**DATA STRUCTURES & ALGORITHMS**

**PROJECT TOPIC: HEAP DATA STRUCTURES**

**Group members:**

Fatima Ehsan (BSCE21016)

Nimra Maqbool (BSCE21012)

**Introduction:**

A Heap is a special Tree-based data structure in which the tree is a complete binary tree.

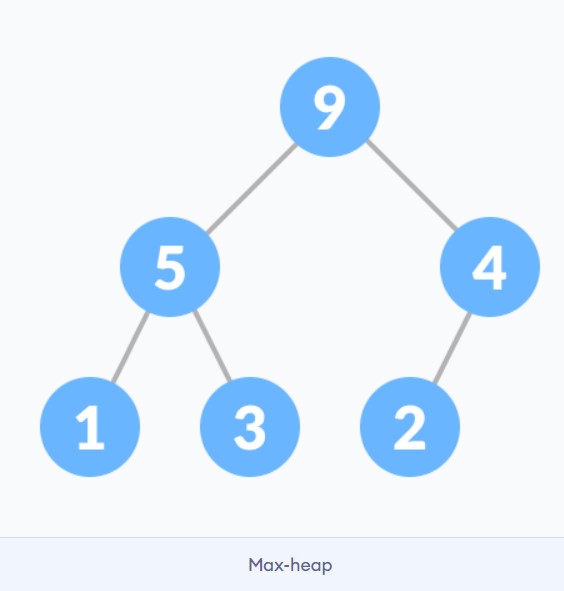
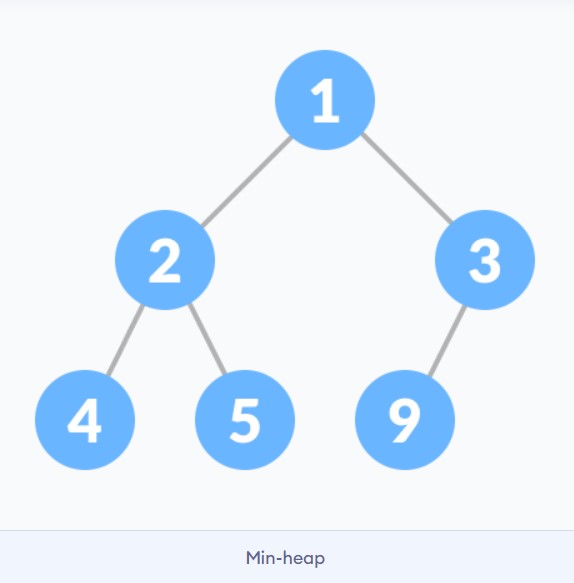
All the nodes of the tree are in a specific order.

For example, if X is the parent node of Y, then the value of X follows a specific order with respect to the value of Y and the same order will be followed across the tree.

**Types of heap DS:**

It satisfies the heap property where any given node is:

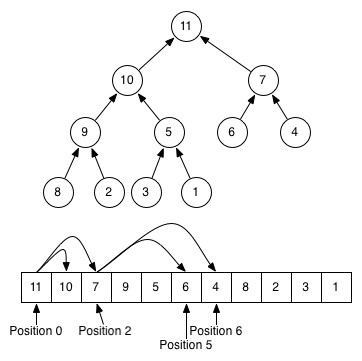
* always greater than its child node/s and the key of the root node is the largest among all other nodes. This property is also called **max-heap property**.
* always smaller than the child node/s and the key of the root node is the smallest among all other nodes. This property is also called **min-heap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

**Major operations of heap data structure:**

* **Heapify:** a process of creating a heap from an array.
* **Insertion:** process to insert an element in existing heap time complexity O(log N).
* **Deletion:** deleting the top element of the heap or the highest priority element, and then organizing the heap and returning the element with time complexity O(log N).
* **Peek:** to check or find the most prior element in the heap, (max or min element for max and min heap).

**Tree representation vs array representation of heap structure:**



* Detailed time and space complexity analysis will be done in the project along with the animation of each operation.

**Sources:** <https://www.geeksforgeeks.org/heap-data-structure/?ref=lbp><https://www.programiz.com/dsa/heap-data-structure>

**PROJECT TOPIC: KMP ALGORITHMS**

**Introduction:**

 KMP algorithm is used **to find a "Pattern" in a "Text"**. This algorithm compares character by character from left to right. But whenever a mismatch occurs, it uses a preprocessed table called "Prefix Table" to skip characters comparison while matching.

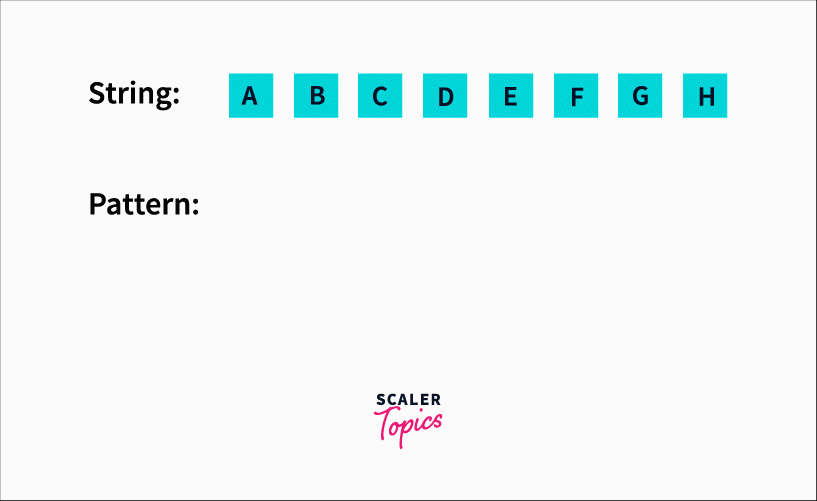
**Example:**

An easy example.

pattern = "def"

string = "abcdefgh"

In the complete string, we need to find a substring that is equivalent to our pattern. The sliding window algorithm would have a window size equal to the length of the pattern. In our case, 3. We would check substrings of length 3 starting from the left (0 index) to right.



**MAJOR OPERATIONS:**

1.String matching:

2.Used in sudoku solver.

3.Search engine.

**TIME COMPLEXITY:**

The worst case complexity of the Naive algorithm is O(m(n-m+1)). The time complexity of KMP algorithm is O(n) in the worst case.

**SOURCES:**

<https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/#:~:text=The%20time%20complexity%20of%20KMP,followed%20by%20a%20mismatching%20character>**.**

<https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/#:~:text=The%20time%20complexity%20of%20KMP,followed%20by%20a%20mismatching%20character>**.**

**Some other suggestions:**

**1. Splay trees**

**2.Self-organizing list.**