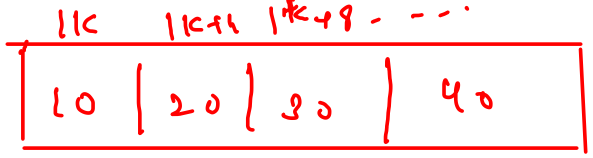
Data Structure

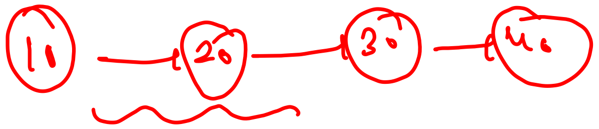
1. Arrays.
2. ArrayList
3. Stack
4. Queue
5. LinkedList

Continuous Memory.

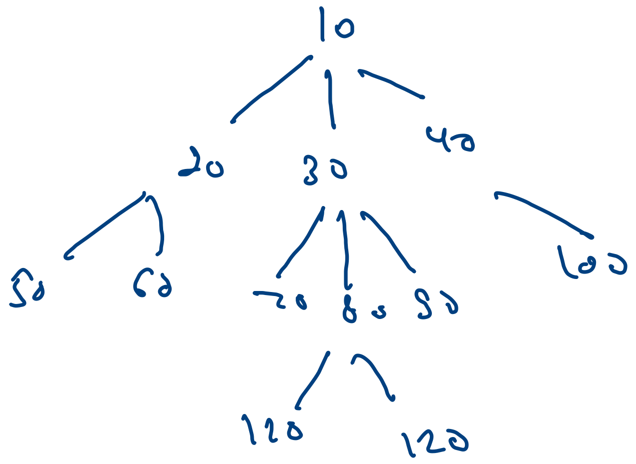
Discontinuous memory.

linear data Structure,





Linear D.S.



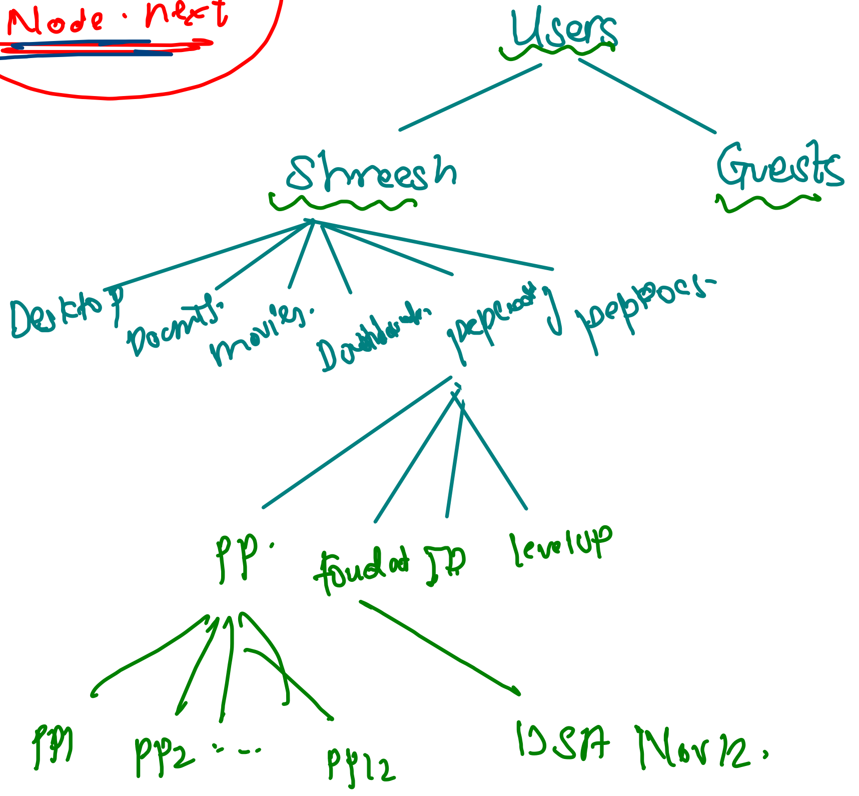
How we can store this??

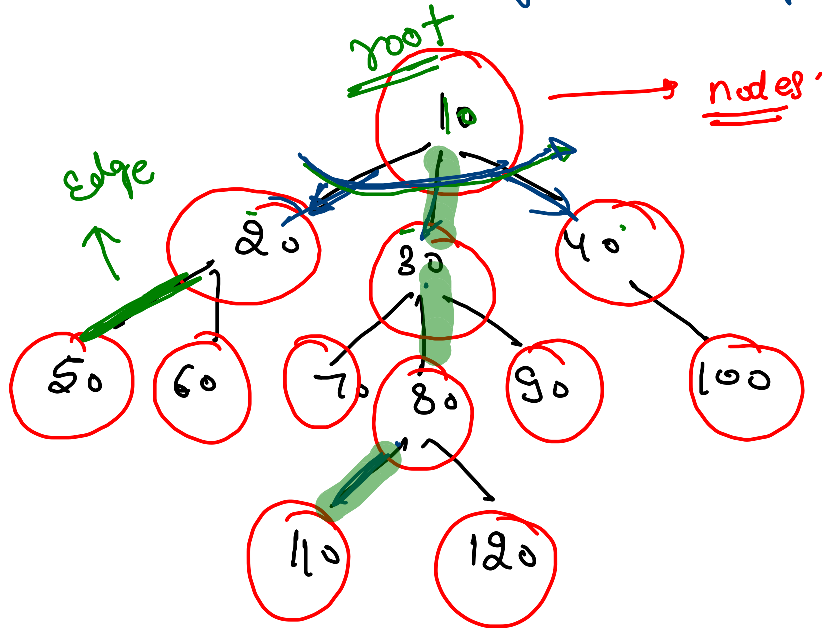
File System

For simplicity, we store Integer Data only.

