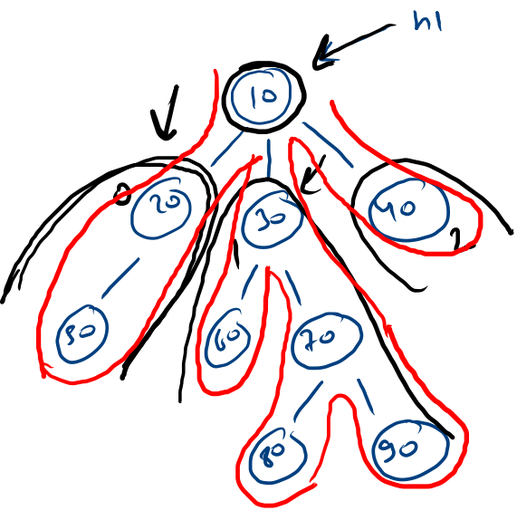
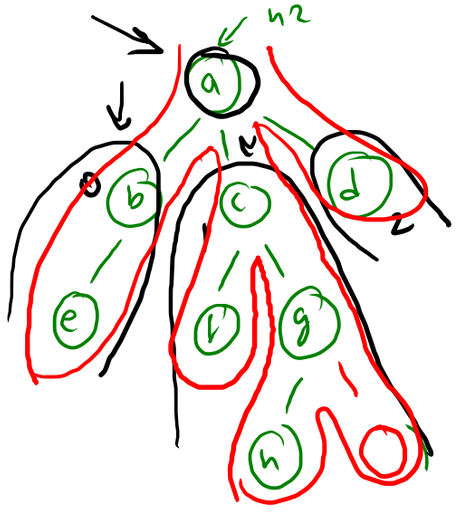
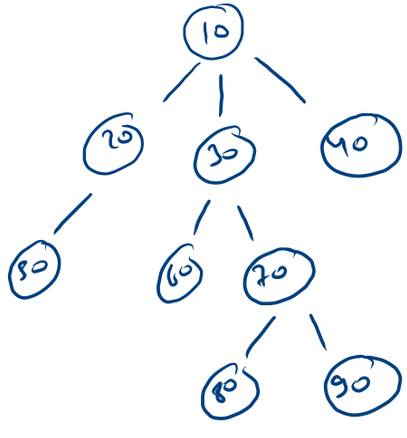


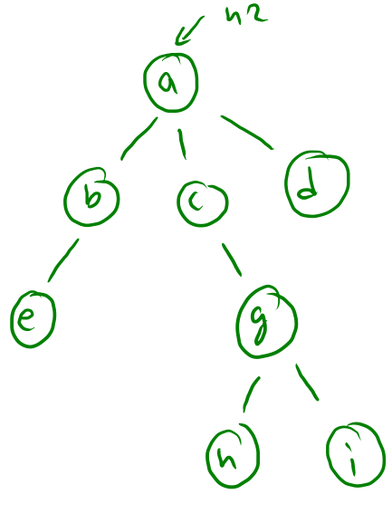


→ true

number of child

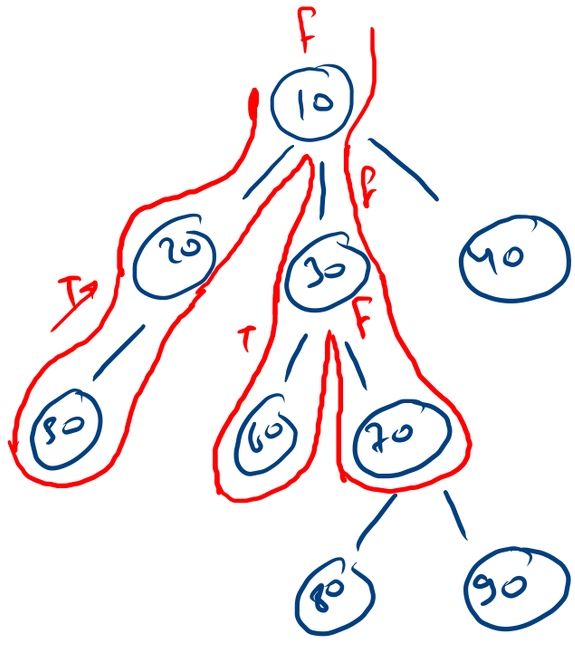
faith

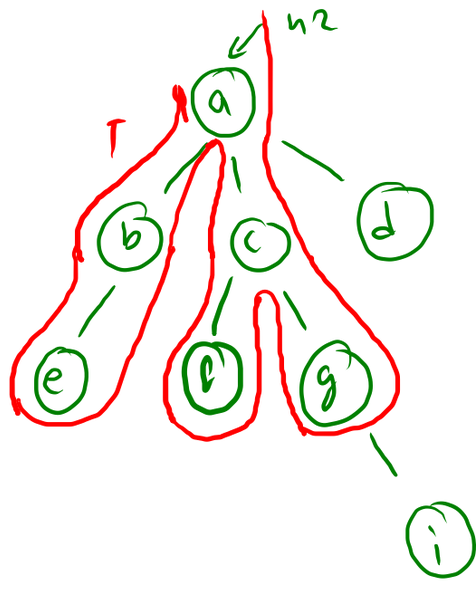


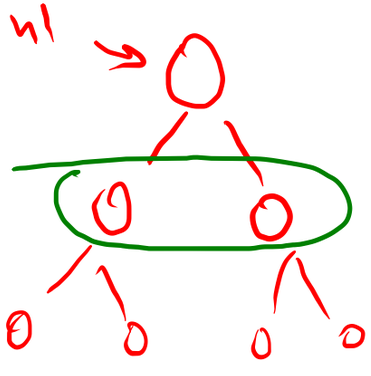


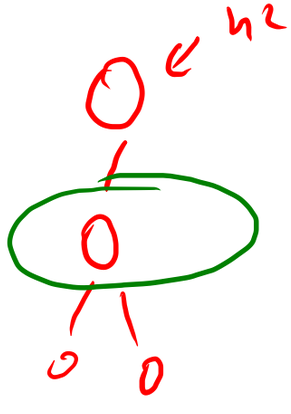
→ false


```java
public static boolean areSimilar(Node n1, Node n2) {

    if(n1.children.size() != n2.children.size()){
        return false;
    }

    for(int i=0;i<n1.children.size();i++){
        Node c1 = n1.children.get(i);
        Node c2 = n2.children.get(i);

        if(areSimilar(c1, c2) == false){
            return false;
        }
    }
    return true;
}
```



