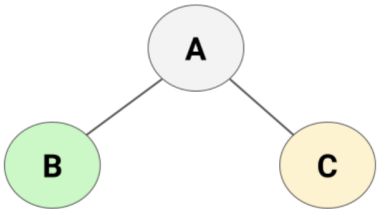
Heap

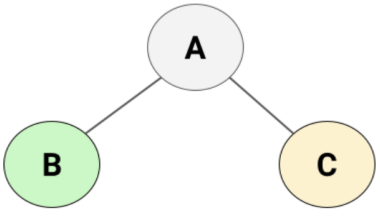
💡 Heap data structure is a complete binary tree that satisfies **the heap property**, where any given node is

→ always greater than its child node/s and the key of the root node is the largest among all other nodes. This property is also called **max heap property**.

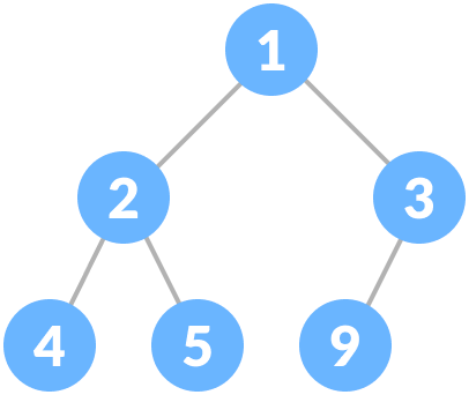
→ always smaller than the child node/s and the key of the root node is the smallest among all other nodes. This property is also called **min heap property**.



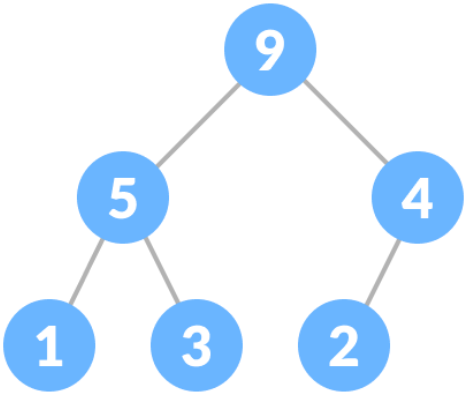
For each parent A and its children B and C
Min-heap property: A <= B, A <= C



