Heap

Specialize binary tree base ds, follow the heap properties.

Purpose of this ds.

Primarily use for implementing the priorityqueue.

Usages of this ds.

For frequently access the min and max element.

Which algoritthoms mainly use this ds?

- Graph algorithm(prims, dijkstra,..)
- Priority scheduling(scheduling job base)
- heap sort

Properties of heap

Heap property :-
root is larger or
bigger than it's
children.

Complete Binary tree.

Types of heap.

Binary Heap.

Max Heap.

Min Heap.

Fibonacci Heap.

Insertion operations

Add the new element to the end of the heap (maintaining the complete binary tree property).

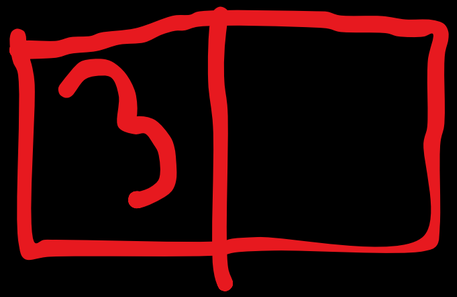
Percolate Up: Repeatedly swap the newly inserted element with its parent until the heap property is satisfied.

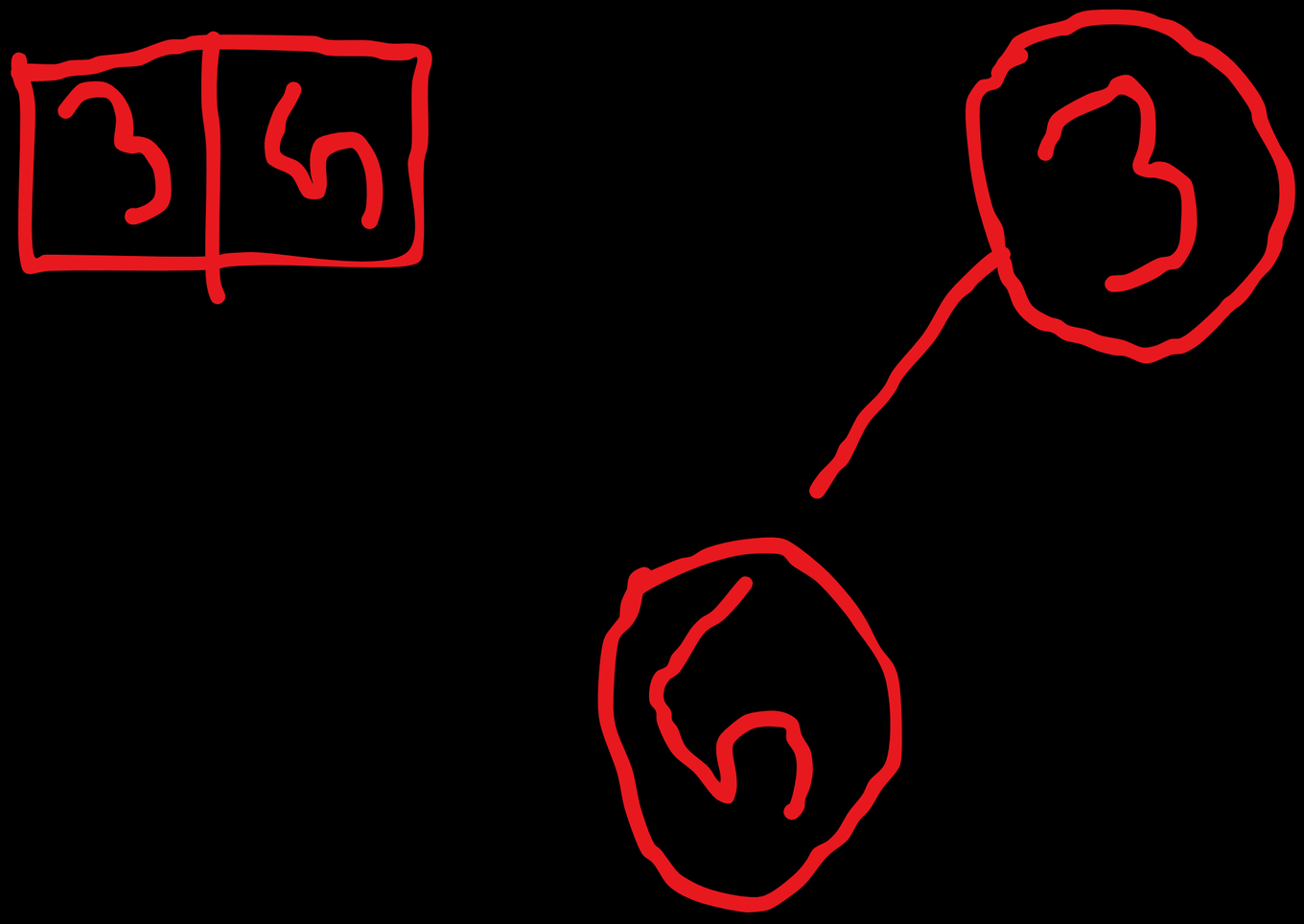
Here if the parent index is i, then left -> 2i+1 and right child -> 2i+2

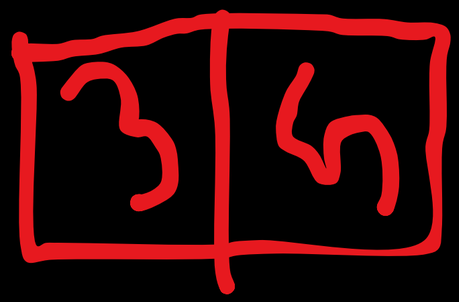
Let’s solve an example for min and max heap.

