Directory.

Generic Tree.

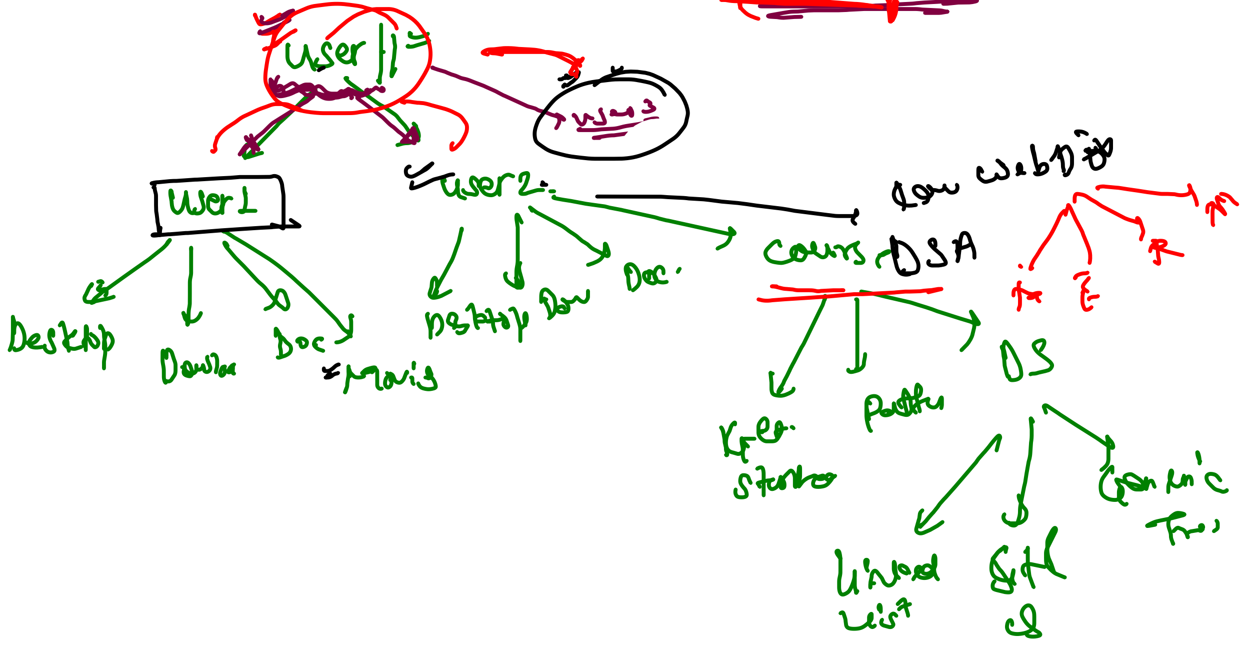
Data Structure.

linear
- (1) Arrays / ArrayList
- (2) Linked list
- (3) Stack & Queue

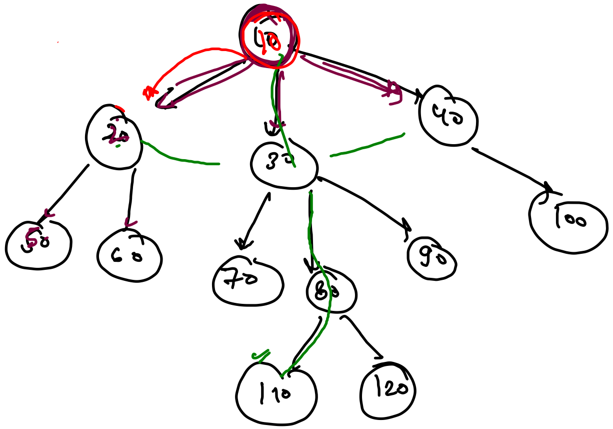
linear Info

Student. Roll . St. Name.

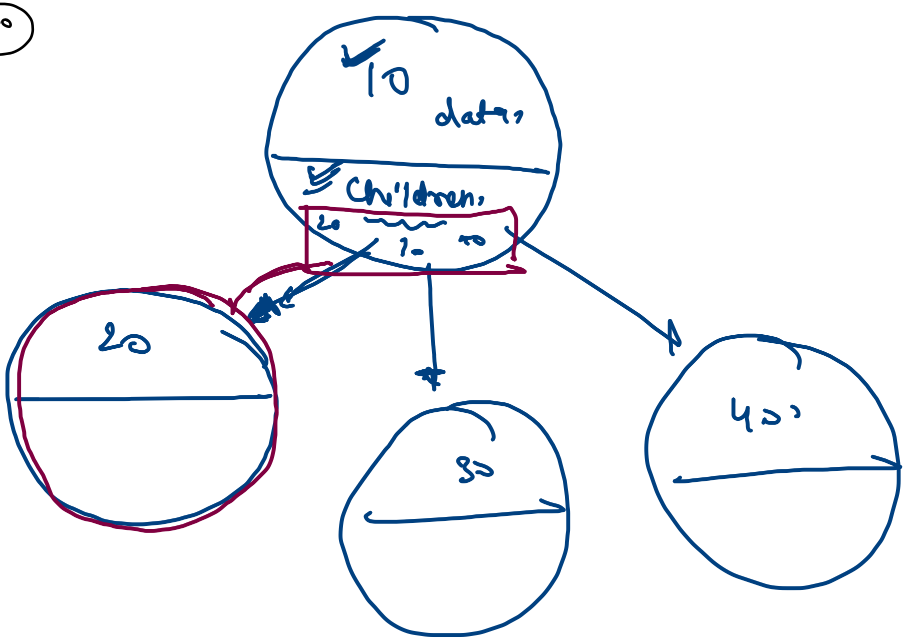
Euler tree.



