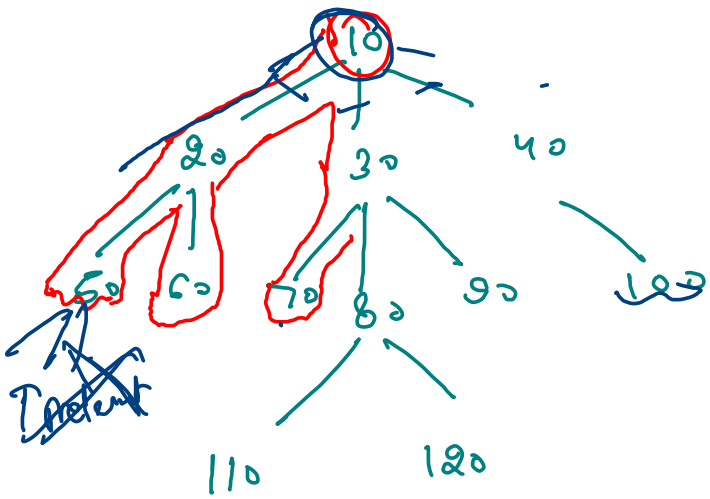
① Why base case is not needed

+

code flow



base case.

10 → 20 30 40

20 → 50, 60

50 → .

60 → .

30 → 70 80 90

70 → .

.
.
.

100 → .

calls → small call

node = null

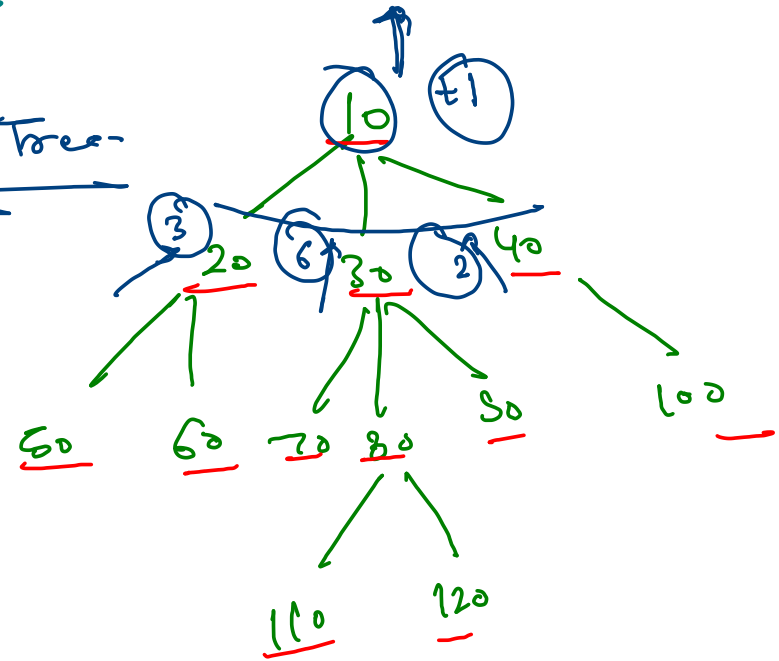
```
public static void display(Node node) {
    // self work
    System.out.print(node.data + " -> ");
    for(Node child : node.children) {
        System.out.print(child.data + " ");
    }
    System.out.println(".");

    // faith work
    for(Node child : node.children) {
        display(child);
    }
}
```

small call + stupid base case

Size of generic Tree.

Size = 12.

maximum in Generic Tree.



Expectation →

size(10) → 12

faith →

size(20) → 3

size(30) → 6

size(40) → 2.

