(Abstract Data types (ADT))

(Abstract Data type)

Abstraction → (OOP)

(Show essential info and Hide the implementation)

Example

Mobile phone.

Programe Example

Array, Stack, Queue.

(Abstract data type on Array)

Abstrait/logical view

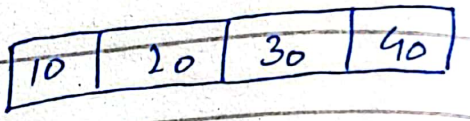
create();

set();

get();

remove();

Data Structure eg



Implementation view →

```
int arr[5] = {10, 20, 30, 40}
cout << arr[1];
arr[2] = 10;
```

(Advantage)

Let say, if someone wants to use the stack in the program, then he can simply use push and pop operation without knowing its implementation. Also, if in future, the implementation of stacks is changed from linked list, then the user program will work in the same way without being affected..

(Operations on Arrays)

(Traversal)

(visiting every element of the array at once). O(n-p)

(Insertion)

(Insert element at specific position) O(n-p)

(Deletion)

(Delet element from array at specific position). O(n-p)

(Searching)

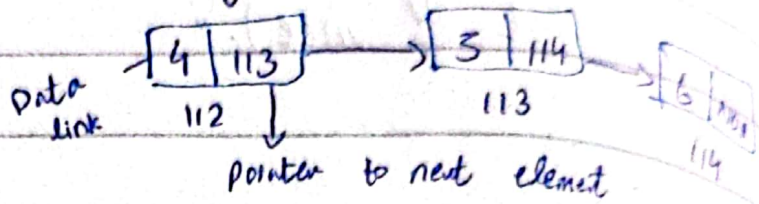
(Search element from the array)

Linear Search
O(n)

Binary search
O(logn) (fast code)

(Linked List)

Linked list are similar to arrays. In linked list elements are stored in non-contingous memory locations.



Why linked list?

In linked list, we can keep adding removing element without any capacity constaints.

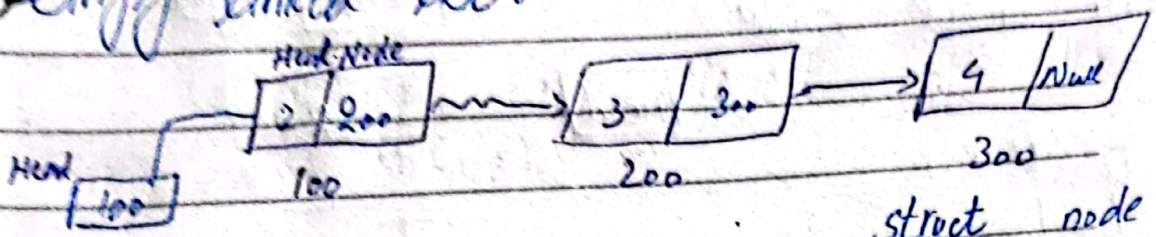
Advantages and Draw-Backs

Extra meomry space for pointers is required.
Random access are not allowed.
Insertion and Deletion easier from array.
Time complexity of accessing element is O(n).
Binary search is not possible in linked list.
It is of dynamic size.

Array VS linked list

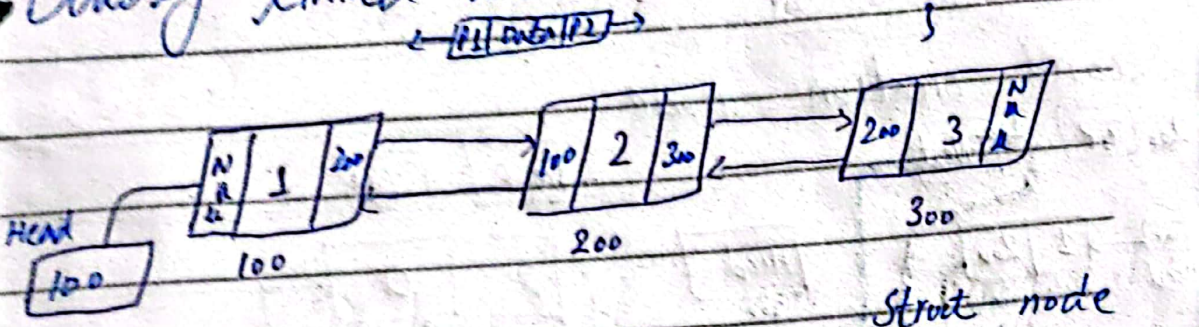
(Types of linked list)

Singly linked list :-

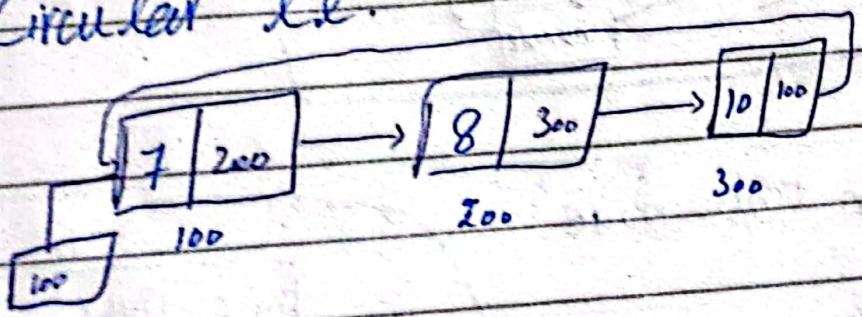



```
struct node
{ int data;
  struct node * next;
}
```

