Sorting algorithms [edit]

See also: Sorting algorithm § Comparison of algorithms

Algorithm	Data structure	Time complexity:Best	Time complexity:Average	Time complexity:Worst	Space complexity:Worst
Quick sort	Array	O(n log(n))	O(n log(n))	O(n^2)	O(n)
Merge sort	Array	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Heap sort	Array	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Smooth sort	Array	O(n)	O(n log(n))	O(n log(n))	O(1)
Bubble sort	Array	O(n)	O(n^2)	O(n^2)	O(1)
Insertion sort	Array	O(n)	O(n^2)	O(n^2)	O(1)
Selection sort	Array	O(n^2)	O(n^2)	O(n^2)	O(1)

Data structures [edit]

See also: Search data structure § Asymptotic amortized worst-case analysis

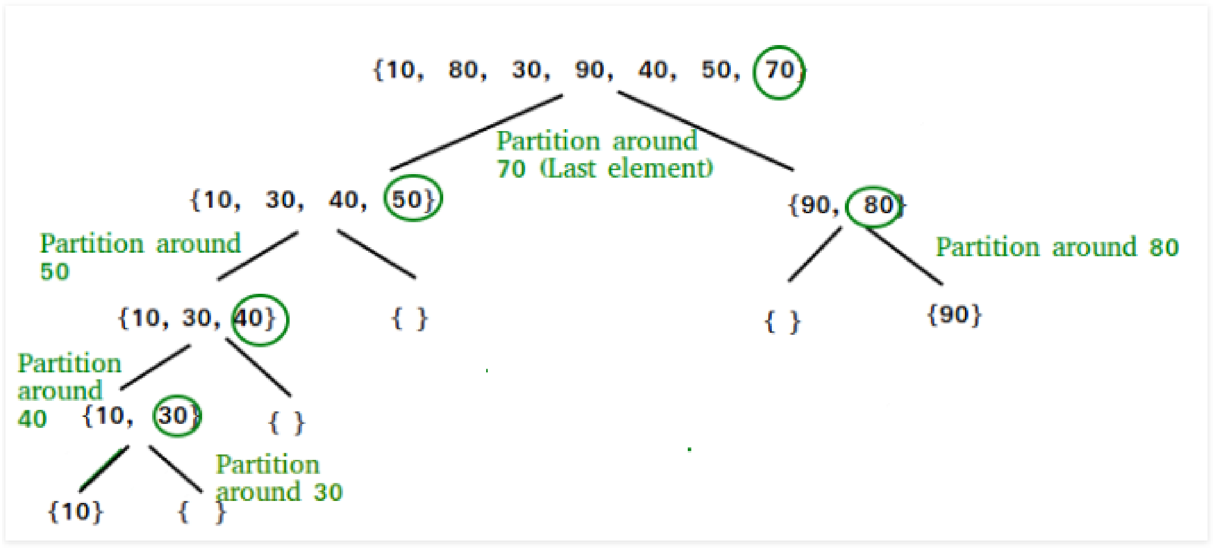
Data structure	Time complexity								Space complexity
	Avg: Indexing	Avg: Search	Avg: Insertion	Avg: Deletion	Worst: Indexing	Worst: Search	Worst: Insertion	Worst: Deletion	Worst
Basic Array	O(1)	O(n)	—	—	O(1)	O(n)	—	—	O(n)
Dynamic array	O(1)	O(n)	O(n)	—	O(1)	O(n)	O(n)	—	O(n)
Singly linked list	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly linked list	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Hash table	—	O(1)	O(1)	O(1)	—	O(n)	O(n)	O(n)	O(n)
Binary search tree	—	O(log (n))	O(log (n))	O(log (n))	—	O(n)	O(n)	O(n)	O(n)
B-tree	—	O(log (n))	O(log (n))	O(log (n))	—	O(log (n))	O(log (n))	O(log (n))	O(n)
Red-black tree	—	O(log (n))	O(log (n))	O(log (n))	—	O(log (n))	O(log (n))	O(log (n))	O(n)
AVL tree	—	O(log (n))	O(log (n))	O(log (n))	—	O(log (n))	O(log (n))	O(log (n))	O(n)

Quick Sort:-

Quicksort is a divide and conquer algorithm. Quicksort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays.

