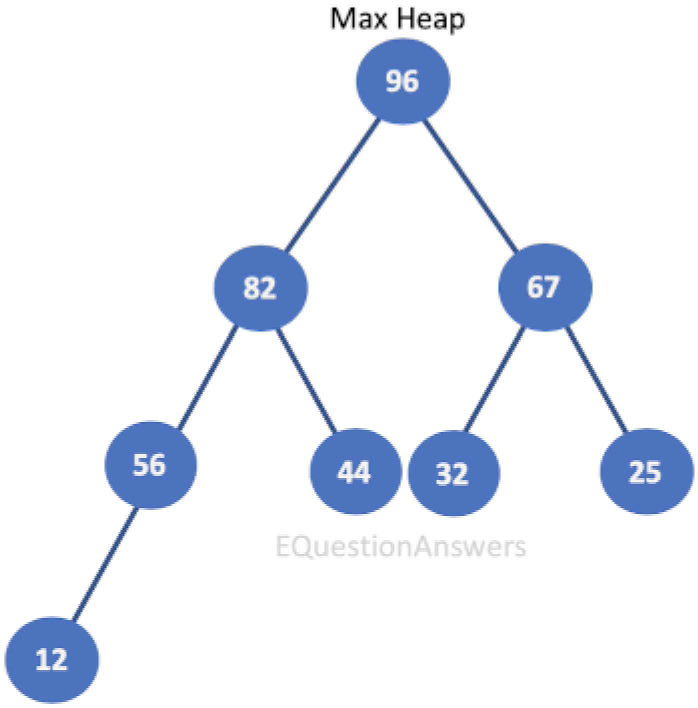
Heap Sort

Heap Sort: Heap sort is a comparison-based sorting technique based on Binary Heap data structure.

Heap Data Structure: Heap is a special binary tree based data structure. A binary tree is said to follow a heap data structure if it is a complete binary tree. There are two types of heap structure. Max heap and min heap.

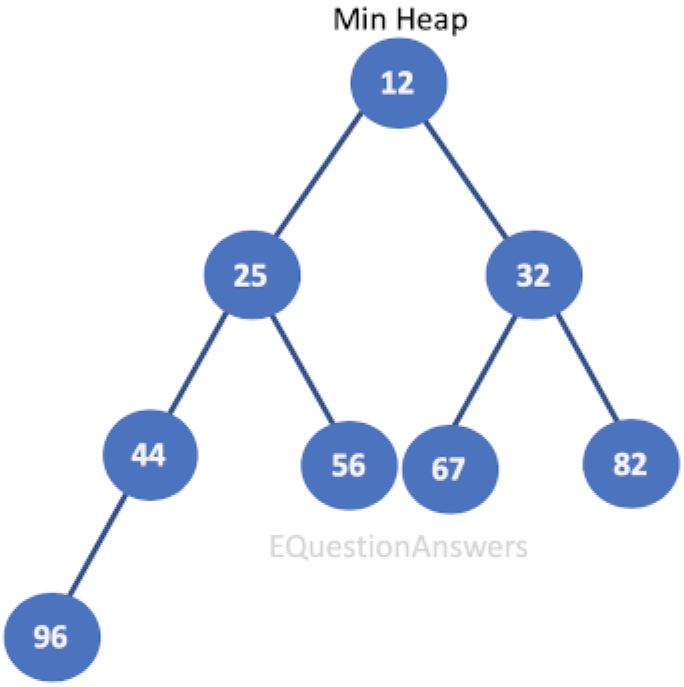
i. **Max Heap/Descending Heap:**

Max heap is a heap structure where parent element is always larger than child elements. This ordering rule is maintained in the entire tree. Thus, all the parent elements are larger than children elements in any level of the tree and the root contains the largest value.



ii. **Min Heap/Ascending Heap:**

Min heap is a heap structure where parent element is lower than child elements.