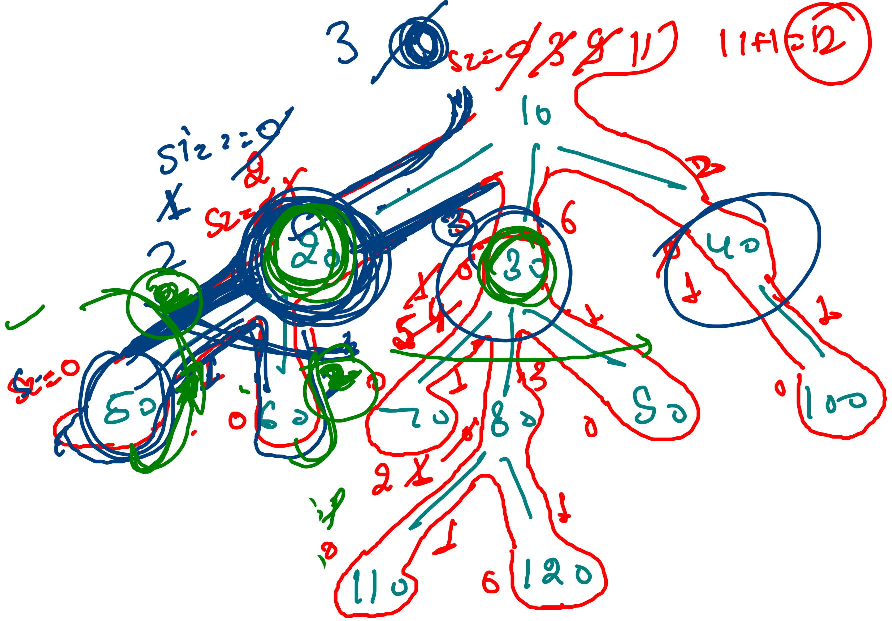
Merging → size = 0

size += size(20)

size += size(30)

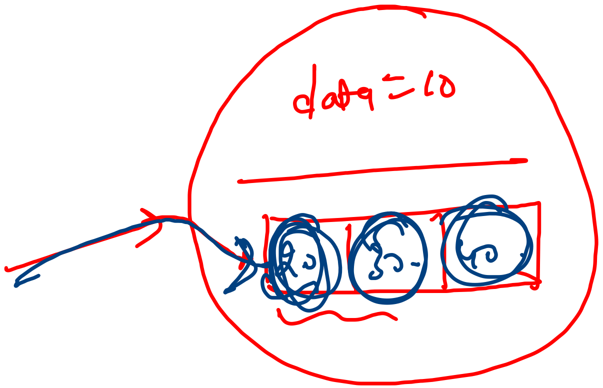
size += size(40)

return size + 1 (mysize)




```java
public static int size(Node node){
    if(node == null) return 0;
    int sz = 0;
    for(int i = 0; i < node.children.size(); i++) {
        Node child = node.children.get(i);
        sz += size(child);
    }
    return sz + 1;
}
```

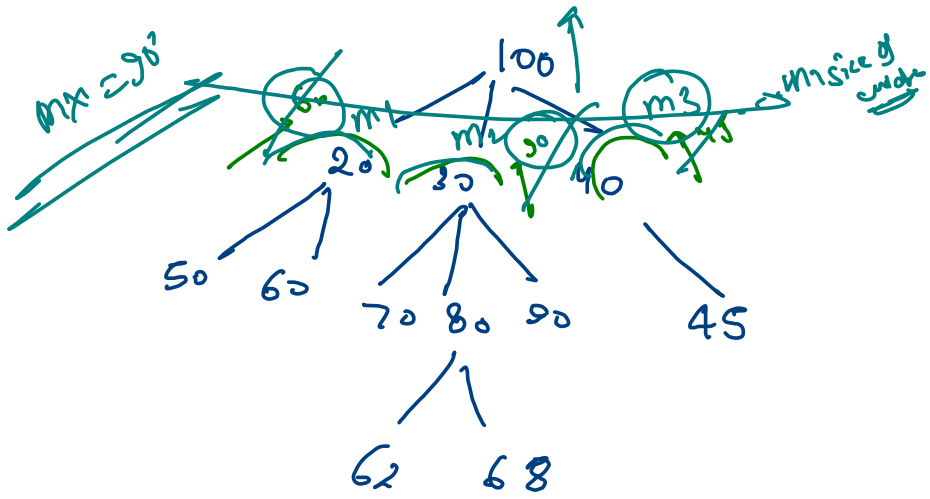
12 is size of overall tree.



3

for(Node child : node.children){
    child
}

Max.



