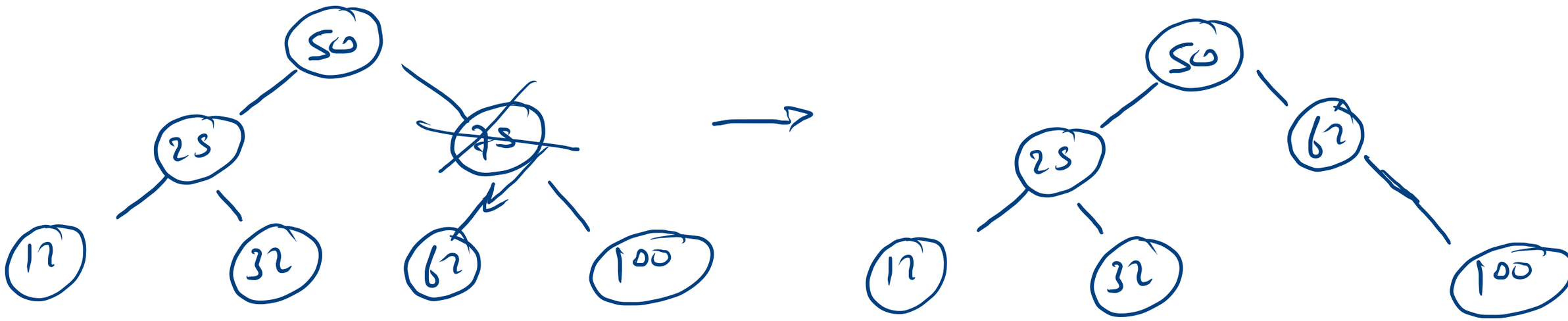
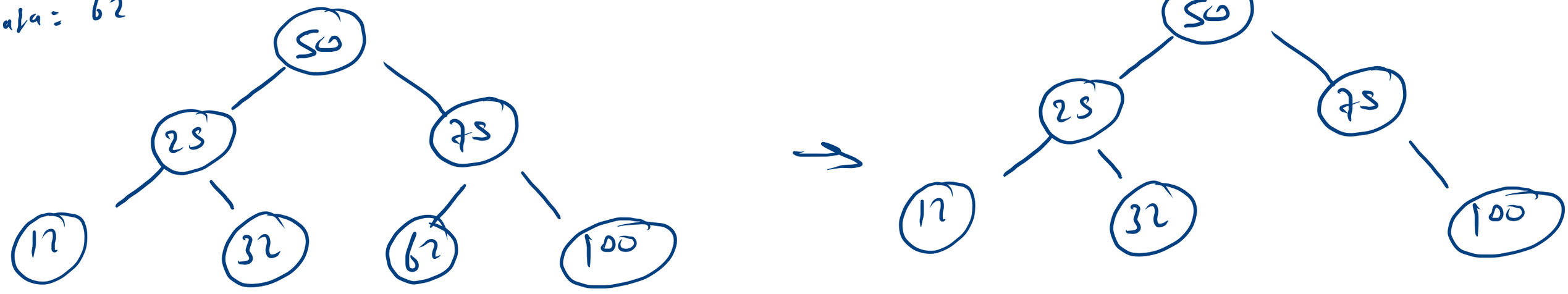
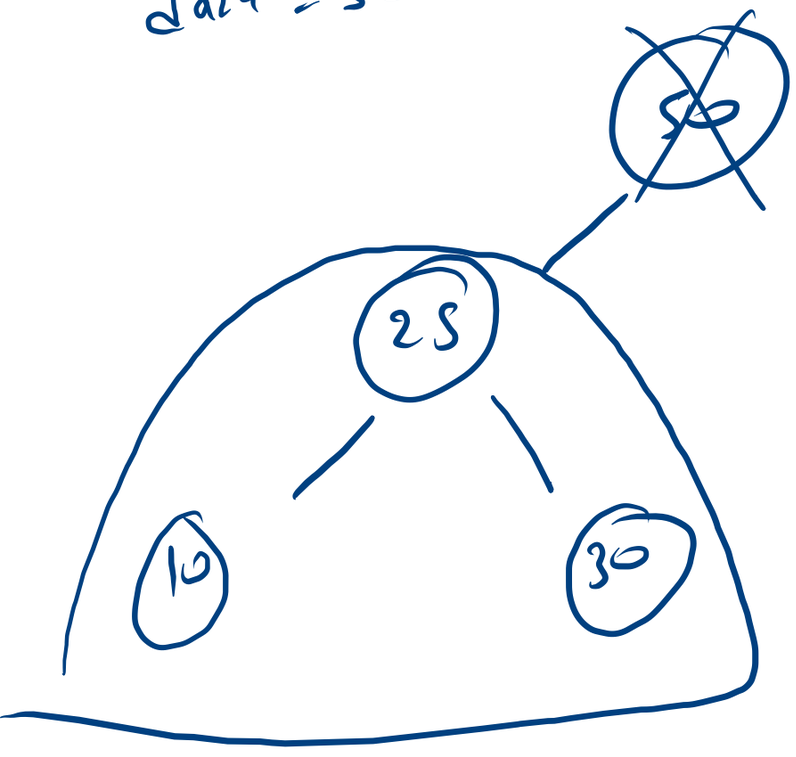
data -> 75

