Heap sort is a comparison-based sorting technique based on Binary Heapdatastructure. It is similar to the selection sort where we first find the maximumelement and place the maximum element at the end. We repeat the sameprocess for the remaining element.

**General Algorithm Given below is the general algorithm for heap sort technique.**

∙ Build a max heap from the given data such that the root is the highest element of the heap.

∙ Remove the root i.e. the highest element from the heap and replace or swap it with the last element of the heap.

∙ Then adjust the max heap, so as to not to violate the max heap properties (heapify). ∙ The above step reduces the heap size by 1.

∙ Repeat the above three steps until the heap size is reduced to 1. As shown in the general algorithm to sort the given dataset in increasing order, we first construct a max heap for the given data.

**Let us take an example to construct a max heap with the following dataset.** 6, 10, 2, 4, 1

**We can construct a tree for this data set as follows.**

****

In the above tree representation, the numbers in the brackets represent the respective positions in the array.

In order to construct a max heap of the above representation, we need to fulfill the heap condition that the parent node should be greater than its child nodes. In other words, we need to “heapify” the tree so as to convert it to max-heap.

**After heapification of the above tree, we will get the max-heap as shown below.**

****As shown above, we have this max-heap generated from an array.

Next, we present an illustration of a heap sort. Having seen the construction of max-heap, we will skip the detailed steps to construct a max-heap and will directly show the max heap at each step.

**Illustration**

Consider the following array of elements. We need to sort this array using the heap sort technique.



**Let us construct a max-heap as shown below for the array to be sorted. **

**Once the heap is constructed, we represent it in an Array form as shown below.**

****

Now we compare the 1st node (root) with the last node and then swap them. Thus, as shown above, we swap 17 and 3 so that 17 is at the last position and 3 is in the first position. Now we remove the node 17 from the heap and put it in the sorted array as shown in the shaded portion below.

Now we again construct a heap for the array elements. This time the heap size is reduced by 1 as we have deleted one element (17) from the heap.

**The heap of the remaining elements is shown below.**

****

In the next step, we will repeat the same steps.

**We compare and swap the root element and last element in the heap.**

****

After swapping, we delete the element 12 from the heap and shift it to the sorted array. 

Once again we construct a max heap for the remaining elements as shown below. 

Now we swap the root and the last element i.e. 9 and 3. After swapping, element 9 is deleted from the heap and put in a sorted array.



At this point, we have only three elements in the heap as shown below. 

We swap 6 and 3 and delete the element 6 from the heap and add it to the sorted array. 



Now we construct a heap of the remaining elements and then swap both with each other.



After swapping 4 and 3, we delete element 4 from the heap and add it to the sorted array. Now we have only one node remaining in the heap as shown below**. **

So now with only one node remaining, we delete it from the heap and add it to the sorted array.



Thus the above shown is the sorted array that we have obtained as a result of the heap sort.

In the above illustration, we have sorted the array in ascending order. If we have to sort the array in descending order then we need to follow the same steps but with the min-heap.

Heapsort algorithm is identical to selection sort in which we select the smallest element and place it into a sorted array. However, heap sort is faster than selection sort as far as the performance is concerned. We can put it as heapsort is an improved version of the selection sort.

The most important function in both the implementations is the function “heapify”. This function is called by the main heapsort routine to rearrange the subtree once a node is deleted or when max-heap is built.

When we have heapified the tree correctly, only then we will be able to get the correct elements in their proper positions and thus the array will be correctly sorted.