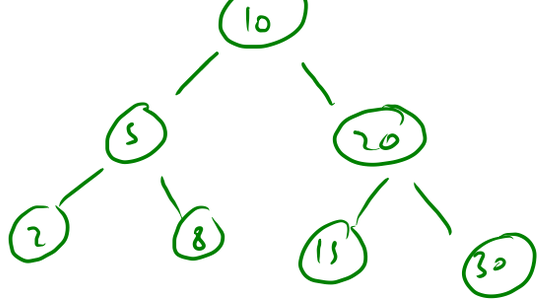
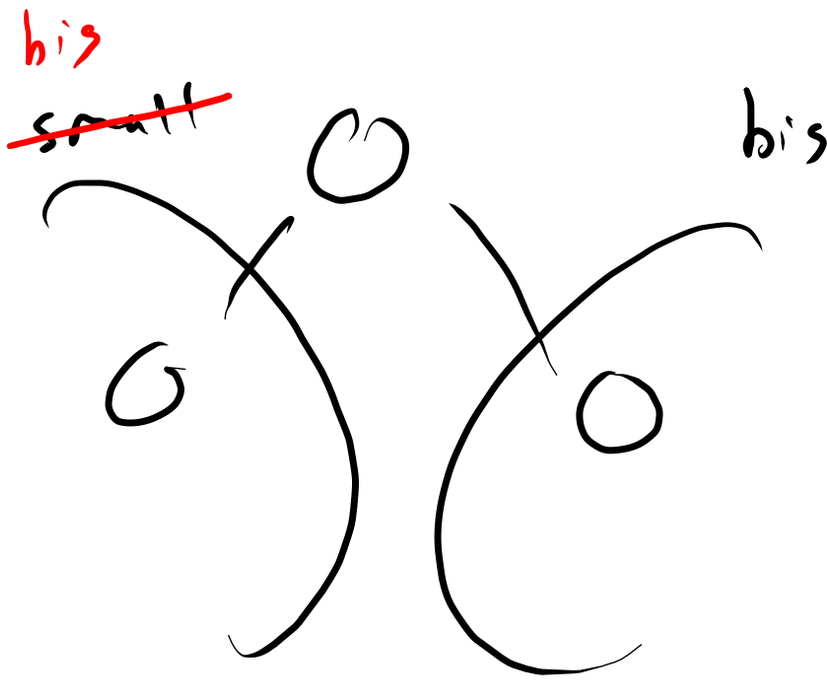
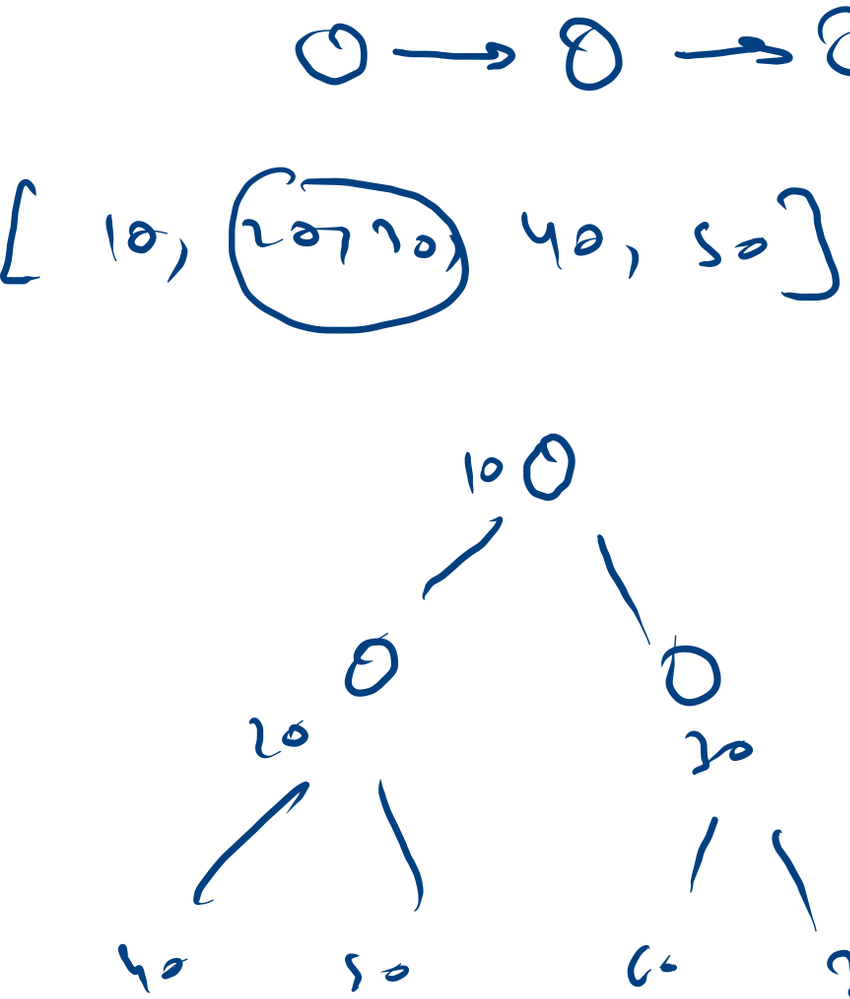
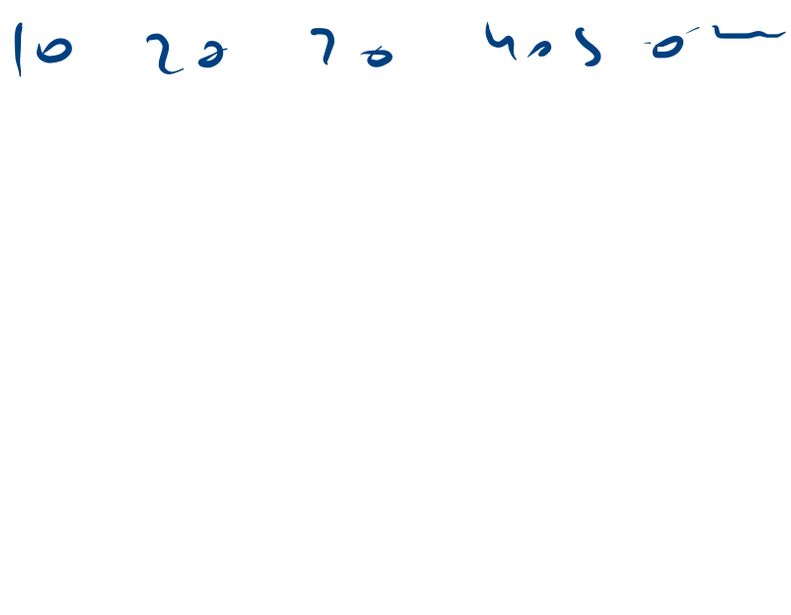
[~~10~~, 20, 40, 25, 35]

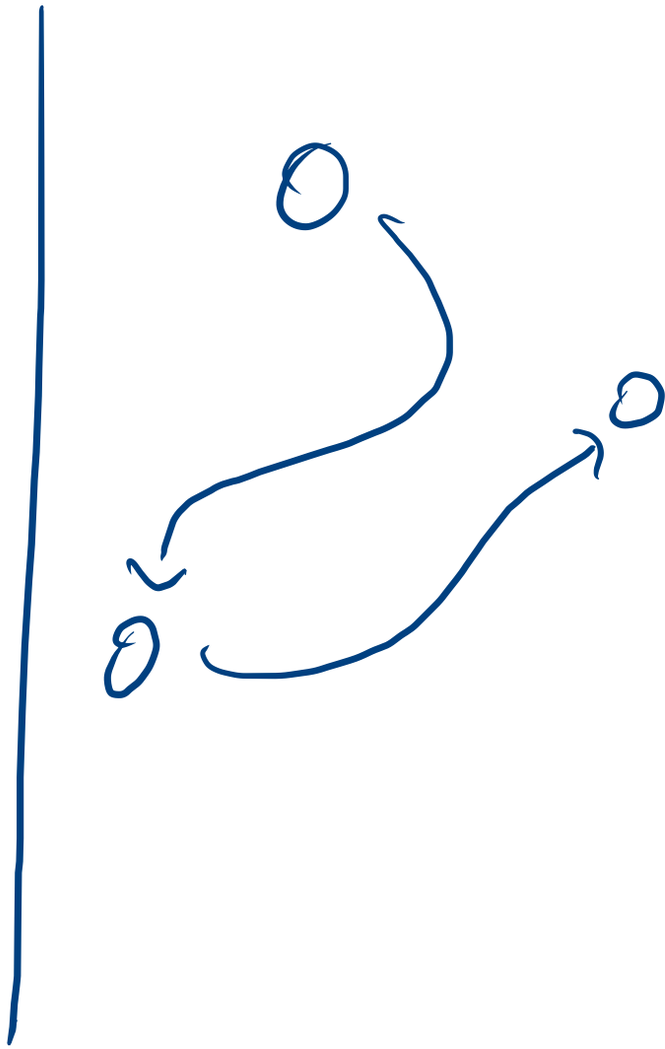
	PQ	ArL	BST/AVL
add →	log(n)	O(1)	log(n)
remove →	log(n)	O(n)	log(n)
peek →	O(1)	O(n)	log(n)