Doubly linked list

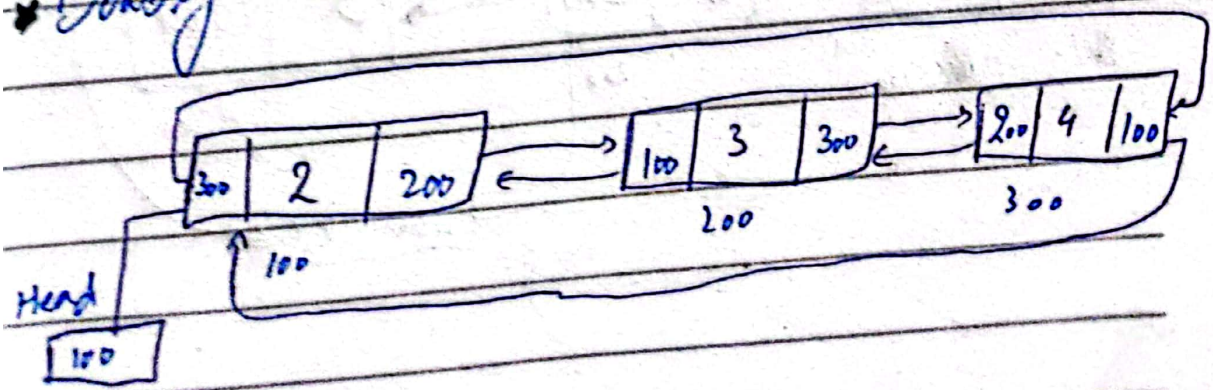


Circular l.l :-



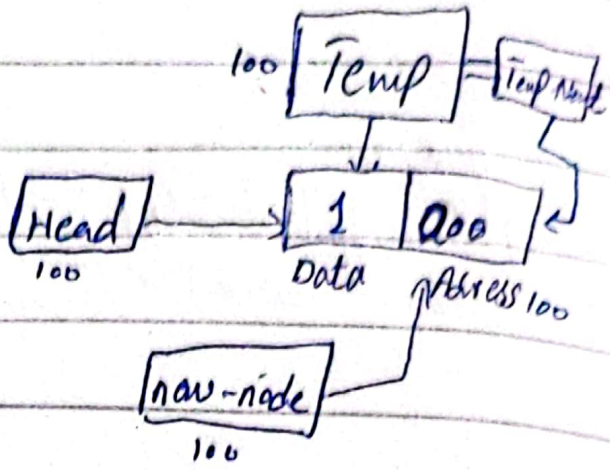

```
Strut node
{ int data;
  strut node * next;
  strut node * prev;
}
```

Doubly Circular ll :-



Singly Linked List

First create node

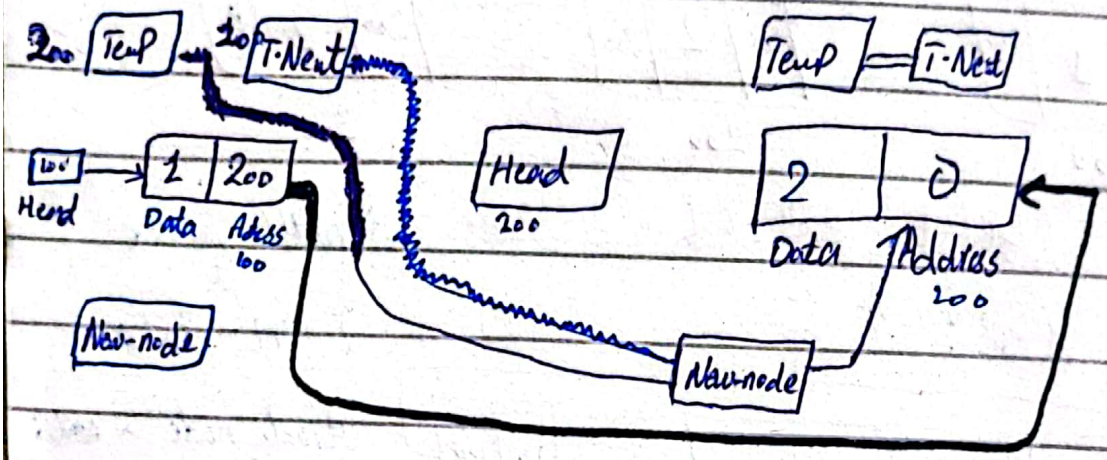


if (true)

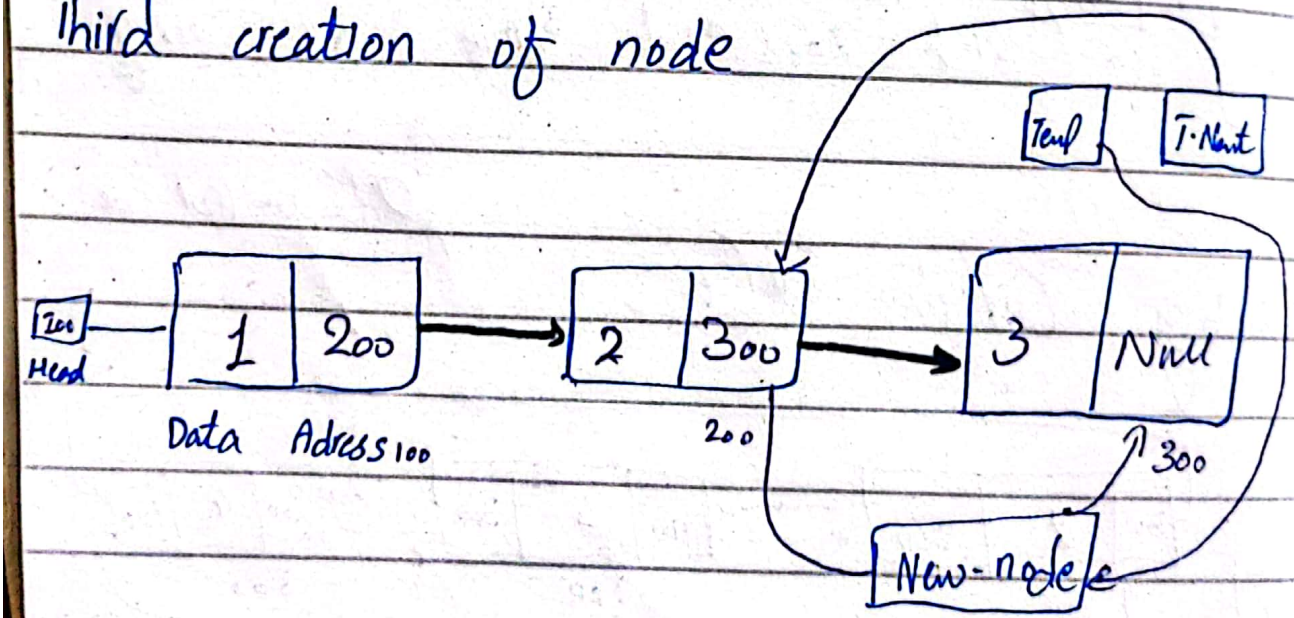
head == new-node

head = 100

Second creation node

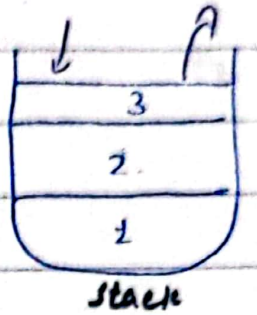


Third creation of node



Stack

Stack is a linear data structure.



