DATA STRUCTURE

WEEK 13

Contact Details:

Email: sobia.iftikhar@nu.edu.pk

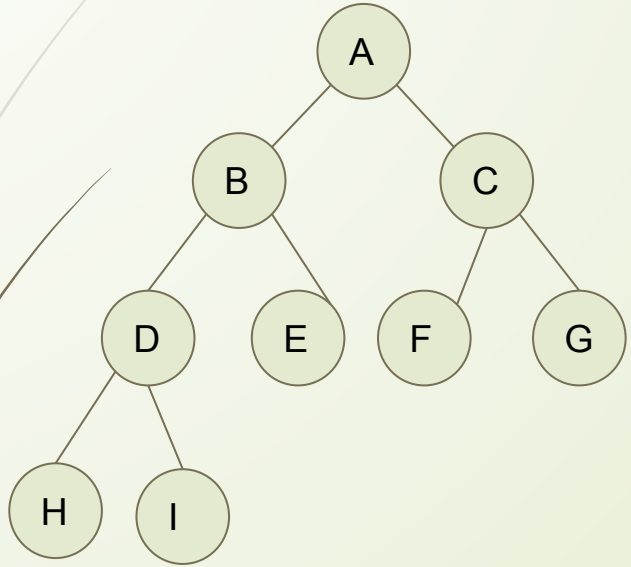
GCR:

Content

Heap

- Array Representation of BT
- Complete Binary Tree
- Heap
- Insert & Delete
- Heap Sort
- Heapify
- Priority Queue

Array representation of binary tree (Sequential Representation)



Case -01 if i start from 0

- If the index of any element in the array is i,
- element in the index 2i+1 will become the left child
- element in 2i+2 index will become the right child.
- the parent of any element at index i is (i-1)/2.

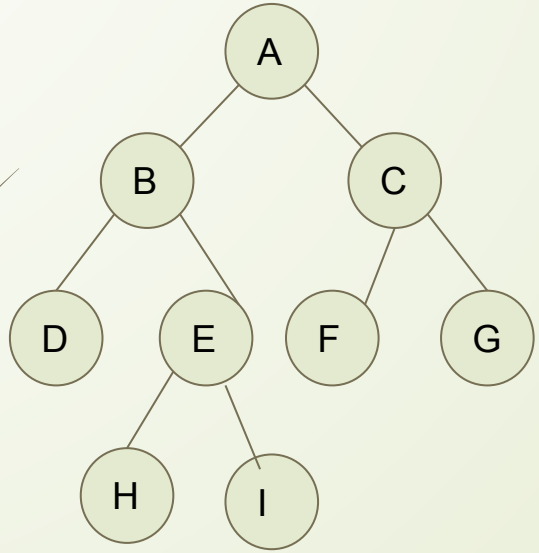
Case -02 if i start from 1

- If the index of any element in the array is i,
- element in the index 2*i will become the left child
- element in 2i+1 index will become the right child.
- the parent of any element at index i is (i)/2.

A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9

Array representation of binary tree (Sequential Representation)



Case -01 if i start from 0

- If the index of any element in the array is i,
- element in the index 2i+1 will become the left child
- element in 2i+2 index will become the right child.
- the parent of any element at index i is (i-1)/2.

Case -02 if i start from 1

- If the index of any element in the array is i,
- element in the index 2*i will become the left child
- element in 2i+1 index will become the right child.
- the parent of any element at index i is (i)/2.

A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9

Heap

Heap is special type of tree base data structure , which tree is complete binary tree

It has two types:

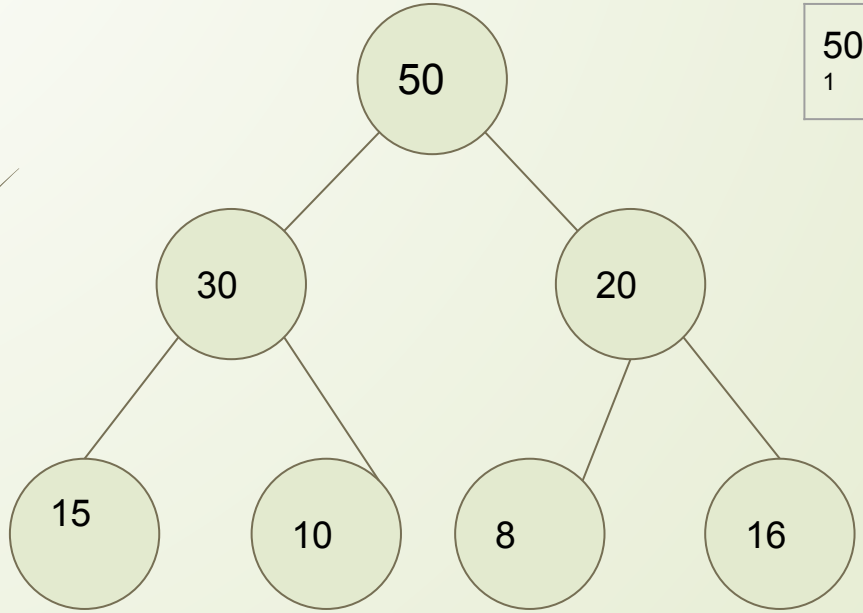
1. Min-heap
2. Max-heap

Construction of Heap

Heapify Mehtod O(n)

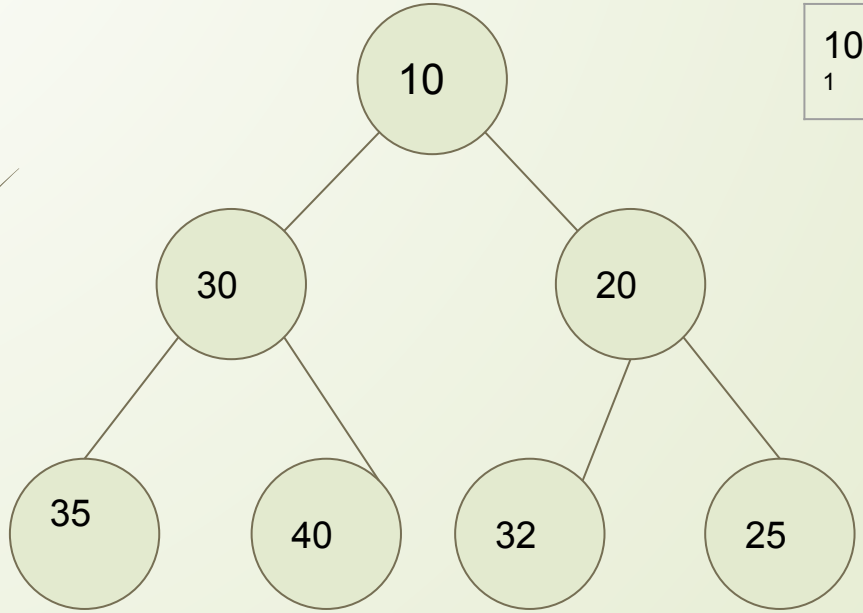
Insert key one by one O(nlogn)