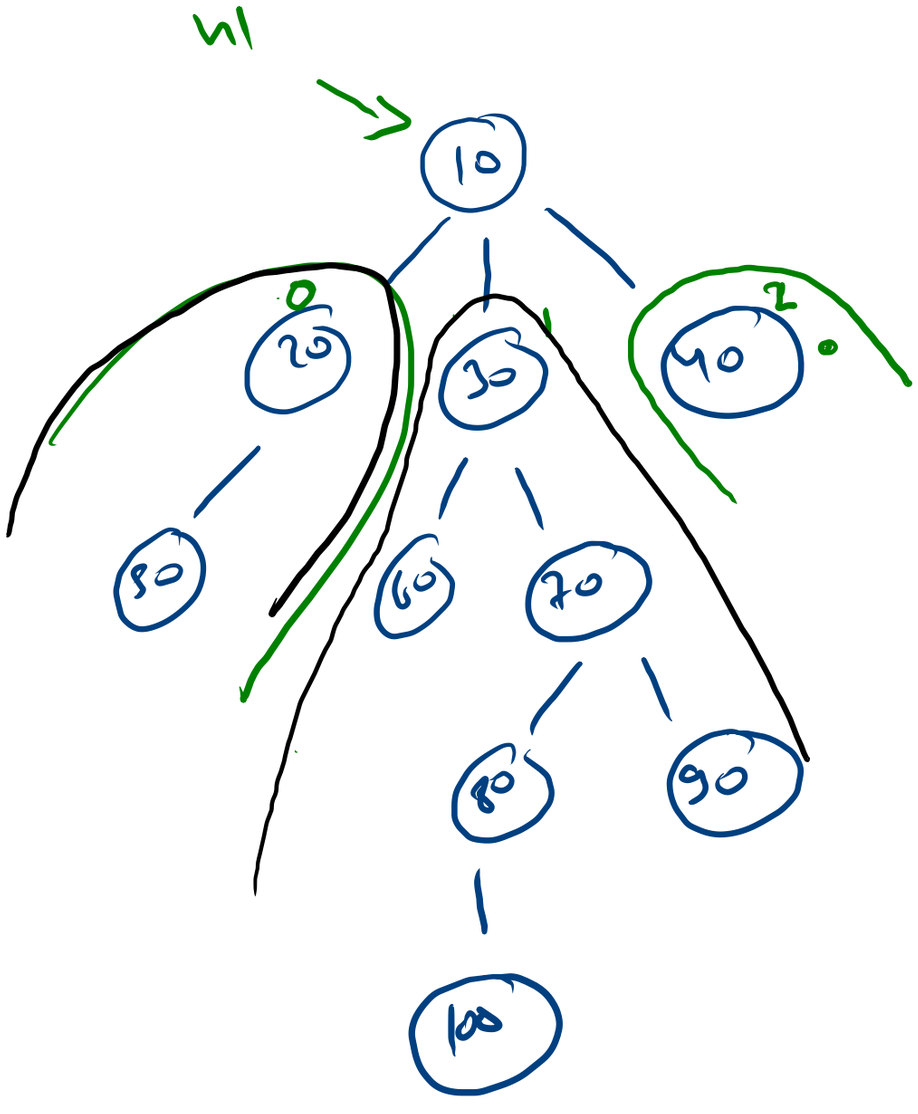


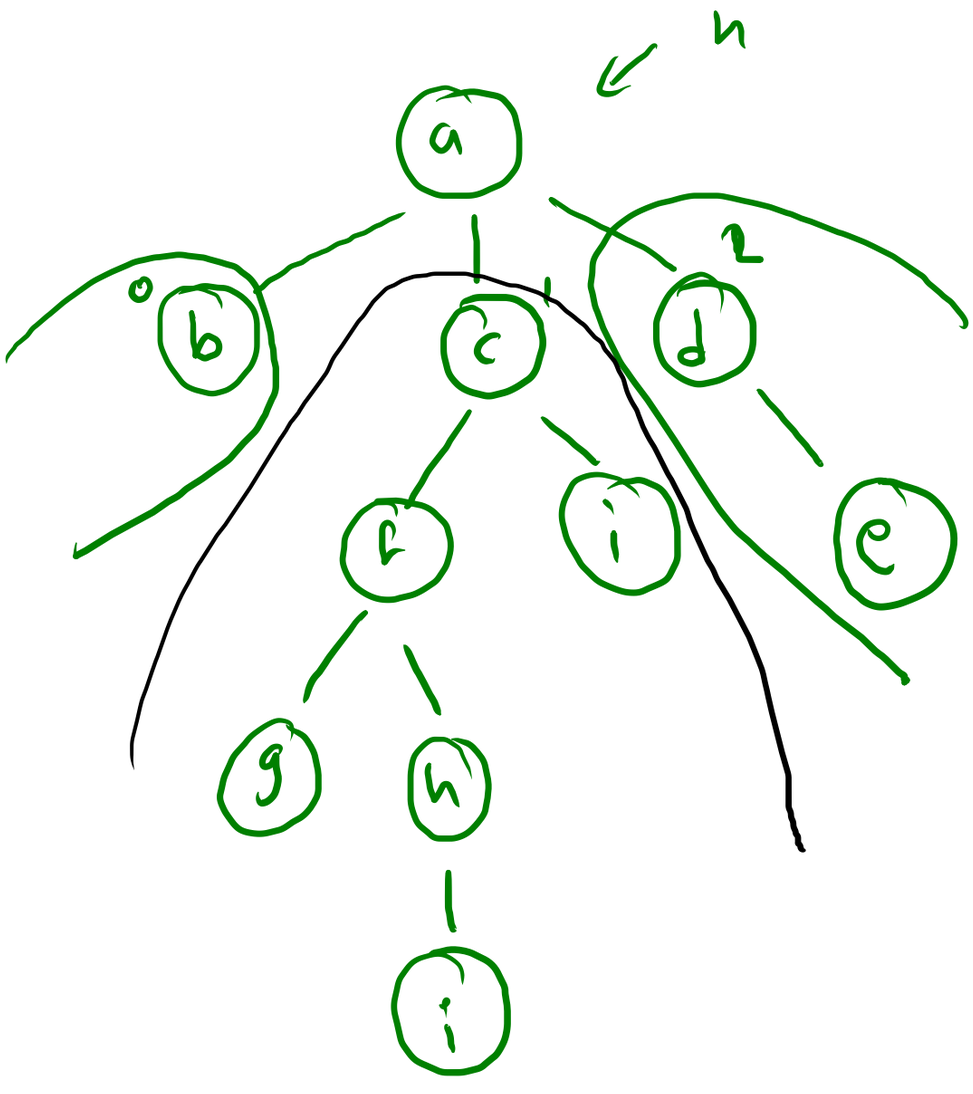




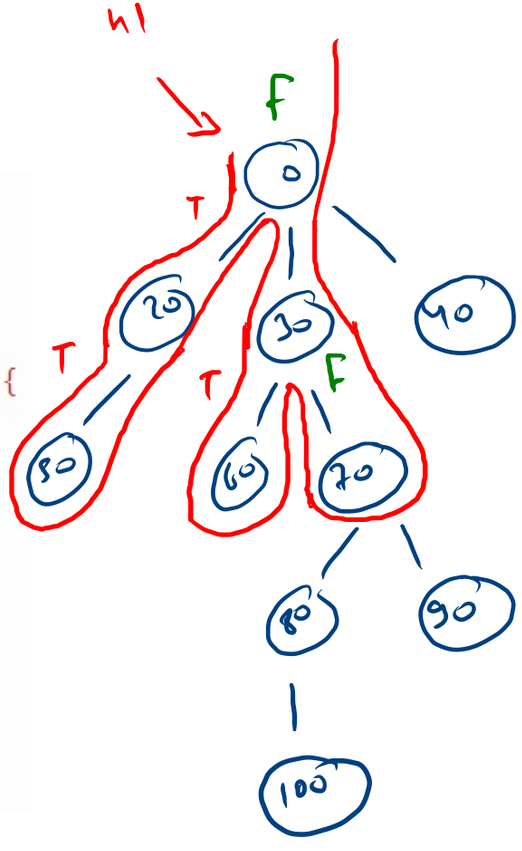
$O(n)$

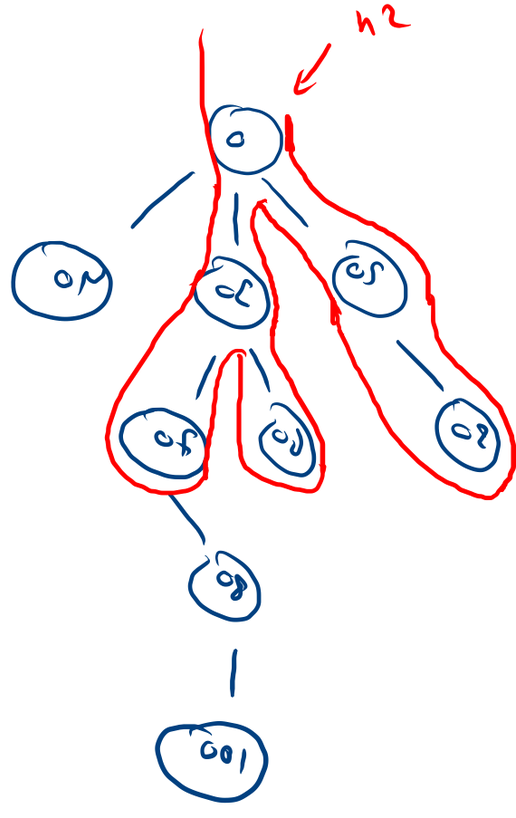
$\Omega(1)$

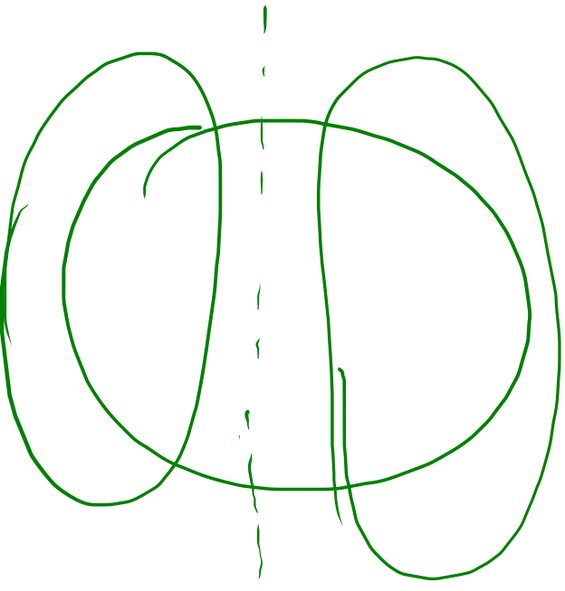

h1
10
20
30
40
50
60
70
80
90
100
0
1
2

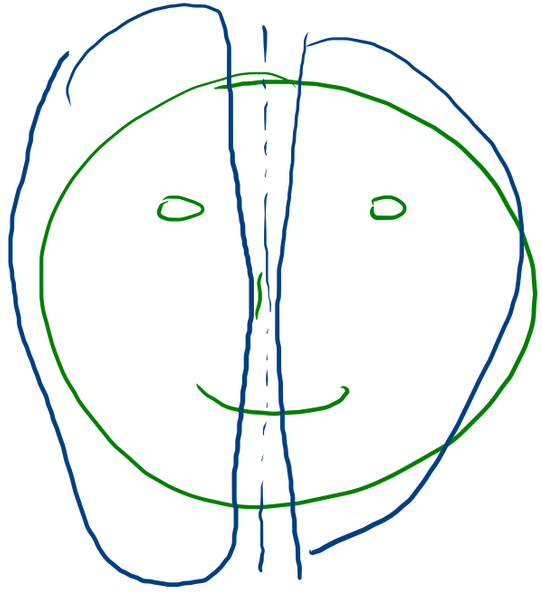

h
a
b
c
d
e
f
g
h
i
j
0
1
2

```java
public static boolean areMirror(Node n1, Node n2) {
    if(n1.children.size() != n2.children.size()){
        return false;
    }

    for(int i=0, j=n1.children.size()-1; j>=0;i++, j--){

        Node c1 = n1.children.get(i);
        Node c2 = n2.children.get(j);

        if(areMirror(c1, c2) == false){
            return false;
        }
    }

    return true;
}
```