Rule:-

Insertion and deletion is possible only from one end. Stack are performed in LIFO and FILO order.

Operations on stack:-

Push (a)
POP ()
Peek () / Top()
isEmpty()
isfull()

A stack can be implemented using a array or a linked list.

Applications:-

Reverse a string
Undo (Text editor
Recursion / Function c
{ }
Infinin to postin/p

$$\begin{array}{r} 20 \\ 2A \\ \hline 4A \end{array}$$

$$\begin{array}{r} 20 \\ 8F \\ \hline AF \end{array}$$

$$\begin{array}{r} 24 \\ -16 \\ \hline 8 \end{array}$$

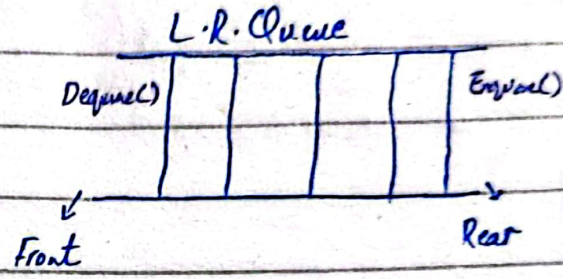
$3 \times 8 = 24$

8F

Queue

It is a linear data structure
It is also an abstract data type.

Rule:-
Insertion and deletion preform two (both) ends. Queue are performed in FIFO and LILO order.
T.C (O(1))



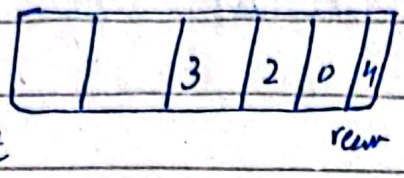
Operations on Queue
Enqeue()
Dequeue()
Front() / Peek()
isFull()
isEmpty()

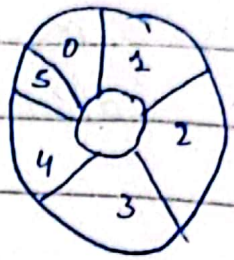
Applications
Printer
Single shareable source
Customer Care
Processor

Queue can be implemented by using arrays, linklist and stacks.

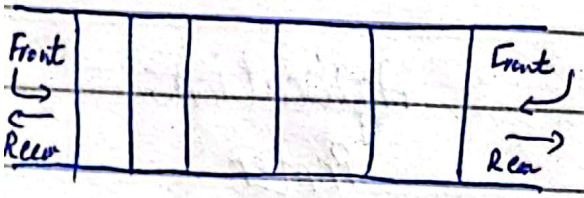
Circular Queue

Draw-back of Queue

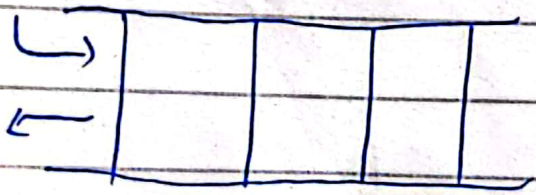




Deque (Double Ended Queue)



This is also perform stack operations. By using some restrictions.



Applications

Redo / Undo

Palandrom checker

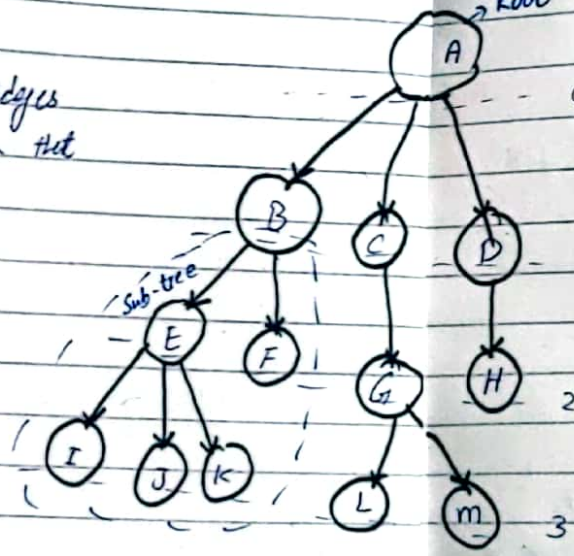
(Multiprocessor Scheulding)

Non-Linear Data Structure

Tree can be as a collection of entities (nodes) linked together to simulate hierarchy.

Height of node = No. of edges in the longest path from that to a leaf node.

Nodes = (n-1) edges



Ancestor:-

Any predecessor (previous) node on the path from root to that node

- L => (A, C, G)

Decendant

Any successor node on the path from that node to leaf node

C => (C, L, M)

Tree

Height of tree = It measured of how deep the tree is or how many levels it has.

max. level = = Height of tree

Level of tree is 3

Root = A

nodes = A to M

Parent nodes = G is parent to L & M

Child nodes = L & M are child of G

Leaf node = I, J, K, F, L, M, H

Non-leaf node = A, B, C, D, E, G

Path = Sequence of consecutive edges from source node to destination node

Sibling = Children of same parent.

Degree = Degree of node is no. of children of that node

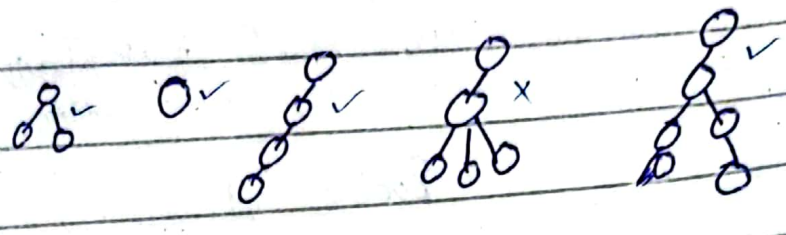
Degree of tree max degree among all node

Depth of node = Length of path from root to that node

Level = Height of tree

Binary Tree

Each node can have atmost 2 children (0, 1, 2)



Max no of nodes possible at any level i is 2^i (i) power.

Formula to find maximum no. of nodes.

