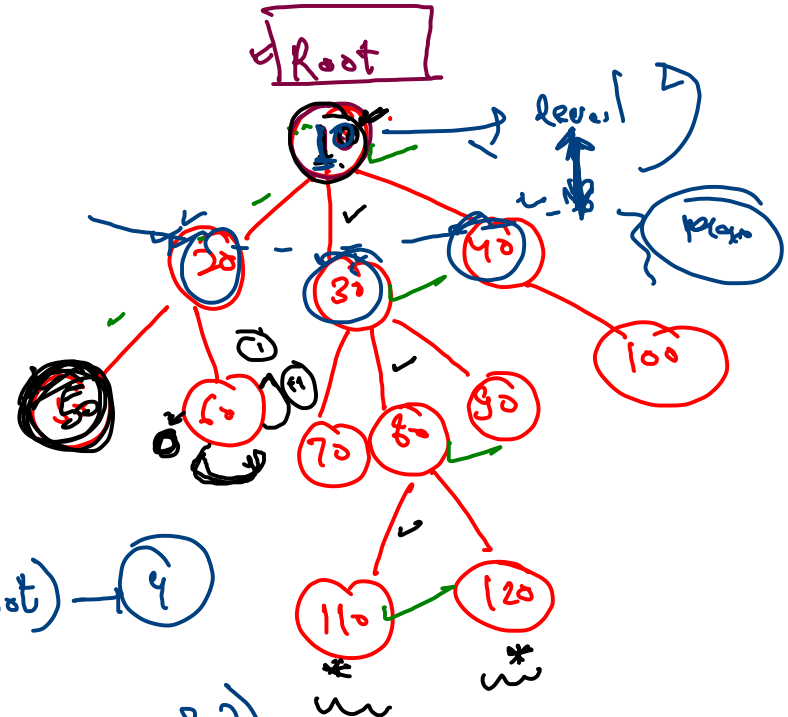
Height of GenericTree -

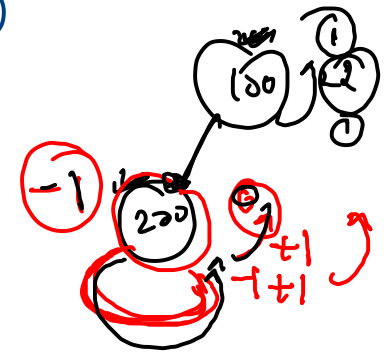
distance of deepest Node from the root.

Distance

- on the basis of Edge:
  - Edge → 3
  - Node → 4
- on the basis of Node:
  - Implementation
  - Expectation → ht(Root) → 4



* Introduction
1. Construction
2. Display.
3. Size, Max, Min



on the basis of Edge:
ht(20) = 0

Node → 4

on the basis of Node -

ht(50) = 1

Edge

ht(20) → 1
ht(30) → 2
ht(40) → 1

(1 vs 2 vs 1) + 1 = 2 + 1 = 3

faith → ht(Root.children[i]).

↳ ht(20) → 2
ht(30) → 2
ht(40) → 2

max of these heights + 1;

(2 vs 3 vs 2) + 1 = 3 + 1 = 4

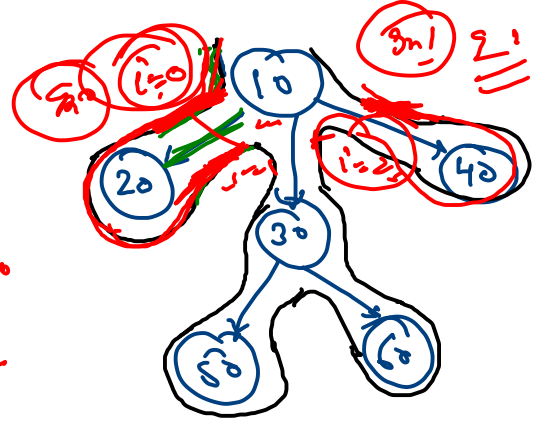
Node →

Edge →

(0+1)

(1+1)

- Node pre.
- Node post



- Edge pre 10 -- 20
- Edge post 10 -- 20

pre 10	pre 60
pre 20	post 60
post 20	post 30
→ gn1 10	→ gn2 → 10
pre 30	pre 40
pre 50	post 40
post 50	post 10
→ gn1 30	

