HEAP SORT

Heap is a complete binary tree data structure with the property that the value of a parent node is either greater than or equal to (max heap) or less than or equal to (min-heap) the values of its children. Heaps are used for efficient algorithms in sorting, selection, and as a priority queue.
Heap sort is a comparison-based sorting algorithm that sorts an array by creating a binary heap data structure. It has an average and worst-case time complexity of O(n log n)

What is Heap Sort in Data Structure?

Heap sort is a comparison-based sorting algorithm that sorts an array in ascending or descending order by transforming it into a binary heap data structure. It uses the concepts of max-heap and min-heap, where in max-heap, the largest element is the root node, and in min-heap, the smallest element is the root node. The algorithm uses the binary heap structure to efficiently sort the elements.

Properties of Binary Heap:

- **Complete Binary Tree:** All levels except the last level are completely filled and the last level is filled from left to right.
- **Shape Property:** The value of each node is less than or equal to the value of its parent.
- **Heap Property:** The value of the parent node is always greater than the value of its child node in a max heap, and the value of the parent node is always less than the value of its child node in a min-heap.

What is Heapify?

Heapify is the process of converting a binary tree into a valid heap data structure. This is usually achieved by starting from the last internal node (i.e., the parent of the rightmost leaf node) and working towards the root node, swapping the node with its largest (or smallest, depending on whether it is a max or min heap) child node until the heap property is satisfied. The process is repeated for the affected subtrees until the entire tree is a valid heap. The time complexity of heapify is O(log n), where n is the number of nodes in the tree.

Heap Sort Algorithm

Here are the steps of the Heap Sort algorithm:

- Build a max-heap from the input data: The first step is to build a binary max-heap from the input data. This is done by transforming the array into a complete binary tree where the largest value is at the root node.
- Swap the first and last elements of the array: After building the max heap, the largest value will be at the root node. Swap this value with the last element of the array. This will place the largest value at the end of the array, which is its final position in the sorted array.
- Heapify the remaining data: After swapping the first and last elements, the heap size is reduced by one. The root node may now violate the max-heap property, so the algorithm needs to reheapify the data to restore the max-heap.
- Repeat steps 2 and 3 until the heap size is reduced to 1: The algorithm continues to repeat steps 2 and 3 until the heap size is reduced to 1. This means that the entire array has been sorted in ascending order.
- Return the sorted array: After the heap size has been reduced to 1, the array is sorted and can be returned as the final result.