Data structures

Big O notation is used to measure how running time or space requirements for your program grow as input size

time complexity :-

for loop including all array values => O(n)

when we given the index to the specified value => O(1)

two for loop which is inside one loop => O(n\*n)

linear data-structure and non-linear data-structures

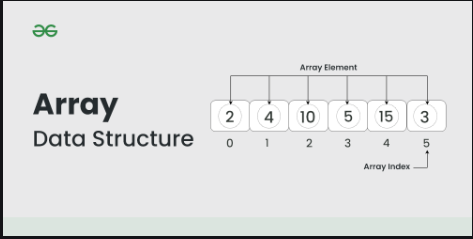
* Data structure where data elements are arranged sequentially or linearly where each and every element is attached to its previous and next adjacent is called a linear data structure.
* In linear data structure, single level is involved. Therefore, we can traverse all the elements in single run only.
* Linear data structures are easy to implement because computer memory is arranged in a linear way.
* Its examples are [array](https://www.geeksforgeeks.org/array-data-structure/), [stack](https://www.geeksforgeeks.org/stack-data-structure/), [queue](https://www.geeksforgeeks.org/queue-data-structure/), [linked list](https://www.geeksforgeeks.org/data-structures/linked-list/), etc.

………………………………….

* Data structures where data elements are not arranged sequentially or linearly are called non-linear data structures.
* In a non-linear data structure, single level is not involved.
* Therefore, we can’t traverse all the elements in single run only.
* Non-linear data structures are not easy to implement in comparison to linear data structure. It utilizes computer memory efficiently in comparison to a linear data structure.
* Its examples are [trees](https://www.geeksforgeeks.org/data-structures/) and [graphs](https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/).

array :-

* An array is a collection of items stored at contiguous memory locations.
* The idea is to store multiple items of the same type together.
* This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array



Time complexity for array:-

* Lookup by index => O(1)
* Lookup by value => O(n)
* Array traversal => O(n)
* Insert at 1st index => O(n) because we have n number of swaps if we insert at first position
* delete at 1st index => O(n) because we have n number of swaps if we delete at first position
* insert/delete at end => o(1)
* insert/delete at middle => o(n)

array types :-

static array => fixed size

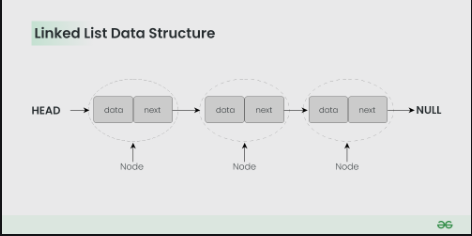
dynamic array => variable size

linked list :-

* A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.
* a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

issues with array that linked list tends to solve :-

in array the insertion and deletion has a time complexity of O(n) due to the process of swapping of elements while in linked list the insertion and deletion operation is simpler there is no need for swapping of elements



Time complexity of linked list :

* Insert or delete at beginning => O(1)
* Insert or delete at end or middle => O(n)
* Linked list traversal => O(n)
* Accessing elements by value => o(n)

Double linked list :-

A doubly linked list (DLL) is a special type of linked list in which each node contains a pointer to the previous node as well as the next node of the linked list



Stack:-

Last in first out

Uses :

Browser history

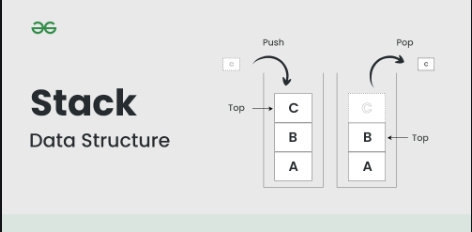
Undo operations

Function calling in recursion

Time complexity :-

Push/pop element => O(1)

Search element => O(n)



Queue

First in first out

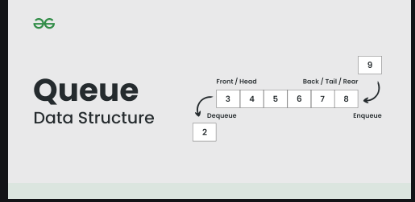
Uses :-

Used in producer consumer problem where one entity produces the information and other entity consumes the information

Time complexity :-

Push/pop element => O(1)

Search element => O(n)



Hash table

Hashing is a technique or process of mapping keys, and values into the hash table by using a hash function.

