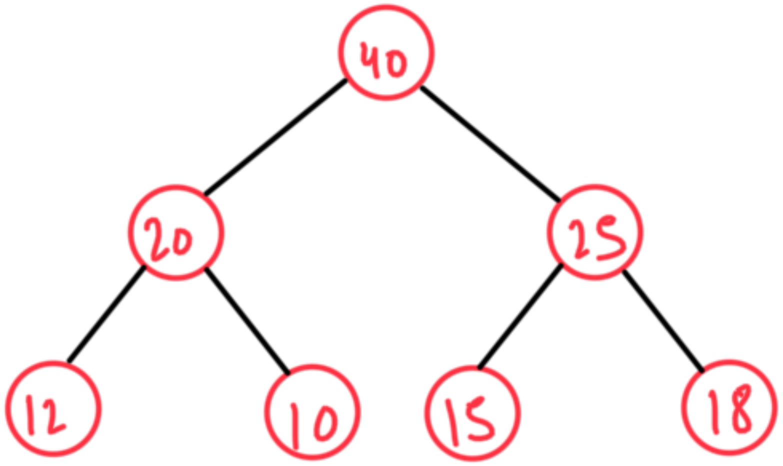
7 Heap Sort

Given array to sort:

10	20	15	30	40
0	1	2	3	4

Step 1: Use heapify to convert the input array into a heap.



	40	20	25	12	10	15	18
0	1	2	3	4	5	6	7

Step 2: Call the delete function and put the max item at the end of same array. Do this n times.

[10, 12, 15, 18, 20, 25, 40]

```
def heap_sort(arr[1...n]):
    heapify(arr[1...n]) — n
    for(i=n to 1): ——— n
        arr[i] = Extract_max(arr[1...i]) — log n
```

arr [i] = Extract_max (arr [1...i]) — log n

Time:

$$T(n) = n + n \cdot \log n = O(n \log n)$$

Best-Case: $O(n)$, when all elements have same value. This means extract_max is done in $O(1)$ as there is no swapping.

Worst/Avg-Case: $O(n \log n)$

Space: $O(1)$ In-place: Yes, aux-space is $O(1)$.

Stable: No, because the relative order of elements is lost during the extracting max and placing it at the end of array.

Ex) arr: $[2_a, 2_b, 2_c] \xrightarrow{heapify} [2_a, 2_b, 2_c]$

$\xrightarrow{get\_max} [2_a\ 2_b, 2_c]$ relative order is NOT maintained!

Online: No, because Heap sort needs to know the entire arr[1...n] during heapify.

Code:

```
def heapsort(array):
    heap = MaxHeap()
    heap.heapify(array)

    n = len(array)
    for i in range(n, 0, -1):
        array[i - 1] = heap.extract_max()

    return array
```