Maximum number of Nodes (N) of h $= 2^{(Height+1)} - 1$

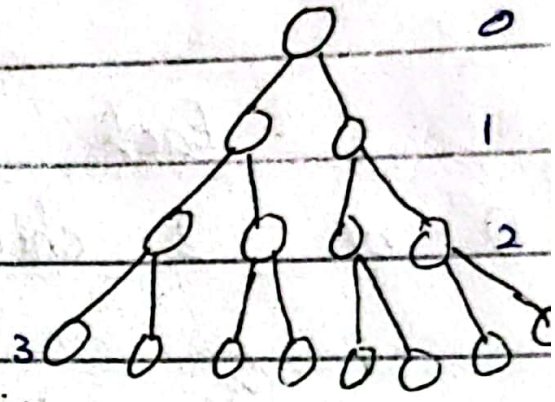
- N represents maximum number of in the BT.
- Height represents the height of BT (measured in term of edges)
- +1 for the root node because levels (o) start (o).

Break-down:-

$2^{(h+1)}$. This represents the max no. of nodes you can have at each level of tree.

-1. This -1 for root node of the tree, which is at level 0 (height 0). Since the root is always present, we substract 1 from the

total number of possible nodes at each level.



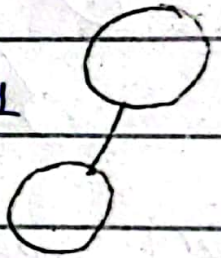
Max no. of nodes $(N) = 2^{(3+1)} - 1$

$= 2^4 - 1$

$= 16 - 1$

max no. of nodes $= 15$ (edges 14)

Formula to find minimum no. of nodes of height h



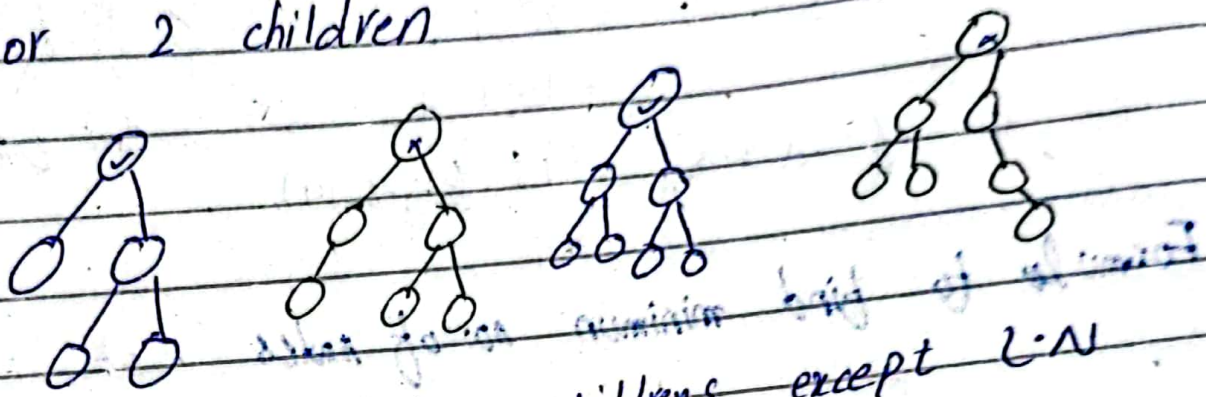
minimum no. of nodes $= h + 1$

$1 = 2$

Types of Binary Tree

- Full / Proper / Strict

Each node can have either 0 or 2 children

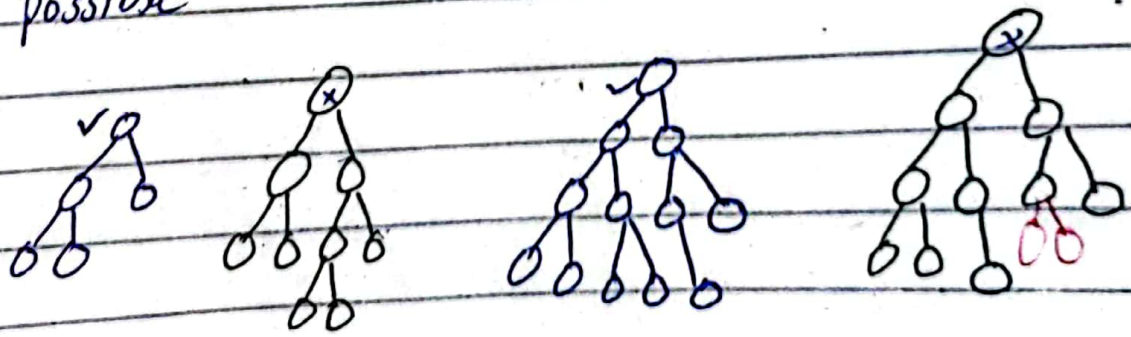


Every node contain two childrens except L.N

No. of leaf nodes == No. of external n+1.

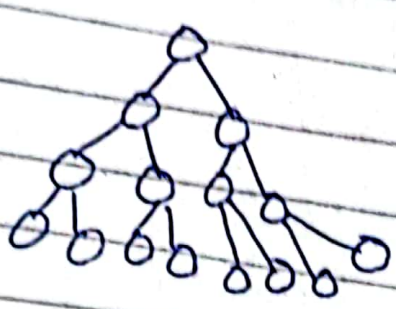
- Complete Binary tree

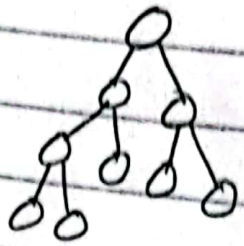
All the levels are completely filled (except possibly the last) and the last level has nodes as left as possible.



- Perfect Binary tree:-

All internals nodes have 2 children and all leaves are at same level.

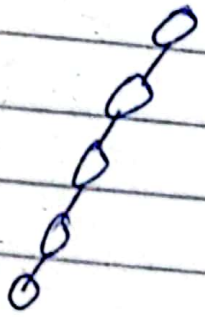




Perfect tree can be full B.T and CBT.