- Node - post - 60
- Edge post - 30 -- 60
- Node post - 30
- Edge post 10 -- 30
- Edge pre 10 -- 40
- Node pre 40
- Node post 40
- Edge post - 10 -- 40
- Node post - 10

Euler path:

- Node pre 10
- Edge pre 10 -- 20
- Node pre 20
- Node post - 20
- Edge post 10 -- 20
- Edge pre 10 -- 30
- Node pre - 30
- Edge pre 30 -- 50
- Node pre - 50
- Node post - 50
- Edge post 30 -- 50
- Edge pre 30 -- 60
- Node pre - 60

level Order

D.S → Queue.

What?

Why?

DS → Queue.

- Get *
- Remove
- Mark *
- Work *
- Add Children.

1. What?
2. Why?
3. How?

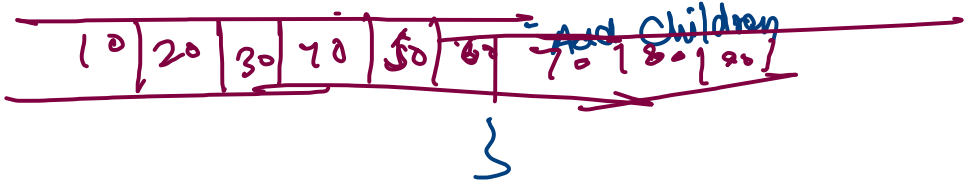
1. Get
2. Remove
3. Work → printing of data.
4. Add Children

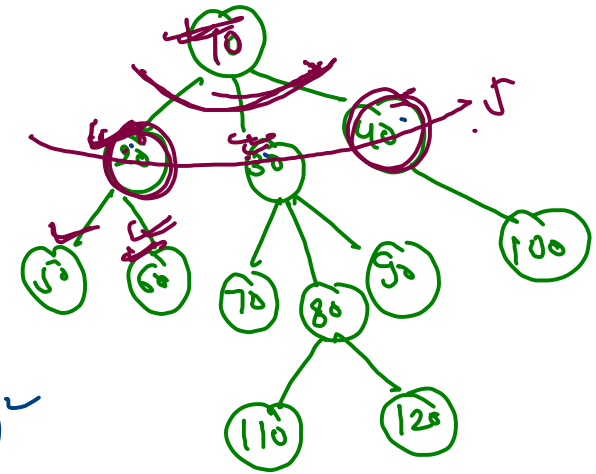
while(que.size() > 0)

