Binary Search Tree data structure.

A type of binary tree. Organaized in a sequential way.

why we lean this?

Ordered structure

Efficient searching.

Efficient insertion and deletion.

balance vs un balanced tree.

Memory efficiency.

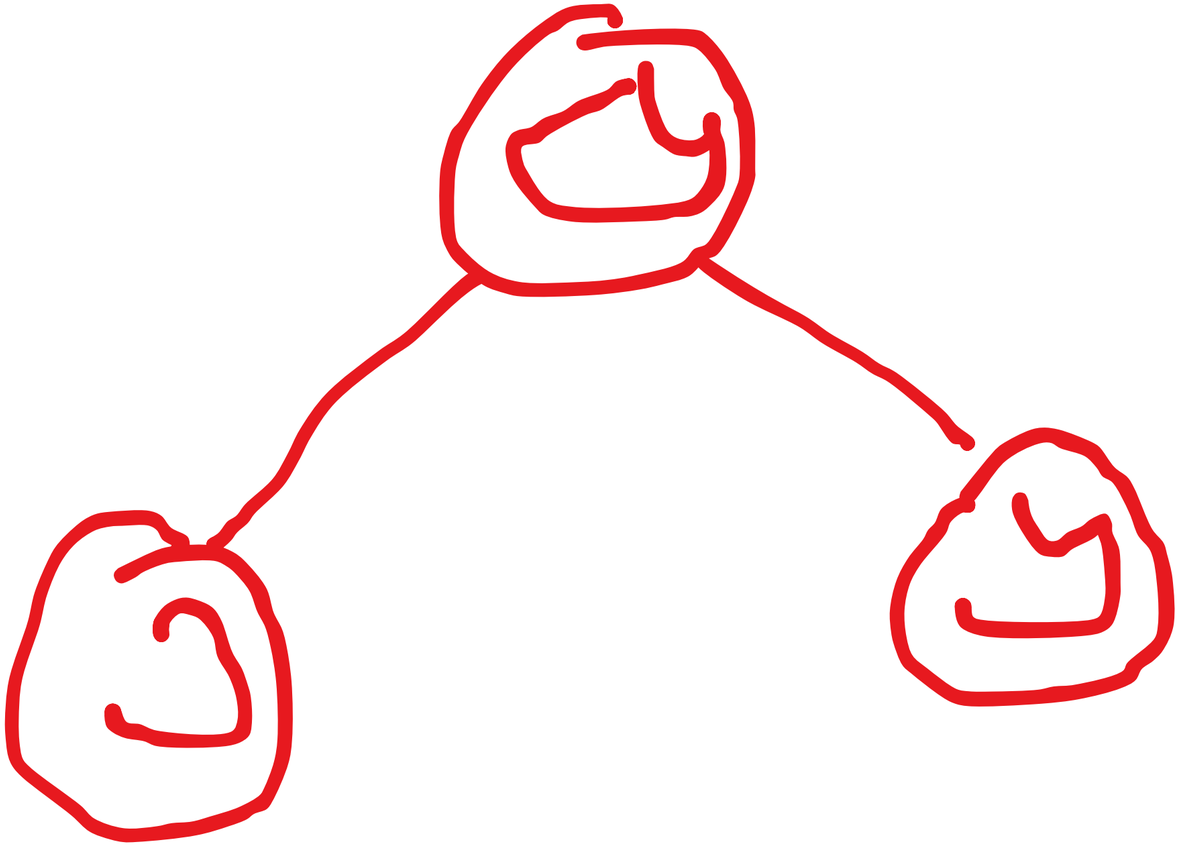
Versatile Applications

properties

each node has two child, one is left and other is right.

Left child is always smaller than parent.

right child is always greater than parent node.



addition operations

find a null node?

create a new

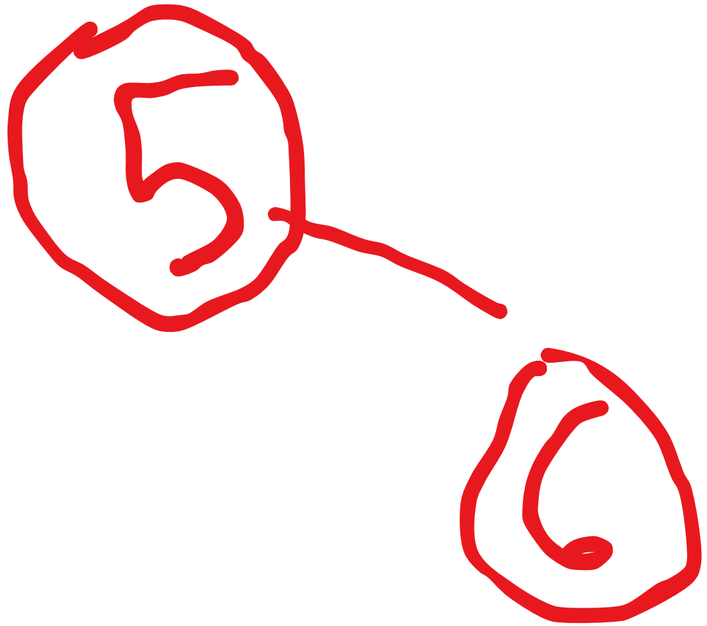
node that time.

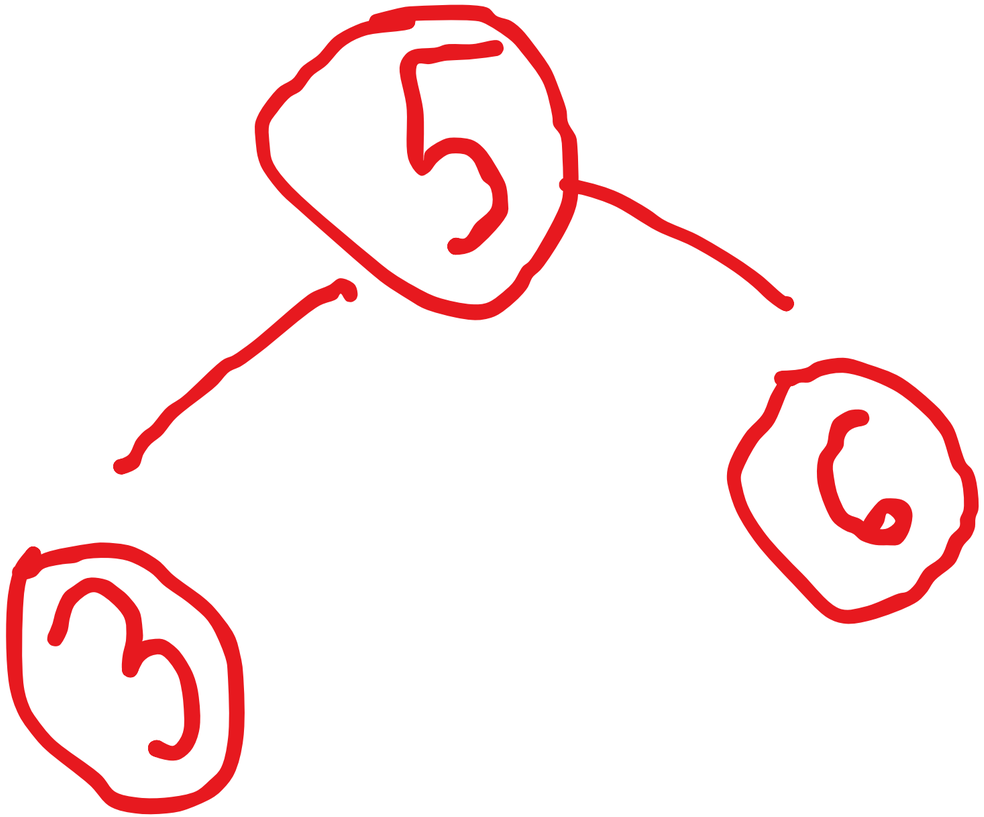
our inserted data is less than the node value-> left node.

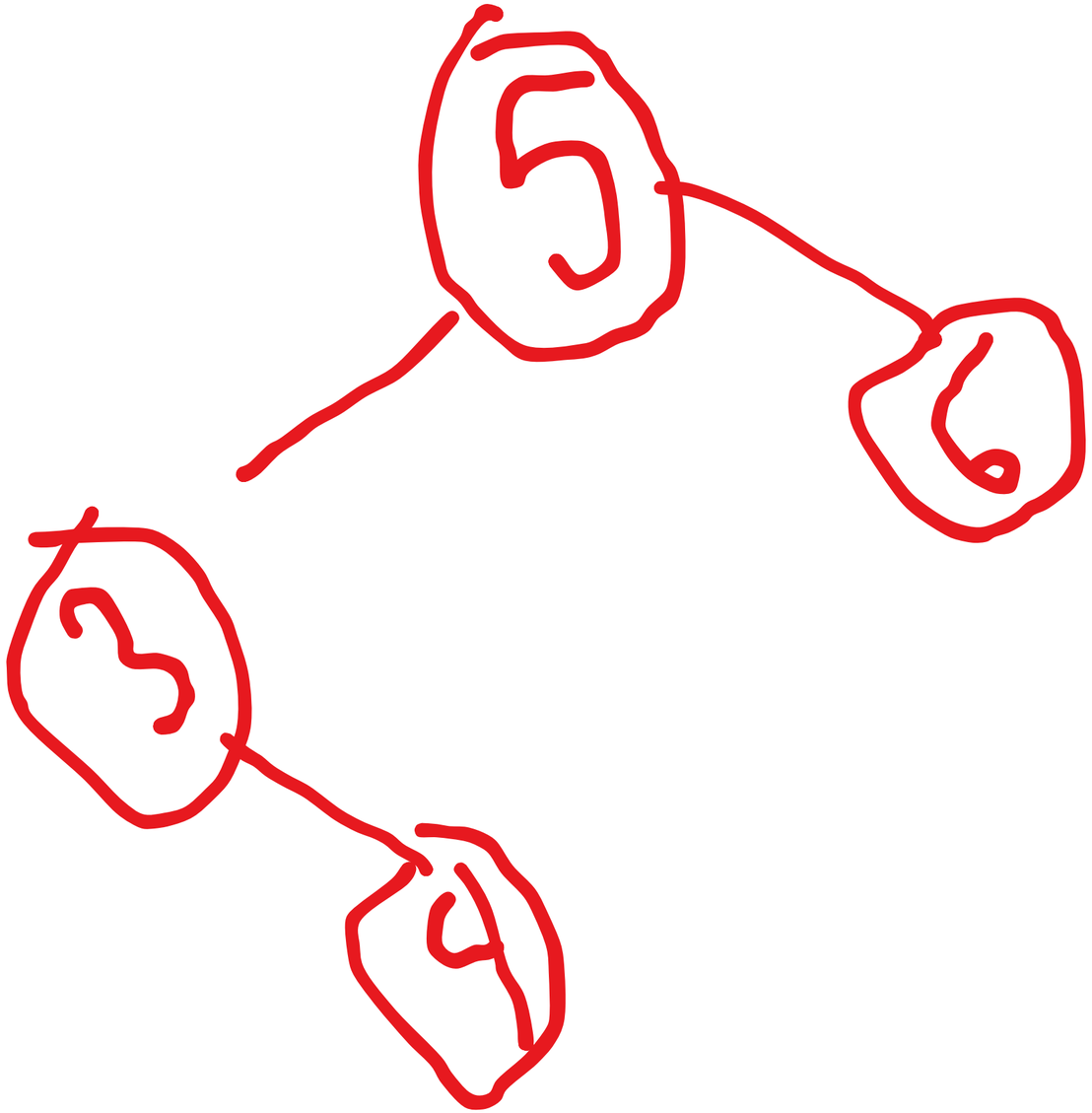
our inserted data is greater the node data go right.