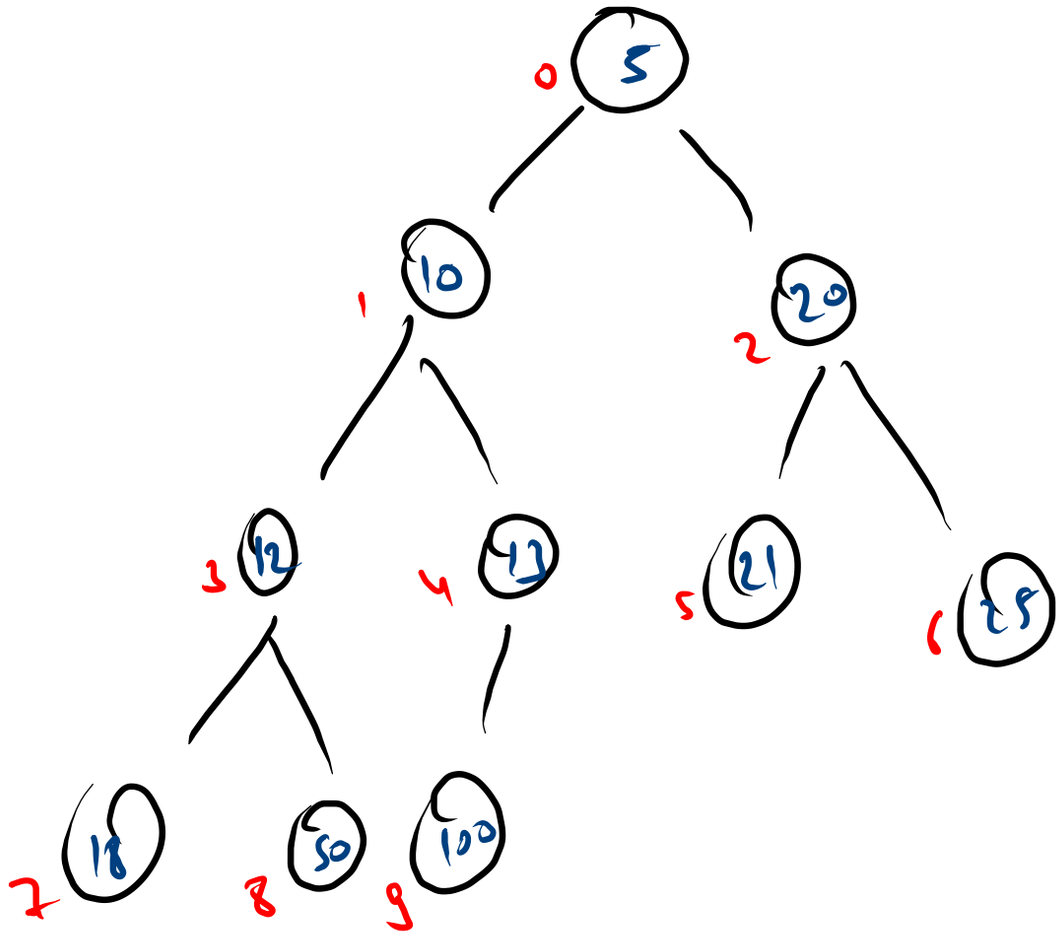






min-> max

[5 10 20 12 13 21 25 18 50 100]
0 1



- Map
- CBT (ولي)

child → 2xi+1
→ 2xi+2

parent → (i-1)/2

$\log_2(n)$

HOT

CBT

$\gamma_{comm} \rightarrow S$ ✓✓

downheapify

0 10

1 12

2 20

3 18

4 17

5 21

6 25

7 100

8 50

9 5

10

first

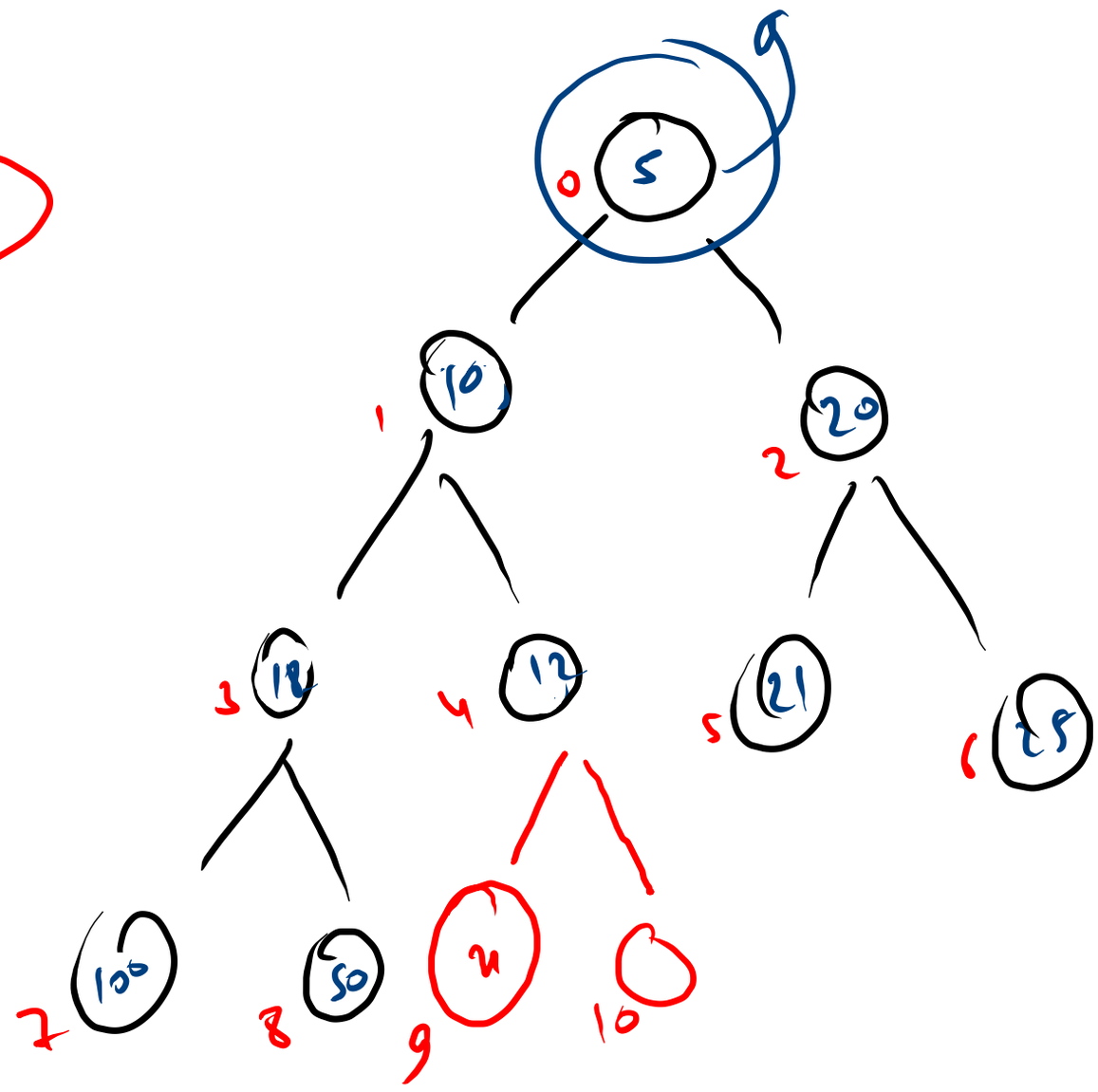
a. compare to (h)