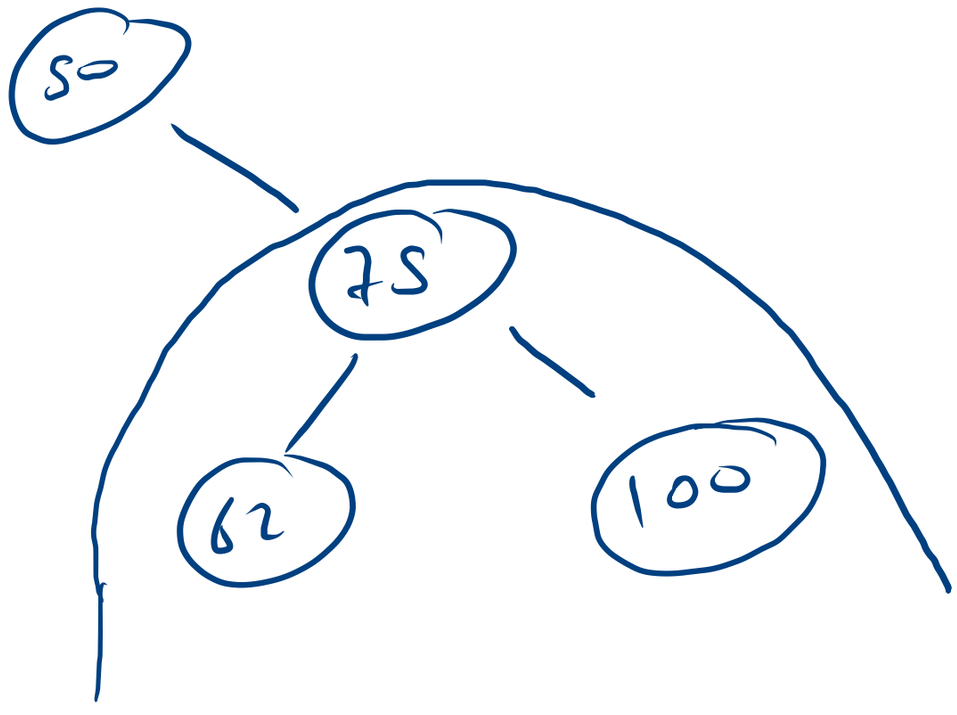


data: 62

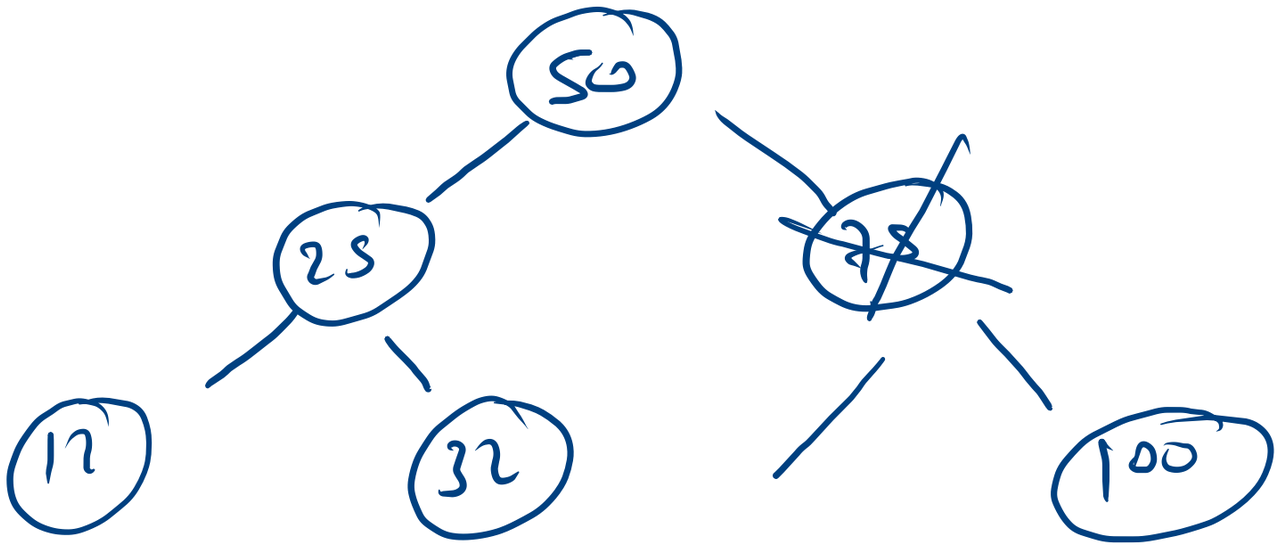


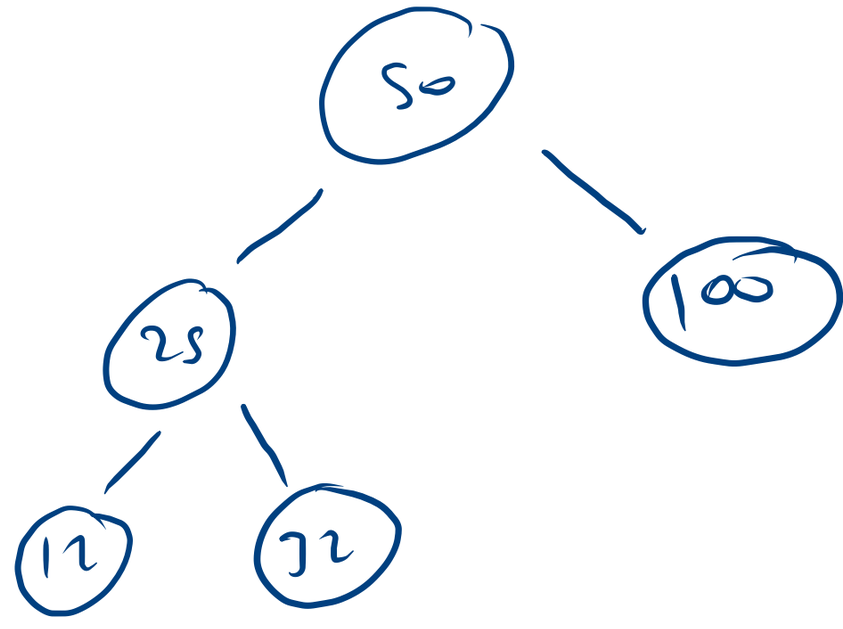
data = 50

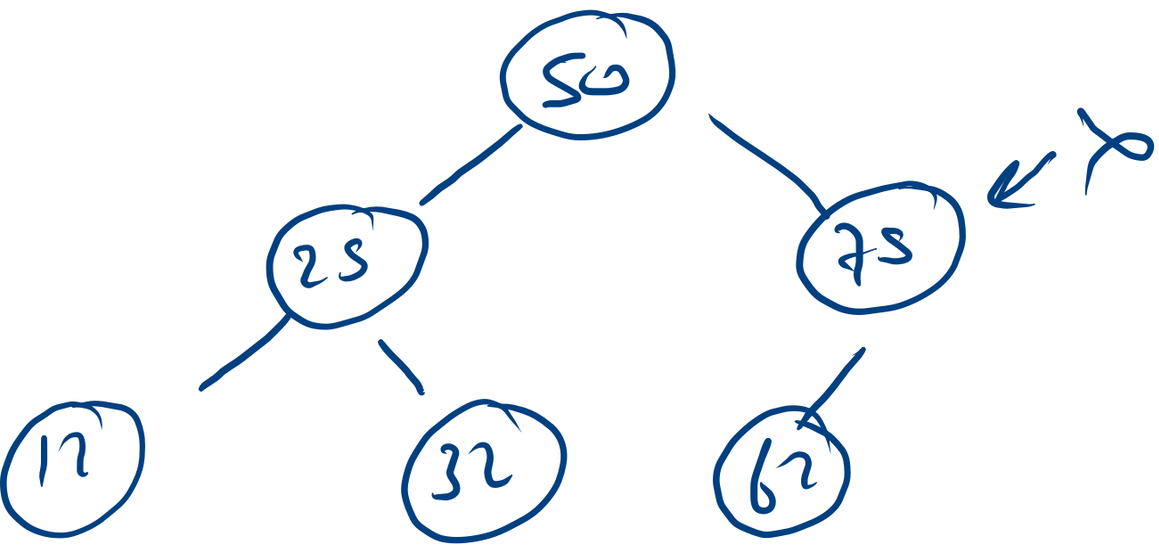


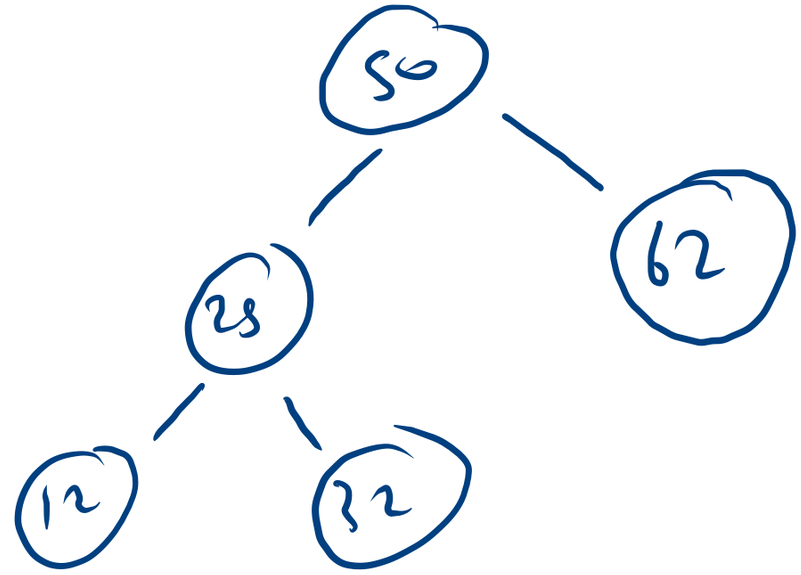


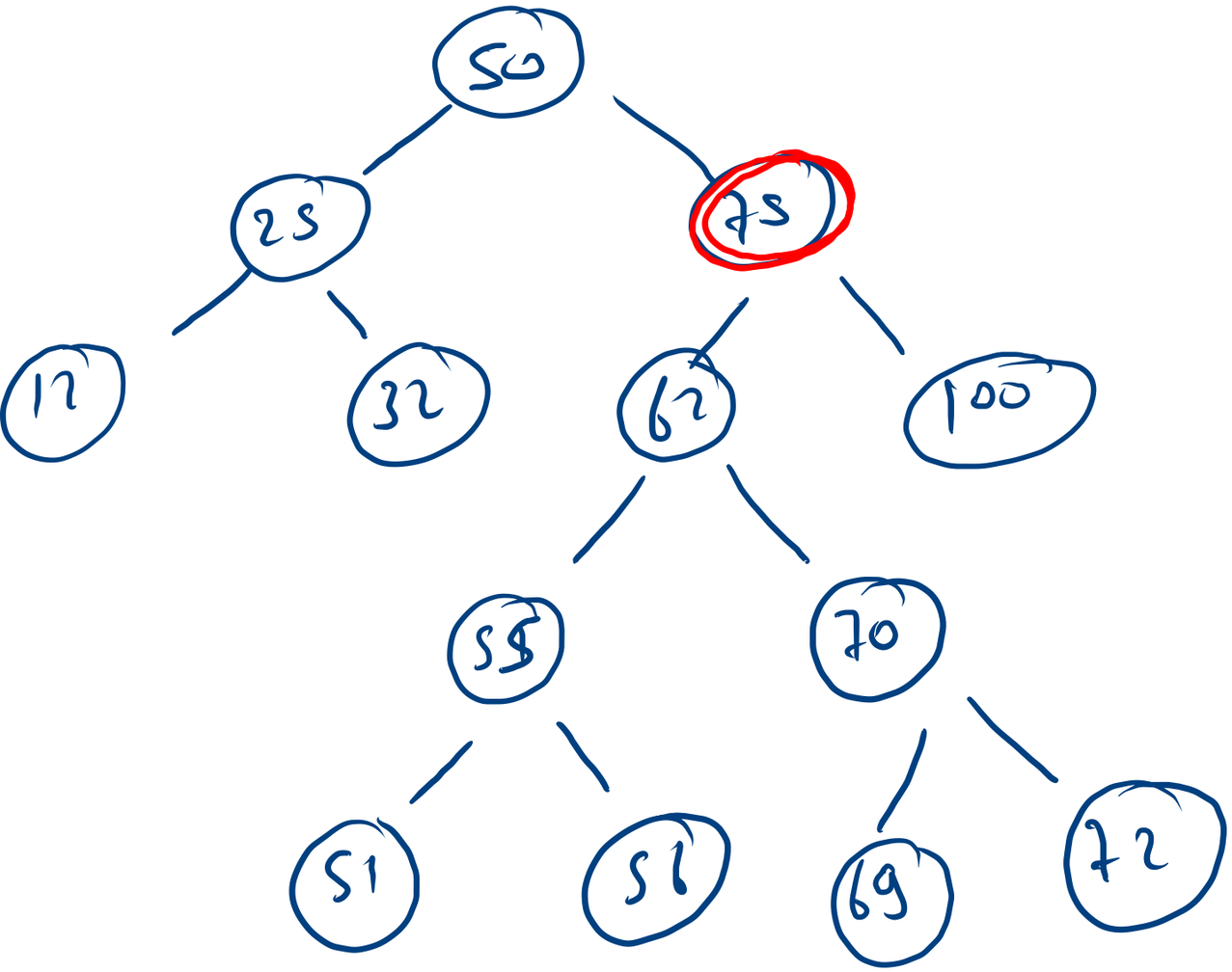
→ null

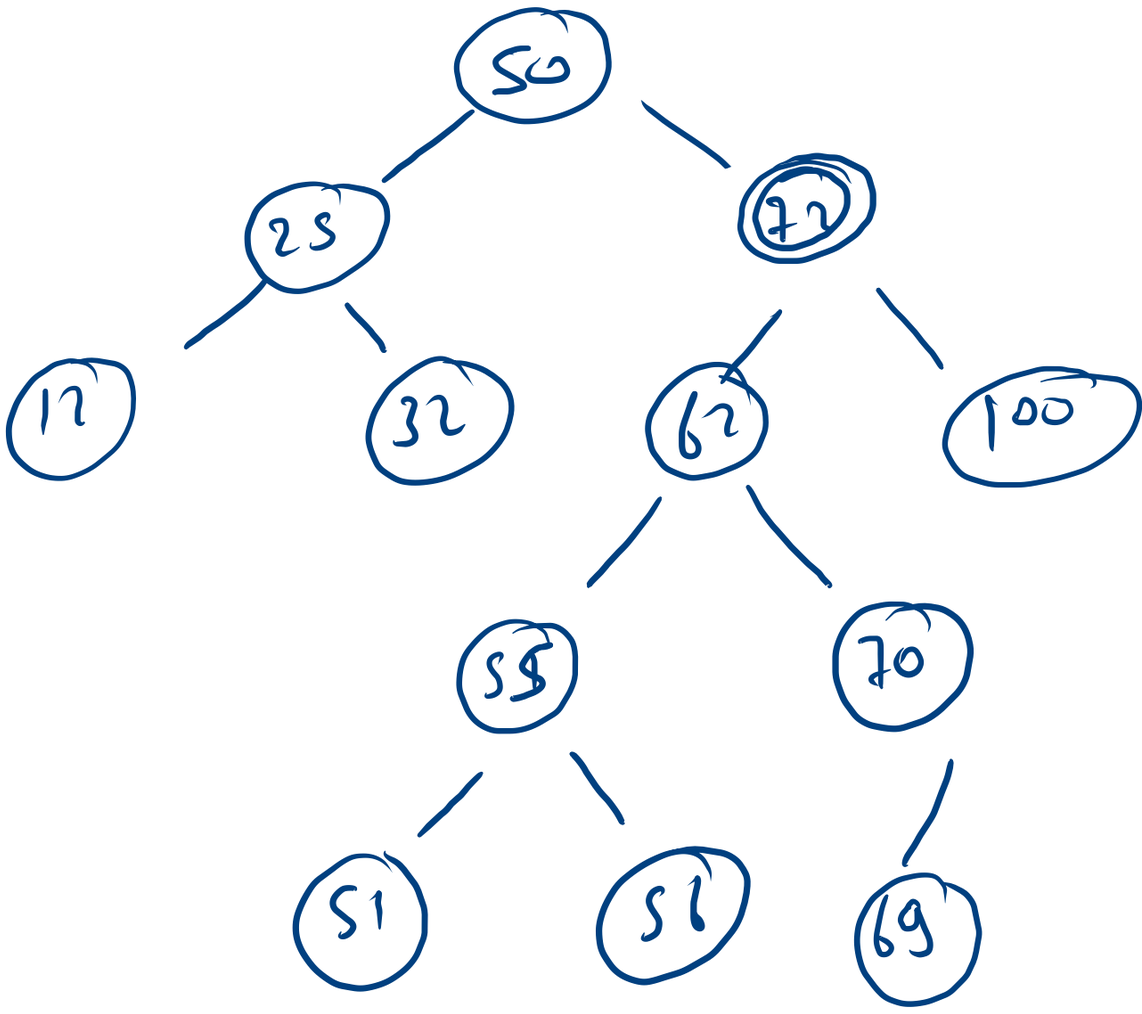

50
25
75
17
32
100

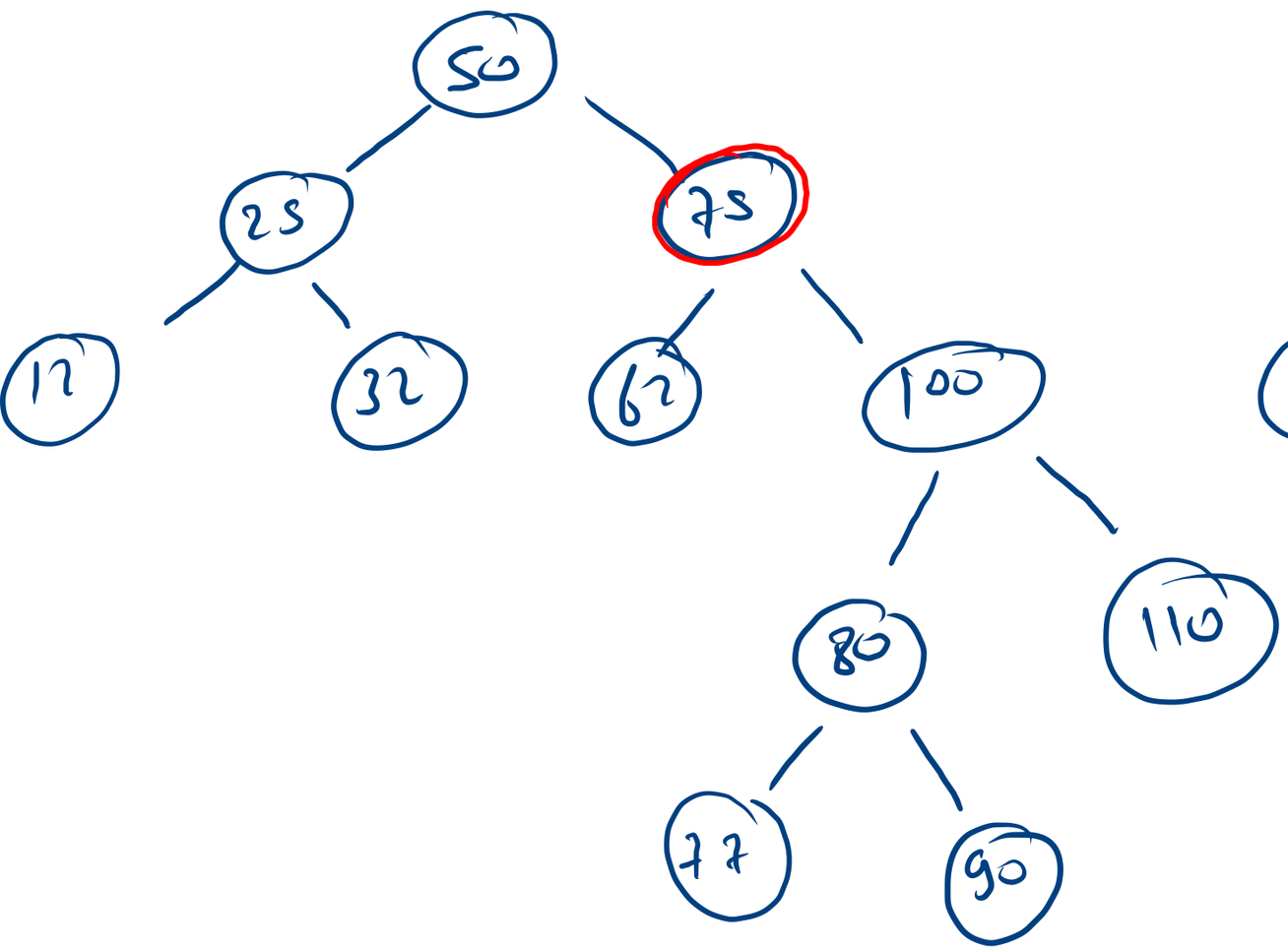

50
25
100
17
32