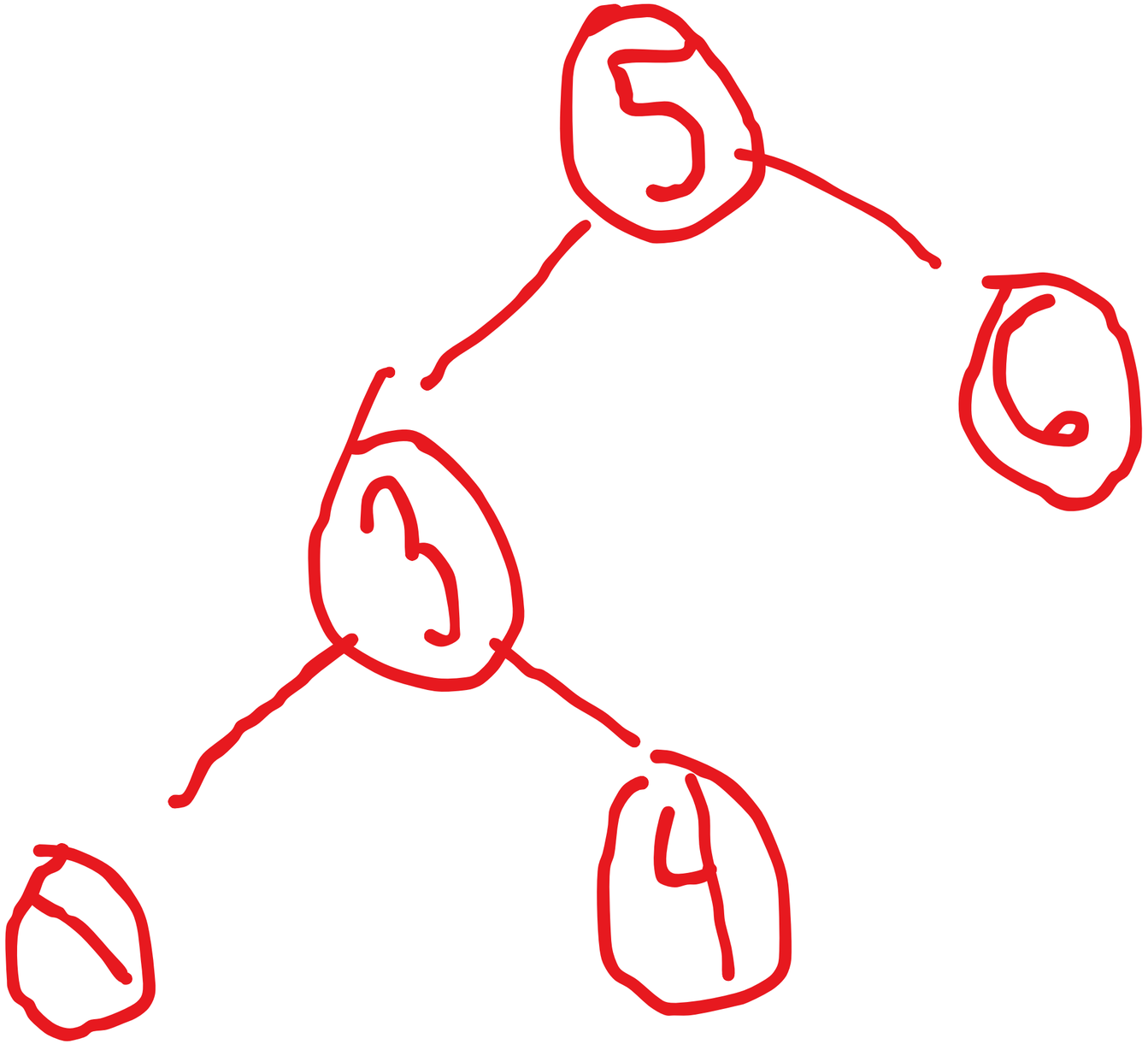
Let’s solve an example.

