Data Structure:

It is a way to organising data in a way such that enables it to be processed in the efficient way.

Abstract data types:

The abstract datatype is special kind of datatype, whose behaviour is defined by a set of values and set of operations.

Algorithms:

Set of rules followed to solve a problem

Array:

Collection of similar data types

Properties:

- Memory is allocated in a sequential manner

- size of an array is fixed and cannot be changed

- contain data of similar data type

- each element has an unique index, thus accessing an element in an array is easy

Disadvantages:

The size of an array is fixed and the memory allocated cannot be deleted in an array

Linked list:

A linked list is a linear data structure where each element is a separate object. Each element (node) of a list comprises of two items-data and a reference to next node. The most powerful feature of linked list is that it is of variable size.

Object => { data, ref }

Array vs Linked List:

* Separate object
* Variable size
* Random access

|  |  |
| --- | --- |
| Array | Linked List |
| Entire array is a single object | Each element is an object |
| Has fixed size | Has variable size |
| Any element can be accessed | Element can only be accessed through traversal |

Node: Contains data and reference to next node

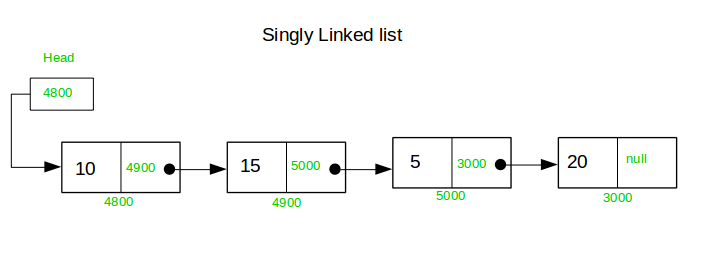
Head: reference to the first node in the list

Tail: reference to the last node in the list

[ The last node will contain a null value stored in ref ]

Single linked list:

In a singly linked link list each node is the list stores the data of the node and a reference to the next node in the list. It does not store any reference to the previous node



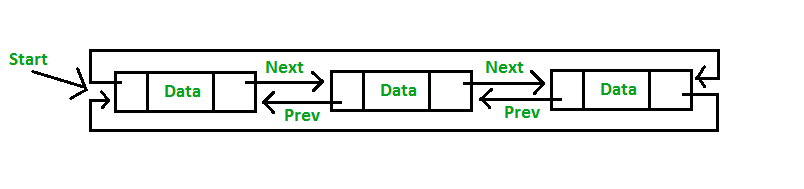
Double linked list:

In double linked list each node contains two references, that references to the previous and next node



Circular double linked list:

In the case of circular double linked list ,the only change that occurs in that the end of the given list linked back to front of the list and vice versa



Tree:

* Used to represent data in hierarchical form
* Every Node (ideally) has 2 components (Data & reference)
* It has a root node and 2 disjoint binary tree called left subtree and right subtree

Tree Terminologies:

1. Root: Node with no parent
2. Edge: Link from parent to child
3. Leaf: Node with no children
4. Sibling: Children of same parent
5. Ancestor: predecessors for a given node
6. Depth of node: length of path from root to node
7. Height of node: length of the paths from that node to the deepest node
8. Height of tree: same as the height of root node
9. Depth of tree: Same as the depth of tree
10. Predecessor: Predecessor of a node is the intermediate previous node in order traversal of the binary tree
11. Successor: Successor of a node is the immediate next node in order traversal of the binary tree

Binary Tree:

A tree is called as binary tree if each node of a tree has a maximum of two nodes.

It is used in solving:

* Huffman coding,
* Heap(priority queue)
* Expression parsing

Types:

* Strict binary tree: if each has two children or none
* Full binary tree: if each has two children and all the nodes are at the same level
* Complete binary tree: if all levels are completely filled except possibly the last level has all the keys as left as possible

Graph:

It is a pair of V and E where E is the set of edges and V is the set of vertices in the graph, vertices are connected with the edges in a graph

1. Vertices: Vertices are the nodes of graph
2. Edges: Path connecting two nodes
3. Unweighted graph: A graph not having a weight associated with any edge
4. Weighted graph: A graph having weight associated with each edge
5. Undirected graph: It is graph that is a set of vertices connected by edges, where the edges don’t have a direction associated with them
6. Directed graph: Edge associated with a direction in graph
7. Cyclic graph: A graph having at least one loop
8. Acyclic graph: A graph having zero loop
9. Tree: It is a special case of Directed Acyclic graph(DAG)
10. Adjacency Matrix: It is a square matrix used to represent graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in a graph
11. Adjacent List: It is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbours of a vertex in the graph

Sorting:

Types:

* Internal: In the same device
* External: In external device

In place sort:

No extra memory is required to sort

Not in place sort:

Extra memory is required to sort

Iterative sort:

* Selection sort
* Bubble sort
* Insertion sort

Recursive sort:

* Quick sort
* Merge sort