$q < b$ (-V)

[5 10 20 18 12 21 25 100 50 n]
8 9

add 12 → 11

upheapify



nop

put(k,v) , get(k) , containsKey(k) , keySet() , size()

not exist — add new — null — false

exist — update — value — true

complexity → O(1) O(1) O(1) O(n) O(1)

get(k)

put(k,v)

"abc" → 5

bc2 → 4

"else"

size → n

hash(k) % N

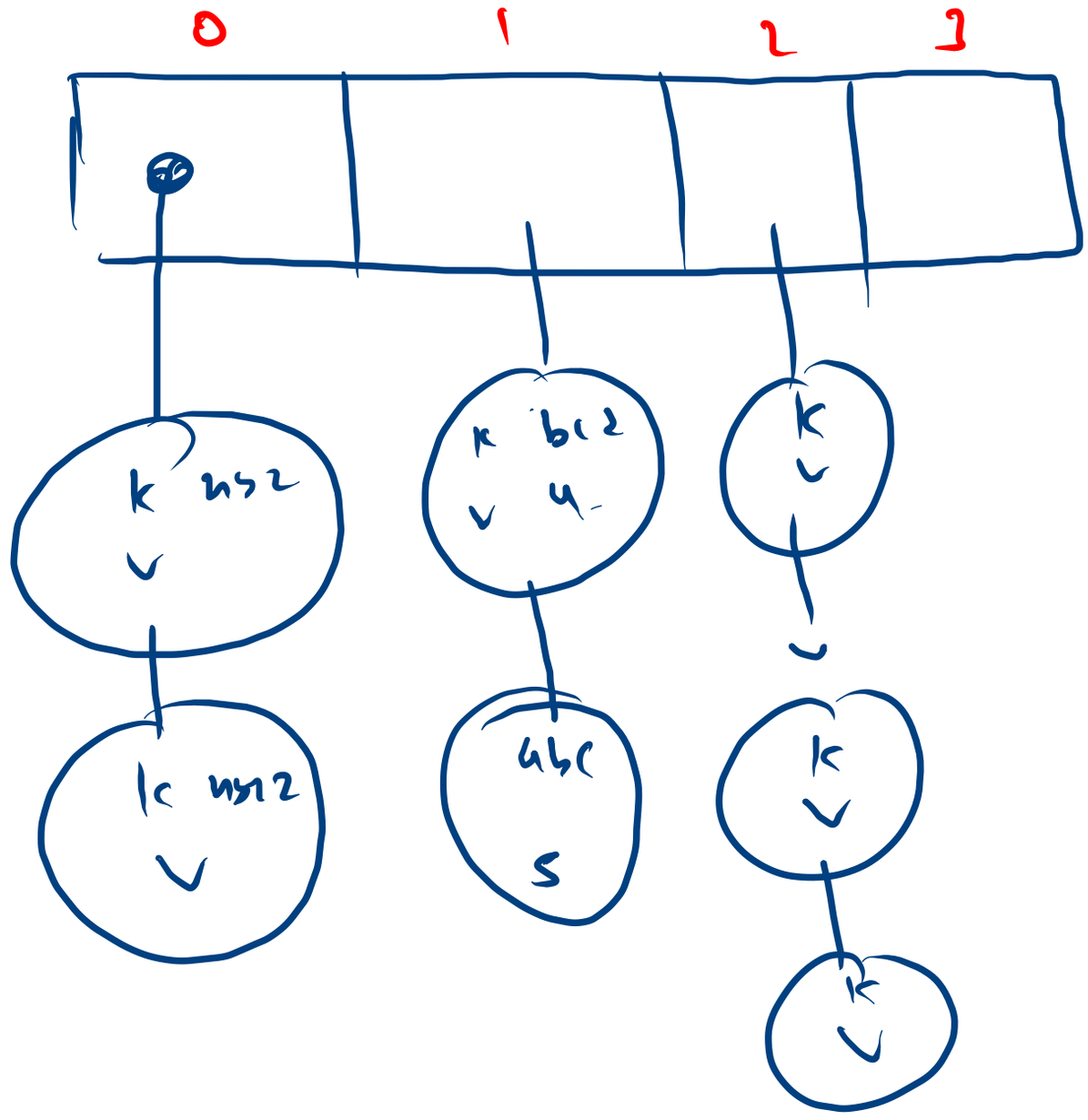
$\frac{n}{N} \leq \lambda$

hash Cache

k

"abc"