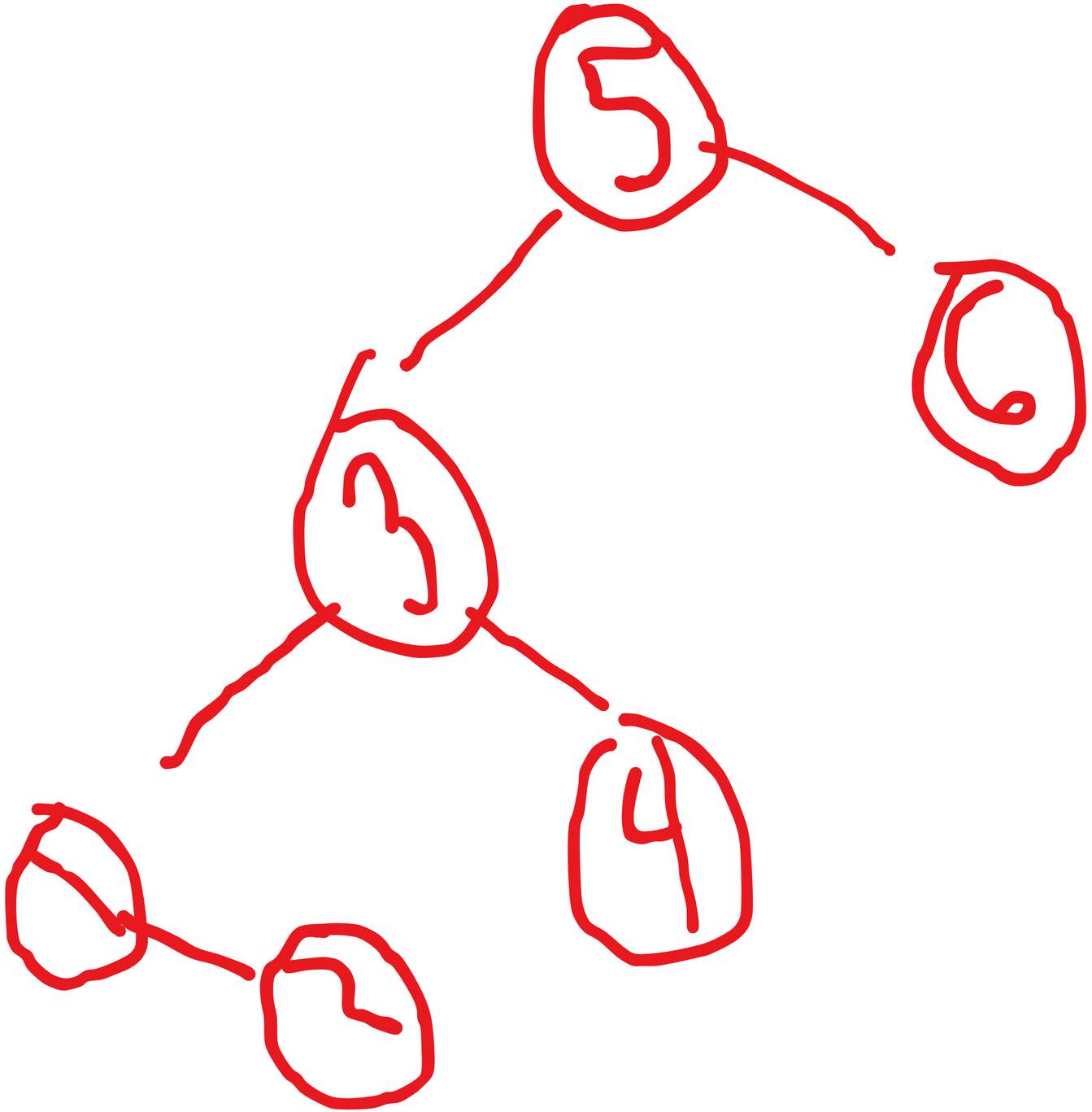
5, 6, 3, 4, 1, 2, 11, 10, 7

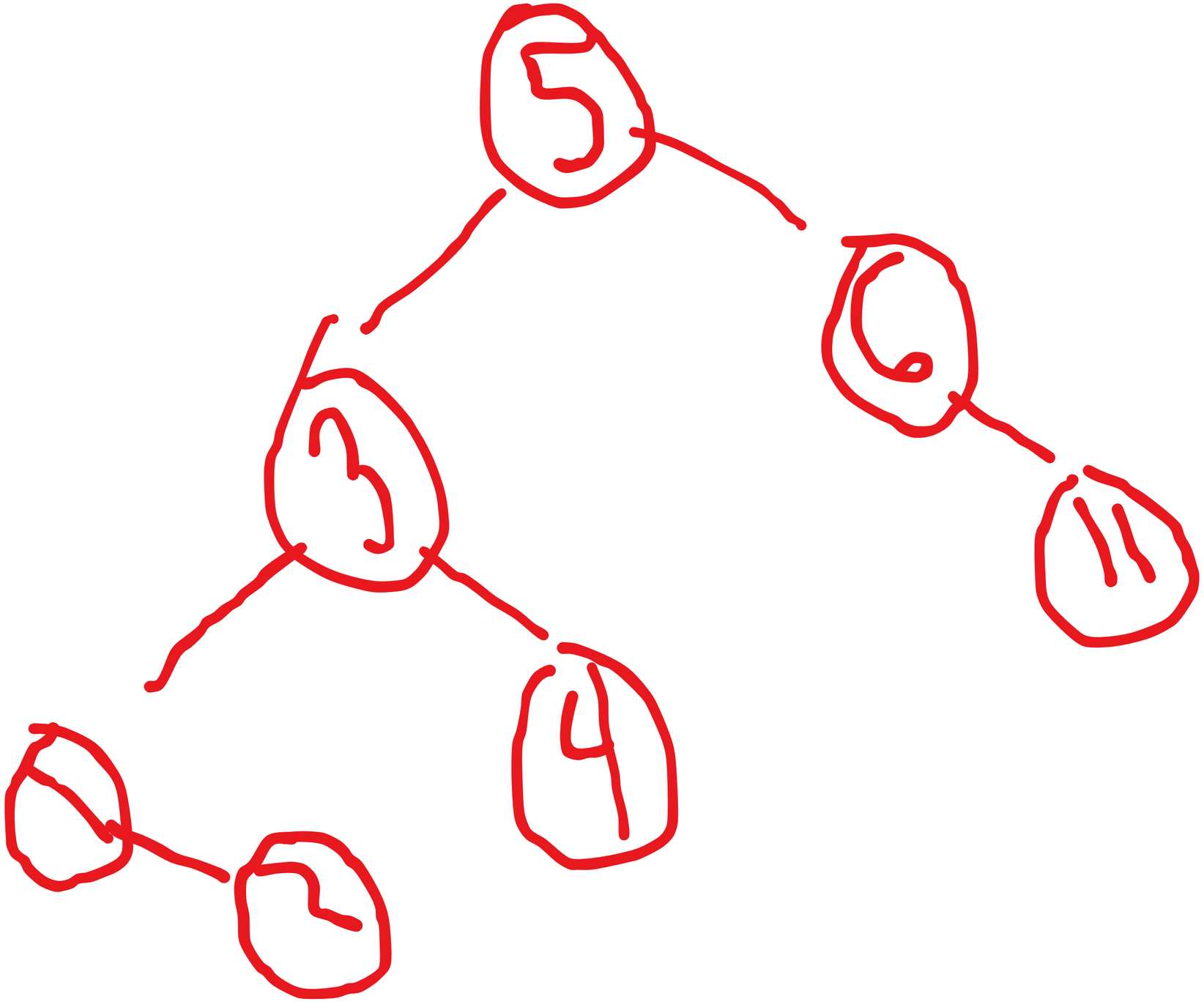


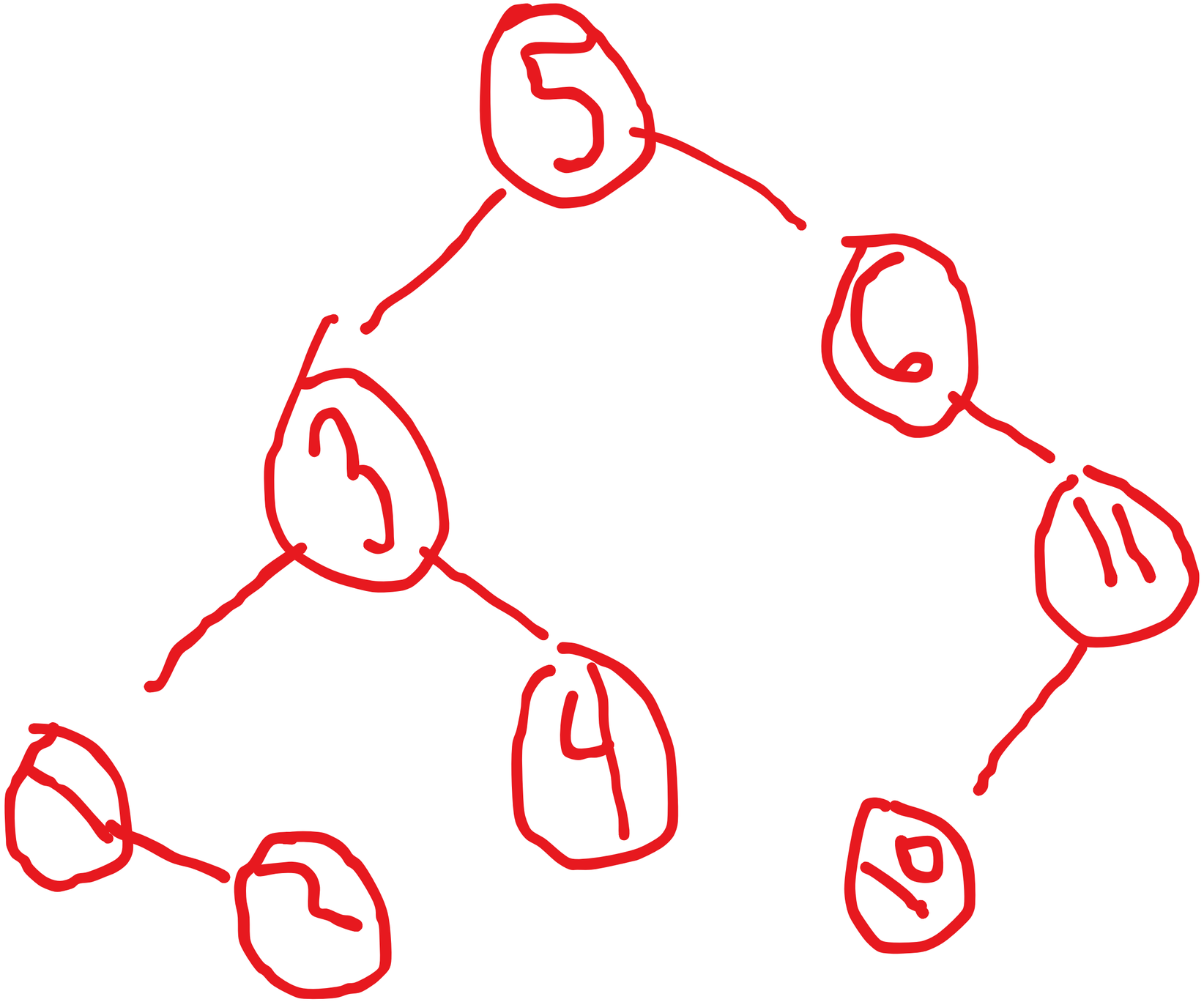

5
3
6

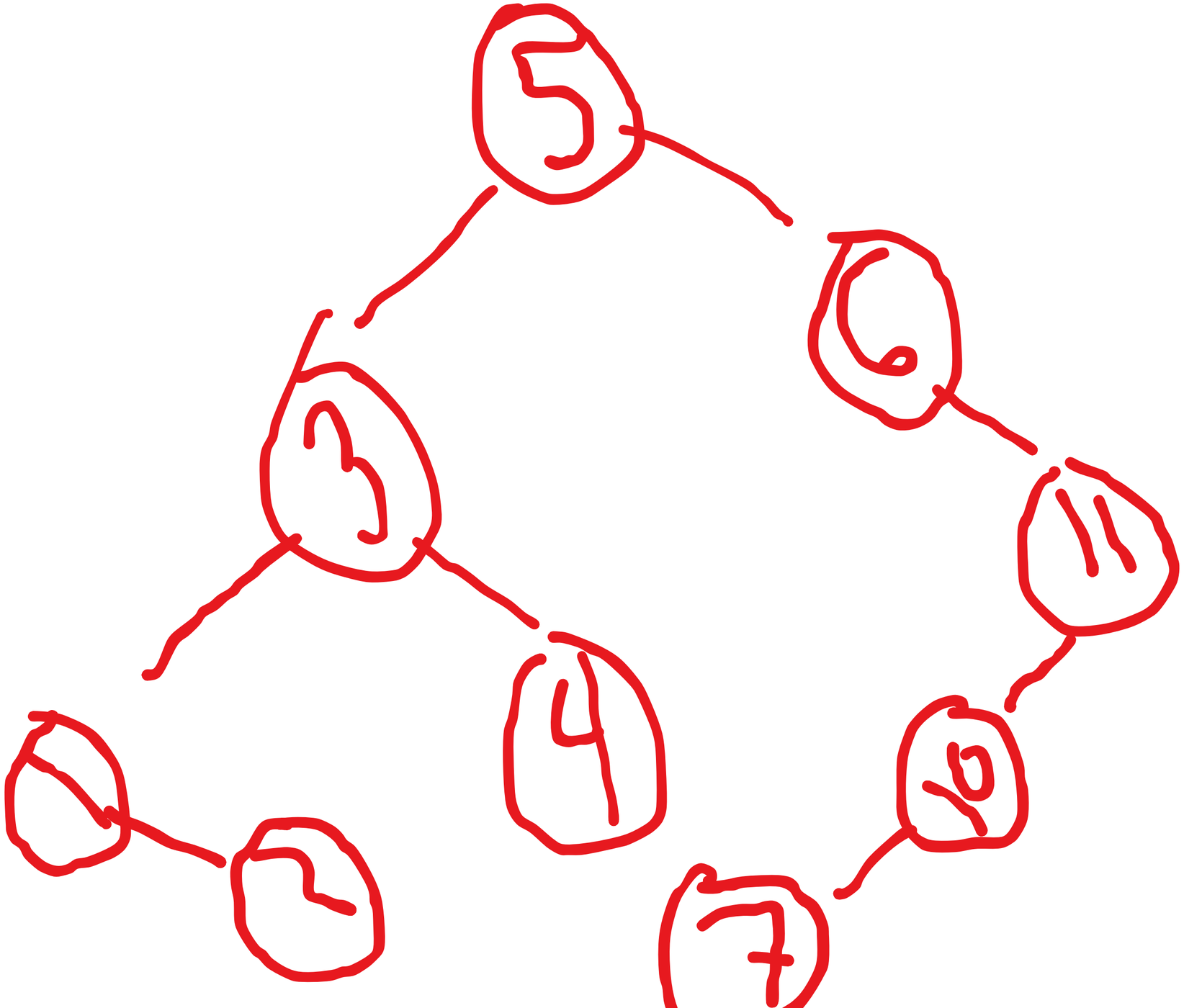

5
6
3
9


5
6
3
1
4


5
6
3
4
1
2

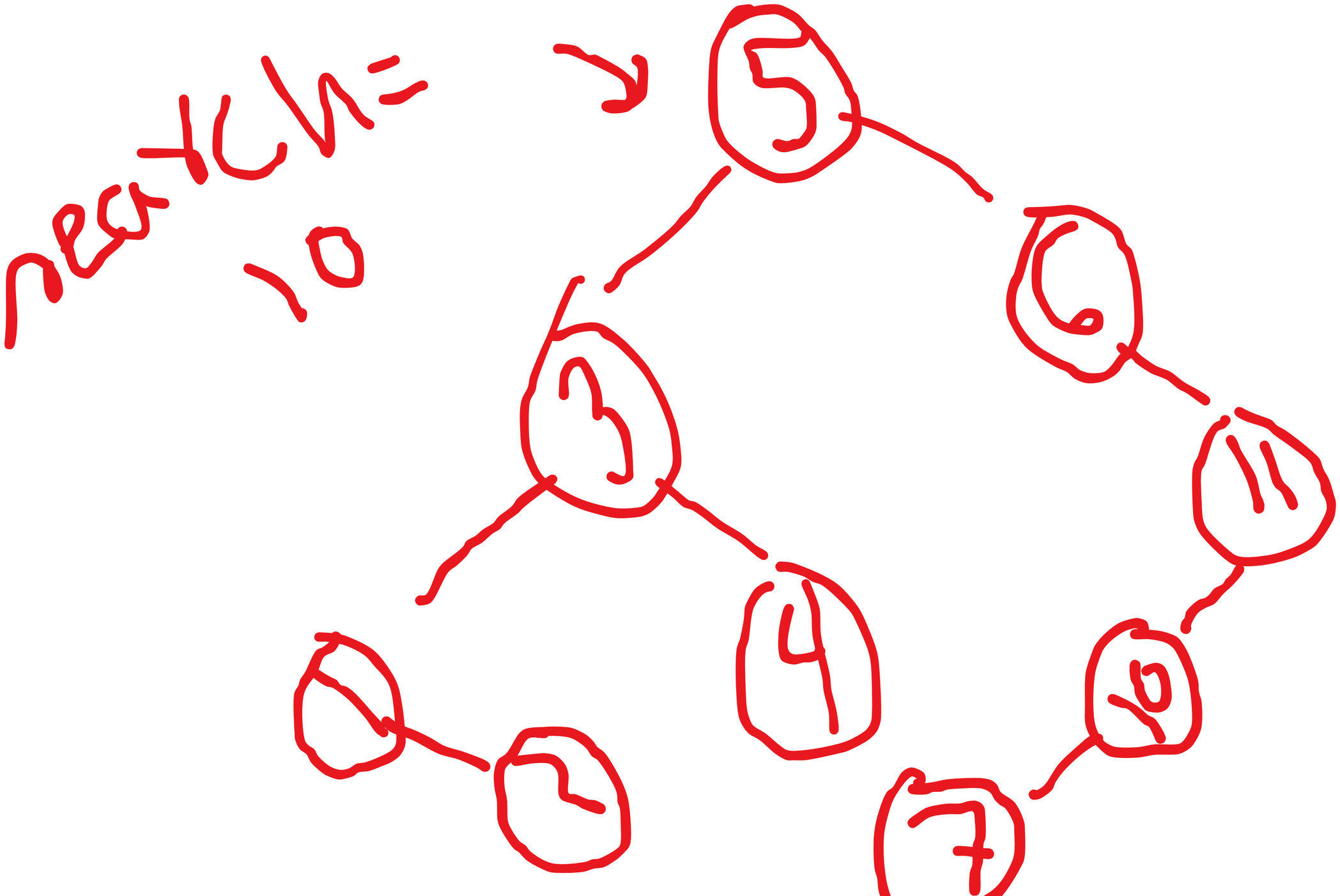

5
3
6
4
1
2

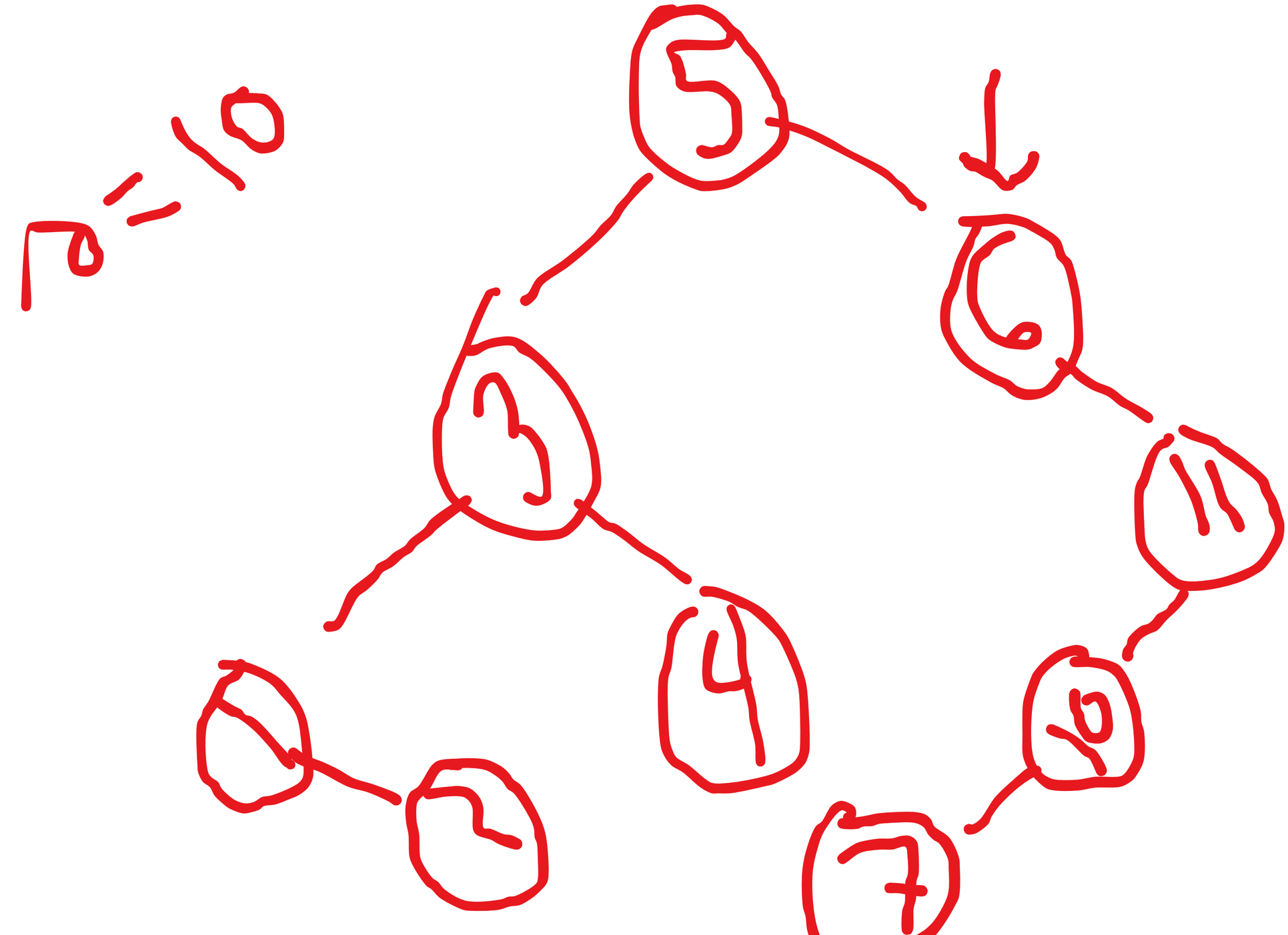

5
3
6
4
1
2
1
0

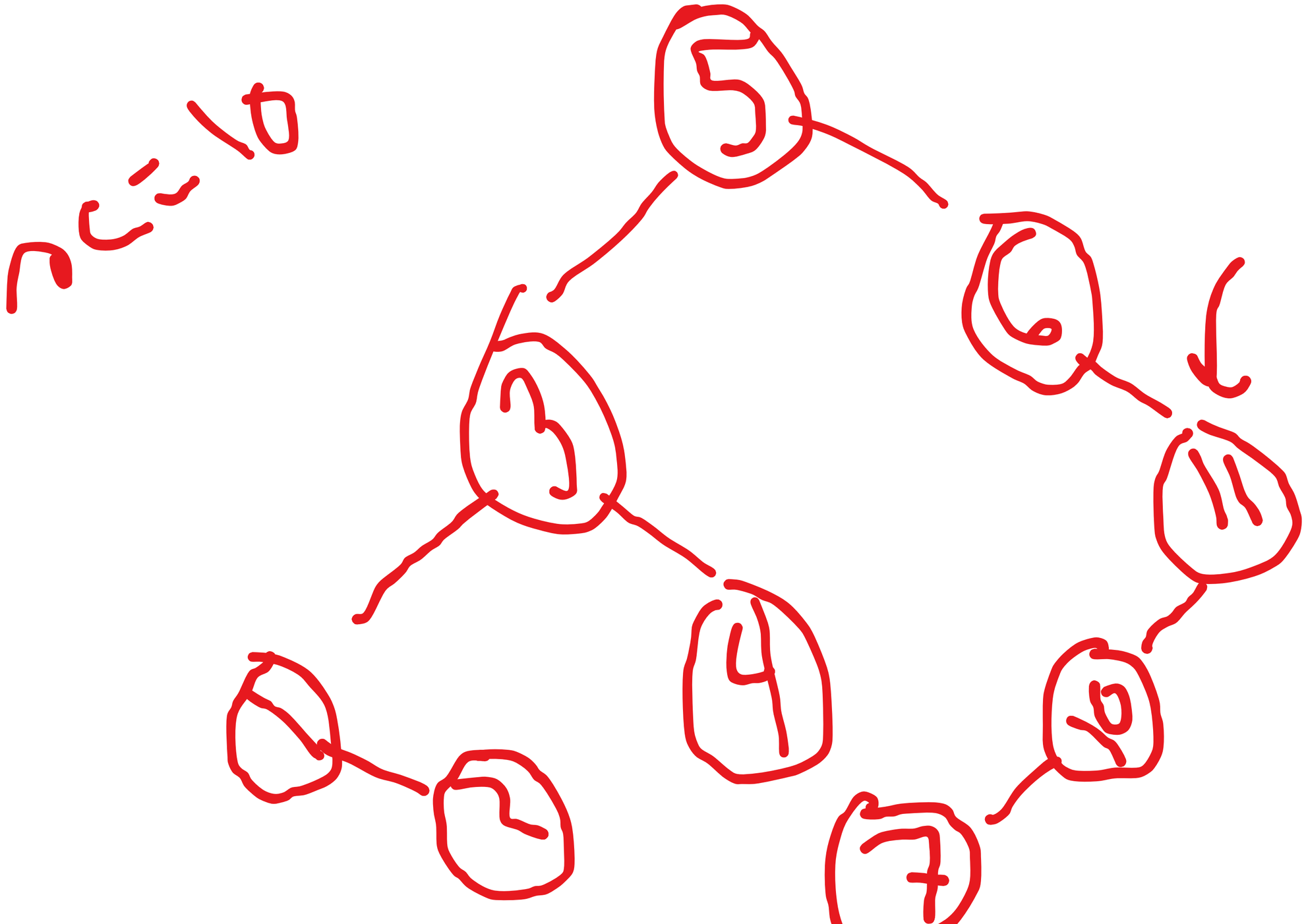

5
3
6
1
4
2
10
7

