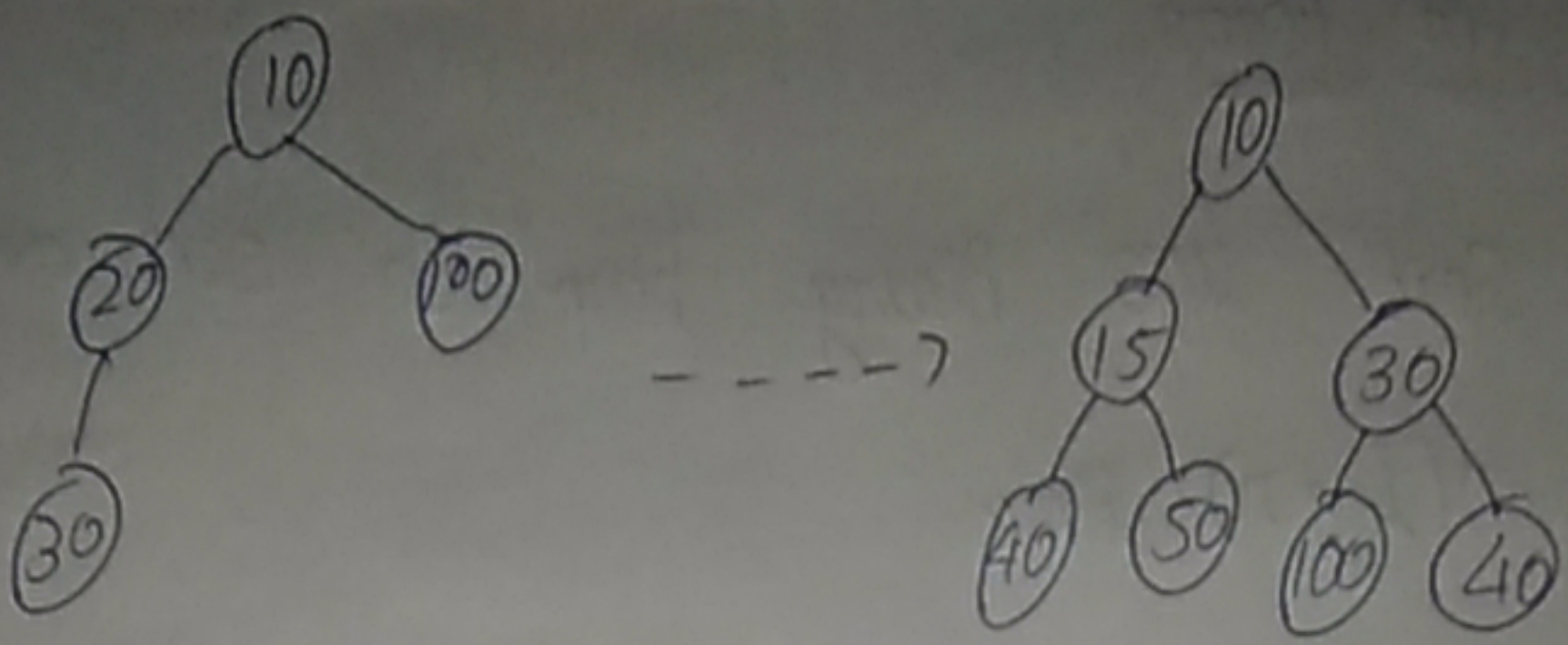
Day 27

Binary Heap

A binary heap is a binary tree with following properties.

* it is a Complete tree (All levels are Completely filled except possibly the last level and the last level has all keys as left as possible). This property of binary heap makes them suitable to be stored.

* A binary heap is either min Heap or Max heap. In a min binary tree Heap, the key at root must be minimum among all keys present in binary heap. The same property must be recursively true of all nodes in binary tree.



A binary heap is a Complete binary ~~heap~~ tree. A binary heap is typically represented as an array.