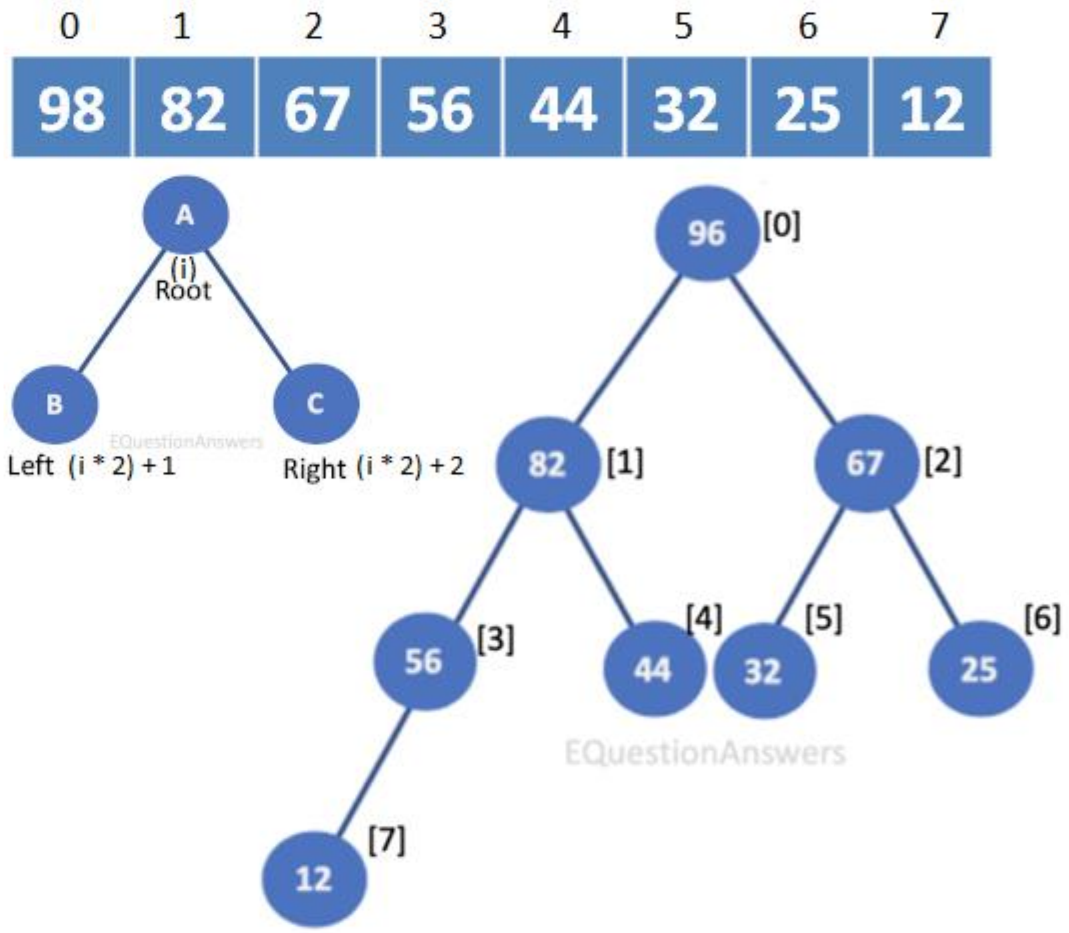
Binary Heap and Array Elements Mapping

There is a relation between binary tree and array. A Binary Heap is a complete Binary Tree and it can be represented as an array. Each element index in array can be mapped to a node in binary tree.

if the parent node is stored at index i,

the left child can be calculated by (2 * i + 1)

and right child by (2 * i + 2).



0	1	2	3	4	5	6	7
98	82	67	56	44	32	25	12

Heap Sort Algorithm :

Sorting can be in ascending or descending order. Either Max heap or min heap logic can be taken depending on the need.

1. Build a max/min heap using Heapify() from the input data.
2. At this point, the largest/smallest item is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of heap by 1. Finally, heapify the root of tree.
3. Repeat above steps while size of heap is greater than 1.