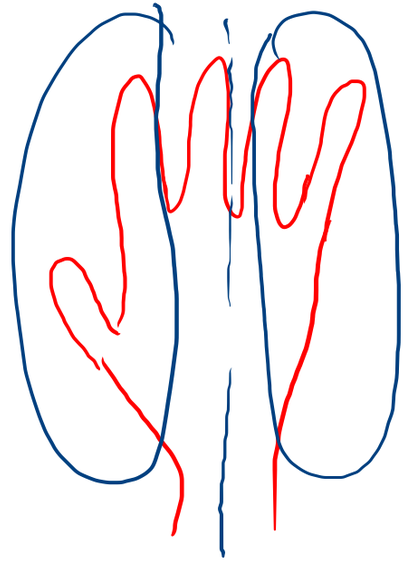




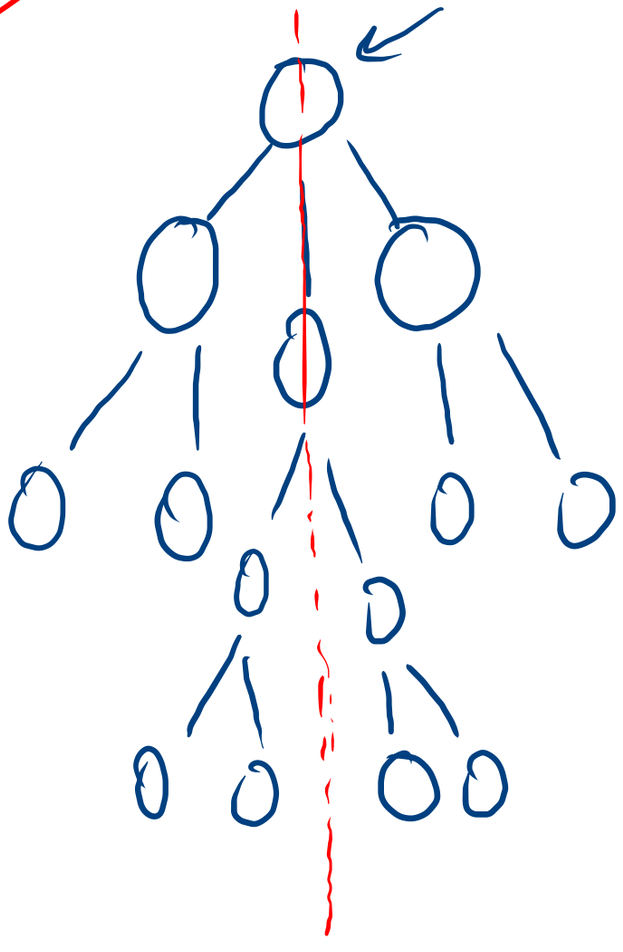


symmetric

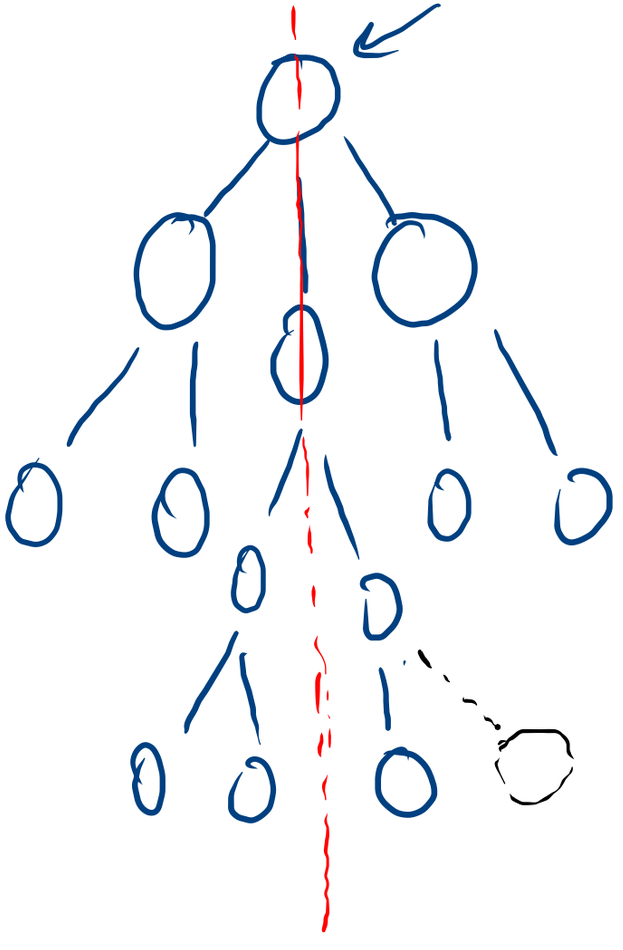
non-symmetric



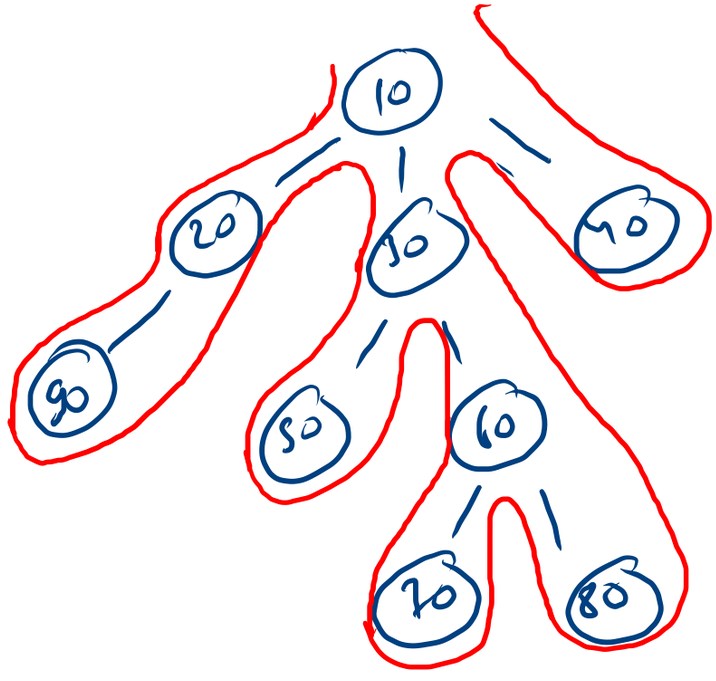
N.W.