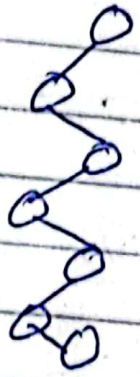
- Degenerate tree

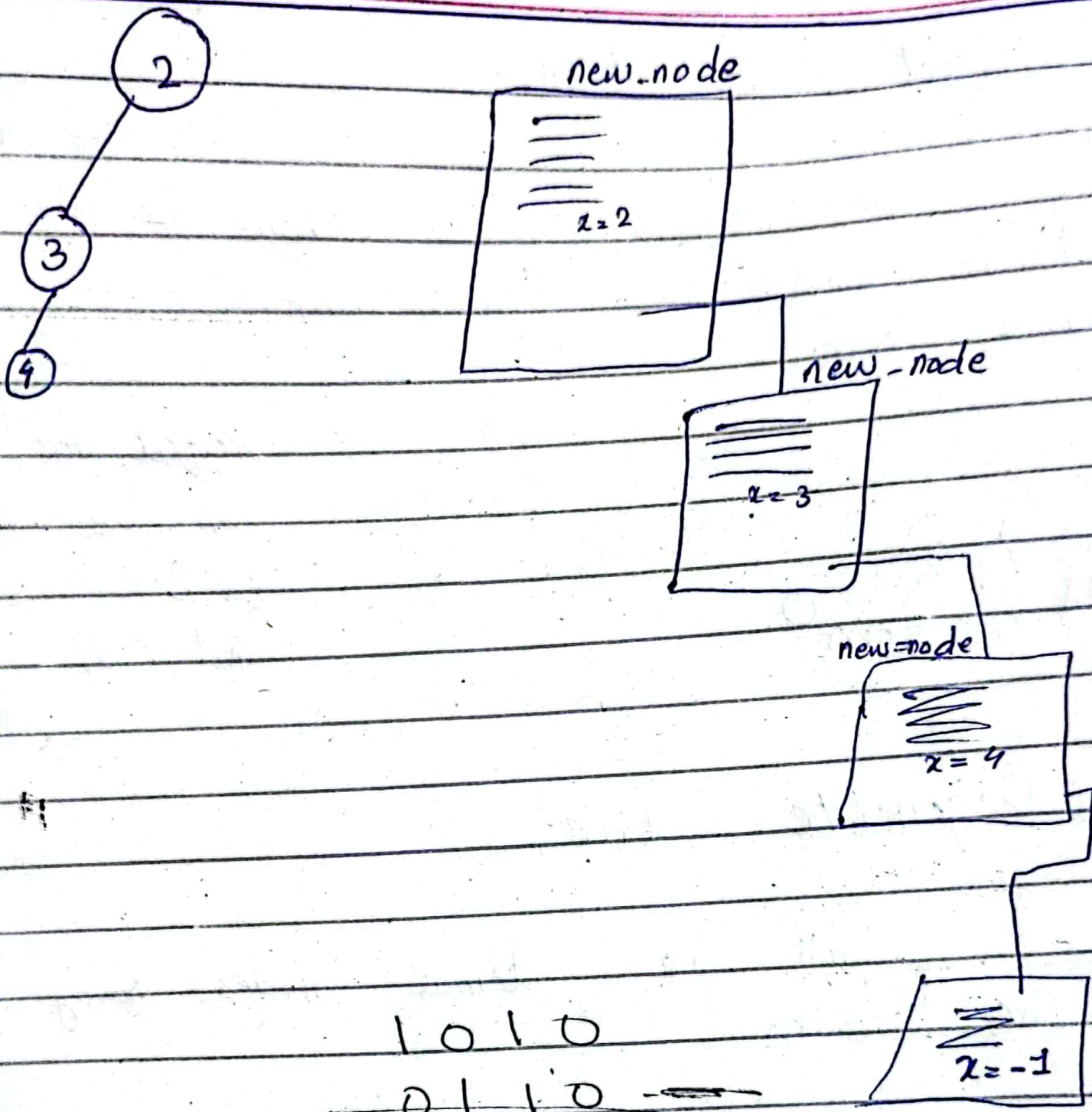
All the internal nodes having one child.



Left skewed B.T

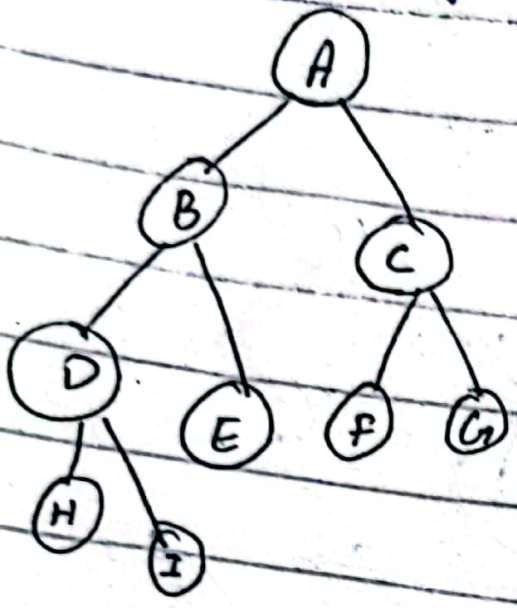
Right skewed B.T





1 0 1 0
0 1 1 0
0 0 0 0
1 0 1 0 0
0 0

Array Representation (B.T)



0	1	2	3	4	5	6	7	8
A	B	C	D	E	F	G	H	I

Formulas to find :-

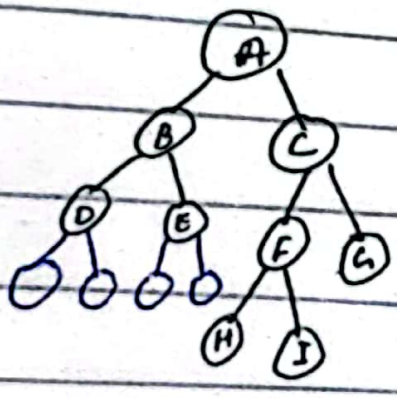
node is at i^{th} index

left child would be at $[(2*i)+1]$

Right child would be at $[(2*i)+2]$

Parent = $\left[\frac{(i-1)}{2}\right]$

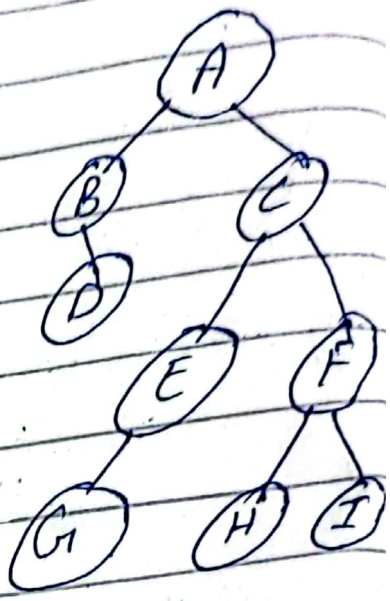
For appling these formula Binary tree should be completely Binary tree.



