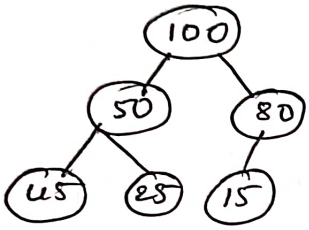
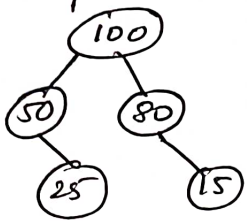
Heap sort : A Heap is a complete Binary tree or almost complete Binary tree satisfying the following two conditions

* All its levels are full Except possibly the last level if the last level is not full, all the nodes should be filled only from left to right
* The key at each node should be greater than or = to key of its childrens (max heap)



Heap



Not Heap

→ heap sort is a sorting technique used to arrange numbers in ascending or descending order

This sorting technique uses creating a max-heap so that root is having greater value than its childrens.

The same condition must be true for each subtree of a Heap

→ Heap sort technique consists of 2 phases

* Heap creation phase ⇒ Heap can be created using Bottom up approa
* Sorting phase ⇒ Consists of 2 steps.
  - → Exchange the root item with last Element of Heap
  - → decrement the size of Heap by 1 & Heapify. (Heap construction)

In min-heap property → The value of each node, or child is greater than or equal to its parent. (with min value at the root node)

In max-heap ⇒ The value of each node or child is less than or = to the value of its parent with max value at the root node.

Heapsort Algorithm (n, a[])

i/p:- list of numbers a[]

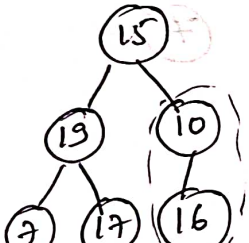
o/p: sorted list of numbers

```
Bottom_up_Heapify (n, a)    // Construct a Heap
  for i = n-1 down to 0
      Exchange (a[0], a[i])
      Bottom_up_Heapify();
  End for
```

The time complexity of Heapsort Algorithm is $O(n \log n)$

Construct the Heap for the list by Bottom-up approach: (MAX heap) (ascending order)