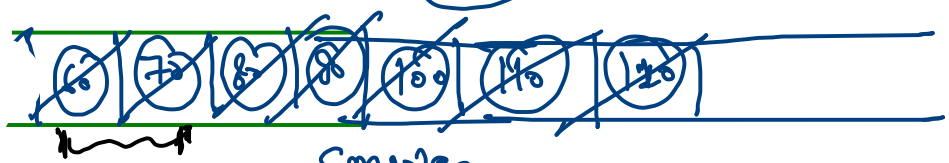
get
Remove
} Java Remove

Work

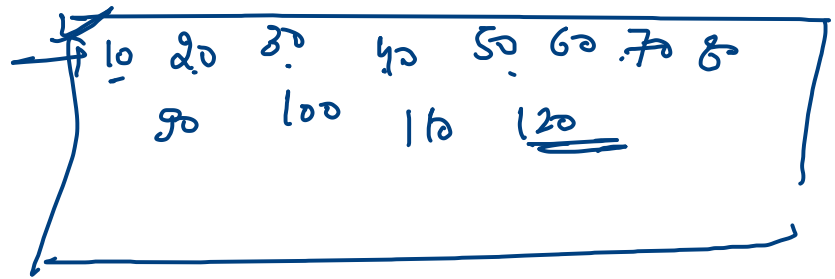
Add Children

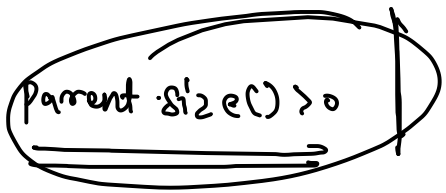


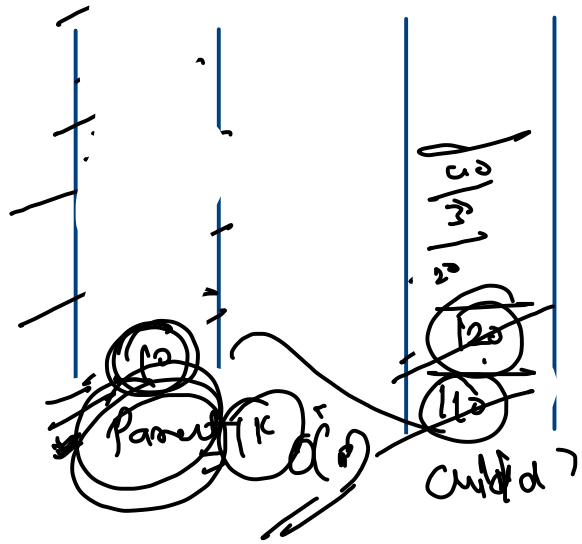


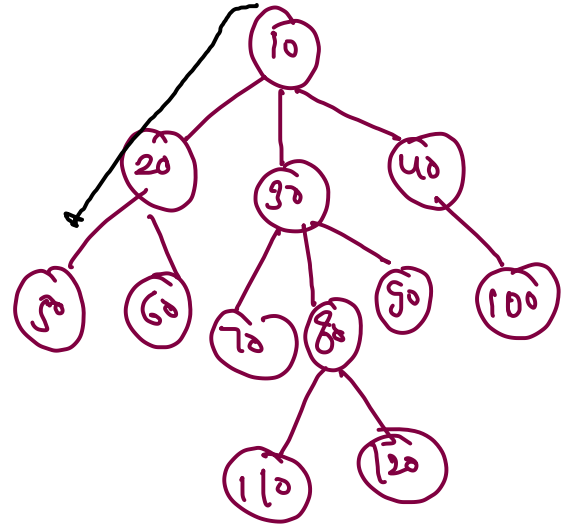


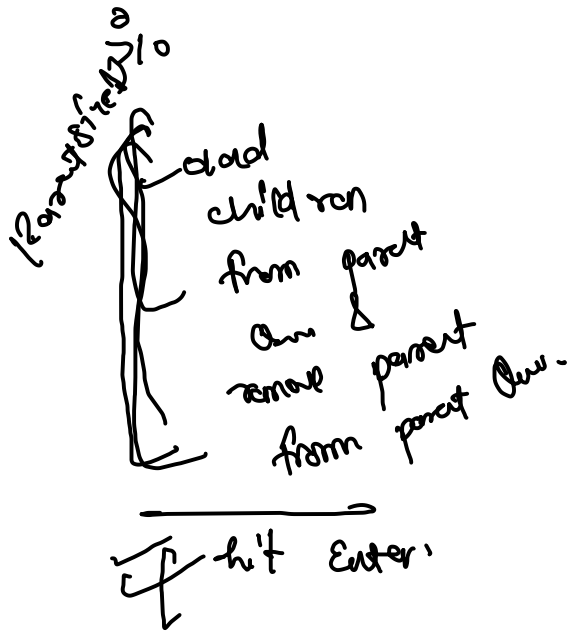
Console

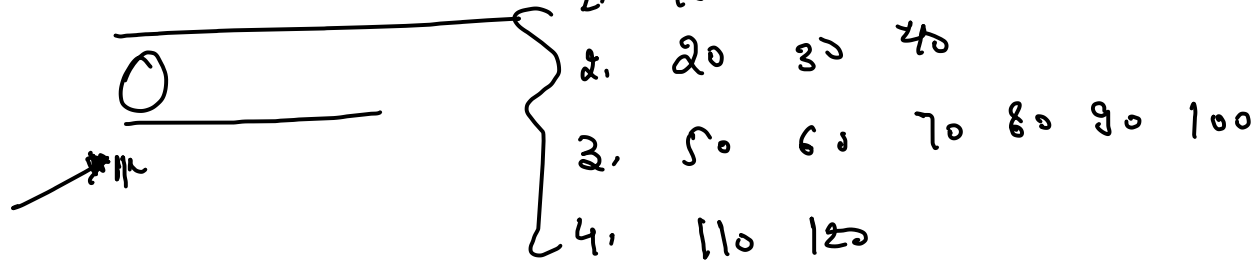












Interchange Ques. ⟶ parent → child

child → new ArrayDeque<>();

level order (zig zag)