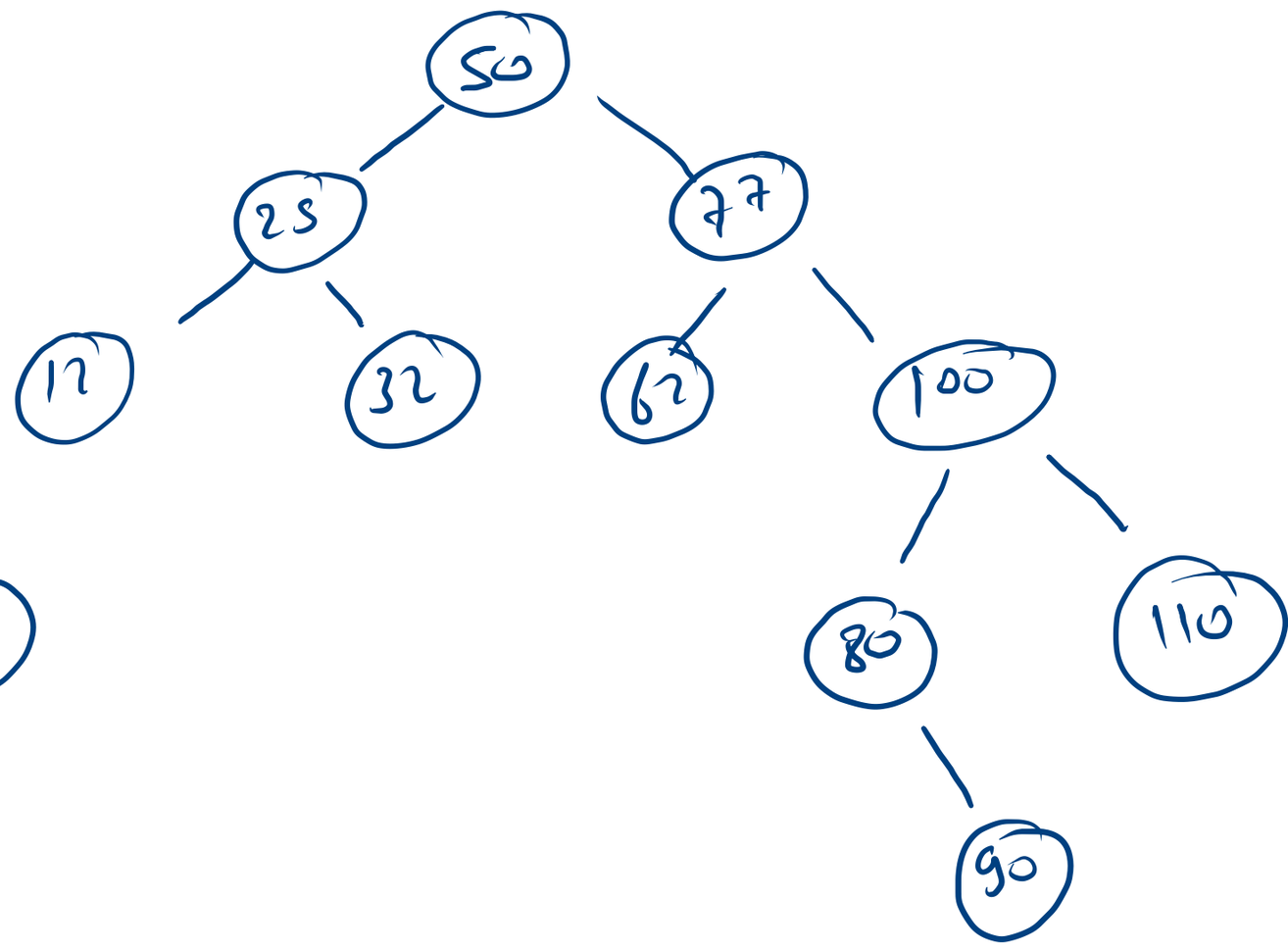

50
25
75
17
32
67


50
25
67
17
32


50
25
75
17
32
67
100
55
70
51
51
69
72


50
25
72
17
32
67
100
55
70
51
51
69


50
25
75
17
32
67
100
80
110
77
90


50
25
77
17
32
67
100
80
110
90

data = 50

50

X X

→ null child

50

25

X

17 32

→

25

17 32

only left child
only right child

both left and right

50

77K

65

SK 62 100 6K

55 90 110

50

uK 77 K

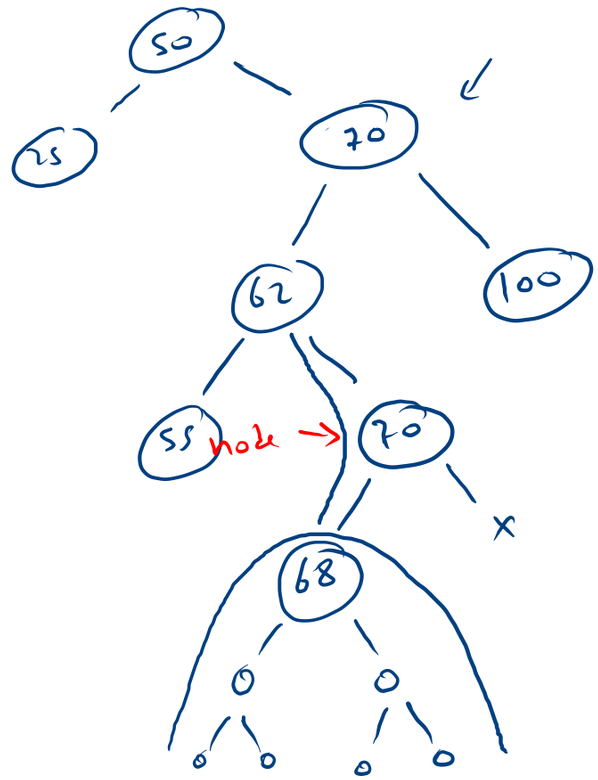