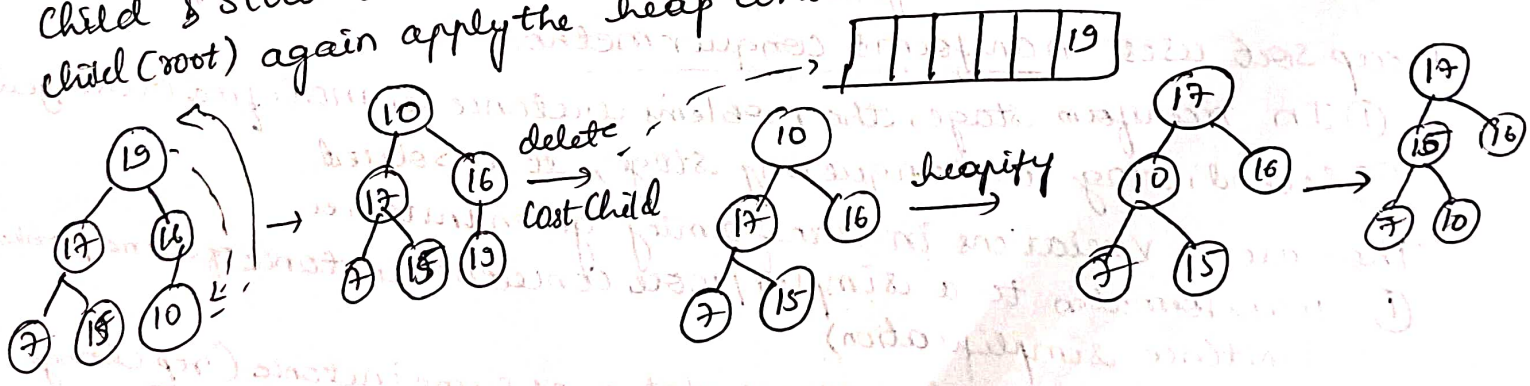
15, 13, 10, 7, 17, 16

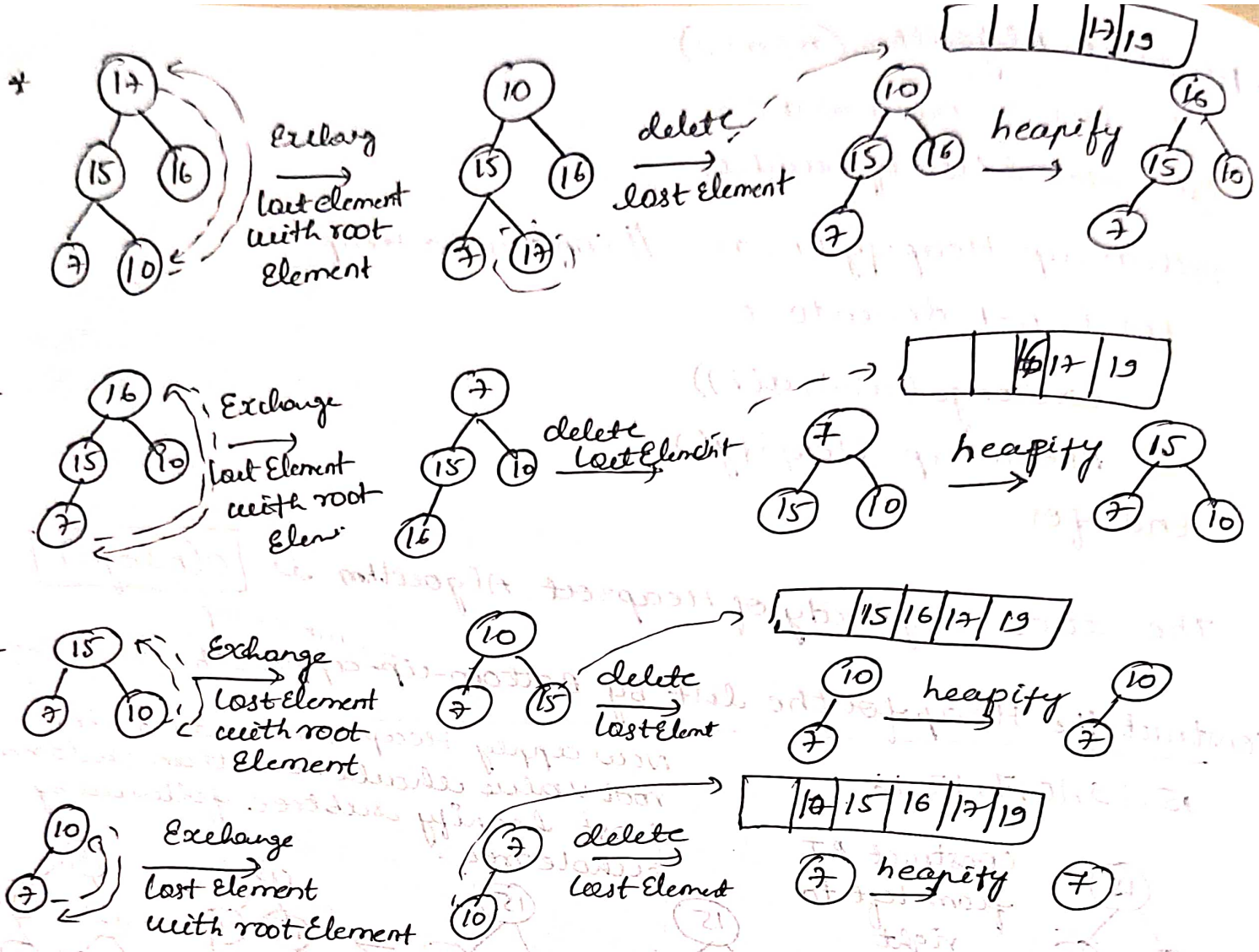


Construct BT from left to right

Now apply Heap construction rule, root value should be > than children first heapify subtree followed by whole tree.

Once after completion of Heap construction (Heapify), In Every individual step, Exchange root value with last child value & delete the last child & store the value in array. Once after deleting the last child (root) again apply the heap construction (Heapify)





* (7) → delete the last node & place it on array

7	10	15	16	17	19

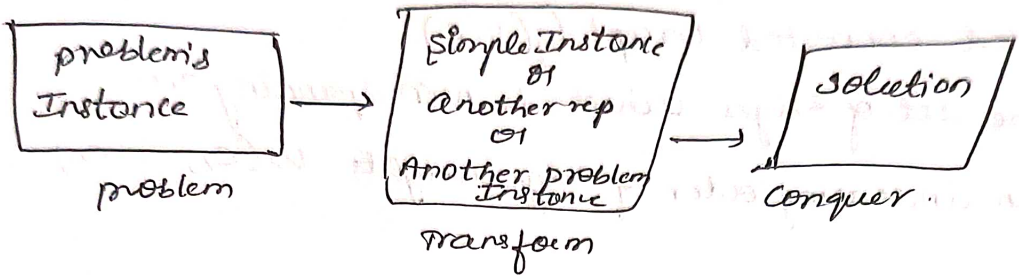
* Sort the below list using heap sort 3, 2, 4, 1, 6, 5 (Assignment)

Heap sort uses Transform conquer method.

(1) In transform stage, the problem's instance is modified or changed
(2) Second stage is conquering stage, it is solved

There are 3 variations in transforming given instance.

(1) transformation to a simpler/more convient instance of same problem (instance simplification)
(2) Transformation to diff representation of same instance (rep change)
(3) Transformation to an instance of diff problem for which an alg is already available (problem reduction)


problem's
Instance
problem
Simple Instance
of
another rep
of
Another problem
Instance
Transform
solution
Conquer