10

40 30 20

50 60 70 80 90 100

120 110

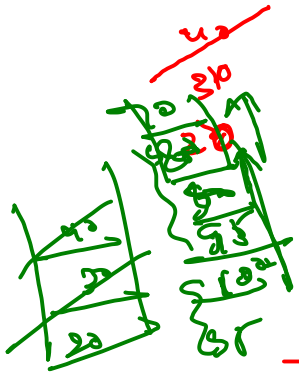
DS → Stack

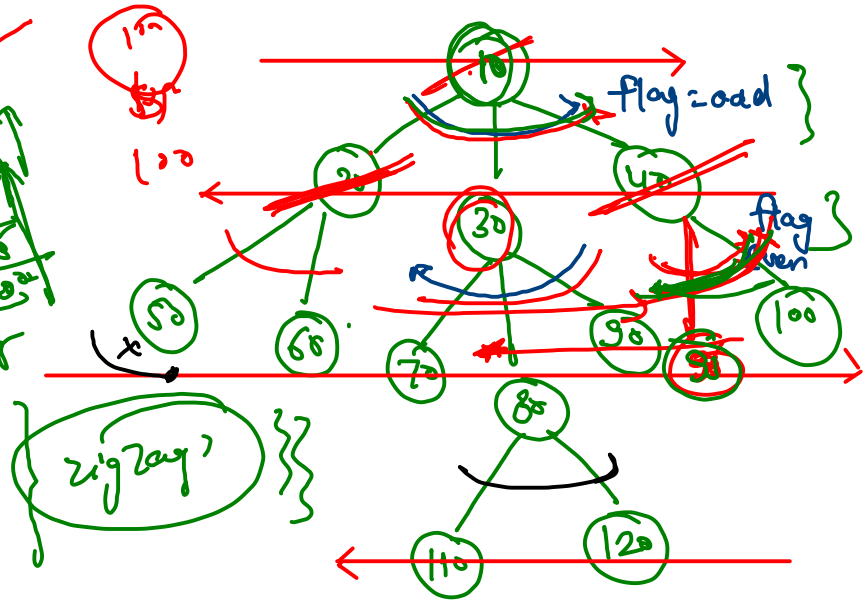
flag = ~~1~~ ~~2~~, ~~3~~, 4

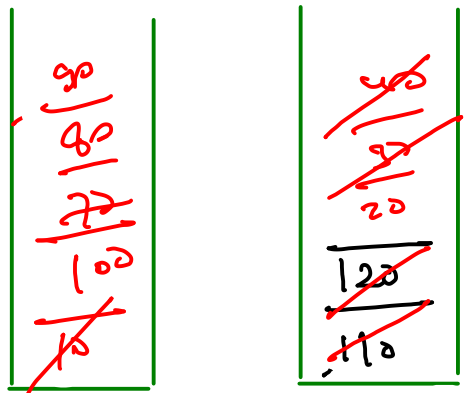
Manage loops for add Even of flag

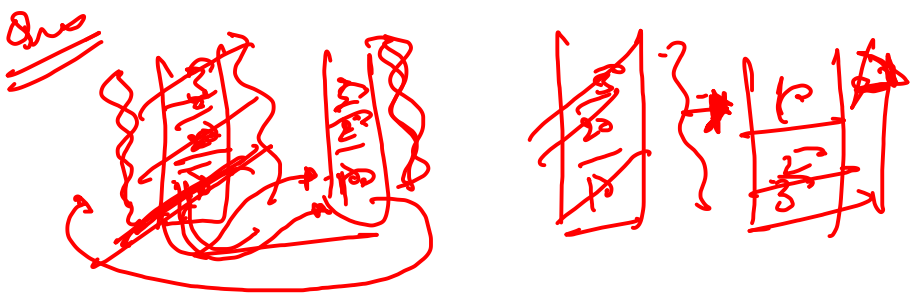
flag++;