0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	-	-	-	-	H	I

So, this is the wastage of space.

Traversal in B.T



In-Order

Left-Root-Right

(B D A G E C H F I)

Pre-Order

Root-left-Right

(A B D C E G F H I)

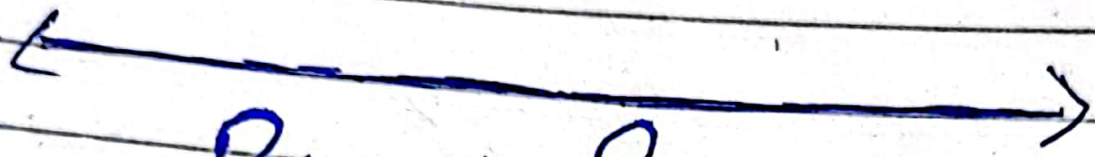
Post-Order

Left-Right-Root

(D B G E H I F C A)

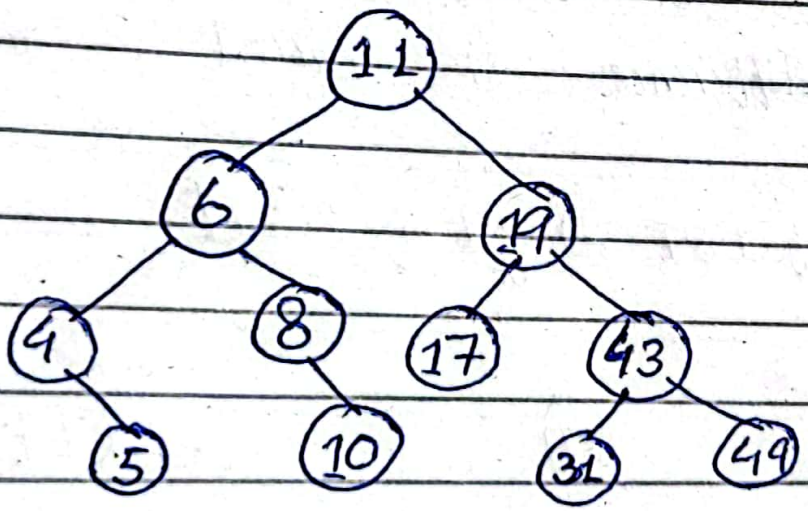
Level-order

(A B C D E F G H I)



Binary Search Tree

11, 6, 8, 19, 4, 10, 5, 17, 43, 49, 31



In-order (T) always sorted order in BST

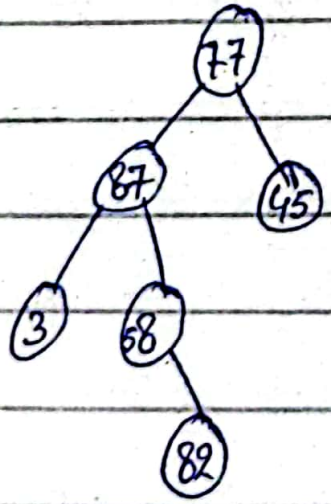
AVL (TREE)

(Avl tree is a self balancing (BST) Tree. where the difference between heights of left & right sub-trees cannot be more than for all nodes.

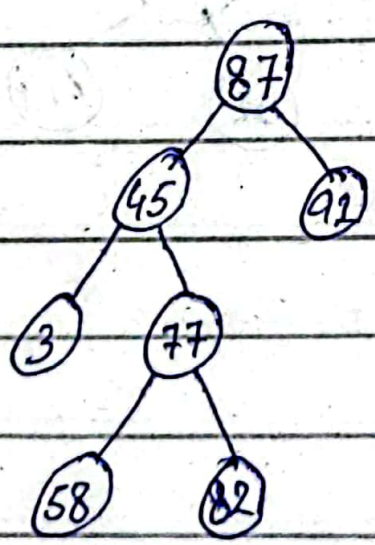
This difference is called Balance factor

BF (-1, 0, 1).

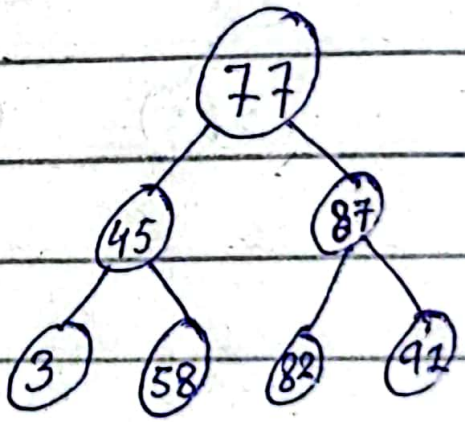
| Height of L.S.T - Height of R.S.T | = BF



