# Heap Sort – Data Structures and Algorithms Tutorials

Last Updated : 28 Sep, 2024

**Heap sort** is a comparison-based sorting technique based on Binary Heap Data Structure. It can be seen as an optimization over selection sort where we first find the max (or min) element and swap it with the last (or first). We



instead of O(n) and hence achieve the O(n Log n) time complexity.

### Table of Content



## Heap Sort Algorithm

First convert the array into a max heap using **heapify**, Please note that this happens in-place. The array elements are re-arranged to follow heap properties. Then one by one delete the root node of the Max-heap and replace it with the last node and **heapify**. Repeat this process while size of heap is greater than 1.

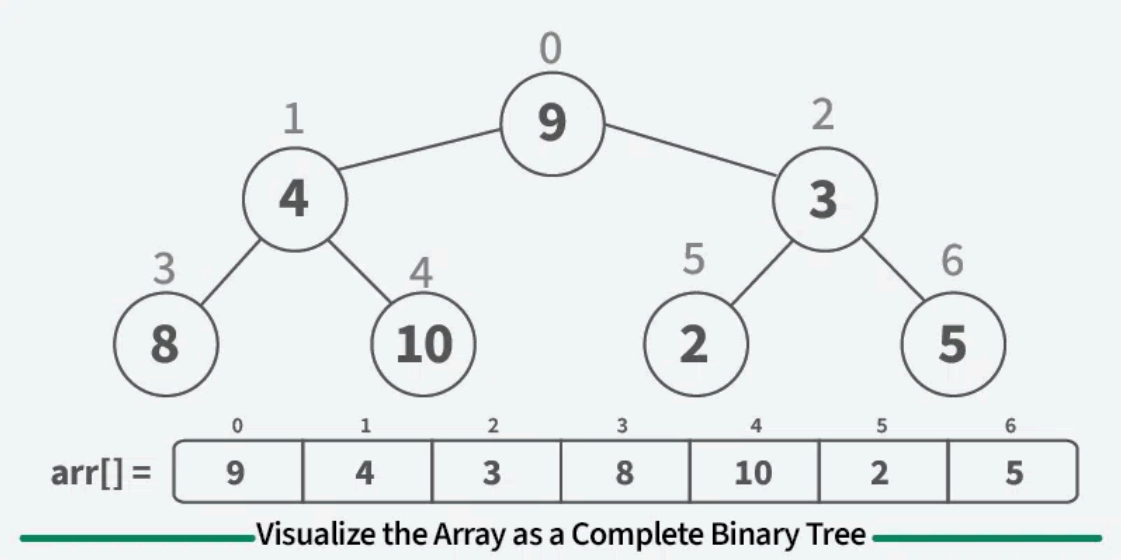
- Rearrange array elements so that they form a Max Heap.
- Repeat the following steps until the heap contains only one element:
    - Swap the root element of the heap (which is the largest element in current heap) with the last element of the heap.
    - Remove the last element of the heap (which is now in the correct position). We mainly reduce heap size and do not remove element

from the actual array.
    - Heapify the remaining elements of the heap.
- Finally we get sorted array.