int data
Node next

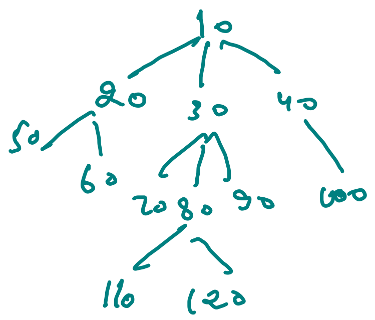
Node of Linkedlist

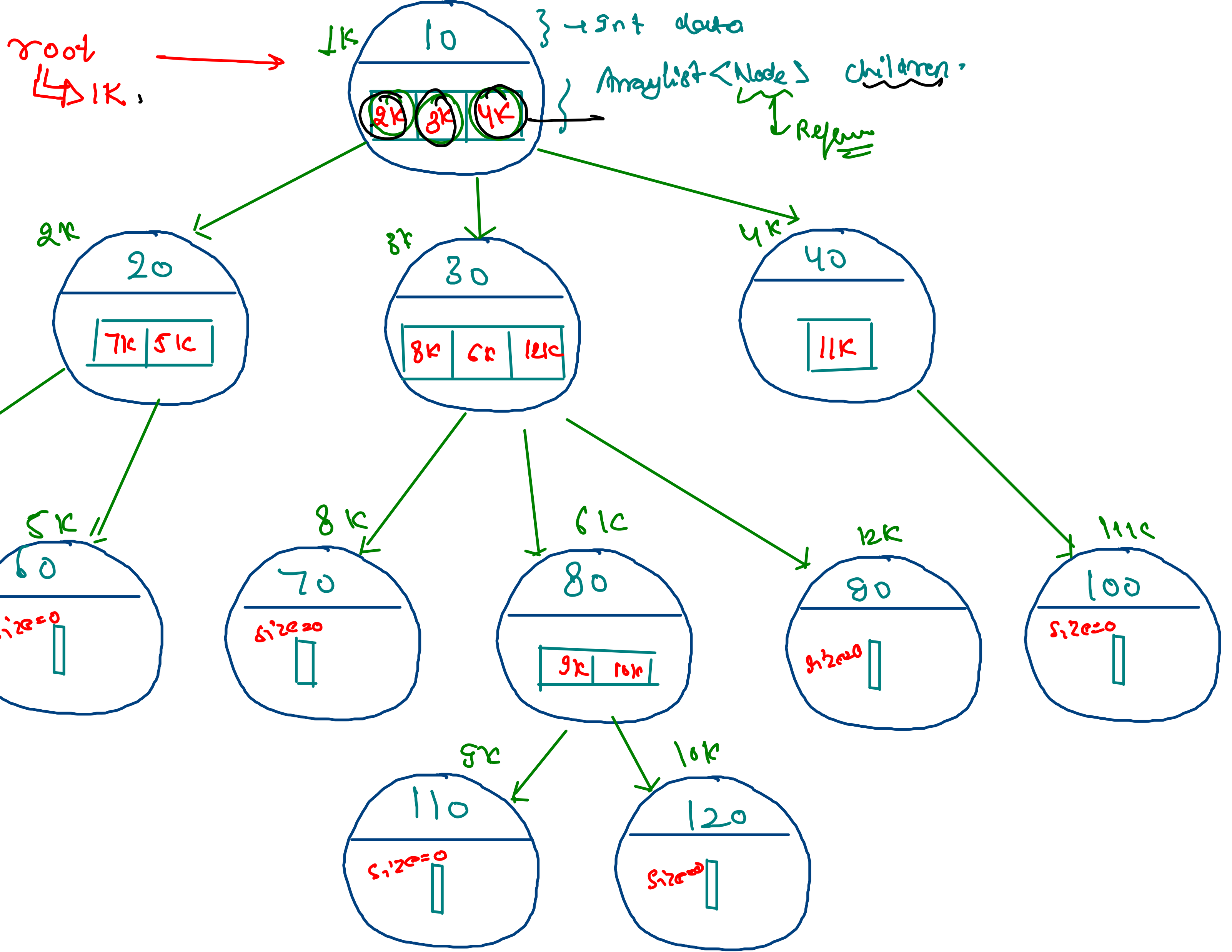




root - have no parent
Child - Nodes below current Node
Edge
Node
parent
Ancestor

Components in Node:
int data
ArrayList<Node> children


10
20
30
40
50
60
70
80
90
100
110
120

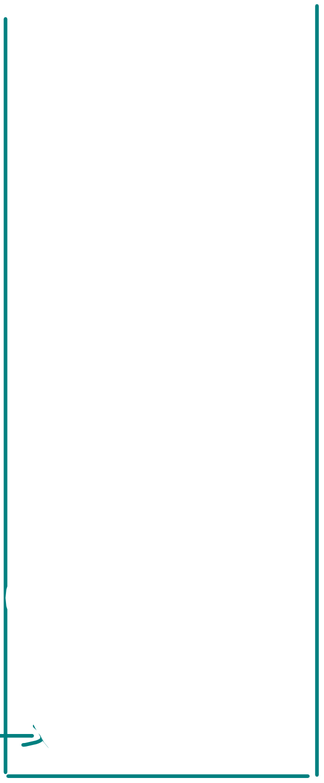

root
1K.
1K
10
int data
2K 3K 4K
Arraylist<Node> children.
Reference
2K
20
7K 5K
3K
30
8K 6K 12K
4K
40
11K
7K
50
size=0
5K
60
size=0
8K
70
size=0
6K
80
9K 10K
12K
90
size=0
11K
100
size=0
9K
110
size=0
10K
120
size=0

construction·   Euler traver sal →

root = null

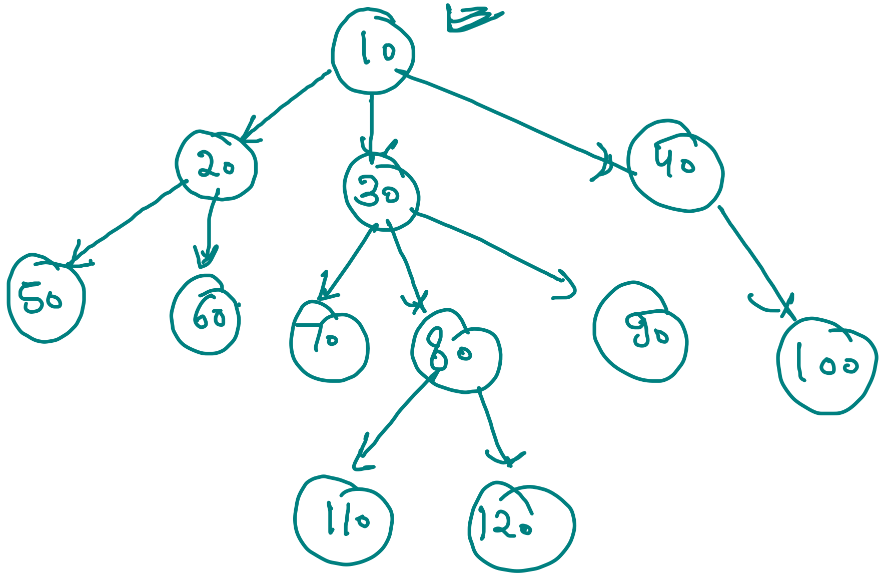
data → Integer

i 10	→ 120
i → 20	→ null
i → 50	→ null
i → null	→ 90
→ 60	→ null
→ null	→ null
→ null	→ 40
→ 30	→ 100