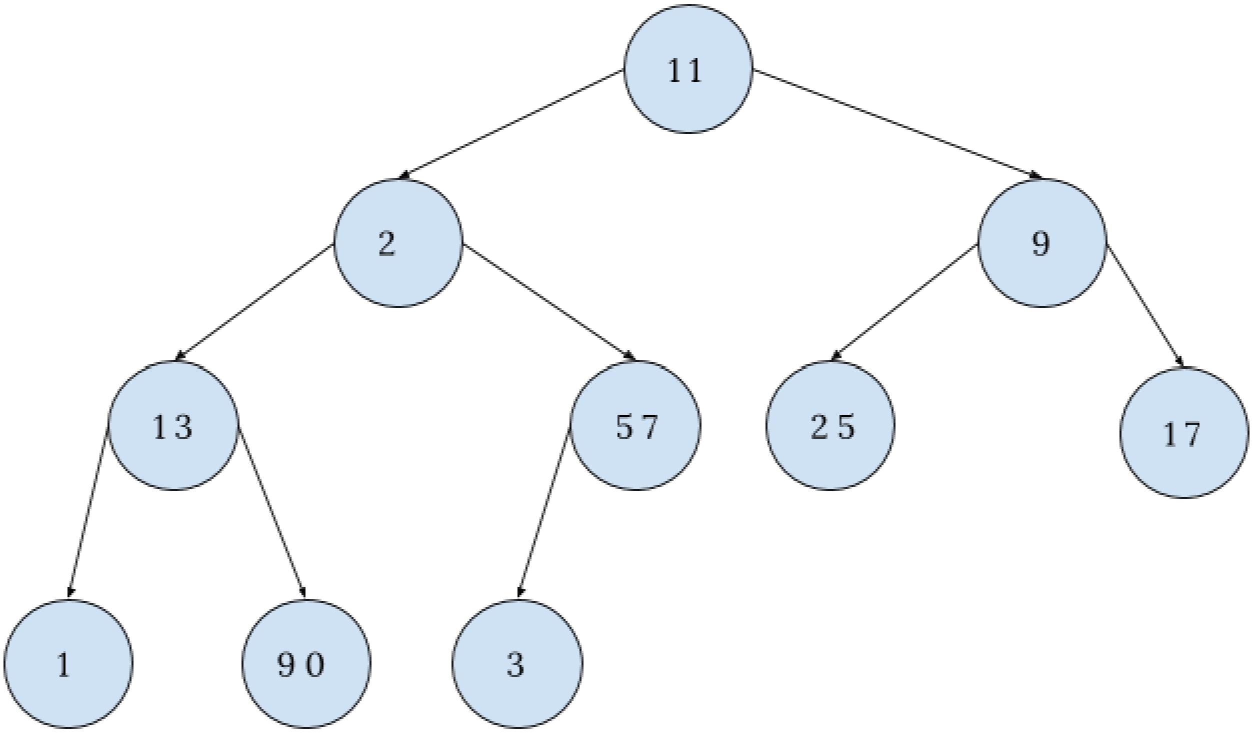
Question:

Suppose the array that is to be sorted contains the following elements: 11, 2, 9, 13, 57, 25, 17, 1, 90, 3.

Arrange elements in ascending order using Heap Sort. (use Max Heap).

Solution:

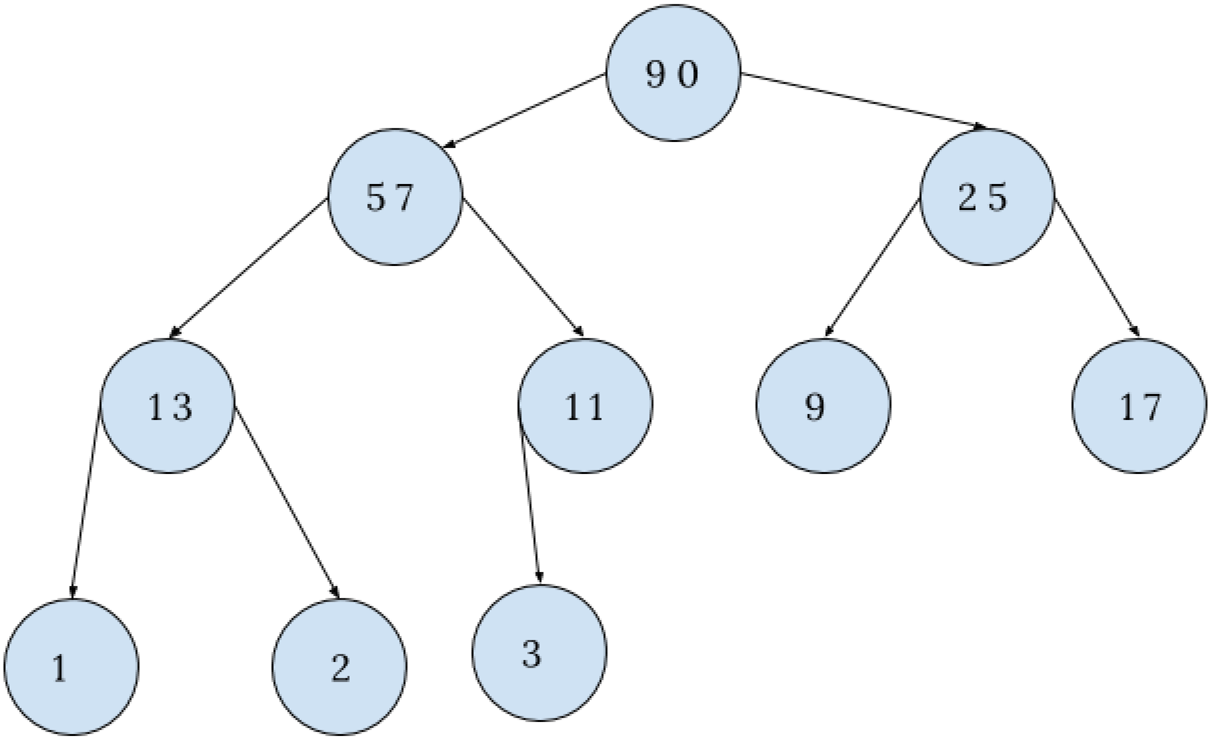
The first step now is to create a heap from the array elements



11	2	9	13	57	25	17	1	90	3

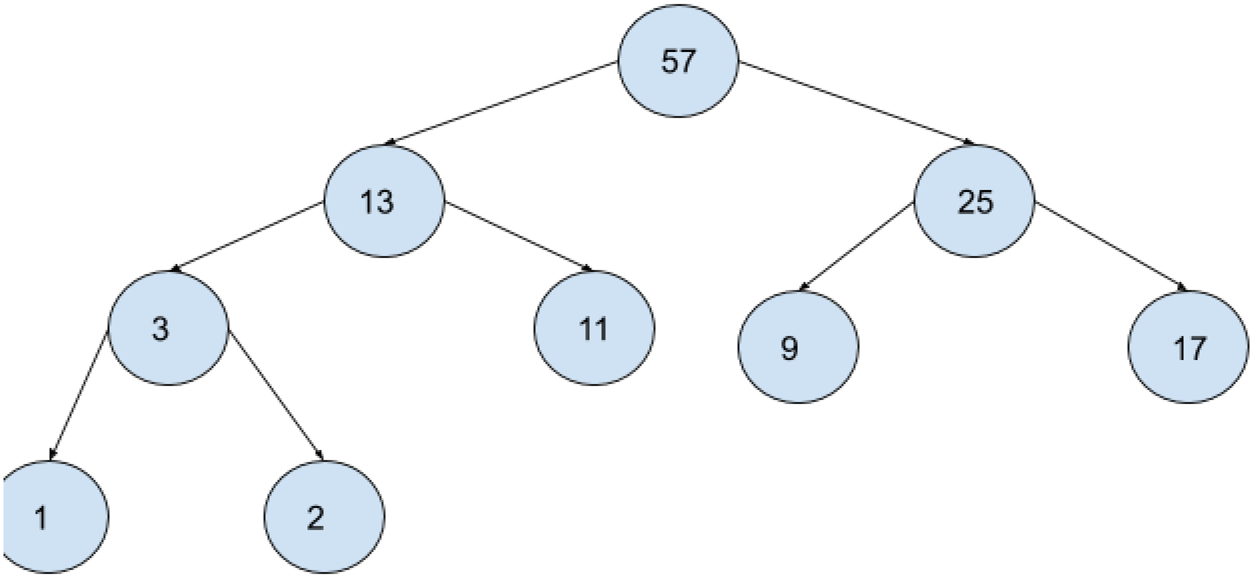
The above diagram depicts the binary tree version of the array.