Max-Heap



50 1	32 2	20	15	10	8	16

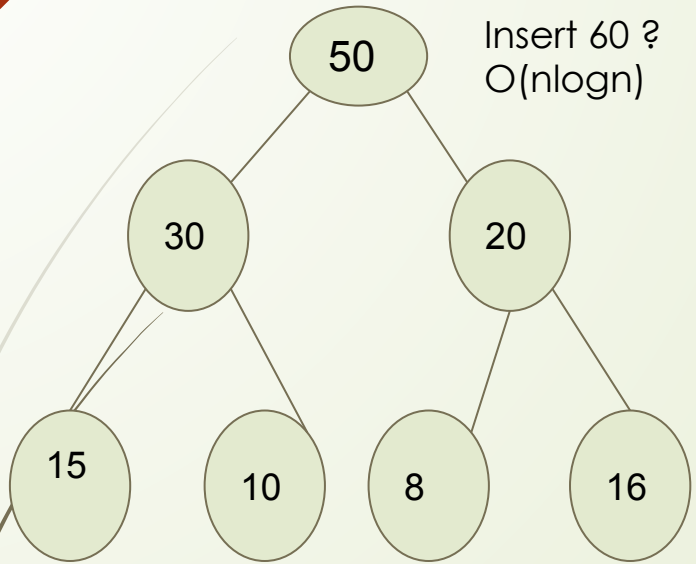
Min-Heap



10 1	30 2	20	35	40	32	25

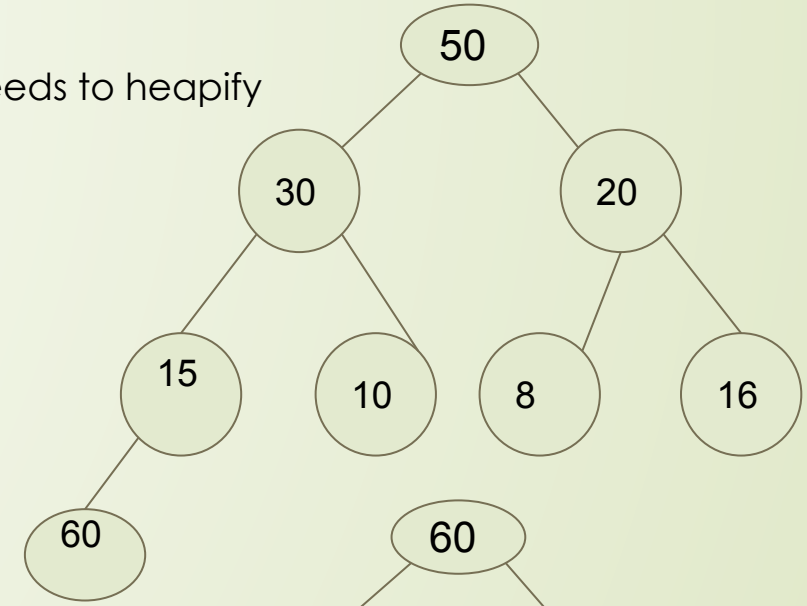
Max-Heap-Insertion

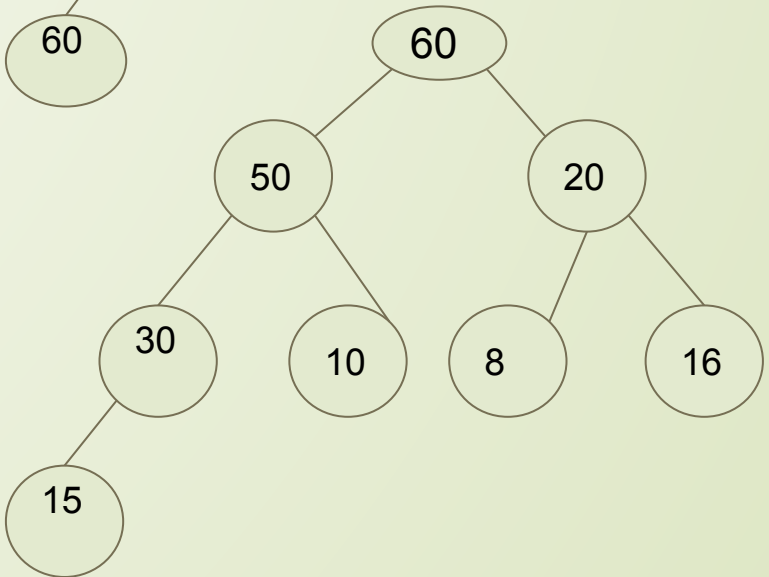
Insert 60 ?
O(nlogn)