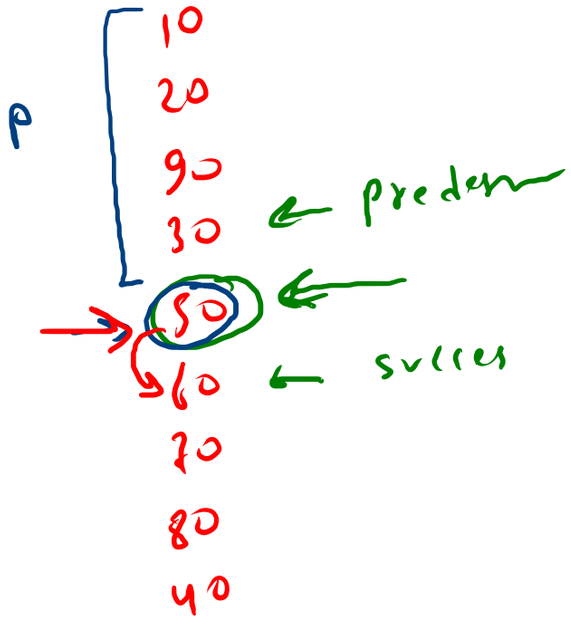


Symmetric



non - Symmetric





d → 50

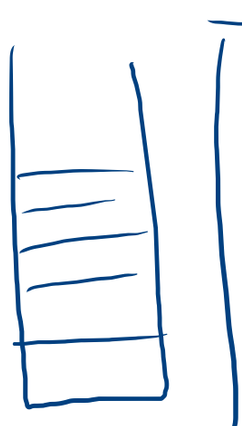
pre → 30

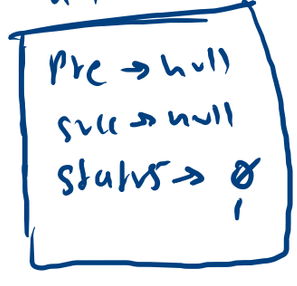
succ → 60

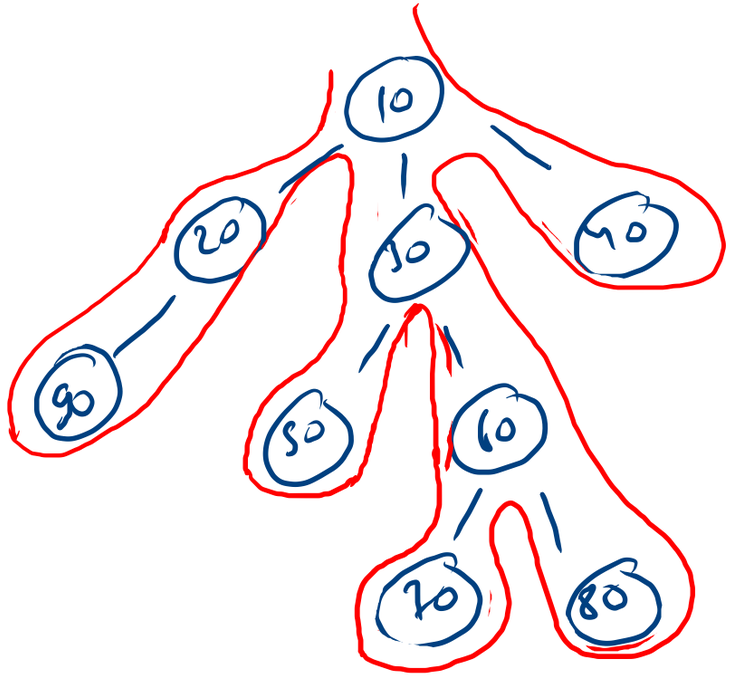
d → 10

pre → null

succ → 20







d = 50

pred = * ~~10~~ ~~20~~ ~~90~~ 30 ✓

succ = * 60

status = ~~0~~ ~~1~~ 2

status

0 → set pred

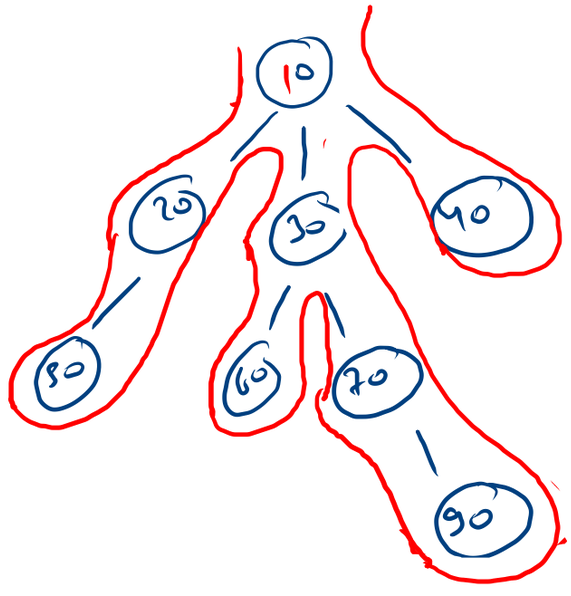
data == d → status = 1

1 succ set
status = 2

2 return

data = 40

```
static Node predecessor;
static Node successor;
static int status=0;
public static void predecessorAndSuccessor(Node node, int data) {

    if(node.data == data){
        status = 1;
    }else if(status == 0){
        predecessor = node;
    }else if(status == 1){
        successor = node;
        status = 2;
    }

    for(Node child: node.children){
        predecessorAndSuccessor(child, data);
    }
}
```



pre ✗ 10 20 30 40 50 60 70 90

succ ✗ ←

status 0 1

data → 60

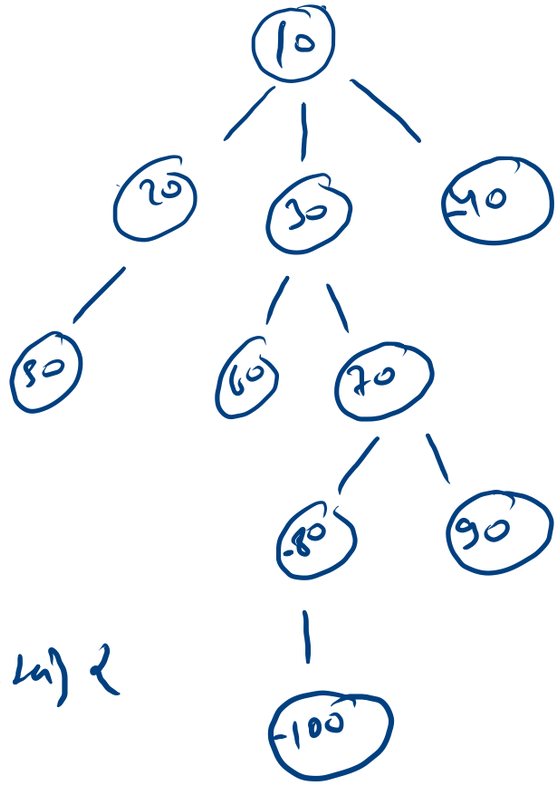
ceil → 70

floor → 50

data 65

ceil → 70

floor → 60



if (n.data < data) {

}

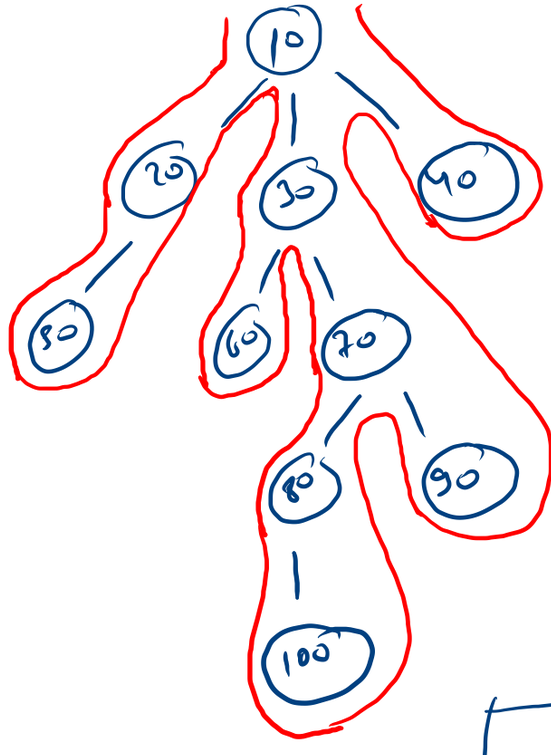
data → 10

ceil → 20 +∞

floor → -∞

data = 60

```
static int ceil;
static int floor;
public static void ceilAndFloor(Node node, int data) {

    if(node.data < data){
        floor = Math.max(floor, node.data);
    }
    if(node.data > data){
        ceil = Math.min(ceil, node.data);
    }

    for(Node child: node.children){
        ceilAndFloor(child, data);
    }
}
```