Expectation ⇒ max(100) → 100 (max overall)

faith → max(20) → 60

max(30) → 90

max(40) → 45

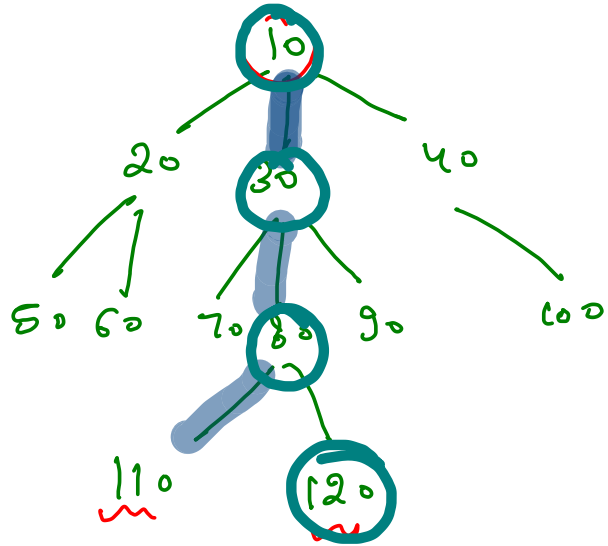
Logic: [Get max from faith]

final res = (m1 vs. m2 vs m3 .... m_size) vs. node.data

return res;

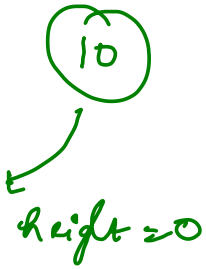
90 vs. 100

Height - Distance of farthest node from root is known as Height



Edge basis →

height = 3



Node basis →

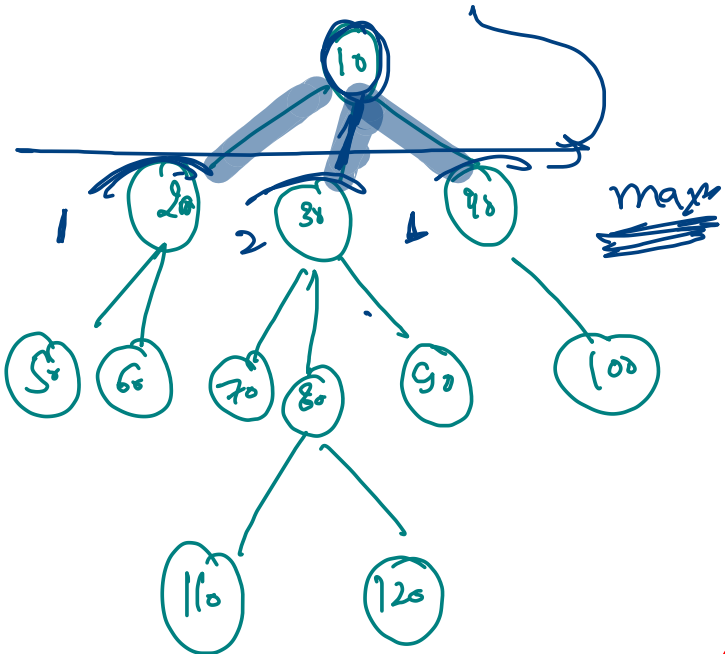
Edge = 1

Node basis

Height → 4

10

Height = 1

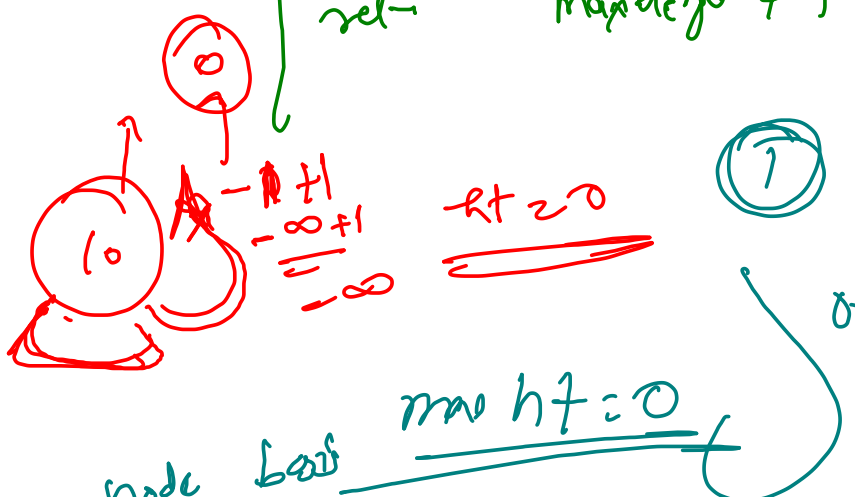


height — farthest distance b/w root to leaf.