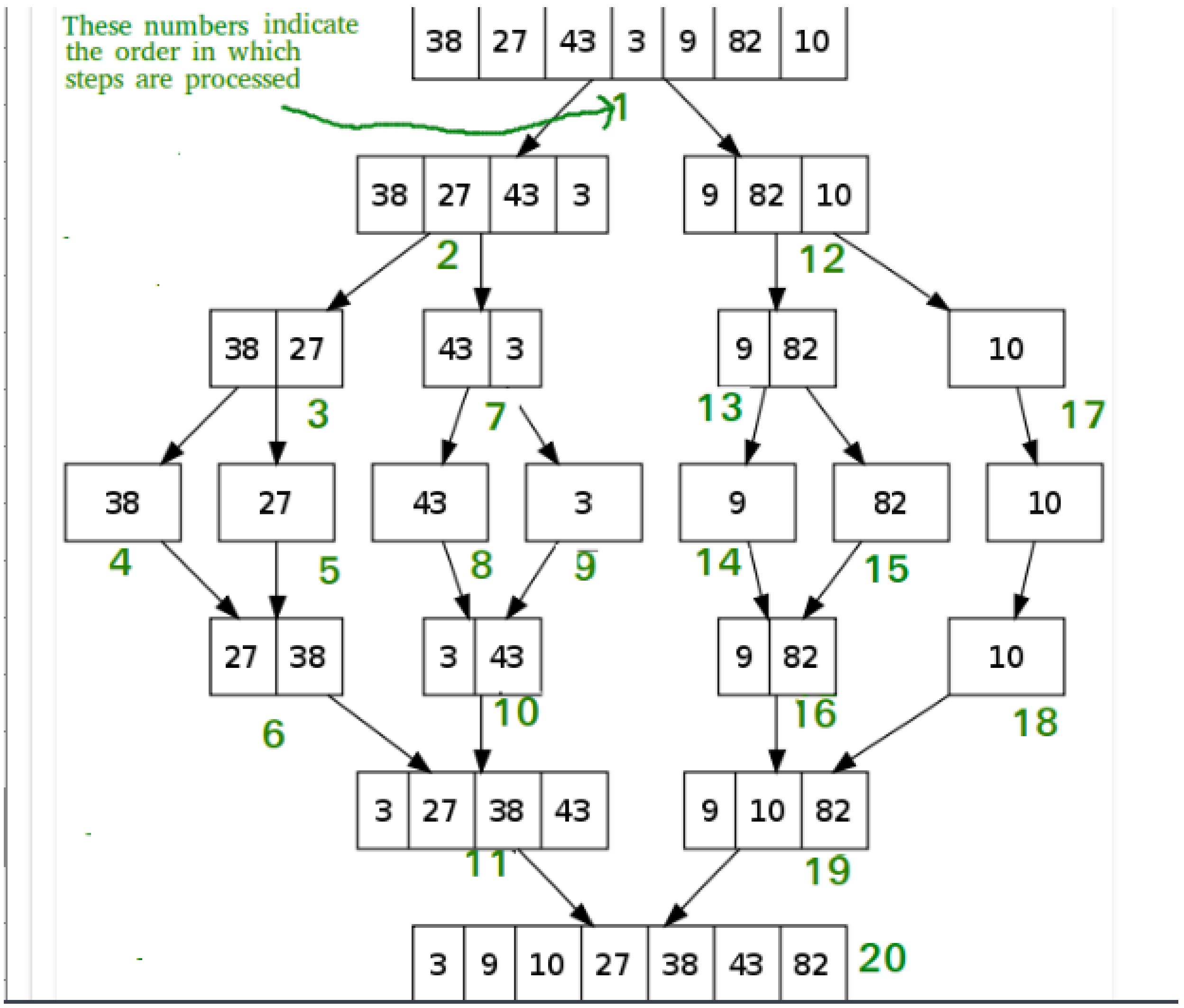
The steps are:

1. Pick an element, called a *pivot*, from the array.
2. *Partitioning*: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.
3. Recursively apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values



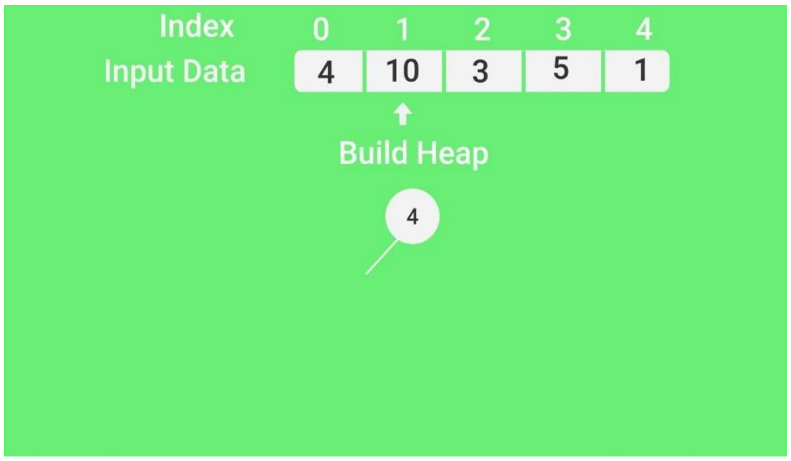
Merge Sort:-

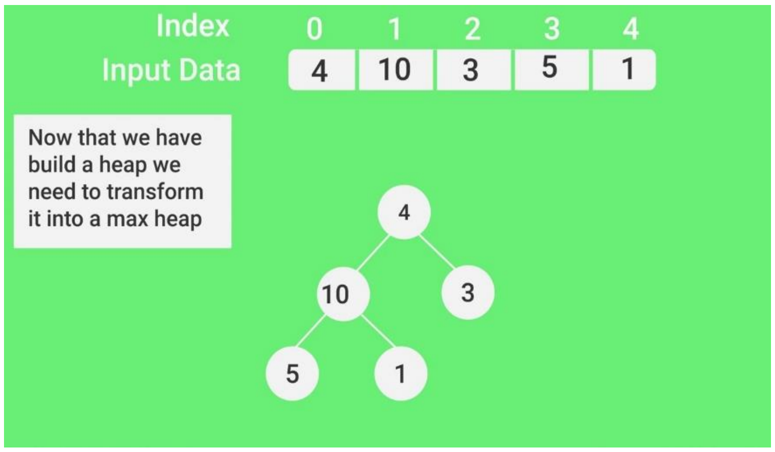
Like QuickSort, Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.

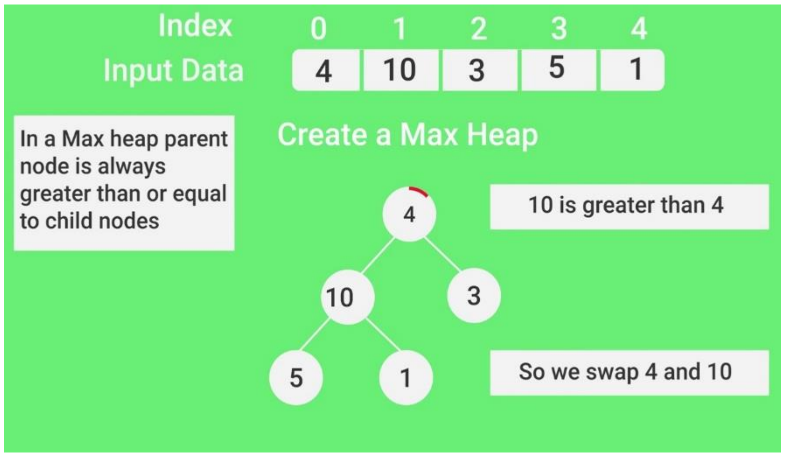


Heap Sort:-

Heap sort is a comparison based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the maximum element and place the maximum element at the end. We repeat the same process for remaining element