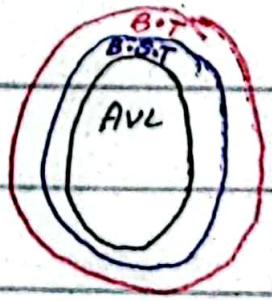
Binary Tree



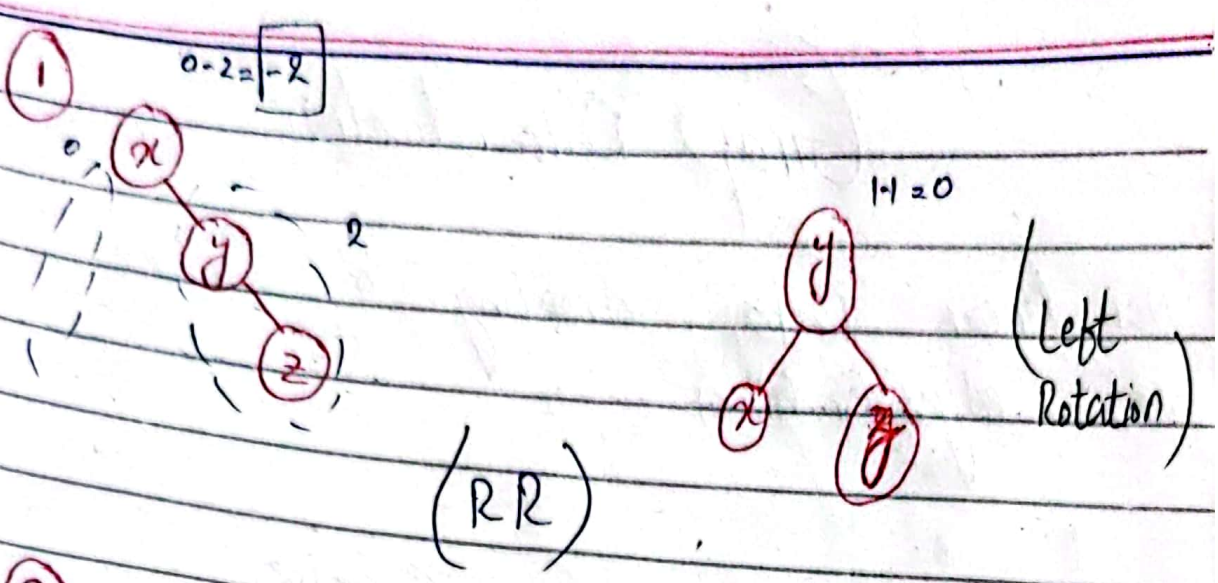
Binary Search Tree



Avl Tree



1

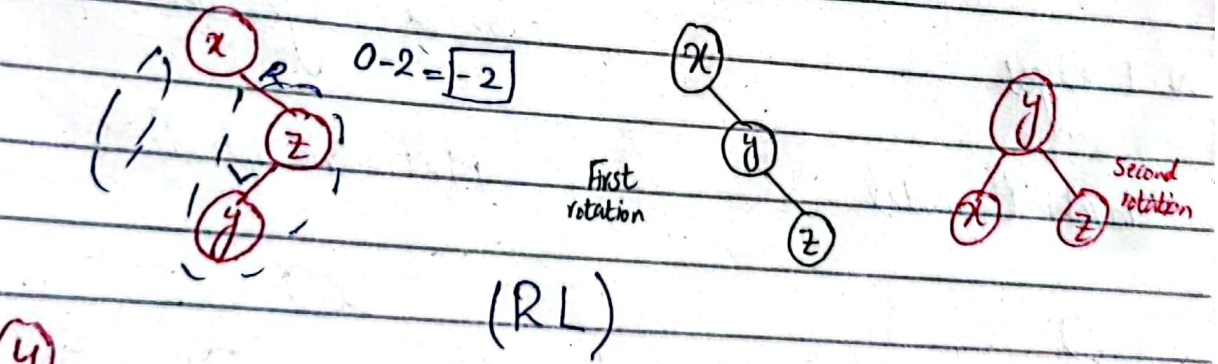


(RR)

2

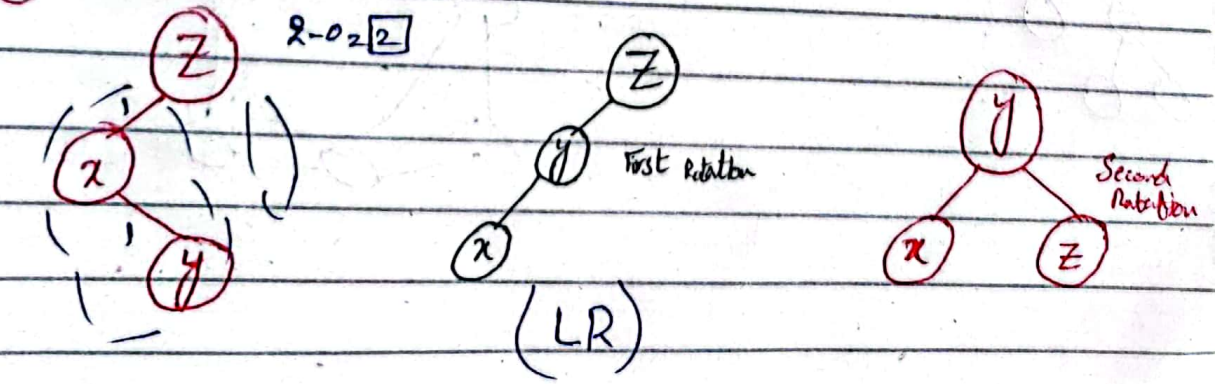
(LL)

3



(RL)

4

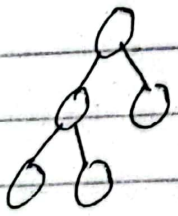


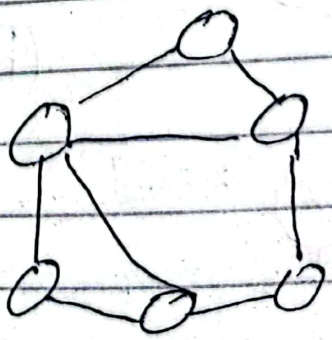
(LR)

Graph Data Structure

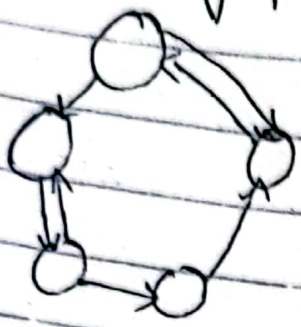
Non-linear Data consisting of nodes and edges. (or links)

Trees	Graphs
Only one path/edge b/w two nodes	Multiple paths b/w 2 nodes
Has a root	No root node
Don't have loops	Can have loops
N-1 edges	No. of edges not def
Hierarchical model	Network model.

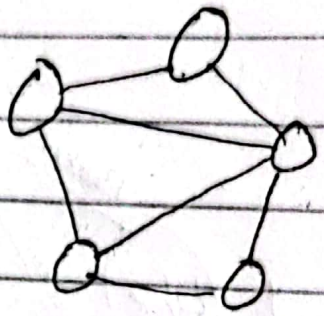




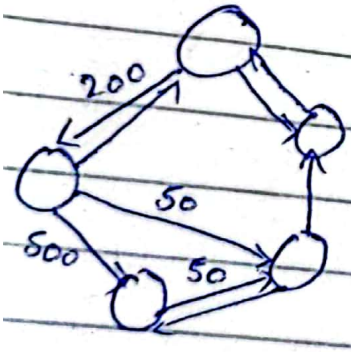
Directed graph



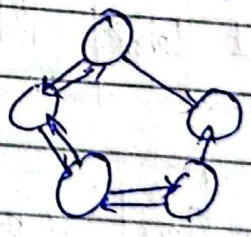
Undirected Graphs



Weighted Graph
It is a graph in which each branch is given numerical value.