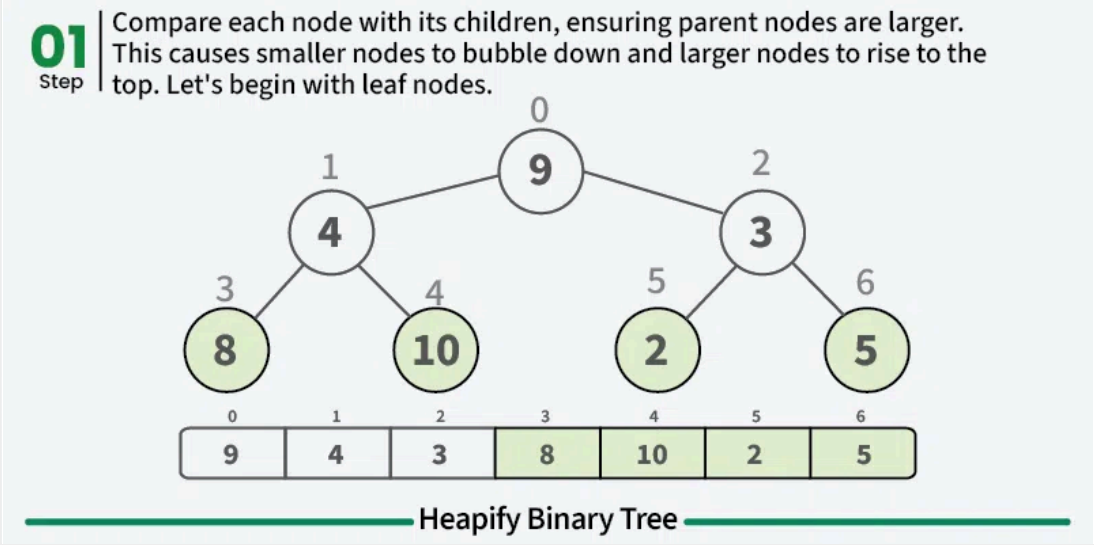
## Detailed Working of Heap Sort

### Step 1: Treat the Array as a Complete Binary Tree

We first need to visualize the array as a **complete binary tree**. For an array of size **n**, the root is at index **0**, the left child of an element at index **i** is at **2i + 1**, and the right child is at **2i + 2**.

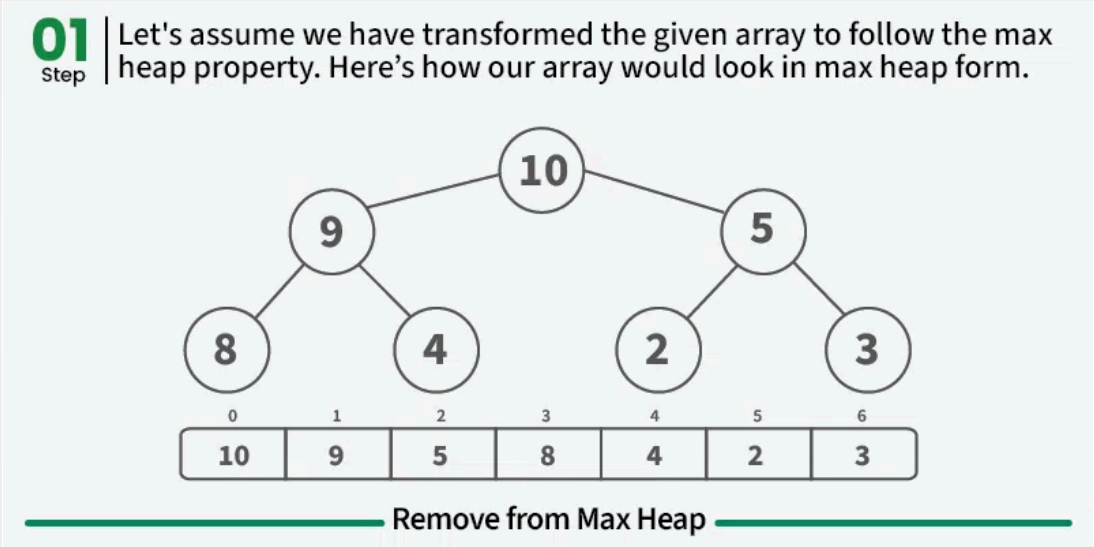



### Step 2: Build a Max Heap



1 / 7

### Step 3: Sort the array by placing largest element at end of unsorted array.




1 / 6

In the illustration above, we have shown some steps to sort the array. We need to keep repeating these steps until there's only one element left in the heap.

## Implementation of Heap Sort

C++ C Java Python C# JavaScript PHP

```python
# Python program for implementation of heap Sort

# To heapify a subtree rooted with node i
# which is an index in arr[].
def heapify(arr, n, i):

     # Initialize largest as root
    largest = i

    #  left index = 2*i + 1
    l = 2 * i + 1

    # right index = 2*i + 2
    r = 2 * i + 2

```

```python
    # If left child is larger than root
    if l < n and arr[l] > arr[largest]:
        largest = l

    # If right child is larger than largest so far
    if r < n and arr[r] > arr[largest]:
        largest = r

    # If largest is not root
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]  # Swap

        # Recursively heapify the affected sub-tree
        heapify(arr, n, largest)

# Main function to do heap sort
def heapSort(arr):

    n = len(arr)

    # Build heap (rearrange array)
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)

    # One by one extract an element from heap
    for i in range(n - 1, 0, -1):

        # Move root to end
        arr[0], arr[i] = arr[i], arr[0]

        # Call max heapify on the reduced heap
        heapify(arr, i, 0)

def printArray(arr):
    for i in arr:
        print(i, end=" ")
    print()

# Driver's code
arr = [9, 4, 3, 8, 10, 2, 5]
heapSort(arr)
print("Sorted array is ")
printArray(arr)
```

**Output**

```
Sorted array is
2 3 4 5 8 9 10
```

## Complexity Analysis of Heap Sort

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

**Auxiliary Space:** O(log n), due to the recursive call stack. However, auxiliary space can be O(1) for iterative implementation.