Data Structure
- Data Set
- Data Retrieve
- Data Get



Node Next :



Info,

data,

children → ✓ generic { size is not }

✓ Arraylist <Node> fixed

ens'

Root → 10

children → next
level node.

parent → upper
level node,

Ancestor → upper level
till Root

Eg - 110 → Ancestors
80, 30, 10

✓ Siblings →

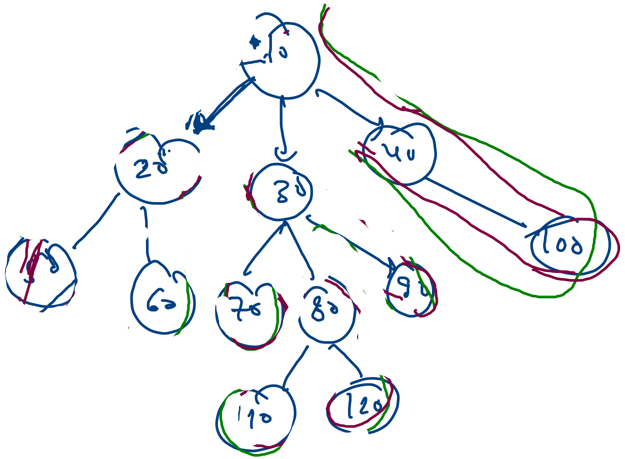
Recursion — Revise.

95%. Recur.

1. Generic Tree.
2. Binary Tree.
3. B.S.T.
4. AVL

5. Graph.

6. DP. → 80%
contribution



data

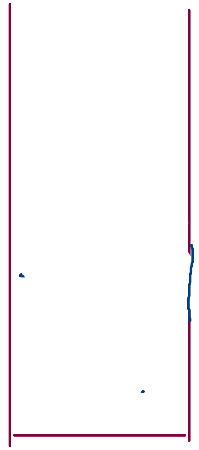
✓
✓
✓50
✓-1
✓60
✓-1
✓-1
✓30
✓70
✓-1
✓80
✓110

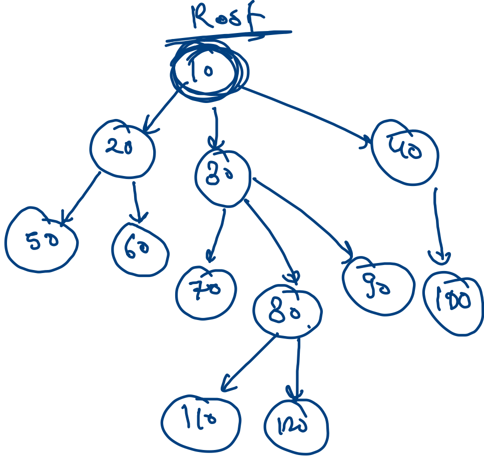
pre → data
post → -1

✓-1
✓120
✓-1
✓-1
✓90
✓-1
✓-1
✓40
✓100
✓-1
✓-1
✓-1

Data Structure.

Stack





Display. →

gnfo (Tree) Nodes

[10] → 20, 30, 40

[20] → 50, 60

[50] → -

[60] → -

[30] → 70, 80, 90

[70] → -

[80] → 110, 120

[110] → -

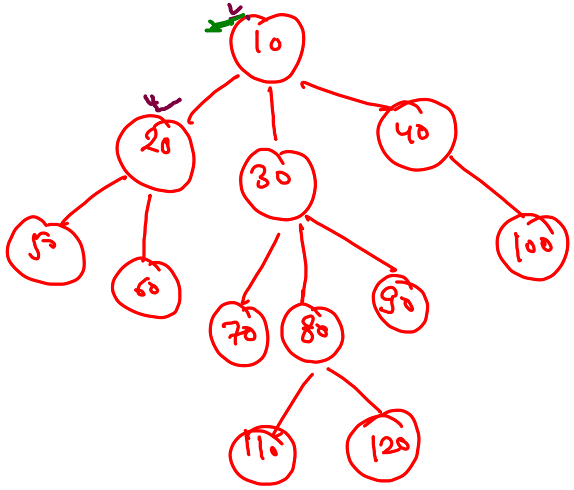
[120] → -

[90] → -

[40] → 100

[100] → -

(10) → [20] [30] (40)]