GT

ceil ~~+∞~~ 70

floor ~~-∞~~ 50 20 50 50 50

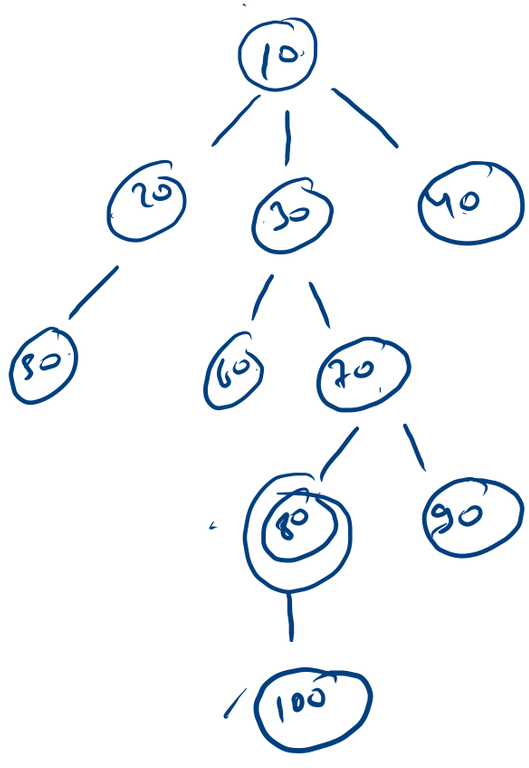
k 1 → 100

2 90

3 80

4 70

5 60



k=1 { data → ∞ ; floor 100

k=2 { data - 100 floor = -∞ ; floor → 90

k=3 { data = 90 floor = -∞ ; flor → 80

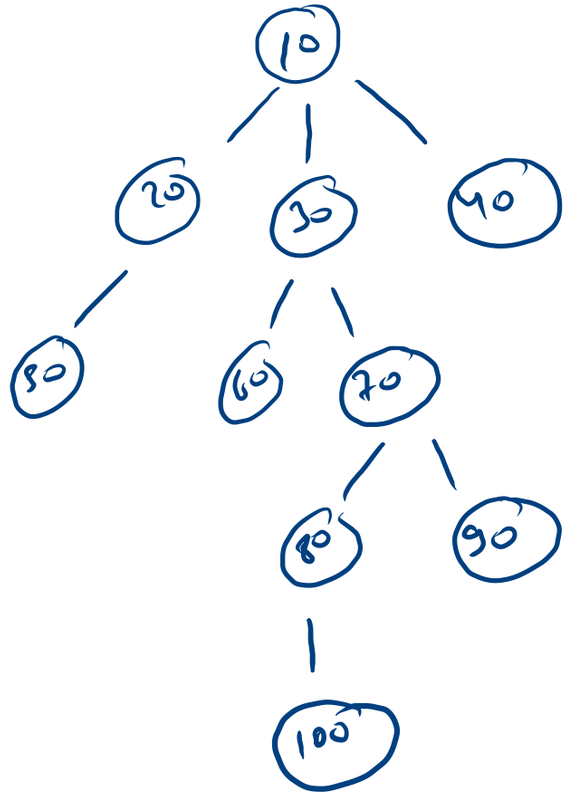
10 →

flor -∞

n x k

n x n $O(n^2)$

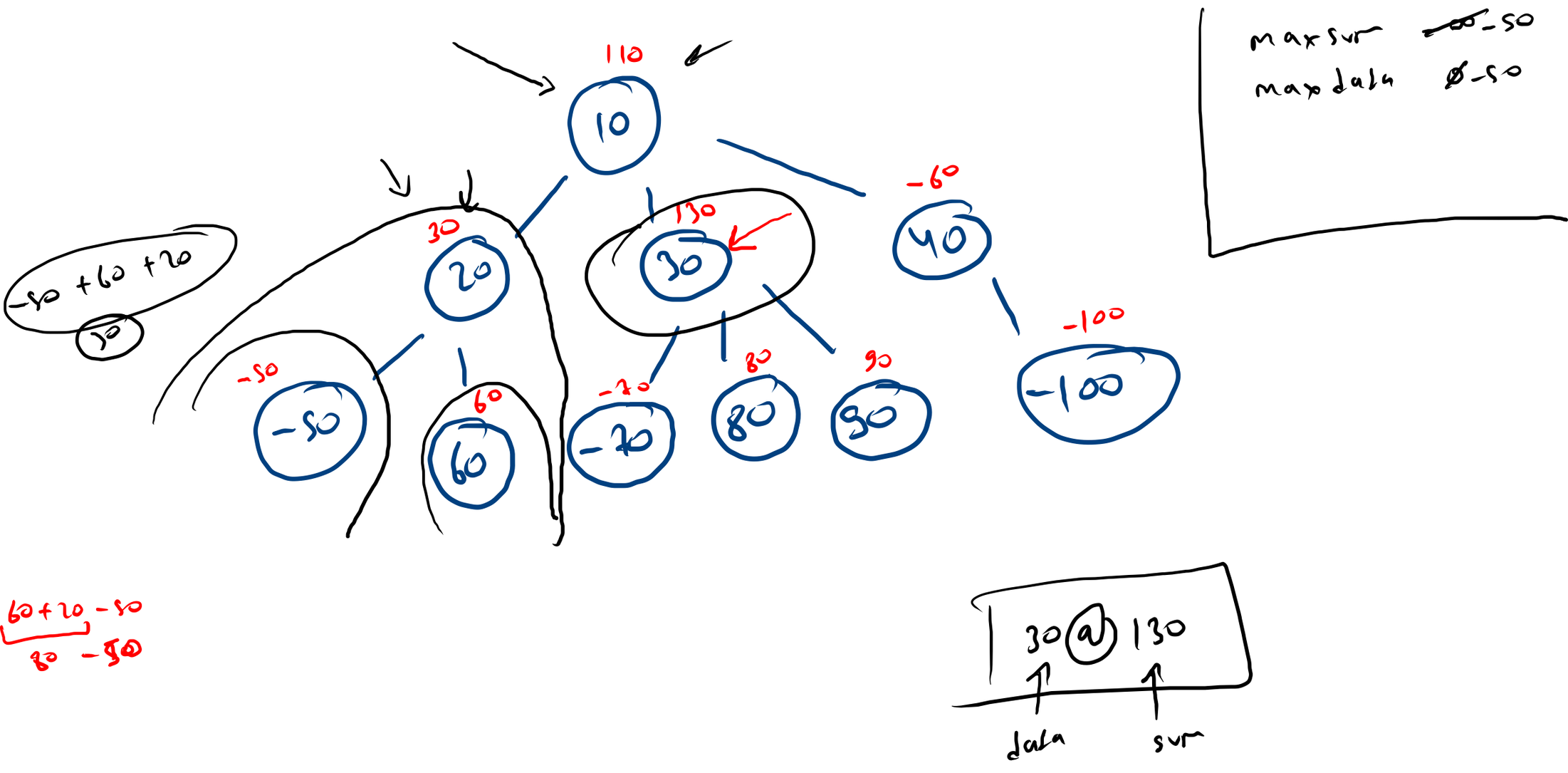
```java
public static int kthLargest(Node node, int k){
    int data = Integer.MAX_VALUE;
    for(int i=1;i<=k;i++){
      floor = Integer.MIN_VALUE;
      ceilAndFloor(node, data);
      data = floor;
    }
    return data;
}
```



floor ~~-∞~~ ~~100~~ ~~-∞~~ 90 ~~-∞~~ 80

data ~~∞~~ ~~100~~ 90 80

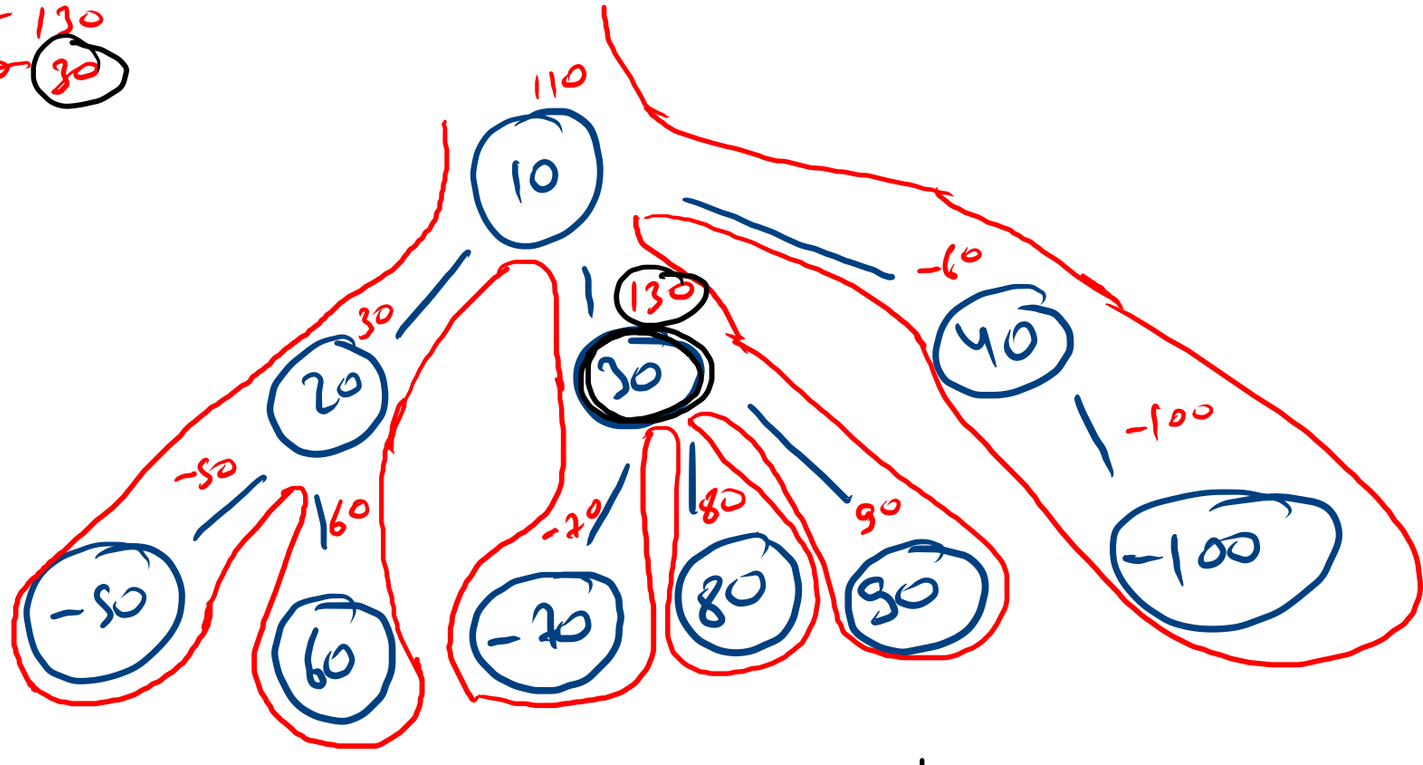
k - 1 2 3

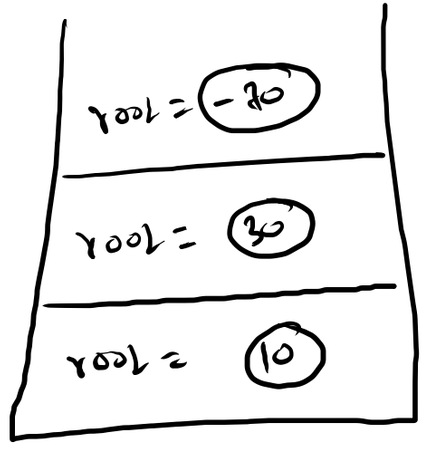

110
10
30
20
130
30
-60
40
-50 +60 +20
30
-50
-50
60
60
-70
-70
80
80
90
90
-100
-100
maxsum -∞ -50
maxdata ∅ -50
60+20-50
80 -50
30 @ 130
data
sum