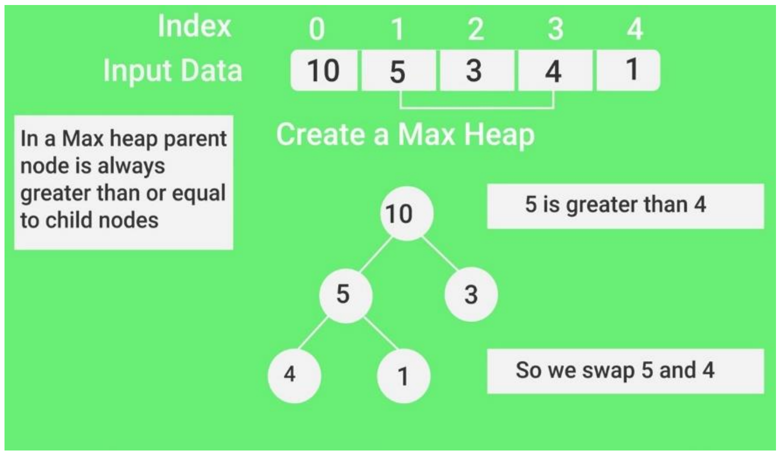

Now that we have build a heap we need to transform it into a max heap


In a Max heap parent node is always greater than or equal to child nodes

Create a Max Heap

10 is greater than 4

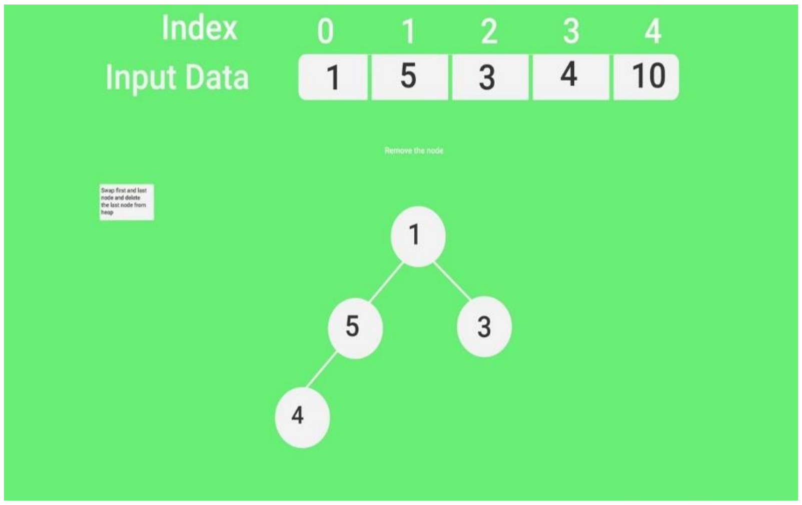
So we swap 4 and 10


In a Max heap parent node is always greater than or equal to child nodes

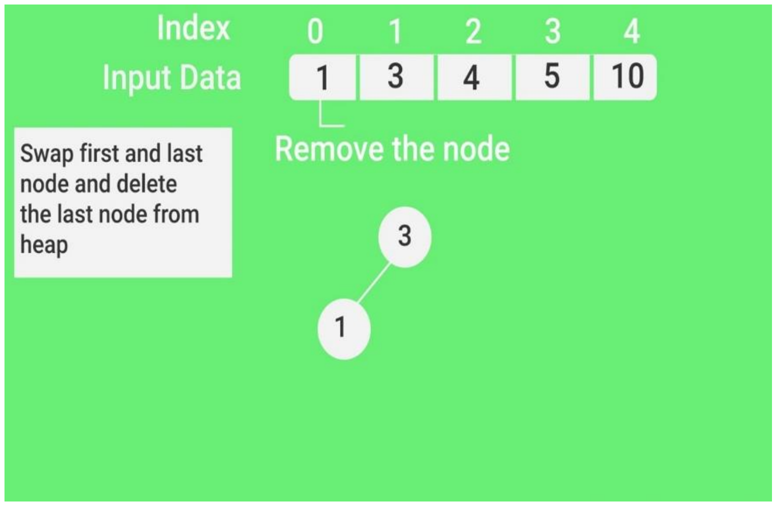
Create a Max Heap

5 is greater than 4

So we swap 5 and 4


Remove the node

Swap first and last node and delete the last node from heap


Swap first and last node and delete the last node from heap

Remove the node

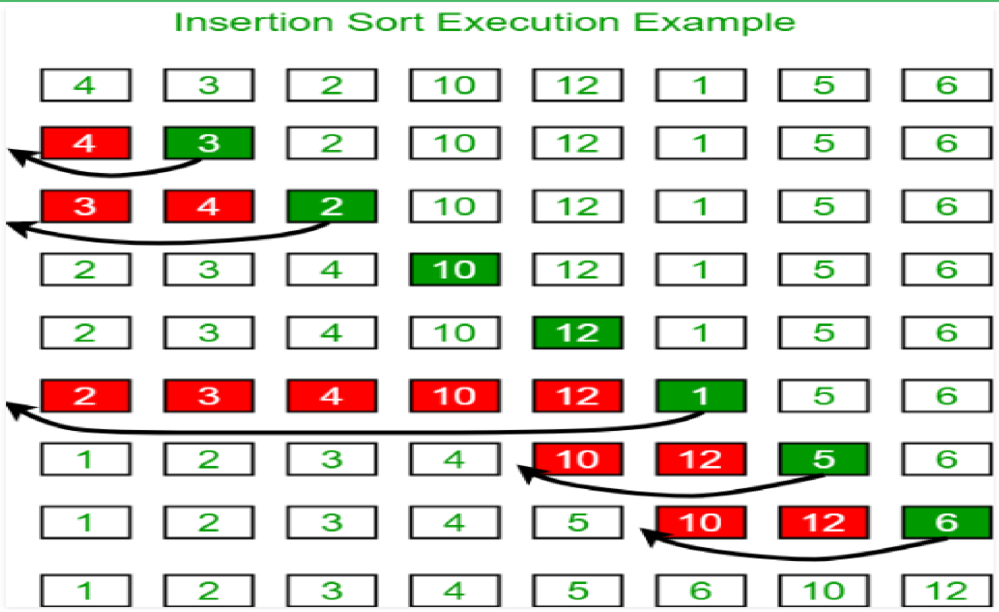
Smooth Sort:-

Like heapsort, smoothsort builds up an implicit heap data structure in the array to be sorted, then sorts the array by repeatedly extracting the maximum element from that heap. Unlike heapsort, which places the root of the heap and largest element at the beginning (left) of the array before swapping it to the end (right), smoothsort places the root of the heap at the right, already in its final location. This, however, considerably complicates the algorithm for removing the rightmost elem

Bubble Sort:-

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
Example:
First Pass:
(5 1 4 2 8) –> (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 5 4 2 8) –> (1 4 5 2 8), Swap since 5 > 4
(1 4 5 2 8) –> (1 4 2 5 8), Swap since 5 > 2
(1 4 2 5 8) –> (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.
Second Pass:
(1 4 2 5 8) –> (1 4 2 5 8)
(1 4 2 5 8) –> (1 2 4 5 8), Swap since 4 > 2
(1 2 4 5 8) –> (1 2 4 5 8)
(1 2 4 5 8) –> (1 2 4 5 8)
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Insertion Sort:-



Selection Sort:-

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning

Basic Array:-

An array is a set of values, which are termed elements, that are logically related to each other. ... Similarly, an array may consist of a student's grades for a class; each element of the array is a single grade. It is possible individual variables to store each of our data items

Dynamic Array:-

In computer science, a dynamic array, growable array, resizable array, dynamic table, mutable array, or array list is a random access, variable-size list data structure that allows elements to be added or removed. It is supplied with standard libraries in many modern mainstream programming languages.