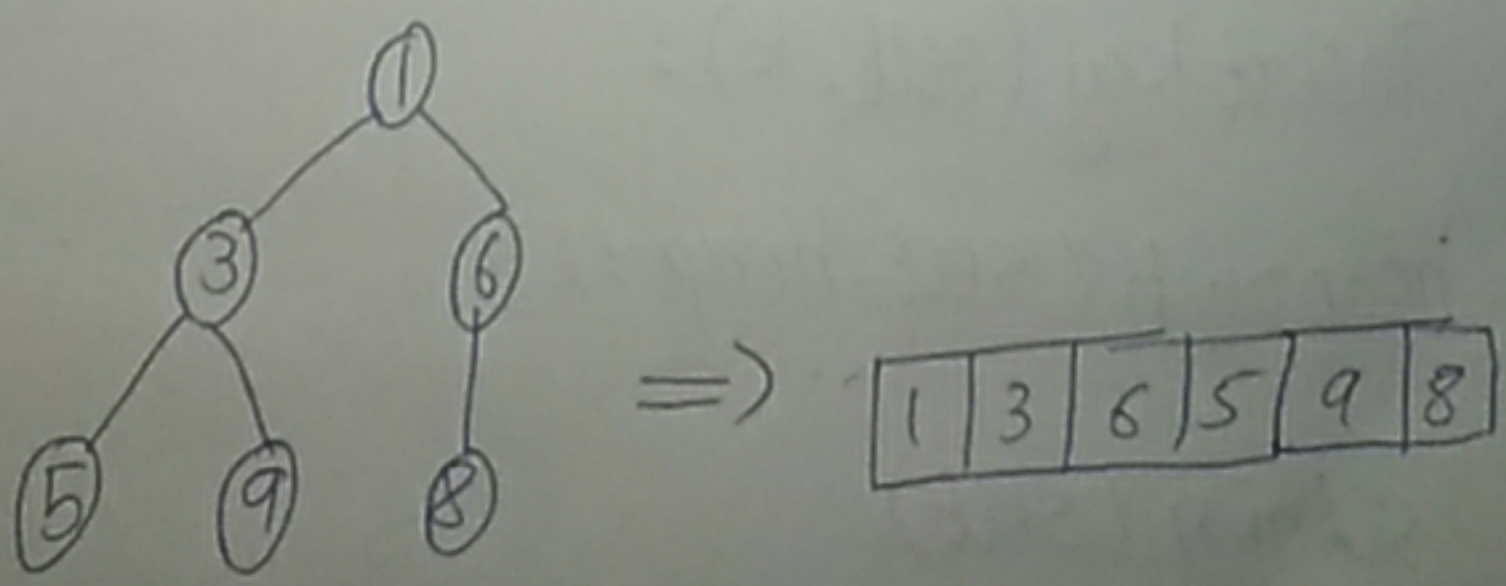
* The root element will be at Arr[0]

* Below table shows indexs of other nodes for the i^{th} node.

Arr[(i-1)/2] — Return the parent node

Arr[(2*i)+1] — Return the left node

Arr[(2*i)+2] — Return the right node



Application Of Heaps

* Heap Sort uses Binary heap to sort an array in O(log n) time

* priority Queue can be efficiently implemented using binary heap.

Implementation :

```
from heapq import heappush, heappop, heapify

class Minheap:

    def __init__(self):

        self.heap = []

    def parent(self, i):

        return (i-1)/2

    def insertkey(self, k):

        heappush(self.heap, k)

    def getmin(self)

        return self.heap[0]
```

```
def extractmin(self):
    return heappop(self.heap)

def decreasekey(self, i, new_val):
    self.heap[i] = new_val
    while (i != 0 and self.heap[self.parent(i)]
                                > self.heap[i])
        self.heap[i], self.heap[self.parent(i)]
            = (self.heap[self.parent(i)], self.heap[i])

def deletekey(self, i)
    self.decreasekey(i, float("-inf"))
    self.extractmin()
```

Input:

```
heap = minheap()
heap.insertkey(3)
heap.insertkey(2)
heap.insertkey(1)
heap.insertkey(15)
heap.insertkey(4)
print(heap.extractmin())
print(heap.getmin())
heap.decreasekey(2,1)
```

output

=>

```
2
4
1
```