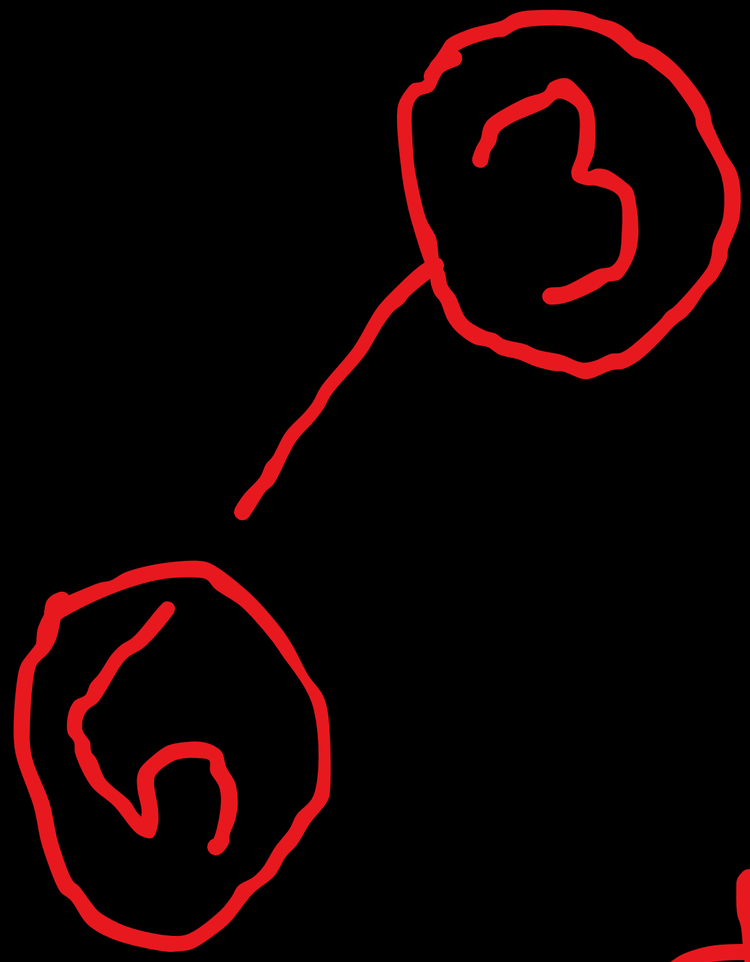
3, 5, 4, 6, 1, 2, 7, 9,

0, 2, 1

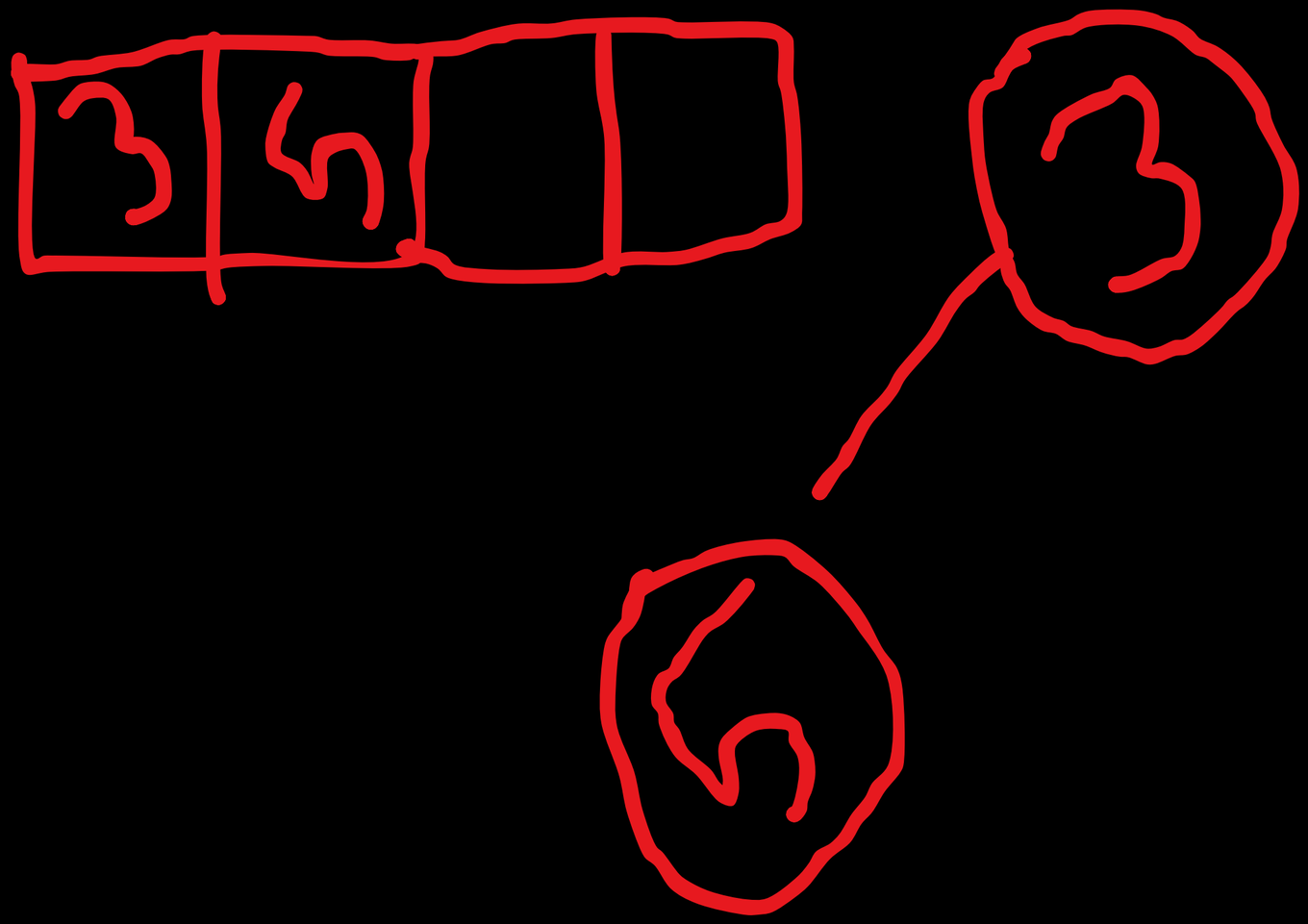


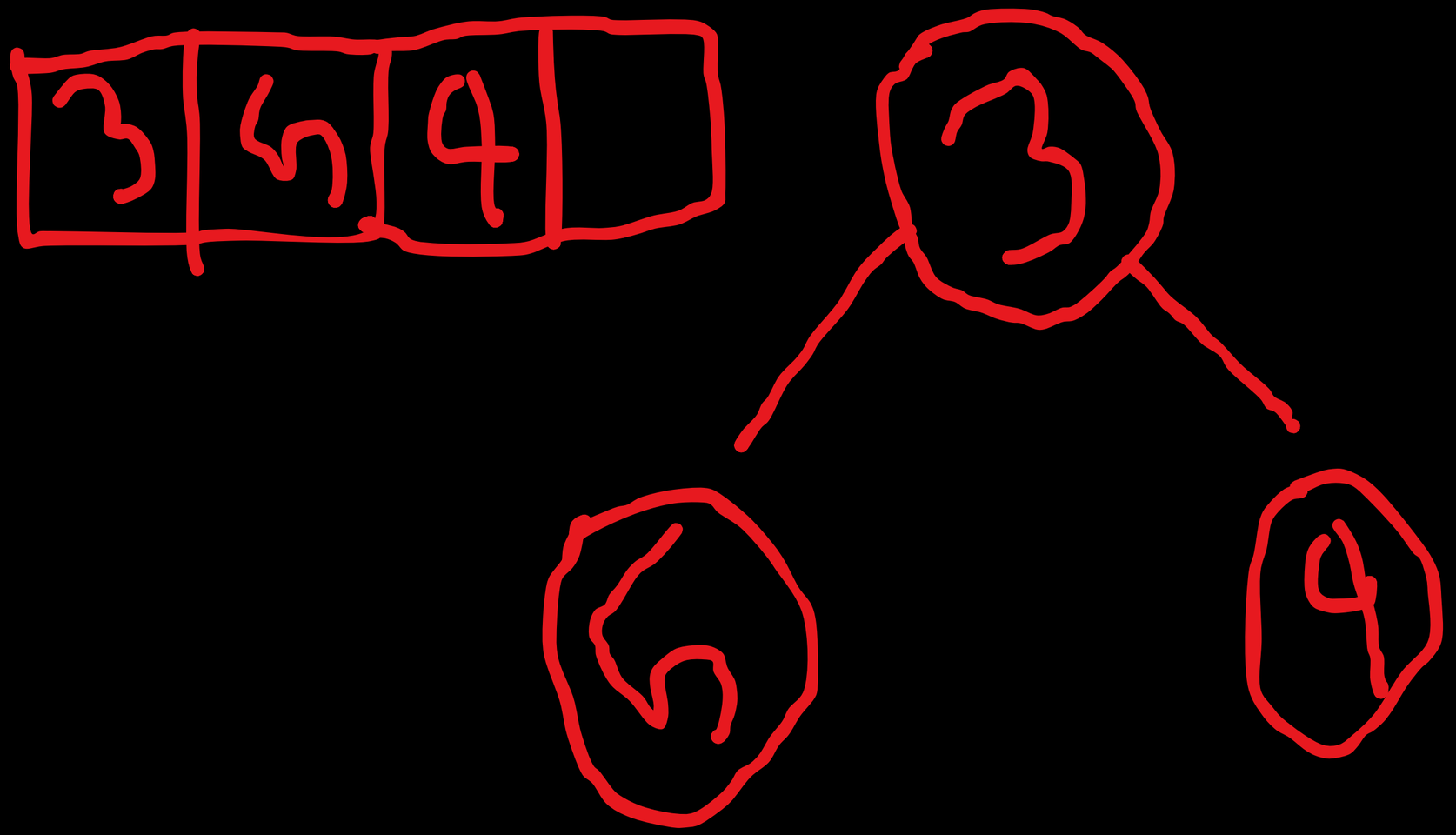


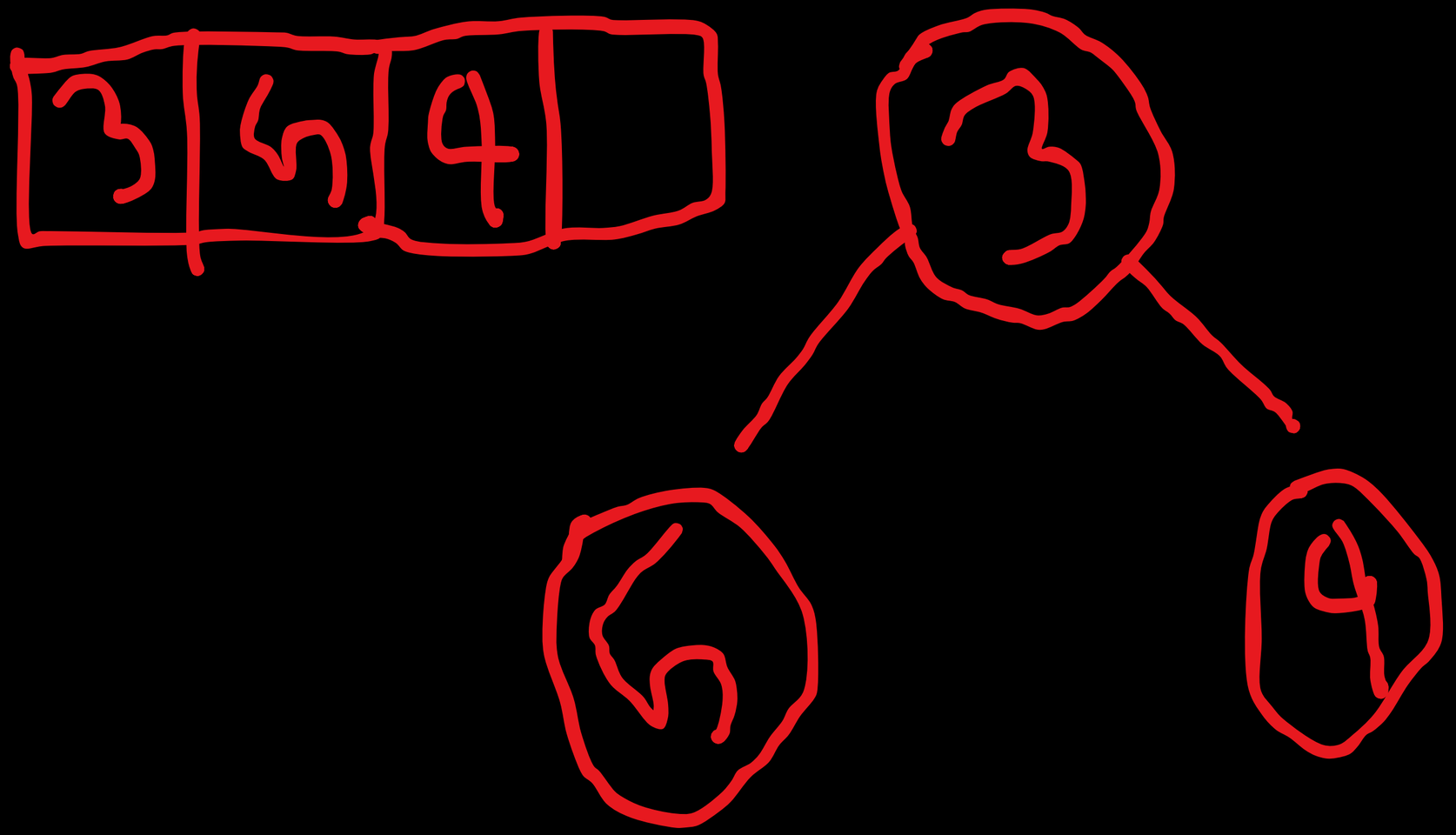


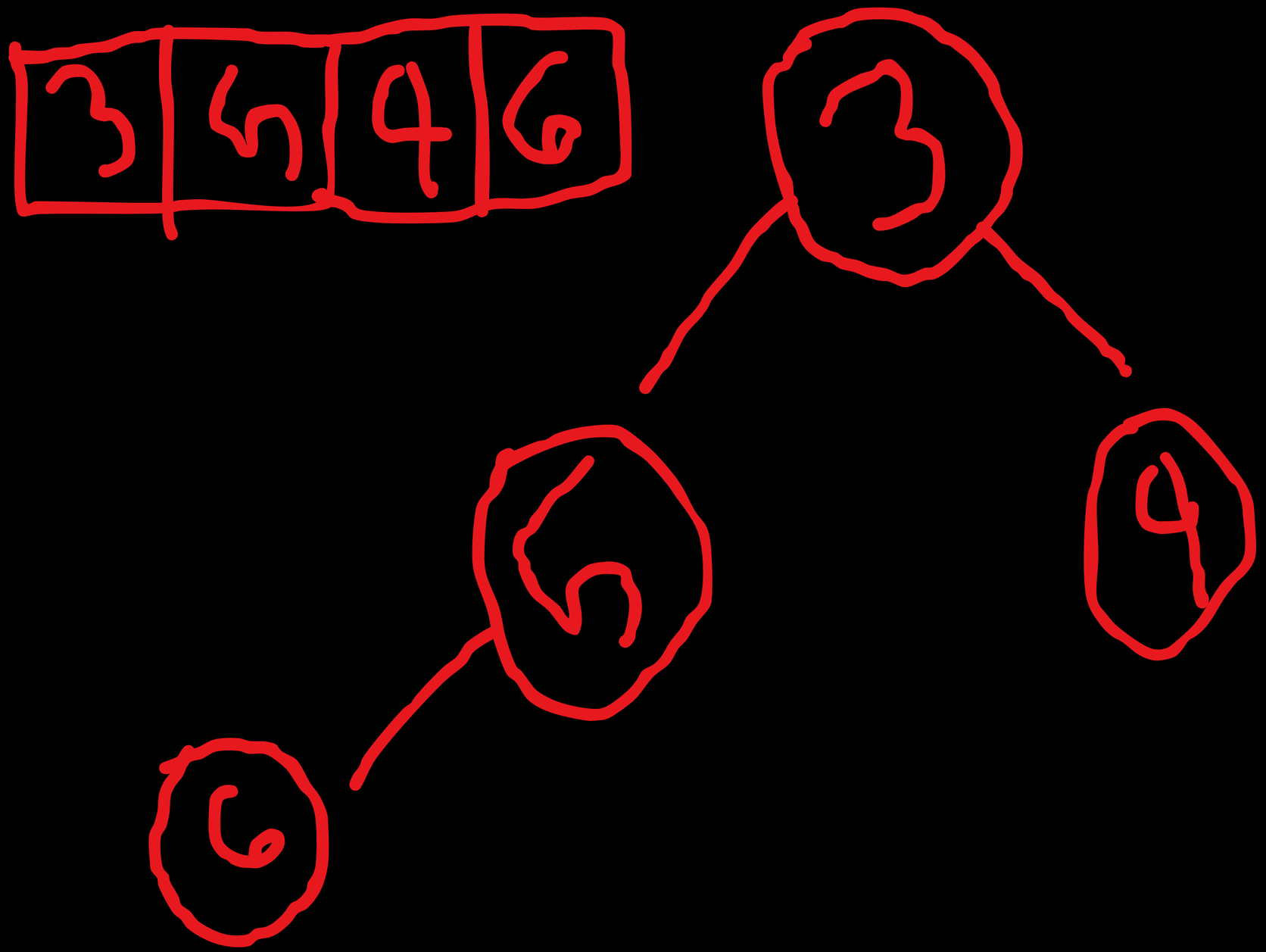


Heapify the array

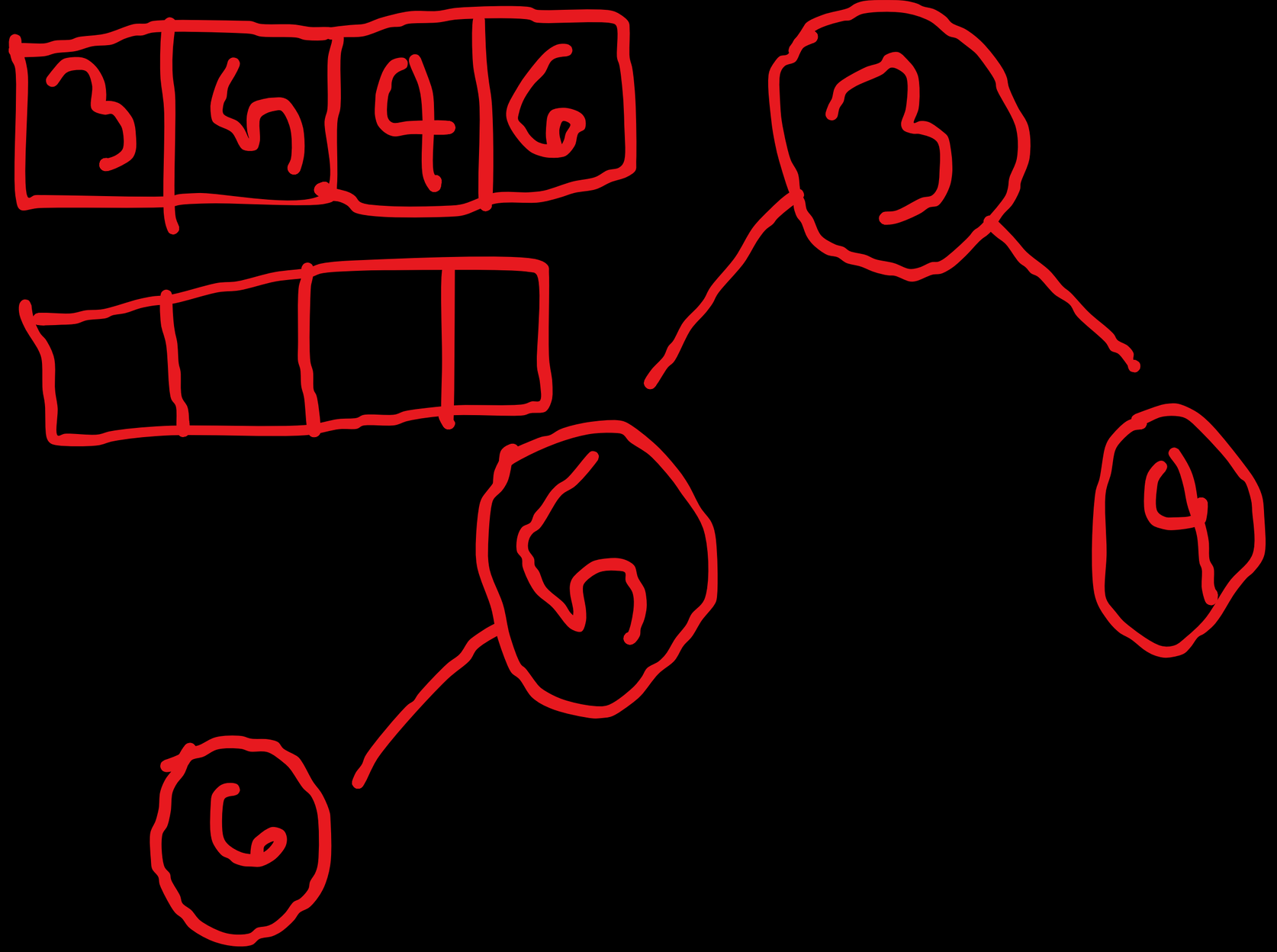


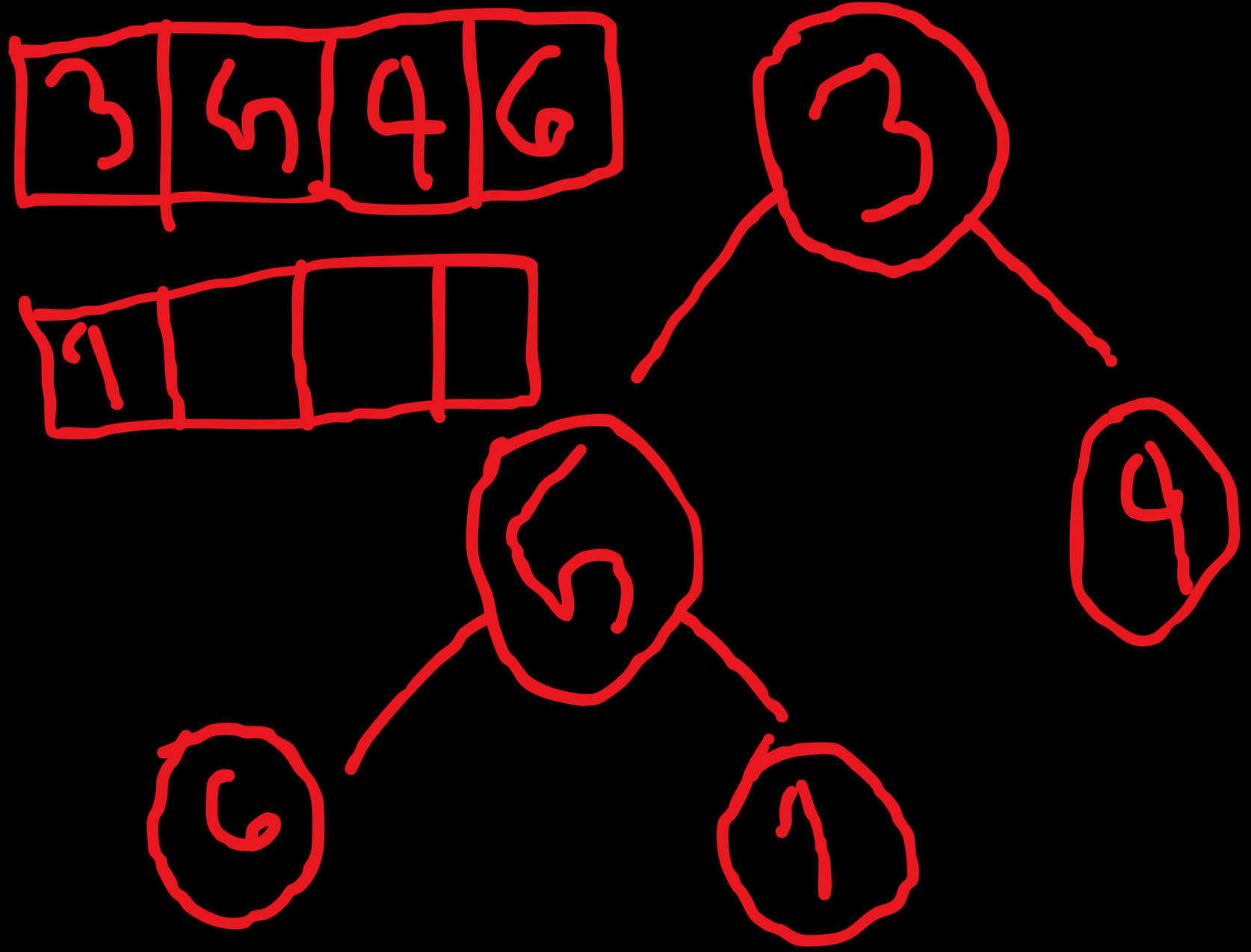

3
6
4
3
6
9

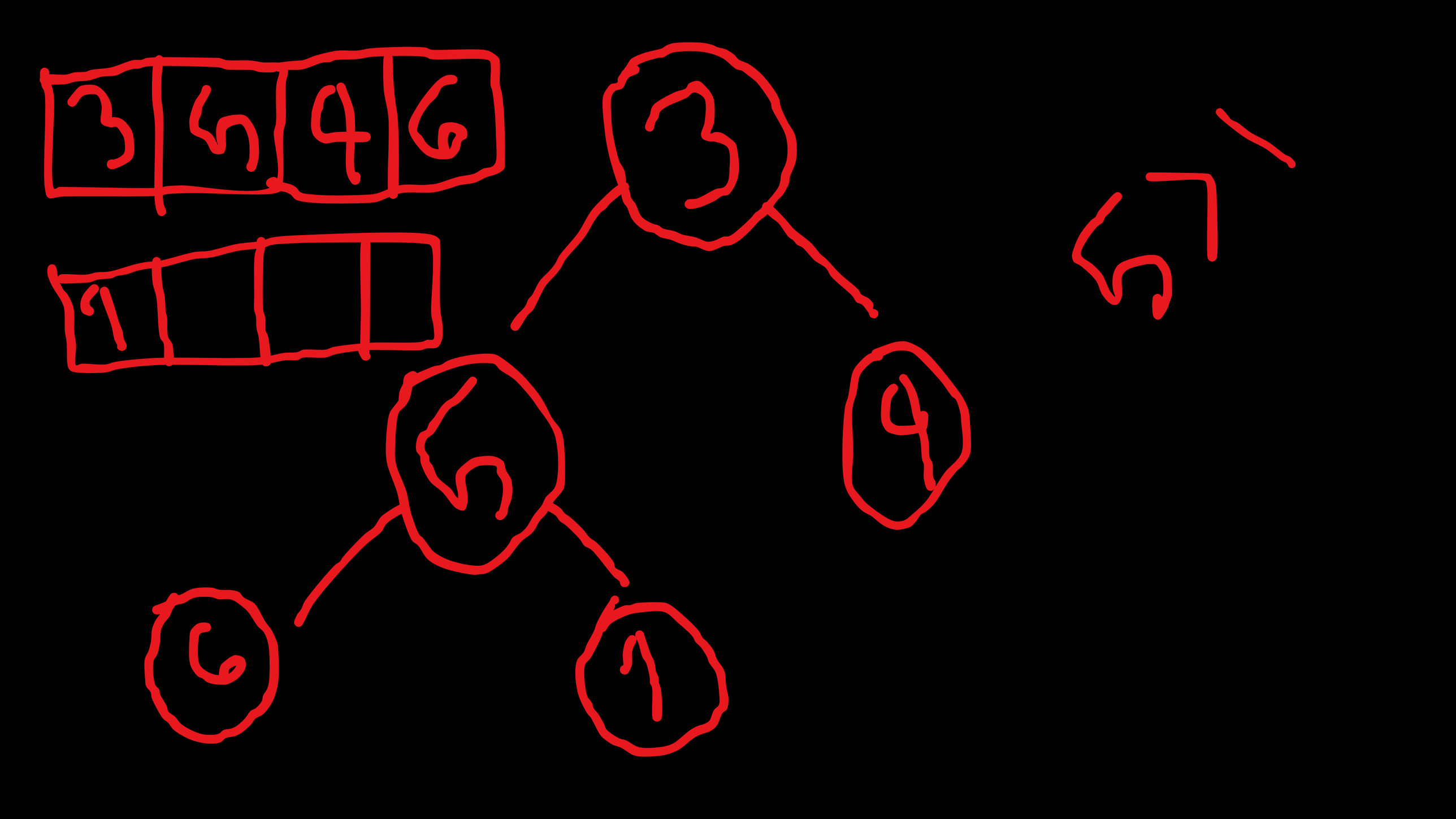

3
6
4
3
6
4

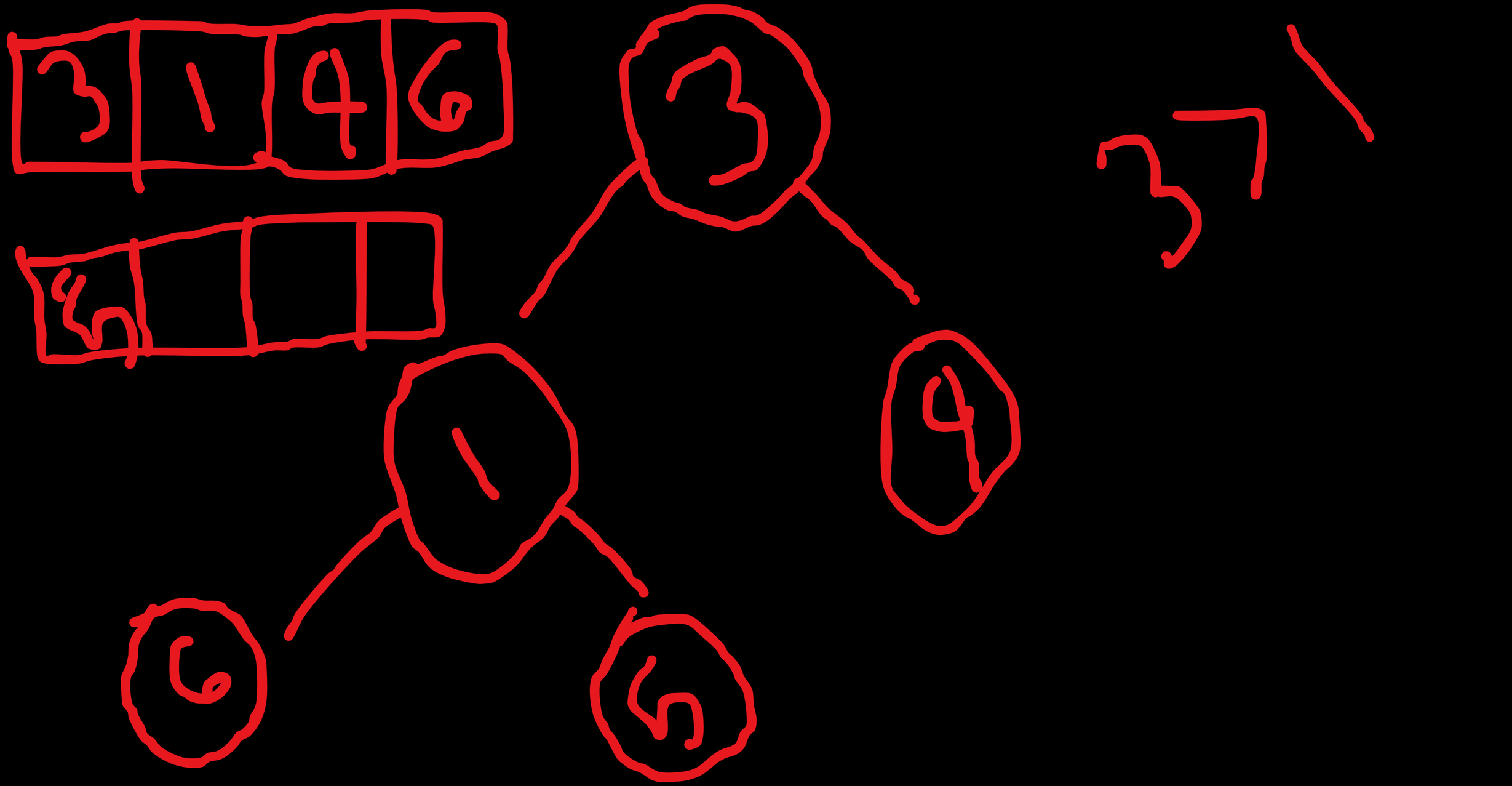

3
5
4
6
3
5
9
6

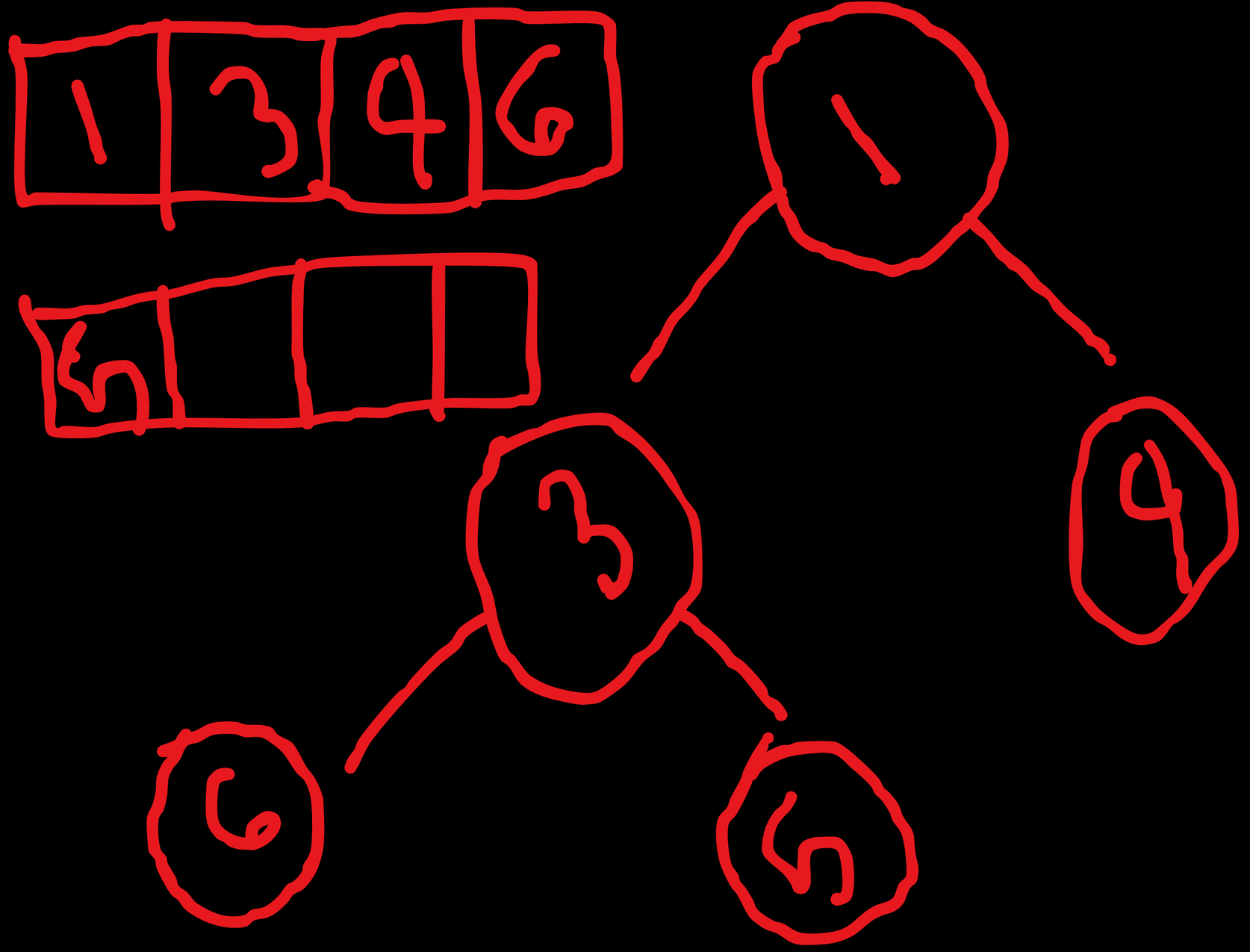
3 5 4 6
3
5
4
6

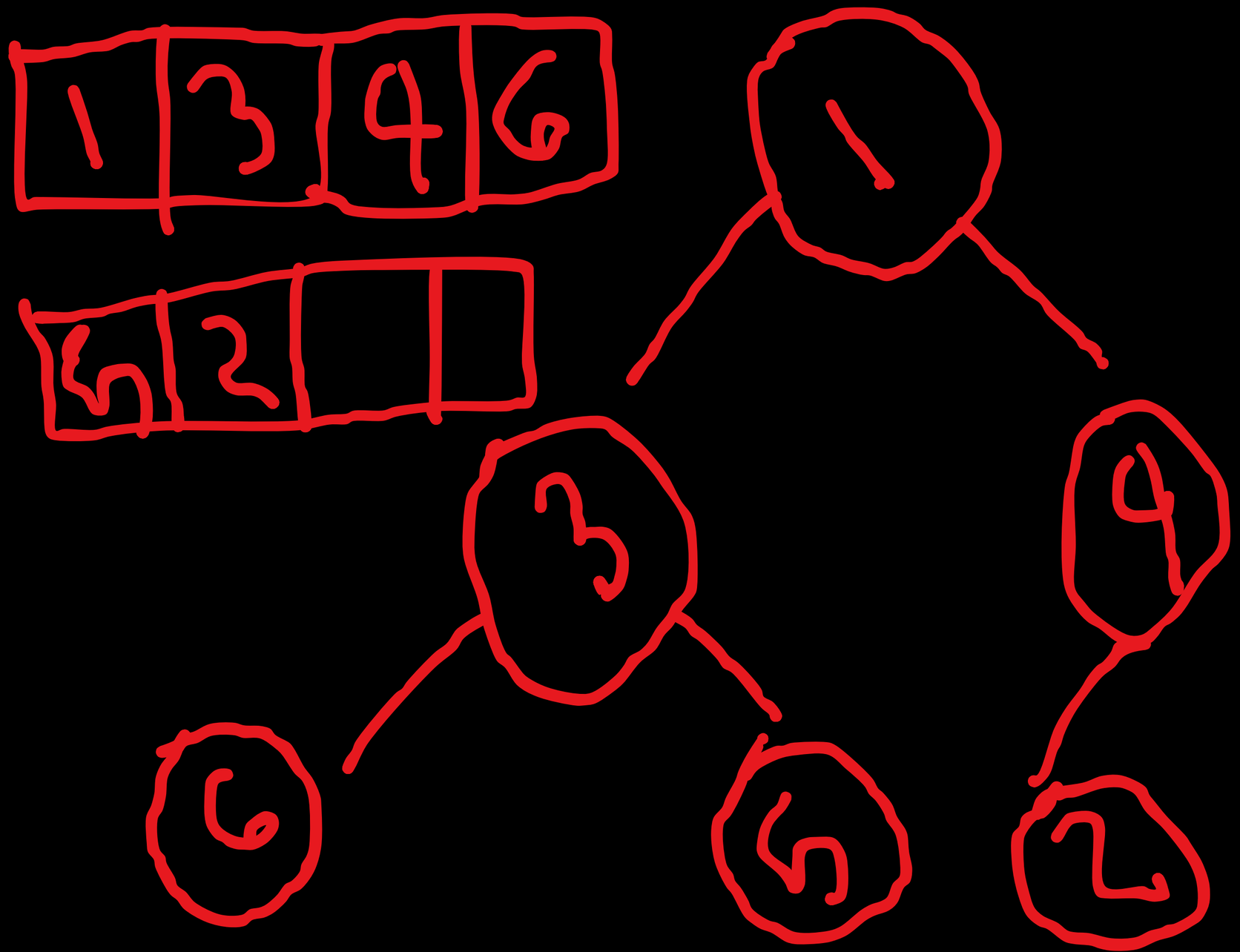

3
5
4
6
3
5
9
6

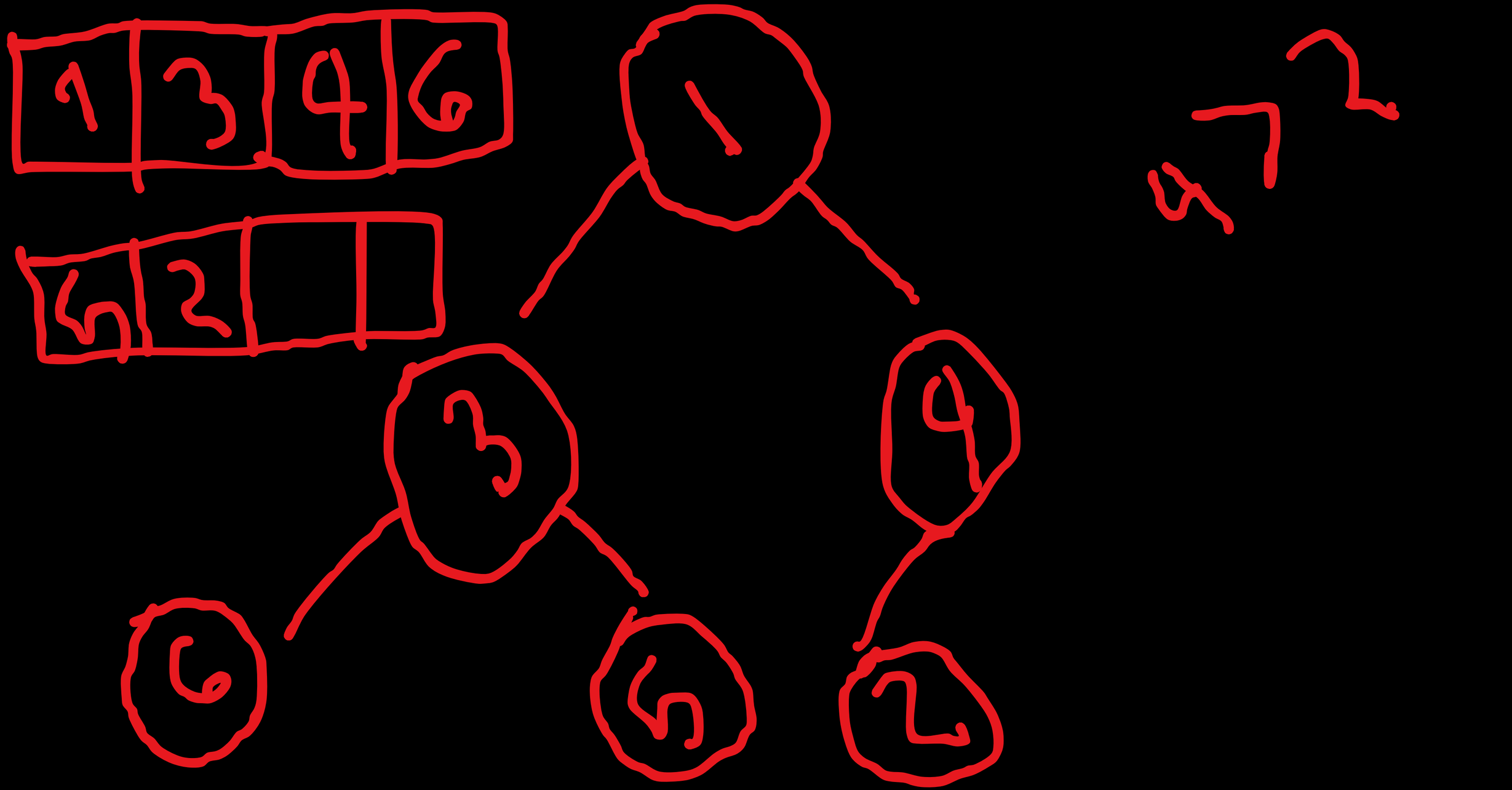

3 5 4 6
1
3
6
9
6
1

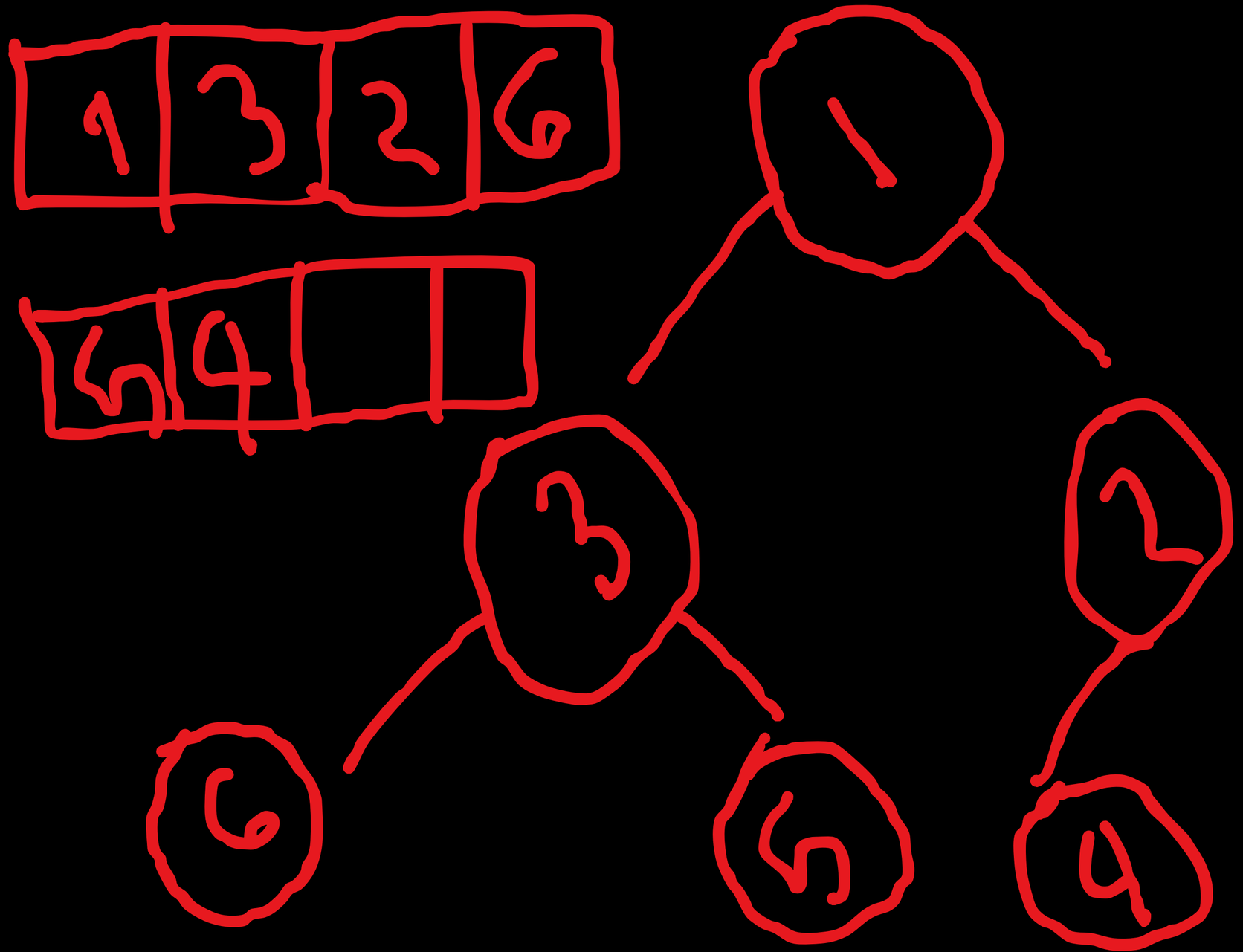

3 5 4 6
1
3
5
9
6
1

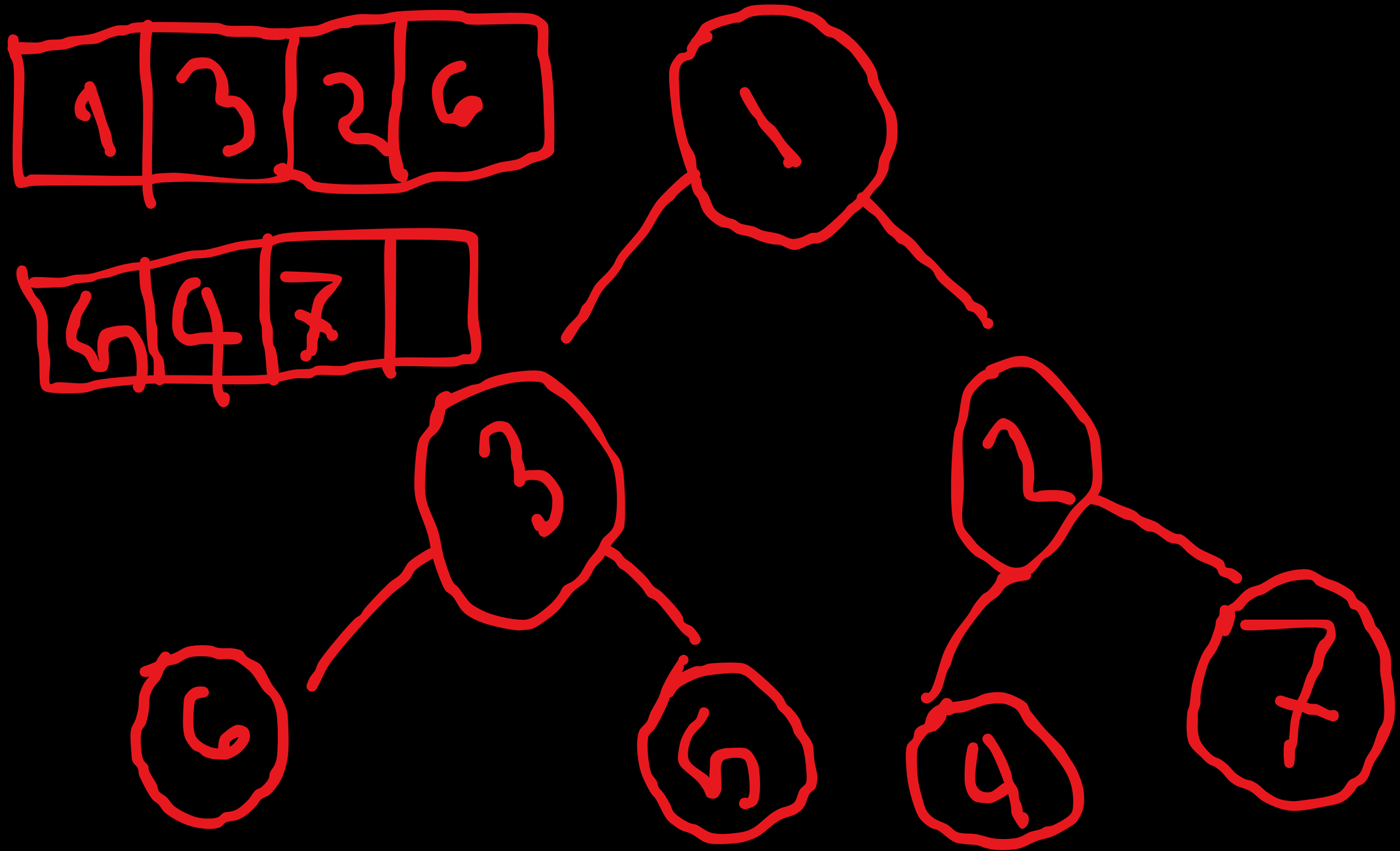

3 1 4 6
3
1
9
6
5

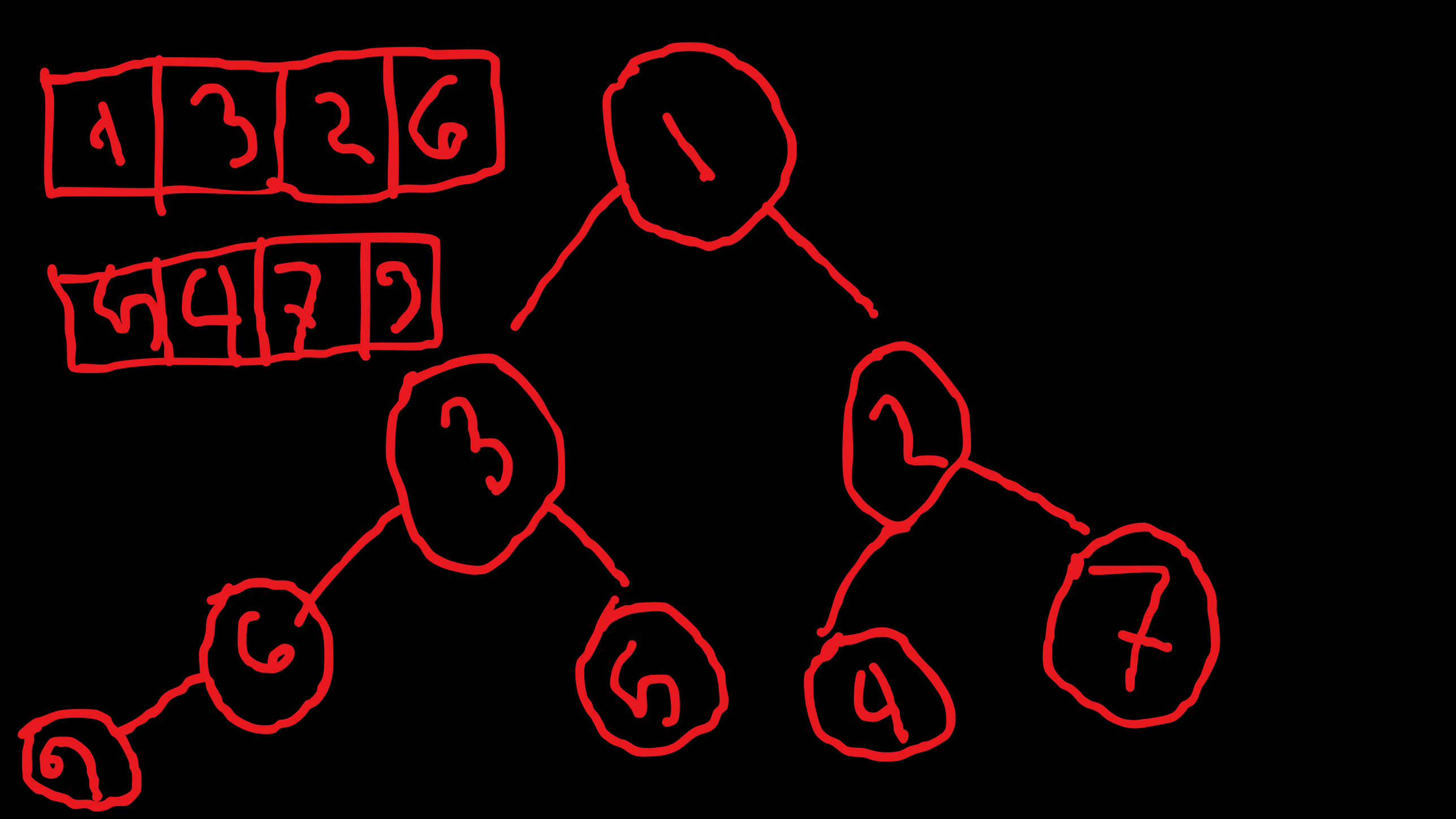

1 3 4 6
5
1
3
9
6
5

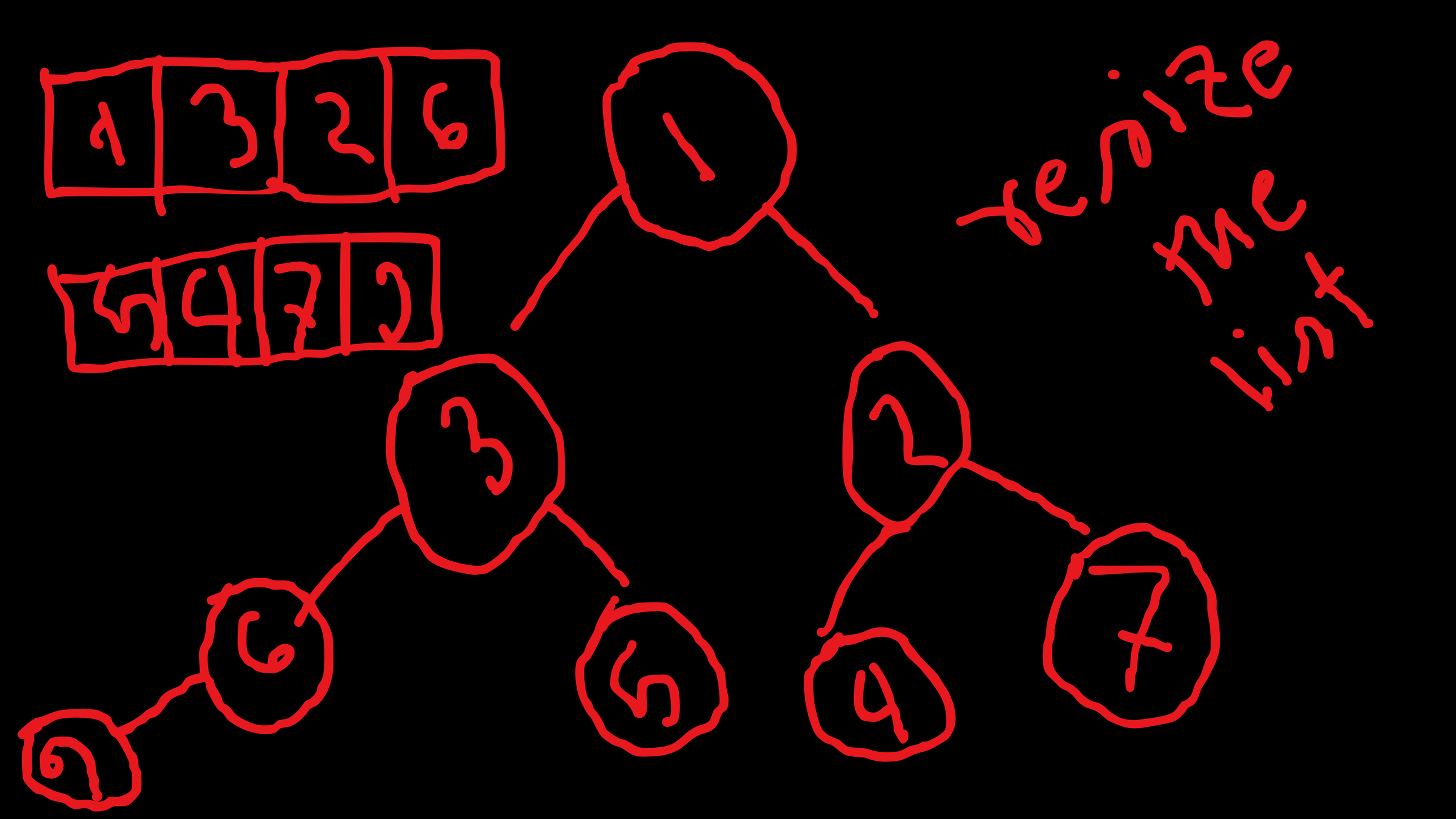

1 3 4 6
6 2
1
3
4
6
5
2

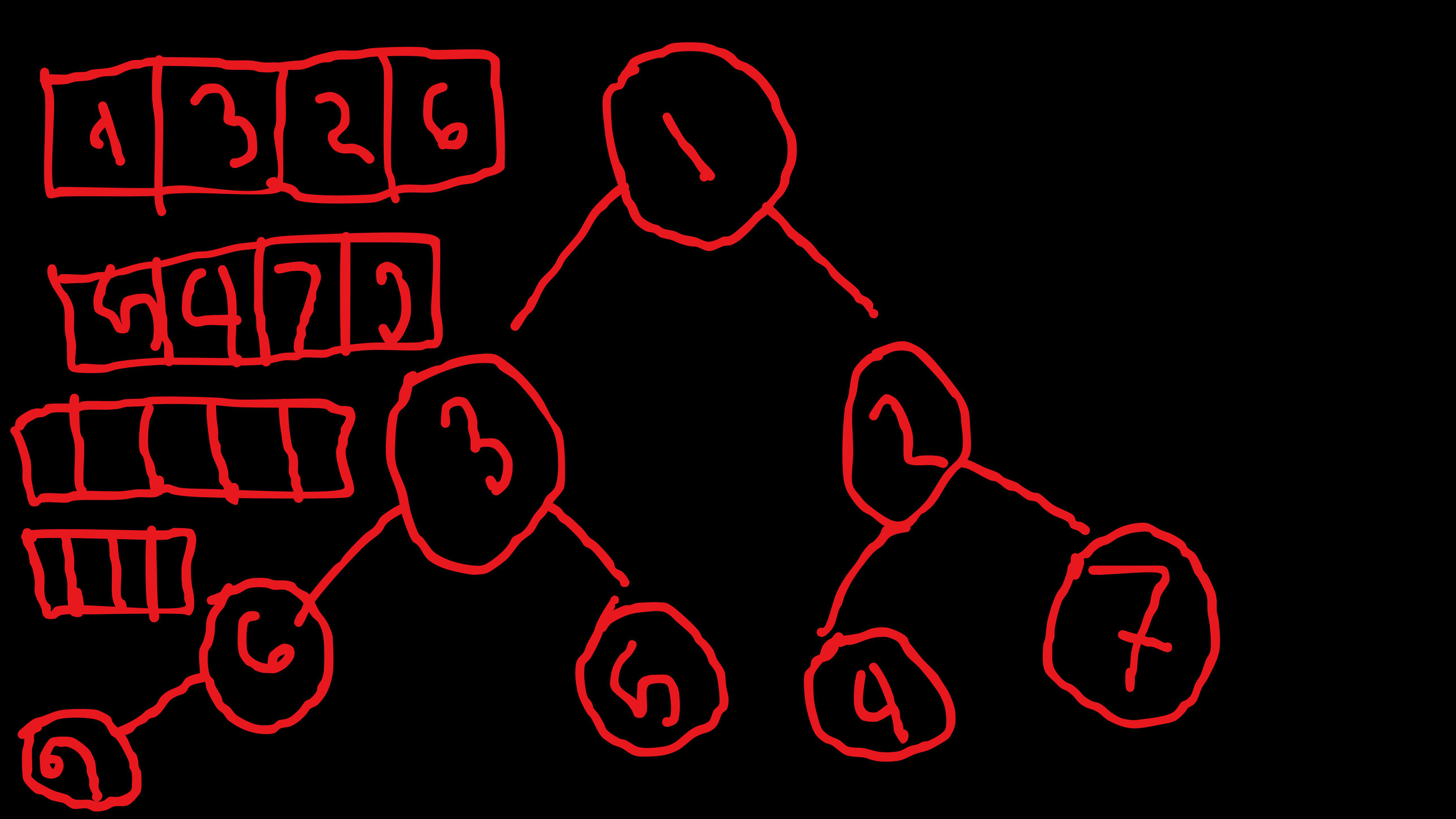

1 3 4 6
6 2
1
3
4
6
5
2