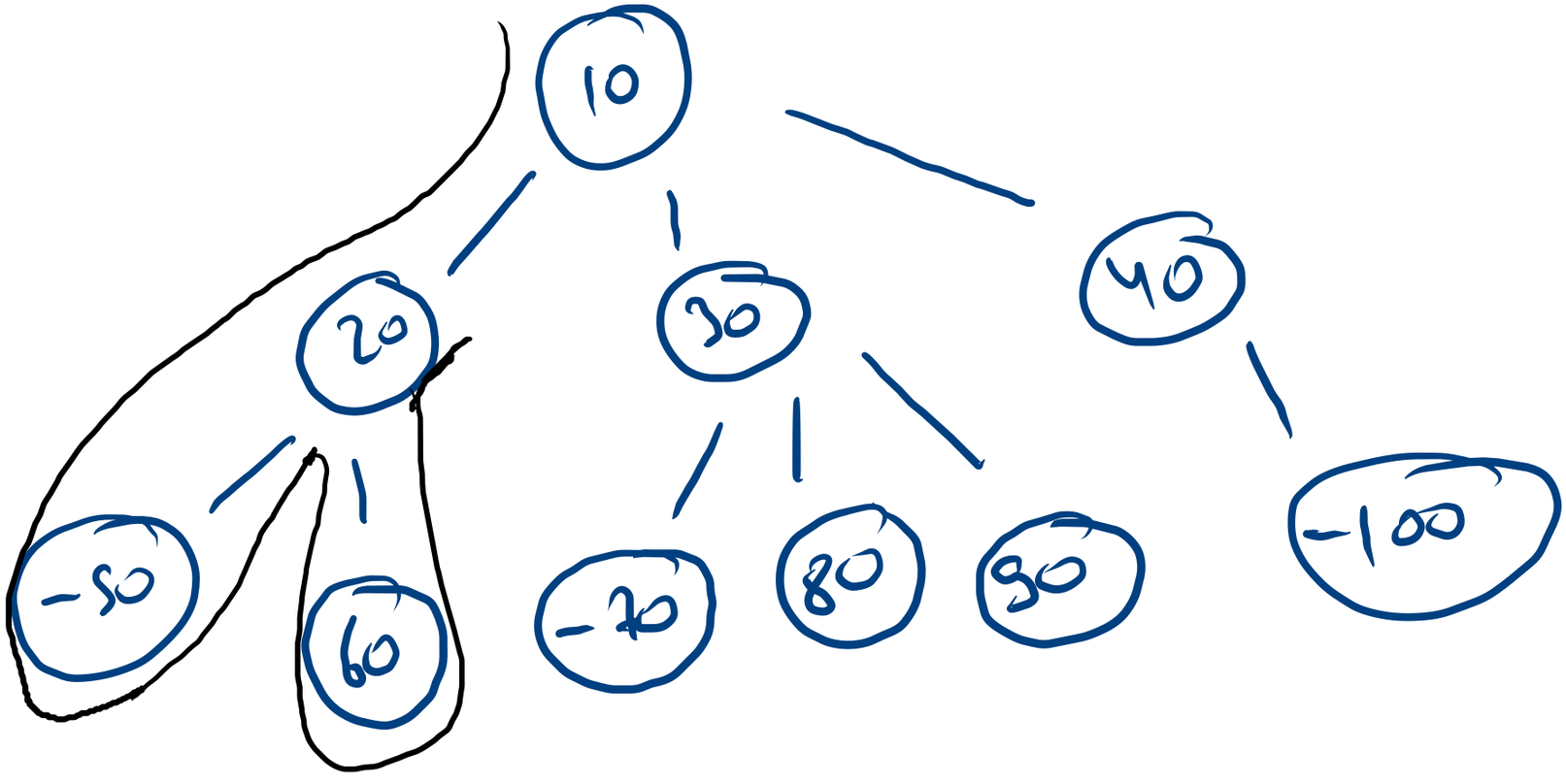
maxsum = ~~-∞~~ ~~-50~~ ~~60~~ ~~80~~ ~~90~~ 130

maxdata = ~~0~~ ~~-50~~ ~~60~~ ~~20~~ ~~90~~ 30

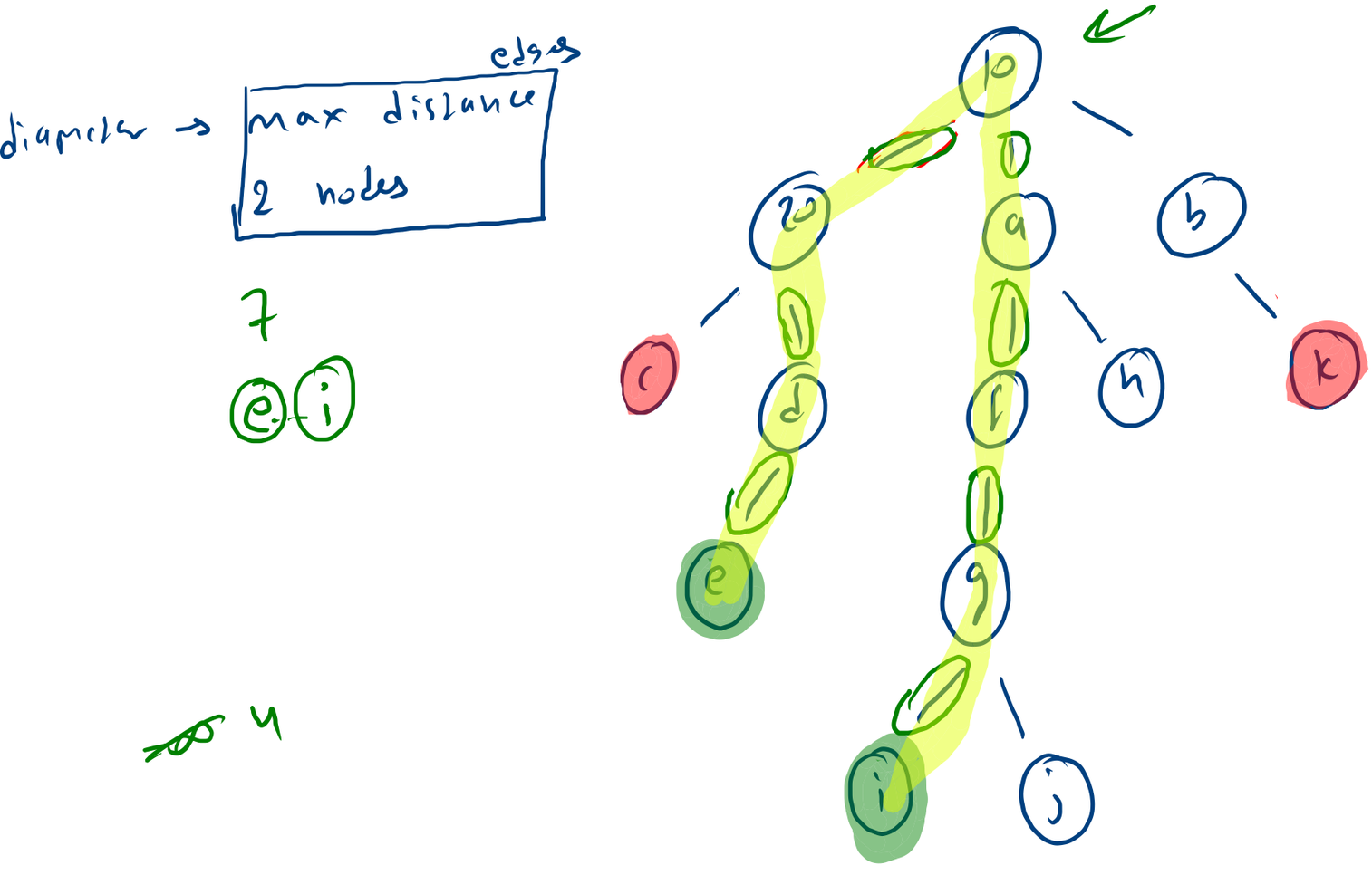
```java
static int maxsum = Integer.MIN_VALUE;
static int maxdata = 0;
public static int maxST(Node root){

    int sum = root.data;
    for(Node child: root.children){
        sum += maxST(child);
    }

    if(sum > maxsum){
        maxsum = sum;
        maxdata = root.data;
    }

    return sum;
}
```

