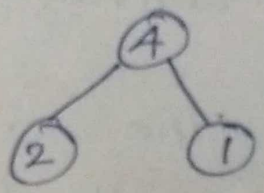
BINARY HEAP

Binary heap is a complete binary tree where each node has at most two children and it satisfies heap order property

Types of Heap

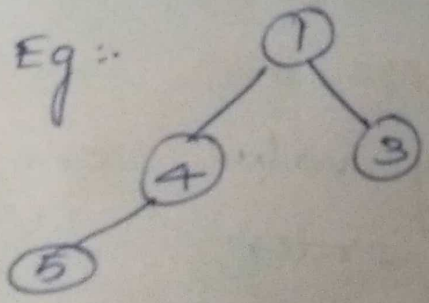
1) Max heap — it is a complete binary tree in which each value of the parent node is greater than or equal to the values in its child nodes

Eg:-



2) Min Heap — Here the key present in the parent node should be less than or equal to the key in the child node.

Eg:-

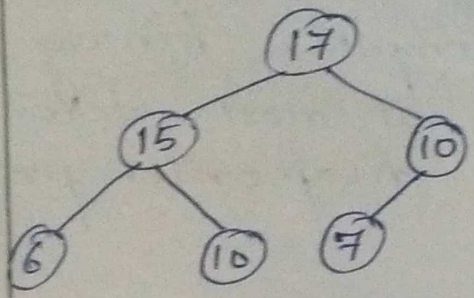


Heap order property

1) Min ~~Max~~ heap property — Value of each node is greater than or equal to its parent
2) Max heap property — Value of each node is smaller than or equal to its parent.

Binary Heap Representation

It is represented by storing its level order traversal in an array.



Element	~~7~~	17	15	10	6	10	7
Index (i)	~~0~~	1	2	3	4	5	6

↳ Sentinel node

Here, the root is not stored in the Zeroth index. Instead it is stored in the 1st index (or as the second item). The Zeroth index of the array is left blank for the convinience of implementation

Sentinal node:- The 0^{th} index of the array is called sentinal node. It helps avoiding coding errors with pointers.

For an element at i^{th} index in array

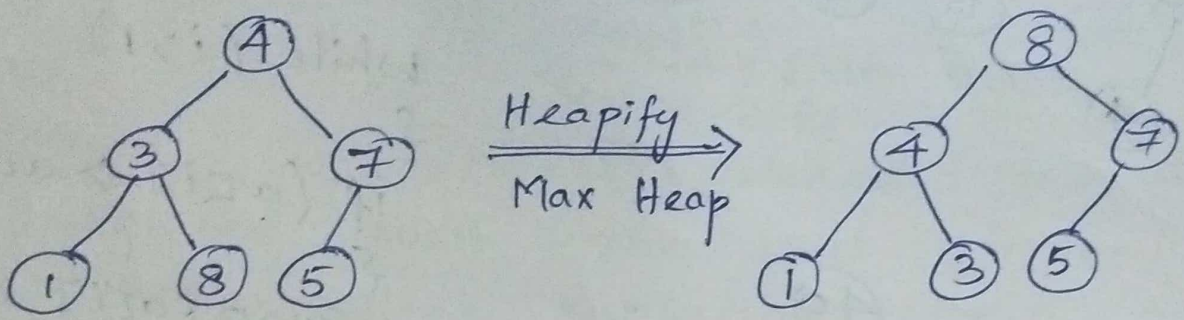
* Parent is located at $\lfloor i/2 \rfloor$ index
* Left child is located at $2i$ index
* Right child is located at $2i+1$ index

Heap tree construction

1) Insert Key value one by one in the given order.
2) Apply Heapify method to form the heap.

Eg:- Construct a heap with the following elements. 4, 3, 7, 1, 8, 5

Initial Binary Complete tree



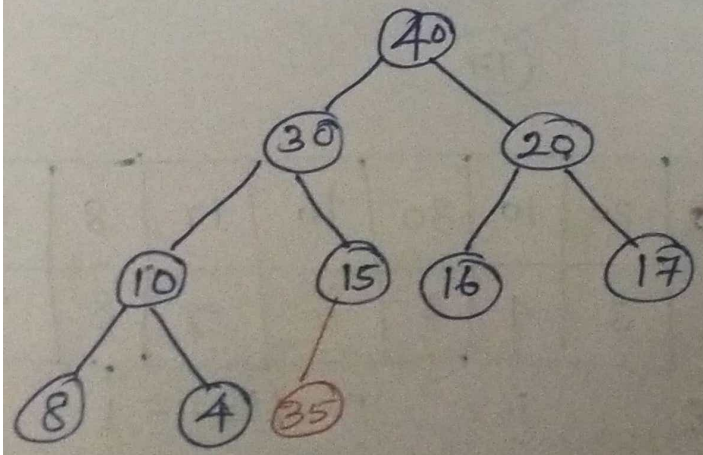
Heapify — It is a method to rearrange the heap to sustain heap order property.