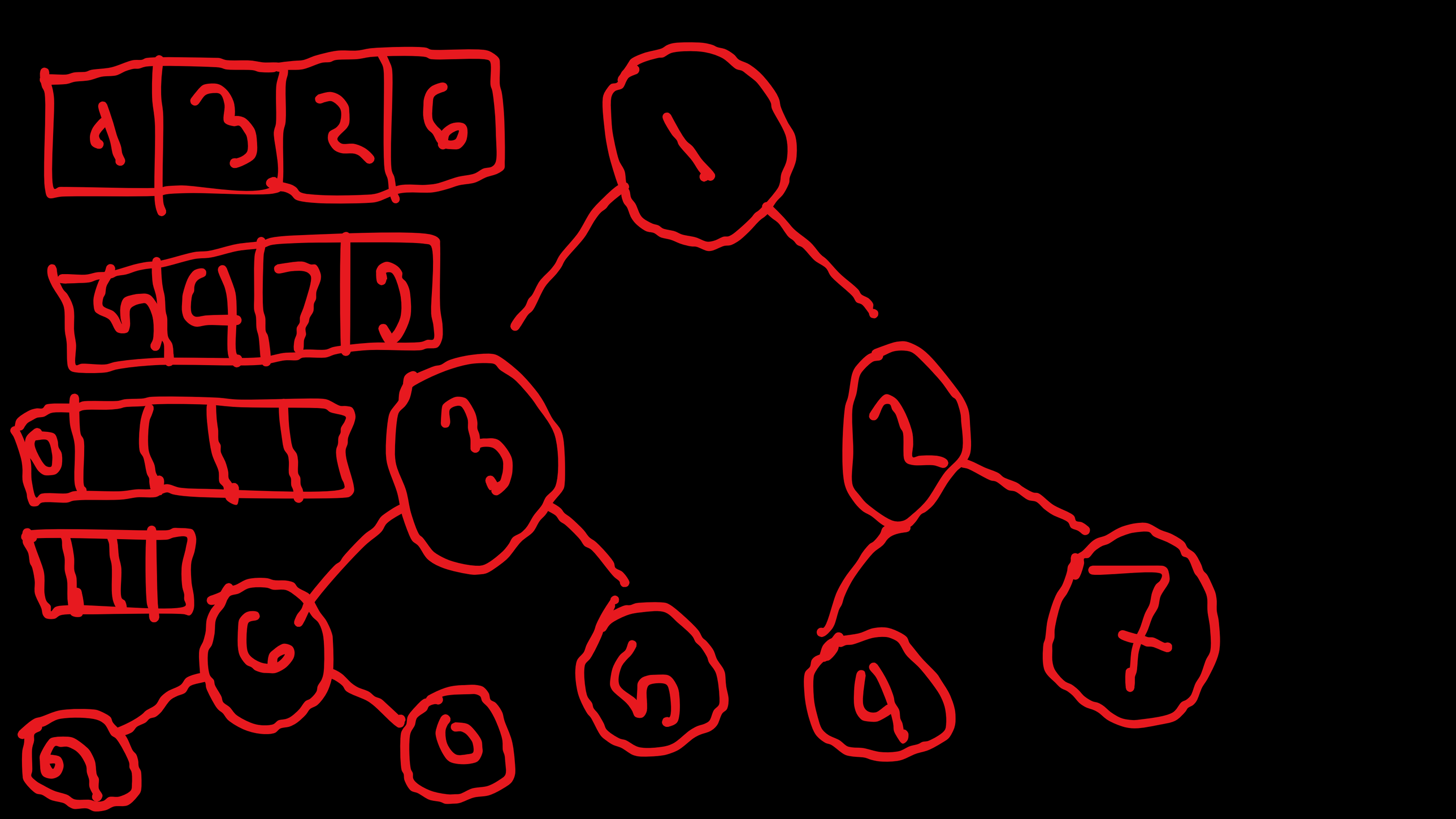

1 3 2 6
6 4
1
3
2
6
5
4

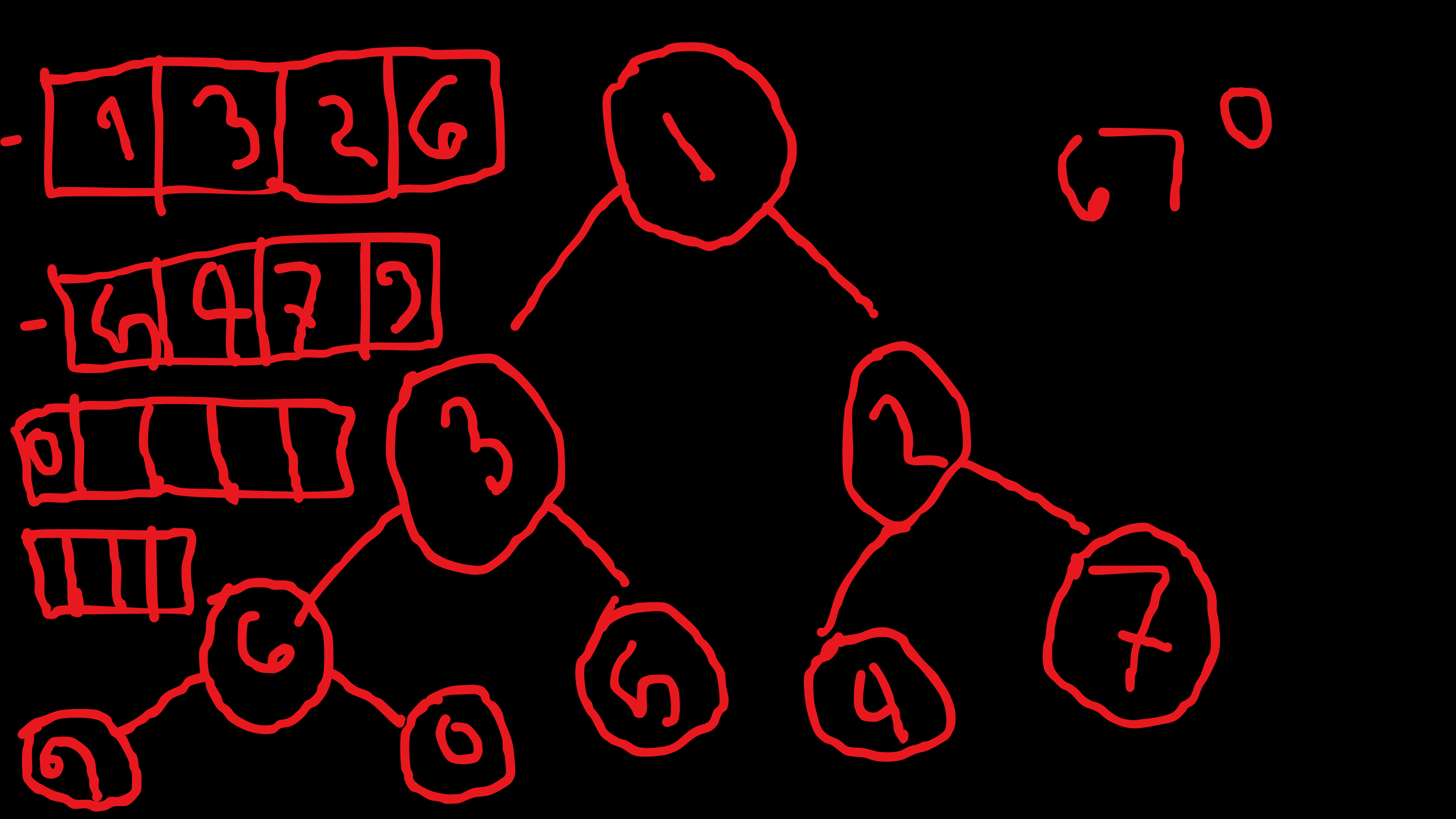

১ ৩ ২ ৬
৬ ৪ ৭
১
৩
২
৬
৫
৪
৭

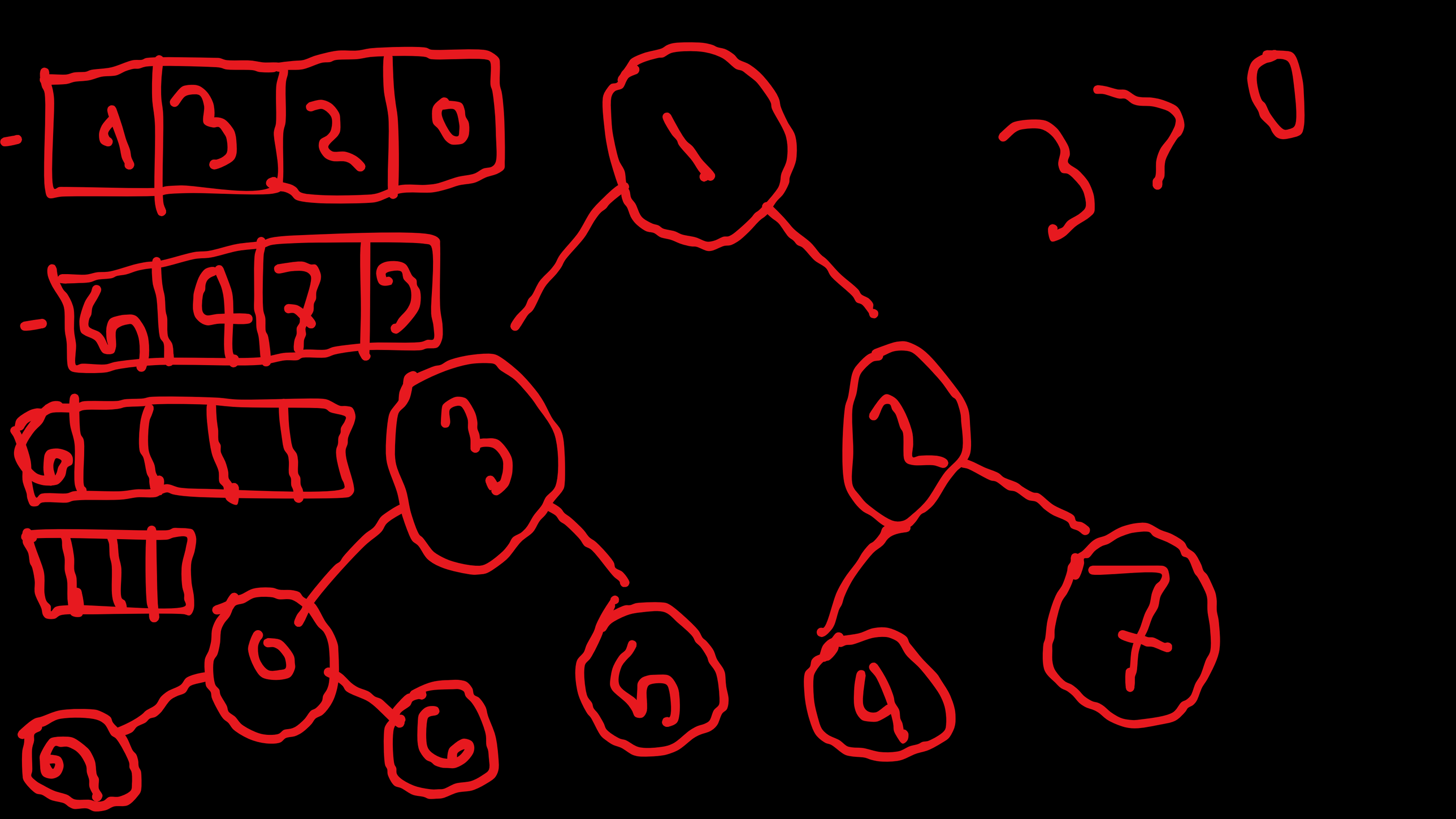

১ ৩ ২ ৬
৪ ৫ ৭ ৩
1
3
2
6
5
4
7

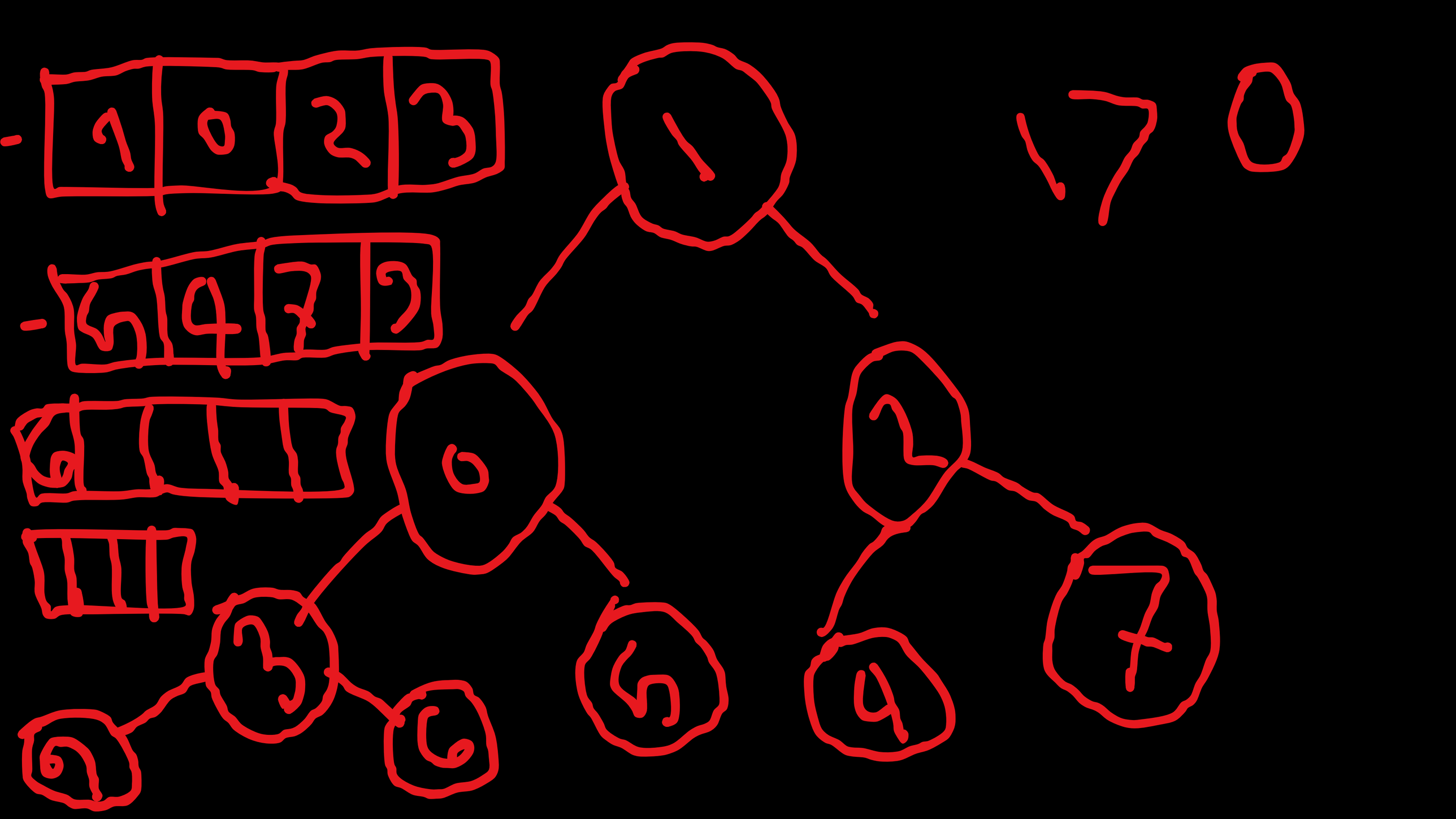

1 3 2 6
1 3 2 6 5 4 7
Serialize the list

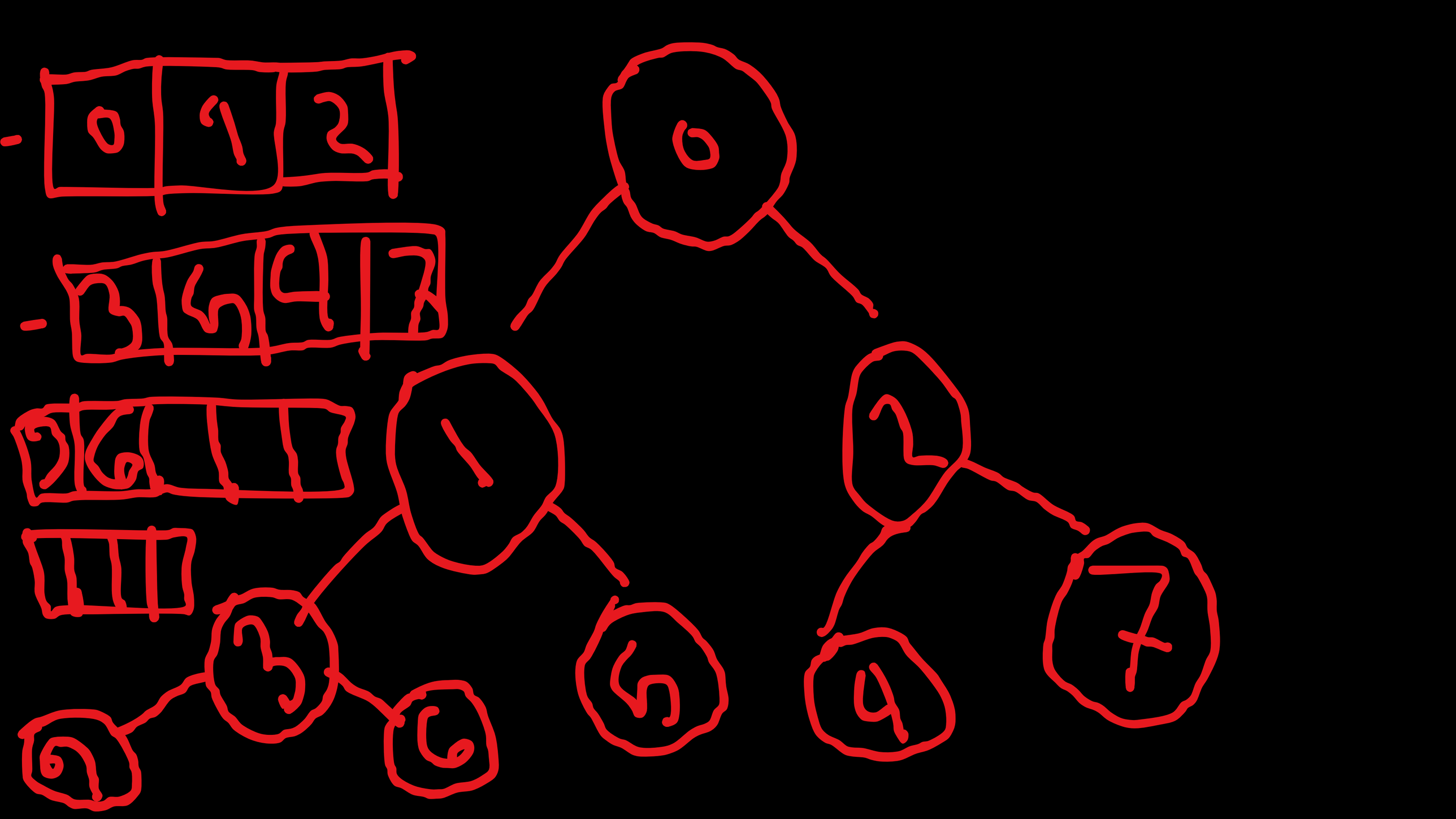

1 3 2 6
5 4 7 3
1
3
2
6
5
4
7
6

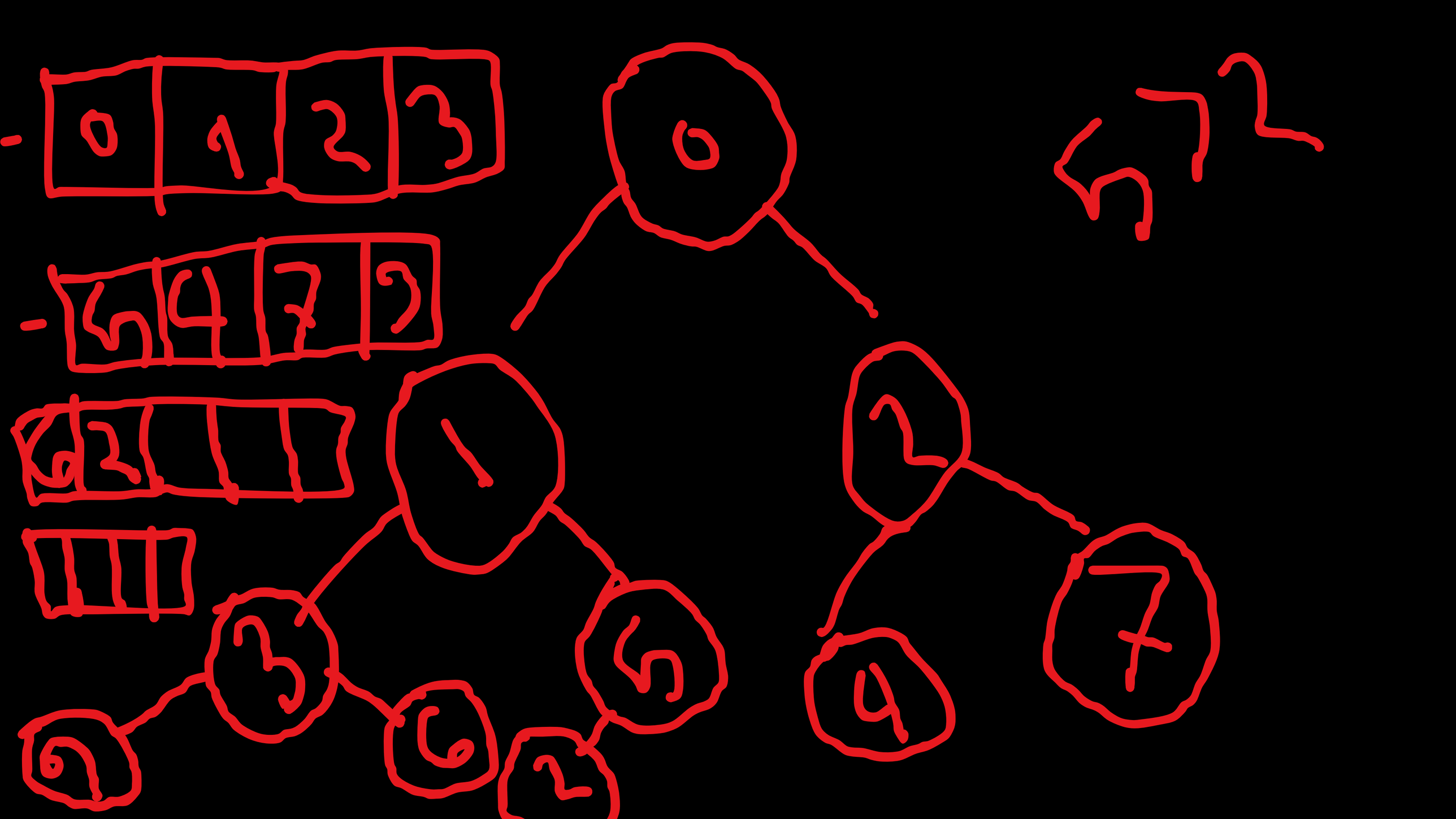

১ ৩ ২ ৬
৯ ৪ ৭ ১
০
১১
১
৩
২
৬
৫
৭
৬
৩
৫
৭

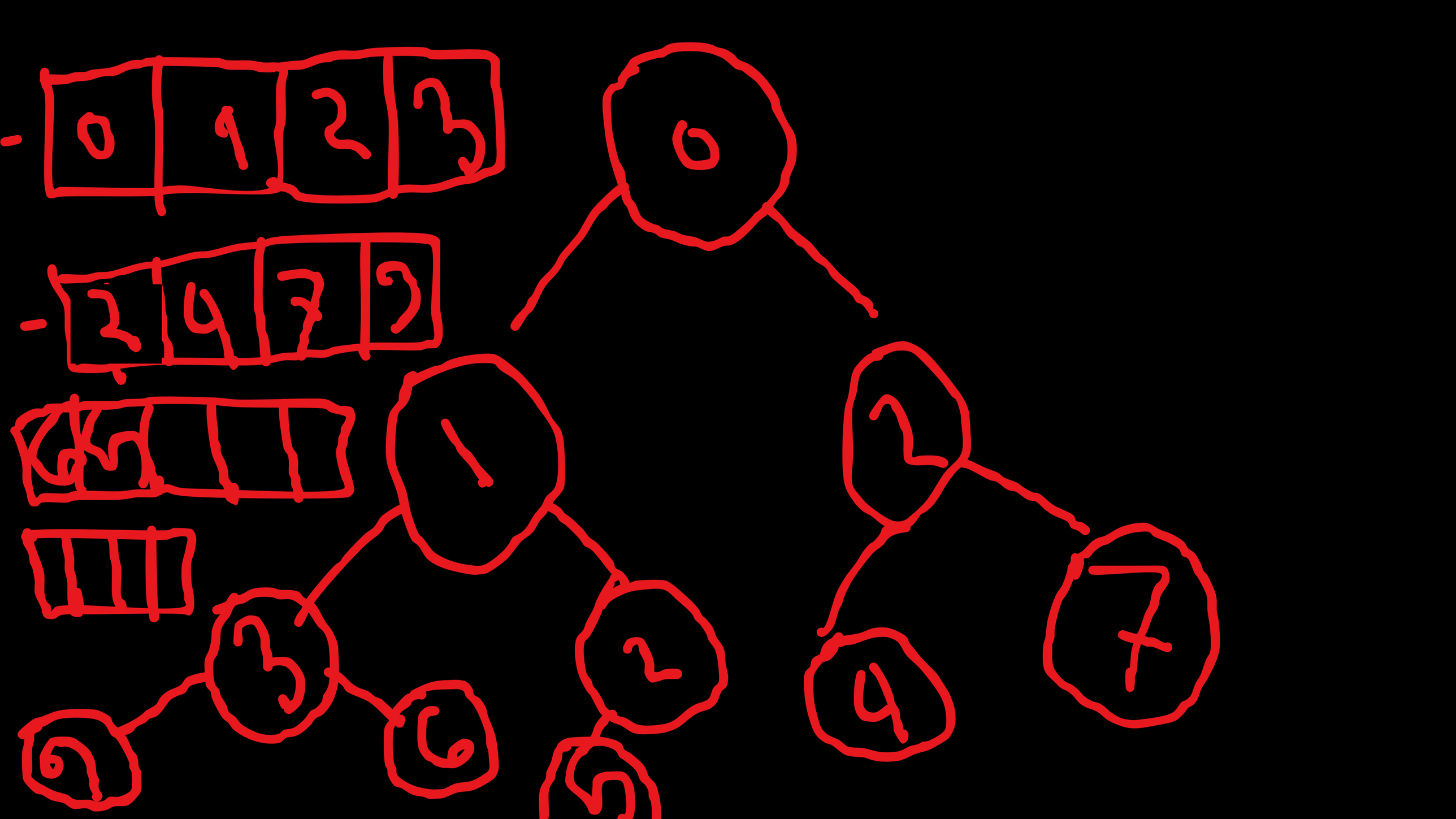


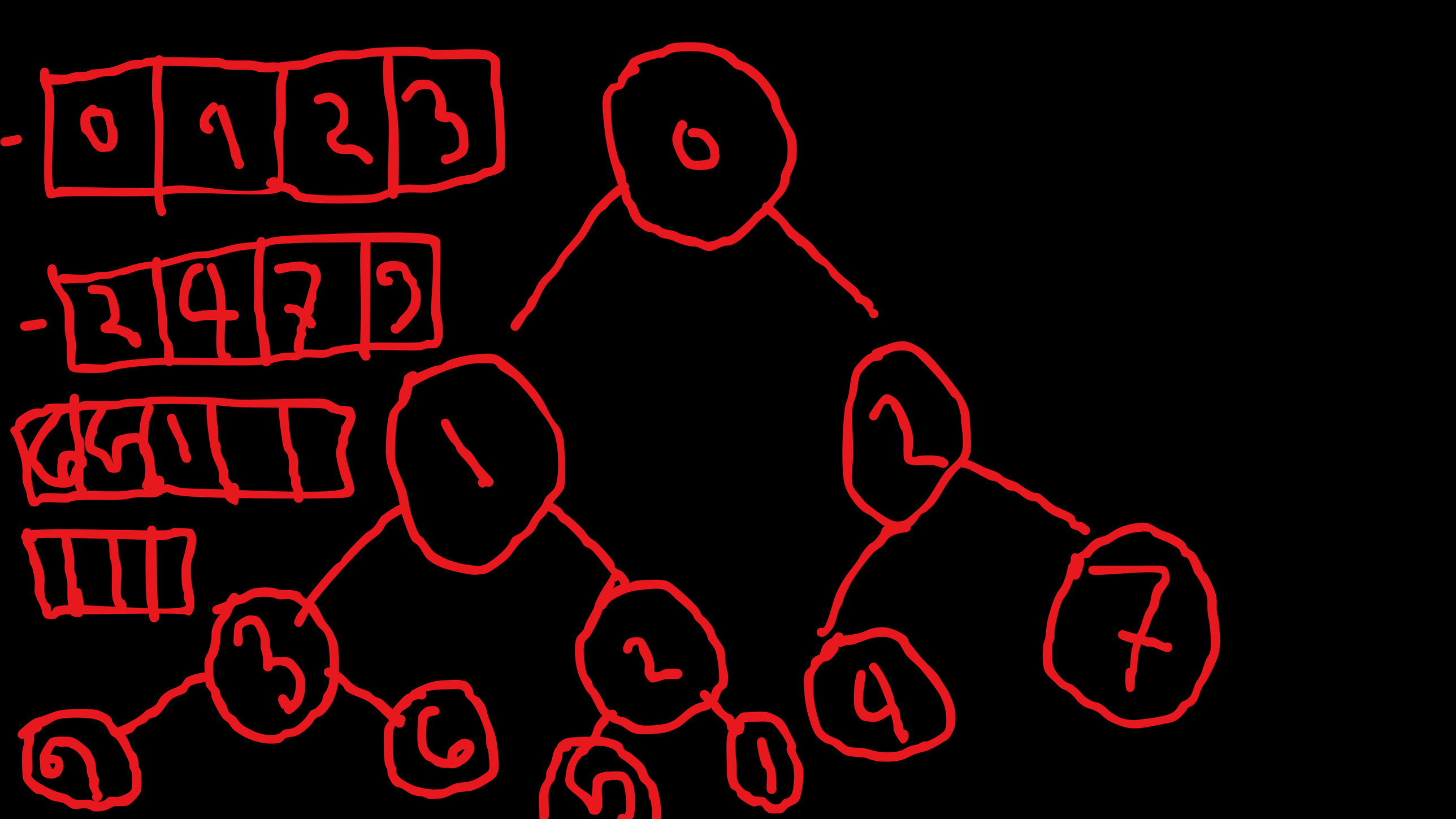


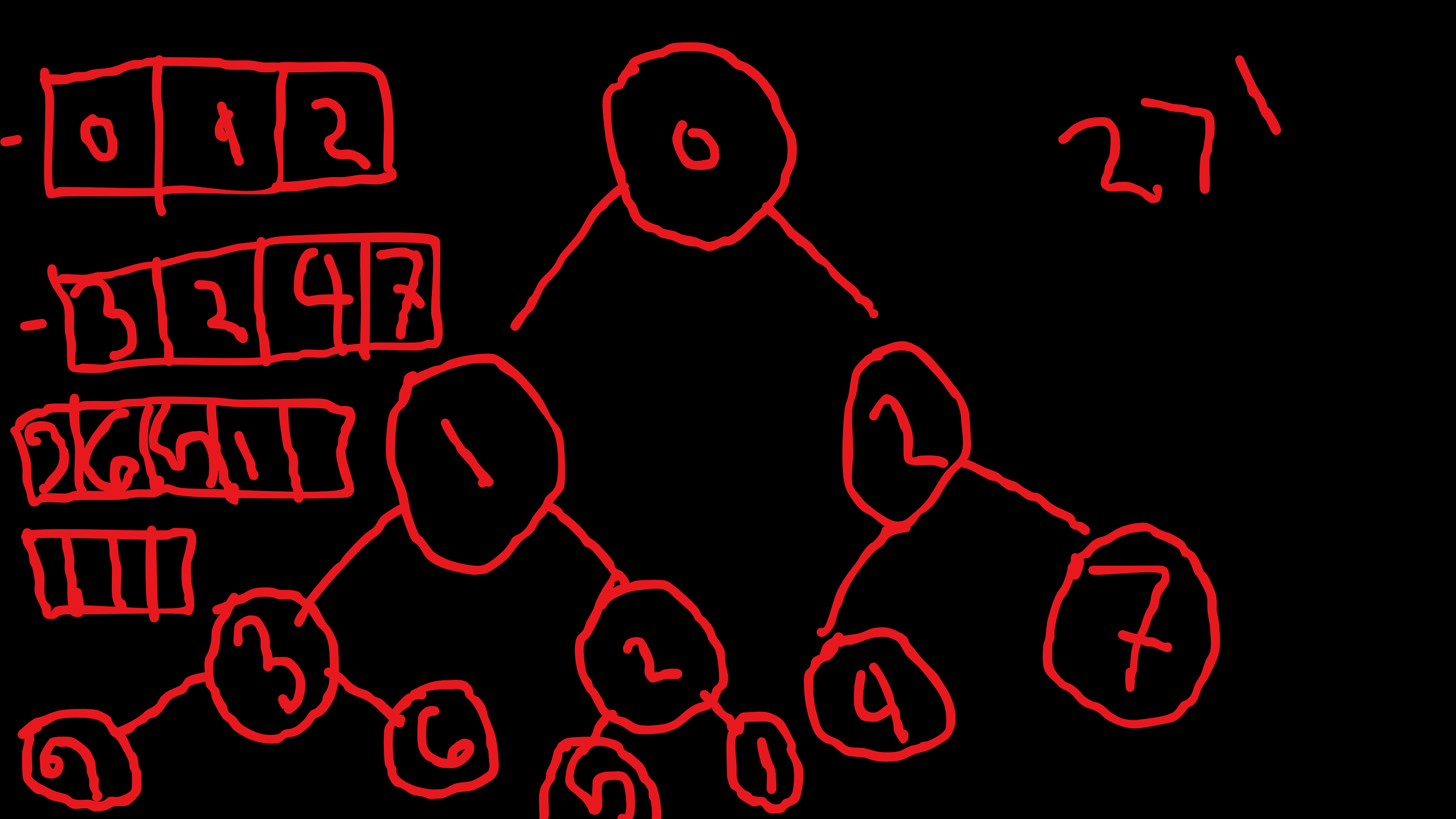

1 0 2 3
4 4 7 7
1
0
2
3
5
6
4
7

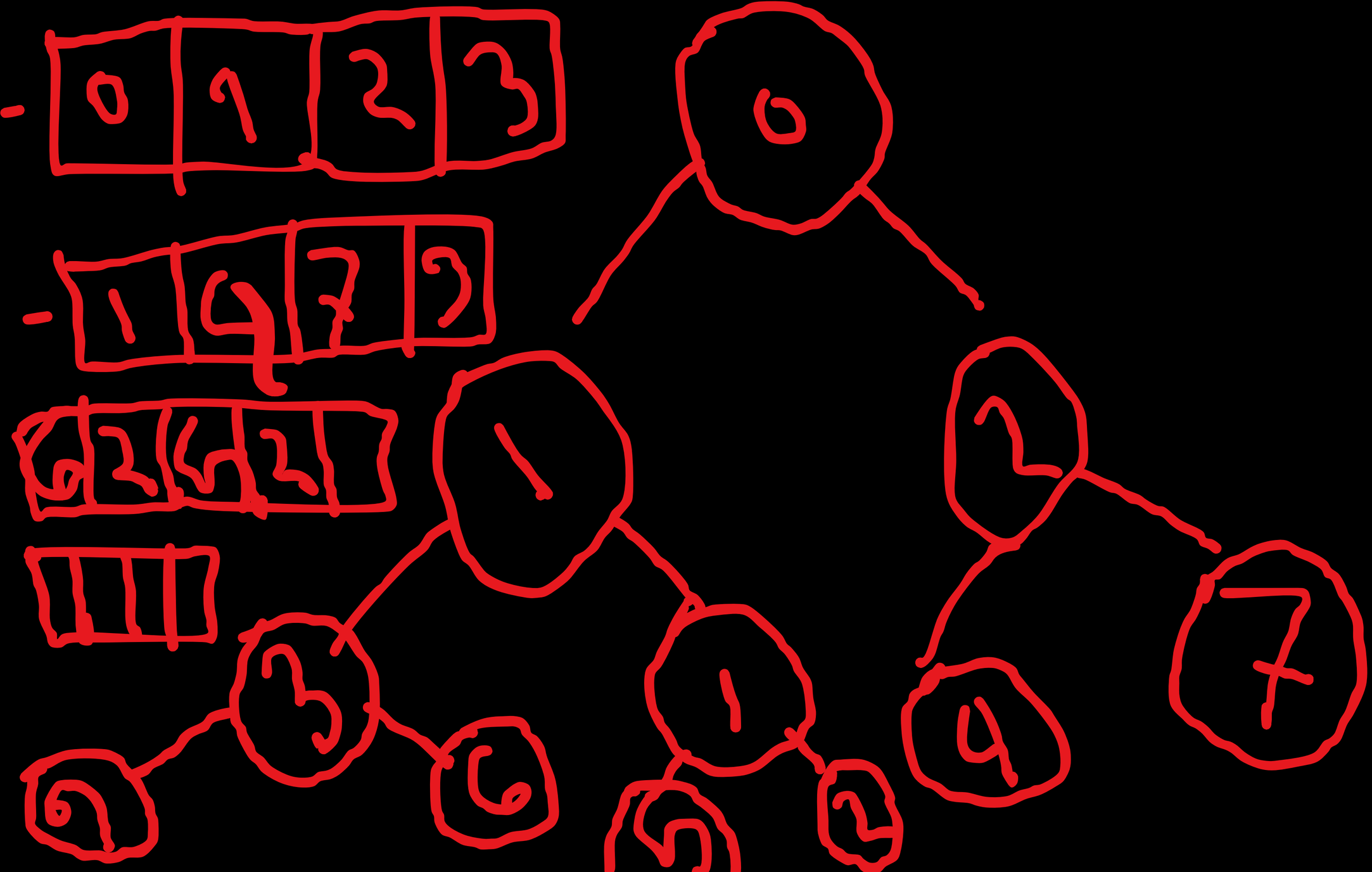

0 1 2
3 6 4 7
0
1
2
3
5
4
7
6


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


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




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



Now, try to solve this same example with the max heap.

Now, come for the deletion of the min element.

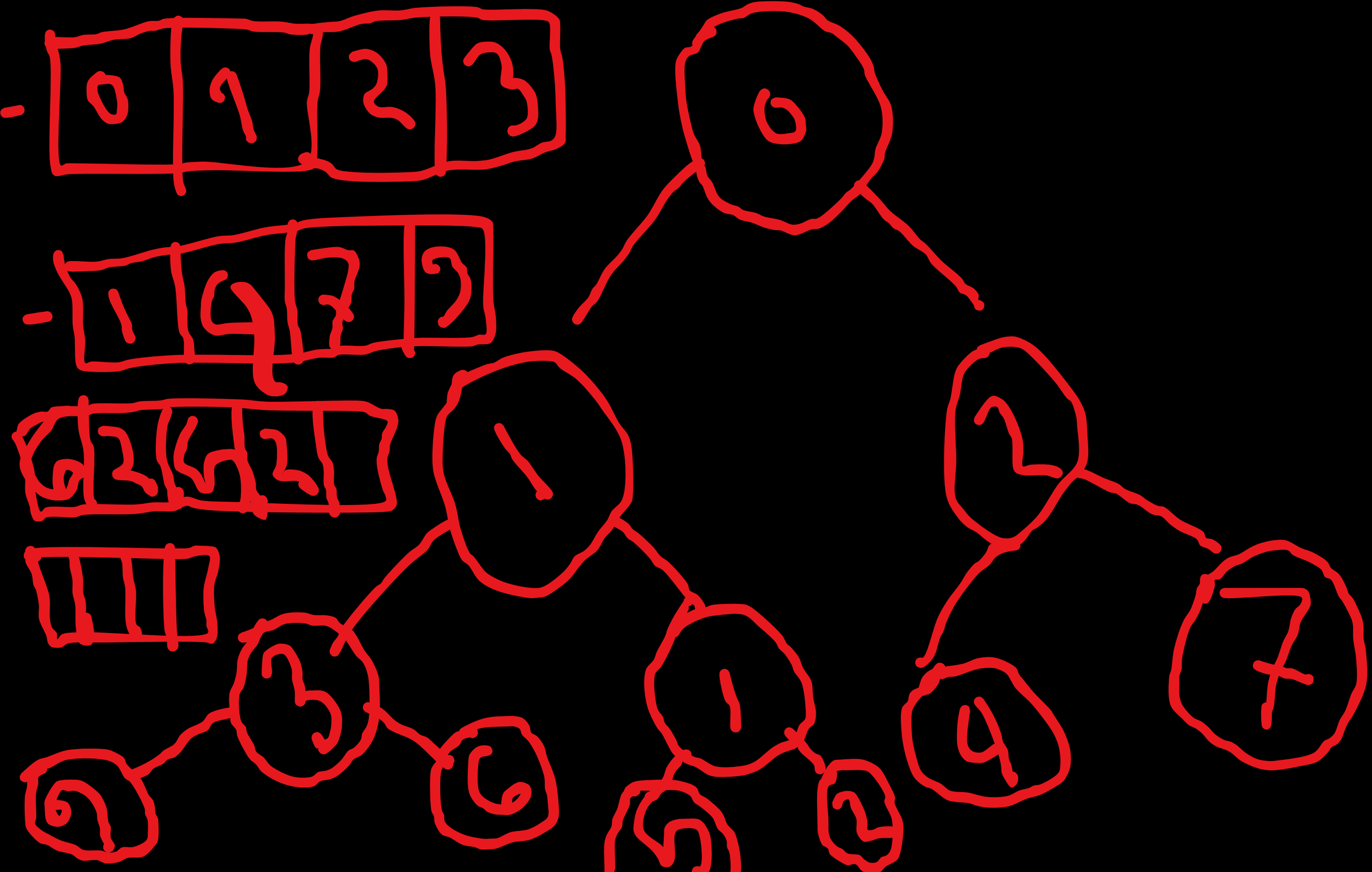
