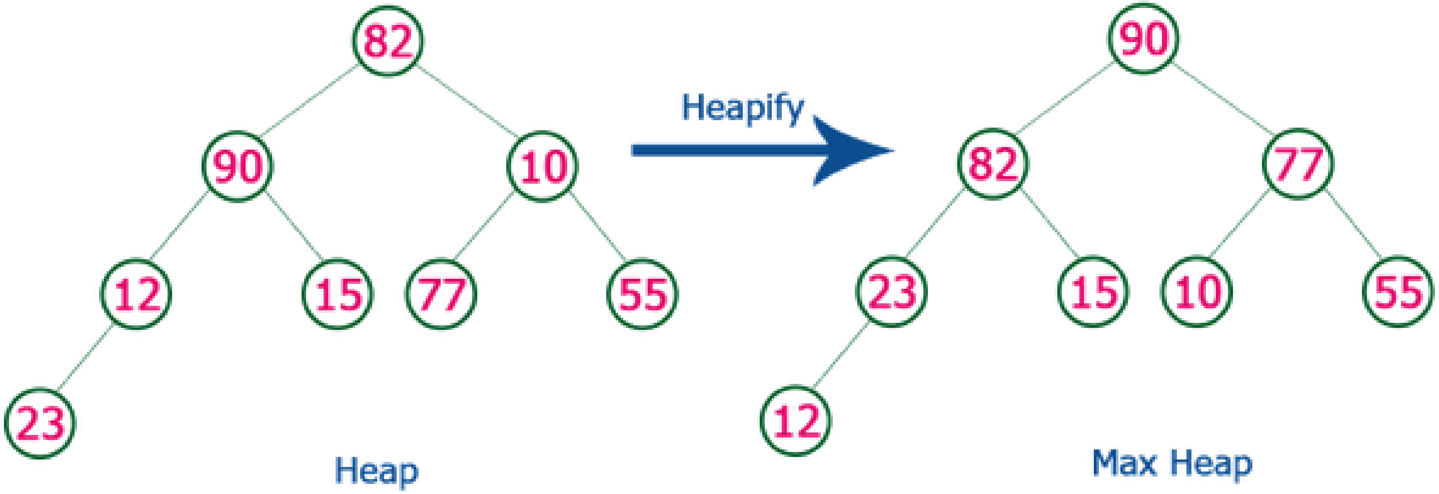
Step 5 - Repeat the same until Min Heap becomes empty.

Step 6 - Display the sorted list.

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

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

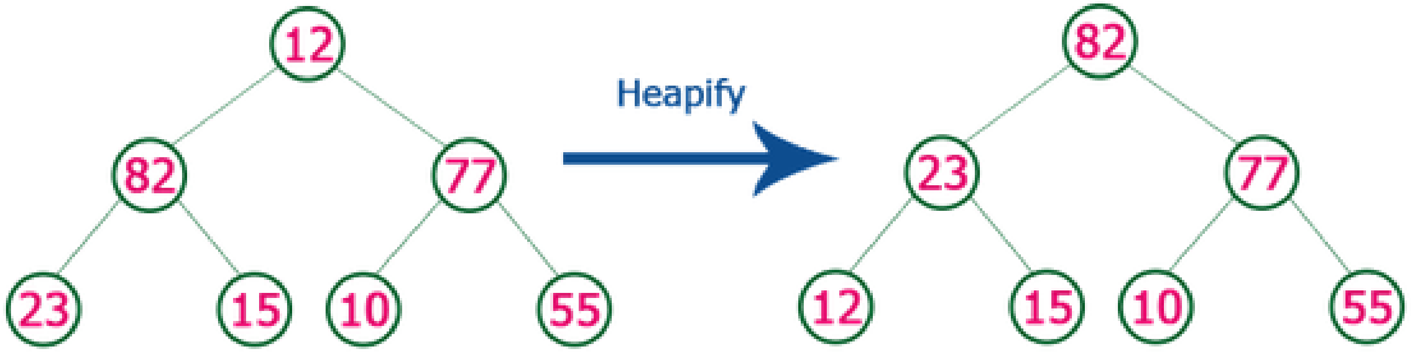
Step 1 - Construct a Heap with given list of unsorted numbers and convert to Max Heap



list of numbers after heap converted to Max Heap

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

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.



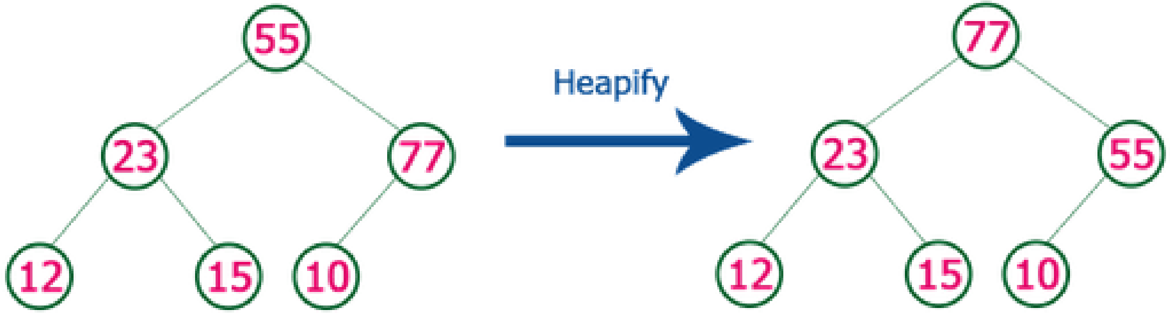
Heap after 90 deleted

Max Heap

list of numbers after swapping 90 with 12.