SK 62 6K 100

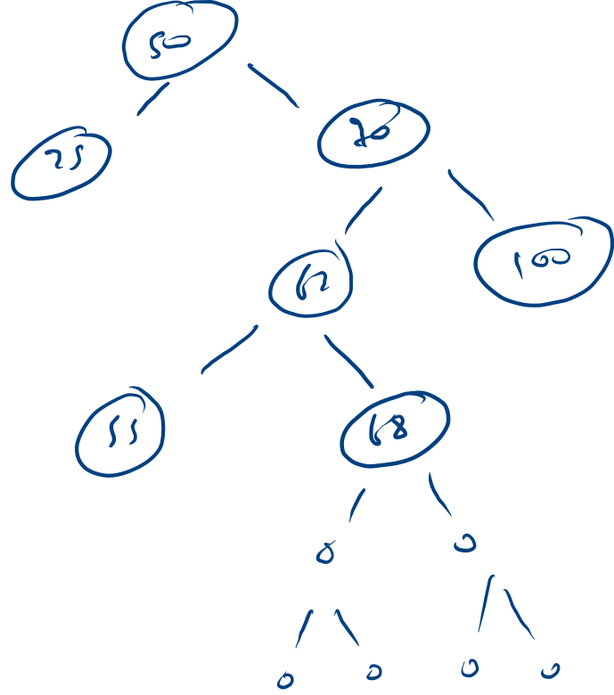
55 7K 65 90 110

→

70

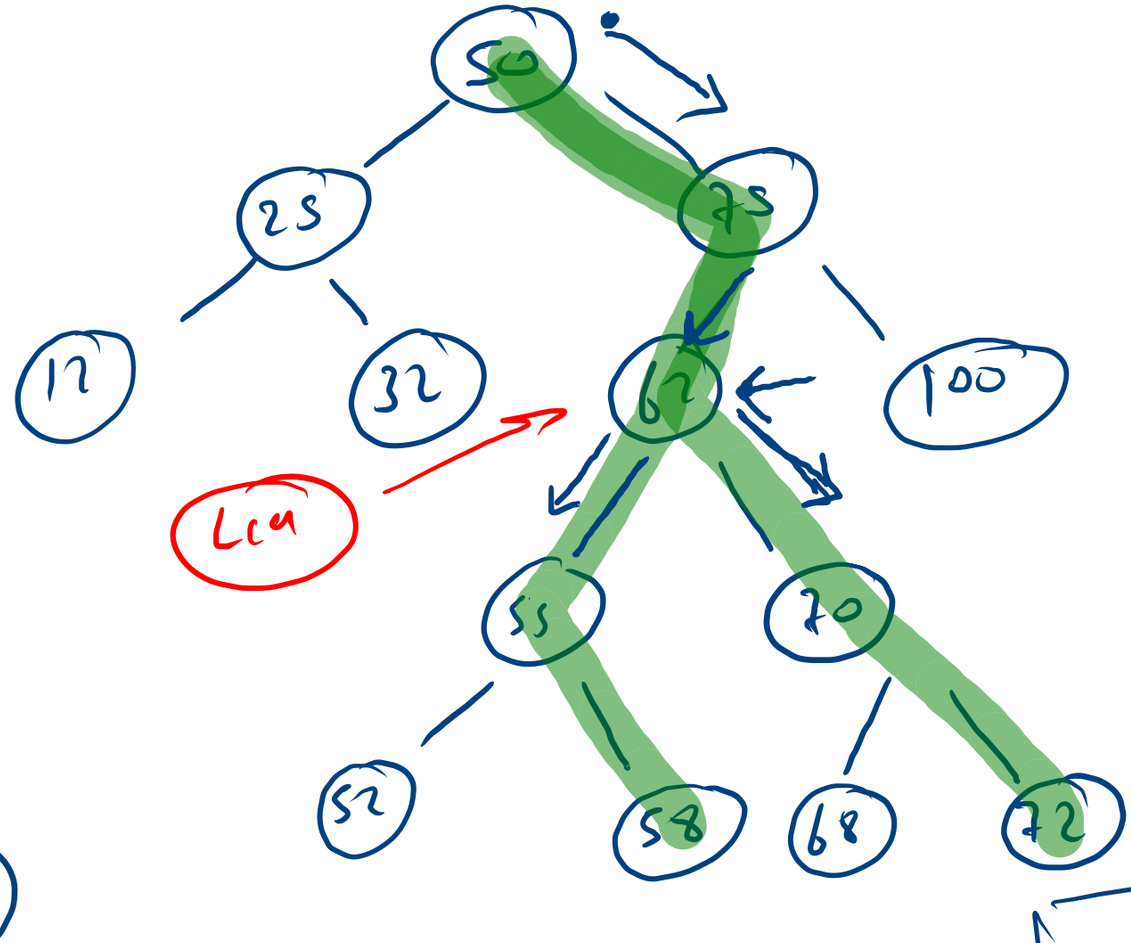
```java
public static Node remove(Node node, int data) {

    if(node.data < data){
        node.right = remove(node.right, data);
        return node;
    }
    else if(data < node.data){
        node.left = remove(node.left, data);
        return node;
    }else{
        if(node.left == null && node.right == null){
            return null;
        }else if(node.left != null && node.right == null){
            return node.left;
        }else if(node.left == null && node.right != null){
            return node.right;
        }else{
            int max = max(node.left);
            node.data = max;
            node.left = remove(node.left, max);
            return node;
        }
    }
```

