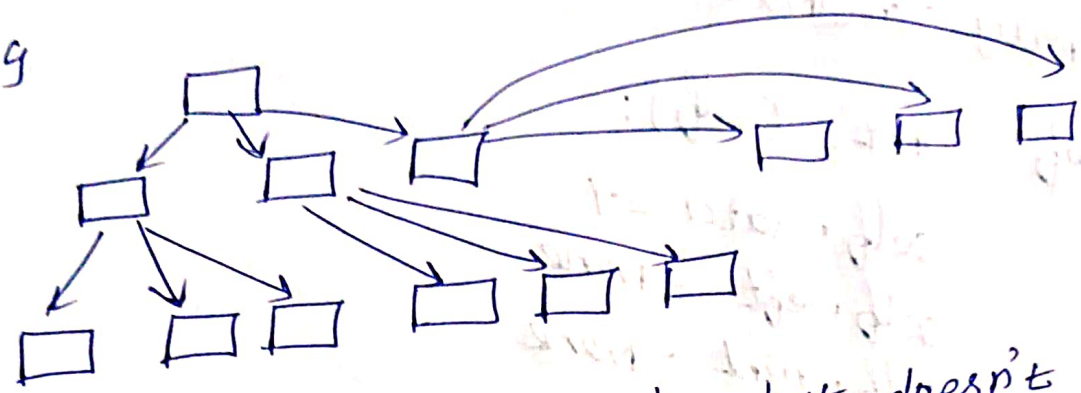
Trees

⇒ Tree is a data structure similar to Linked list instead of Linearly nodes connection, Trees have a parent child relationship.

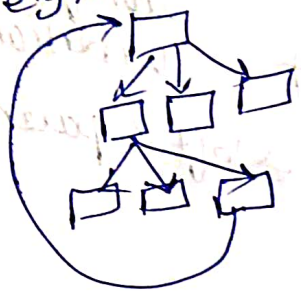
⇒ E.g



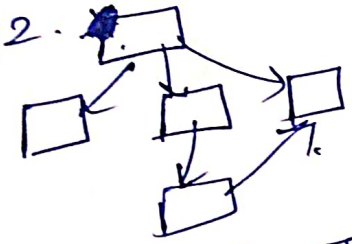
⇒ It is similar to graph but doesn't have,

* cycles

eg1.



eg2.



take can represent Tree's node using, Linked list (or) doubly linked list

E.g :-

Linked List

```
class TreeNode1 :
    def __init__(self):
        self.data = -1
        self.child1 = None
        self.child2 = None
        self.child3 = None
```

doubly Linked list

```
class TreeNode2:
    def __init__(self):
        self.data = -1
        self.child1 = None
        self.child2 = None
        self.child3 = None
        self.parent = None
```

Binary Tree

⇒ A Tree which has max of 2 children.

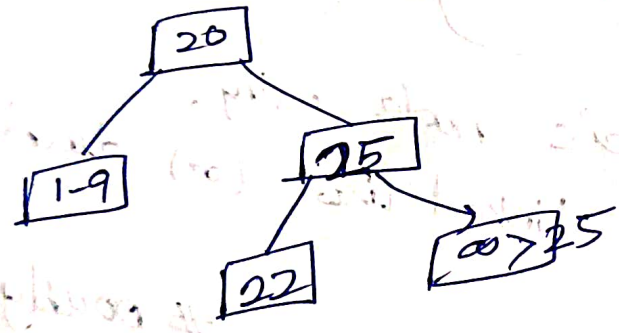
```
class BinaryTreeNode:
    def __init__(self):
        self.data = -1
        self.left = None
        self.right = None
```

Binary Search Tree:

⇒ A binary tree + satisfy condition.

⇒ left < parent ⇒ right > parent.

e.g

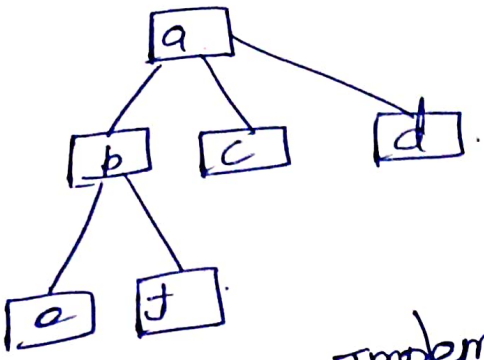


Tree - program:-

```
def main():
    node1 = TreeNode1()
    node1.data = 'a'
    node1child1 = TreeNode1()
    node1child1.data = 'b'
    node1child2 = TreeNode1()
    node1child2.data = 'c'
```

node1.child3 = TreeNode1()

node1.child3.data = 'd'



Level order Traversal (Breadth)

abcdef.

Depth ~~Breadth~~ First Traversal.

abefcd.

Implement using queue.

```
queue = queue()

queue.push(node)

while not queue.empty():
    node = queue.pop()
    if node is None:
        continue
    print(node.data)
    queue.push(node.child1)
    queue.push(node.child2)
    queue.push(node.child3)
```