Basic Heap Operations

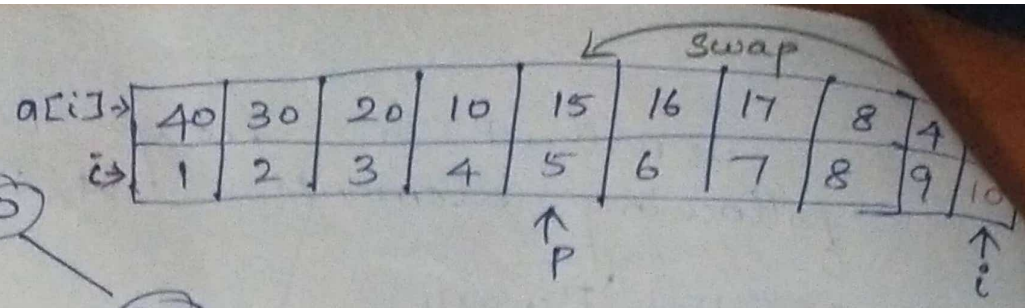
- Insert
- Delete
- Find min / Find max

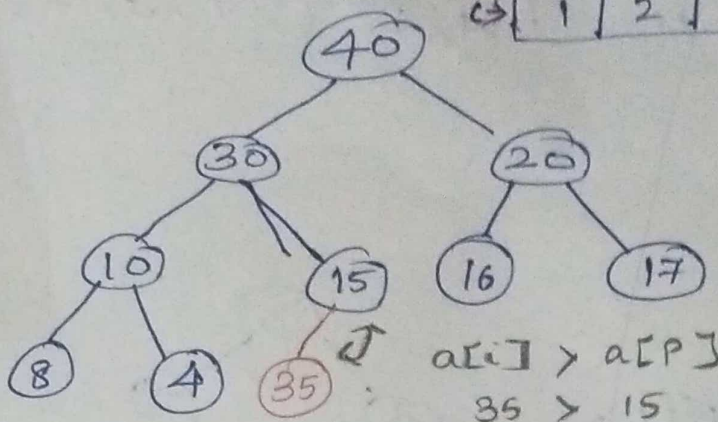
Insert

Insert 35 into the following Heap (Max)



40	30	20	10	15	16	17	8	4
1	2	3	4	5	6	7	8	9



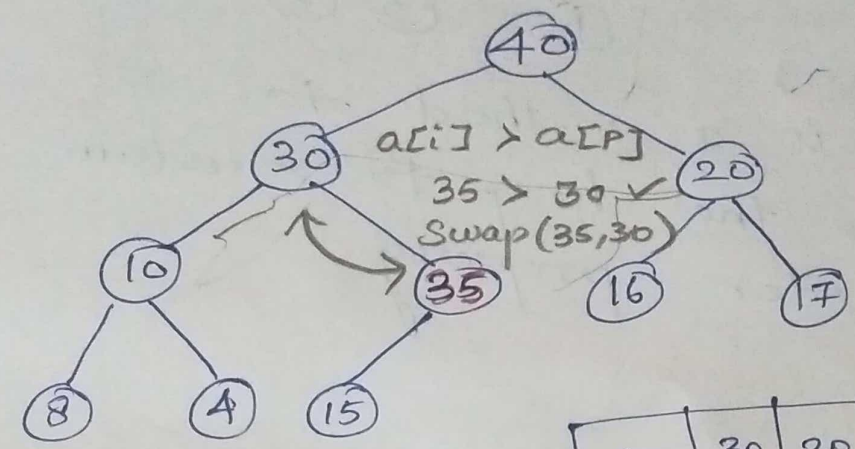


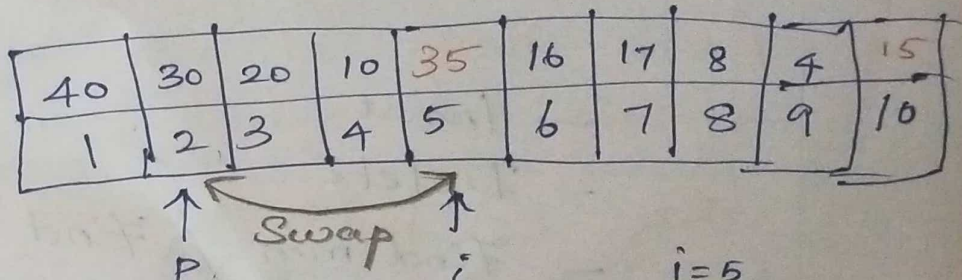
a[i] > a[P]
35 > 15 ✓
swap (35, 15)