50 1	32 2	20	15	10	8	16

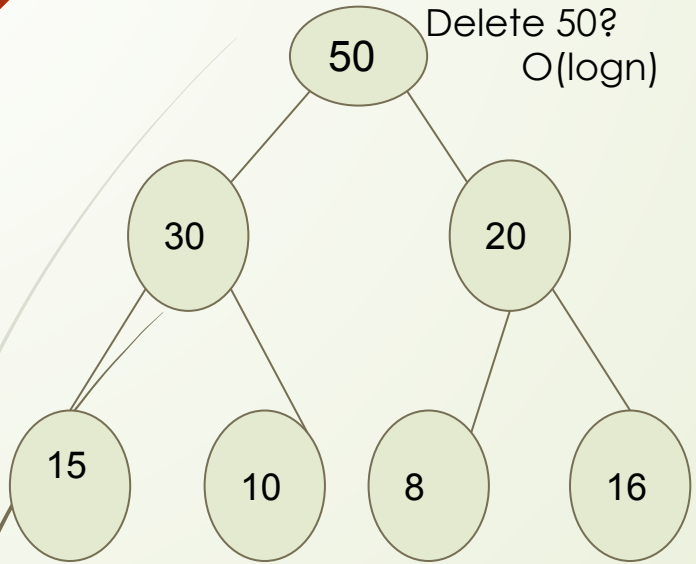
Needs to heapify





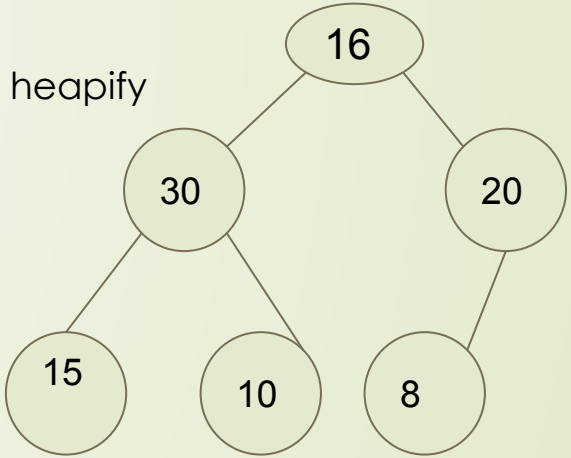
Max-Heap-Deletion

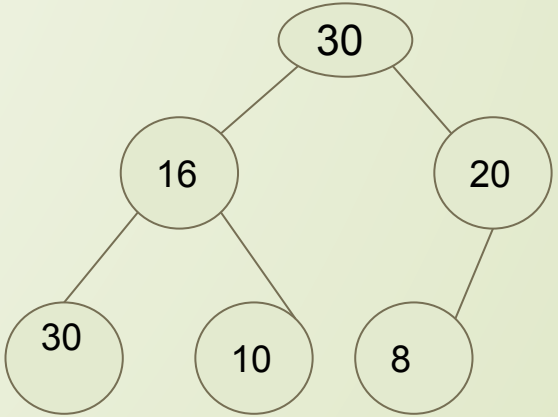
Delete 50?
O(logn)