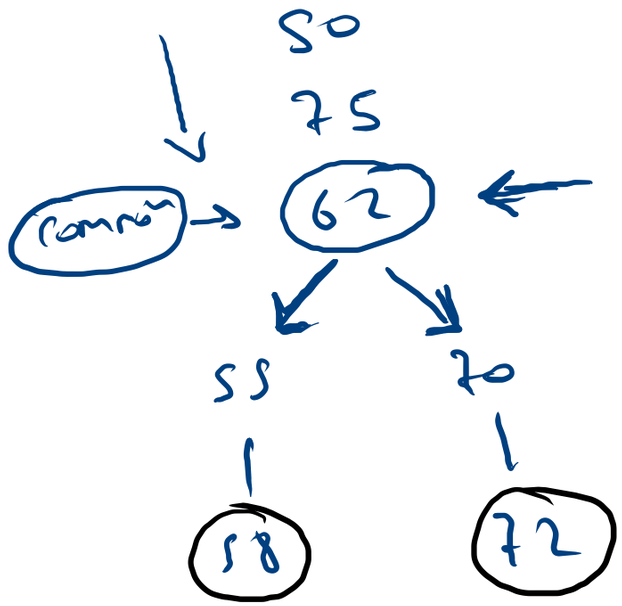


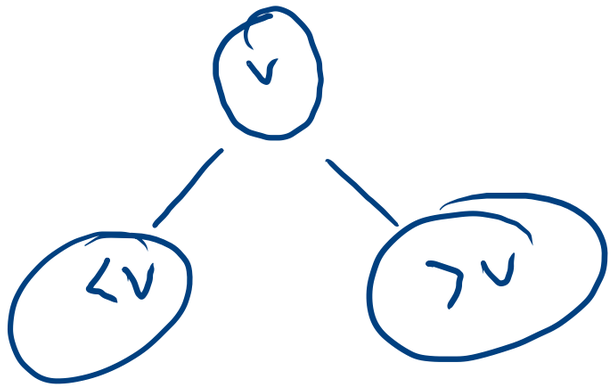


data1 = 58

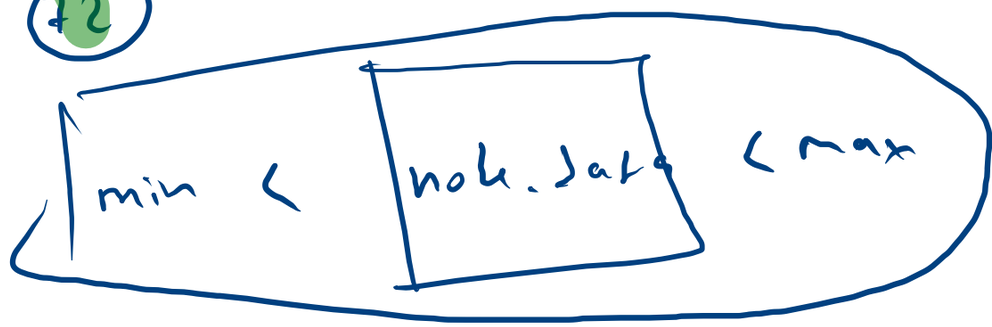
data2 = 72







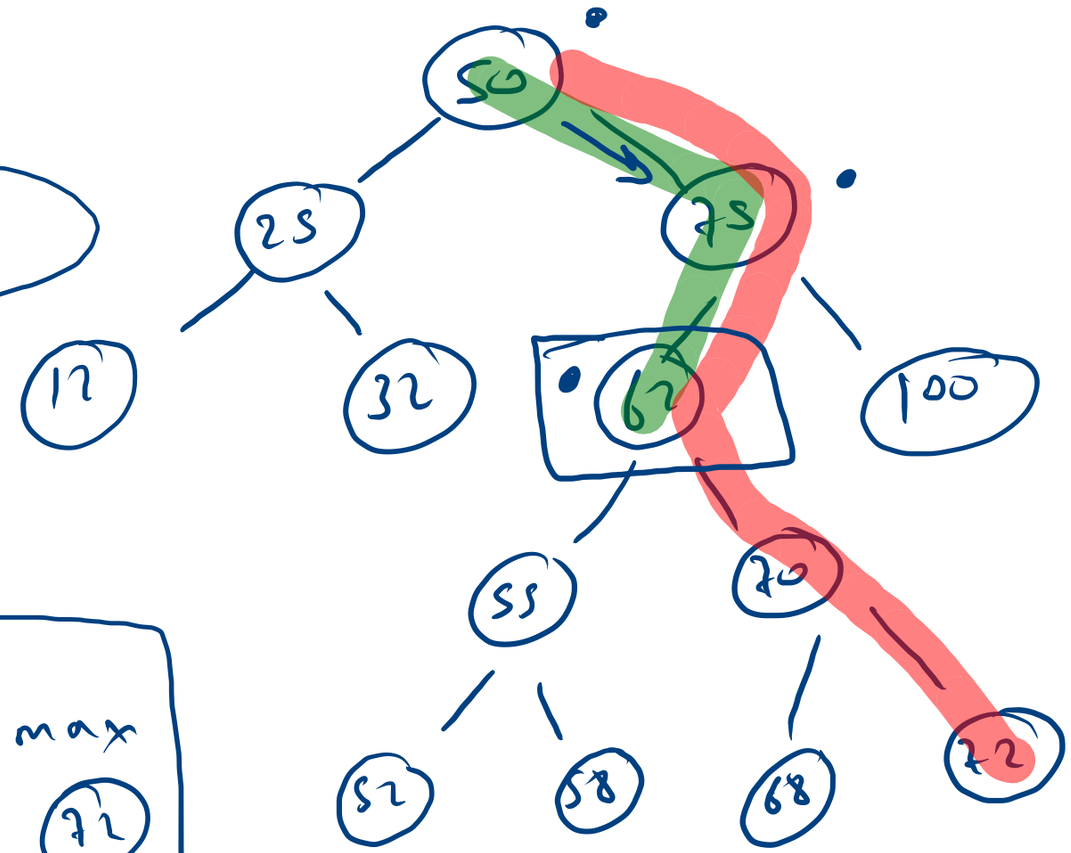
find v

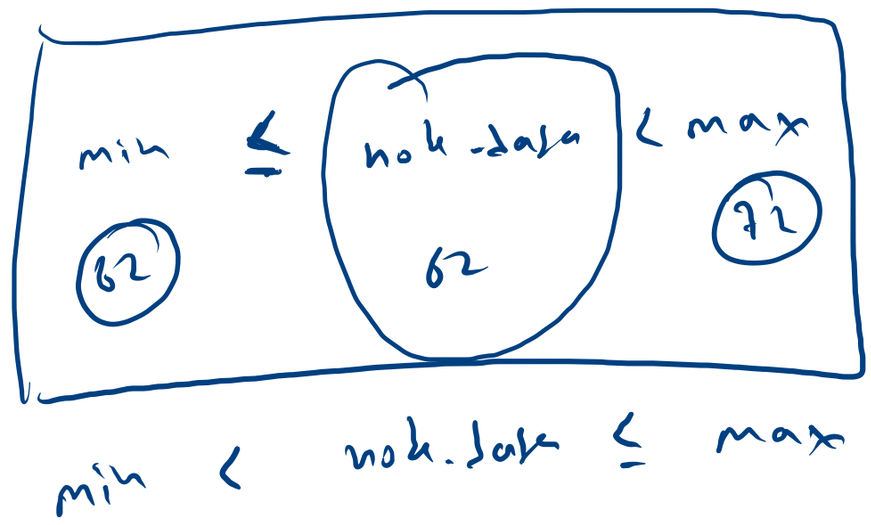


data1 = 62

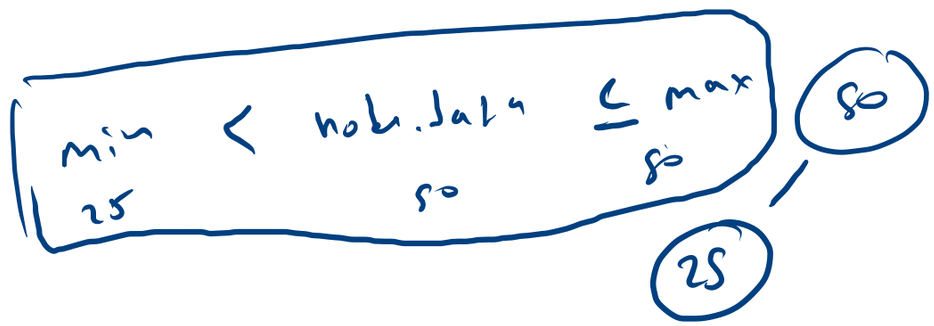
data2 = 72

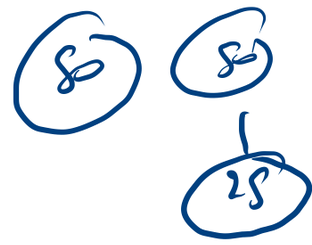
min ≤ nok.data ≤ max

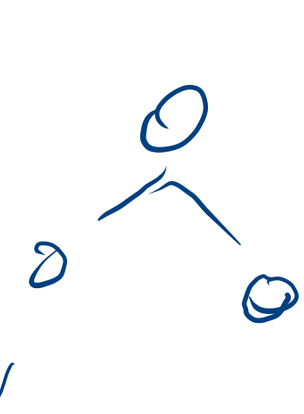




min < nok.data ≤ max



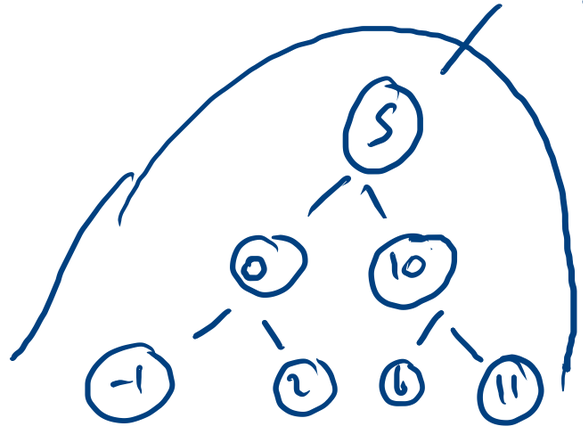




data

	BT	BST
size	•	•
sum	•	•
max	•	✓