hash

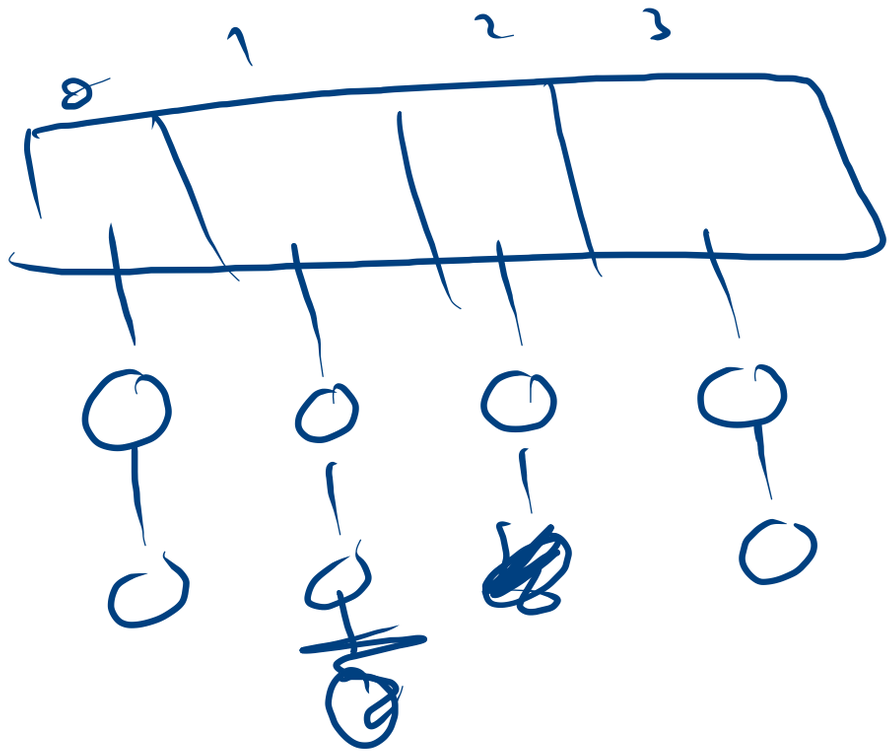


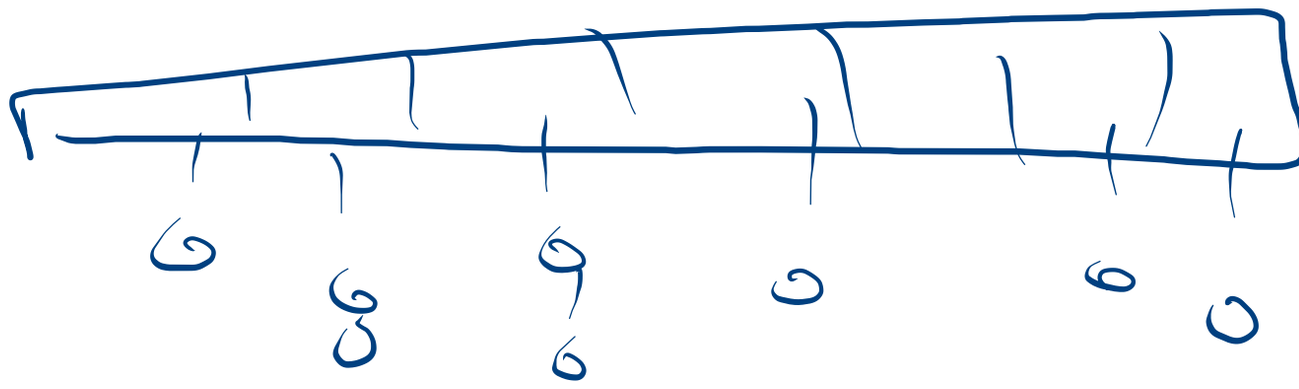
$\lambda = 2$

$\frac{8}{4} = 2$ $\quad$ $\frac{8}{4} = 2$

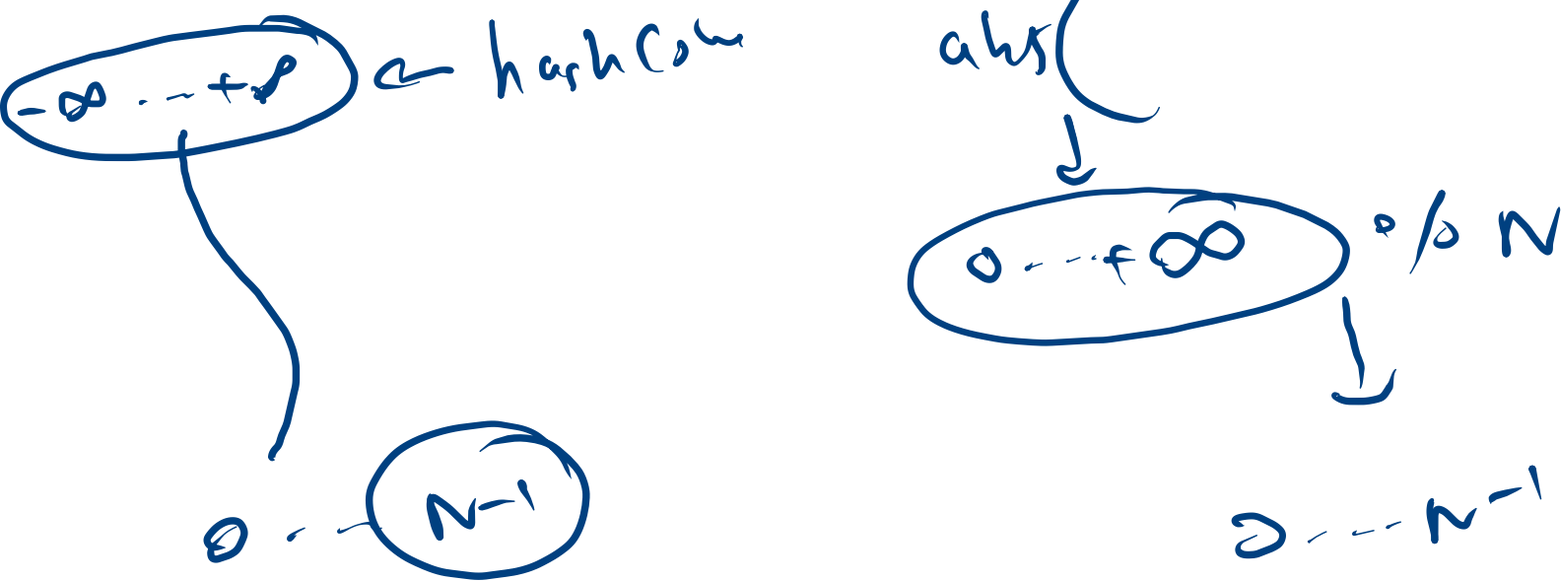
$\frac{7}{4} = 1.75$

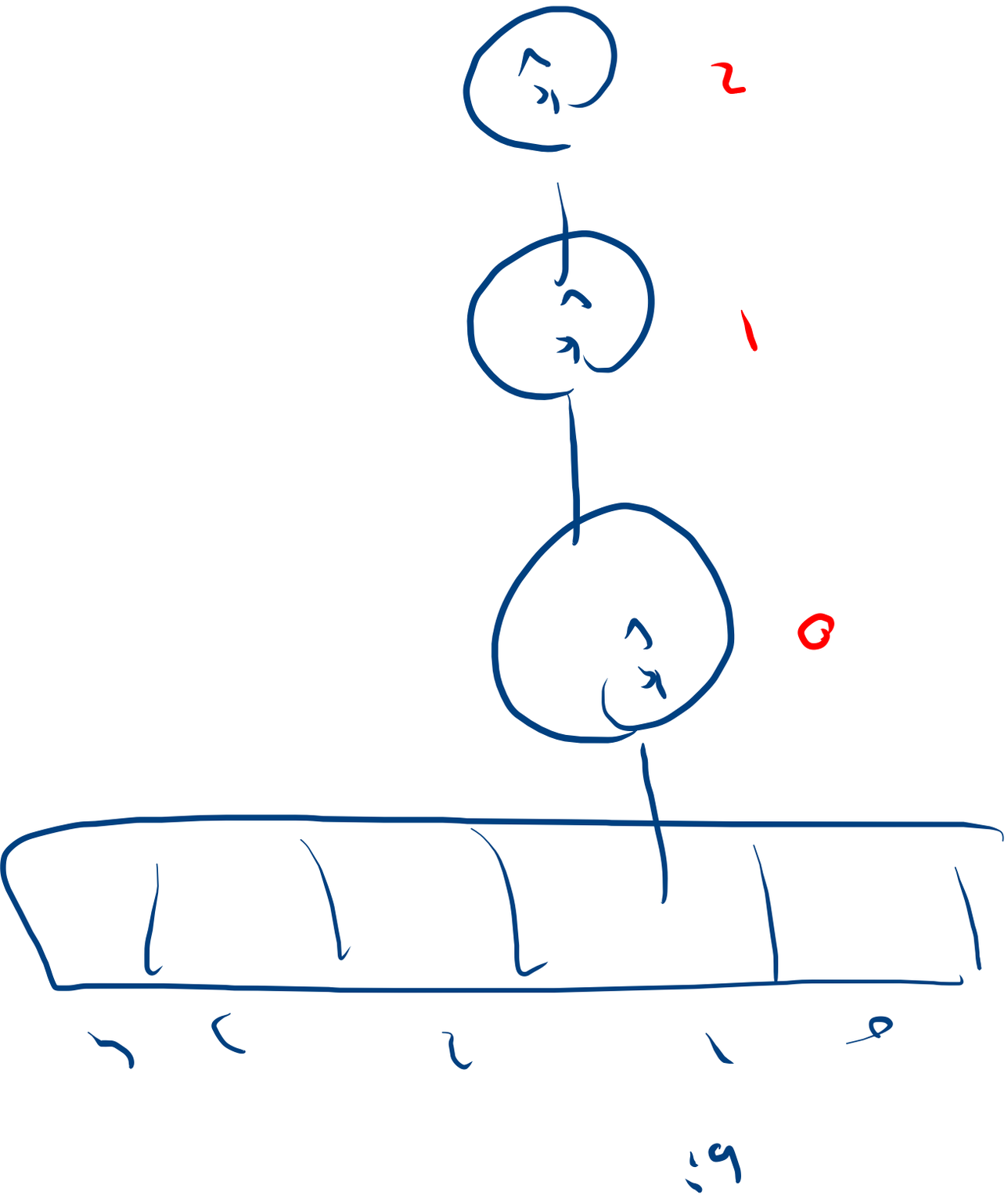
$\frac{5}{4} > 2$

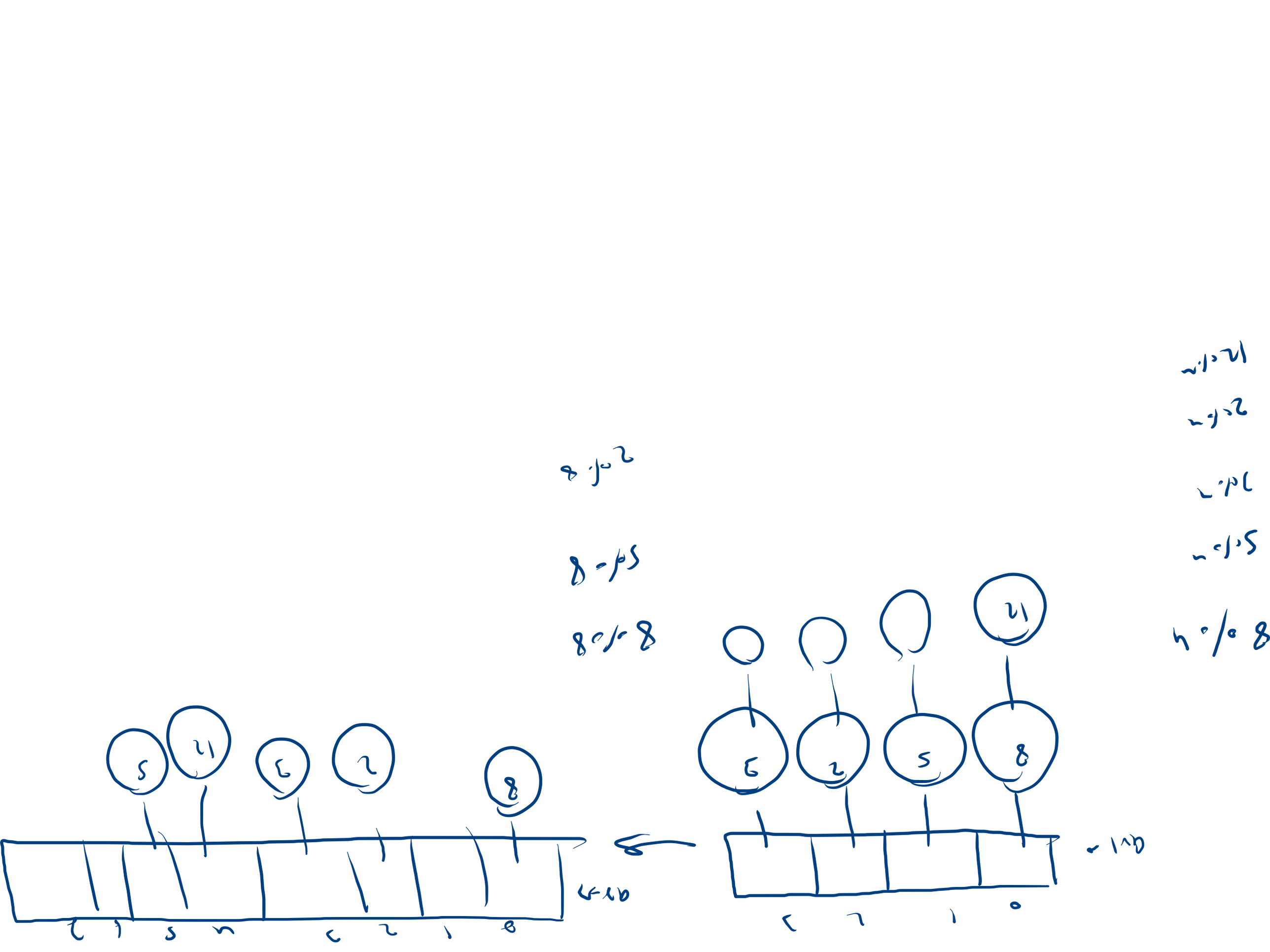