50 1	32 2	20	15	10	8	16

Needs to heapify





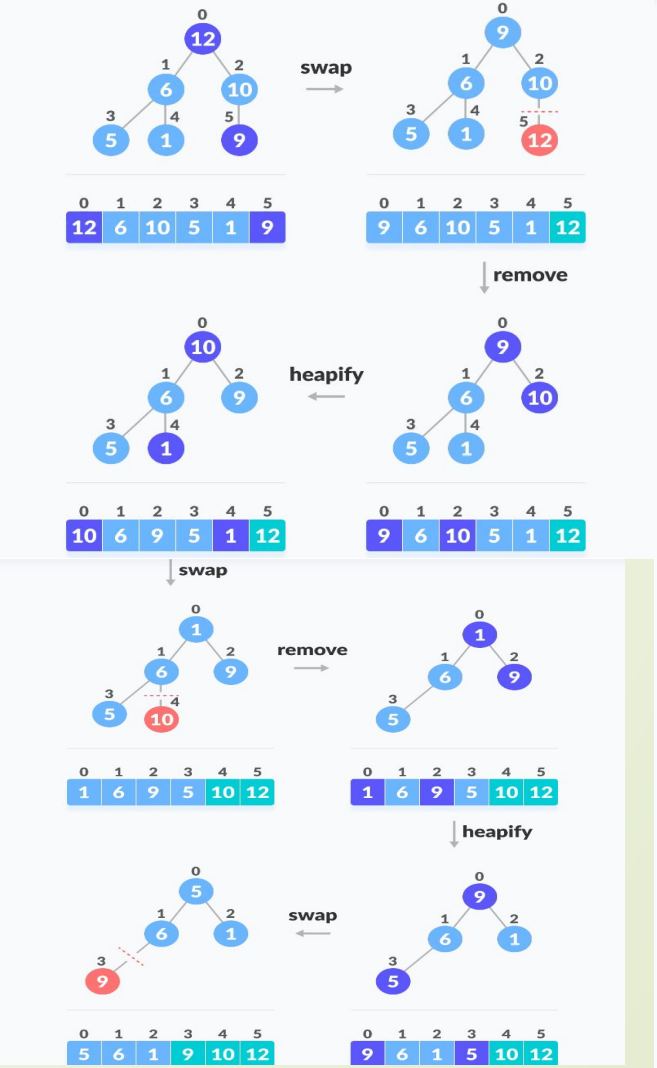
Heap Sort

First create heap

Delete On by element then data will become sort

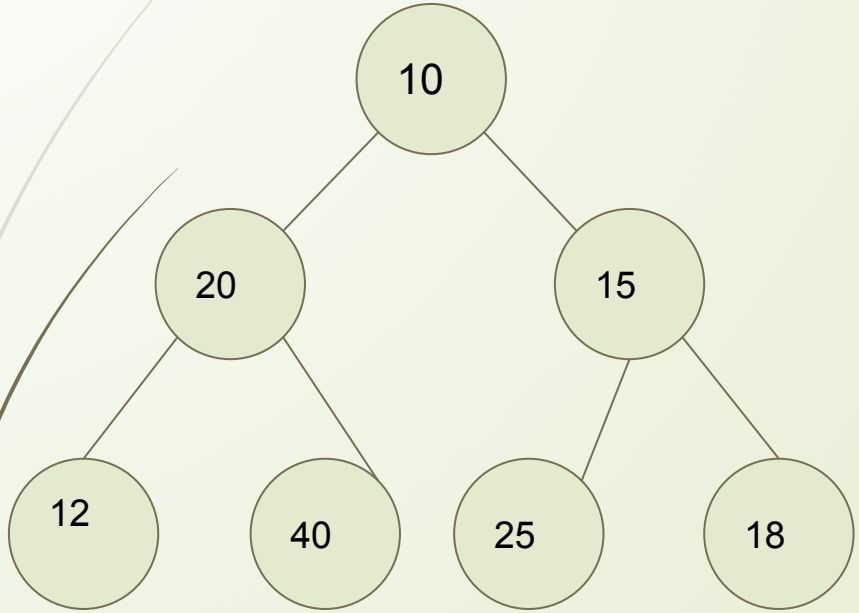
TIME COMP: O(NLOGN)

10	20	15	30	40

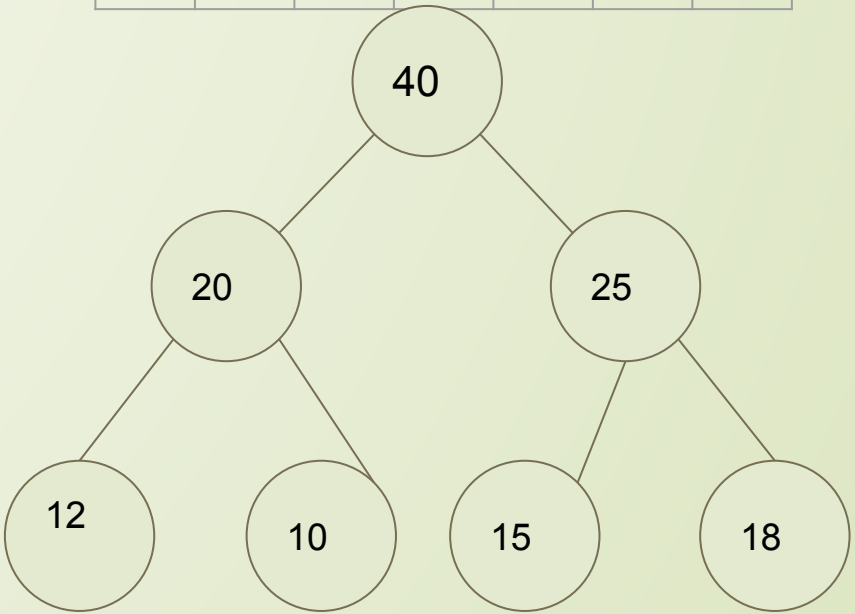


Second method to construct H-TREEHeapify

1. Insert at leaf and adjust upward direction [Left to Right]
2. Can we create heap tree by traversing from Right to left -0(n)



10 1	20 2	15	12	40	25	18



Build Max Heap from Array-O(n)

```
static void buildHeap(int arr[], int N)
	{
		int startIdx = (N / 2) - 1;
		for (int i = startIdx; i >= 0; i--) {
			heapify(arr, N, i);
		}
	}
```

```
static void heapify(int arr[], int N, int i)
	{
		int largest = i; // Initialize largest as root
		int l = 2 * i + 1; // left = 2*i + 1
		int r = 2 * i + 2; // right = 2*i + 2

		// If left child is larger than root
		if (l < N && arr[l] > arr[largest])
			largest = l;

		// If right child is larger than largest so far
		if (r < N && arr[r] > arr[largest])
			largest = r;

		// If largest is not root
		if (largest != i) {
			int swap = arr[i];
			arr[i] = arr[largest];
			arr[largest] = swap;

			// Recursively heapify the affected sub-tree
			heapify(arr, N, largest);
		}
	}
```