Here, i = n (n=10)

$P = \lfloor i/2 \rfloor$

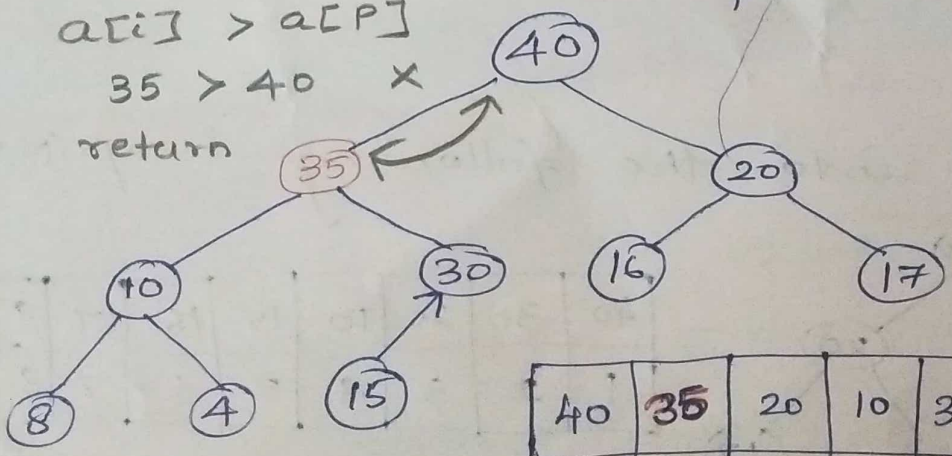
```
while (i>1)
{
  if (a[i] > a[P])
  {
   swap (a[i], a[P])
  }
  else
   return
}.
```

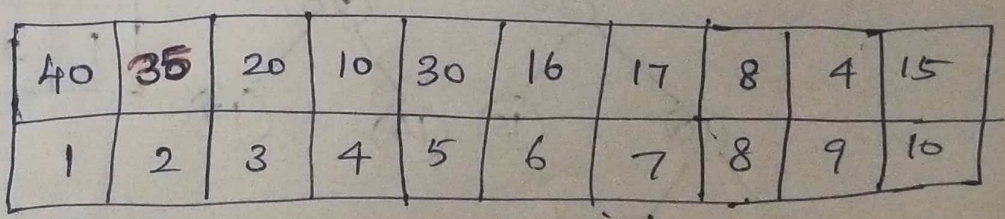




i = 5
$P = \lfloor i/2 \rfloor = 2$

a[i] > a[P]
35 > 40 ✗
return





i = 2 $P = \lfloor i/2 \rfloor = 1$

No swap

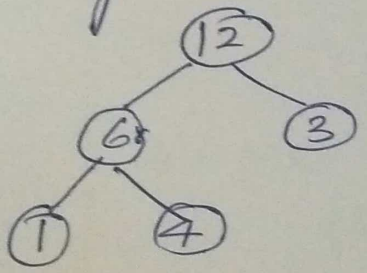
When i = 1, the algorithm stops.

Here, as the value is inserted, it is compared with its parent in the upper level. As we go up, the insert process is called "Perculate up"

Deletion

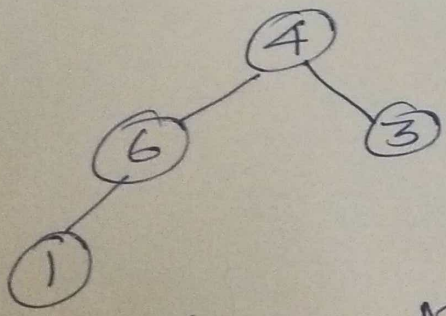
In case of delution, be it a max heap or mix heap, the root element is always deleted. The root is replaced with the last element in the heap. After replacing, check if heap order property is satisfied. Else apply Heapify method to set it.

Eg:- Initial max Heap.



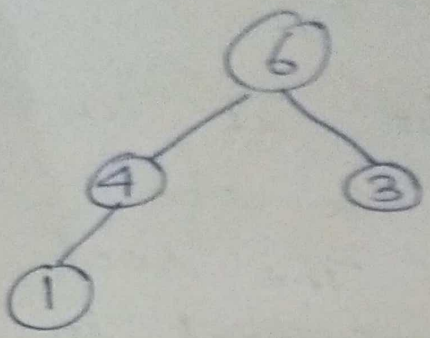
12	6	3	1	4
1	2	3	4	5

The element to be deleted is 12.
It is replaced with 4.



4	6	3	1
1	2	3	4

Here, Max - Heap order property is not satisfied. Swap (4) with the maximum of its child node which is (6)



6	4	3	1
1	2	3	4

Final Max - Heap ↑