## Important points about Heap Sort

- Heap sort is an in-place algorithm.
- Its typical implementation is not stable but can be made stable (See this)
- Typically 2-3 times slower than well-implemented QuickSort. The reason for slowness is a lack of locality of reference.

## Advantages of Heap Sort

- **Efficient Time Complexity:** Heap Sort has a time complexity of O(n log n) in all cases. This makes it efficient for sorting large datasets. The **log n** factor comes from the height of the binary heap, and it ensures that the algorithm maintains good performance even with a large number of elements.
- **Memory Usage:** Memory usage can be minimal (by writing an iterative heapify() instead of a recursive one). So apart from what is necessary to hold the initial list of items to be sorted, it needs no additional memory space to work
- **Simplicity:** It is simpler to understand than other equally efficient sorting algorithms because it does not use advanced computer science concepts such as recursion.

## Disadvantages of Heap Sort

- **Costly**: Heap sort is costly as the constants are higher compared to merge sort even if the time complexity is O(n Log n) for both.
- **Unstable**: Heap sort is unstable. It might rearrange the relative order.

- **Inefficient:** Heap Sort is not very efficient because of the high constants in the time complexity.

## Frequently Asked Question (FAQs) on Heap Sort

### What are the two phases of Heap Sort?

*The heap sort algorithm consists of two phases. In the first phase, the array is converted into a max heap. And in the second phase, the highest element is removed (i.e., the one at the tree root) and the remaining elements are used to create a new max heap.*

### Why Heap Sort is not stable?

*The heap sort algorithm is not a stable algorithm because we swap arr[i] with arr[0] in heapSort() which might change the relative ordering of the equivalent keys.*

### Is Heap Sort an example of the “Divide and Conquer” algorithm?

*Heap sort is **NOT** at all a Divide and Conquer algorithm. It uses a heap data structure to efficiently sort its element and not a “divide and conquer approach” to sort the elements.*

### Which sorting algorithm is better – Heap sort or Merge Sort?

*The answer lies in the comparison of their time complexity and space requirements. The Merge sort is slightly faster than the Heap sort. But on the other hand merge sort takes extra memory. Depending on the requirement, one should choose which one to use.*

### Why is Heap sort better than Selection sort?

*Heap sort is similar to selection sort, but with a better way to get the maximum element. It takes advantage of the heap data structure to get the maximum element in constant time*



**Next Article**

Iterative HeapSort

## Similar Reads

### Difference between Binary Heap, Binomial Heap and Fibonacci Heap

Binary Heap:A Binary Heap is a Binary Tree with following properties. It's a complete binary tree i.e., all levels are completely filled except possibly the last...

2 min read

### Heap Sort for decreasing order using min heap

Given an array of elements, sort the array in decreasing order using min heap. Examples: Input : arr[] = {5, 3, 10, 1} Output : arr[] = {10, 5, 3, 1} Input : arr[] = {1...

13 min read

### Data Structures | Heap | Question 2

A max-heap is a heap where the value of each parent is greater than or equal to the values of its children. Which of the following is a max-heap? (GATE CS 201...

1 min read

### Data Structures | Heap | Question 5

Consider a binary max-heap implemented using an array. Which among the following arrays represents a binary max-heap? (More than one option correct)...

1 min read

### Data Structures | Heap | Question 6

A min- Heap is a complete binary tree. (A) True (B) False Answer: (A) Explanation: A max-heap is a special Tree-based data structure in which the tre...

1 min read

### Data Structures | Heap | Question 7

We have a binary heap on n elements and wish to insert n more elements (not necessarily one after another) into this heap. The total time required for this is (...

1 min read

### Data Structures | Heap | Question 11

Given two max heaps of size n each, what is the minimum possible time complexity to make a one max-heap of size from elements of two max heaps? (...

1 min read

### Data Structures | Heap | Question 3

A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is...

1 min read

### Data Structures | Heap | Question 4

Suppose the elements 7, 2, 10 and 4 are inserted, in that order, into the valid 3-ary max heap found in the above question, Which one of the following is the...

1 min read

### Data Structures | Heap | Question 8

In a min-heap with n elements with the smallest element at the root, the 7th smallest element can be found in time: (A) theta(nlogn) (B) theta(n) (C)...

1 min read

Article Tags : DSA Heap Sorting 24*7 Innovation Labs +8 More

Practice Tags : 24*7 Innovation Labs Amazon Belzabar Intuit +6 More