Singly linked list:-

Singly Linked Lists are a type of data structure. In a singly linked list, each node in the list stores the contents and a pointer or reference to the next node in the list. It does not store any pointer or reference to the previous node. ... The last node in asingle linked list points to nothing.

Doubly Linked List:-

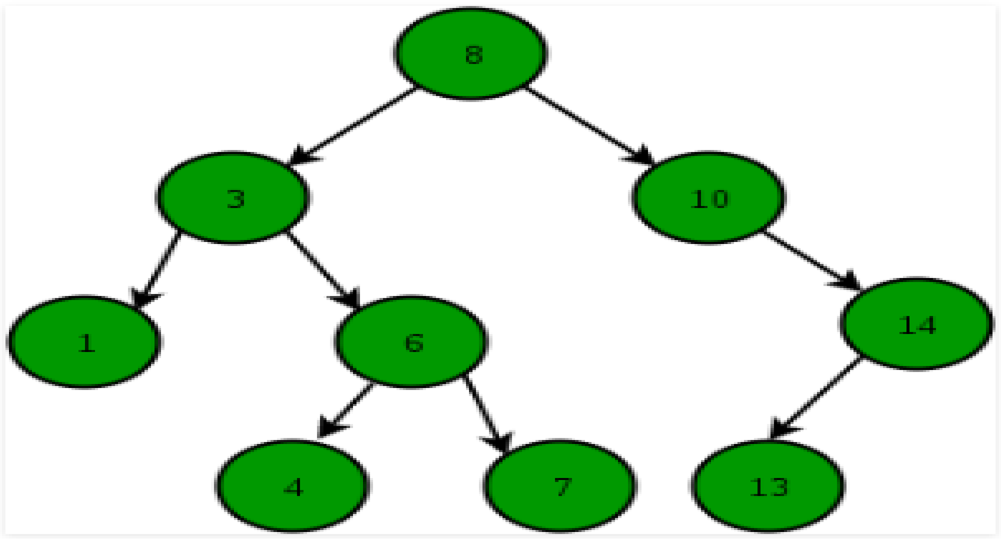
Doubly linked list. In computer science, a doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Eachnode contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes.

Hash table:-

Hash Table is a data structure which stores data in an associative manner. In hash table, the data is stored in an array format where each data value has its own unique index value.