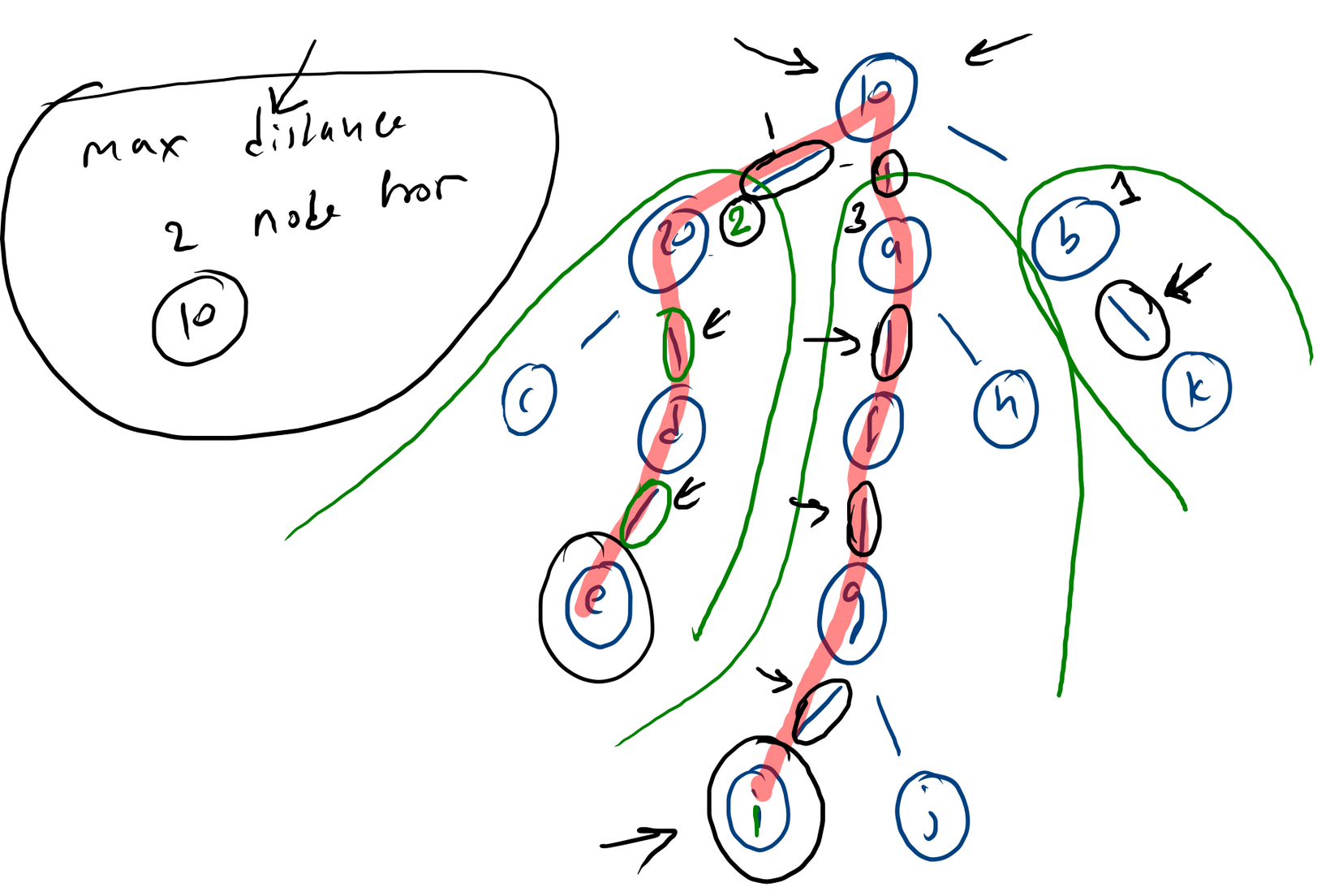




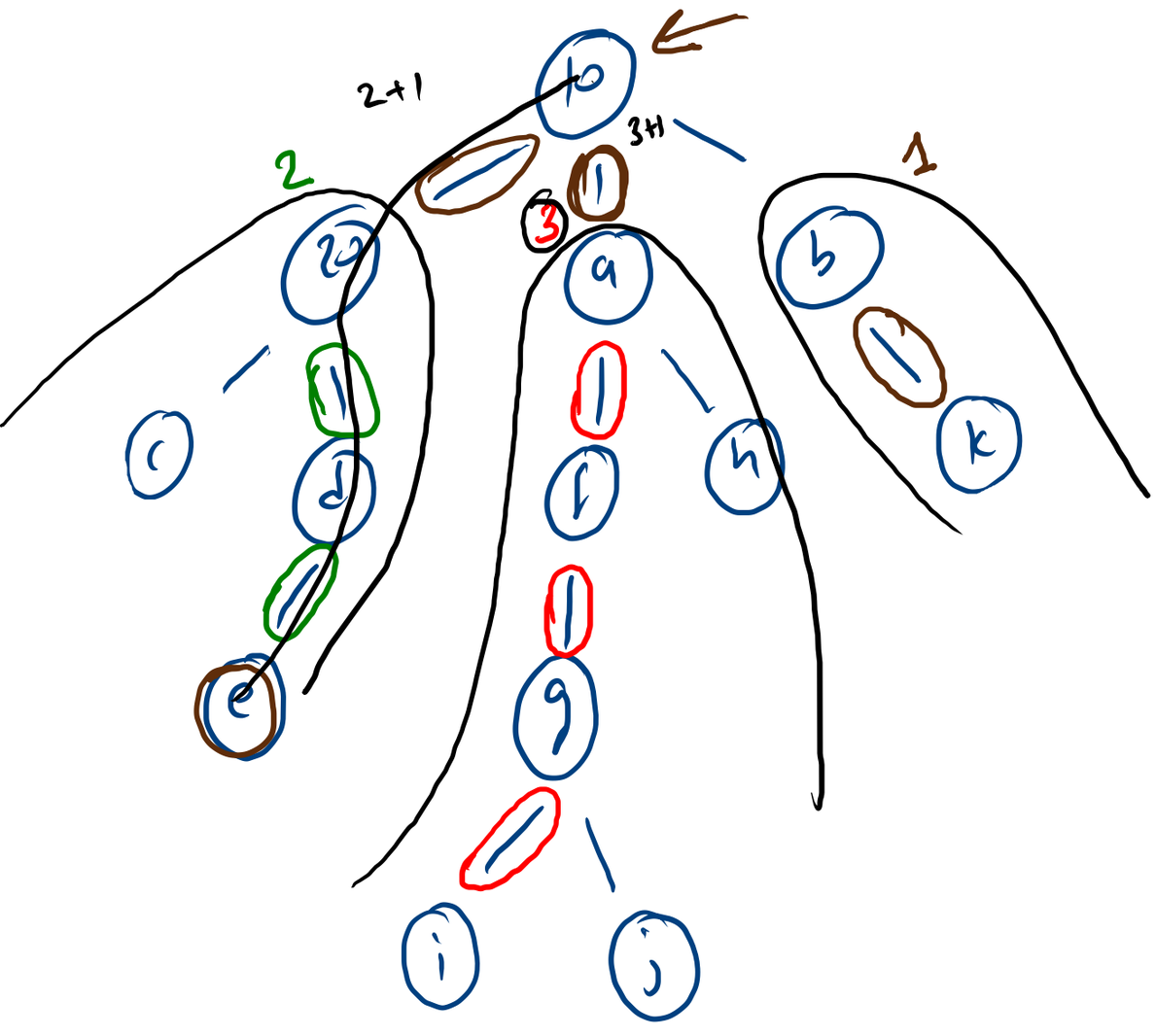


-50 +10


edges
diameter →
max distance
2 nodes
7
c
d
e
i
10
a
b
h
k
9
j
c ... k → 4
e ... i → 7
10
diameter → 0
4