For each parent A and its children B and C
Max-heap property: A >= B, A >= C



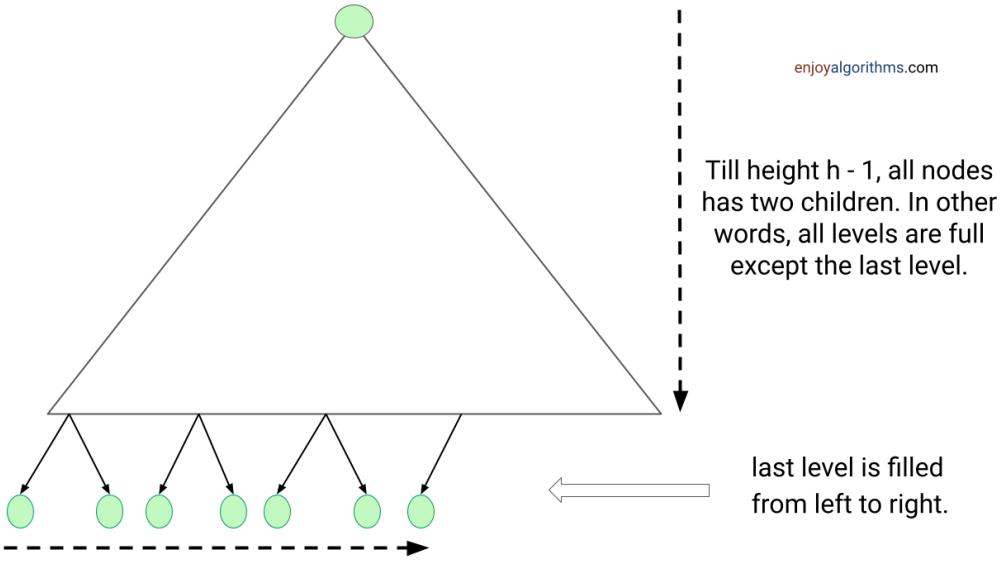
Min Heap Data Structure Example

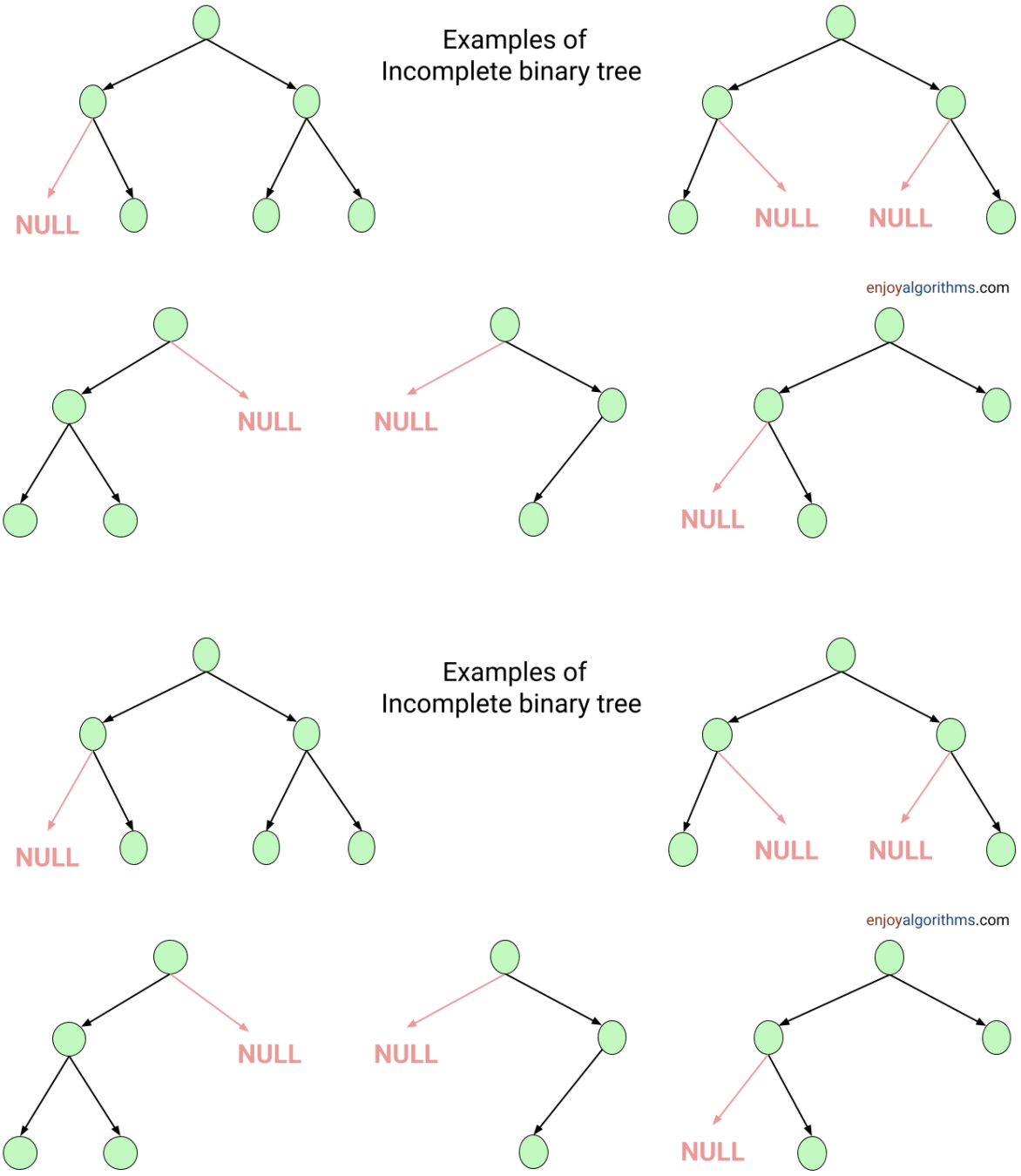


Max Heap Data Structure Example

Also called **binary heap**.