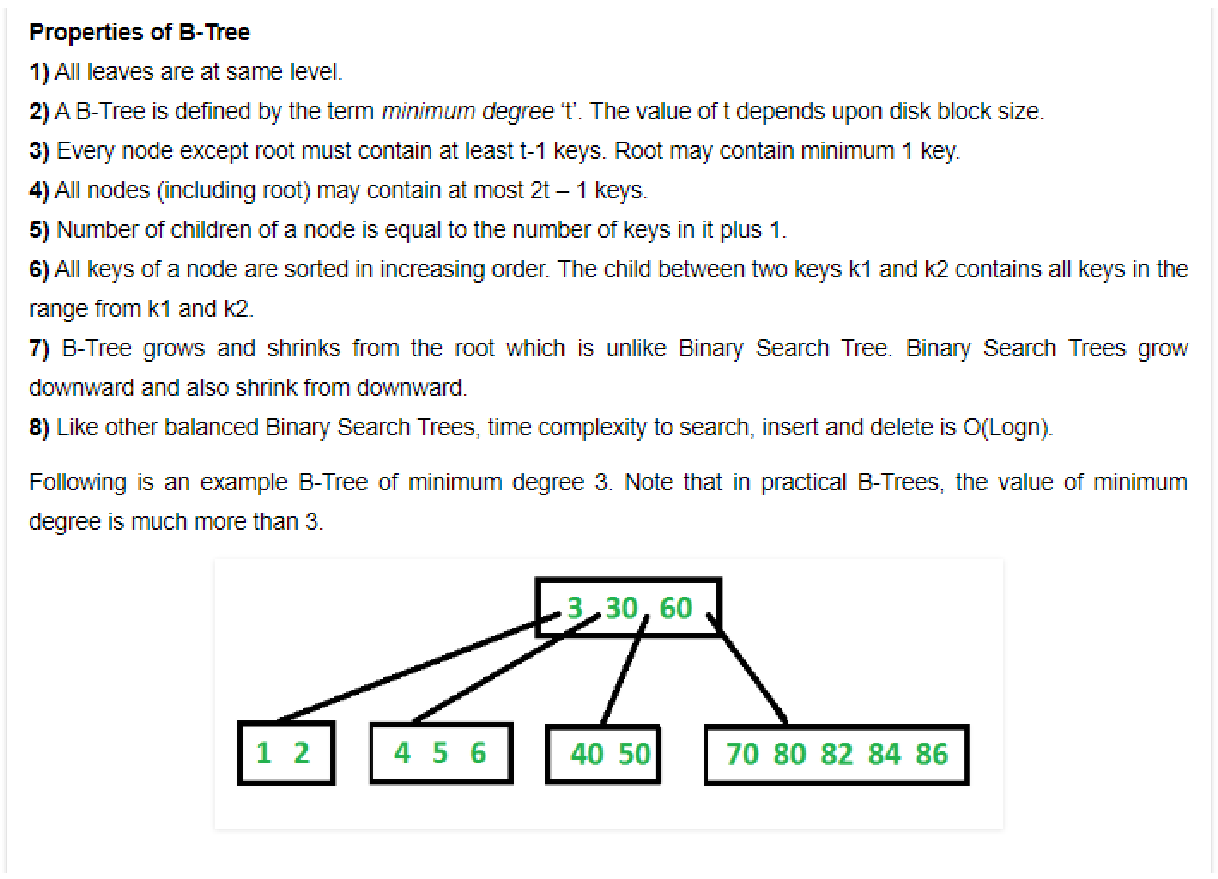
Binary Search Tree:-

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties − The left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key.



B Tree:-

B-Tree is a self-balancing search tree. In most of the other self-balancing search trees (like AVL and Red-Black Trees), it is assumed that everything is in main memory. To understand the use of B-Trees, we must think of the huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses.



Properties of B-Tree

1) All leaves are at same level.

2) A B-Tree is defined by the term *minimum degree* 't'. The value of t depends upon disk block size.

3) Every node except root must contain at least t-1 keys. Root may contain minimum 1 key.

4) All nodes (including root) may contain at most 2t – 1 keys.

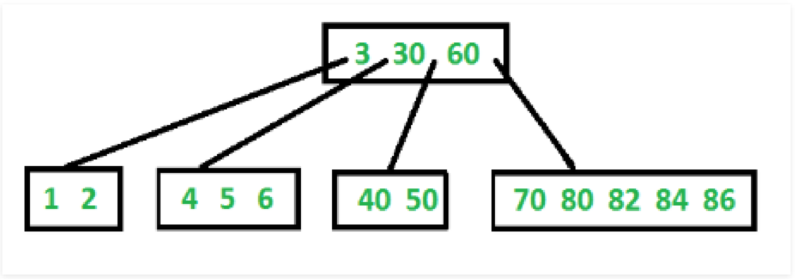
5) Number of children of a node is equal to the number of keys in it plus 1.

6) All keys of a node are sorted in increasing order. The child between two keys k1 and k2 contains all keys in the range from k1 and k2.