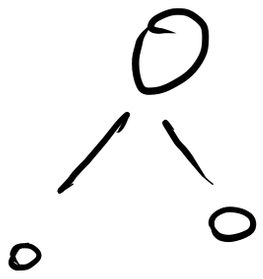
N




2
1
0

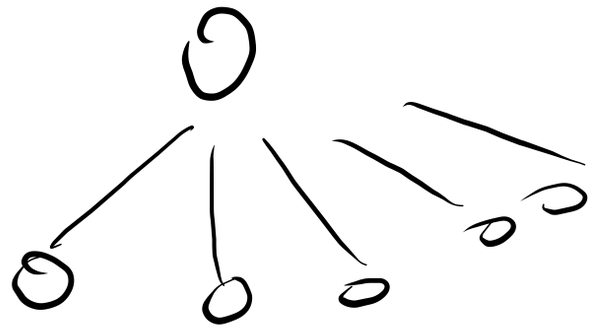


BT

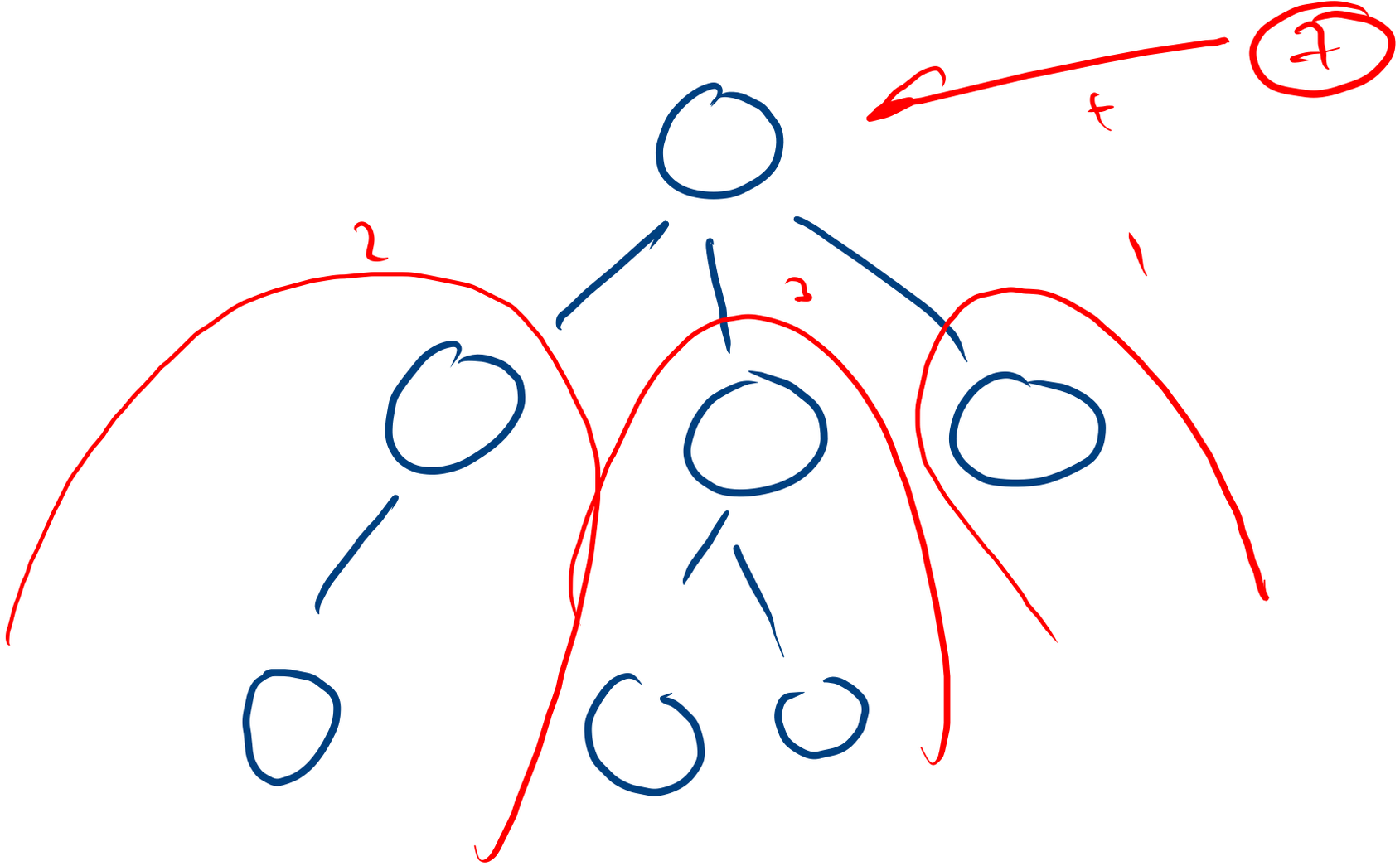


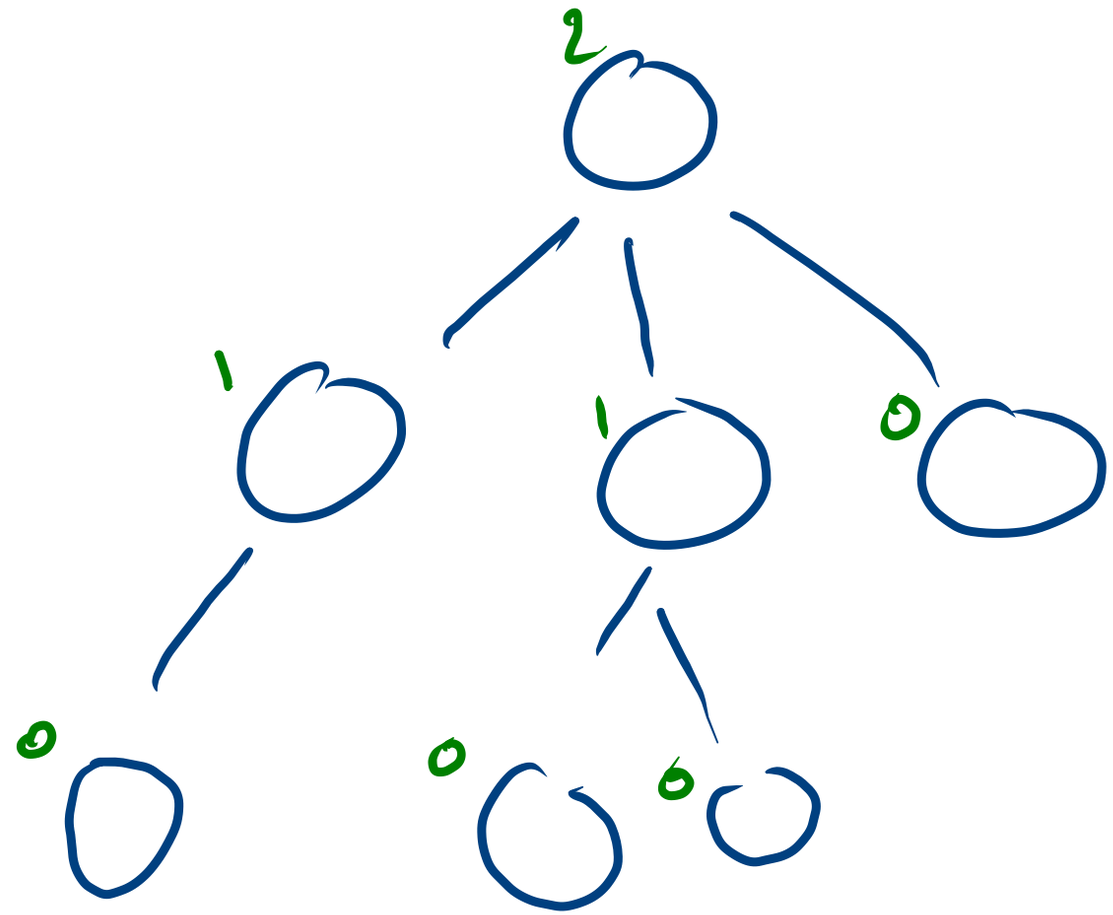
{
int data
N left
}

GT

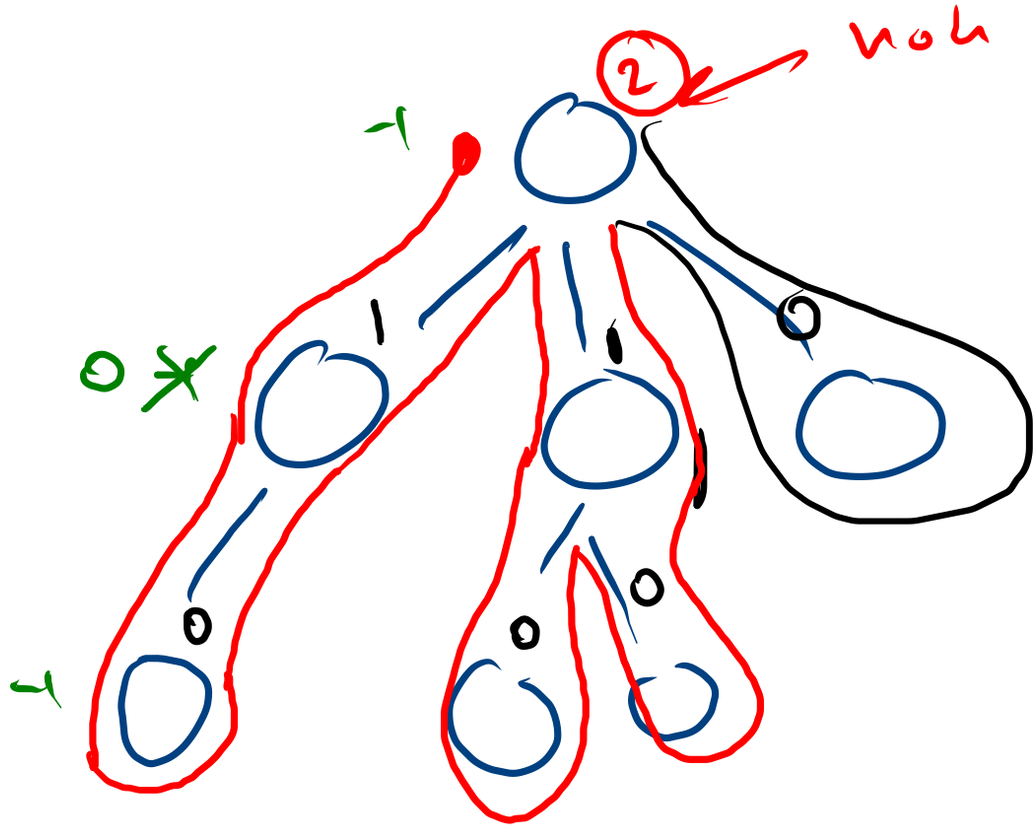


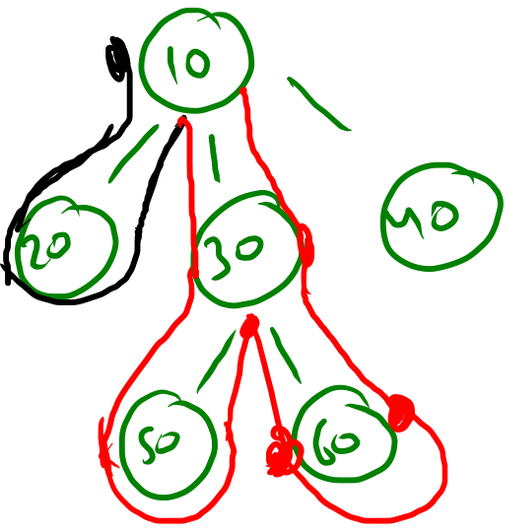
{
int data
AL<NODE>
}


f
f
2
3
1

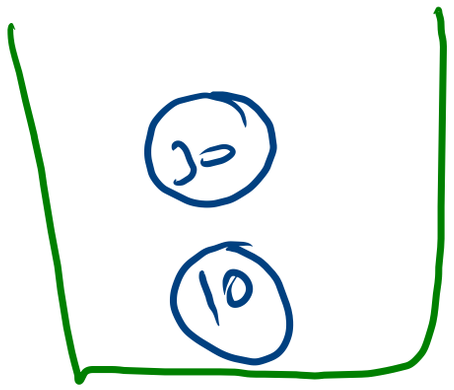

2
1
1
0
0
0
0

```java
public static int height(Node node) {

    int h = -1;

    for(Node child: node.children){
        h = Math.max(h, height(child));
    }

    return h+1;
}
```





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 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