What is Complete Binary Tree?

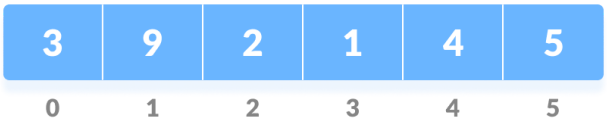


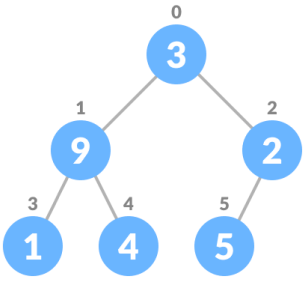


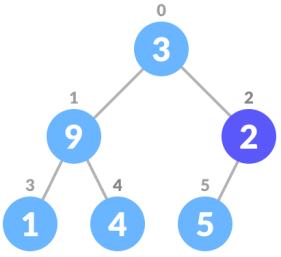
Operations of Heap

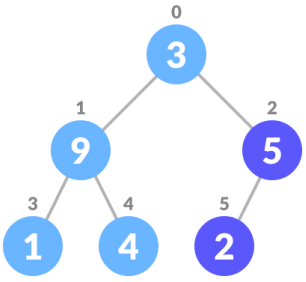
Heapify

→ Heapify is the process of creating a heap data structure from a binary tree. It is used to create a Min-Heap or a Max-Heap.





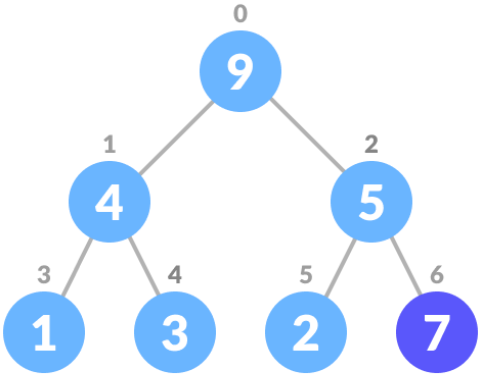


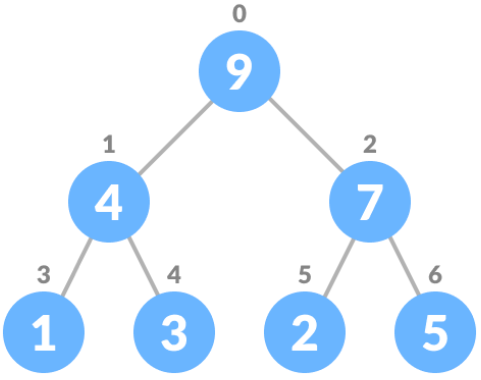


Insertion

```
If there is no node,
  create a newNode.
else (a node is already present)
  insert the newNode at the end (last node from left to right

heapify the array
```