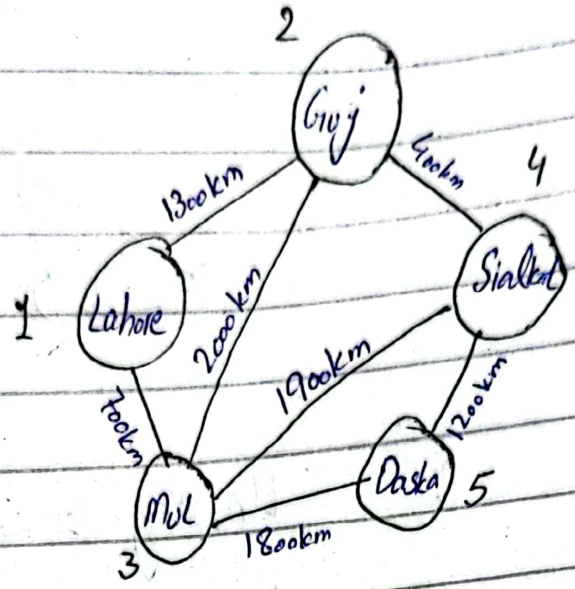


Un-Weighted Graph
It is a graph in which all are considered to have same weight.

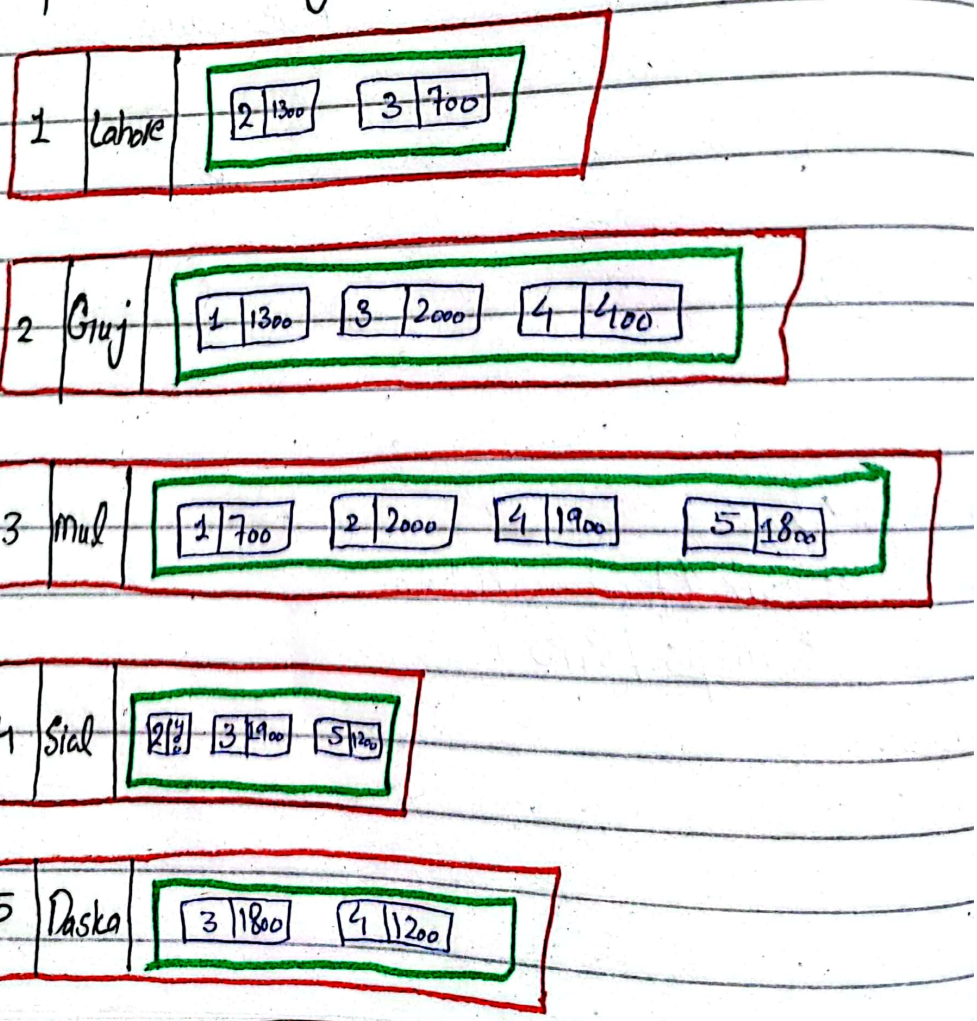


Examples
Social network (facebook, linkedin
Maps (G.Maps / GPS)
WWW

{Implementation}

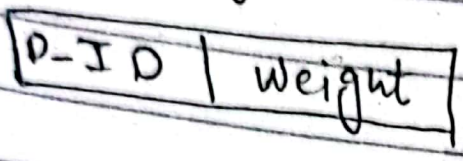


Graph Object :-



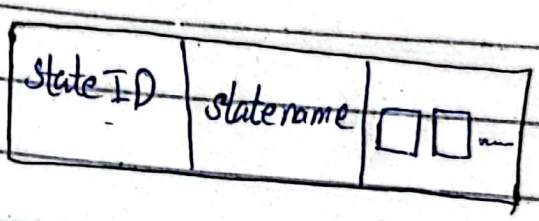
Edge Class:-

Dest_ID
weight;



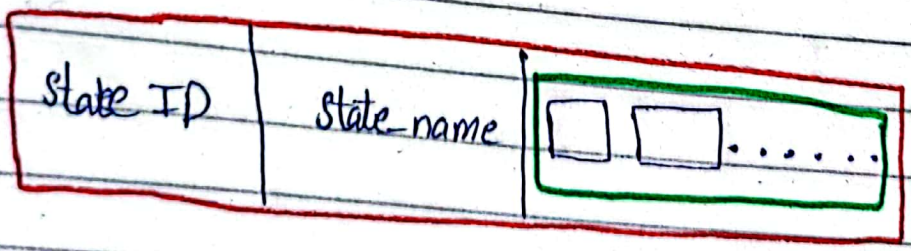
Vertex class:-

Stat_ID
state_name
[edge] edge list

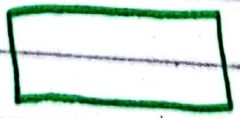


Graph class:-

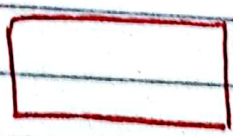
[Vertex] veitices



Edge object



Edge list



Vertex object