min	•	✓
find	•	✓

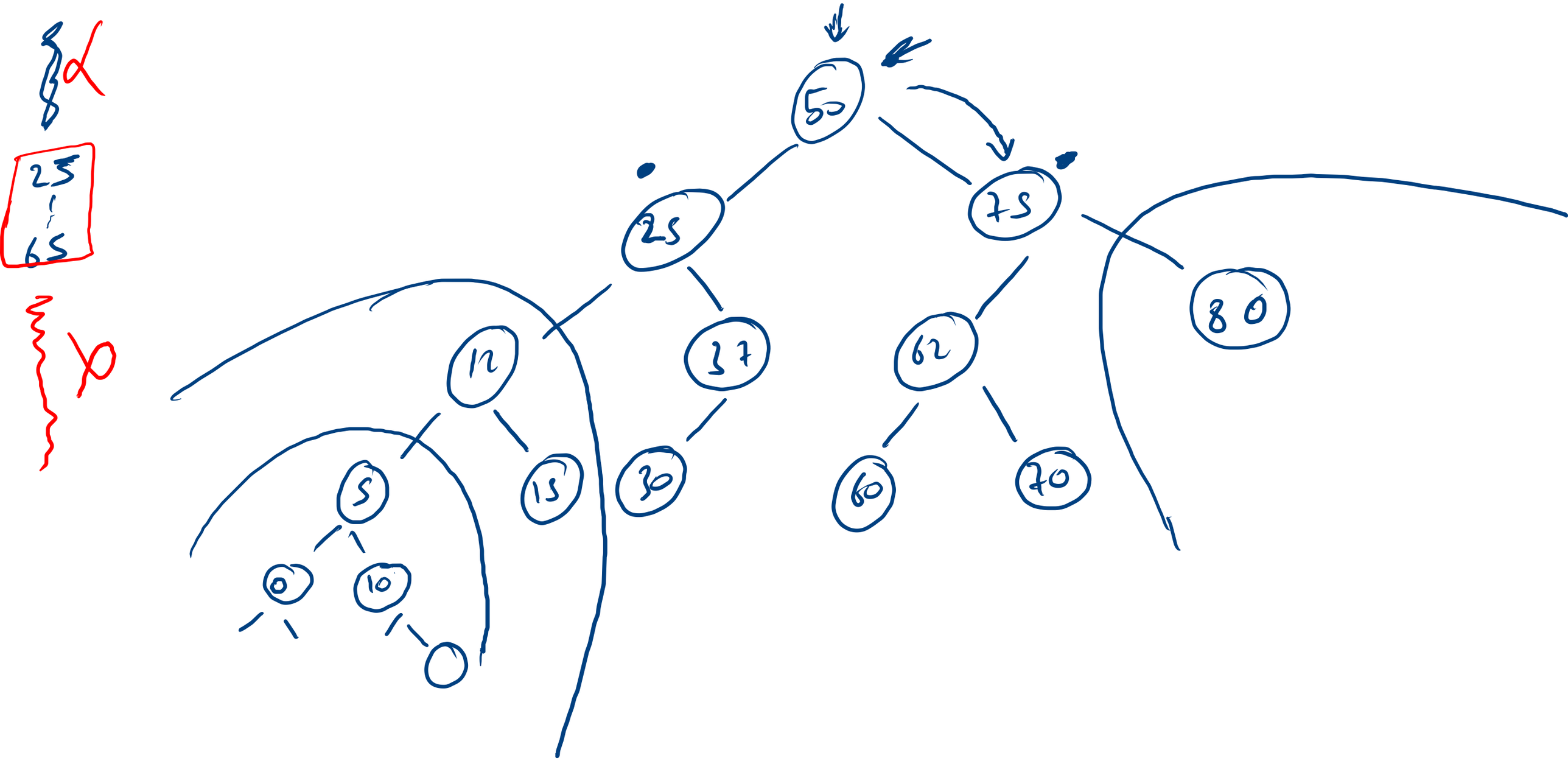
d1 → 12
d2 → 65



inorder → ascending order

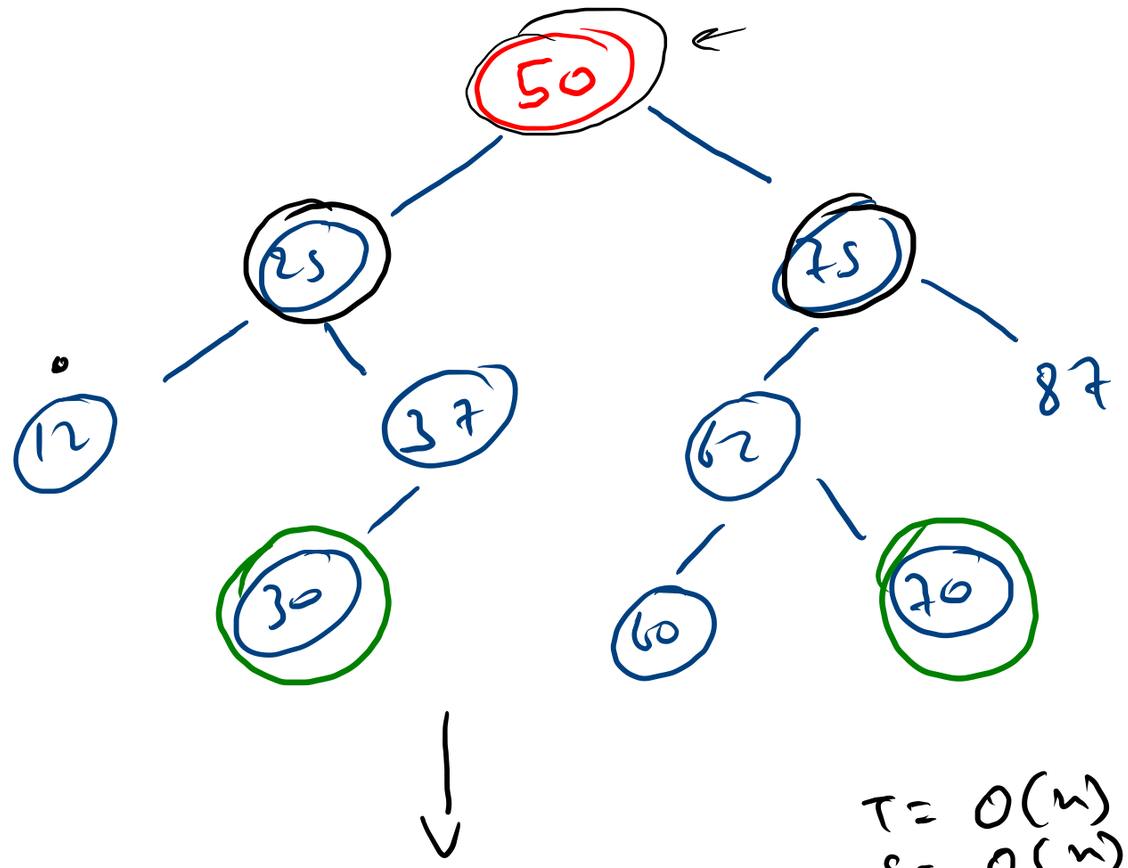
12
25
30
37
50
60
62

5 12 15 25 30 37 50 60 62 70 75 87


x
25
65
y
50
25
75
12
37
62
80
5
15
30
60
70
0
10

target → 100

a + b = target



30 + 70 = 100

25 + 75 = 100

30	70
25	75

★ 12 + [?] = 100

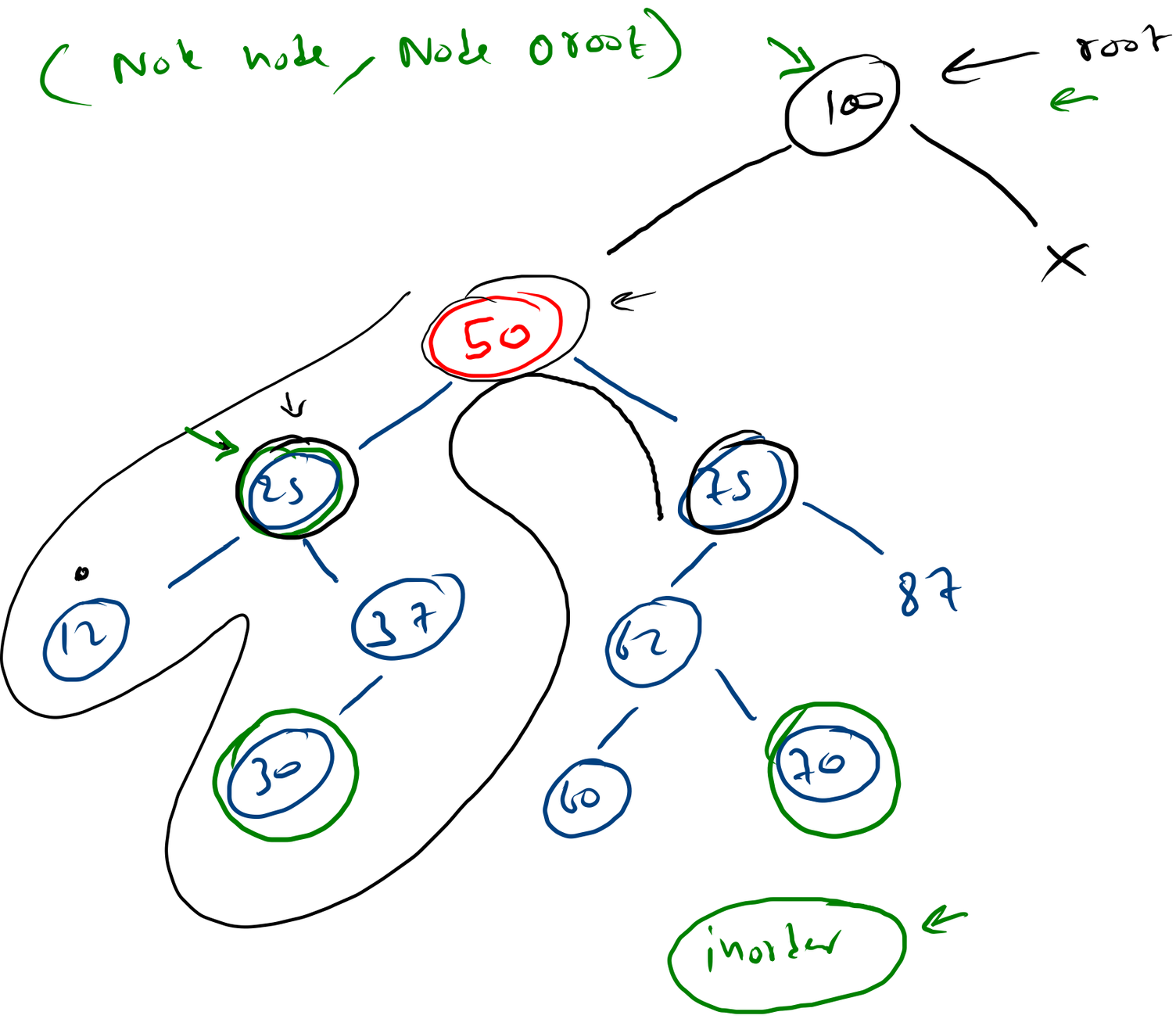
comp = 100 - 12

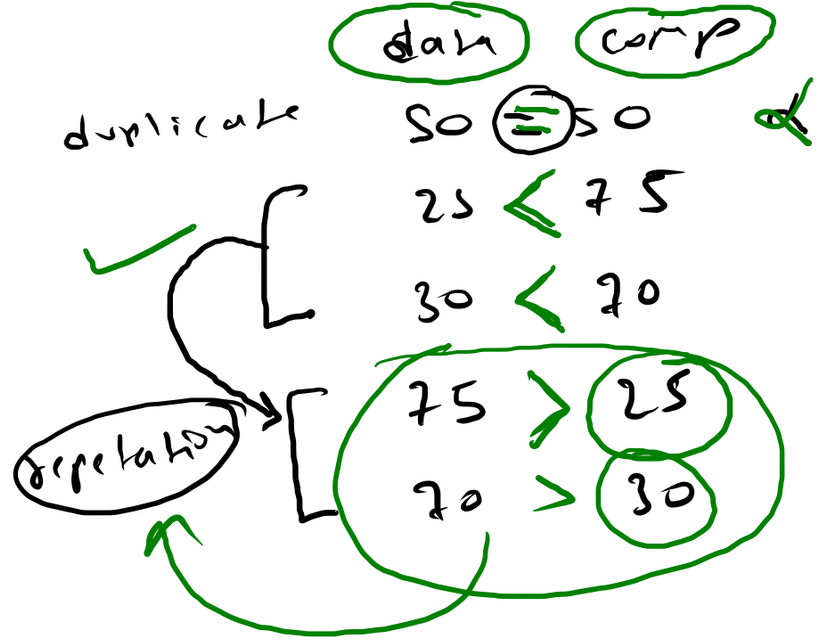
88

T = O(n)
S = O(n)

Analisi ✓

(Node node, Node Oroot)





[travel all nodes]