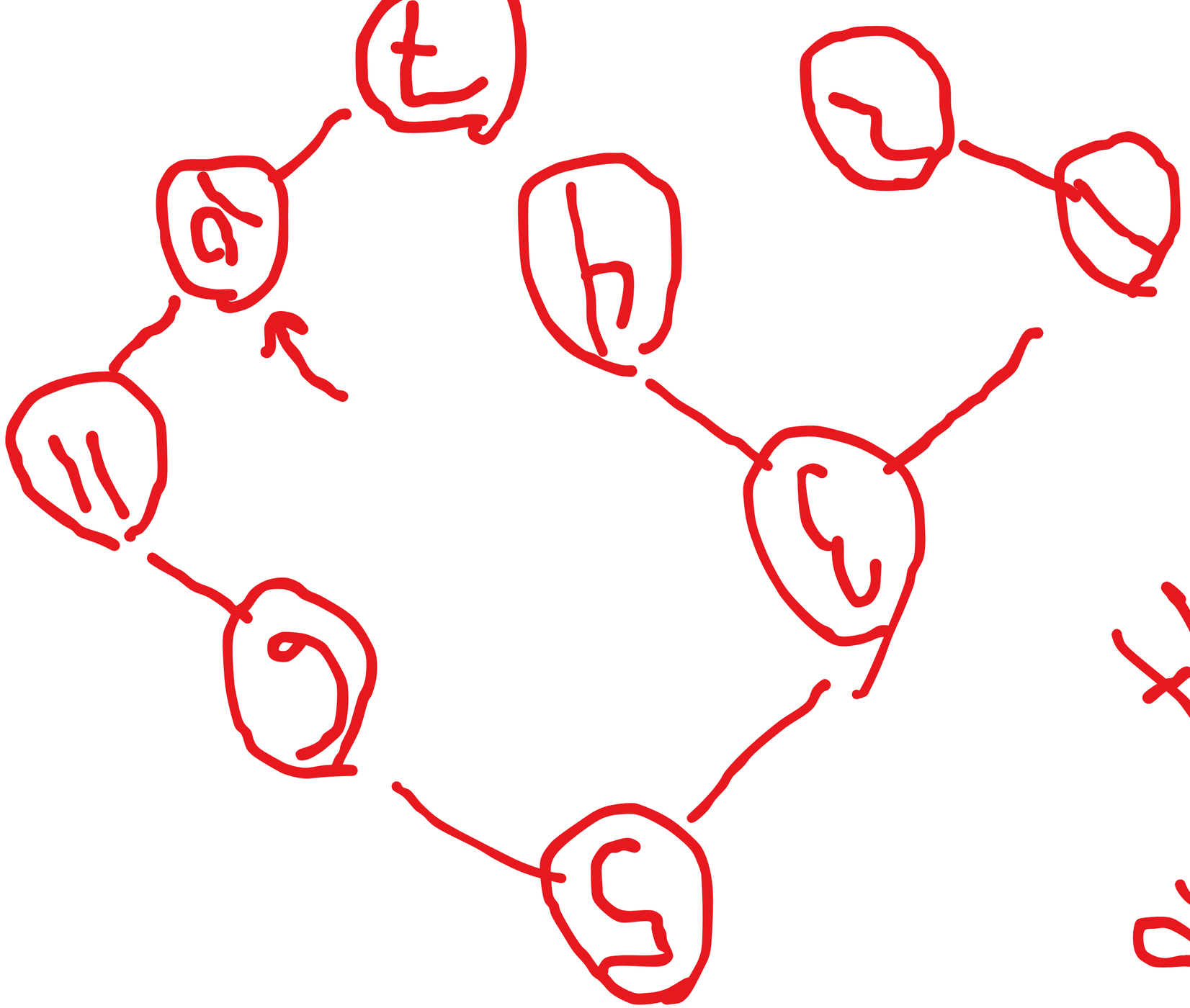
Now see for the search operations.

This is as like as same as insertion. small -> left, bigger -> right.


5
3
6
1
4
10
2
3
7


n=10
5
6
3
11
4
10
2
3
7


5
3
6
4
1
2
7
10



Now, let’s see the deletion.

In deletion, we have three step.

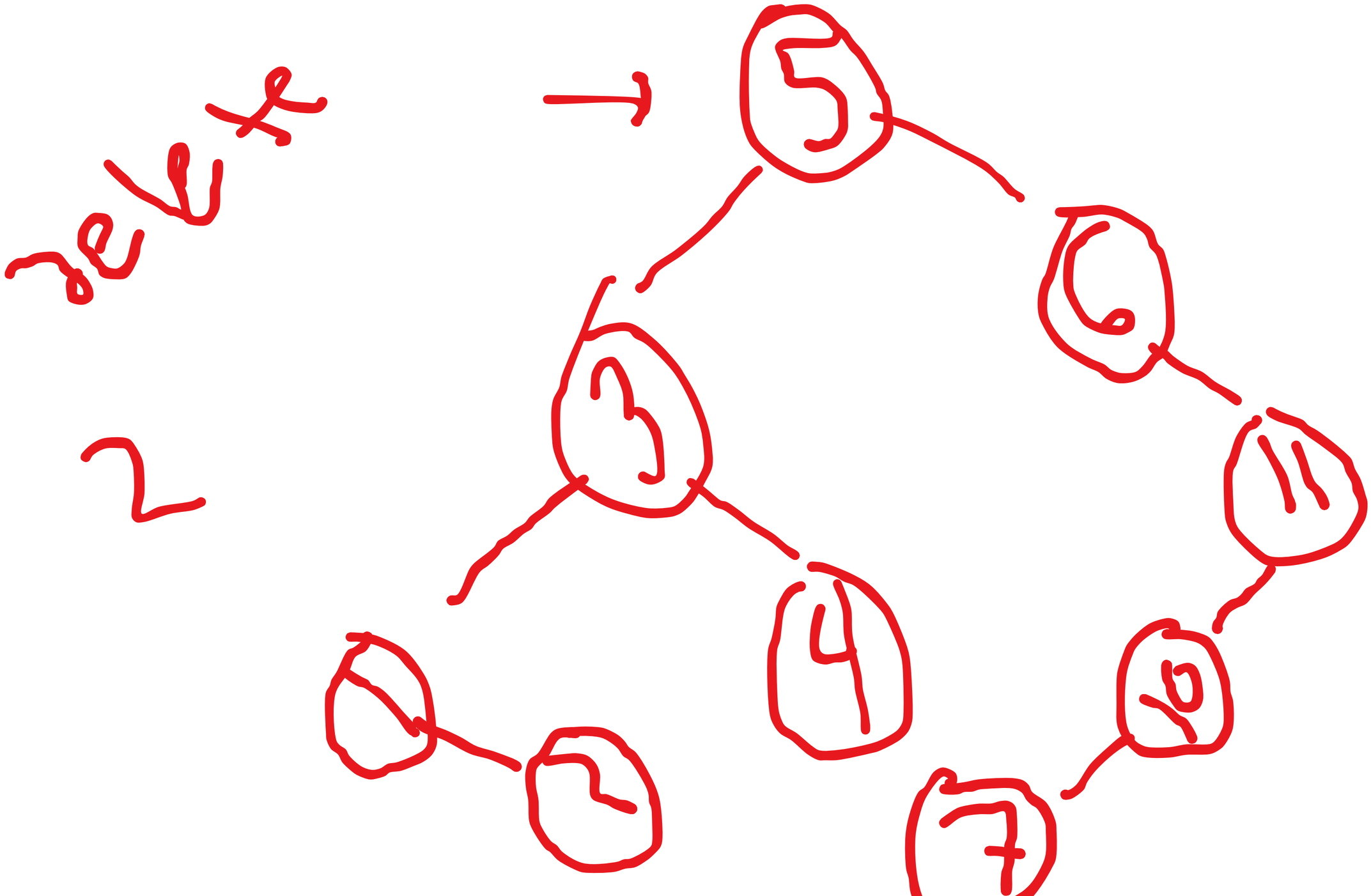
First find the node.

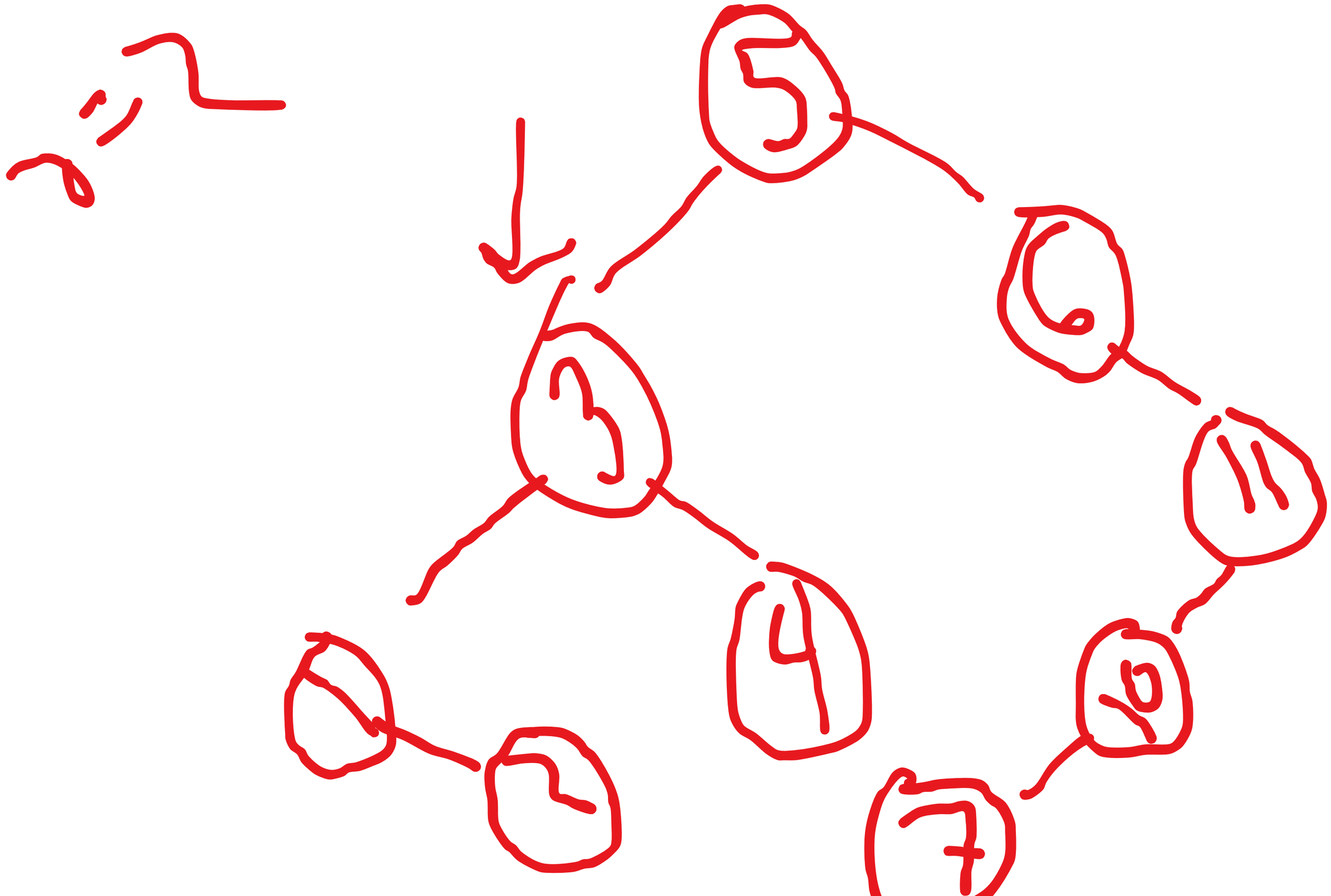
if the node is leaf node, then just delete it.