→ 70	→ null
→ null	→ null
→ 80	→ null
→ 110	
→ null	



Stac.

→ Recursive



```
if( data != null) {
    Node nn = new Node(d);
    st.peek().children.add(nn);
    st.push(nn);
} else{

    st.pop();

}
```

Display

10 → 20, 30, 40 ..

20

20 → 50, 60.

50 → .

60 → .

80

30 → 70, 80, 90.

70 → .

80 → 110, 120

110 → .

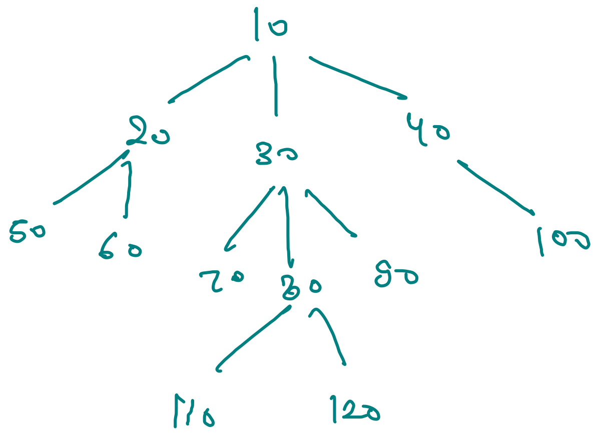
120 → .

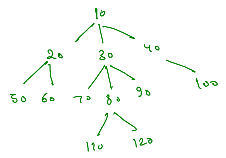
90 → .

40

40 → 100

100 → .





Expectation → display(10)

faith → display(10.children)

Merging → self print

loop { display(10.children)

depend on no. of children

Work for you :

100% achieves

1. Why there is no base case, with proper Analysis.

2. flow of code. (Display)

Revise
1. Constructors
2. Flow code

Generic Tree
1. Binary Tree
2. BST
3. Graph.