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

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.



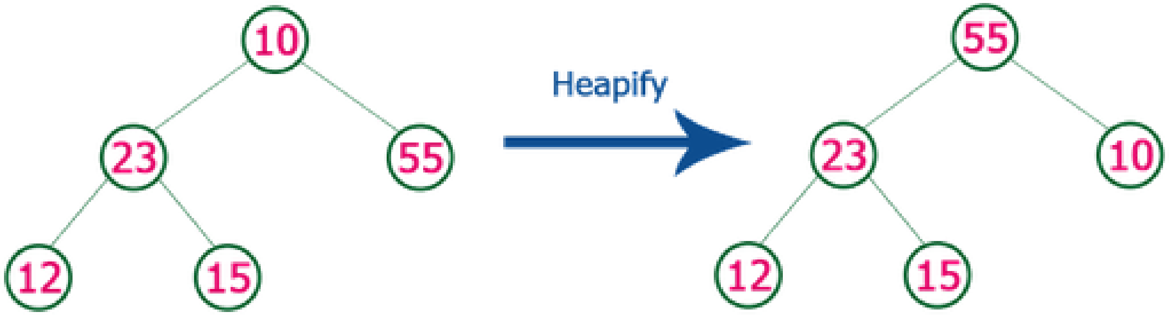
Heap after 82 deleted

Max Heap

list of numbers after swapping 82 with 55.

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

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.



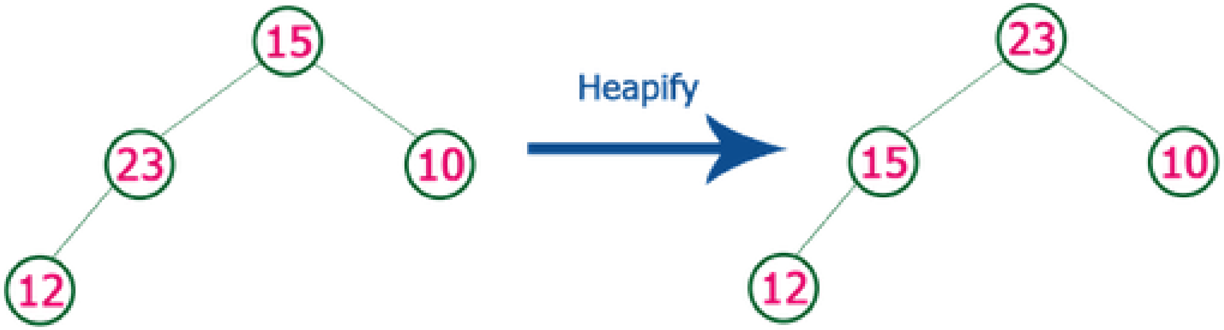
Heap after 77 deleted

Max Heap

list of numbers after swapping 77 with 10.

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

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.



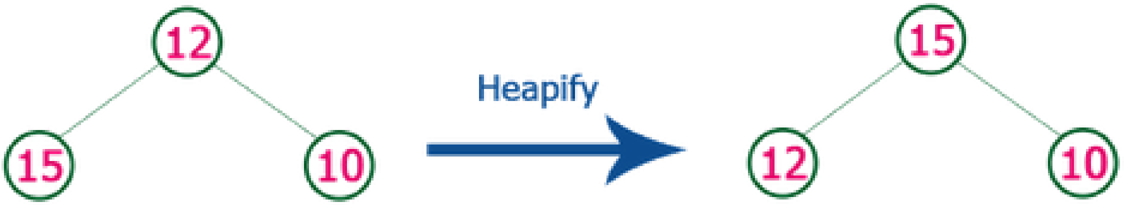
Heap after 55 deleted

Max Heap

list of numbers after swapping 55 with 15.

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

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.



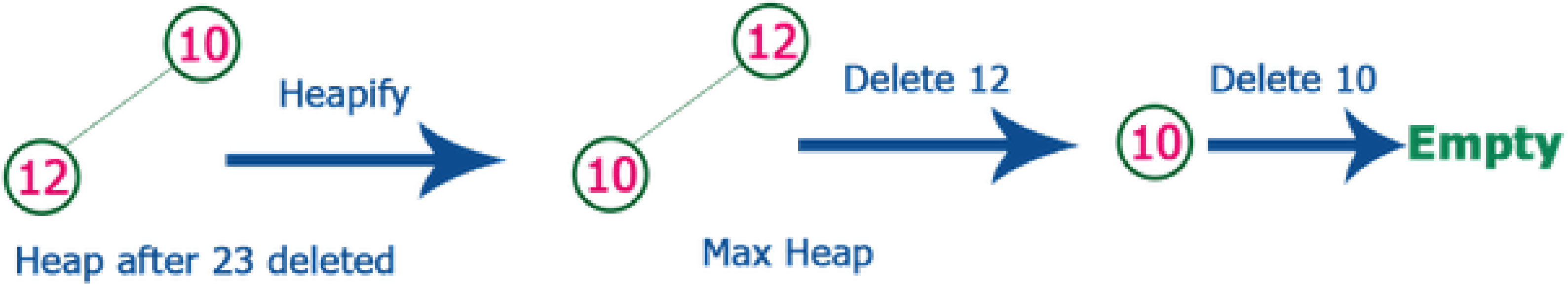
Heap after 23 deleted

Max Heap

list of numbers after swapping 23 with 12.

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

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