14/03/2024

Day 17 of DSA

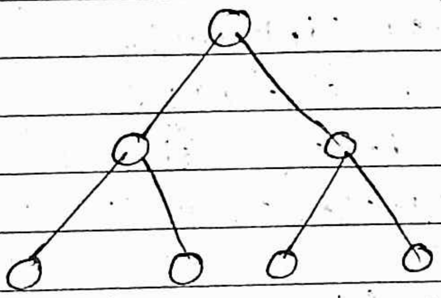
Heap.

CheckBox

- [x] Heap Implementat^n, Invertion, Deletion
- [x] Min Heap & Max Heap' Understanding heap propeties.
- [x] Implement baic heap operations, insertion, deletion, heapify.
- [x] Implement heap Sort Algorithm.

Heap:-

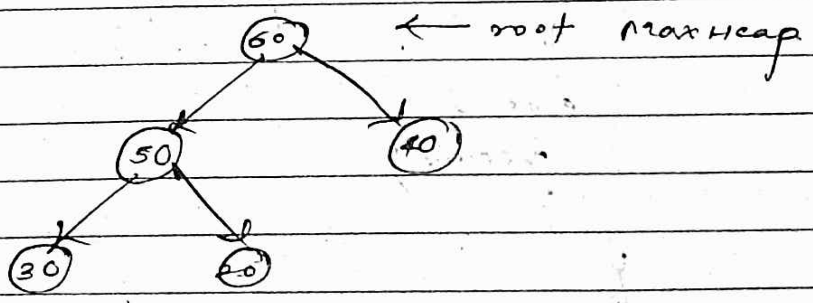
- Heap is Complete Binary Tree.



- All levels should be Filled.

- 1st Filled left to right Nodes.

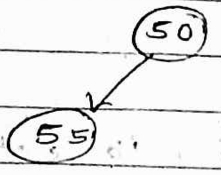
① Insertion:-

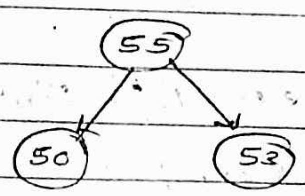


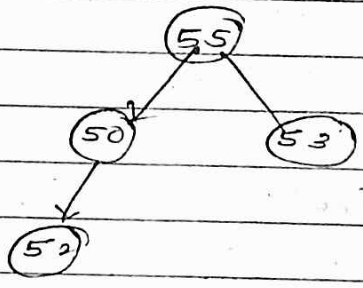
Node = ith Index
left child = 2 * i
Right child = (2 * i + 1)
parent = $\left(\frac{i}{2}\right)$

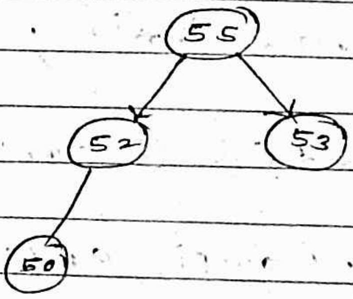
```
clais heap {
  public:
    int arr [100];
    int size =0;

    void insert (int val){
      size =size+1;
      int index= size;
      arr[index]= val;

      while (index >1){
        int parent = index/2;

        if( arr[parent] < arr[index])
        {
          swap (arr[parent], arr[index]);
          index = parent;
        }
        else {
          return;
        }
      }
    }
}
```

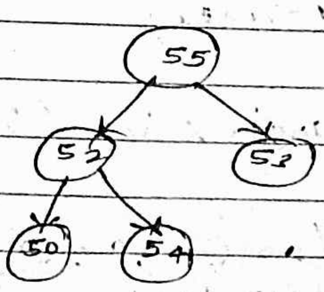
eg. 50, 55, 53, 52, 54

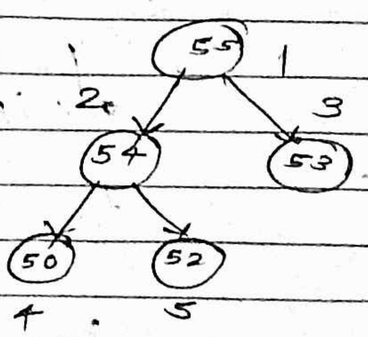
1


2


3


4


5


6


$\boxed{T \cdot C = O(\log n)}$

X	51	54	53	50	52
0	1	2	3	4	5

* Deleting Node:-

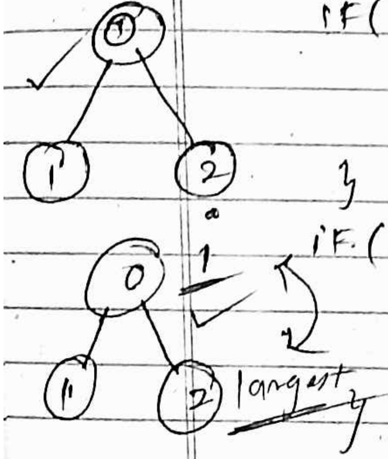
```
void delete() {
    if (size==0)
    {
        cout<<'nothing to delete'<<endl;
        return;
    }
    // Put last element to First Node
    arr[1] = arr[size];
    // Remove last element
    size--;

    int i=1;
    while (i<size)
    {
        int leftIndex = 2*i;       } formulae
        int rightIndex = 2*i+1;    }

        if(leftIndex<size && arr[i] < arr[leftIndex
        {
            swap(arr[i], arr[leftIndex]);
            i = leftIndex;
        }
        else if(rightIndex<size && arr[i] < arr[rightIndex])
        {
            swap(arr[i], arr[rightIndex]);
            i = rightIndex;
        }
        else {
            return;
        }
    }
}
```

# Heapify Algorithm :-

For Creating Heap.

```
void heapify(int arr[], int n, int i){
    int largest = i;
    int left = 2*i;
    int right = 2*i+1;

    if(left<n && arr[largest] < arr[left]){
        largest = left;
    }
    if(right <n && arr[largest]< arr[right]){
        largest = right;
    }

    if(largest != i){
        swap(arr[largest], arr[i]);
        heapify(arr, n, largest);
    }
}
```

left Index
right Index



Heap Sort:-

```
void heapSort(int arr[], int n)
{
    ✓ int size = n;
      while(size > 1) {
          swap(arr[size], arr[1]);
          heapify(arr, size, 1);
      }
}
```

Swap last element with root.

Use Heapify to create Heap.

~~*~~ Priority Queue :

$$\begin{tikzcd}
 & \text{Priority Queue} \arrow[dl, no head] \arrow[dr, no head] & \\
\text{Max Heap} & & \text{Min-Heap}
\end{tikzcd}$$

#maxheap- #include<queue>.

syntax

```
priority_queue< int> pq;
```

1. push()
2. pop()
3. top()
4. size().
5. empty().

#minHeap-

```
priority_queue <int, vector <int >, greater<int>>
                                          minheap;
```

1. push()
2. pop()
3. top()
4. size()
5. empty()