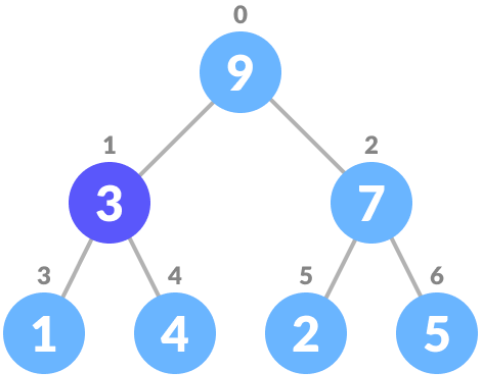


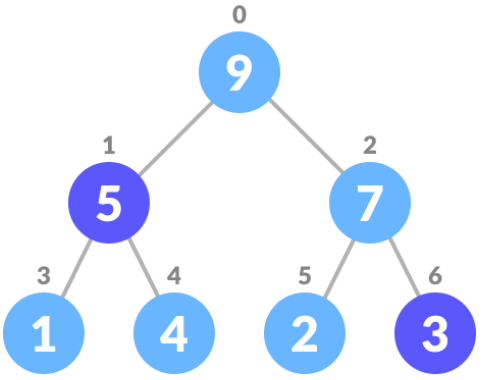


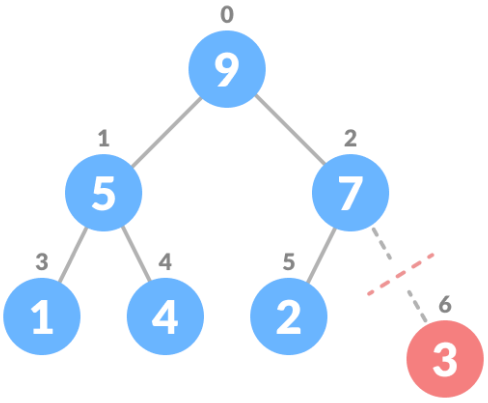
Deletion

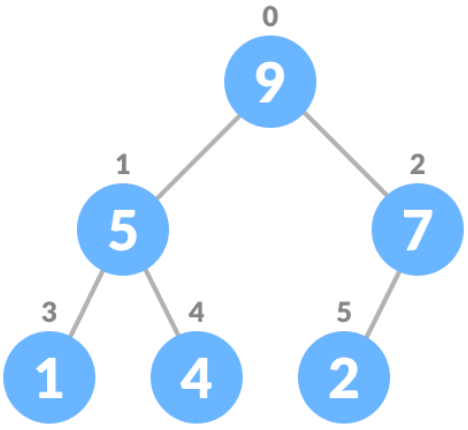
```
If nodeToBeDeleted is the leafNode
  remove the node
Else swap nodeToBeDeleted with the lastLeafNode
  remove noteToBeDeleted

heapify the array
```









Peek(Find min / max value)

Peek operation returns the maximum element from Max Heap or minimum element from Min Heap without deleting the node.

Extract-Max/Min

Extract-Max returns the node with maximum value after removing it from a Max Heap whereas Extract-Min returns the node with minimum after removing it from Min Heap.

Application of Data Structure

1. Priority Queues: Heaps are commonly used to implement priority queues, where elements with higher priority are extracted first. This is useful in many applications such as scheduling tasks, handling interruptions, and processing events.
2. Sorting Algorithms: Heapsort, a comparison-based sorting algorithm, is implemented using the Heap data structure. It has a time complexity of O(n log n), making it efficient for large datasets.
3. Graph algorithms: Heaps are used in graph algorithms such as Dijkstra's shortest path algorithm, Prim's minimum spanning tree algorithm, and the A* search algorithm.
4. File Compression: Heaps are used in data compression algorithms such as Huffman coding, which uses a priority queue implemented as a min-heap to build a Huffman tree.
5. Dynamic programming: Heaps are used in dynamic programming algorithms such as the greedy algorithm, where elements are processed in order of priority.
6. Medical Applications: In medical applications, heaps are used to store and manage patient information based on priority, such as vital signs, treatments, and test results.
7. External sorting: Heaps are used in external sorting algorithms to sort large datasets that do not fit into memory, by processing chunks of data in a priority queue.
8. Load balancing: Heaps are used in load balancing algorithms to distribute tasks or requests to servers, by processing elements with the lowest load first.
9. Online algorithms: Heaps are used in online algorithms, where elements are processed in real-time as they arrive, such as recommendation systems, event processing, and streaming data.
10. Stock market: Heaps are used in financial applications, such as stock market analysis and algorithmic trading, to process and manage large amounts of stock.