7) B-Tree grows and shrinks from the root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink from downward.

8) Like other balanced Binary Search Trees, time complexity to search, insert and delete is O(Logn).

Following is an example B-Tree of minimum degree 3. Note that in practical B-Trees, the value of minimum degree is much more than 3.



Red-Black Tree:-

A red–black tree is a kind of self-balancing binary search tree in computer science. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions.

Red-Black Tree | Set 1 (Introduction)

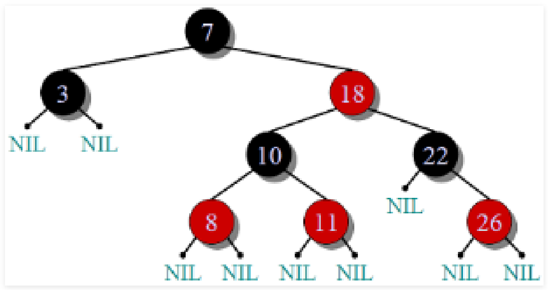
Red-Black Tree is a self-balancing Binary Search Tree (BST) where every node follows following rules.

1) Every node has a color either red or black.

2) Root of tree is always black.

3) There are no two adjacent red nodes (A red node cannot have a red parent or red child).

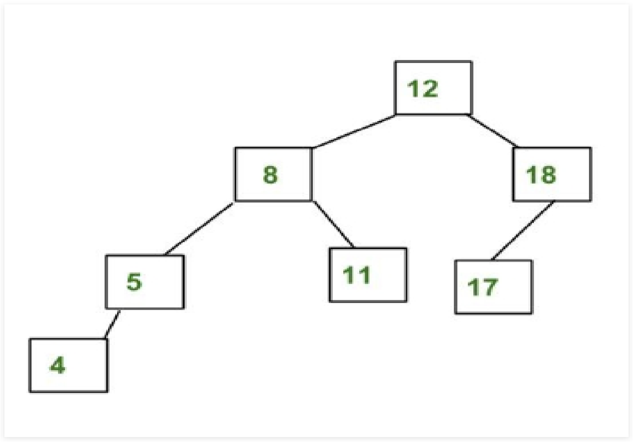
4) Every path from root to a NULL node has same number of black nodes.



AVL Tree:-

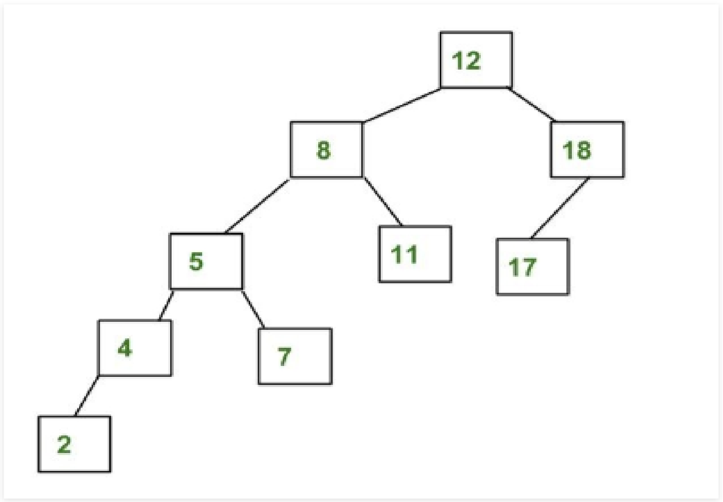
A balanced binary search tree where the height of the two subtrees (children) of a node differs by at most one. Look-up, insertion, and deletion are O(log n), where n is the number of nodes in the tree.

An Example Tree that is an AVL Tree



The above tree is AVL because differences between heights of left and right subtrees for every node is less than or equal to 1.

An Example Tree that is NOT an AVL Tree



The above tree is not AVL because differences between heights of left and right subtrees for 8 and 18 is greater than 1.