comp = tar- node.data

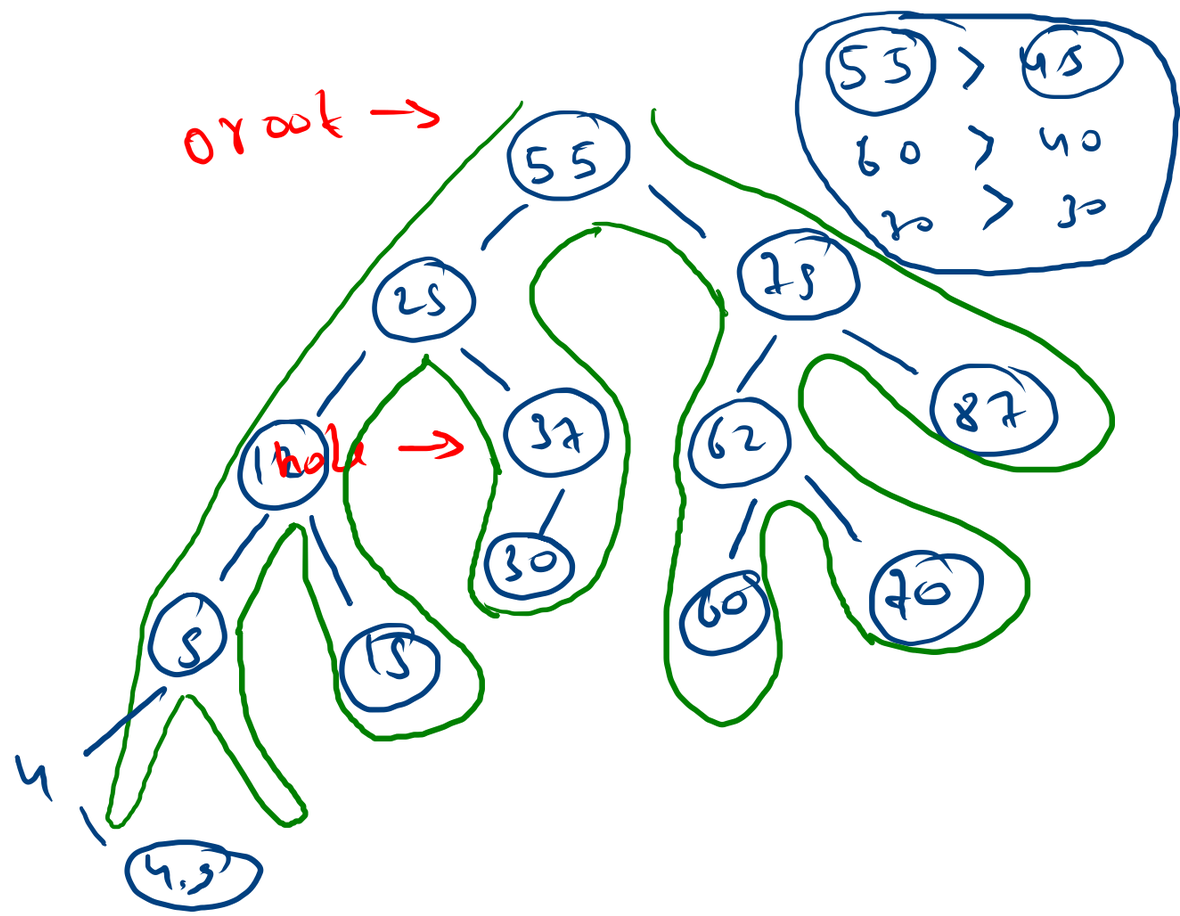
if(node.dat => comp)
X
else{ find
y print

5 + 90 < 100

data + max < target

target → 100

```java
public static void tarsum(Node node, Node oroot, int target){
    if(node == null)return;

    tarsum(node.left, oroot, target);

    int comp = target-node.data;
    if(node.data < comp){
        if(find(oroot, comp)){
            System.out.println(node.data+" "+comp);
        }
    }

    tarsum(node.right, oroot, target);
}
```

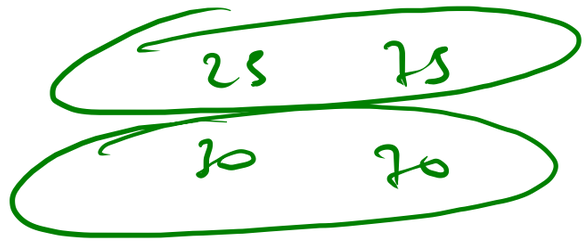


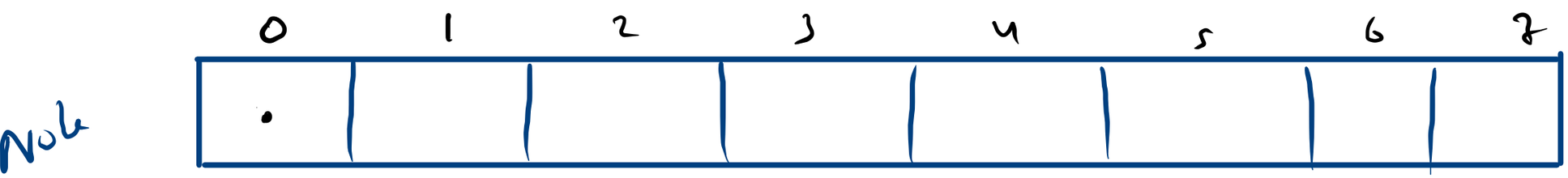
75 < 25

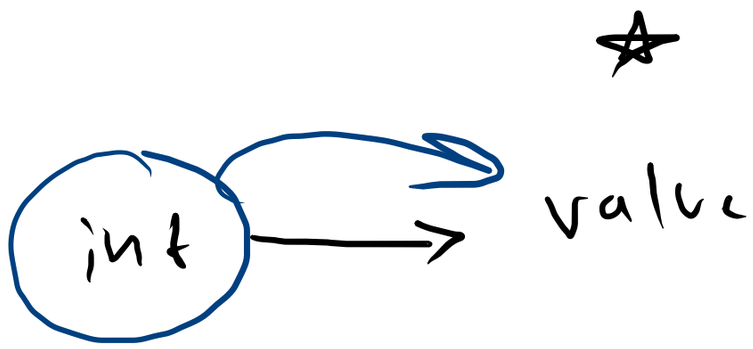
20 < 30

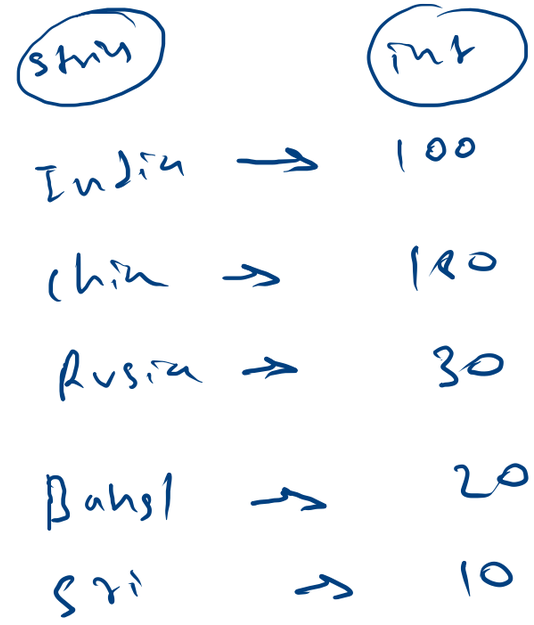
com = 23

30









int → string

str → string

Ind → Delhi

usa → 2yn

abc → cba

0	1	2	3	4	5
4	5	6	7	10	3

ArrayList<String>
ArrayList<Node>

int → Node

int → string

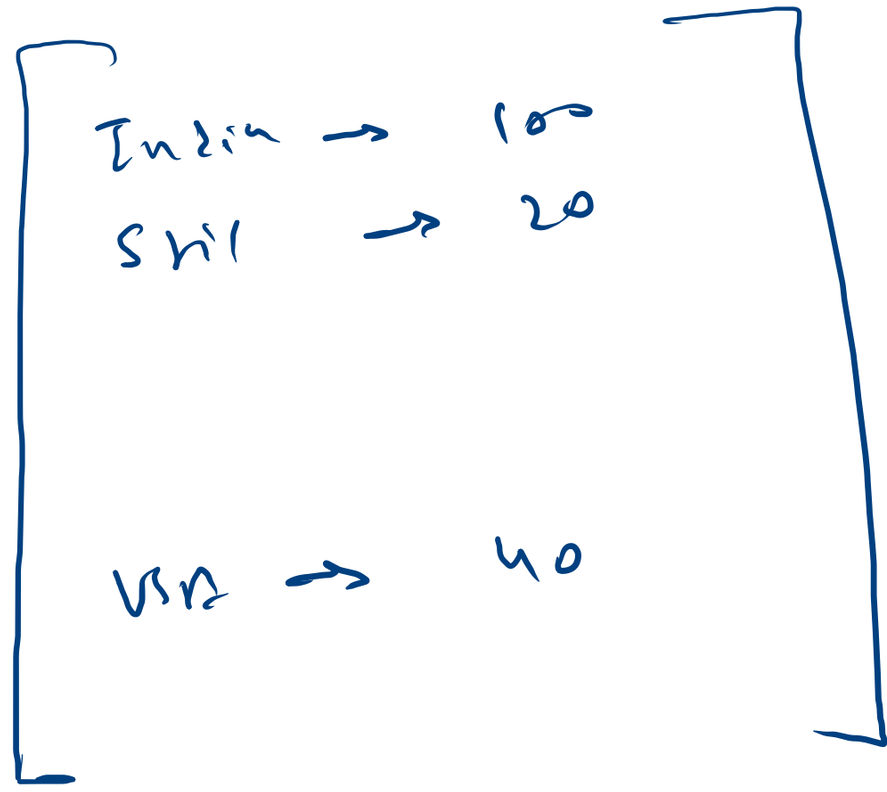
HashMap<String, Integer>

O(1)

put	O(1)
get	O(1)
keySet	O(n)
Size	O(1)
containsKey(key)	

String → int

Indian → 100



zmszeqxllzvheqwr

key → value

Character → Integer

z	3
m	1
s	1
e	2
q	2
x	1
l	2
v	1

h → 1

w 1

r → 1

z