# Heap Sort Algorithm

Heap Sort is one of the best sorting methods being in-place and with no quadratic worst-case running time. Heap sort involves building a **Heap** data structure from the given array and then utilizing the Heap to sort the array.

You must be wondering, how converting an array of numbers into a heap data structure will help in sorting the array. To understand this, let's start by understanding what is a Heap.

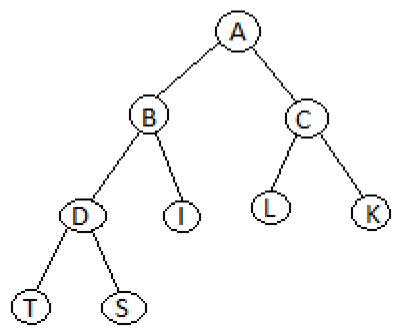
**NOTE:** If you are not familiar with Sorting in data structure, you should first learn what is sorting to know about the basics of sorting.

---

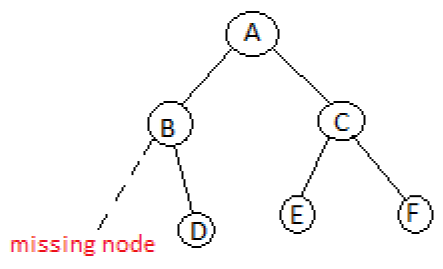
## What is a Heap ?

Heap is a special tree-based data structure, that satisfies the following special heap properties:

1. **Shape Property:** Heap data structure is always a Complete Binary Tree, which means all levels of the tree are fully filled.

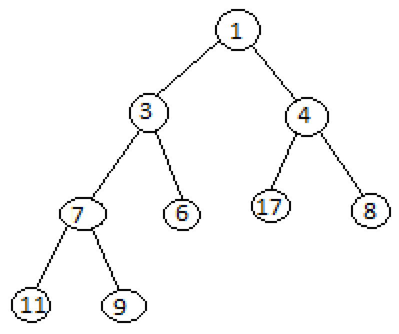


Complete Binary Tree



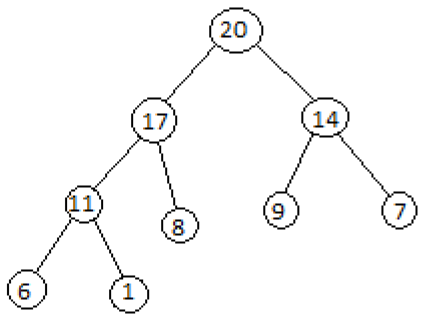
In-Complete Binary Tree

2. **Heap Property:** All nodes are either **greater than or equal to** or **less than or equal to** each of its children. If the parent nodes are greater than their child nodes, heap is called a **Max-Heap**, and if the parent nodes are smaller than their child nodes, heap is called **Min-Heap**.



Min-Heap

In min-heap, first element is the smallest. So when we want to sort a list in ascending order, we create a Min-heap from that list, and picks the first element, as it is the smallest, then we repeat the process with remaining elements.



Max-Heap

In max-heap, the first element is the largest, hence it is used when we need to sort a list in descending order.

---

## How Heap Sort Works?

Heap sort algorithm is divided into two basic parts:

- Creating a Heap of the unsorted list/array.
- Then a sorted array is created by repeatedly removing the largest/smallest element from the heap, and inserting it into the array. The heap is reconstructed after each removal.

Initially on receiving an unsorted list, the first step in heap sort is to create a Heap data structure(Max-Heap or Min-Heap). Once heap is built, the first element of the Heap is either largest or smallest(depending upon Max-Heap or Min-Heap), so we put the first element of the heap in our array. Then we again make heap using the remaining elements, to again pick the first element of the heap and put it into the array. We keep on doing the same repeatedly untill we have the complete sorted list in our array.

In the below algorithm, initially `heapsort()` function is called, which calls `heapify()` to build the heap.

---

## Implementing Heap Sort Algorithm

Below we have a simple C++ program implementing the Heap sort algorithm.

```
/*  Below program is written in C++ language  */


#include <iostream>


using namespace std;


void heapify(int arr[], int n, int i)
{
    int largest = i;
    int l = 2*i + 1;
    int r = 2*i + 2;

    // if left child is larger than root
    if (l < n && arr[l] > arr[largest])
        largest = l;

    // if right child is larger than largest so far
    if (r < n && arr[r] > arr[largest])
        largest = r;

    // if largest is not root

    if (largest != i)

    {

        swap(arr[i], arr[largest]);



        // recursively heapify the affected sub-tree

        heapify(arr, n, largest);

    }

}



void heapSort(int arr[], int n)

{

    // build heap (rearrange array)

    for (int i = n / 2 - 1; i >= 0; i--)

        heapify(arr, n, i);



    // one by one extract an element from heap

    for (int i=n-1; i>=0; i--)

    {

        // move current root to end

        swap(arr[0], arr[i]);



        // call max heapify on the reduced heap

        heapify(arr, i, 0);

    }

}



/* function to print array of size n */

void printArray(int arr[], int n)

{
    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << "\n";

}




int main()

{

    int arr[] = {121, 10, 130, 57, 36, 17};

    int n = sizeof(arr)/sizeof(arr[0]);


    heapSort(arr, n);


    cout << "Sorted array is \n";

    printArray(arr, n);

}
```
Copy

---

**Complexity Analysis of Heap Sort**

Worst Case Time Complexity: **O(n*log n)**

Best Case Time Complexity: **O(n*log n)**

Average Time Complexity: **O(n*log n)**

Space Complexity : **O(1)**

- Heap sort is not a Stable sort, and requires a constant space for sorting a list.
- Heap Sort is very fast and is widely used for sorting.

Now that we have learned Heap sort algorithm, you can check out these sorting algorithms and their applications as well:

- Insertion Sort
- Selection Sort
- Bubble Sort

- Merge sort
- Heap Sort
- Counting Sort