Definition and Classification of Heap

## Abstract Data Type and Data Structure

* Simply put, an ADT (Abstract Data Type) is more of a logical description, while a Data Structure is concrete.
* Think of an ADT as a picture of the data and the operations to manipulate and change it.
* A Data Structure is the real, concrete thing. It can be implemented and used within an algorithm.
* ADT is to an interface (what it does) what a data structure is to a class (how it does it).
* A few examples:
  + ADT: List
  + DS: ArrayList, LinkedList...
  + ADT: Map
  + DS: HashMap, TreeMap...

## Priority Queues

Before introducing a Heap, let's first talk about a Priority Queue.

[Wikipedia](https://en.wikipedia.org/wiki/Priority_queue): a priority queue is an [abstract data type](https://en.wikipedia.org/wiki/Abstract_data_type) similar to a regular [queue](https://en.wikipedia.org/wiki/Queue_(abstract_data_type)) or [stack](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)) data structure in which each element additionally has a "priority" associated with it. In a priority queue, an **element with high priority is served before an element with low priority**.

In daily life, we would assign different priorities to tasks, start working on the task with the highest priority and then proceed to the task with the second highest priority. This is an example of a Priority Queue.

A common misconception is that a Heap is the same as a Priority Queue, which is not true. A priority queue is an abstract data type, while a Heap is a data structure. Therefore, a Heap is not a Priority Queue, but a way to implement a Priority Queue.

There are multiple ways to implement a Priority Queue, such as array and linked list. However, these implementations only guarantee O(1) time complexity for either insertion or deletion, while the other operation will have a time complexity of O(N). On the other hand, implementing the priority queue with Heap will allow both insertion and deletion to have a time complexity of O(logN). So, what is a Heap?

In this chapter, we will learn to:

* Understand the Heap data structure.
* Understand Max Heap and Min Heap.
* Understand the insertion and deletion of a Heap.
* Implement a Heap.

## Definition of Heap

According to Wikipedia, a **Heap** is a special type of binary tree. A heap is a binary tree that meets the following criteria:

1. Is a **complete binary tree**;
   1. A ‘Complete binary tree is defined as a binary tree that has all of its node in from top to bottom and left to right.
2. The value of each node must be **no greater than (or no less than)** the value of its child nodes.

A Heap has the following properties:

1. Insertion of an element into the Heap has a time complexity of O(logN);
2. Deletion of an element from the Heap has a time complexity of O(logN);
3. The maximum/minimum value in the Heap can be obtained with O(1) time complexity.

## Classification of Heap

* There are two kinds of heaps: **Max Heap** and **Min Heap**.
  + Max Heap: Each node in the Heap has a value **no less than** its child nodes. Therefore, the top element (root node) has the **largest** value in the Heap.
  + Min Heap: Each node in the Heap has a value **no larger than** its child nodes. Therefore, the top element (root node) has the **smallest** value in the Heap.

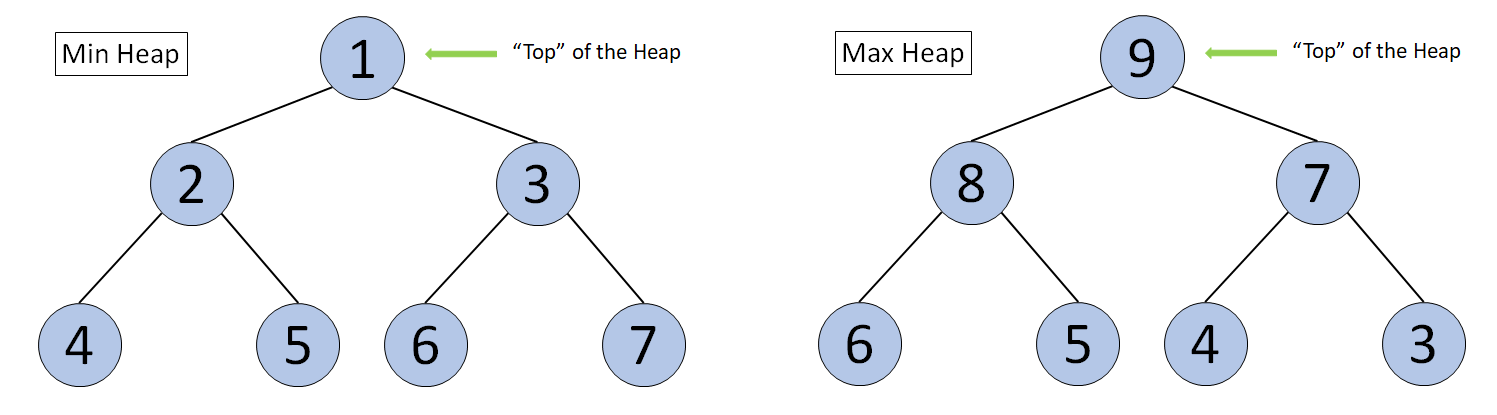


Diagram of a Min Heap and a Max Heap