Node Pre 10

Edge Pre 10--20

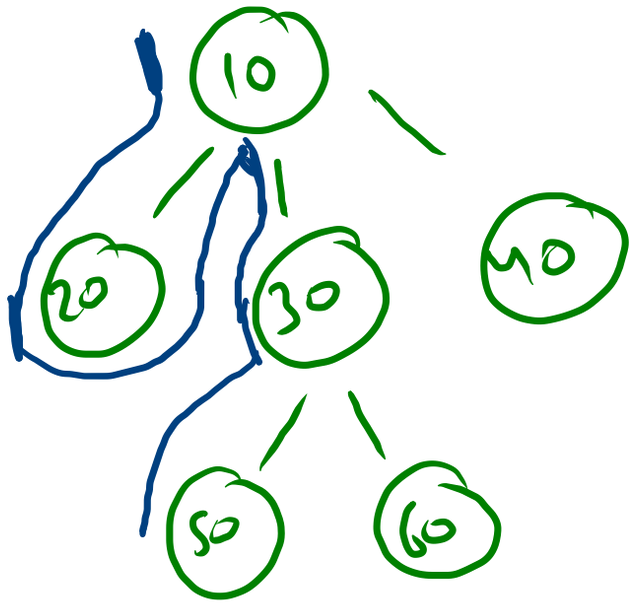
Node Pre 20

Node Post 20

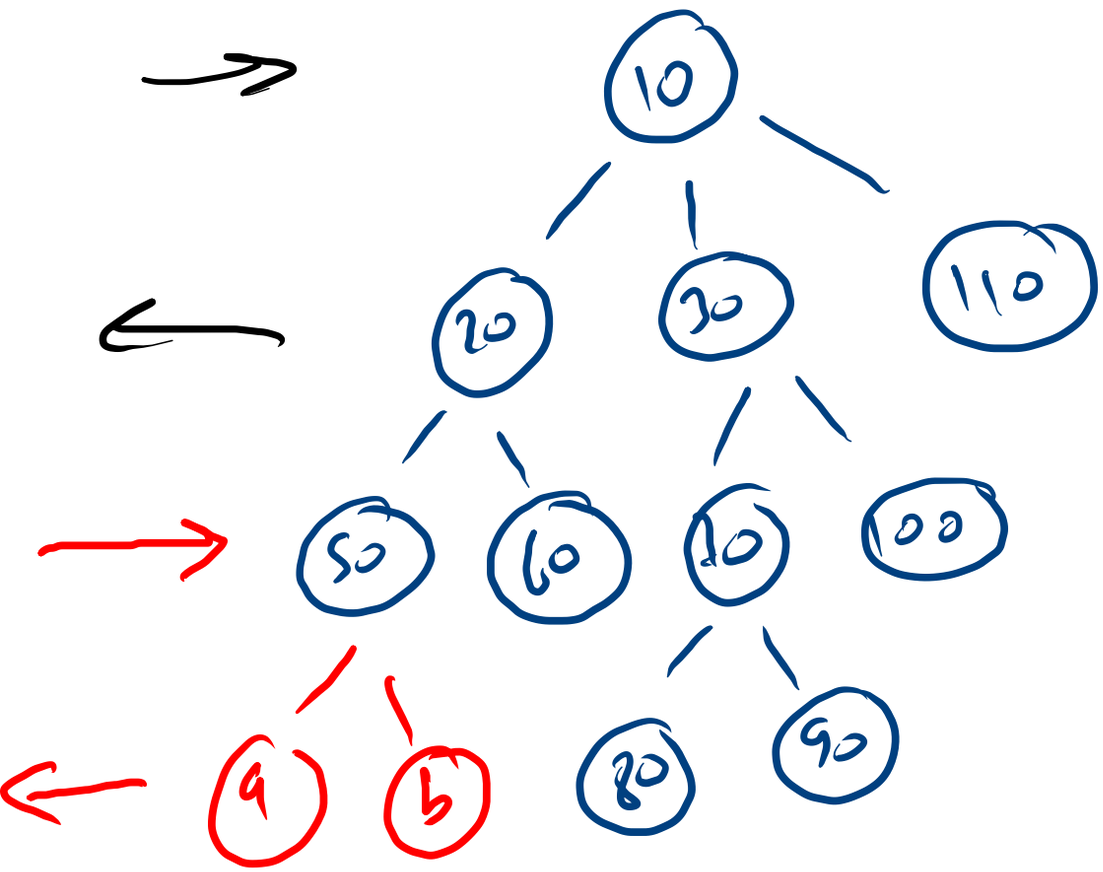
Edge Post 10--20

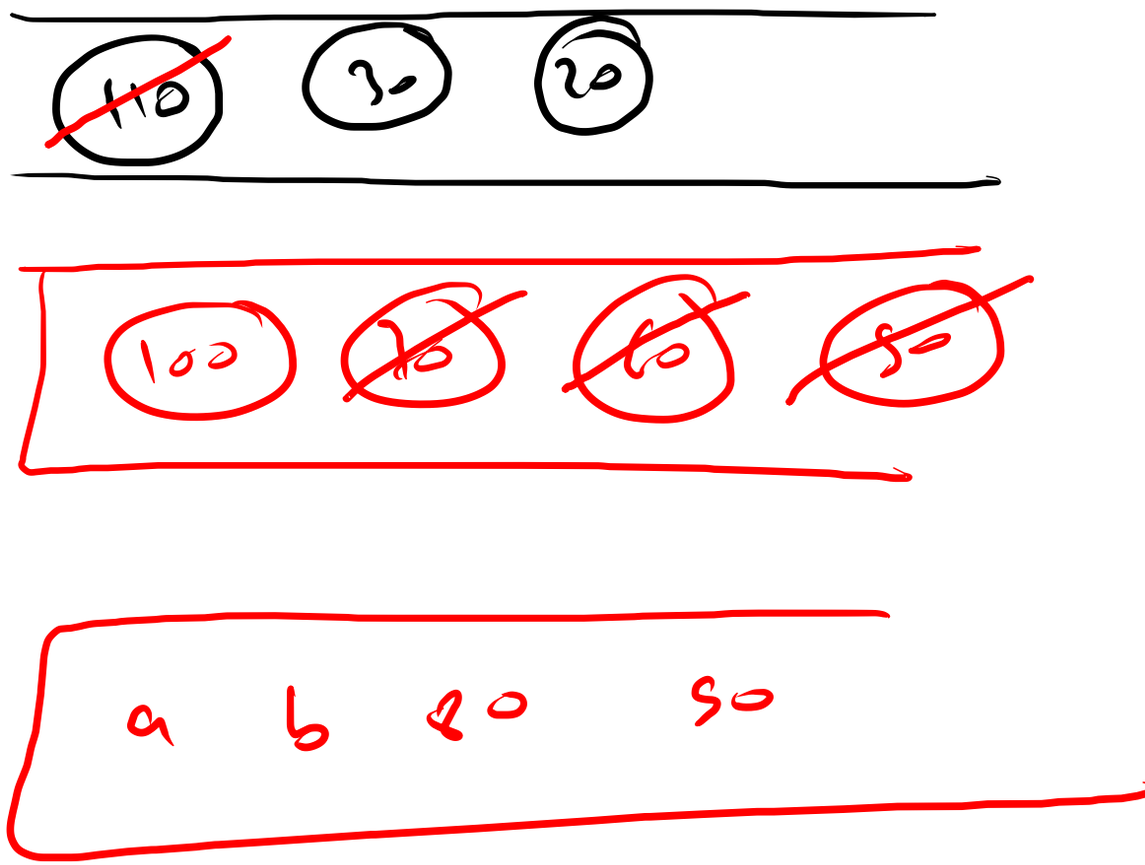
Edge Pre 10--30

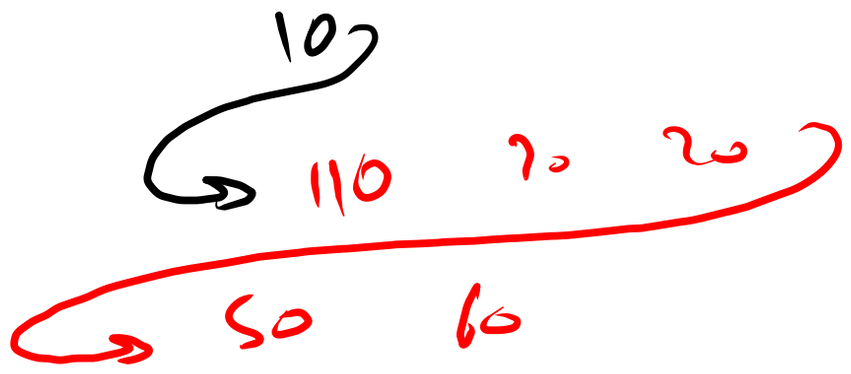
Node Pre 30



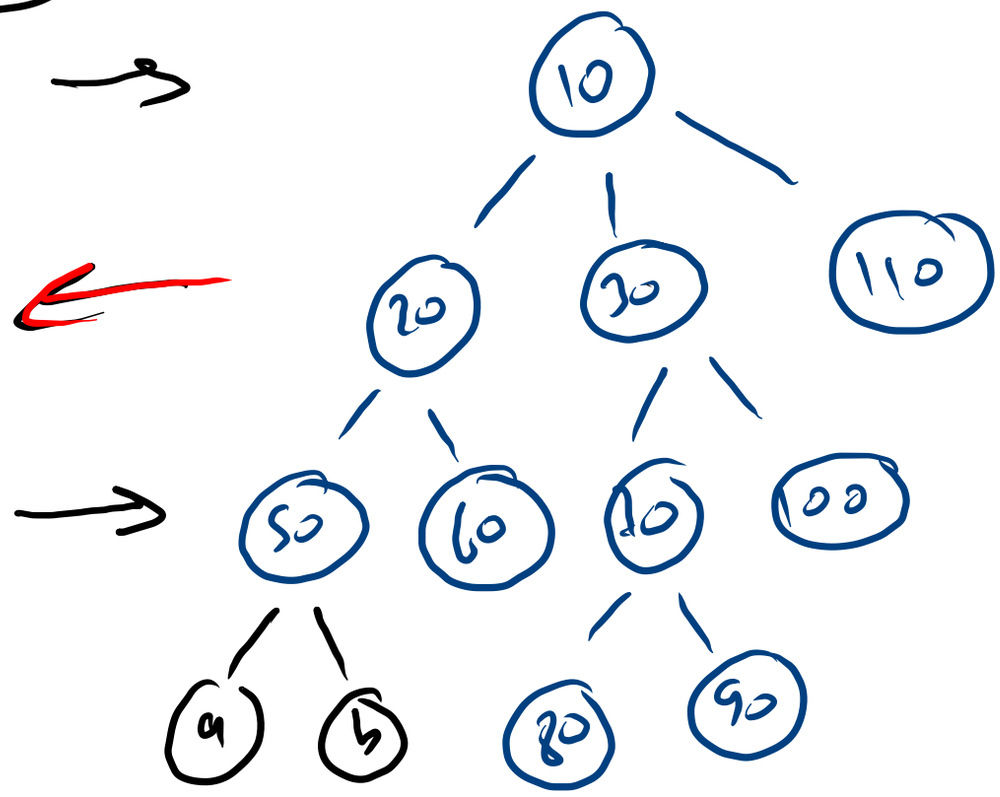

```java
public static void traversals(Node node){
    System.out.println("Node Pre "+node.data);

    for(Node child: node.children){
        System.out.println("Edge Pre "+node.data+"--"+child.data);

        traversals(child);

        System.out.println("Edge Post "+node.data+"--"+child.data);
    }
    System.out.println("Node Post "+node.data);

}
```

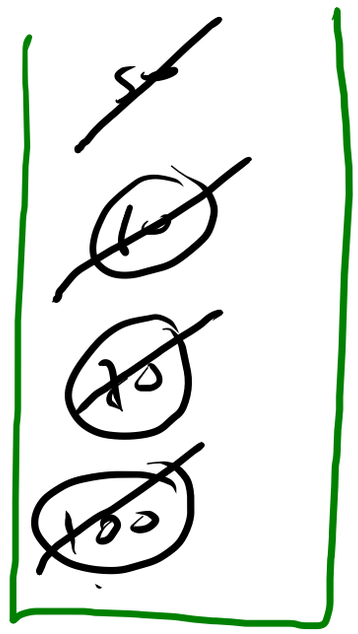

10
20
30
110
50
60
70
00
a
b
80
90

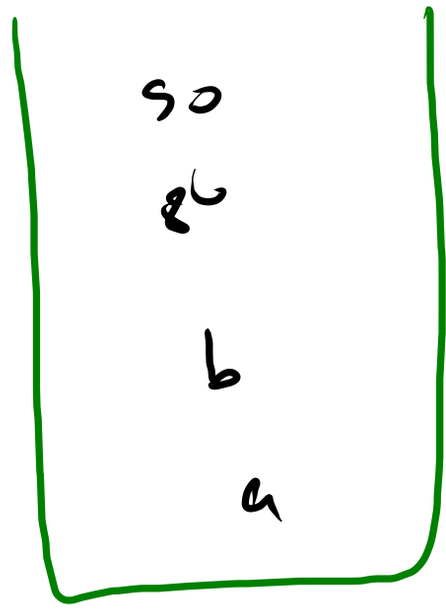

110
30
20
100
70
60
50
a
b
80
90

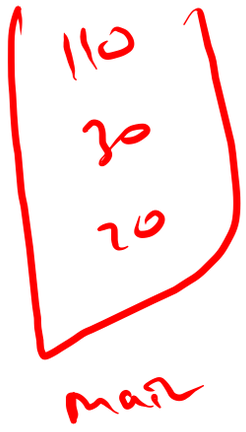