First exchange the root element with the end element of the list

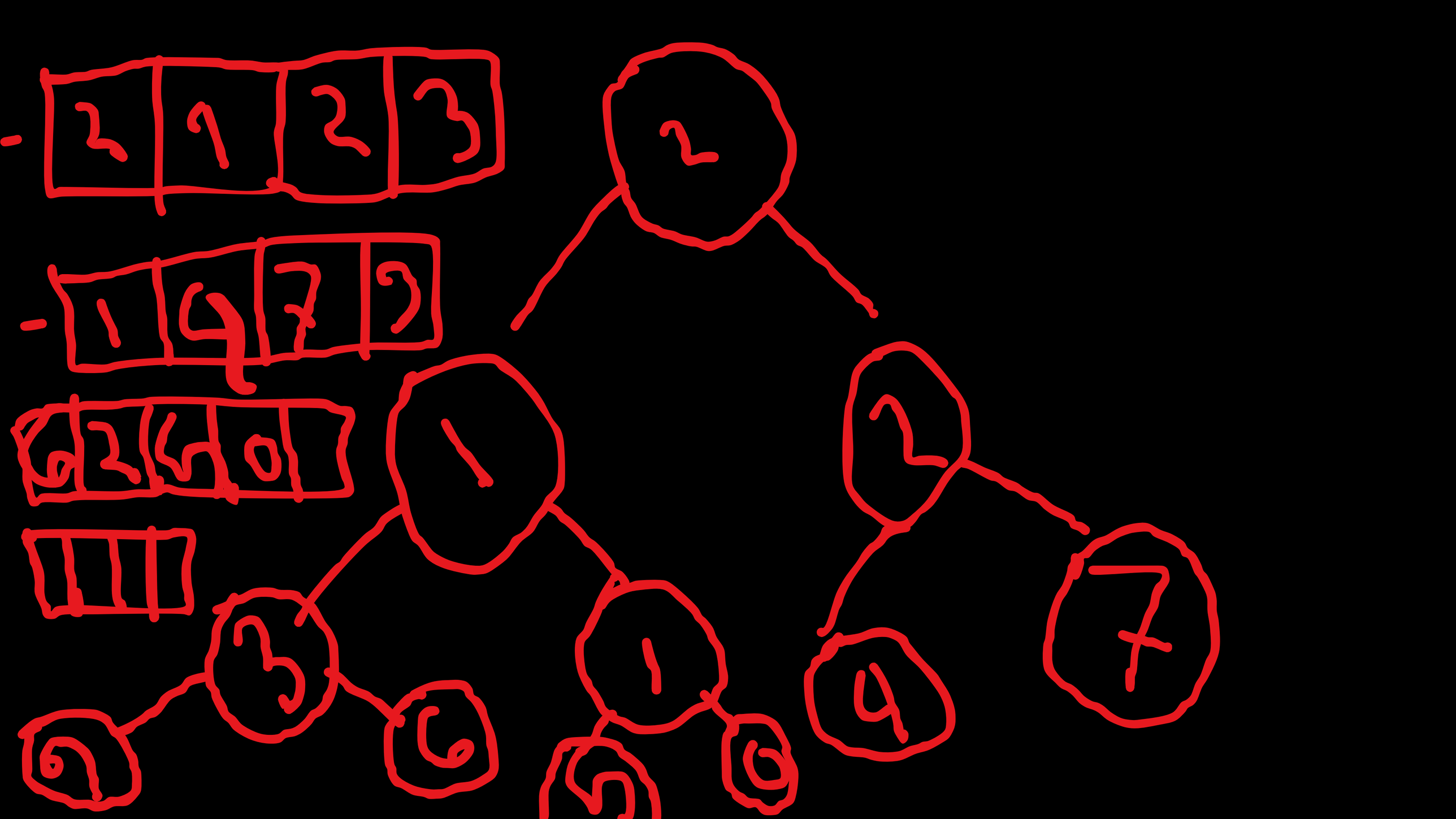
Then continue the heapify process from top to bottom

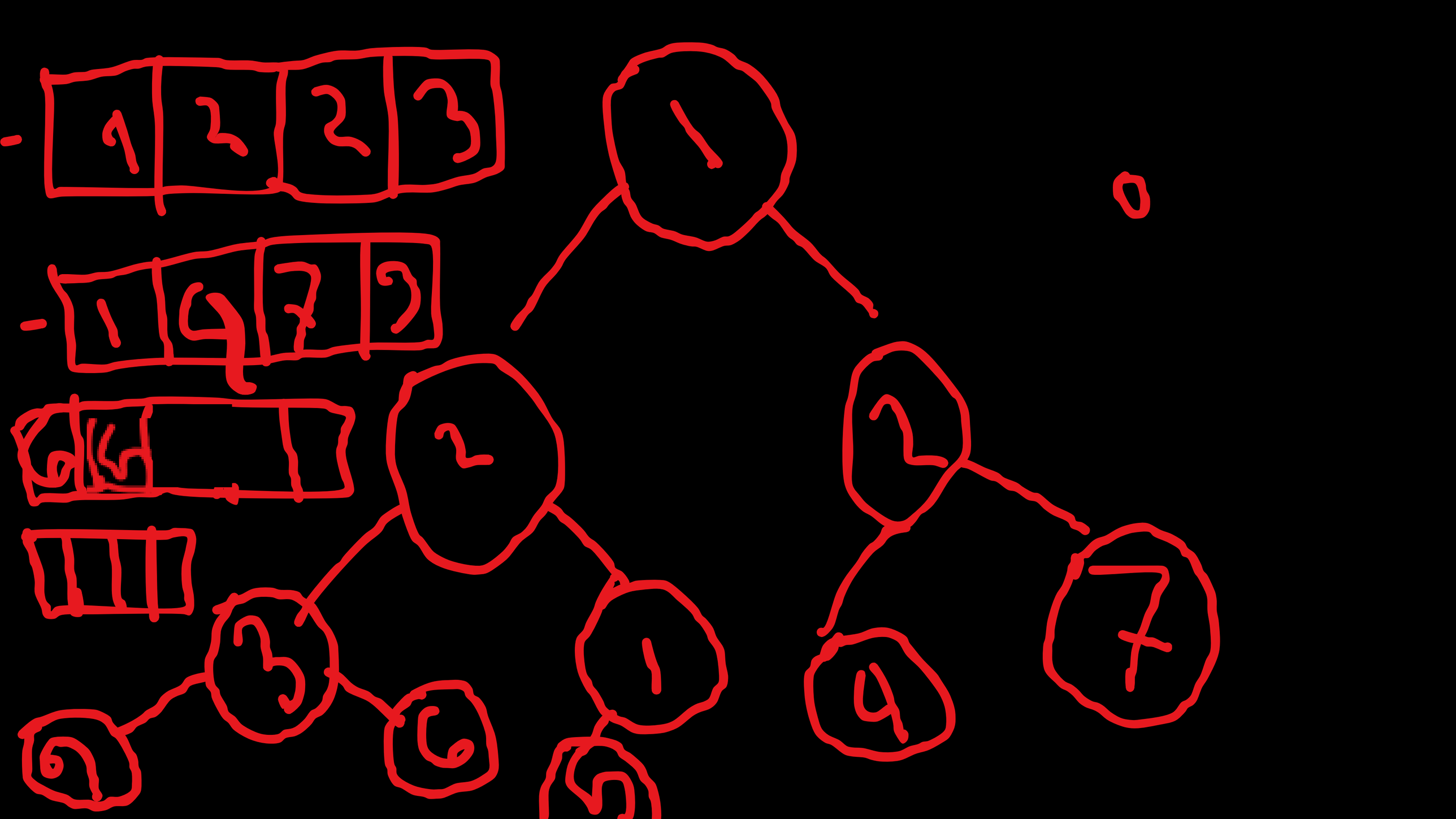
This process is called the downify.

Here from the root it try to find the next fit element for the root by comparing with the left and right child. And finally set the fiter most element for the root.

Let’s solve an example.








here 1 is smaller
than 2

Now, come for searching, this is a simple linear search, take O(n) time.

Now, this is the time for heap sort.

Building heap -> O(n)

Heapify -> O(log(n))

Complexity :-

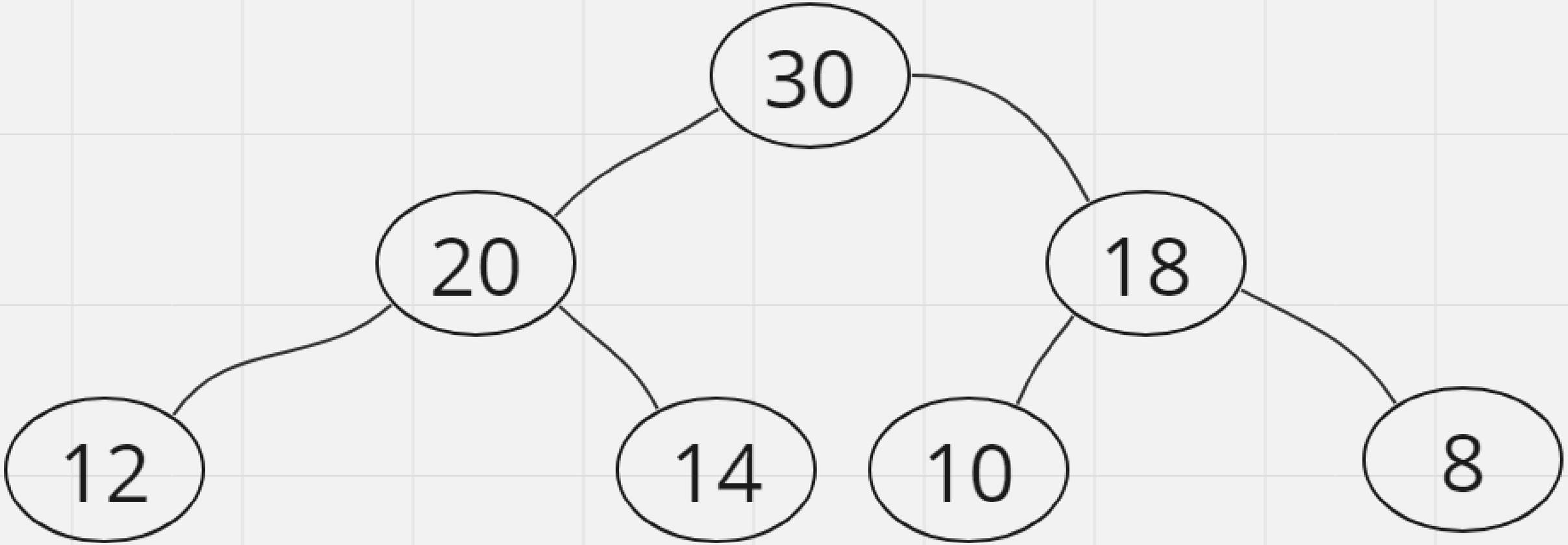
O(nlog(n))

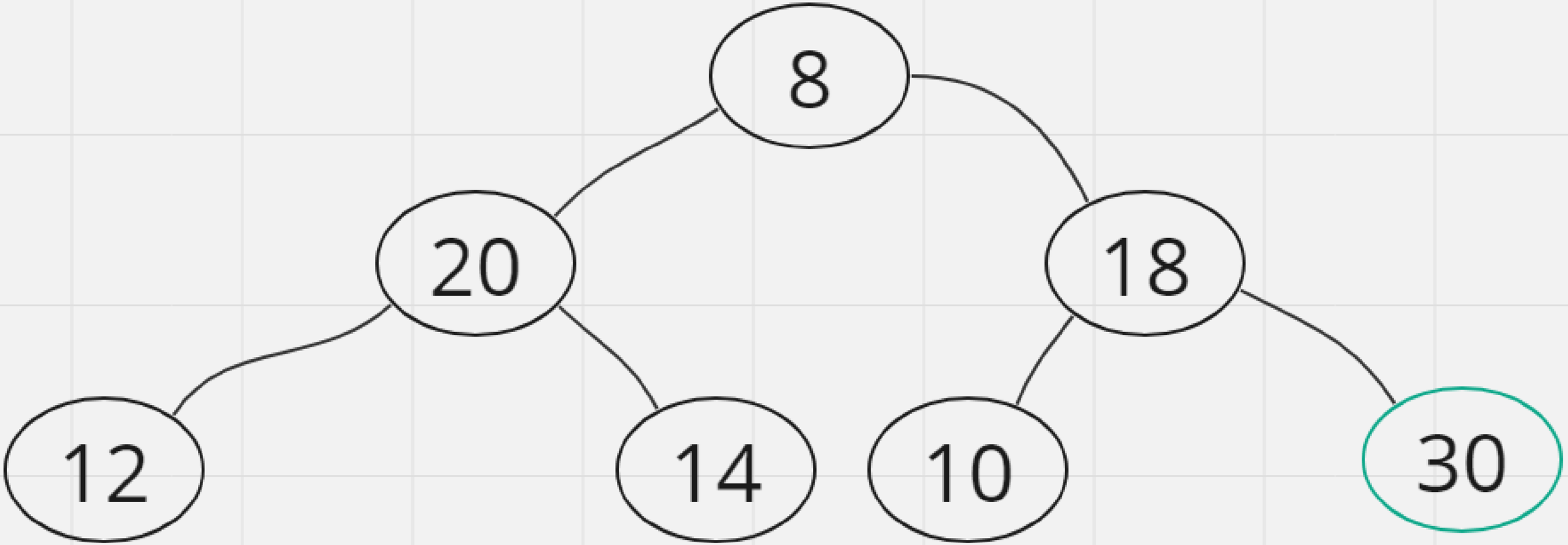
Start from the non-leaf node, and iterate backwards the root node.

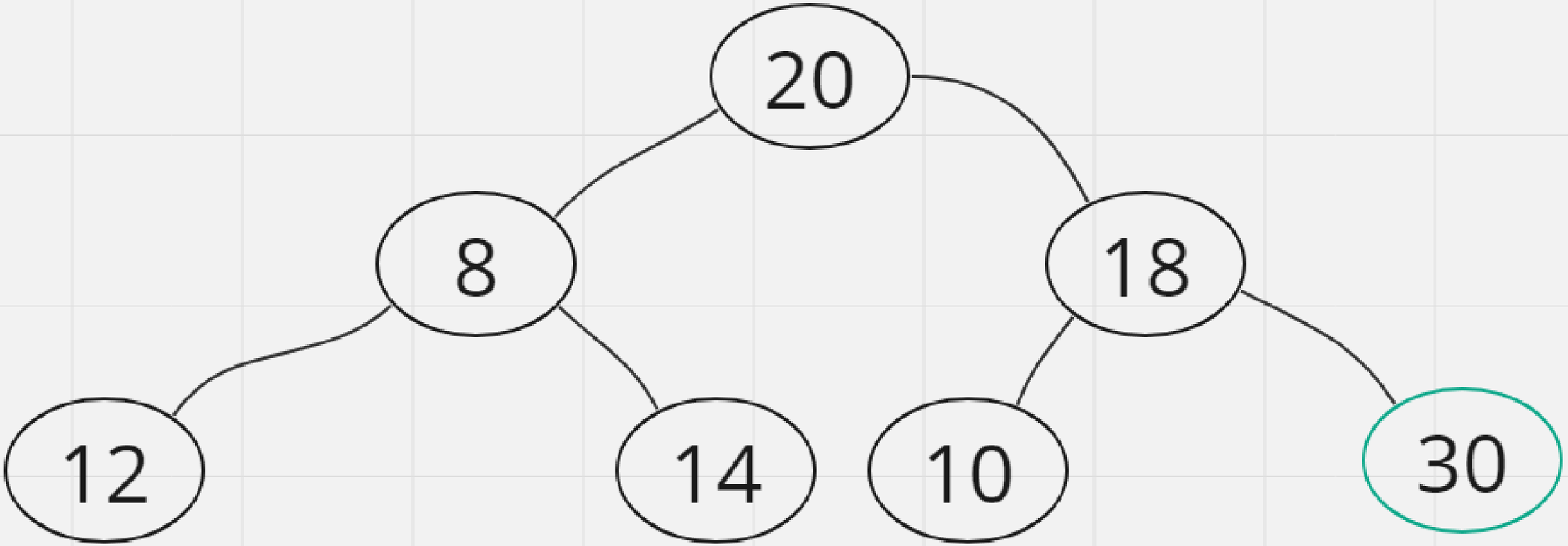
Compare with it's children, and swap if nessesary.

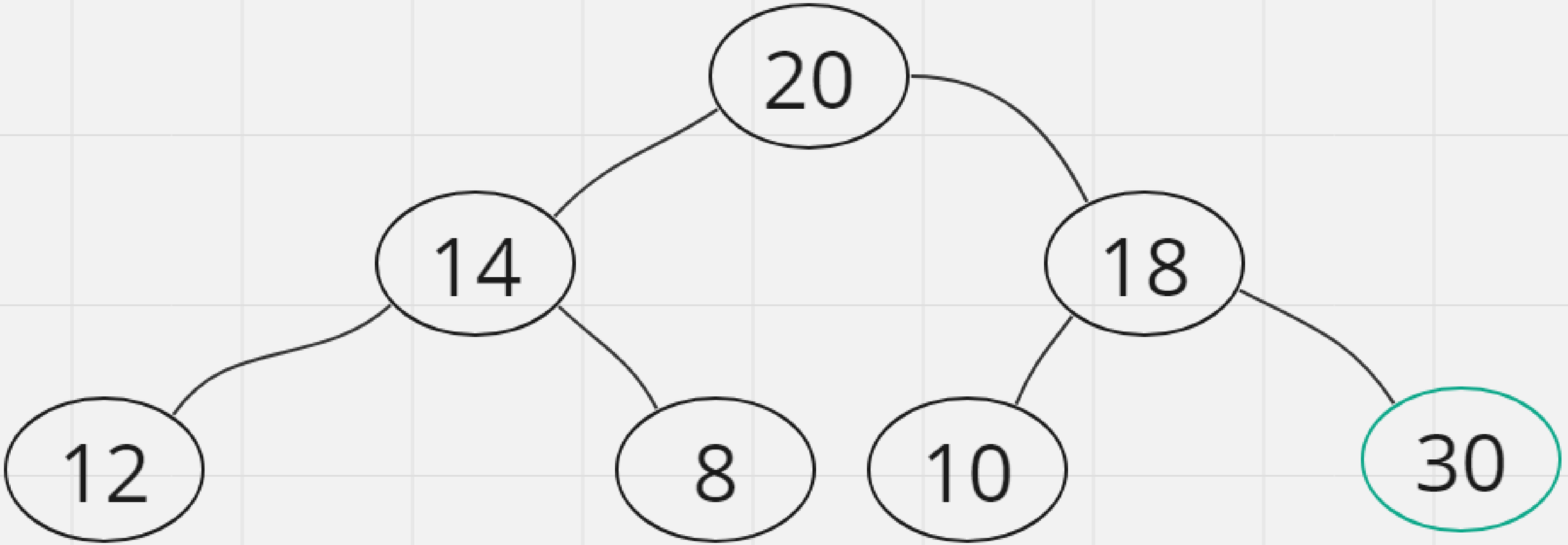
Continue the process untill satisfy all the heap properly.