3, 2 +2



h1 h2

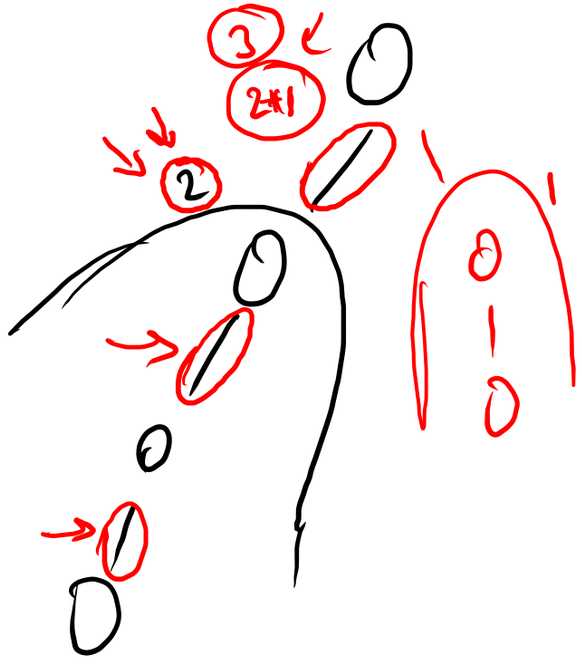
3 , 2

3 + 2 + 2 = 7

2+1 +3+1

max h1 + max h2 + 2

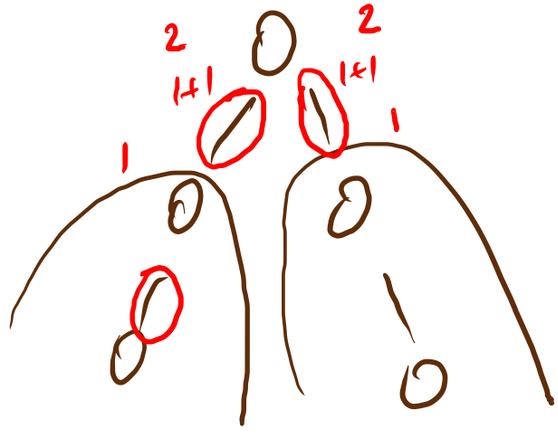
height



h1 = ~~0~~ 3

h2 = ~~0~~ 2

h1+h2 = 3+0 = 3



h1 ~~0~~ 2

h2 0 2

h1+h2 = max distance

dia = 0 + 2 6

```
static int dia=0;
public static int diameter(Node node){

    int h1 = 0;
    int h2 = 0;

    for(Node child: node.children){
        int h = diameter(child) + 1;

        if(h > h1){
            h2 = h1;
            h1 = h;
        }else if(h > h2){
            h2 = h;
        }
    }
    if(h1+h2 > dia){
        dia = h1+h2;
    }
    return h1;
}
```

h1 0 + 4

h2 0 0 1

dvertice

diameter

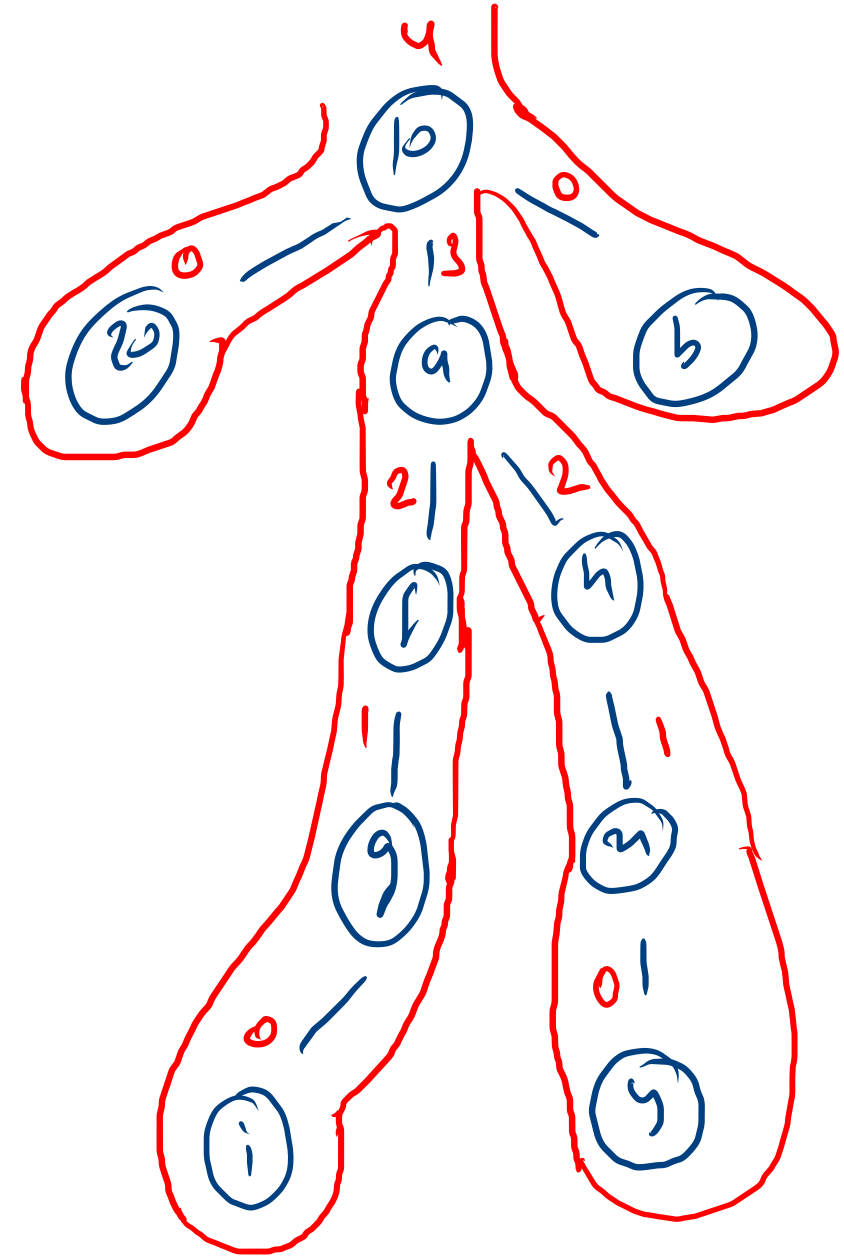
expectation

height

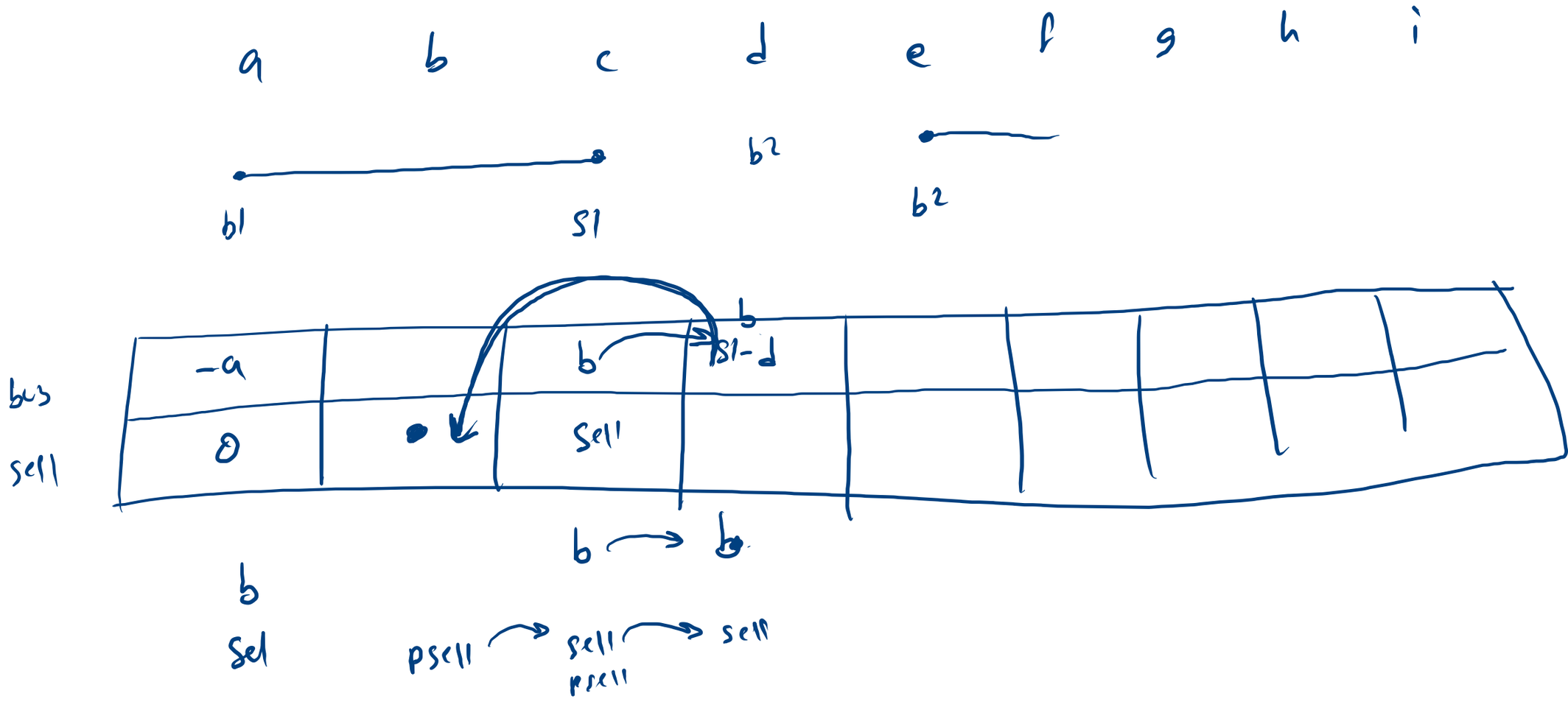
diameter update

faith

height



a b c


a
b
c
d
e
f
g
h
i
b2
b2
b1
s1
b
b
s1-d
-a
bus
0
sell
sell
b
b
b
sel
psell
sell
psell
sell