Recursion

Faith & Expectation

Expectation → Display (Root) ==

Faith → Display [Root . children] have potential to print the info

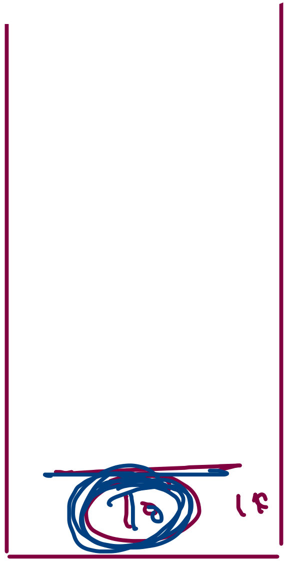
Root . children . data

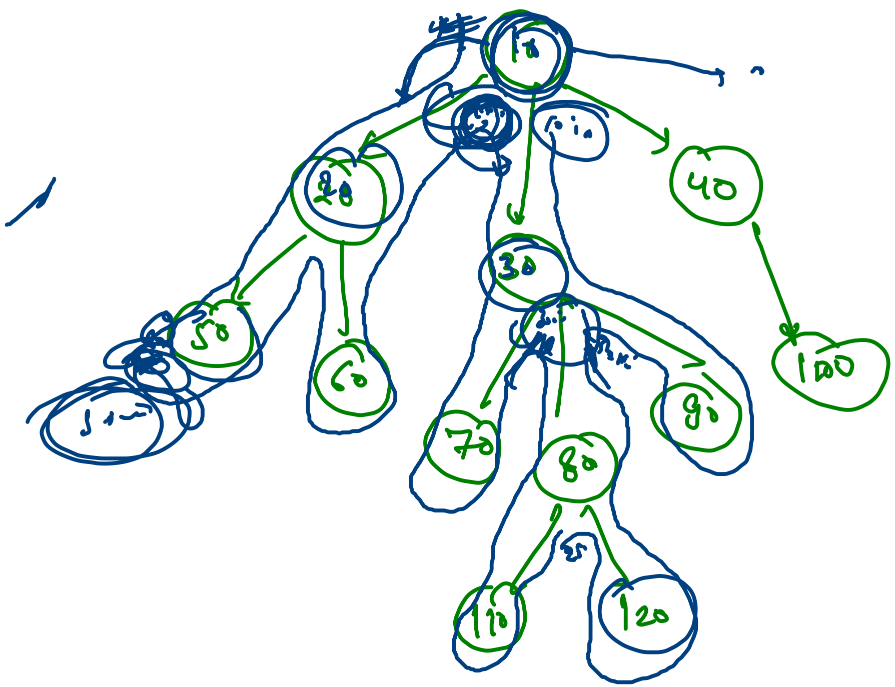
Merging → Sysout [Root.data] → [20 , 30 , 40 ...]

// root = nn;

10
20
50
-1
60
-1
-1
30
70
-1
80
110
-1
120
-1

-1
90
-1
-1
40
100
-1
-1
-1





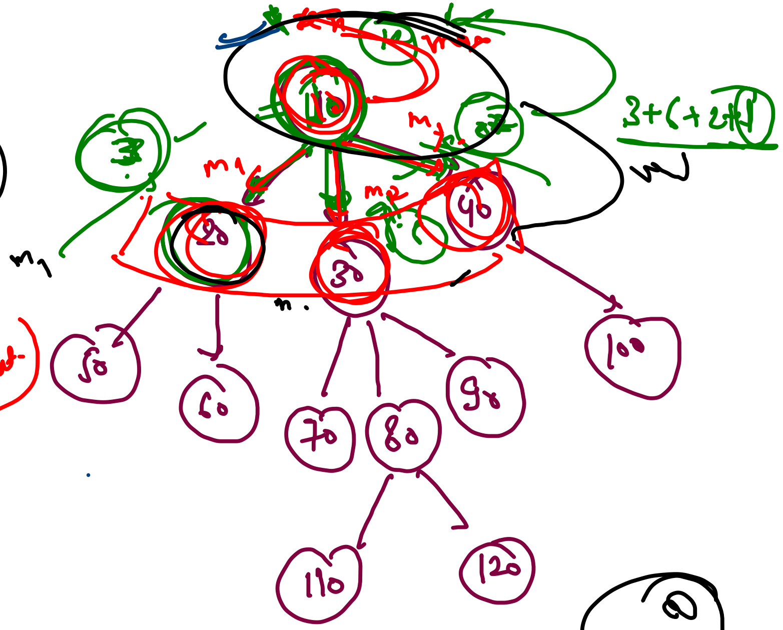
```
data == -1;
st.pop();
else {
    Node nn = new Node(data);
    st.peek().children.add(nn);
    st.push(nn);
}
```

Size → Expectation → size(Root) → 12 =

faith (Root.children[i]) → size I

Merging

$-\infty$



Max(m1, m2, m3, root.data)

Max.

mx = Integer.MIN_VALUE;   // Identity

$a * b = a.$

operator.

b is identity for operator *.

$a \times 1 = a.$

↳ 1 is identity mult

$a + 0 = a$

$a - 0 = a$

or

max(a, b)

$[a \max b] = a$

$b = -\infty$