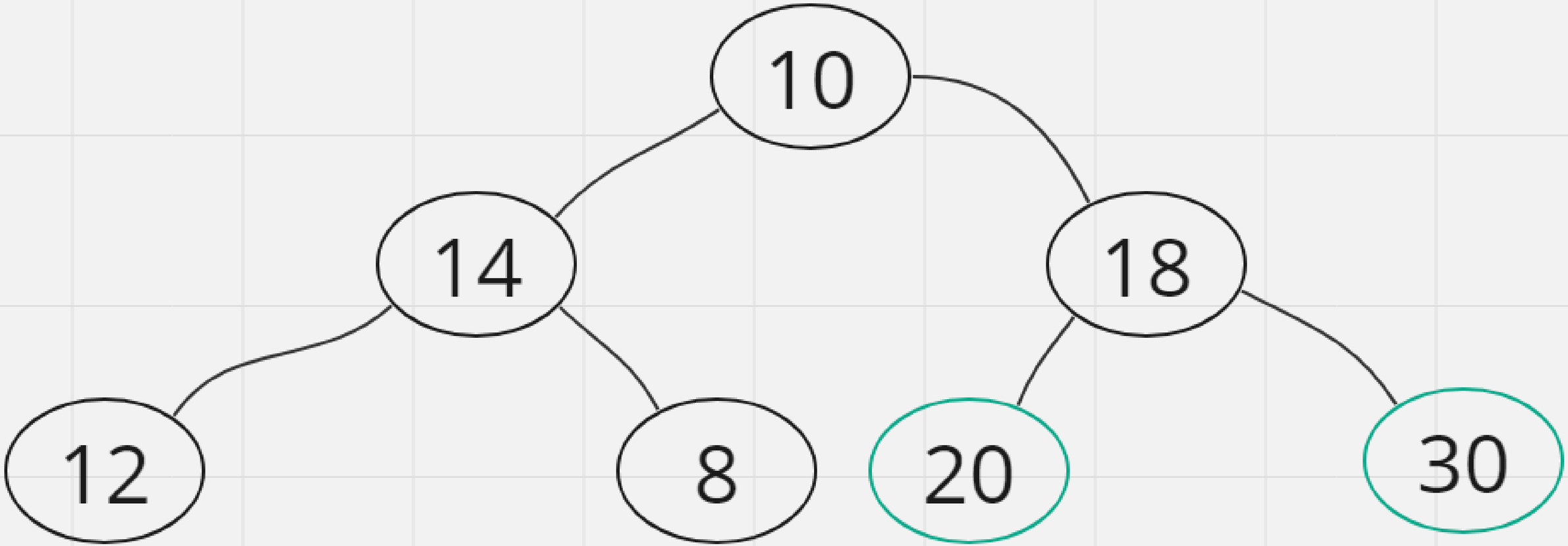
Let’s solve an example.