(a → 20 → 30









1. 10
2. 40 30 20
3. 50 60 70 80 90 100
4. 120 110

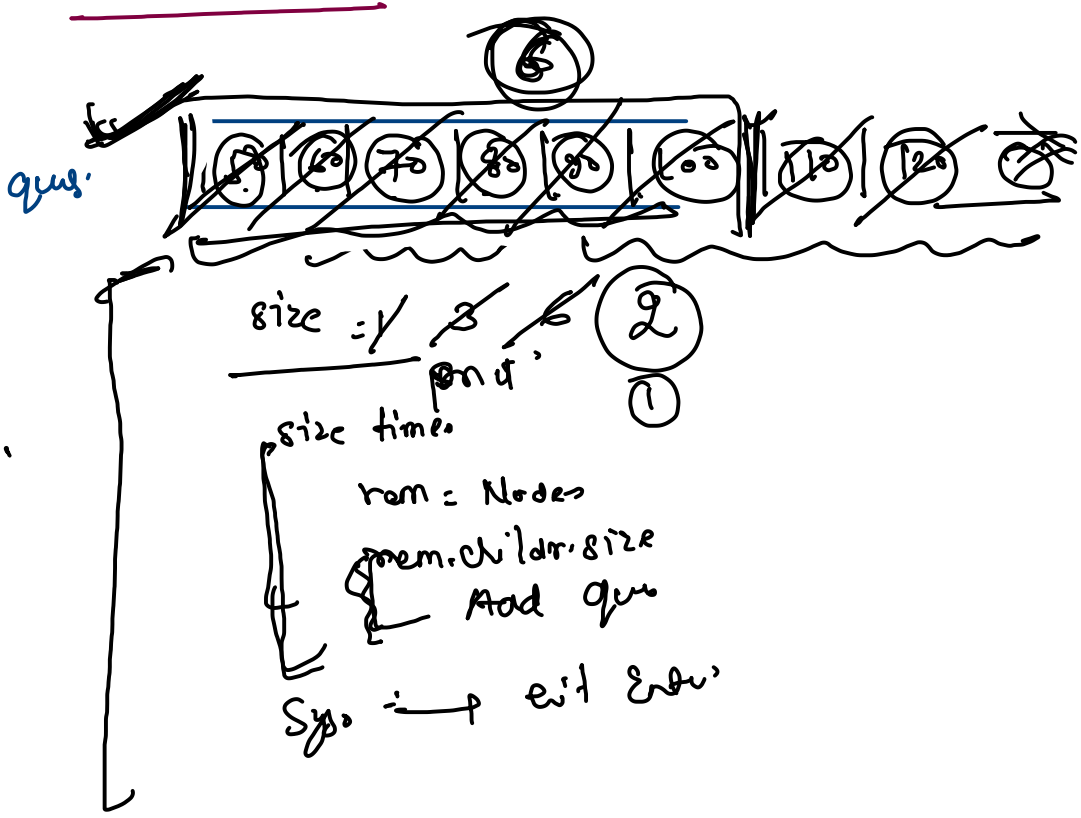
flag → odd
add child from left to right

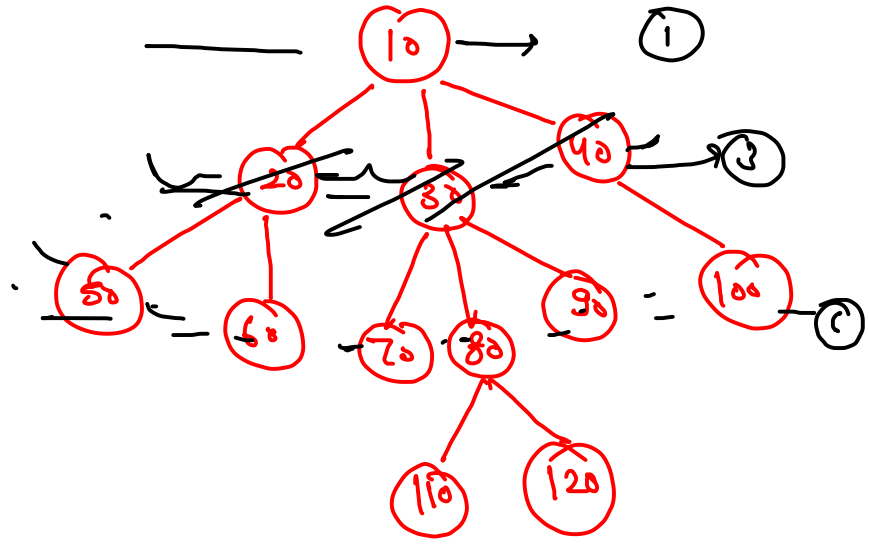
Even,
add child from right to left
→ reverse loop,