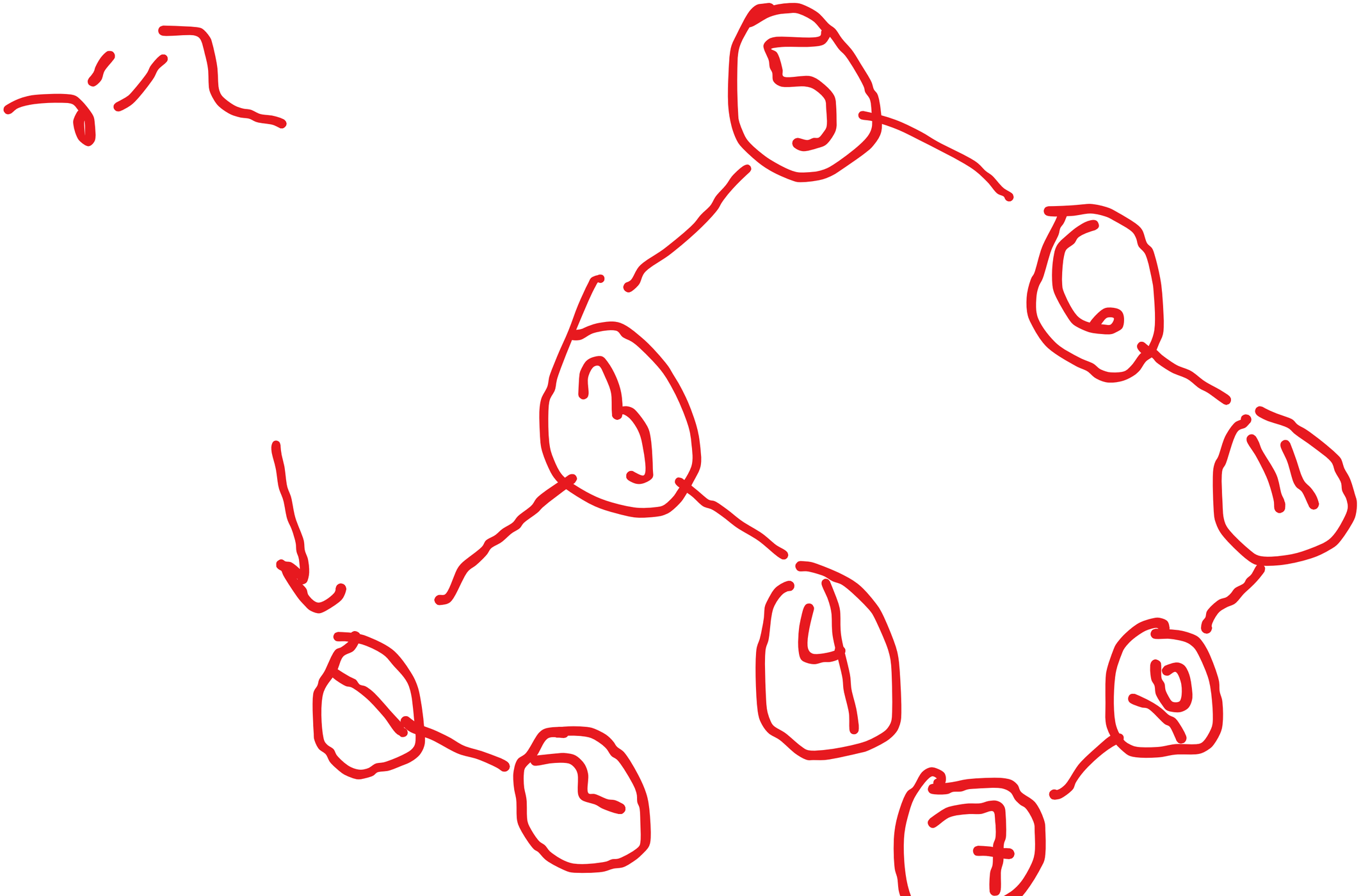
if it has only one children, then just shift with it and delete that node.

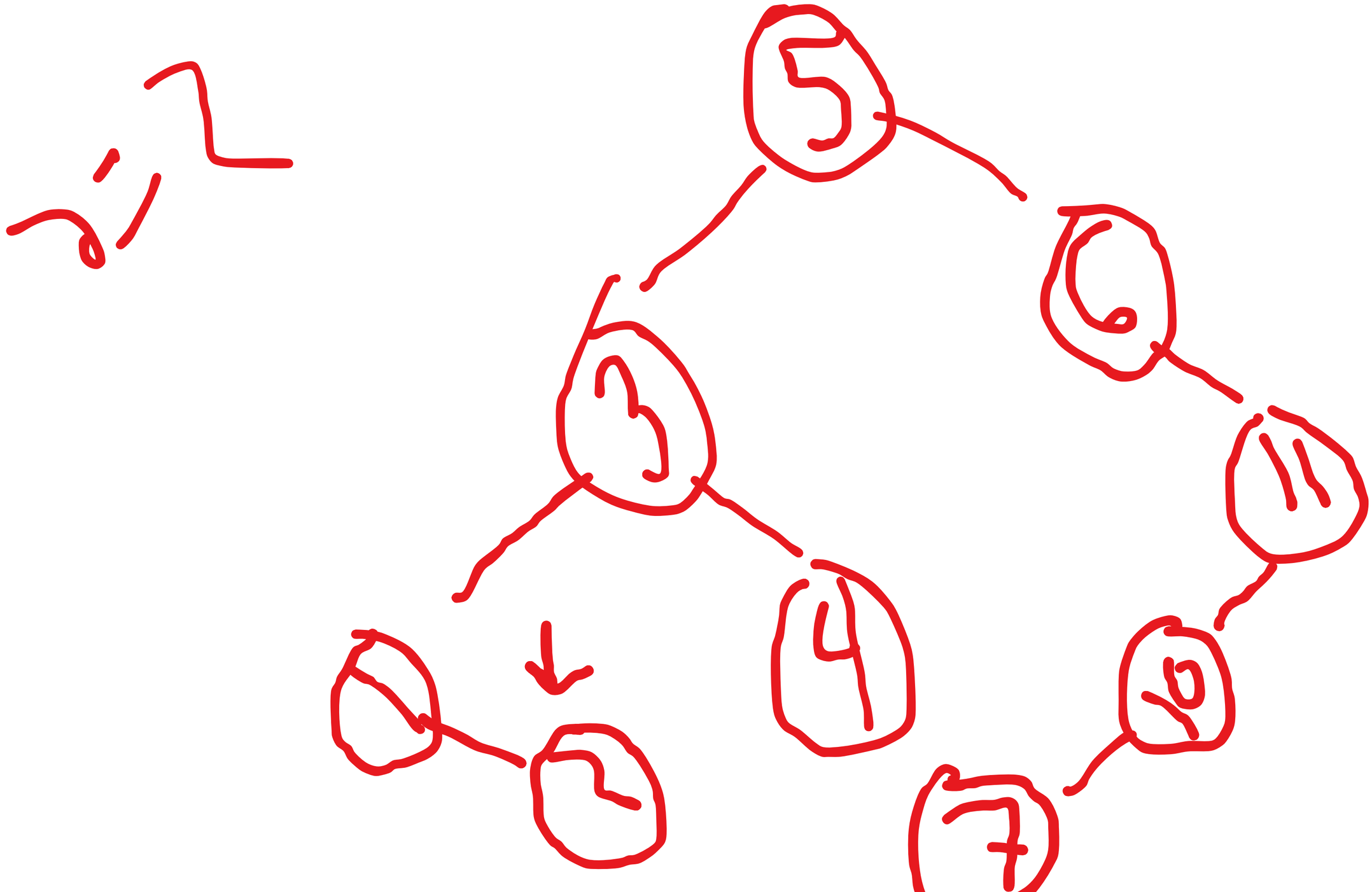
it has the two child, then just shift with the max node from the left side, and delete that node.

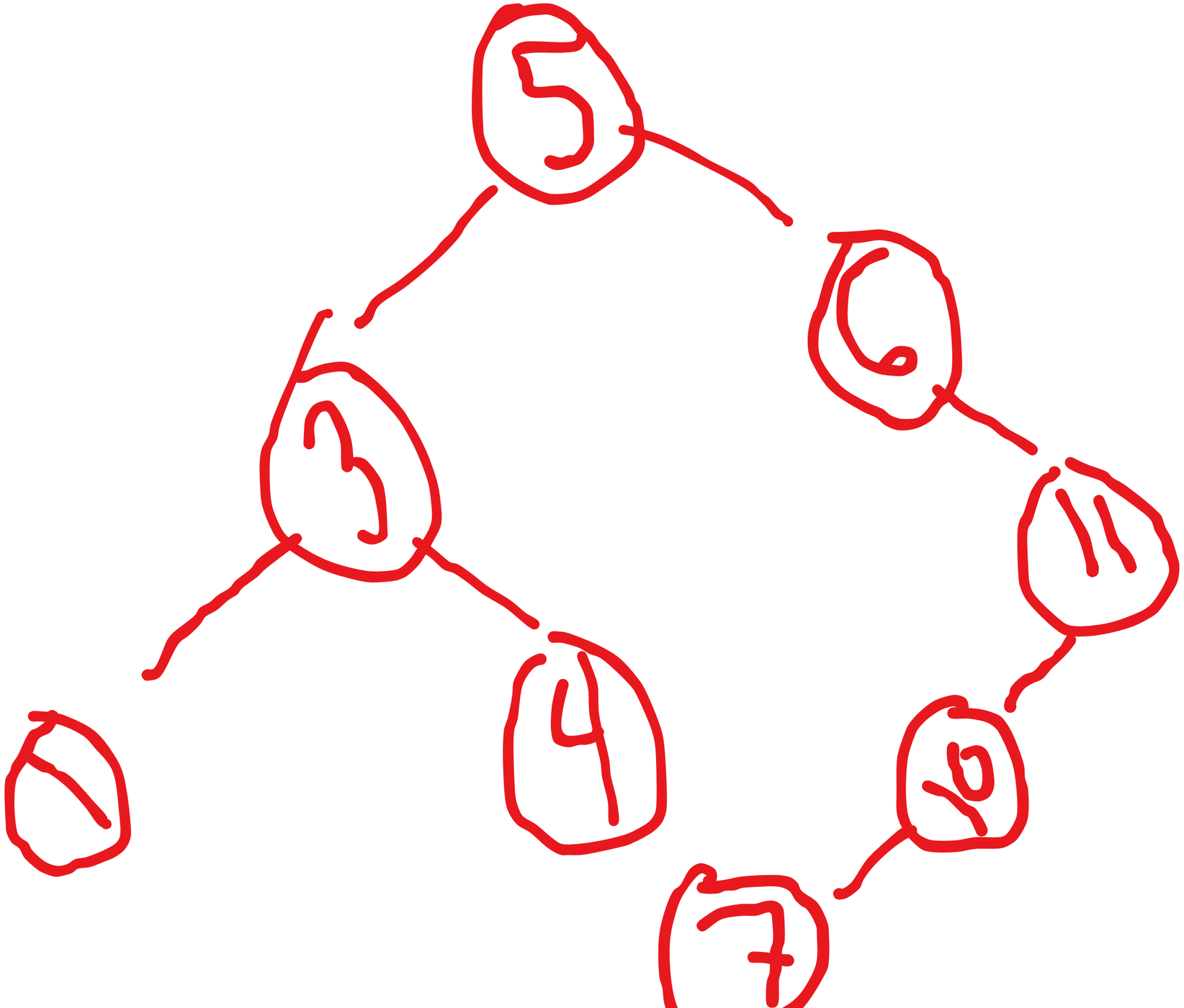
let’s solve an example.