10
110
70
20
50
60

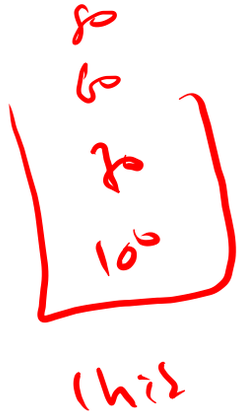
LTR = rm

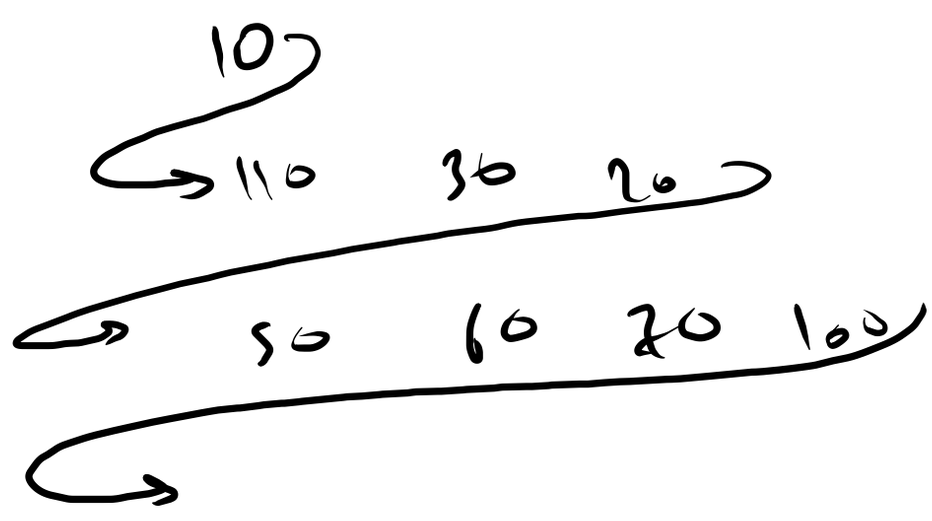

10
20
30
110
50
60
70
00
a
b
80
90


main


50
60
70
b
a


110
30
20
main


50
60
30
10
this


10
110
30
20
50
10
20
100

```java
Stack<Node> main = new Stack<>();
Stack<Node> child = new Stack<>();

main.push(node);
boolean ltr = true;
while(main.size() > 0){

    while(main.size() > 0){
        Node n = main.pop();
        System.out.print(n.data+" ");

        if(ltr){
            for(Node c: n.children){
                child.push(c);
            }
        }else{
            for(int i=n.children.size()-1;i>=0;i--){
                Node c = n.children.get(i);
                child.push(c);
            }
        }
    }

    ltr = !ltr;
    System.out.println();

    var t = main;
    main = child;
    child = t;
}
```