level Order →

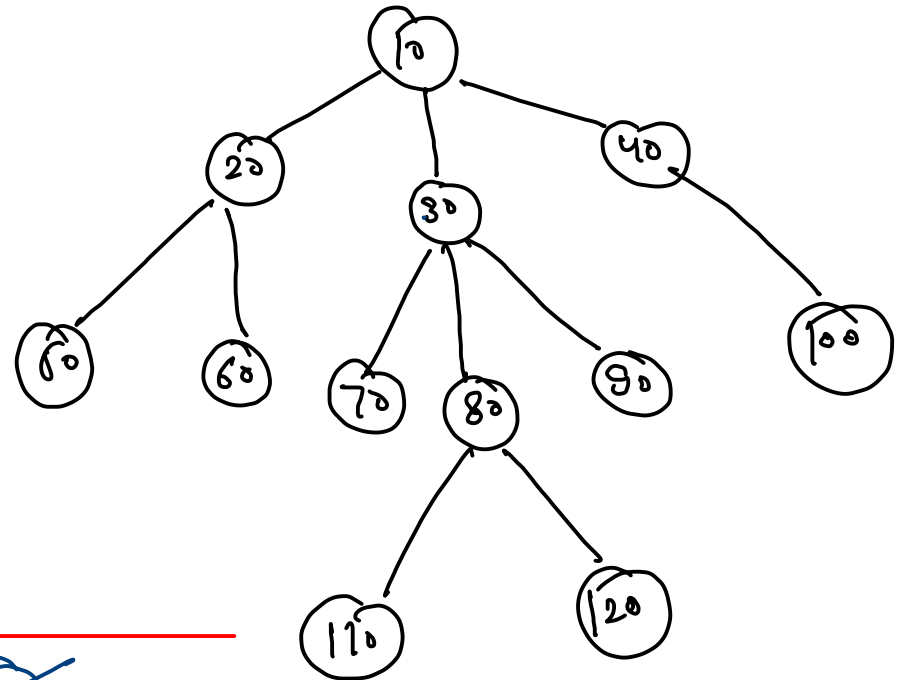
More approaches →

level Order line wise - (1)





```
qu.add(root);

while(qu.size() > 0) {
    int sz = qu.size();

    for(int i = 0; i < sz; i++) {
        Node rem = qu.remove();

        System.out.print(rem.data + " ");

        for(Node child : rem.children) {
            qu.add(child);
        }
    }

    System.out.println();
}
```