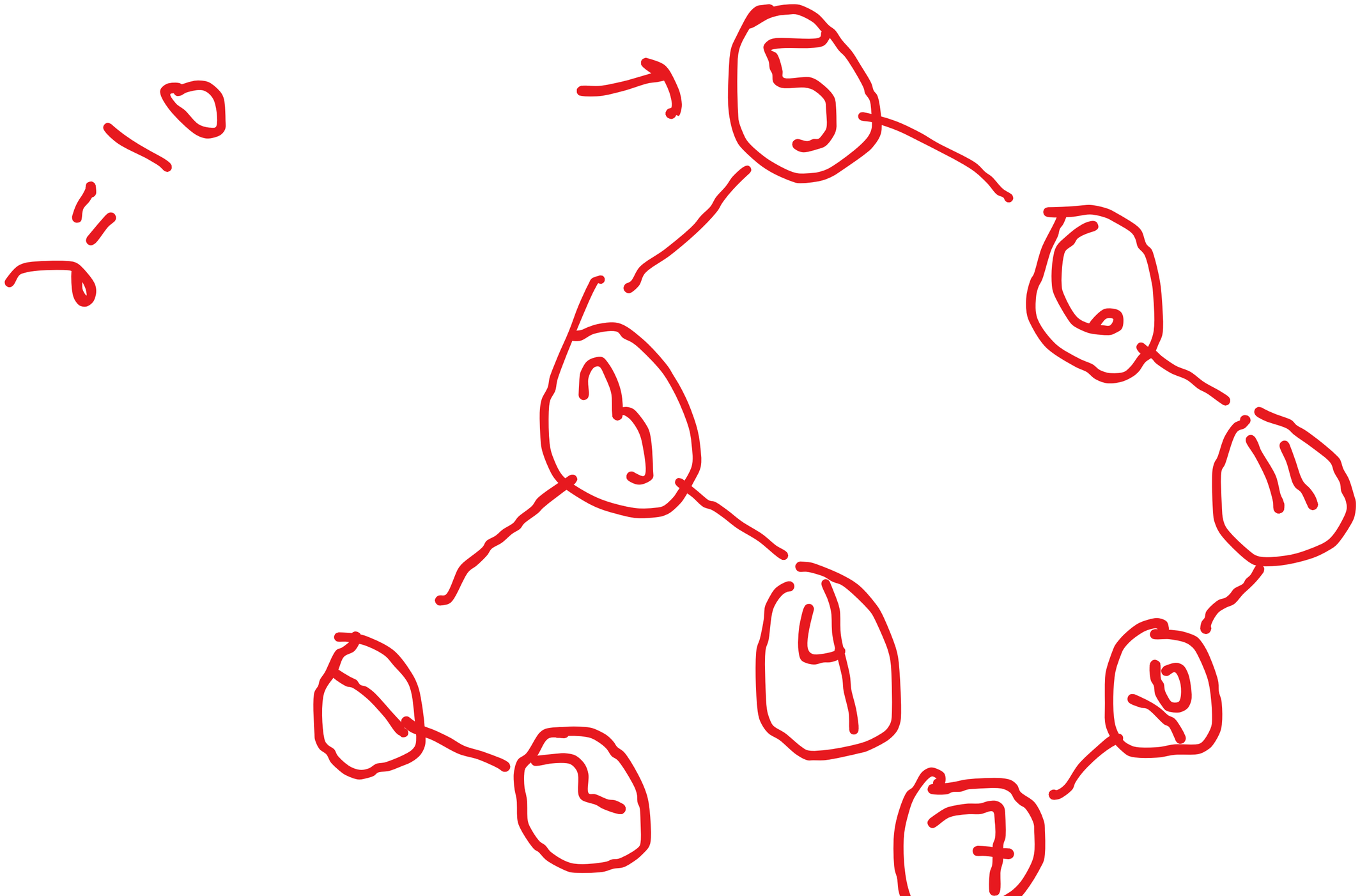

Select
2
5
3
6
1
4
11
10
7


5
3
6
4
11
2
7

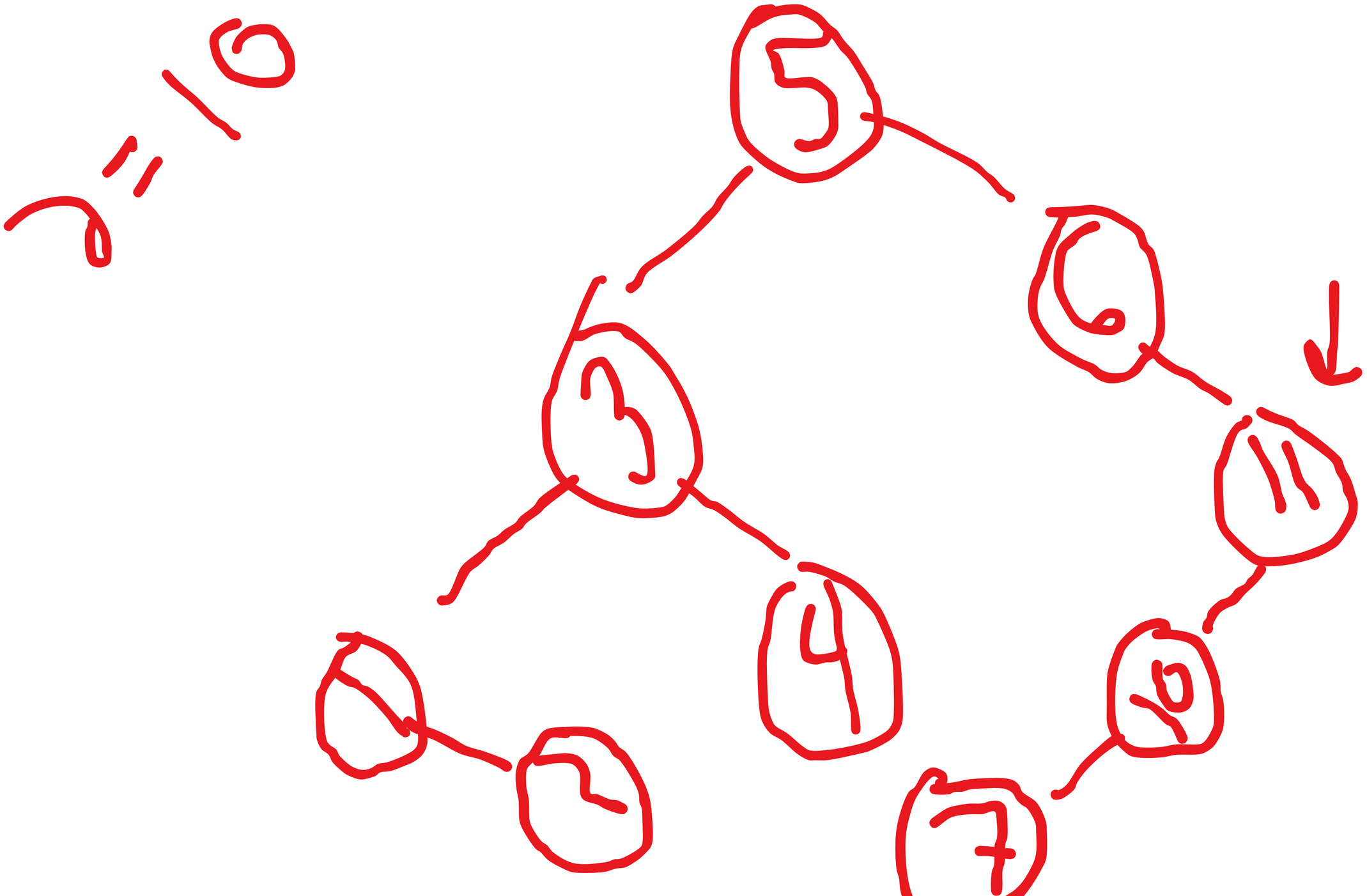

5
3
6
1
4
2
8
7

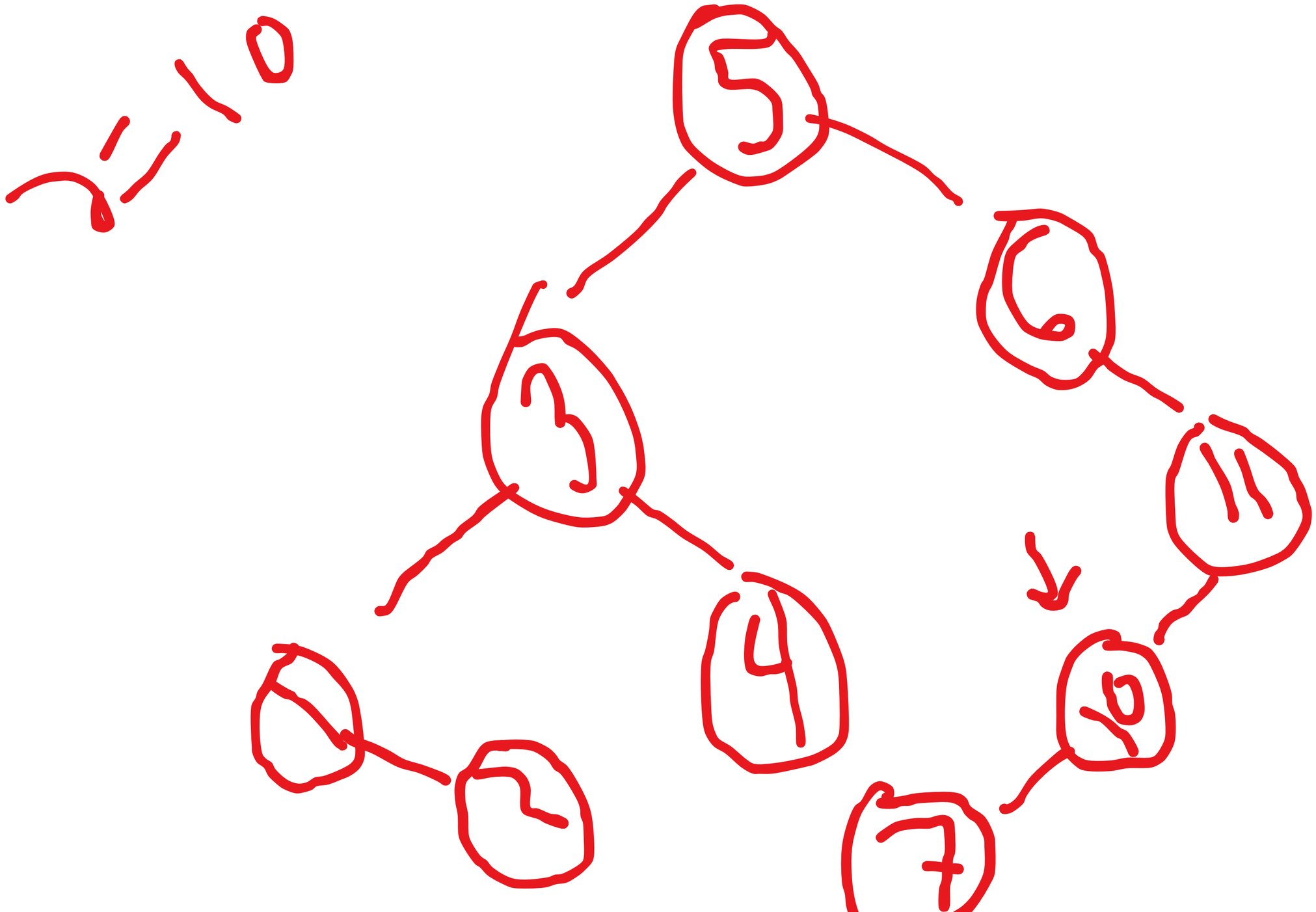

5
3
6
4
11
10
7

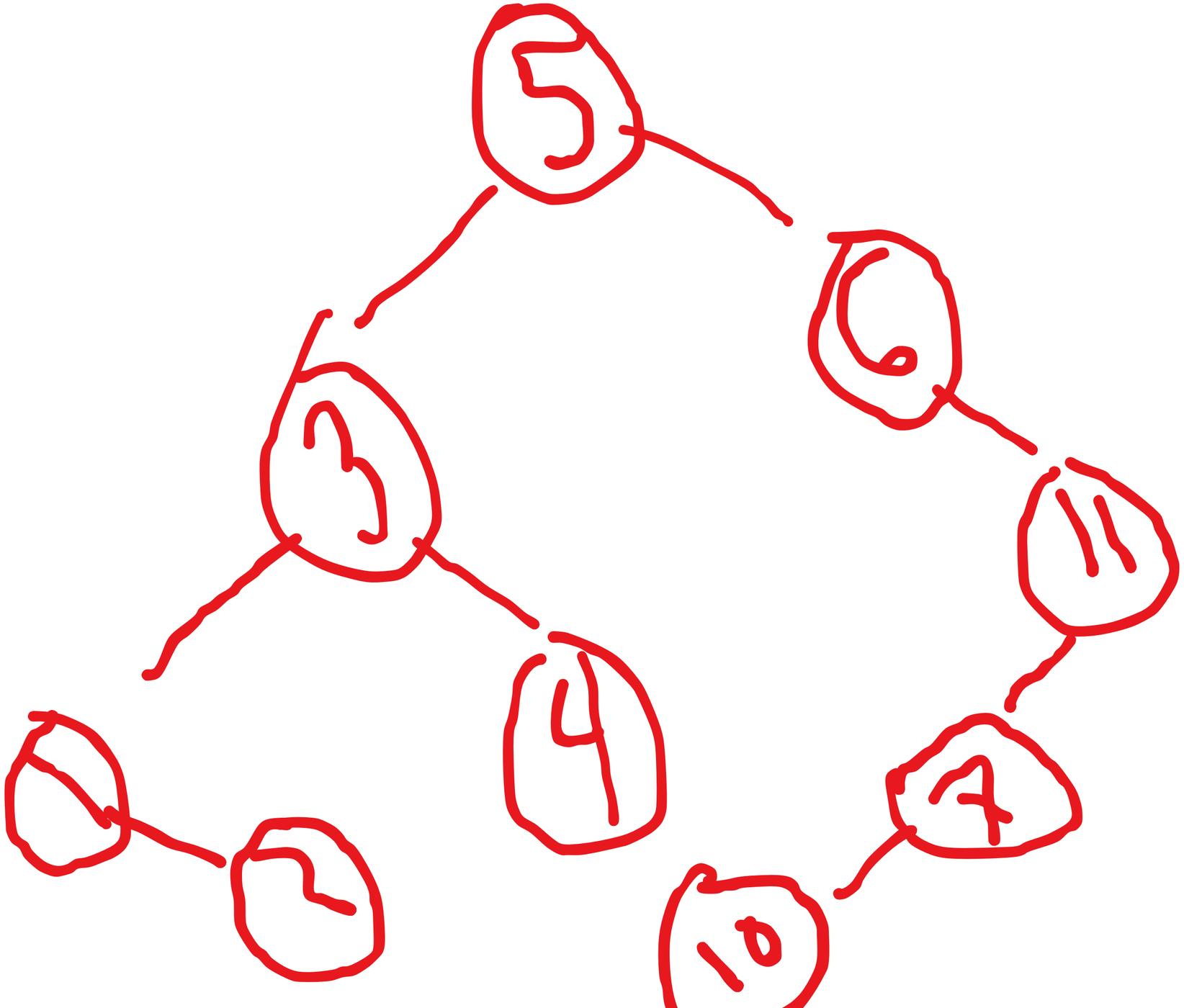

5
3
6
1
4
11
9
7