Advantages of Heap Data Structure:

→ **Efficient insertion and deletion:** The heap data structure allows efficient insertion and deletion of elements. When a new element is added to the heap, it is placed at the bottom of the heap and moved up to its correct position using the heapify operation. Similarly, when an element is removed from the heap, it is replaced by the bottom element, and the heap is restructured using the heapify operation.

→ **Efficient priority queue:** The heap data structure is commonly used to implement a priority queue, where the highest priority element is always at the top of the heap. The heap allows constant-time access to the highest priority element, making it an efficient data structure for implementing priority queues.

→ **Guaranteed access to the maximum or minimum element**: In a max-heap, the top element is always the maximum element, and in a min-heap, the top element is always the minimum element. This provides guaranteed access to the maximum or minimum element in the heap, making it useful in algorithms that require access to the extreme values.

→ **Space efficiency:** The heap data structure requires less memory compared to other data structures, such as linked lists or arrays, as it stores elements in a complete binary tree structure.

→ **Heap-sort algorithm**: The heap data structure forms the basis for the heap-sort algorithm, which is an efficient sorting algorithm that has a worst-case time complexity of O(n log n).

Disadvantages of Heap Data Structure

→ **Lack of flexibility:** The heap data structure is not very flexible, as it is designed to maintain a specific order of elements. This means that it may not be suitable for some applications that require more flexible data structures.

→ **Not ideal for searching:** While the heap data structure allows efficient access to the top element, it is not ideal for searching for a specific element in the heap. Searching for an element in a heap requires traversing the entire tree, which has a time complexity of O(n).

→ **Not a stable data structure:** The heap data structure is not a stable data structure, which means that the relative order of equal elements may not be preserved when the heap is constructed or modified.

→ **Memory management:** The heap data structure requires dynamic memory allocation, which can be a challenge in some systems with limited memory. In addition, managing the memory allocated to the heap can be complex and error-prone.

→**Complexity:** While the heap data structure allows efficient insertion, deletion, and priority queue implementation, it has a worst-case time complexity of O(n log n), which may not be optimal for some applications that require faster algorithms.

Heap Vs Tree

S.No	Heap	Tree
1	Heap is a kind of Tree itself.	The tree is not a kind of heap.
2	Usually, Heap is of two types, Max-Heap and Min-Heap.	Whereas a Tree can be of various types for eg. binary Tree, BST(Binary Search tree), AVL tree, etc.
3	Heap is ordered.	Binary Tree is not ordered but BST is ordered.
4	Insert and remove will take O(log(N)) time in the worst case.	Insert and remove will take O(N) in the worst case in case the tree is skewed.
5	Finding Min/Max value in Heap is O(1) in the respective Min/Max heap.	Finding Min/Max value in BST is O(log(N)) and Binary Tree is O(N).
6	Heap can also be referred to as Priority Queue.	A tree can also be referred to as a connected undirected graph with no cycle.
7	Heap can be built in linear time complexity.	BST: O(N * log(N)) and Binary Tree: O(N).
8	Applications: Prim's Algorithm and Dijkstra's algorithm.	Applications: Spanning Trees, Trie, B+ Tree, BST, Heap.

📚 References

GeeksforGeeks | A computer science portal for geeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive

https://www.geeksforgeeks.org/

Programiz: Learn to Code for Free

Learn to code in Python, C/C++, Java, and other popular programming languages with our easy to follow tutorials, examples, online compiler and references.

https://www.programiz.com

Introduction to Heap Data Structure: Properties and Applications

Heap is a complete binary tree structure where each node satisfies a heap property. We learn two types of heap data structure: 1) Max heap, which satisfies the max heap property, and 2) Min heap, which satisfies the min-heap property. We use heap data structures to implement priority queues and solve several coding

https://www.enjoyalgorithms.com/blog/introduction-to-heap-data-structure

✍🏻 Author → Serhat Kumas

https://www.linkedin.com/in/serhatkumas/

SerhatKumas - Overview

Computer engineering student who loves coding in different fields instead of focusing on a one spesific area. - SerhatKumas

https://github.com/SerhatKumas