It is done for faster access to elements.

The efficiency of mapping depends on the efficiency of the hash function used.

Time complexity :-

Look up by key => O(1)

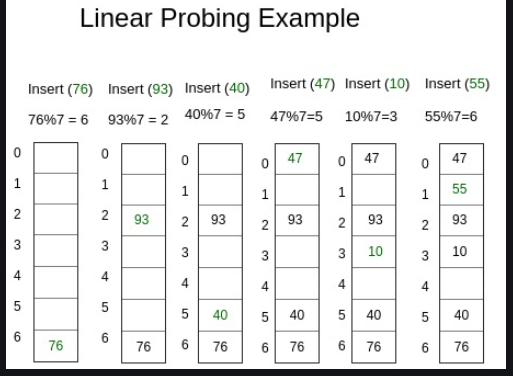
Insertion/deletion => O(1)

When collision occurs and the hash function is poor, searching time complexity => O(n)

A diagram of a number

Description automatically generated

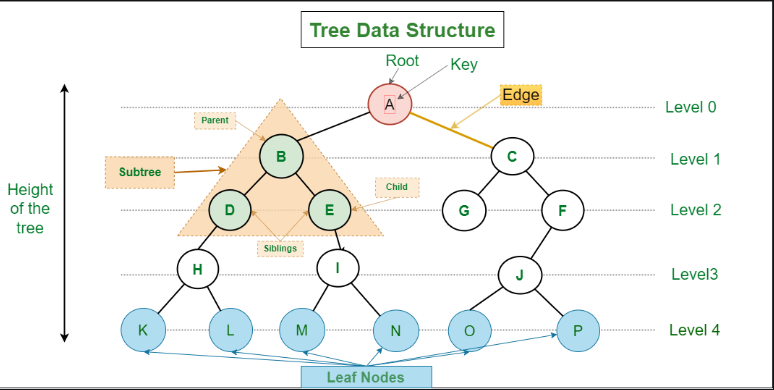
Linear probing is used to solve this problem



Tree

Eg :- ecommerce website

* A tree data structure is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search.
* It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.



Binary tree :-

In a binary tree, a node can have maximum of two children.

Time complexity :-

* Searching => o(n)
* Insertion => o(n)
* Deletion => o(n)

Binary search tree :-

BST is a special type of binary tree in which the left child of a node has a value less than the parent and the right child has a value greater than the parent.

Time complexity :-

* Searching => o(n)
* Insertion => o(n)
* Deletion => o(n)

AVL/ Height Balanced Tree:

AVL tree is a binary search tree with an additional property that the difference between the height of the left sub-tree and the right sub-tree of any node can’t be more than 1

Time complexity :

Searching => O(log(n))

insertion => O(log(n))

deletion => O(log(n))

Graph :-

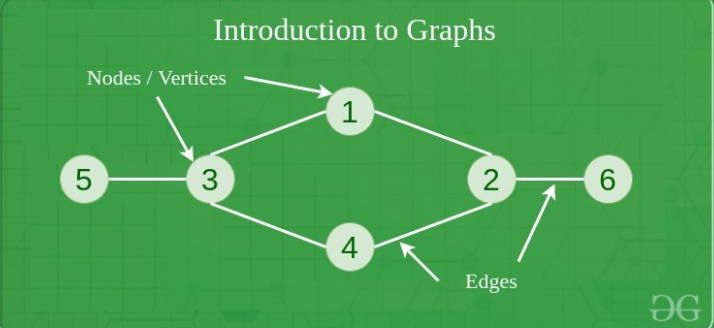
In tree only one path between nodes

But in graph, one node has multiple paths between nodes

Eg :-

Social media friends suggestion

Flight routes



Heap

A diagram of a diagram of a number of data

Description automatically generated with medium confidence

* A Heap is a special Tree-based data structure in which the tree is a complete binary tree.
* A binary tree is said to be a complete binary tree if all its levels, except possibly the last level, have the maximum number of possible nodes

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 first (or can say the top) element of the heap. O(1)