~~110~~ ~~120~~

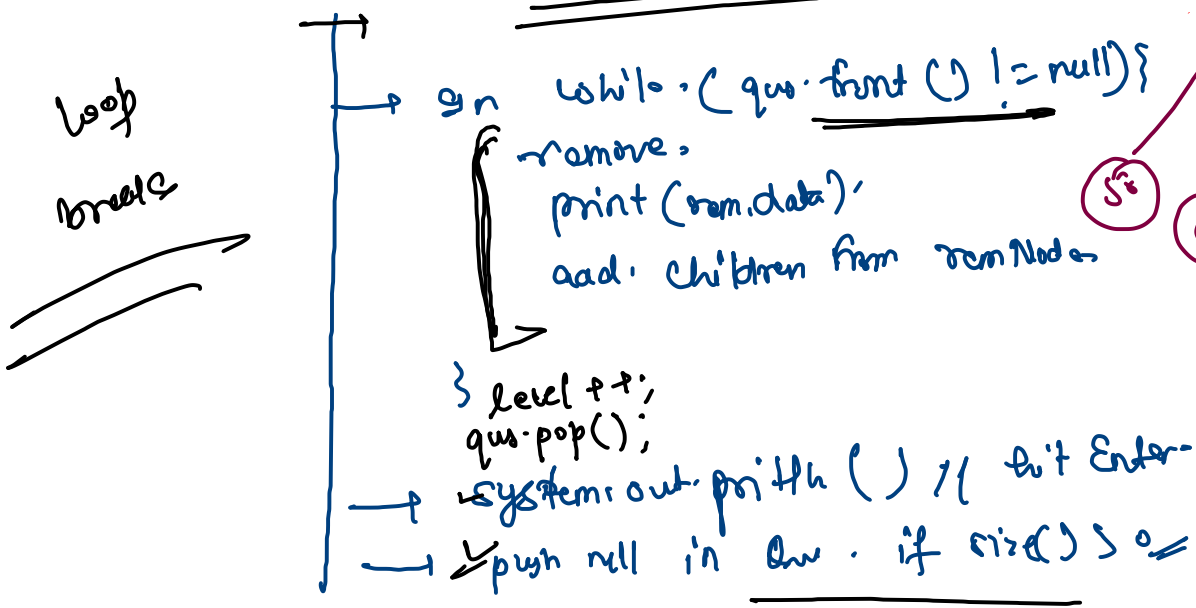
size = ~~1~~ ~~3~~ ~~6~~ 2.

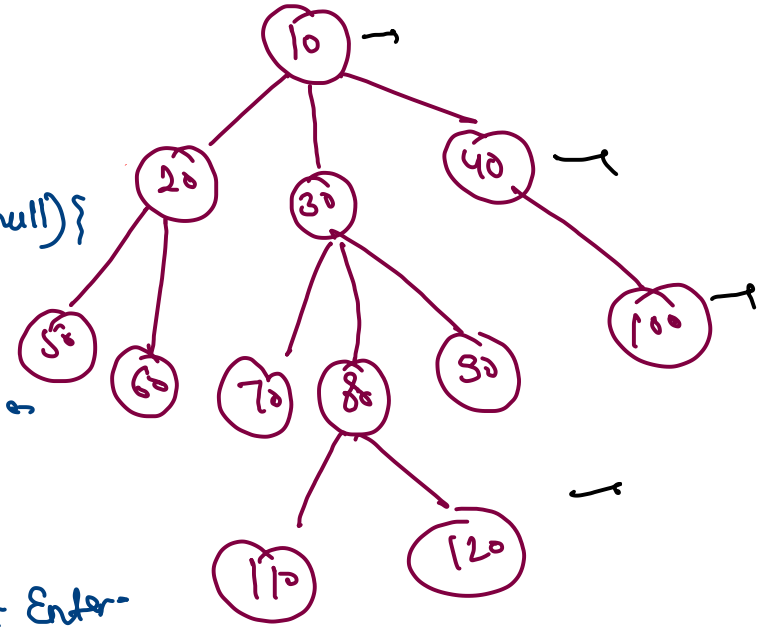
1. 10
2. 20 30 40
3. 50 60 70 80 90 100
4. 110 120
5.

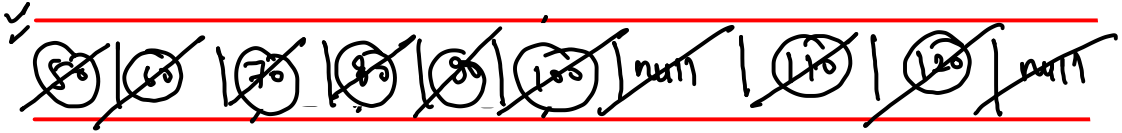
D-S → Queue.

Steps: ① root push + Null push.