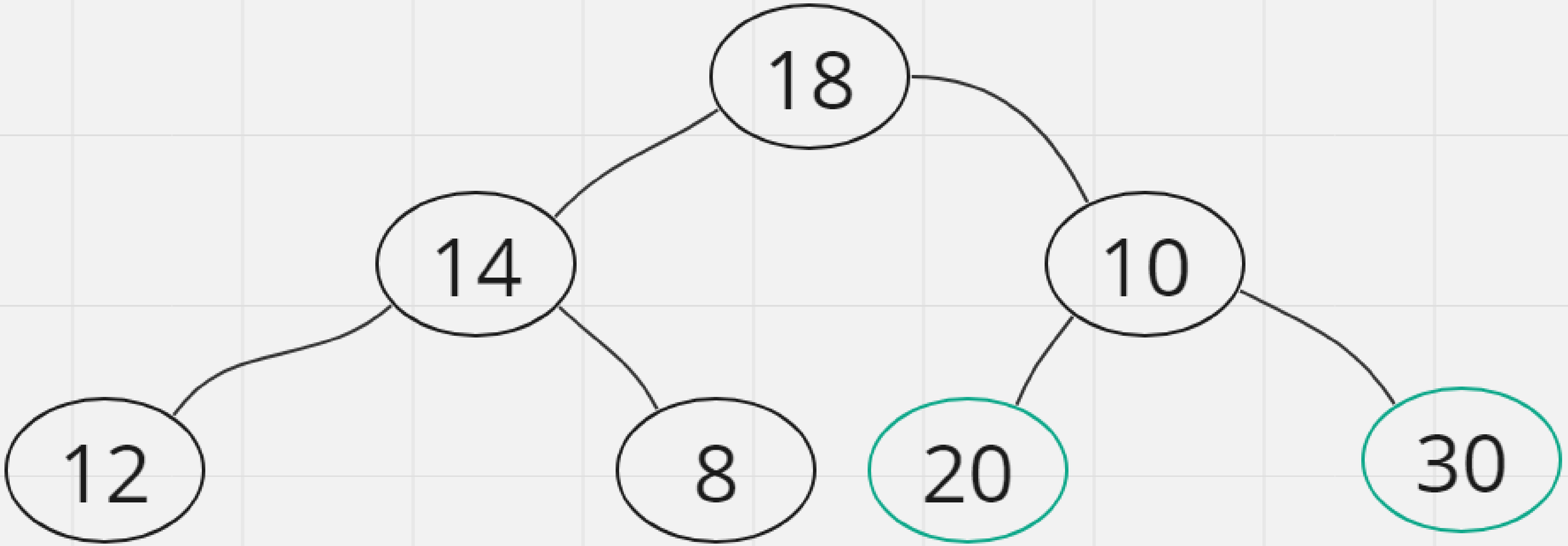

30
20
18
12
14
10
8

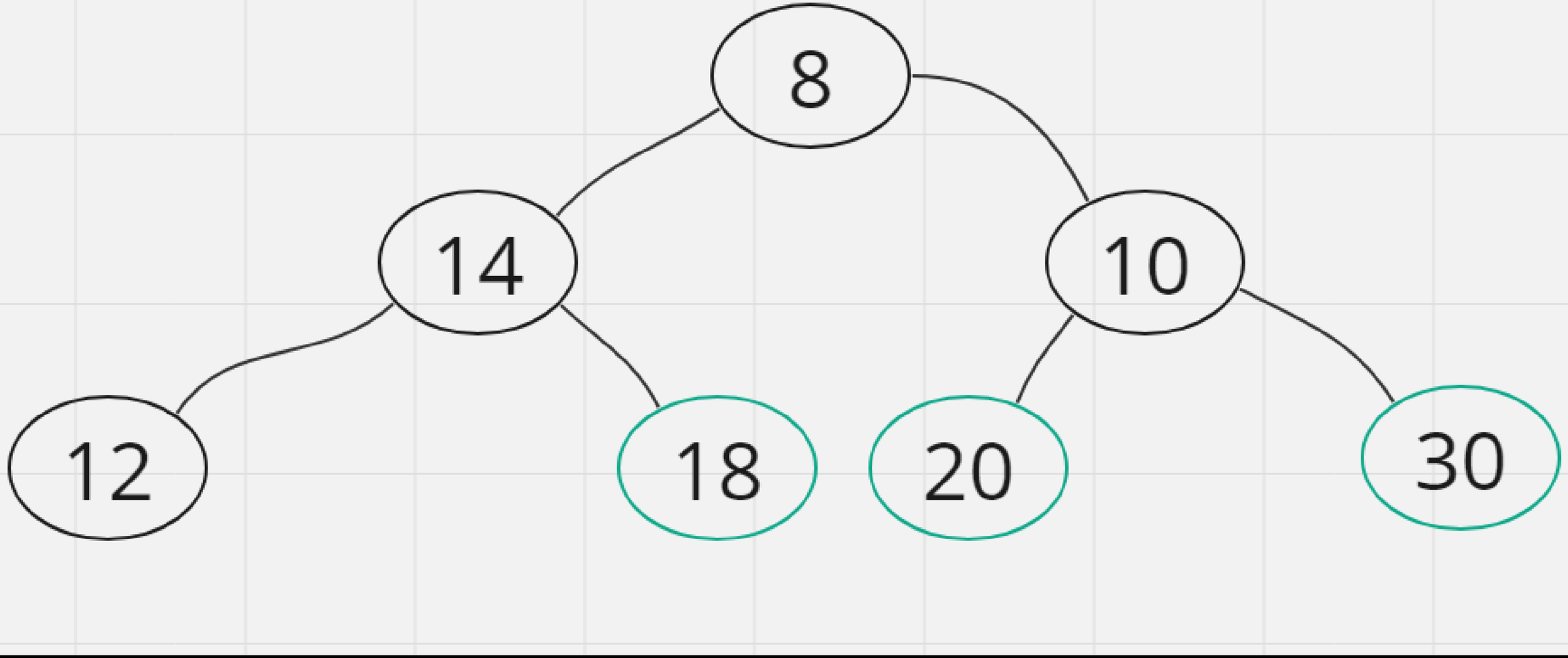

8
20
18
12
14
10
30

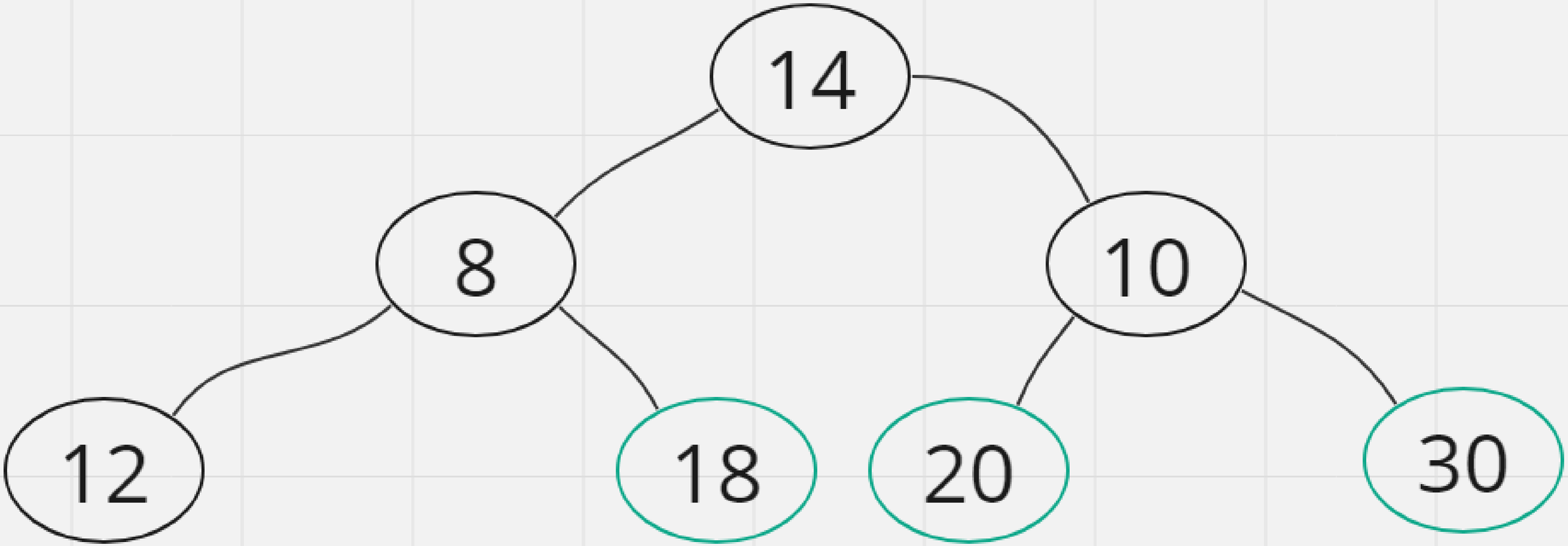

20
8
18
12
14
10
30

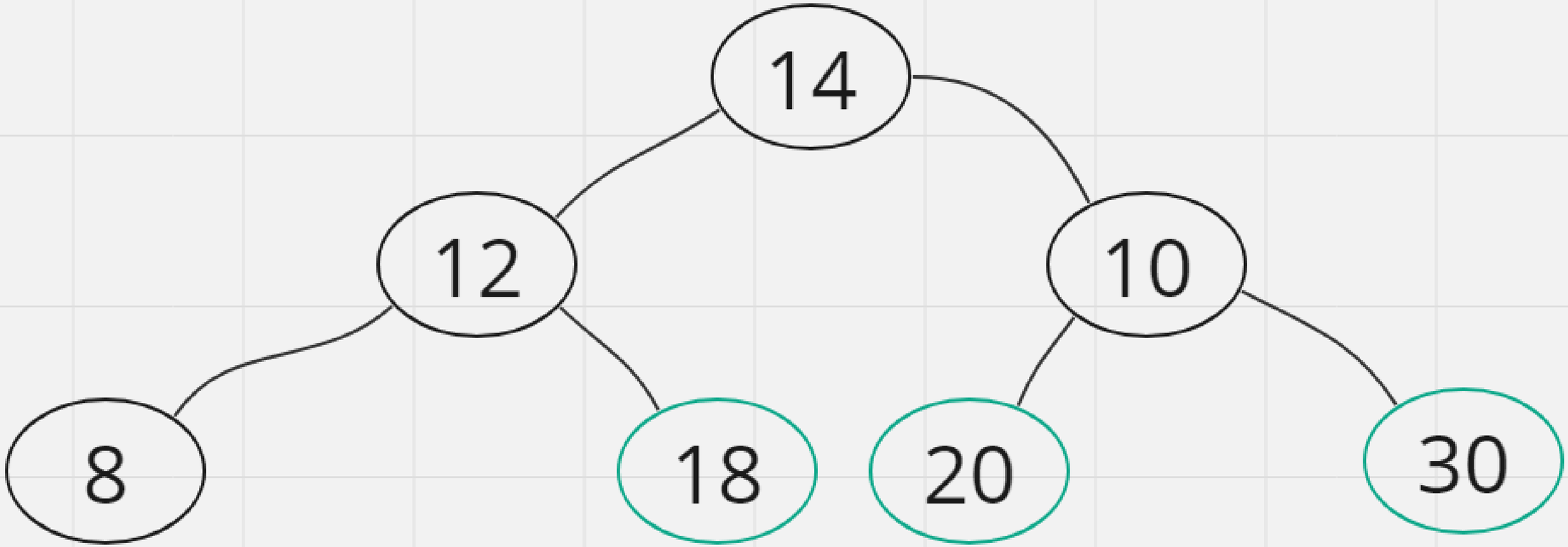

20
14
18
12
8
10
30

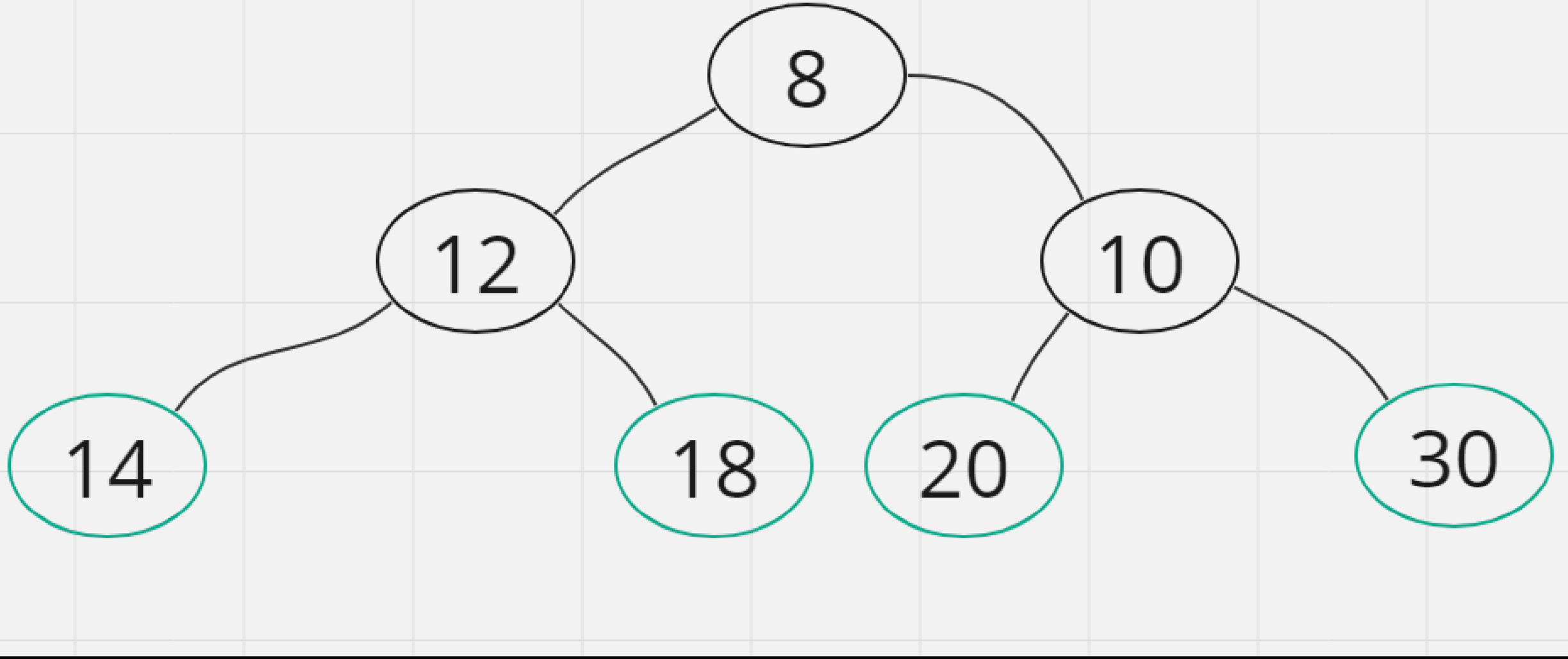

10
14
18
12
8
20
30

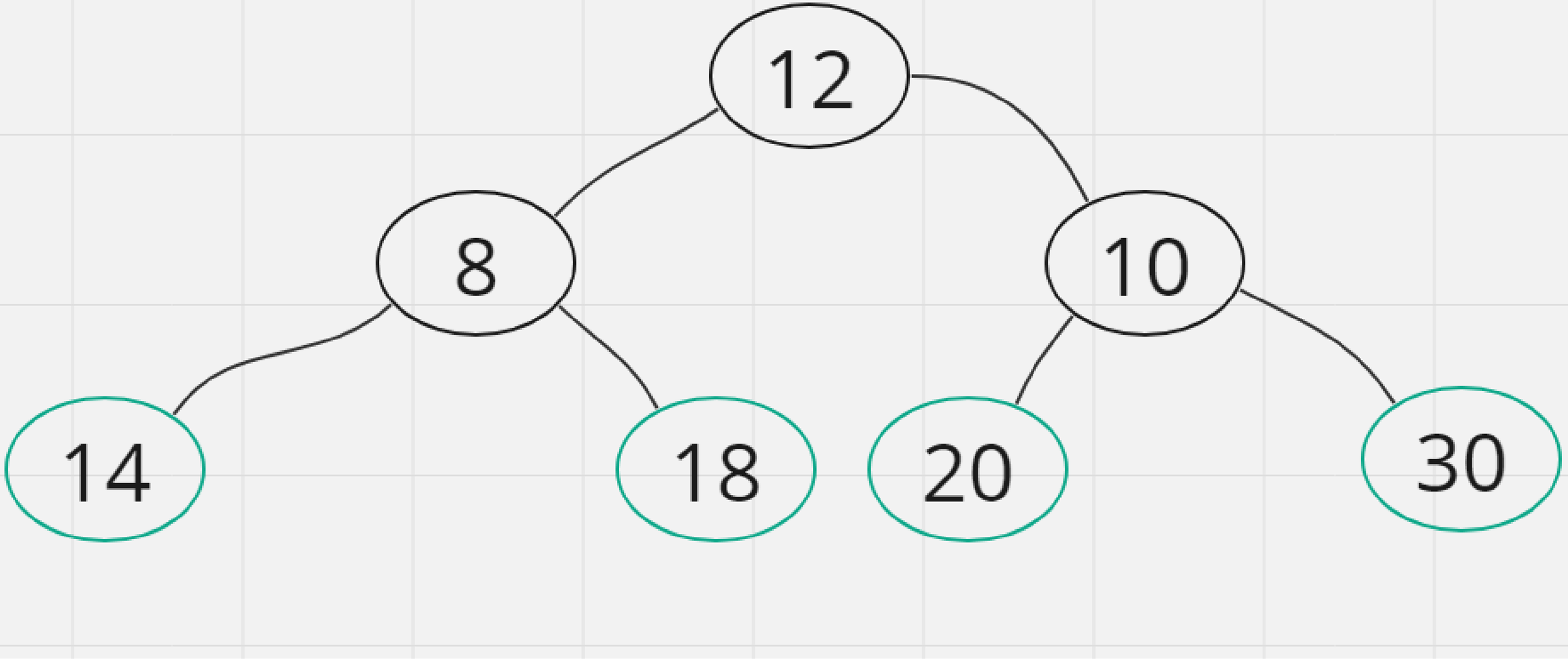

18
14
10
12
8
20
30

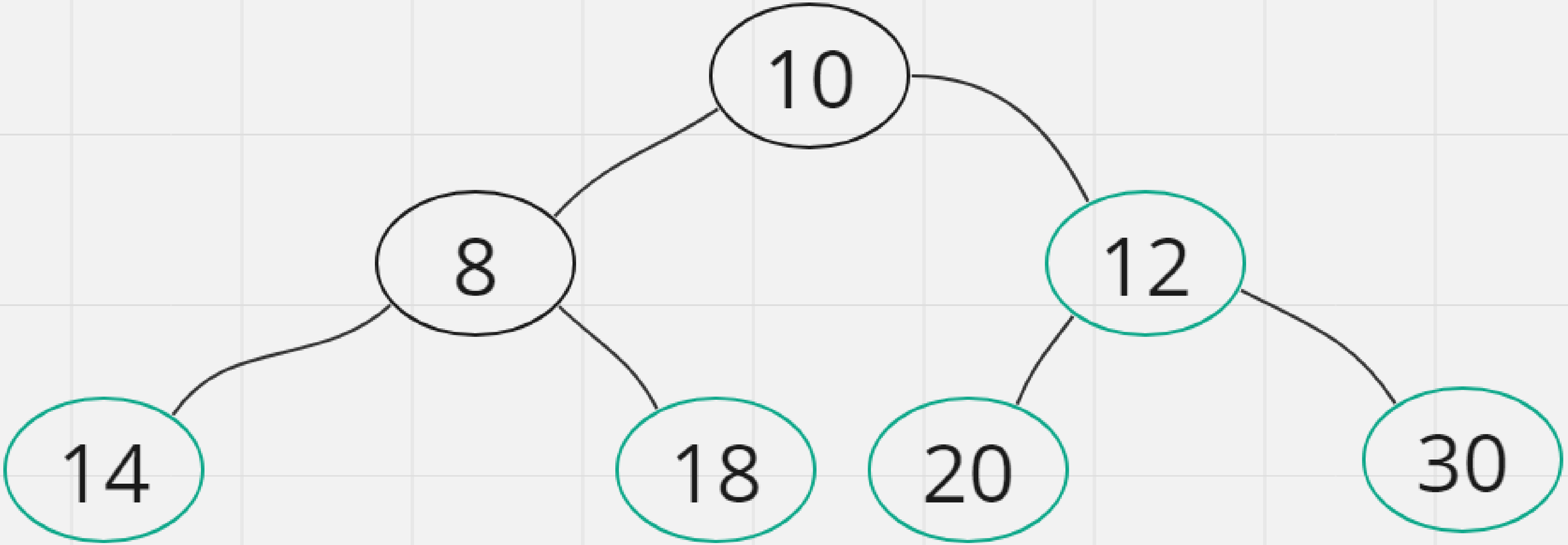

8
14
10
12
18
20
30

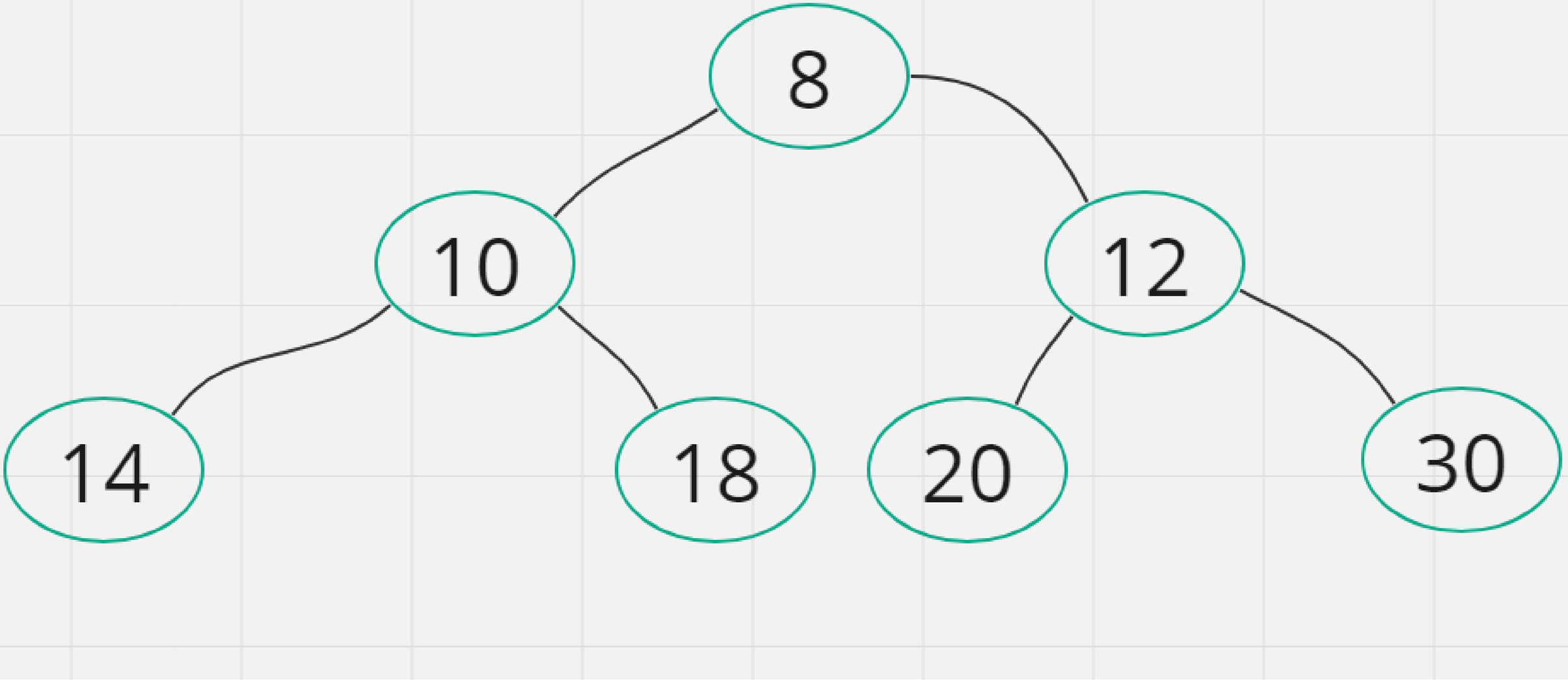

14
8
10
12
18
20
30


14
12
10
8
18
20
30

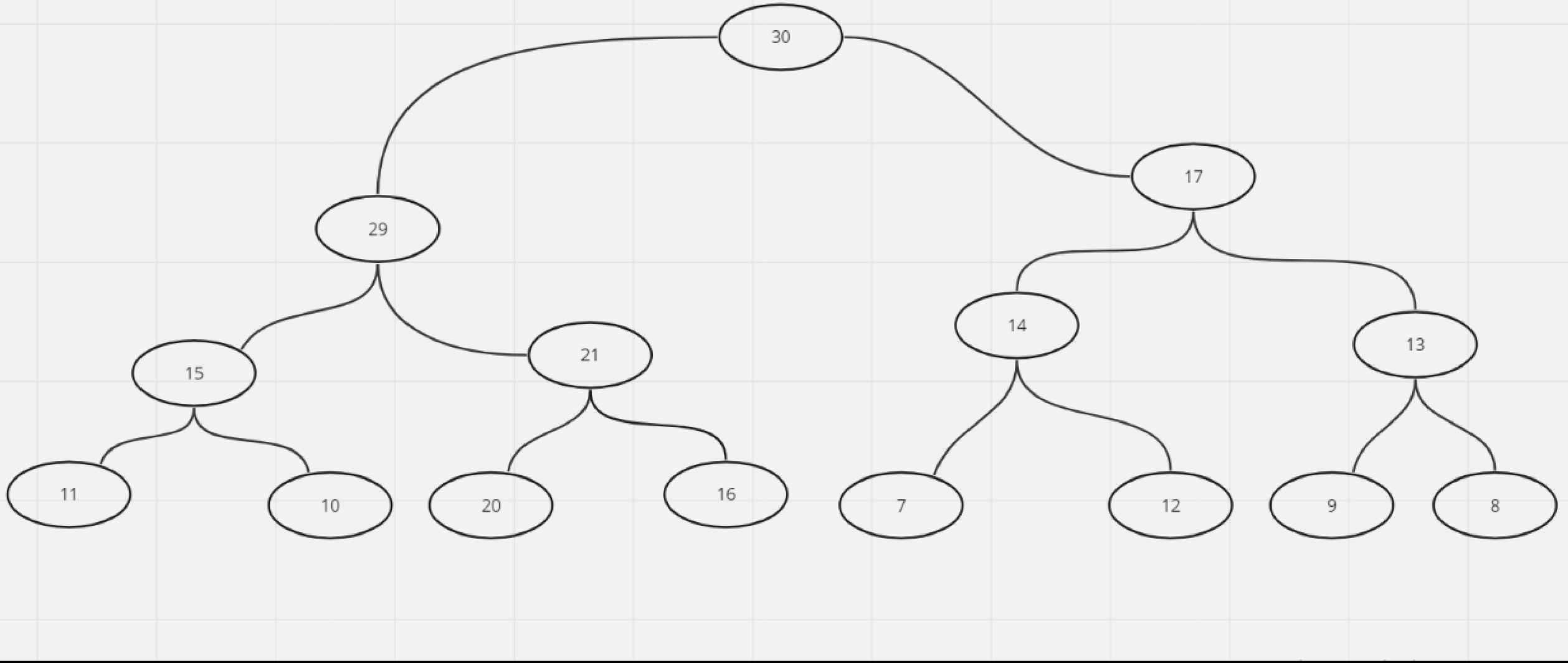

8
12
10
14
18
20
30

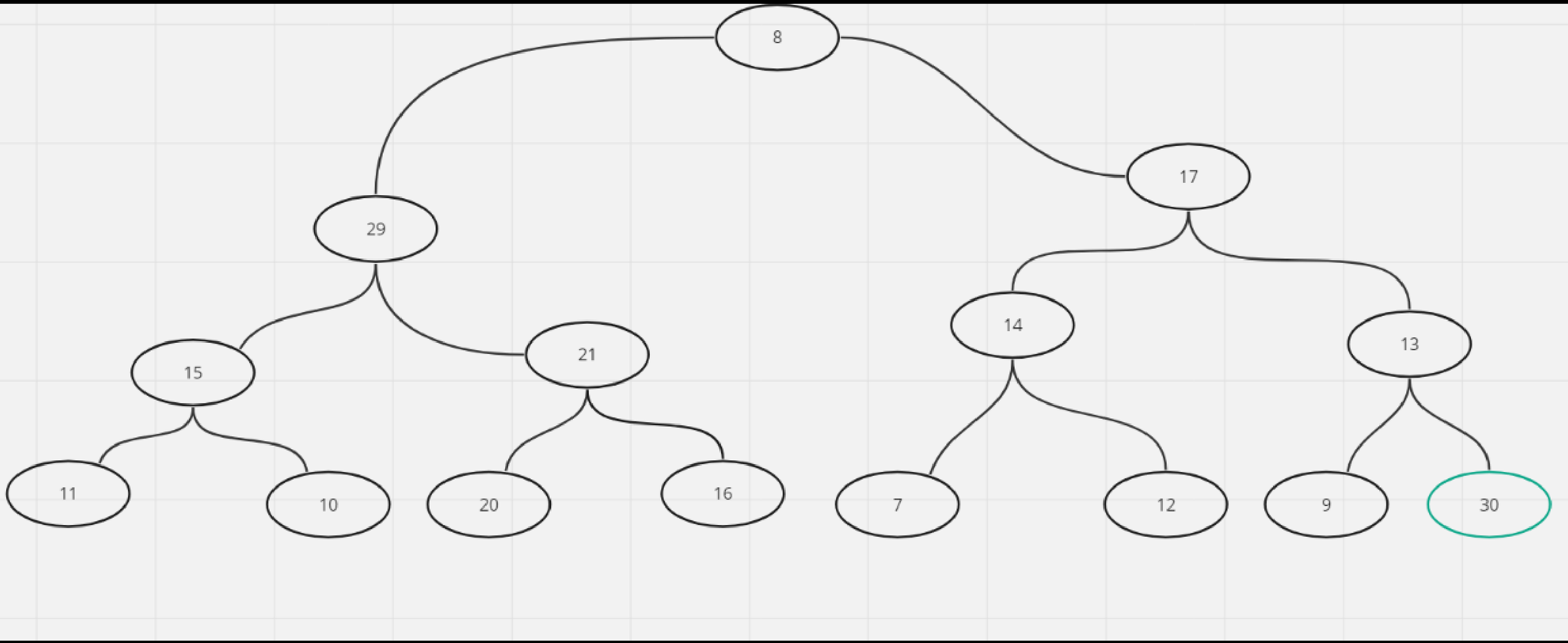

12
8
10
14
18
20
30

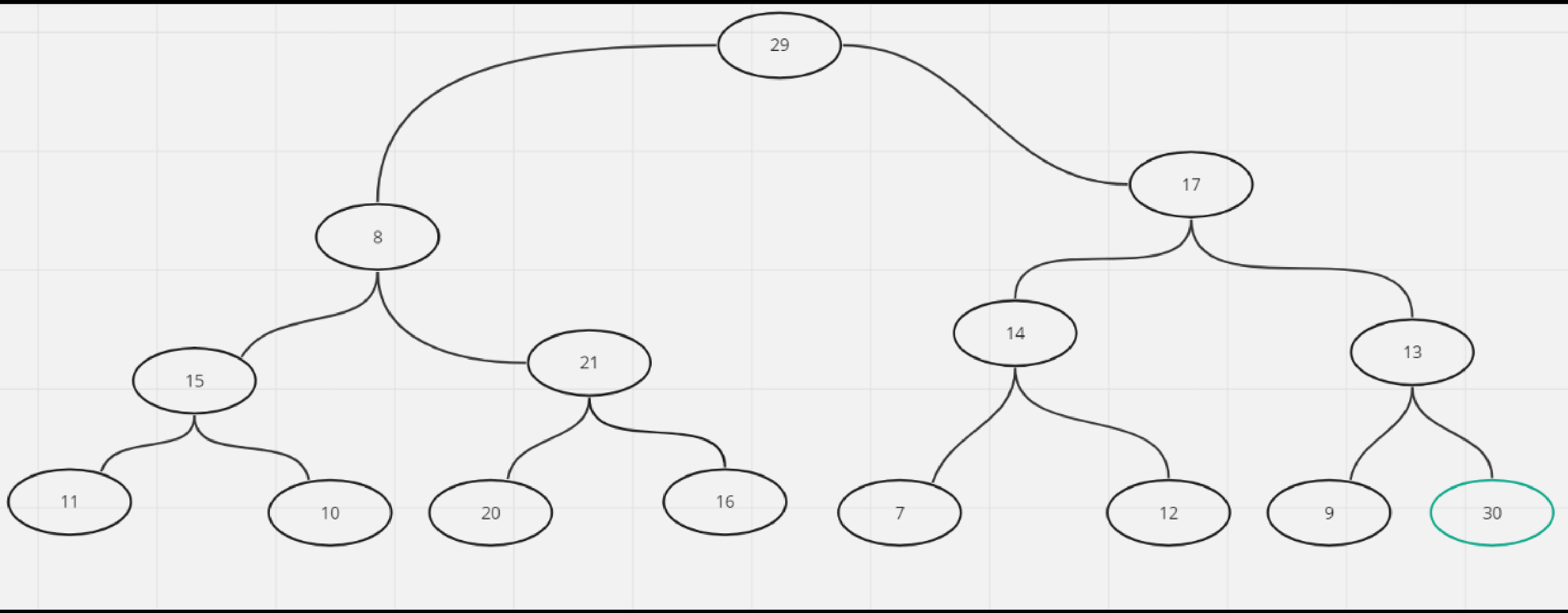

10
8
12
14
18
20
30

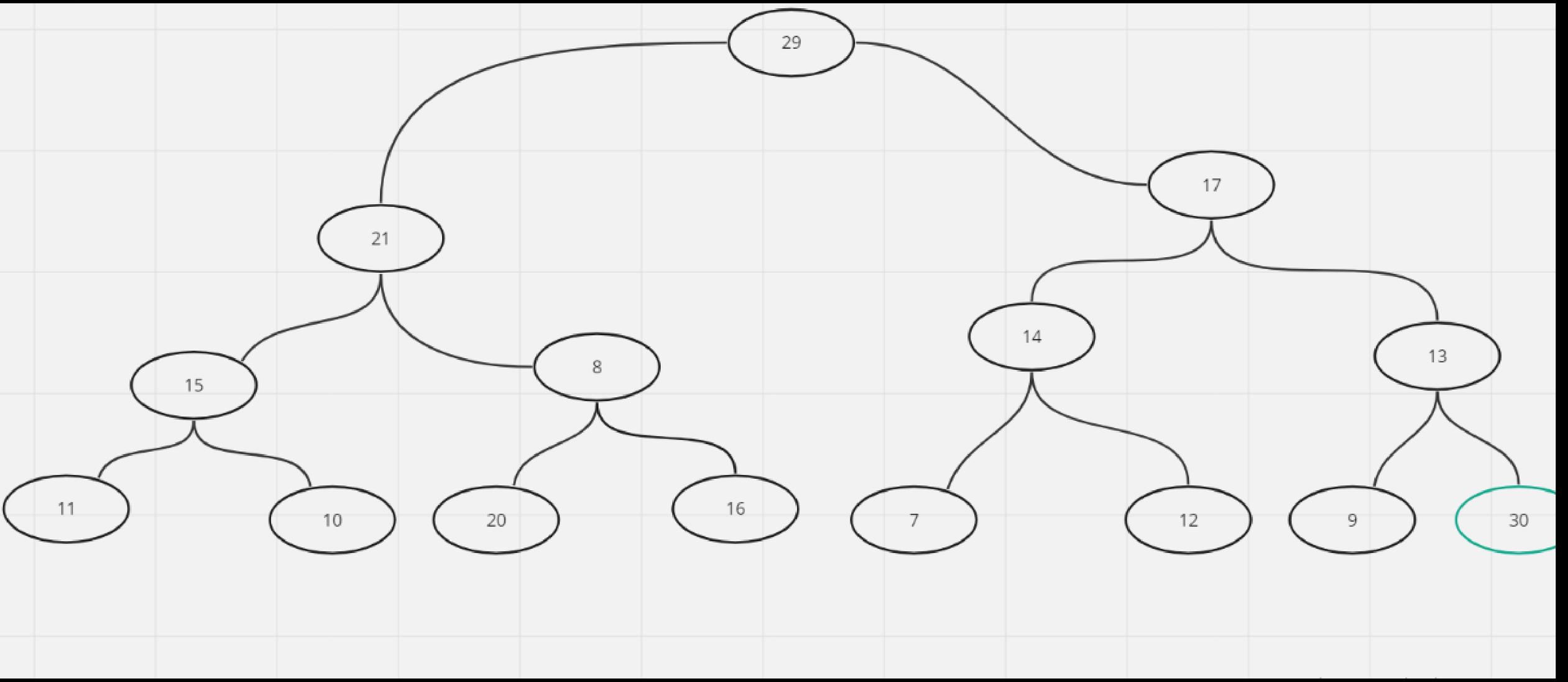

8
10
12
14
18
20
30

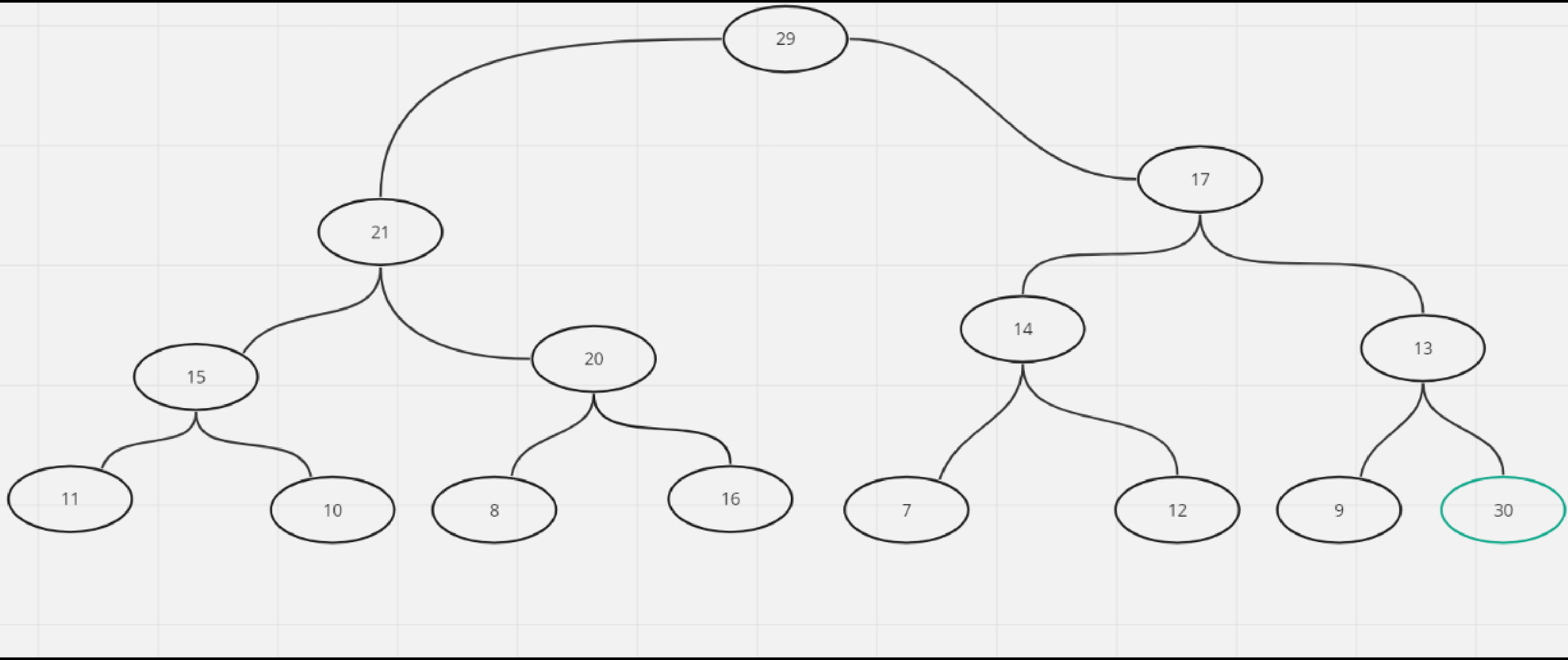
Now, the array is sorted.