Expectation → height(10) = 3

faith → height(20) = 1
height(30) = 2
height(40) = 1
max

max(h1, h2) + 1

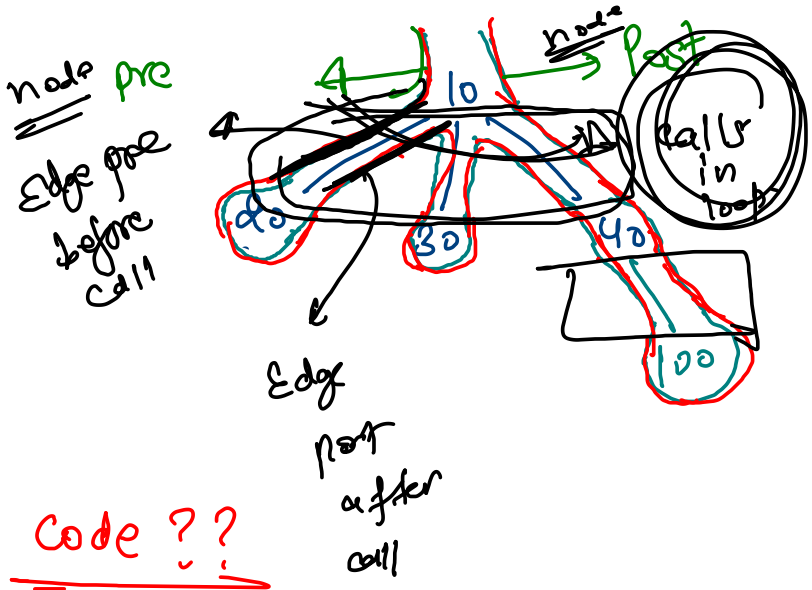


Initialization

mx = -∞

edges ∑ mx = -1

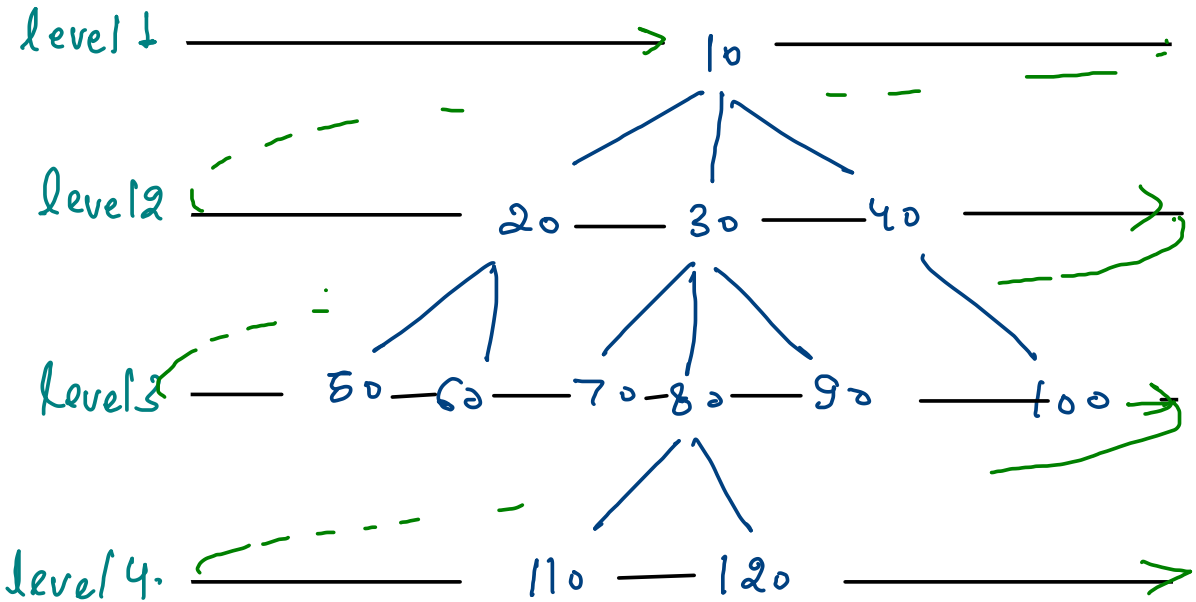
Traversals (PreOrder, PostOrder)



Code ??

