Definition of a Heap:
- It is a Complete Binary Tree
- For every pair of parent-child nodes in the CBT with values say \(p\) and \(c\), we must have \(c < p\).

Note:
- Since in a heap, every parent has a value larger than the values in the child nodes, it is clear that the root of the heap has the largest value.
- Since a heap is a CBT, we can use an array to implement it.

Heap Sort: It uses a Heap Data Structure to arrange numbers in an array in ascending order. Since a CBT can be implemented using an array, we can visualize the given array as a CBT. But it is not a heap.

Step-1) So, the first step is to “heapify” the given array. Now we know that the largest number will be in the root of the heap. The root of the heap is in the array location 0. Naturally, we would like to move it to the last location of the array.

Step-2) We do the following two steps until all numbers are sorted:
- We swap the values in the root and the last location. This swap upsets the heap property of the CBT.
- So, we reset the array to be a heap.

Implementation of each step:

Step-1) Heapify the given array. Starting with the 2\(^{nd}\) number and continuing with successive numbers do an “upheap” on each number:

“Upheap” on a (current) number: Consider all the ancestors of the current number. Do an insertion sort where the numbers are arranged in descending order. That is, place the current number in a temporary location and move (down from a parent, in a CBT representation) all the numbers larger than the current number. Place the current number in the location vacated by the last move.

Step-2) Reset the array to form a heap, after the swap. Due to the swap the root has a value that upsets the heap property.

“Downheap” on the root: Move the value in the root to a temporary location. Now, every time a parent node is vacated in the CBT, do the following until a node is found to place the root value.

If a leaf node is vacated, move the root value from the temporary location to the leaf node. Otherwise, find the child that has the larger value and
- If the larger value is also larger than the root value then move that value up to its parent node.
- If the larger value is not larger than the root value then move the root value from the temporary location to the last vacated node.

Complexity Analysis on Heap Sort:

Step-1) In Upheap we climb up the CBT from the current node until we find a node to place the current number. So, the number of nodes processed in Upheap cannot be larger than the height of the tree. The height of a CBT with \(n\) nodes is \(O(\log n)\). Since Upheap is called \((n-1)\) times, the cost in completing the first step in an Heap sort is \(O(n \log n)\).

Step-2) In Downheap we climb down the CBT from the root until we find a node to place the root value. So, the number of nodes processed in Downheap cannot be larger than the height of the tree \(O(\log n)\). Since a swap is done \((n-1)\) times, the cost in completing the second step in an Heap sort is \(O(n \log n)\).

Hence the total cost is \(O(n \log n)\).