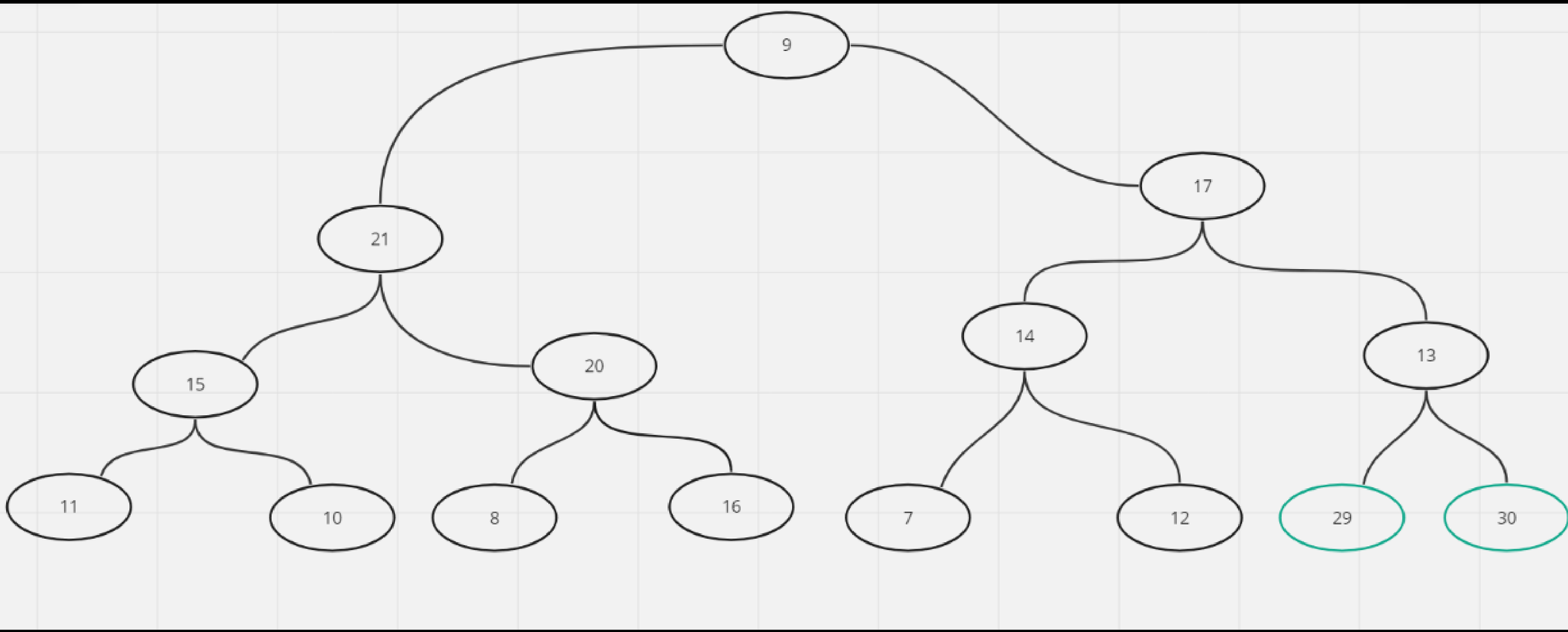

30
29
17
15
21
14
13
11
10
20
16
7
12
9
8

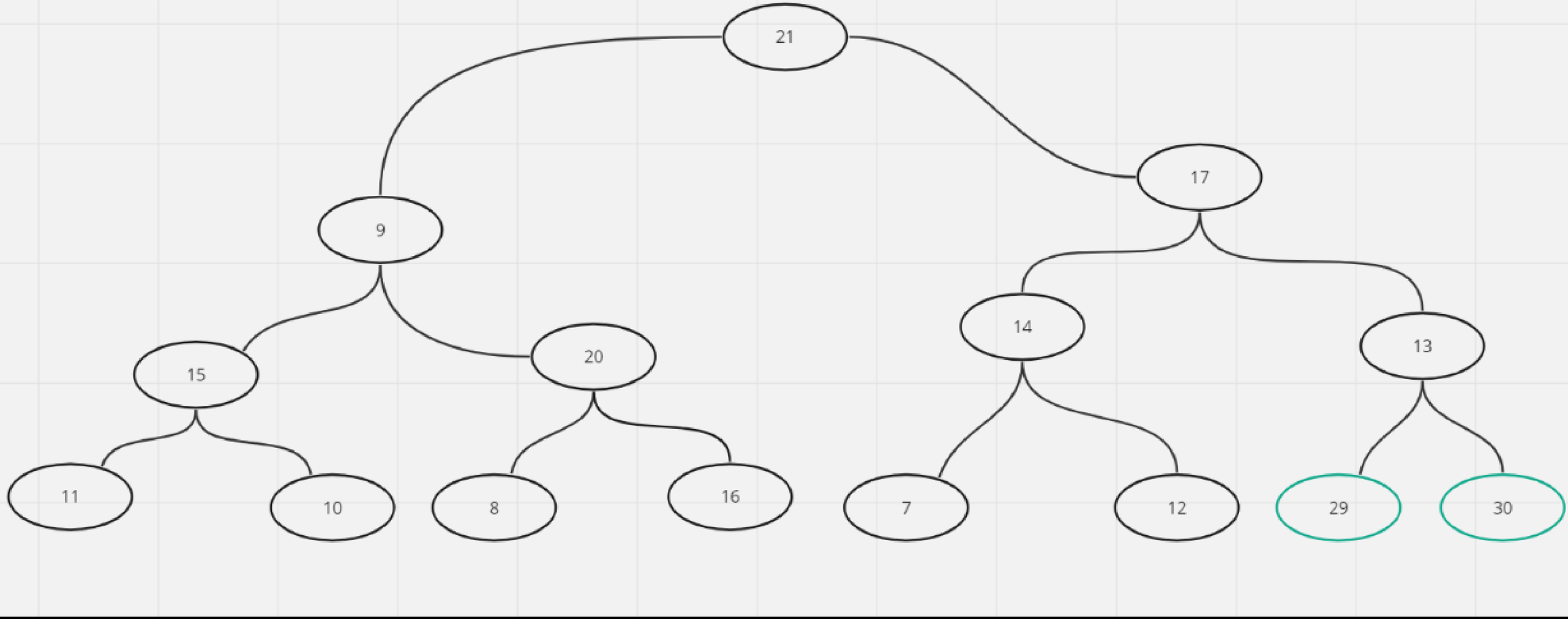

8
29
17
15
21
14
13
11
10
20
16
7
12
9
30

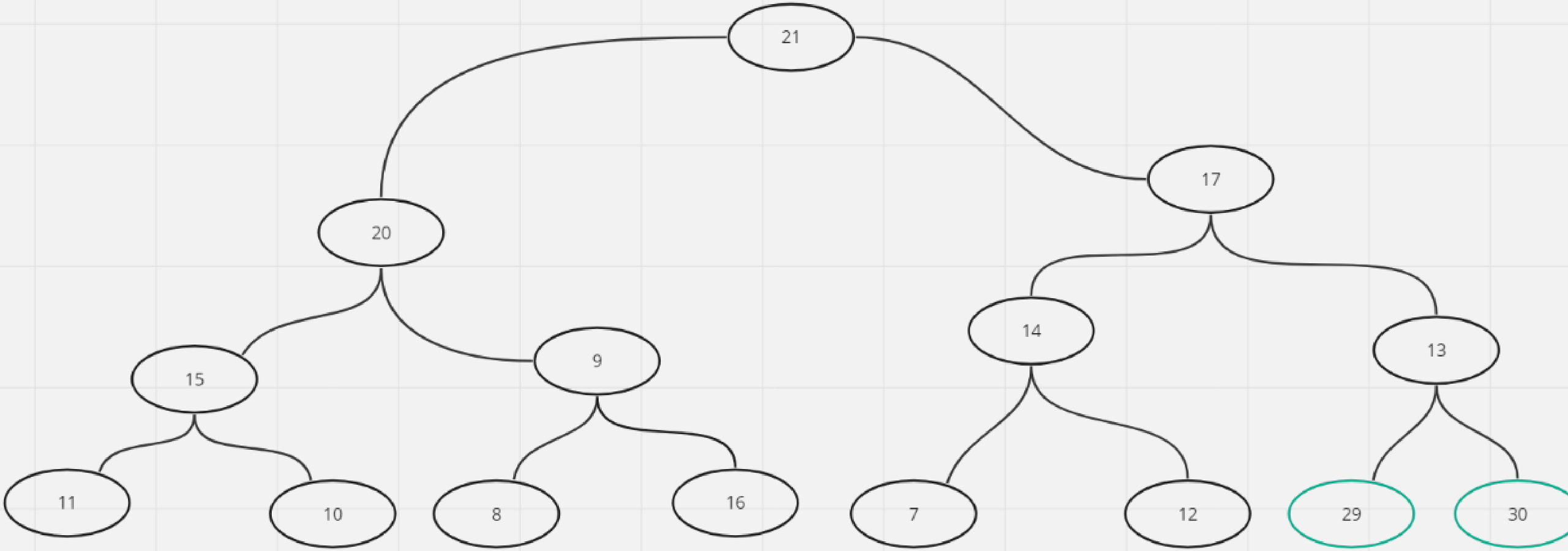

29
8
17
15
21
14
13
11
10
20
16
7
12
9
30

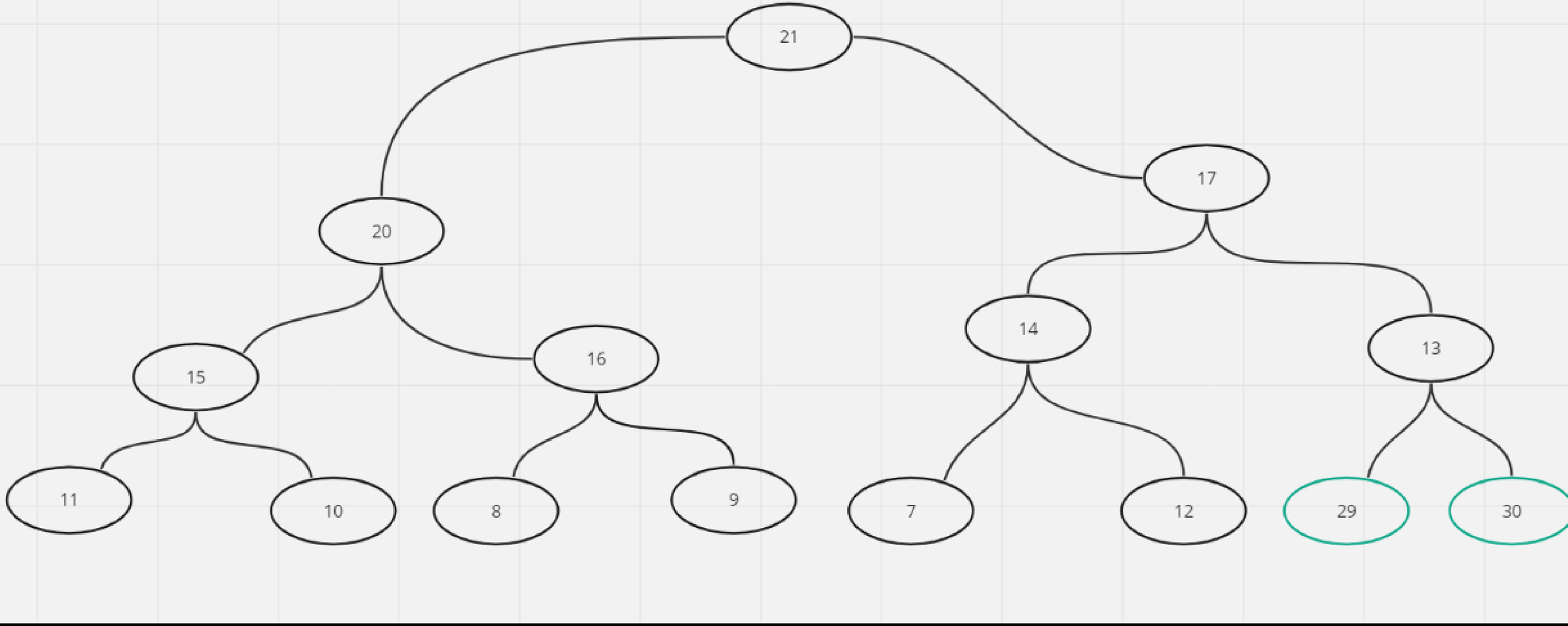

29
21
17
15
8
14
13
11
10
20
16
7
12
9
30

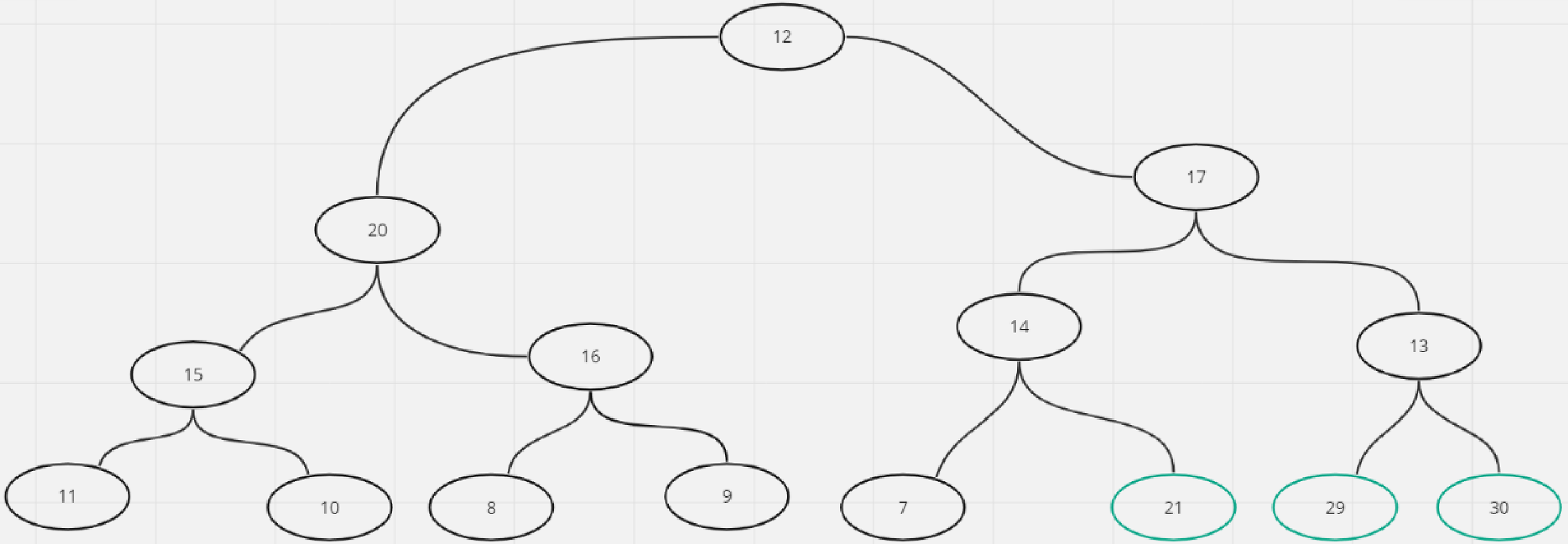

29
21
17
15
20
14
13
11
10
8
16
7
12
9
30

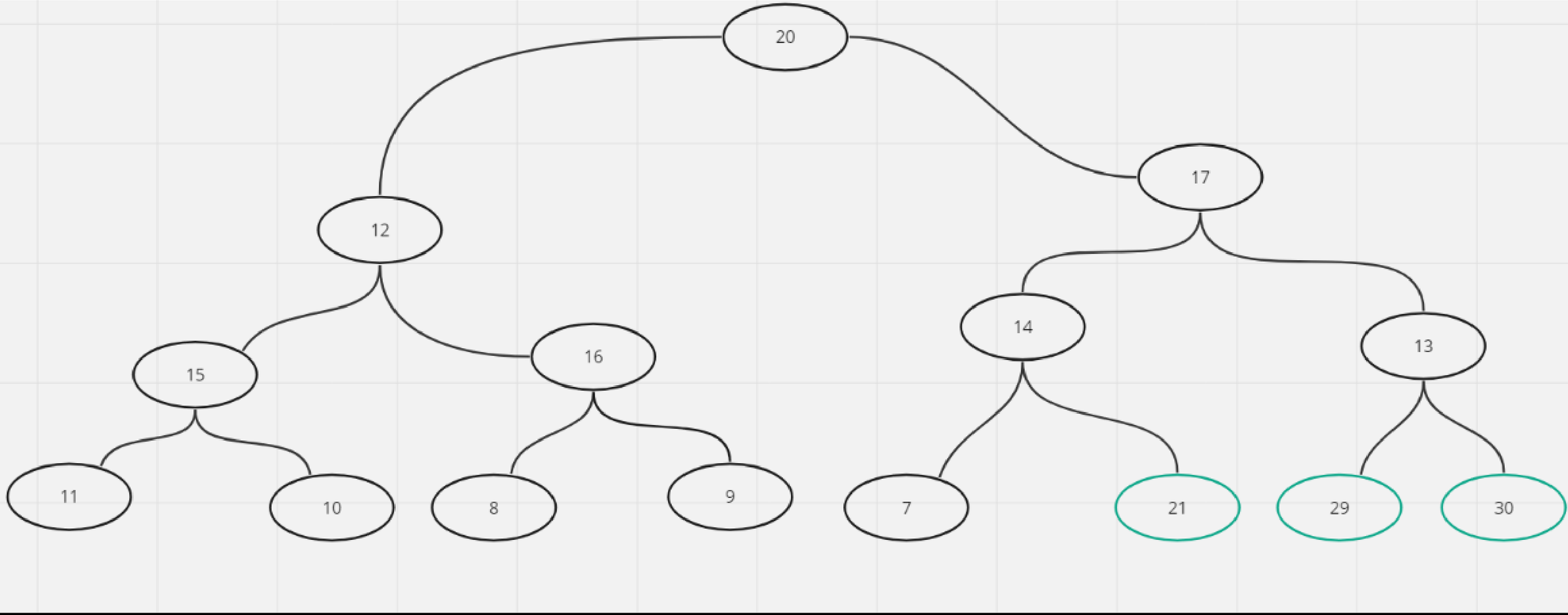

9
21
17
15
20
14
13
11
10
8
16
7
12
29
30

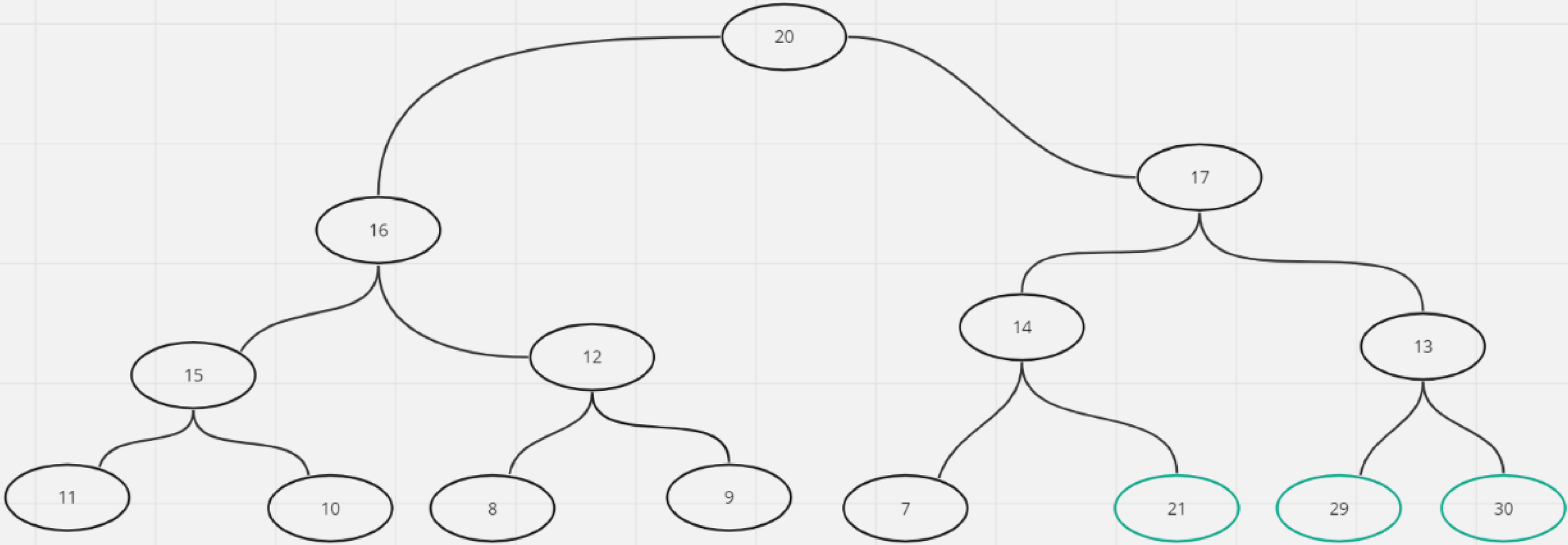

21
9
17
15
20
14
13
11
10
8
16
7
12
29
30

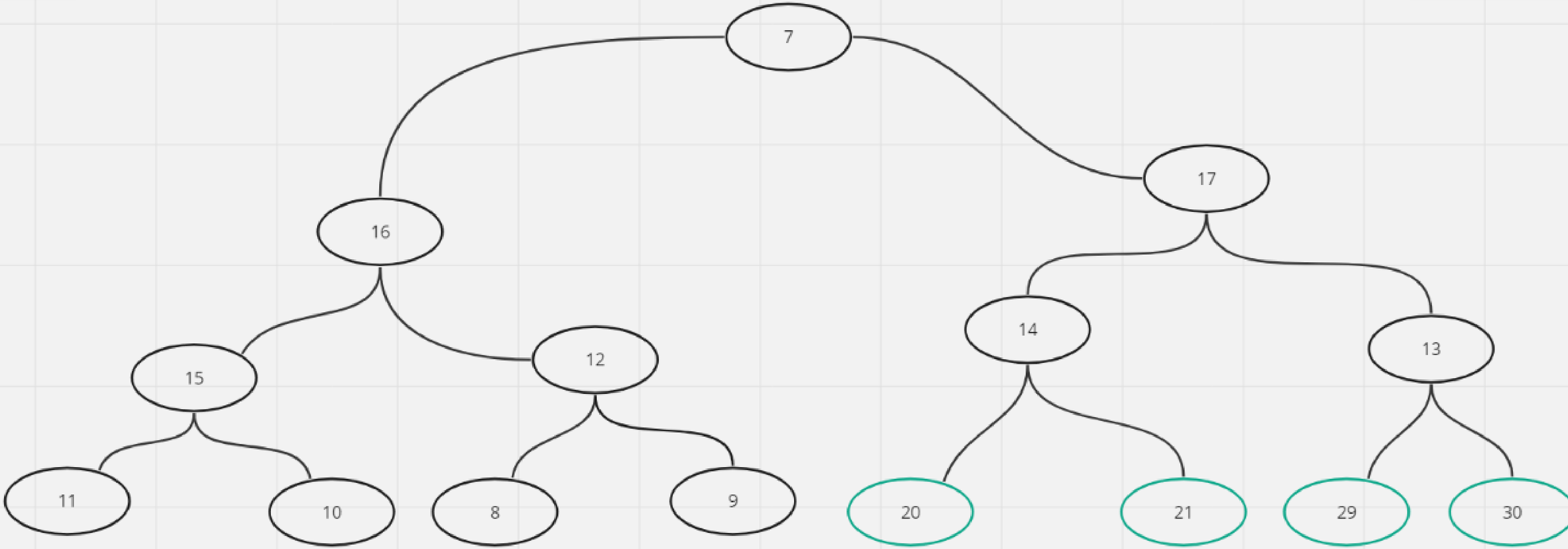


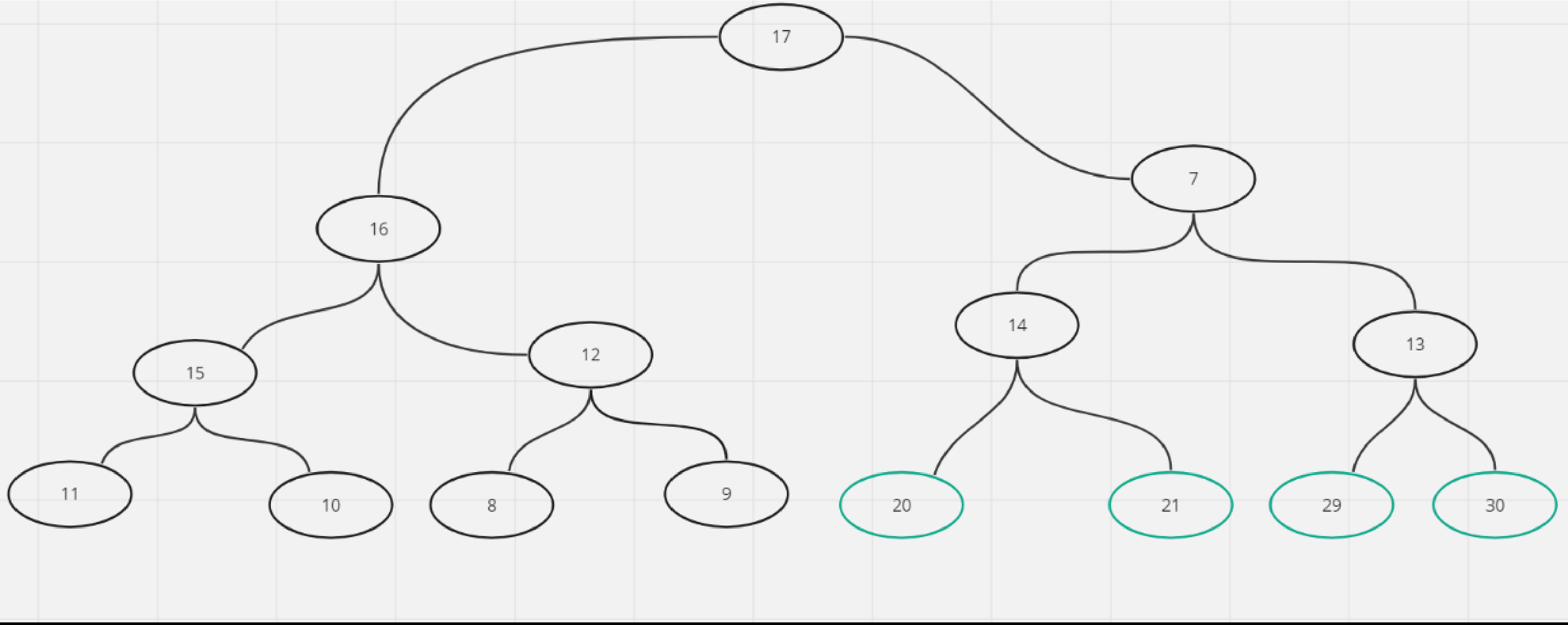

21
20
17
15
16
14
13
11
10
8
9
7
12
29
30

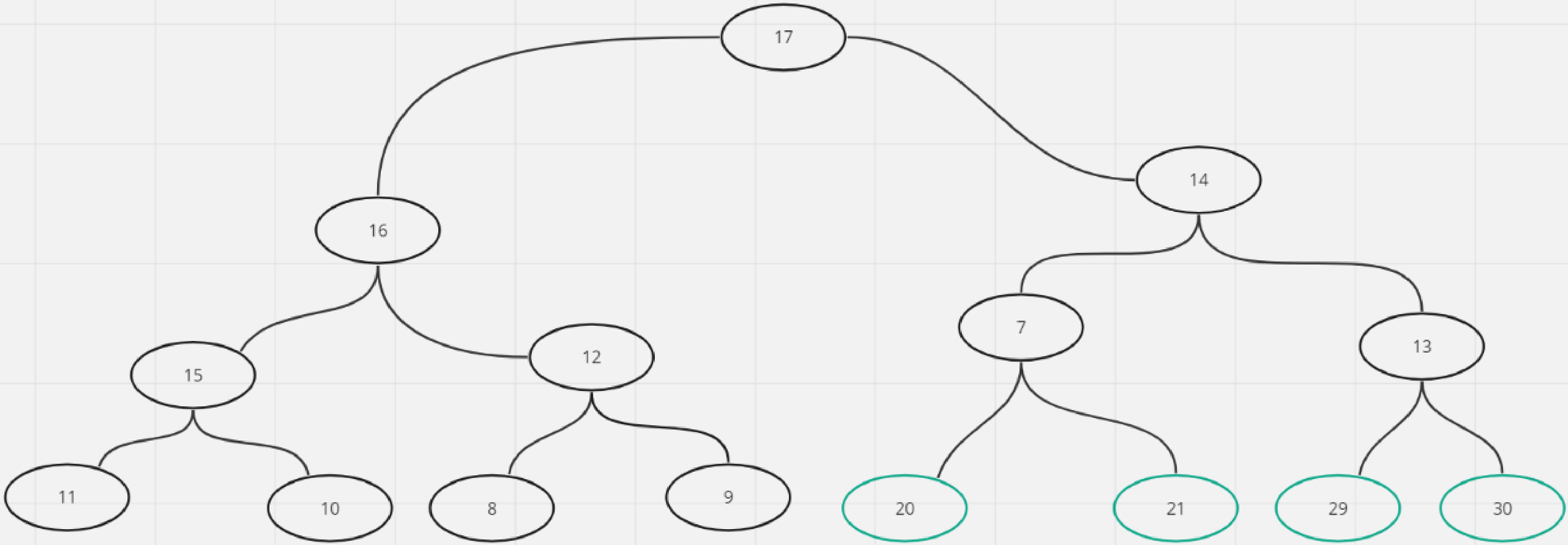


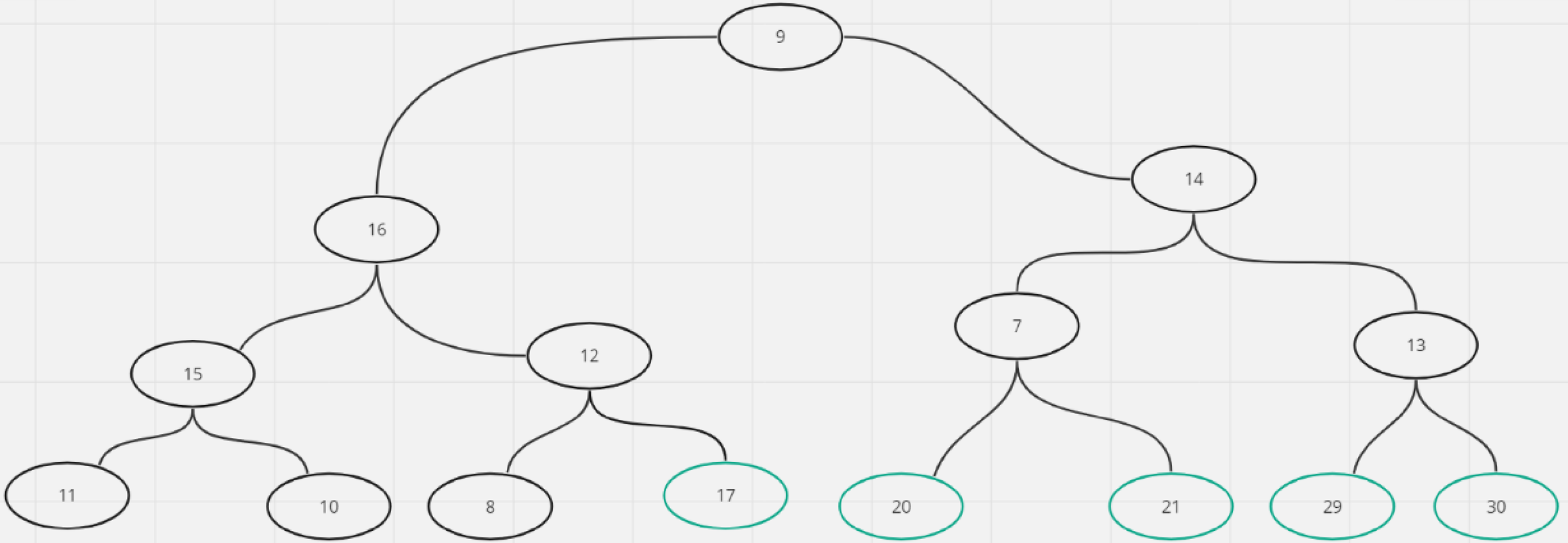

20
12
17
15
16
14
13
11
10
8
9
7
21
29
30

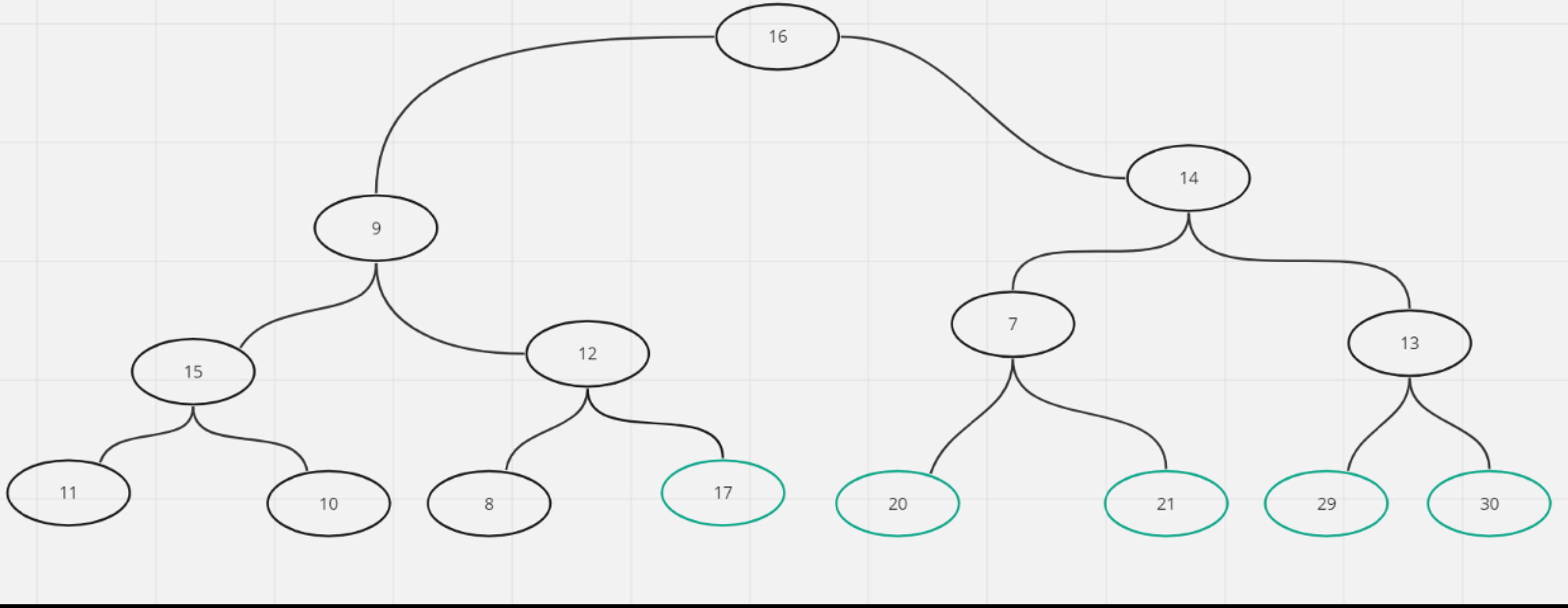

20
16
17
15
12
14
13
11
10
8
9
7
21
29
30

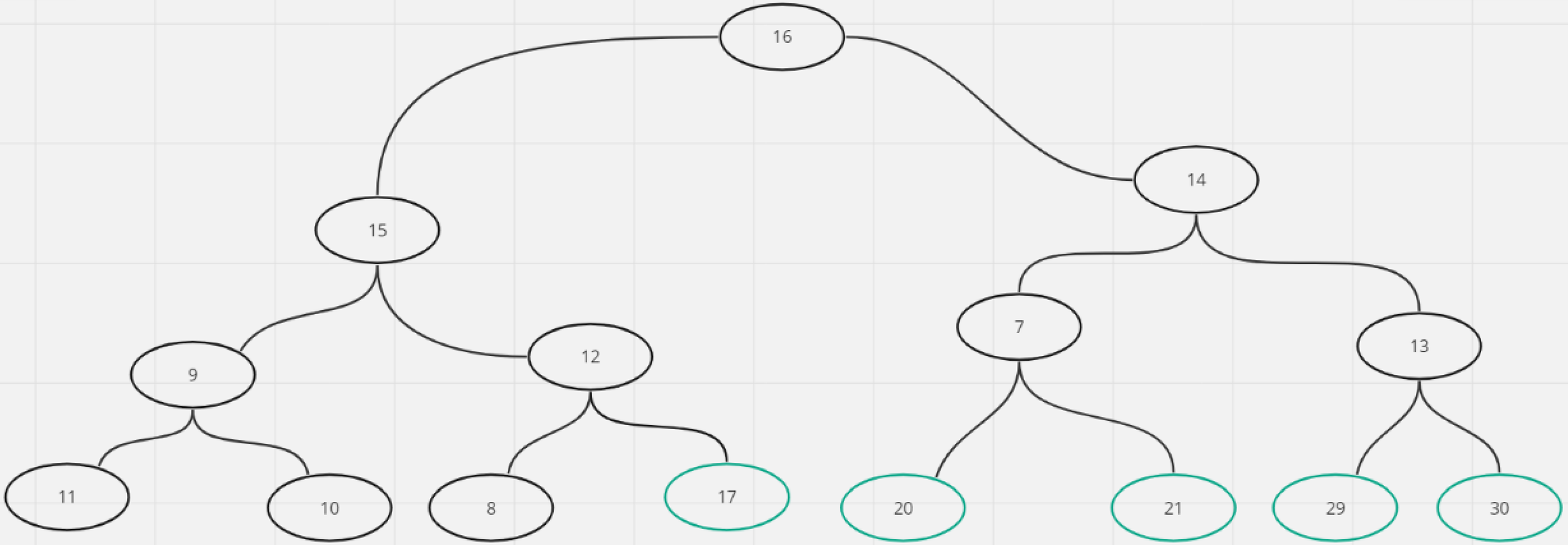

7
16
17
15
12
14
13
11
10
8
9
20
21
29
30

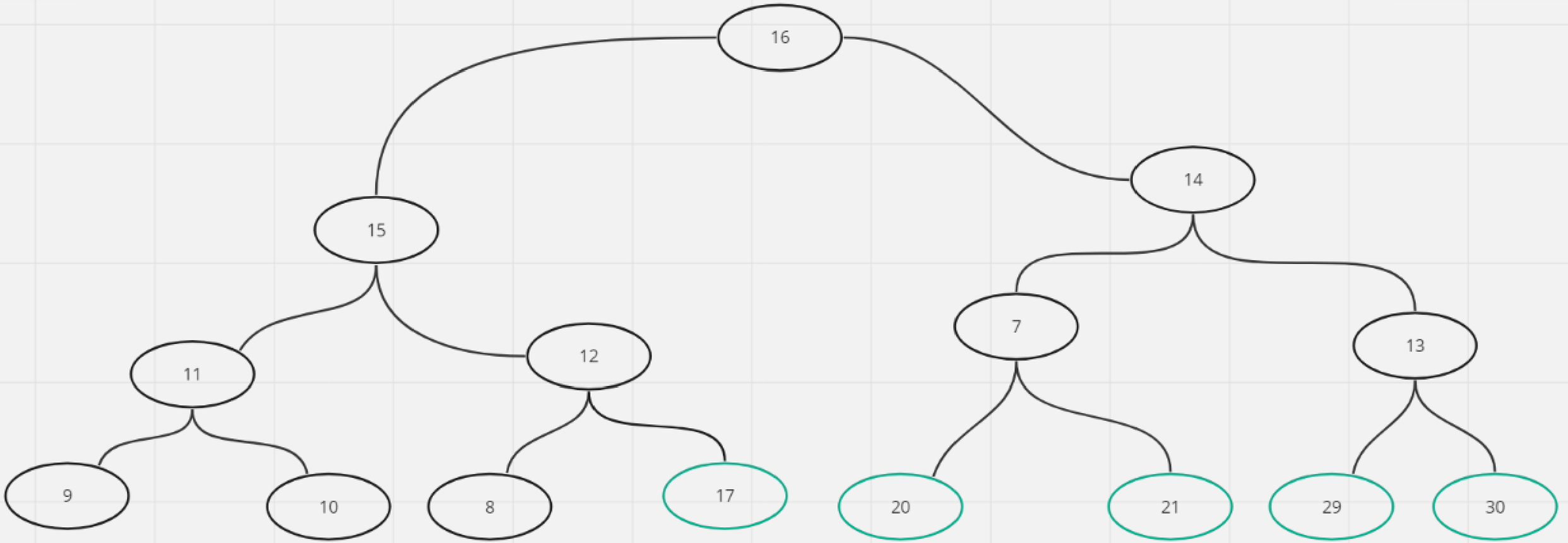

17
16
7
15
12
14
13
11
10
8
9
20
21
29
30

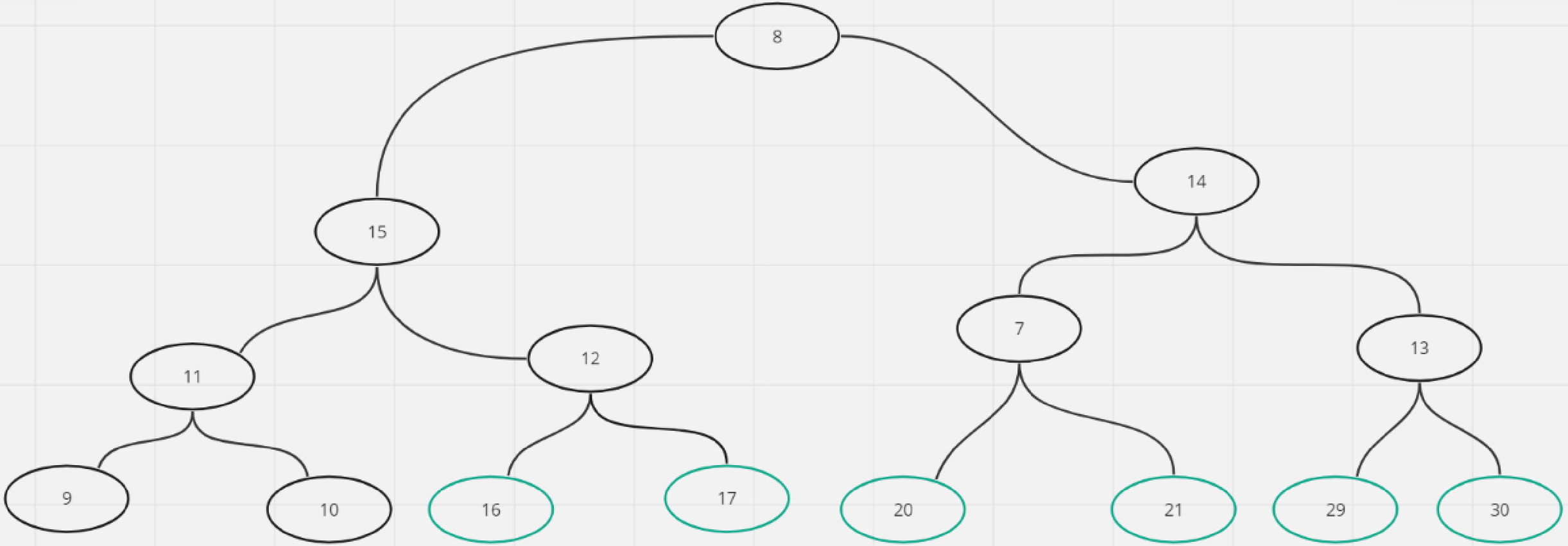