② ↳ In while loop (qu.size() > 0) ✗







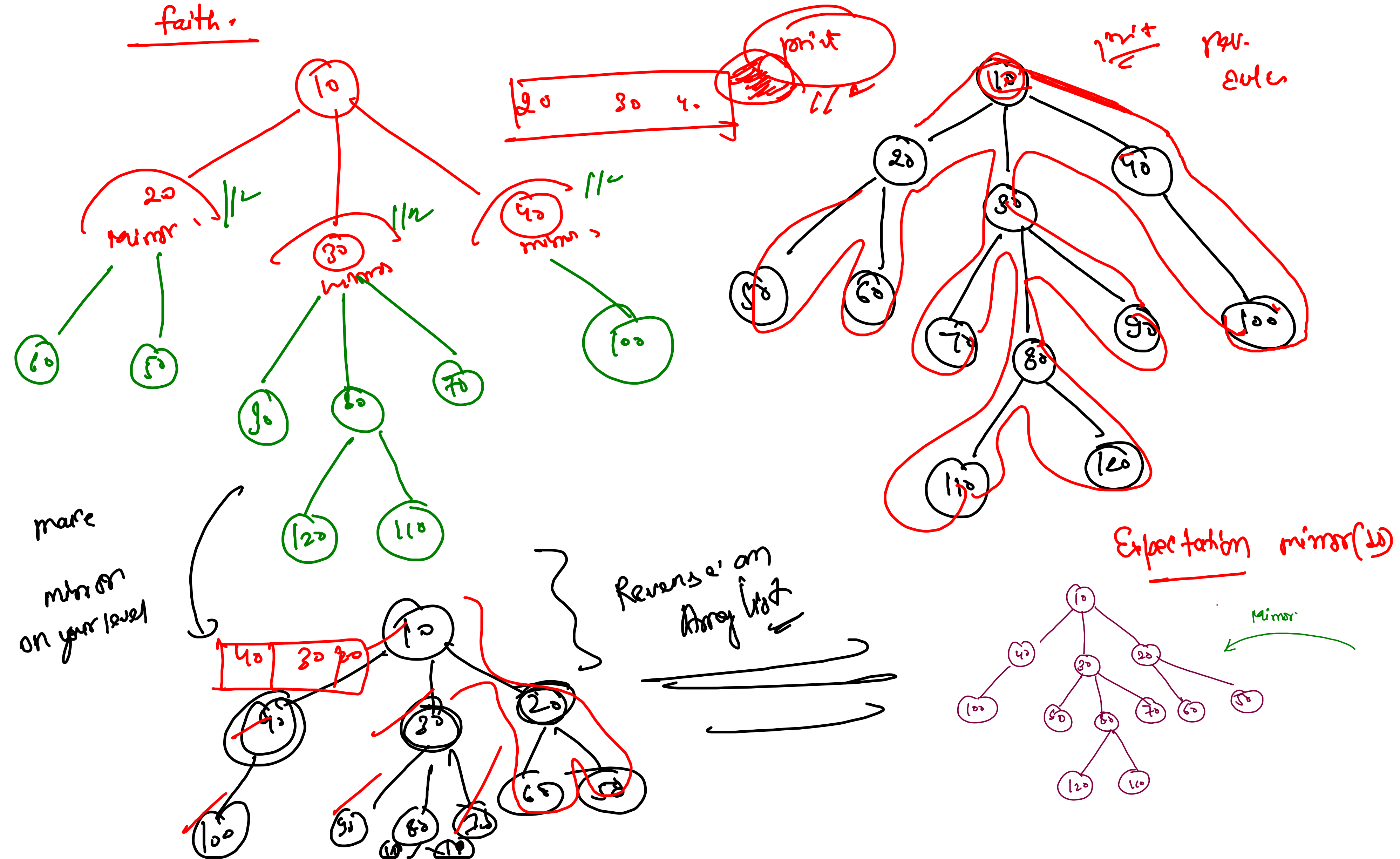
→ 1. 10

→ 2. 20 30 40

→ 3. 50 60 70 80 90 100

→ 4. 110 120

5.


faith
mirror
print
rev.
Euler
mare
mirror
on your level
Reverse on
Array List
Expectation
mirror(10)
Mirror

Find (88)

Expectation → find(10, 88) → True / False

faith → find(20, 88) || → T/F
find(30, 88) || → T/F
find(40, 88) || → T/F

Merging → check your self, dtf == node.data

```
if(dtf == node.data){
    return True;
}
```

Check it in your children.