## **LEC-11 Heap**

**Definition:**

* Binary Tree in which parent (in case of max heap) is greater than the children or less (in case of min heap) than children.
* **Properties of Heap:**
  + Structural property:
    - ACBT(a complete binary tree)/ CBT(complete binary tree)
  + Ordering Property:
    - Max-Heap
      * Parent > children
    - Min-Heap
      * Parent < children
  + Insertion in heap:
    - Adding values element-by-element
    - Add all values without any specific order and then convert that into heap(max-heap or min-heap)
  + E: g 14,24,12,11,25,8,35 (max heap)

**Heap:**

* A heap is a specialized tree-based data structure that satisfies the heap property.
* The heap property specifies a relationship between parent and child nodes, ensuring that each node has a value that is either greater than or equal to (in a max heap) or less than or equal to (in a min heap) the values of its child nodes.
* Heaps are commonly used to implement priority queues, which are fundamental in various algorithms like Dijkstra's shortest path algorithm and heap sort.
* In C++, heaps are typically implemented using arrays, where each element in the array corresponds to a node in the heap.

**Min Heap:**

A min heap is a type of heap where the value of each parent node is less than or equal to the values of its child nodes. This ensures that the minimum element is always at the root of the heap.

# **Example Min Heap:**

2

/ \

7 10

/ \ / \

20 8 15 13

In this min heap, the root node (2) has the smallest value, and the heap property is satisfied for all nodes.

**Sample Code - Creating a Min Heap:**

#include <iostream>

#include <queue>

int main() {

    std::priority\_queue<int, std::vector<int>, std::greater<int>> minHeap;

    minHeap.push(10);

    minHeap.push(5);

    minHeap.push(15);

    while (!minHeap.empty()) {

        std::cout << minHeap.top() << " ";

        minHeap.pop();

    }

    return 0;

}

**Max Heap:**

A max heap is a type of heap where the value of each parent node is greater than or equal to the values of its child nodes. This ensures that the maximum element is always at the root of the heap.

Example Max Heap:

20

/ \

18 10

/ \ / \

15 8 9 3

In this max heap, the root node (20) has the largest value, and the heap property is satisfied for all nodes.

**Sample Code - Creating a Max Heap:**

#include <iostream>

#include <queue>

int main() {

    std::priority\_queue<int> maxHeap;

    maxHeap.push(10);

    maxHeap.push(5);

    maxHeap.push(15);

    while (!maxHeap.empty()) {

        std::cout << maxHeap.top() << " ";

        maxHeap.pop();

    }

    return 0;

}

## **Heap Sort**

**Working of HeapSort :**

void MaxHeap::remove(int& a){

    //check is empty

    a=h[1];

    h[1]=h[currsize];

    currsize--;

    heapify(1);

}

void heapsort(int\* arr, int n){

    MaxHeap mh;

    mh.buildHeap(arr,n);

    for (int i=n-1;i>=0;i--){

        mh.removeMax(arr[i]);

    }

}

void main(){

    //input array

    //display array

    heapsort(arr,n);

    //arr=name of array

    //n=size of array

    //display array

}

//end of main