Max heap Function O(nlogn)

```
1
insert(A , N, value){
N = n+1;
A[n] = value;
i= n;
while(i > 1){
Parent = i/2;
if(A[parent]) < A[i])
        swap(A[parent], A[i])
        i= parent;
Else
        Return;
}
}
```

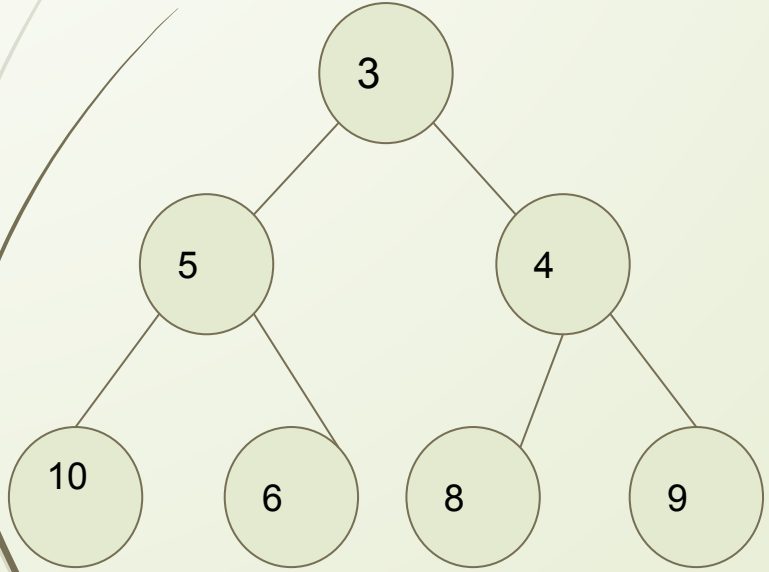
```
main(String[] arg)
   {
Cout <<"The Max Heap is ";

      MaxHeap maxHeap 14;
       maxHeap.insert(24);
      maxHeap.insert(12);
      maxHeap.insert(11);
      maxHeap.insert(25);
      maxHeap.insert(8);
      maxHeap.insert(35);
         maxHeap.print();
     System.out.println("The max val is "
                    +
maxHeap.extractMax());
   }
}
```

Priority Queue
Pq implemented using Heap ds due to time complexity

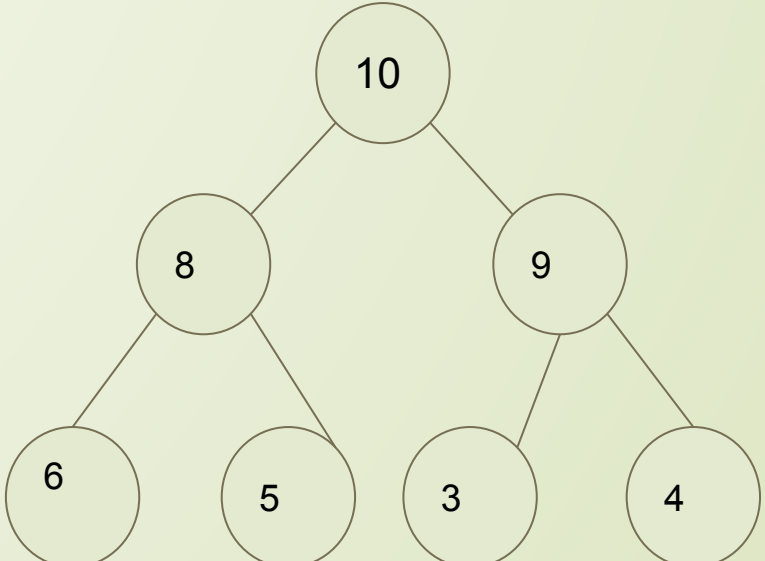
Small number as higher Priority Queue

8	6	3	10	5	4	9



Highest number as higher Priority Queue

8	6	3	10	5	4	9



Max into Min Heap or Min Heap into Max heap