We build the heap from the above binary tree and is shown below:



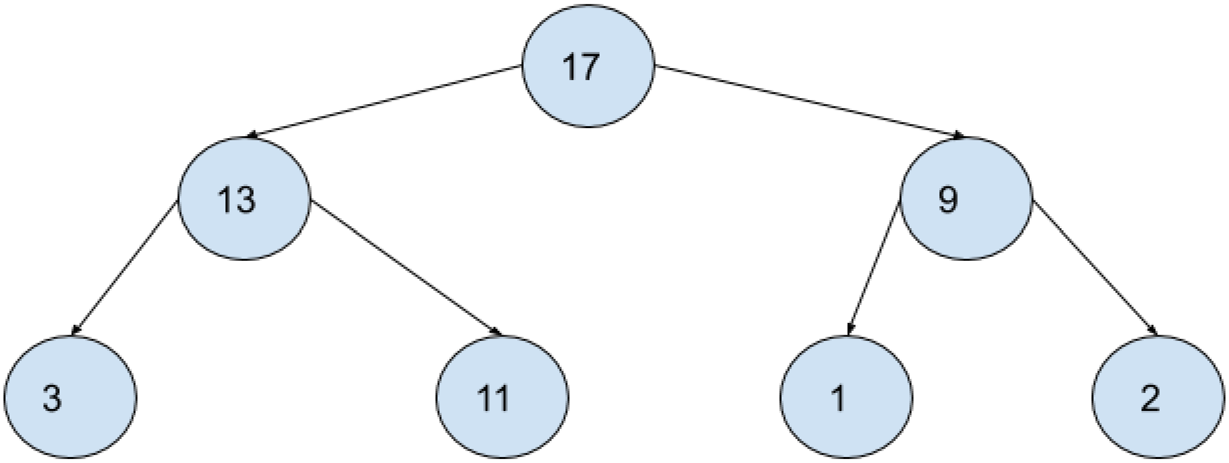
90	57	25	13	11	9	17	1	2	3

Now the root 90 is moved to the last location by exchanging it with 3. Finally, 90 is eliminated from the heap by reducing the maximum index value of the array by 1. The balance elements are then rearranged into the heap. The heap and array look like as shown in the below diagram:



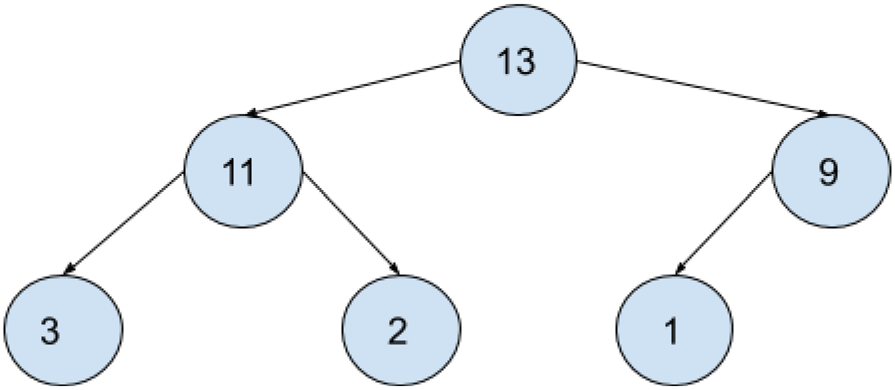
57	13	25	3	11	9	17	1	2	90

Similarly, when the current root 57 is exchanged with 2, and eliminated from the heap by reducing the maximum index value of the array by 1. The balance elements are then rearranged into the heap. The heap and array look like as shown in the below diagram:

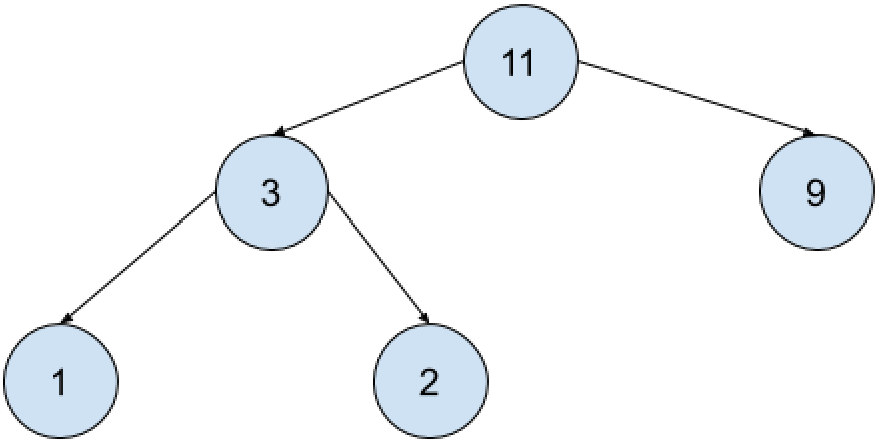


17	13	9	3	11	1	2	25	57	90

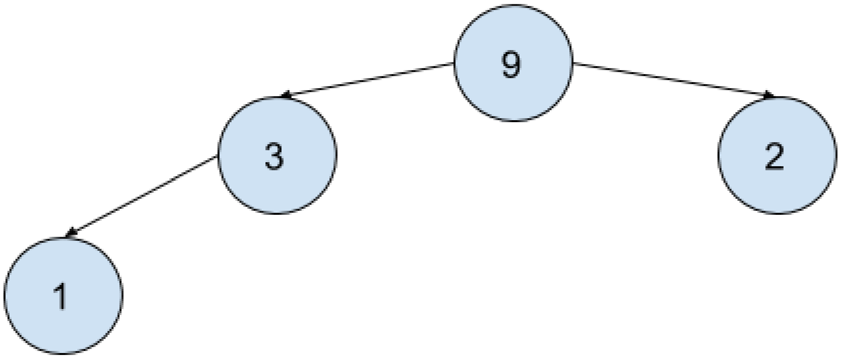
Following the same approach, the following phases are followed until the fully sorted array is achieved.



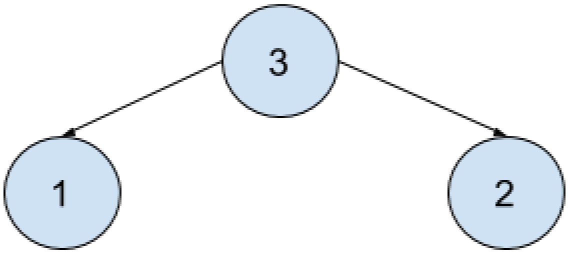
13	11	9	3	2	1	17	25	57	90



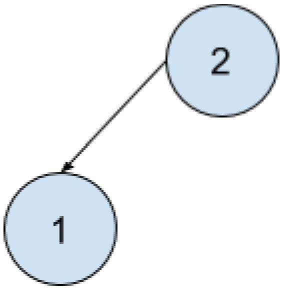
11	3	9	1	2	13	17	25	57	90



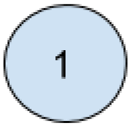
9	3	2	1	11	13	17	25	57	90



3	1	2	9	11	13	17	25	57	90



2	1	3	9	11	13	17	25	57	90



1	2	3	9	11	13	17	25	57	90