Euler traversal

- node pre 10
- Edge pre 10--20
- node pre 20
- node post 20
- Edge post 10--20
- Edge pre 10--30
- node pre 30
- node post 30
- Edge post 10--30
- Edge pre 10--40
- node pre 40
- Edge pre 40--100
- Node pre 100
- Node post 100
- Edge post 40--100
- node post 40
- Edge Post 10--40
- Node post 10



level order.

o/p →

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

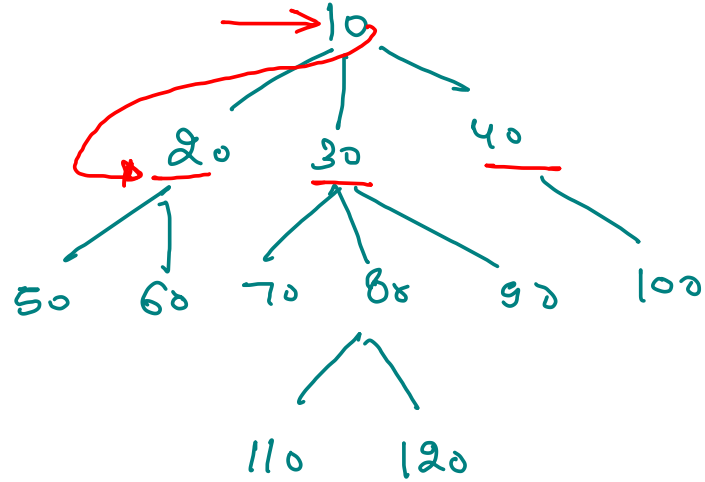
Recursion → Depth-first
Stack.

Queue → Breadth First

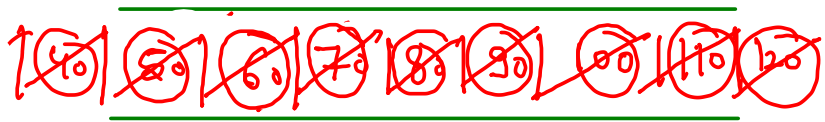
Queue → FIFO

first in first out

provide priority to siblings rather than children.



Recursion
Recursive
stack



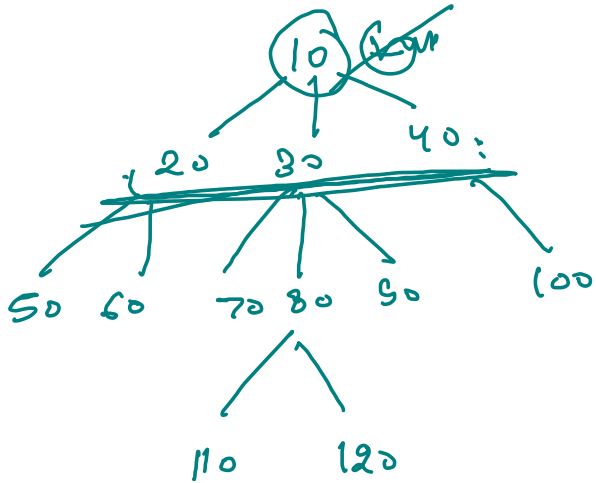
Step for algorithm

while(q.size() > 0)

1. get + rem.
2. print
3. Add children.

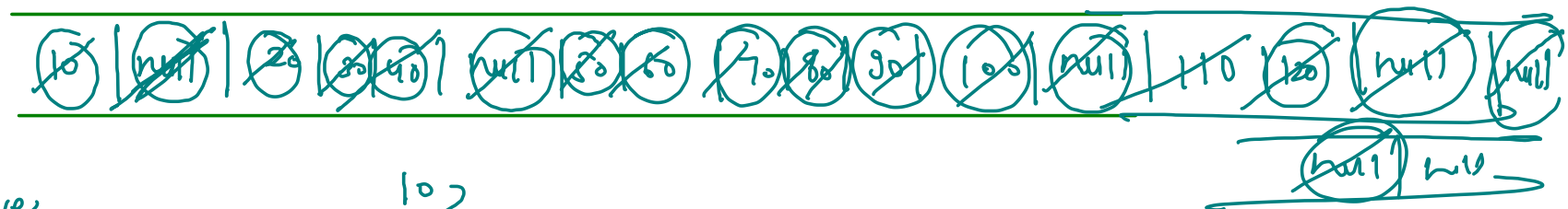
10
20
30
40
50
60
70
80
90
100
110
120

level order line wise:



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

Delimiter → null



(1) get + remove
(2) print
(3) add children

if null → Enter/hit if → push(null) if q.size() > 0

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