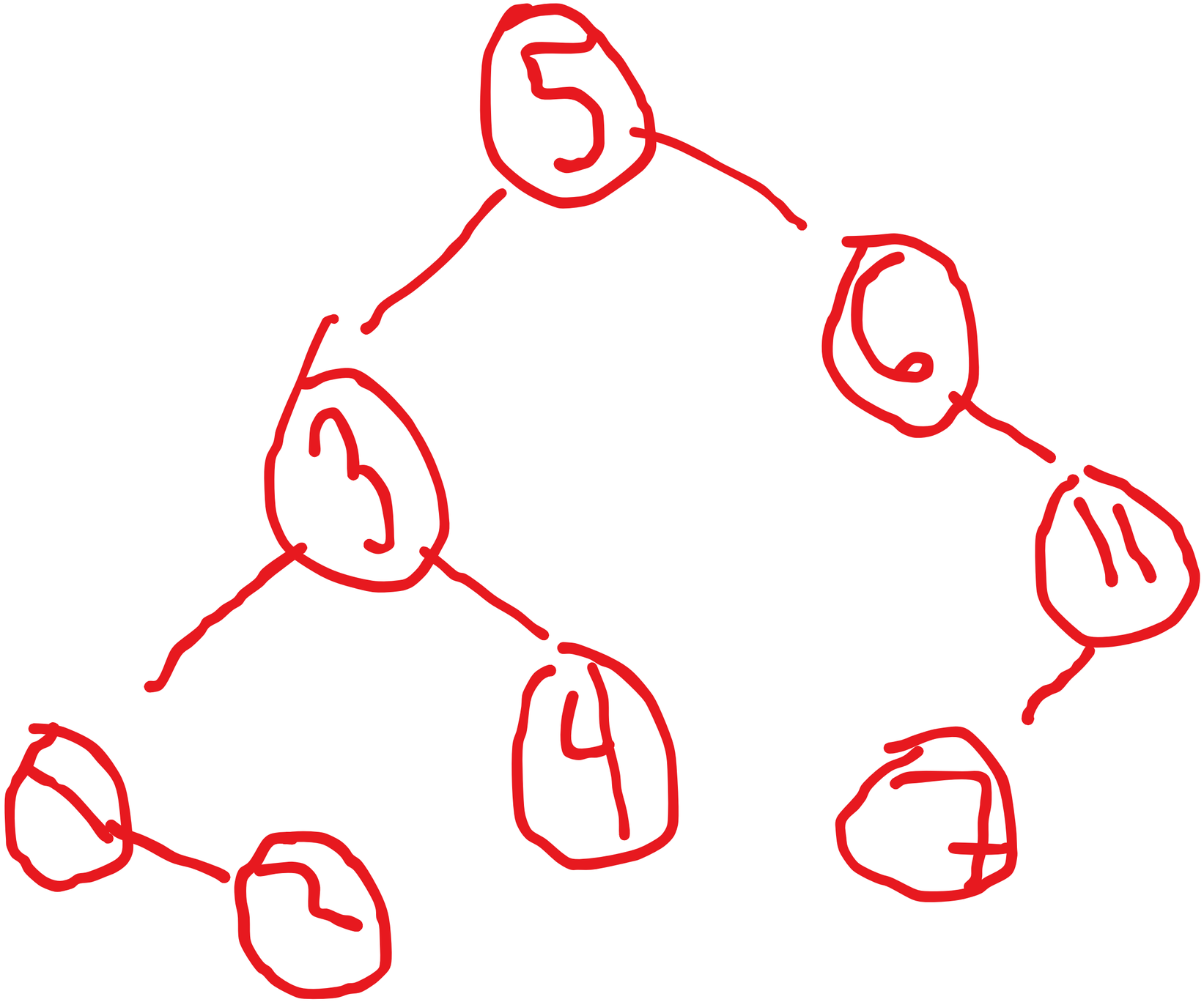

j = 10
5
6
3
11
4
10
2
7

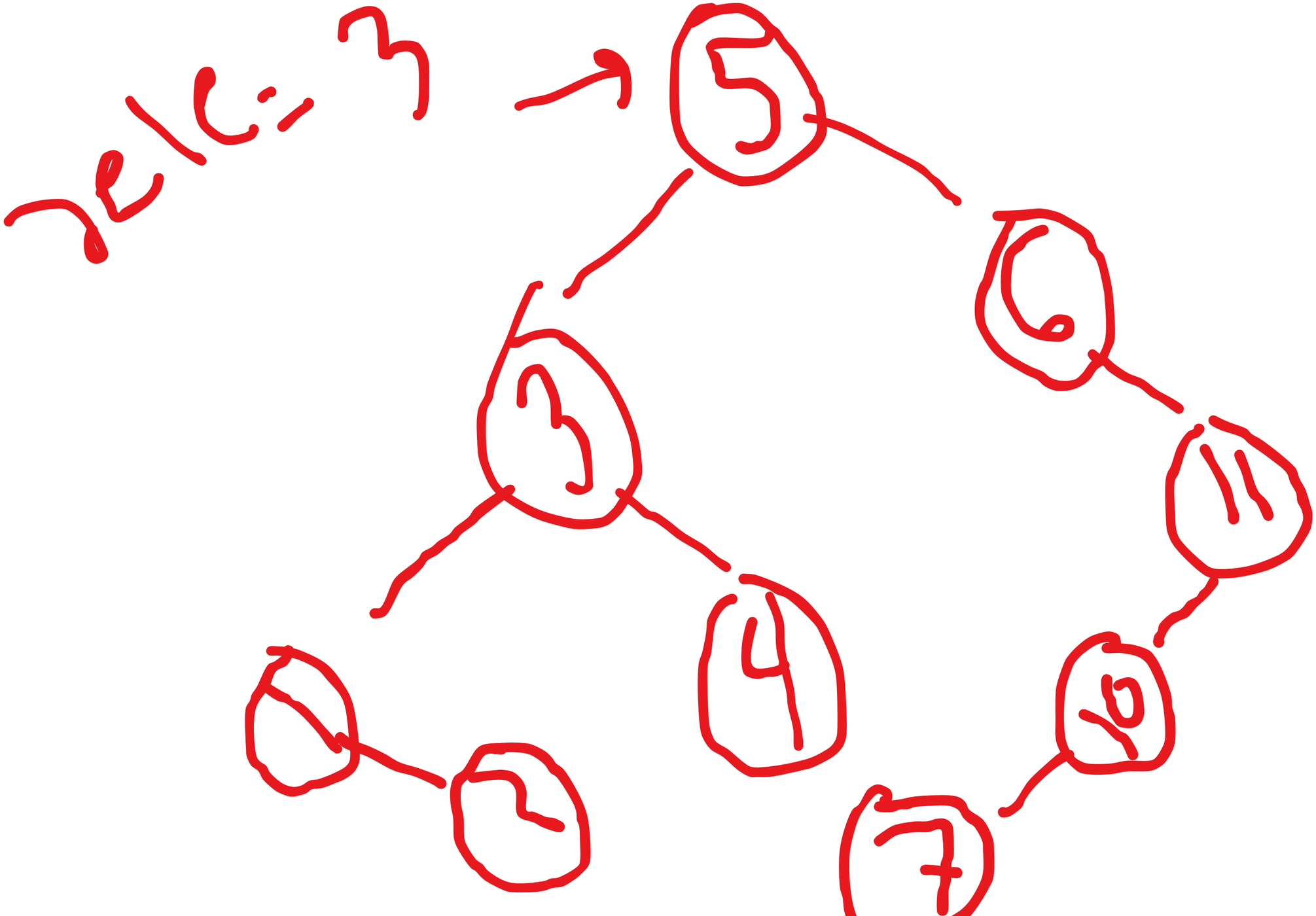
$d = 10$

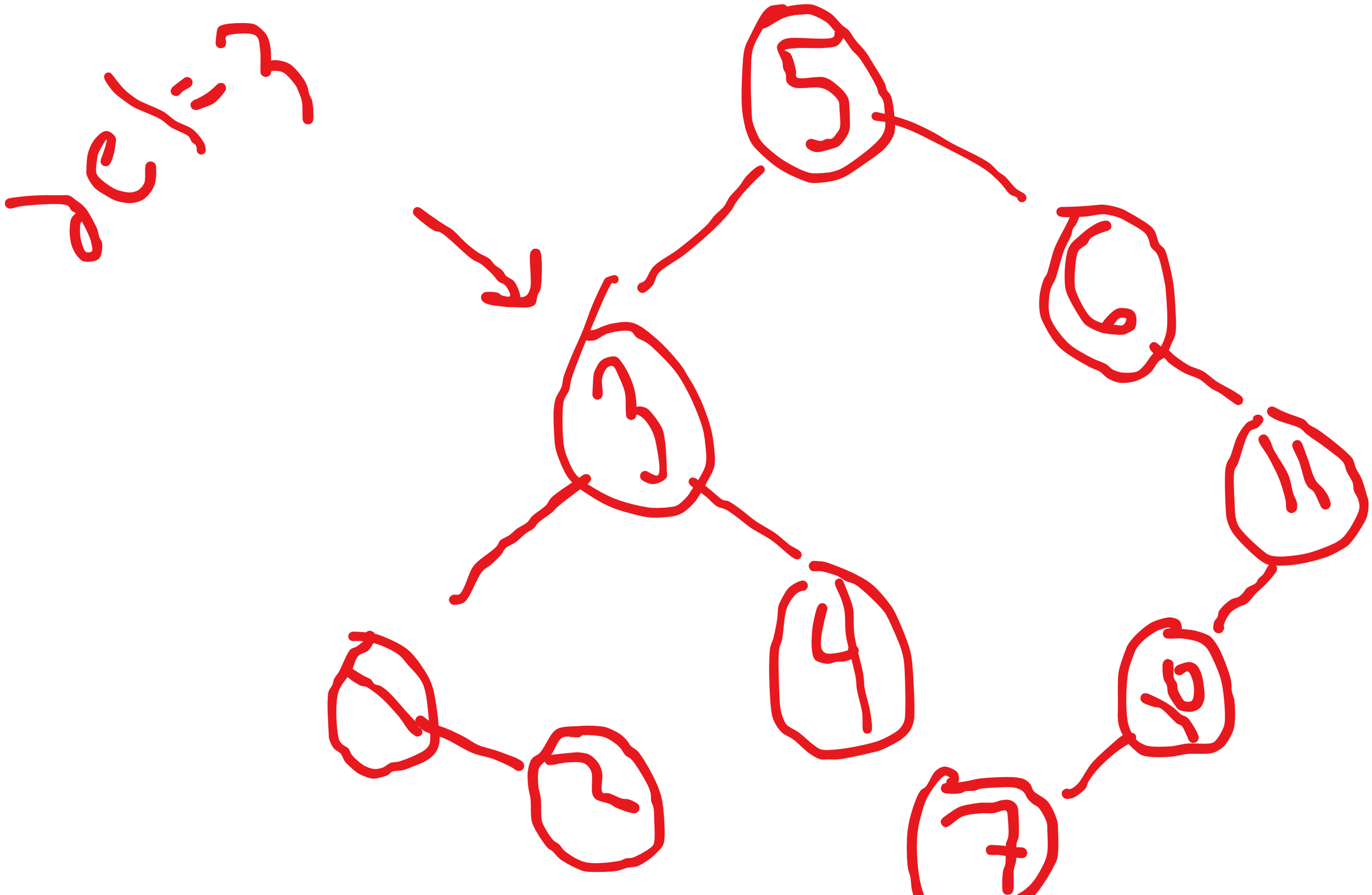

d = 10
5
6
3
11
4
10
2
7

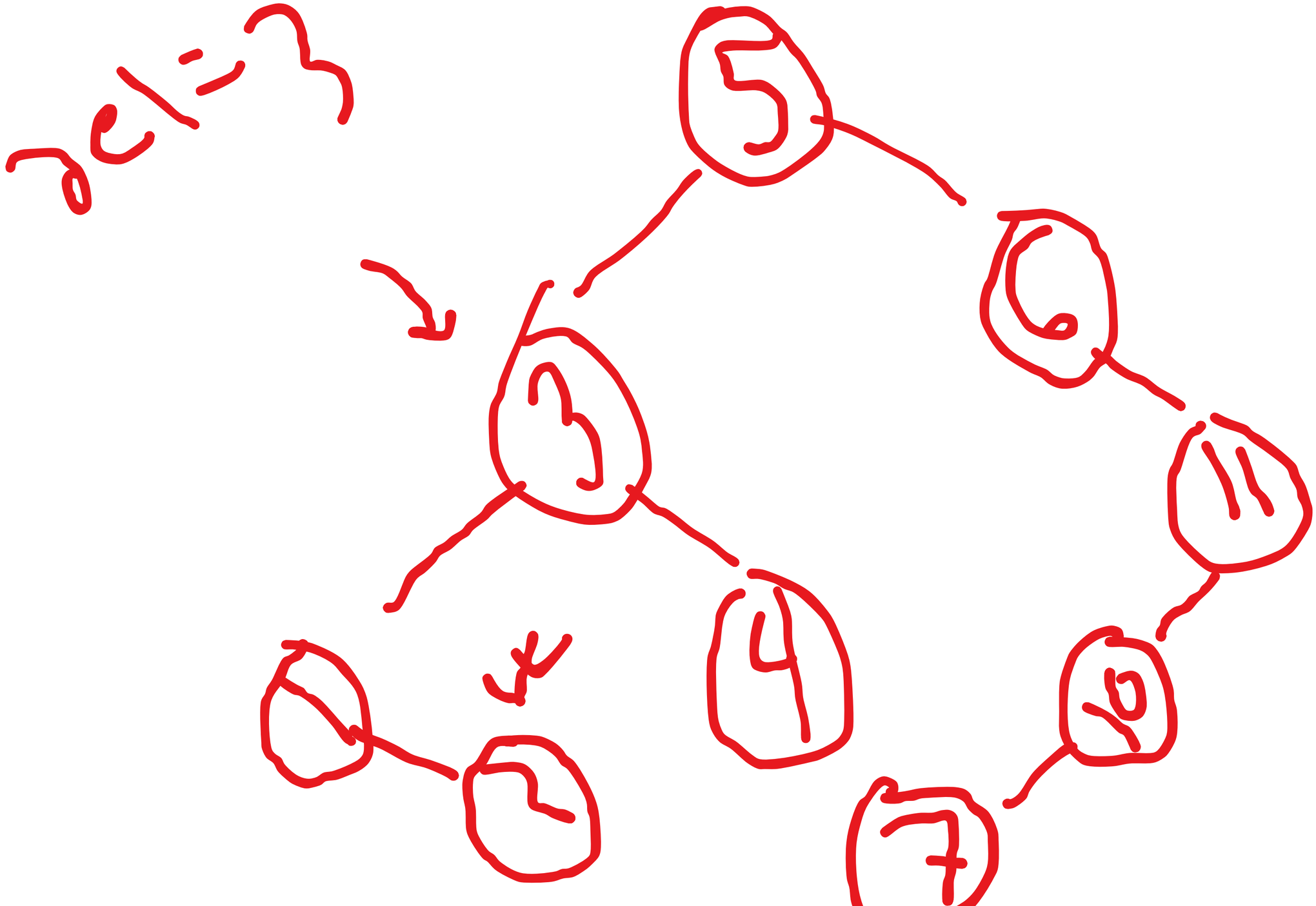

k=10
5
3
6
4
11
10
7

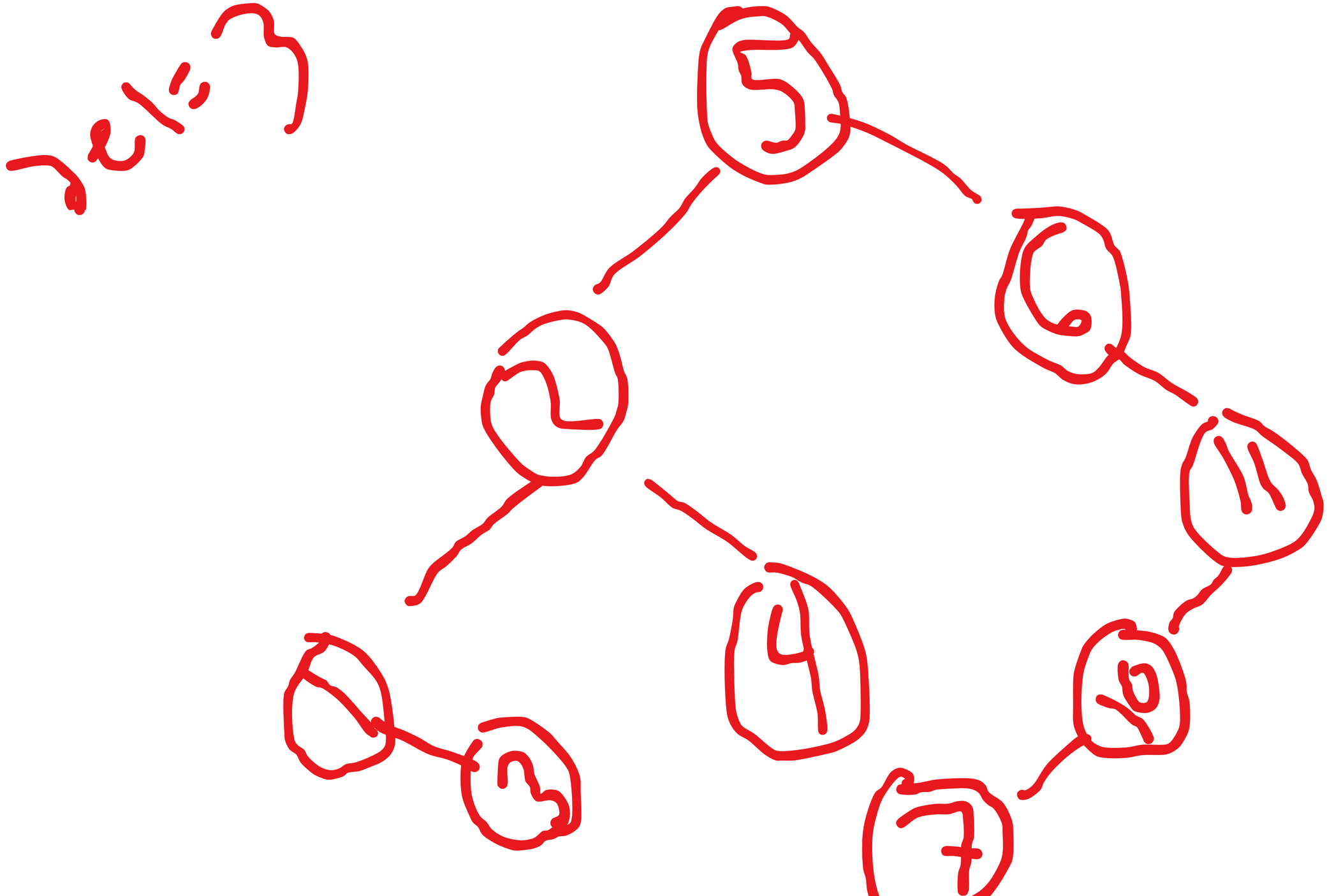

5
3
6
1
4
11
2
7
10

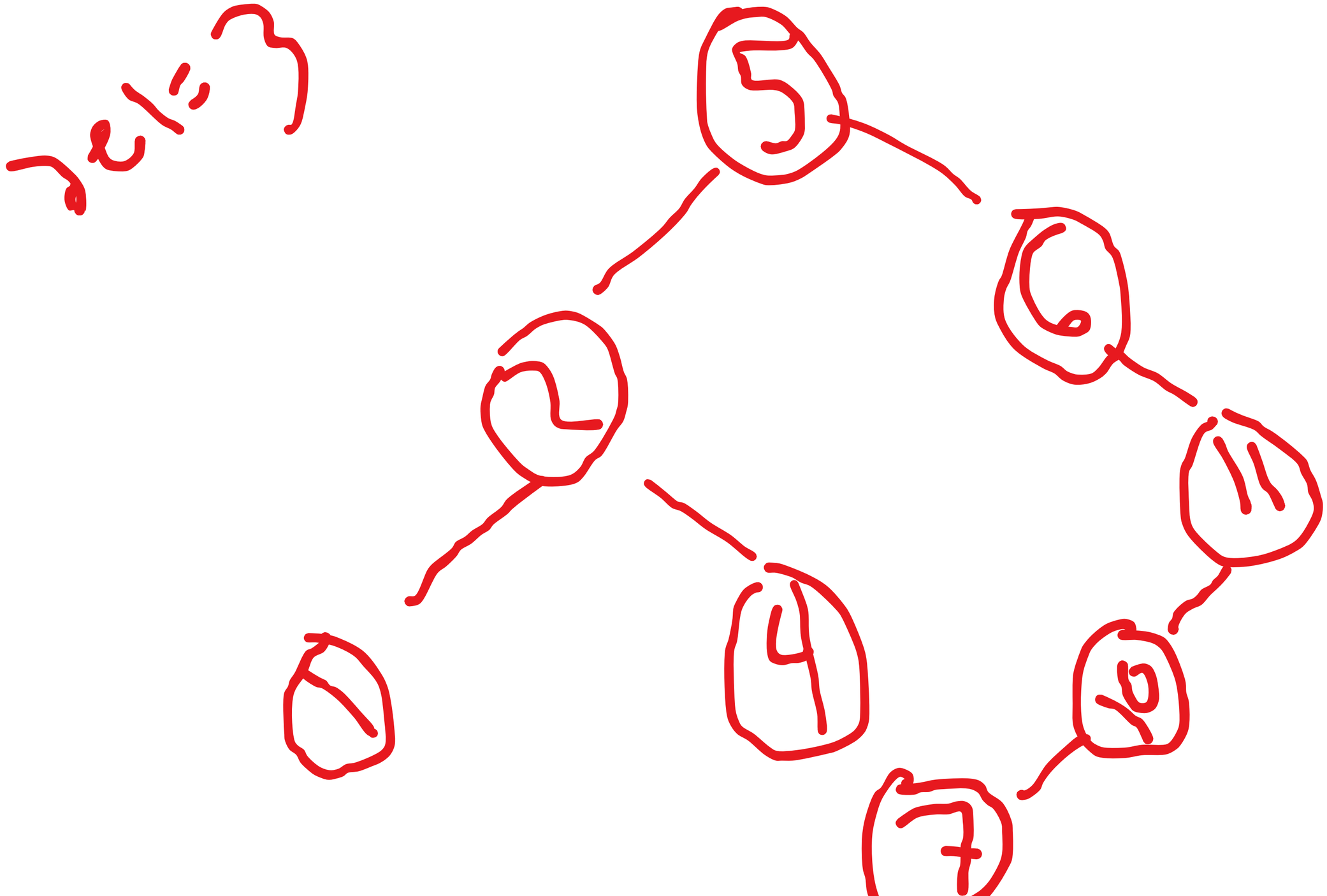

5
3
6
1
4
2
7


delete 3
5
3
6
1
4
2
10
7


del(3)
5
3
6
1
4
11
2
10
7


del=3
5
3
6
4
11
2
10
7


5
2
6
4
11
1
3
10
7


5
2
6
1
4
9
7

All the issues are solved at here.

Let’s go to the implementations