17
16
14
15
12
7
13
11
10
8
9
20
21
29
30

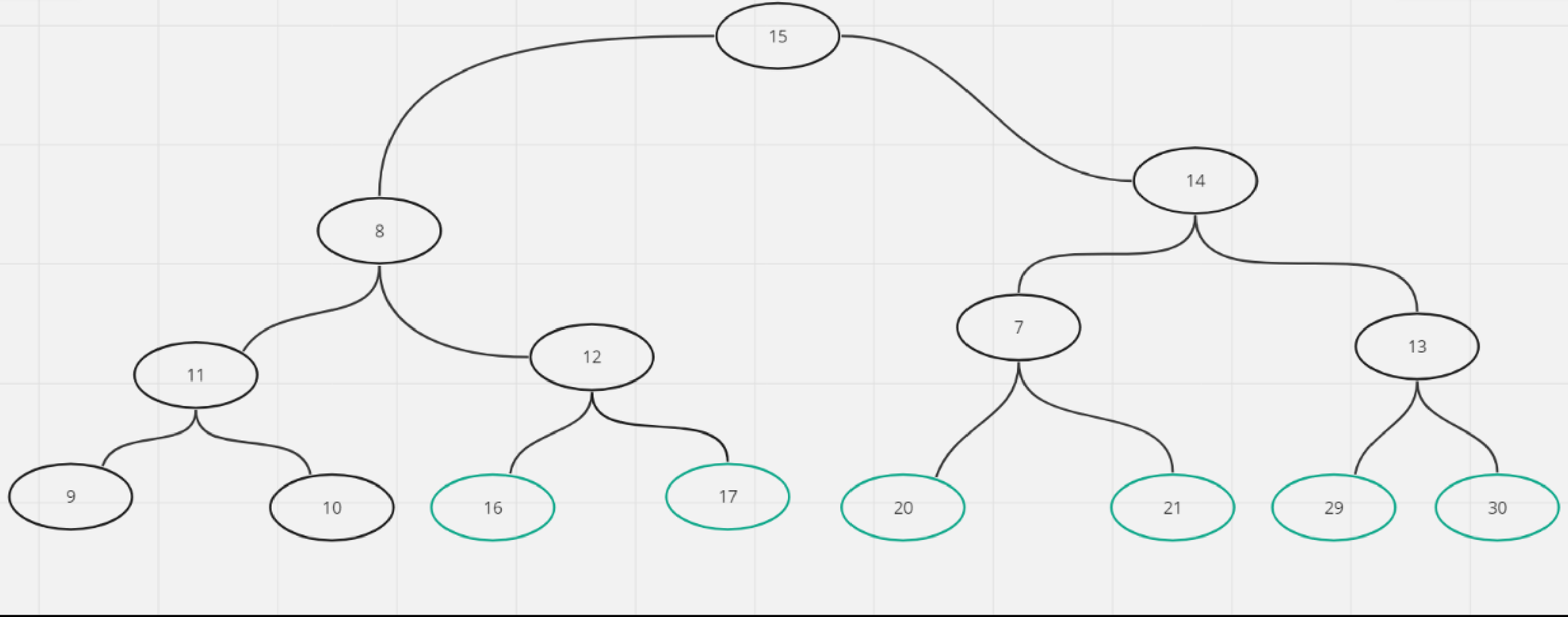

9
16
14
15
12
7
13
11
10
8
17
20
21
29
30

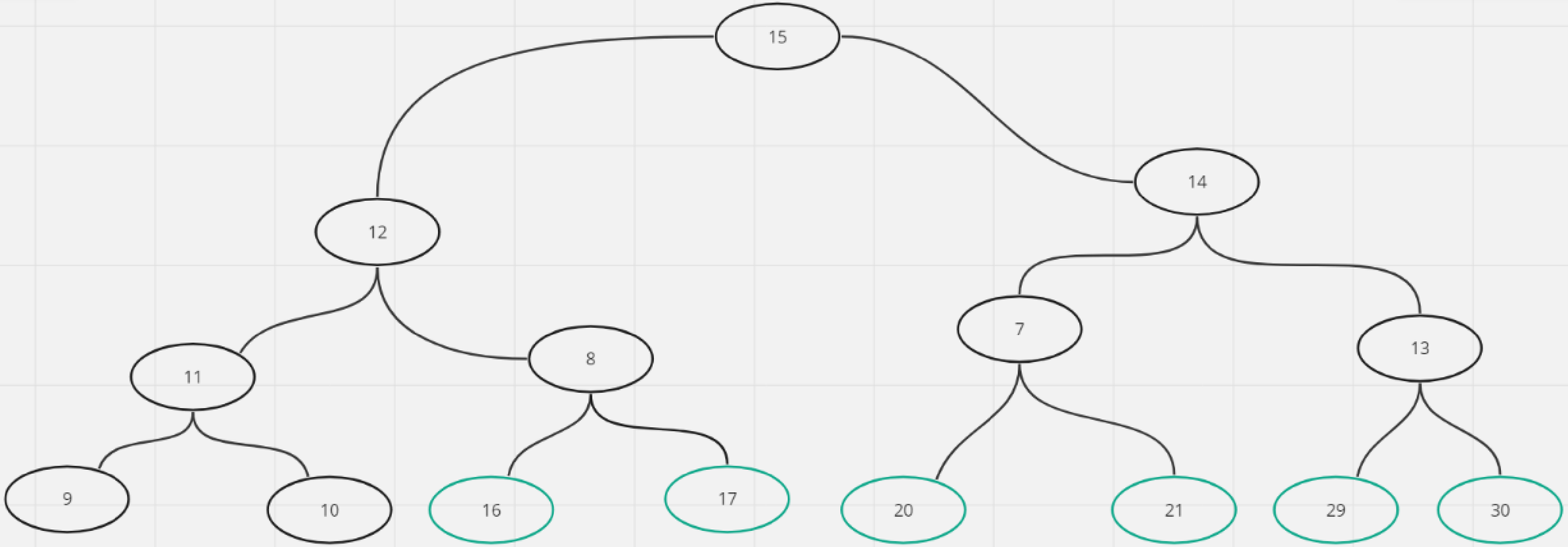

16
9
14
15
12
7
13
11
10
8
17
20
21
29
30

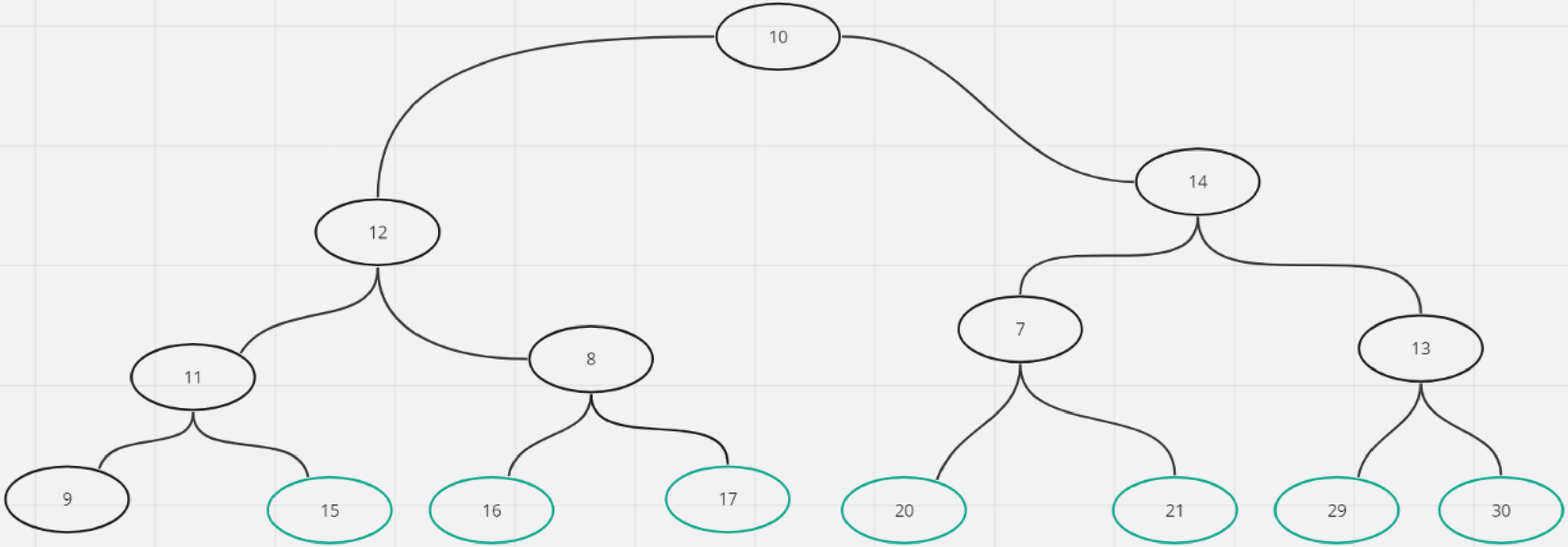

16
15
14
9
12
7
13
11
10
8
17
20
21
29
30

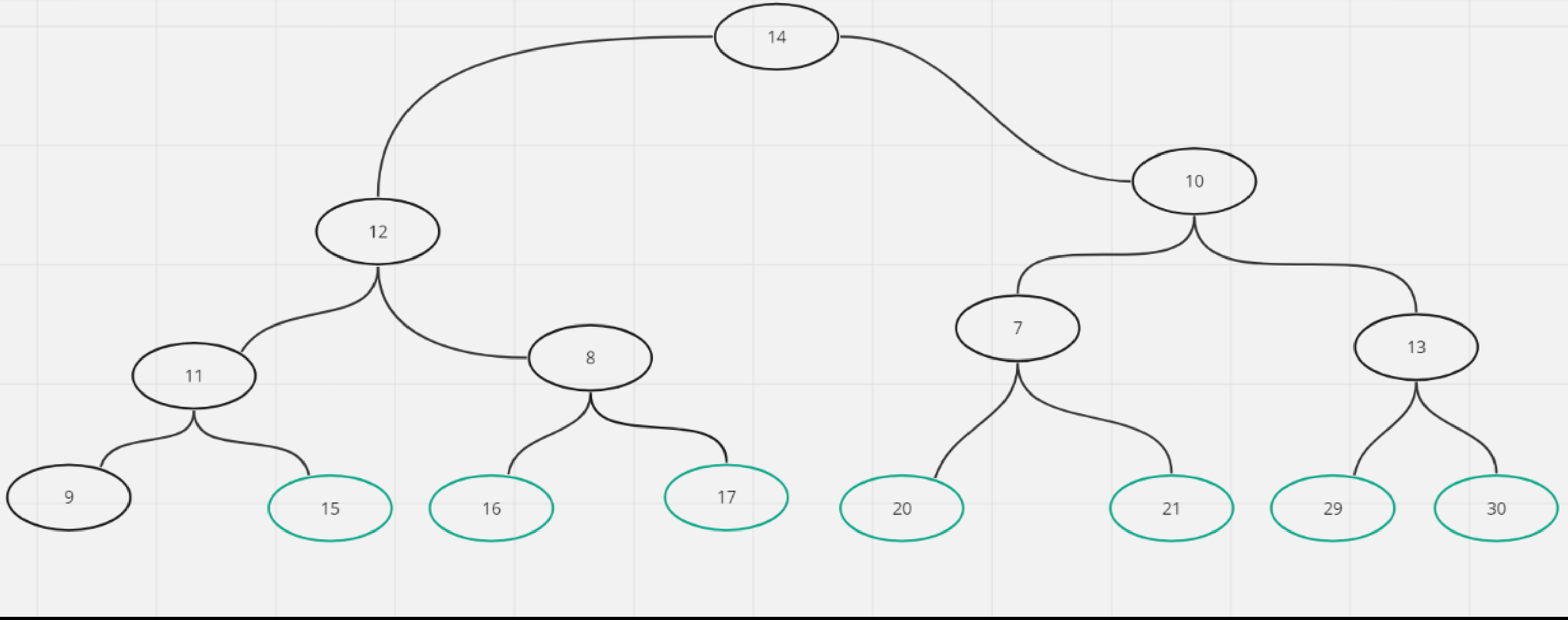

16
15
14
11
12
7
13
9
10
8
17
20
21
29
30

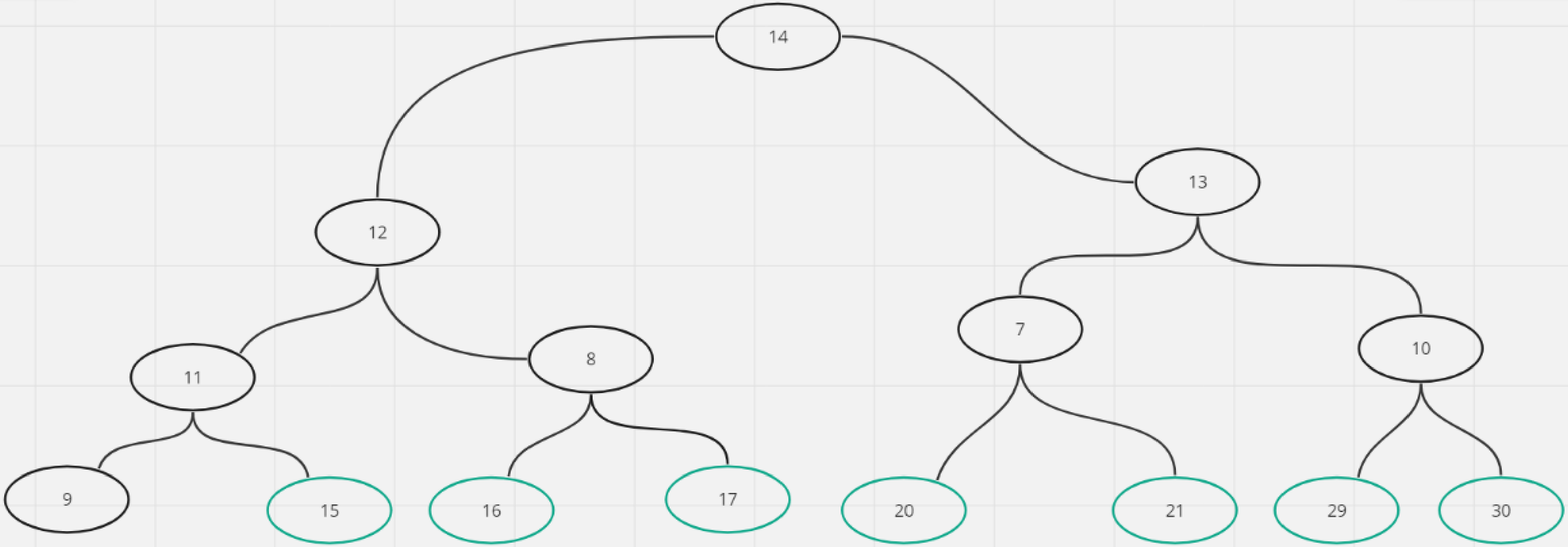


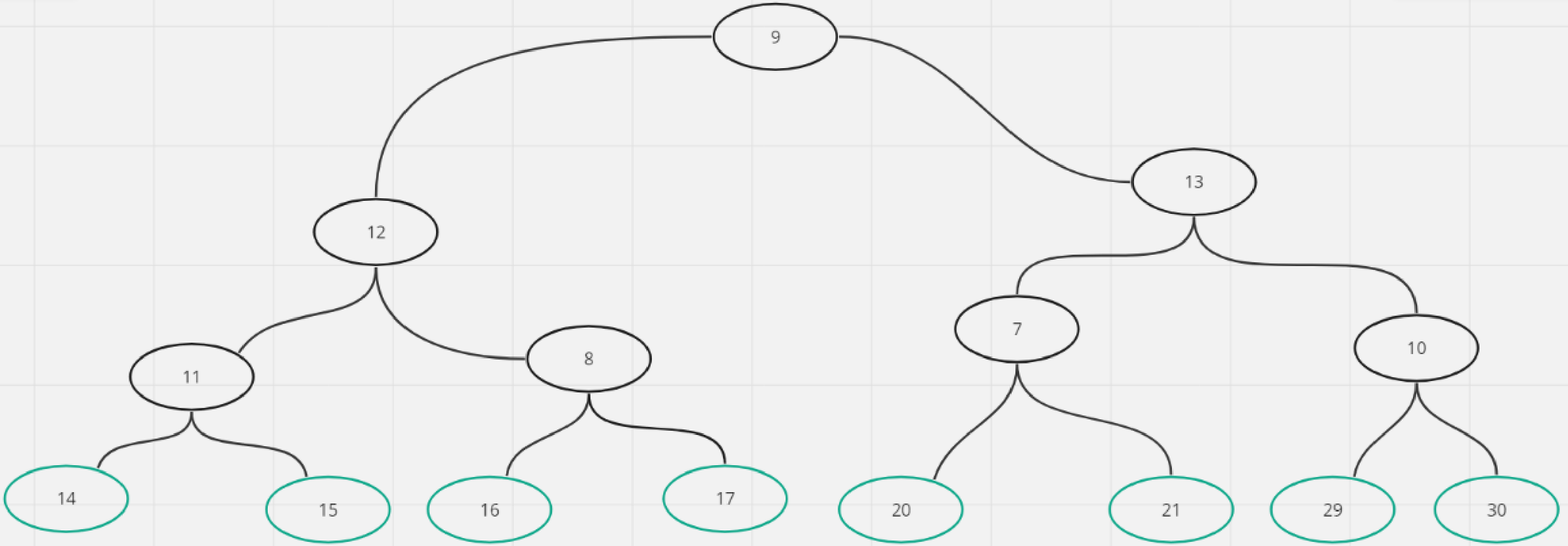

15
8
14
11
12
7
13
9
10
16
17
20
21
29
30

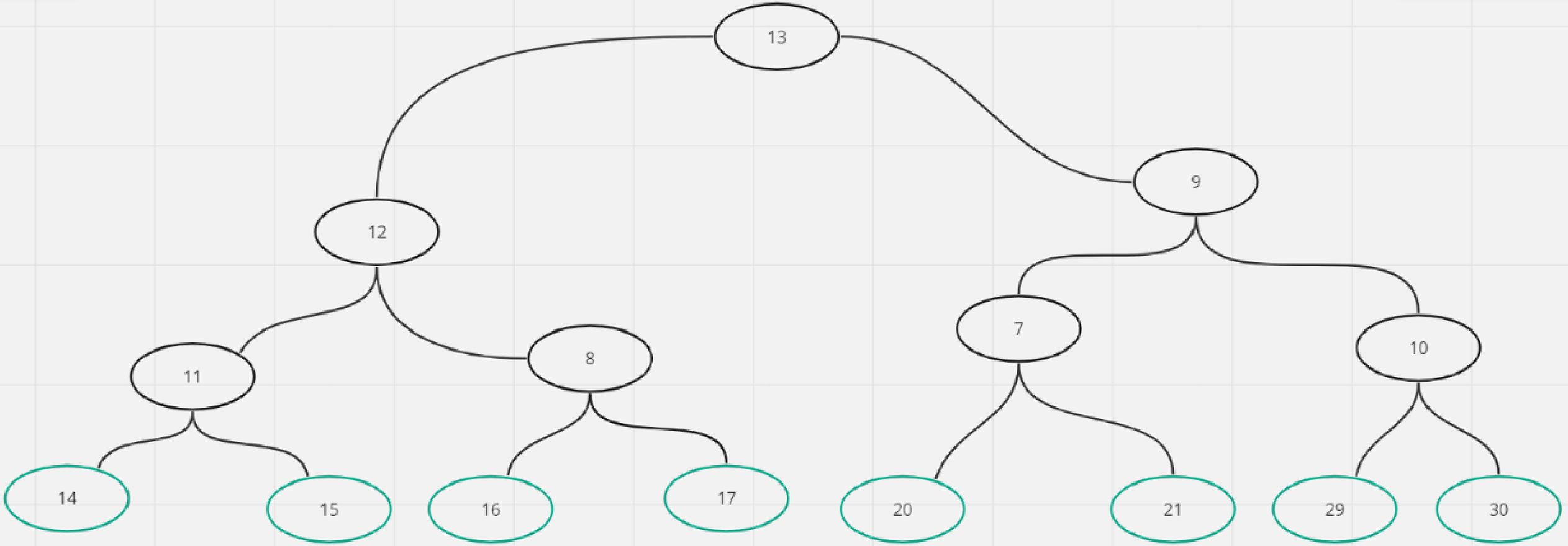


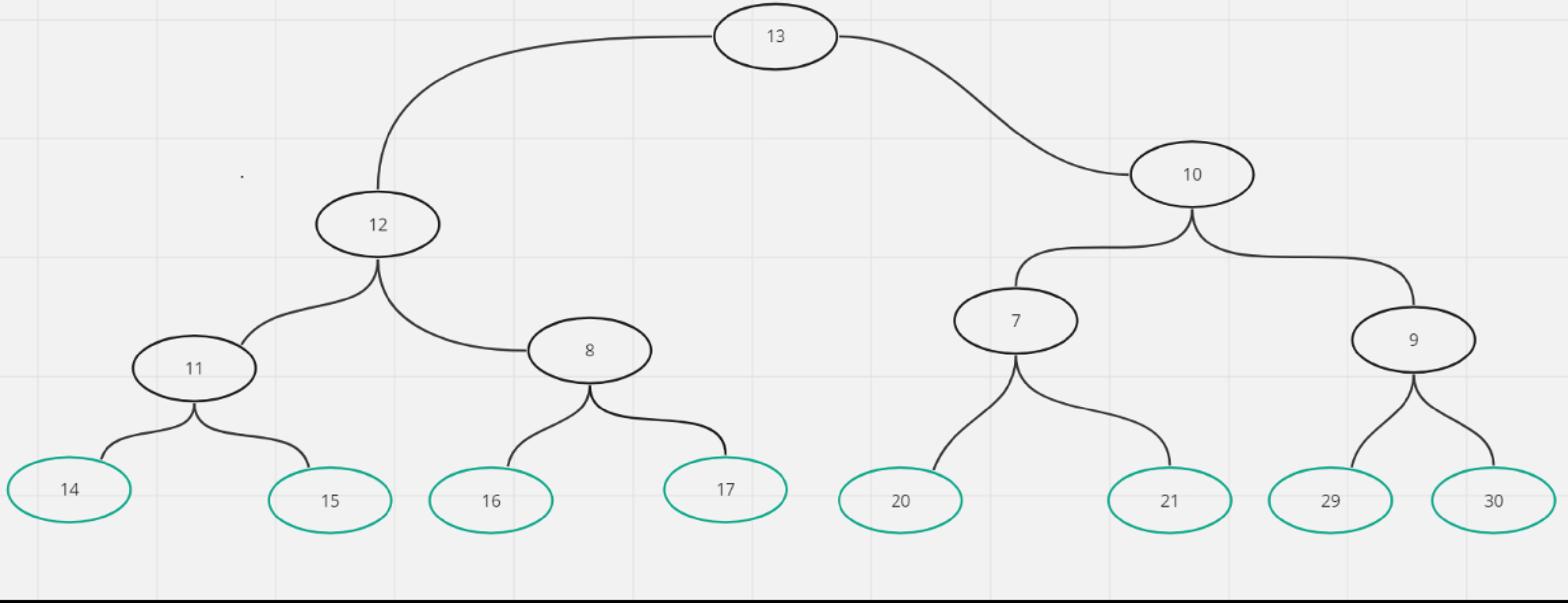

10
12
14
11
8
7
13
9
15
16
17
20
21
29
30

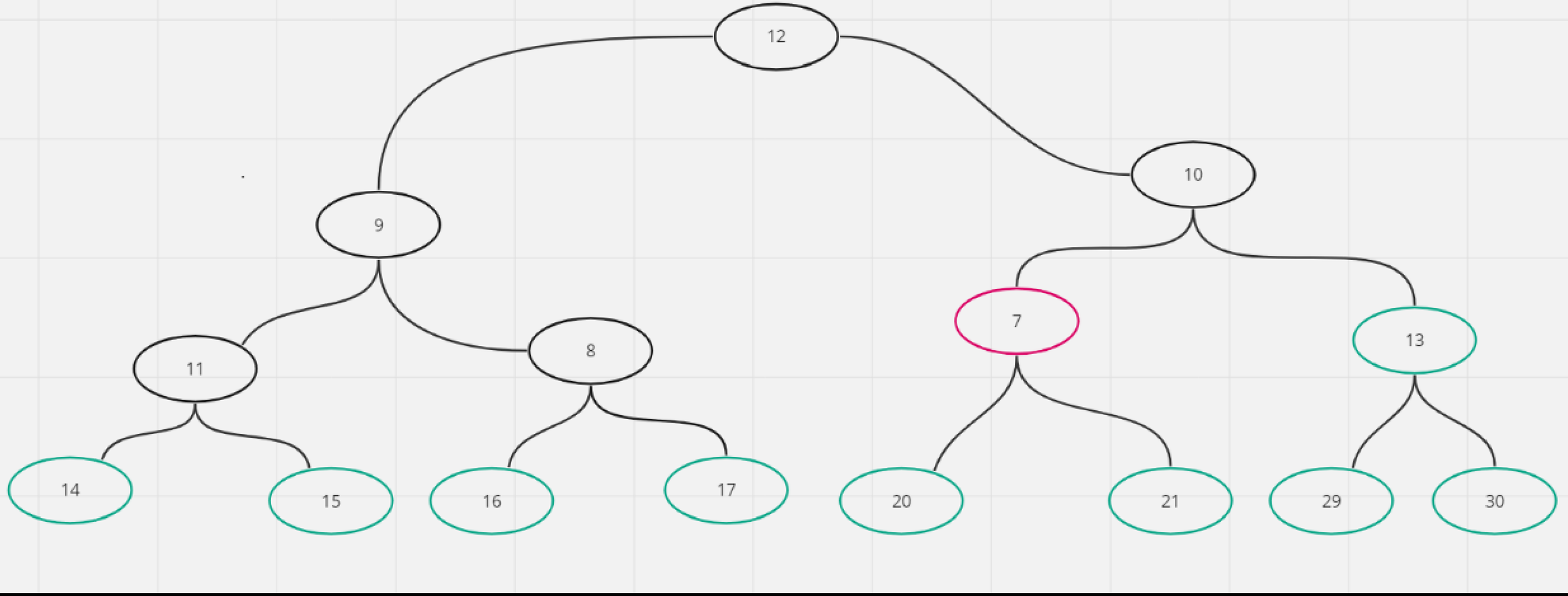

14
12
10
11
8
7
13
9
15
16
17
20
21
29
30

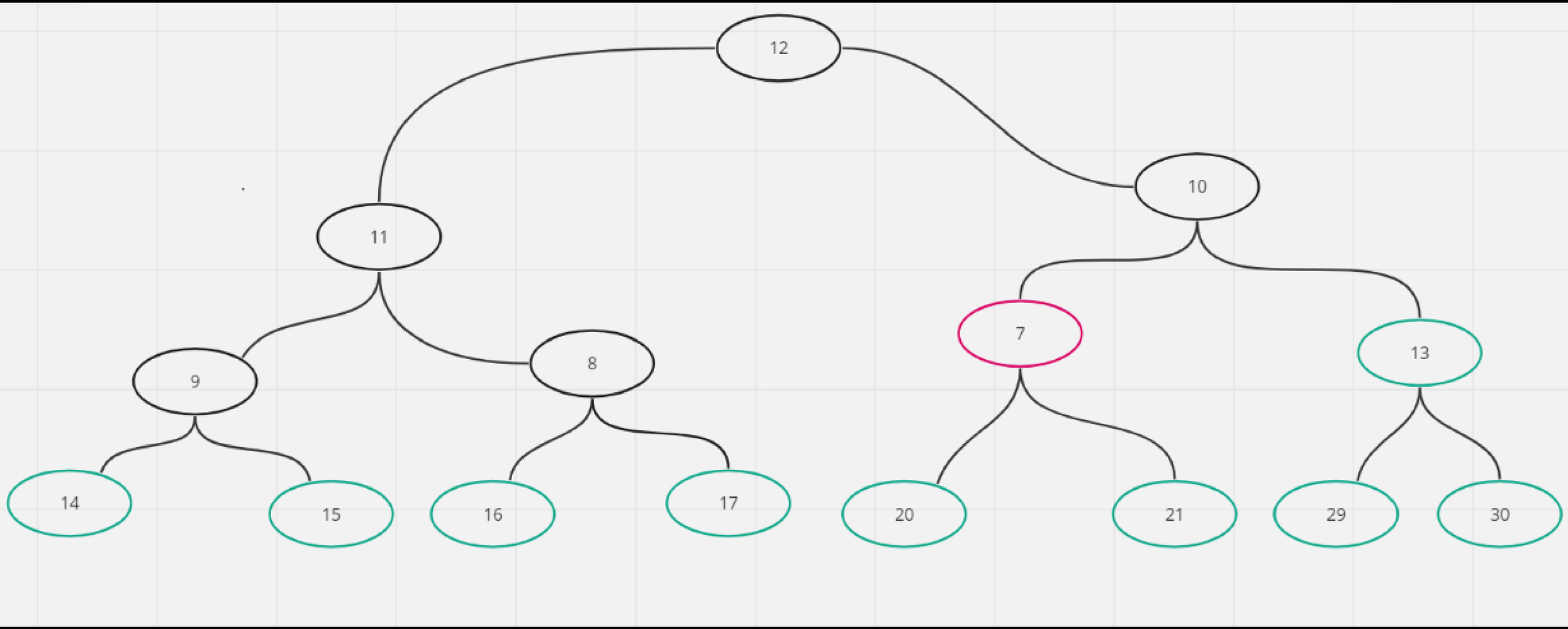

14
12
13
11
8
7
10
9
15
16
17
20
21
29
30

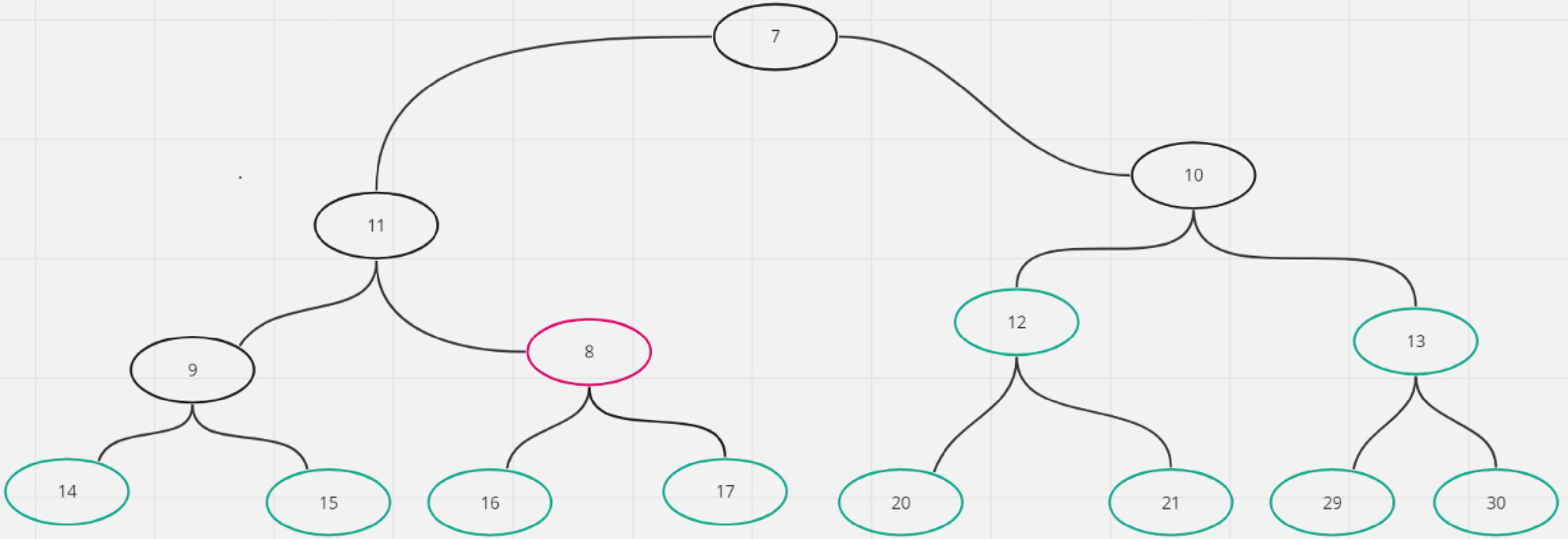

9
12
13
11
8
7
10
14
15
16
17
20
21
29
30

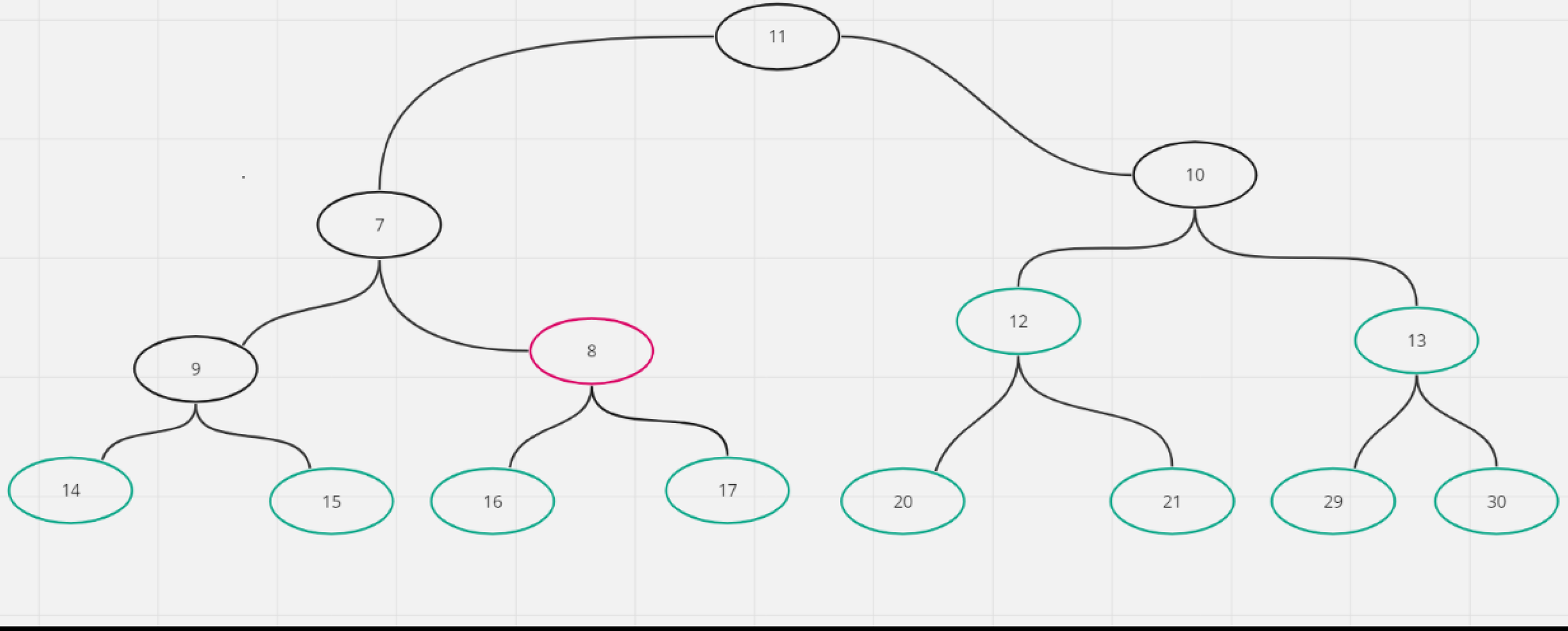

13
12
9
11
8
7
10
14
15
16
17
20
21
29
30

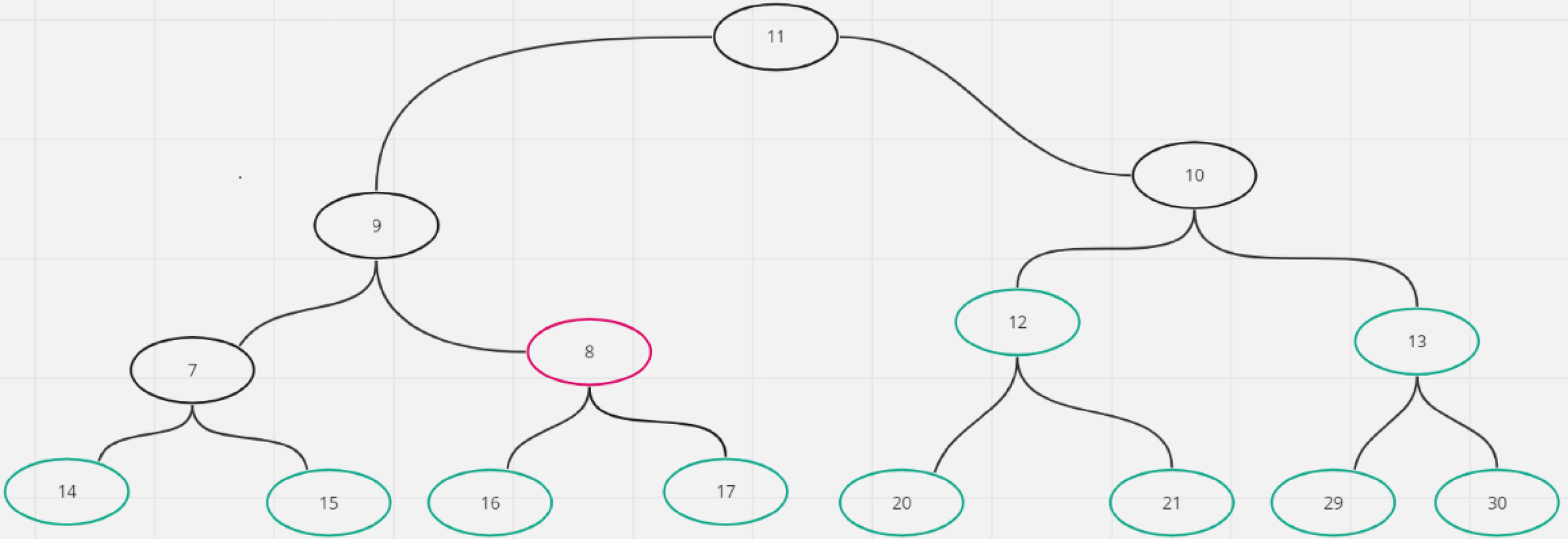

13
12
10
11
8
7
9
14
15
16
17
20
21
29
30

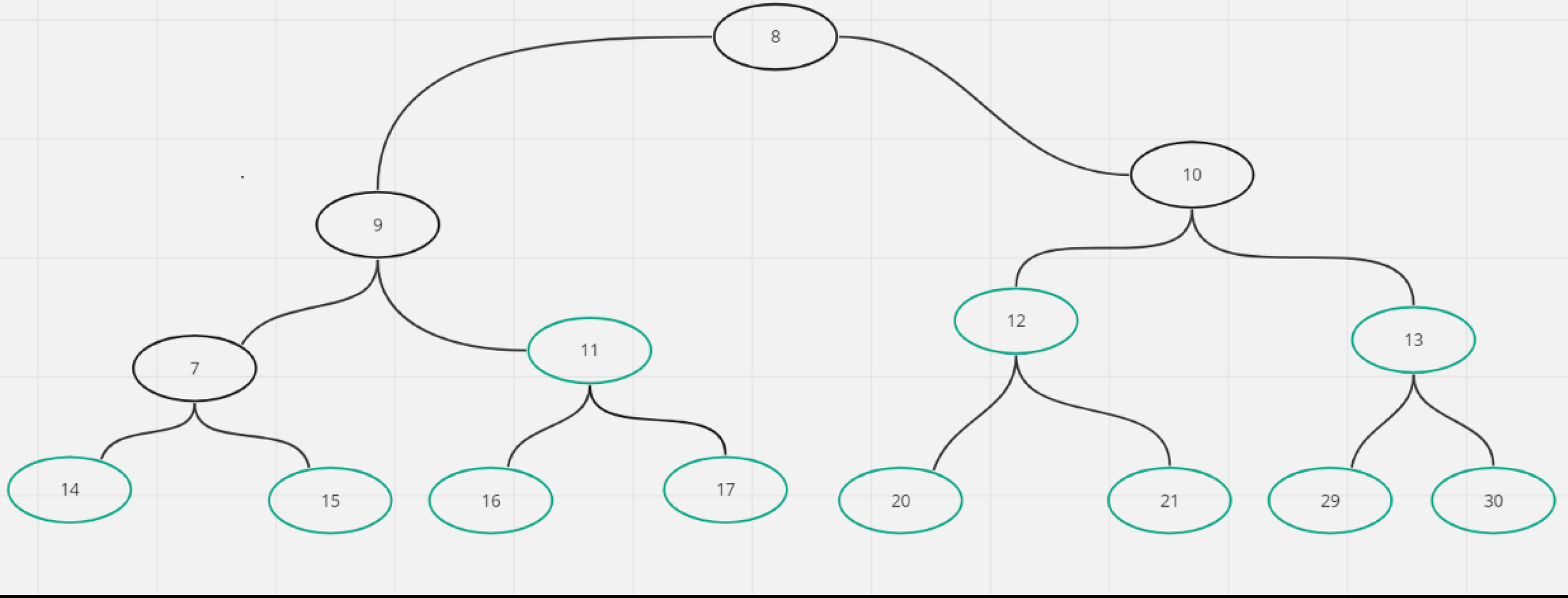

12
9
10
11
8
7
13
14
15
16
17
20
21
29
30

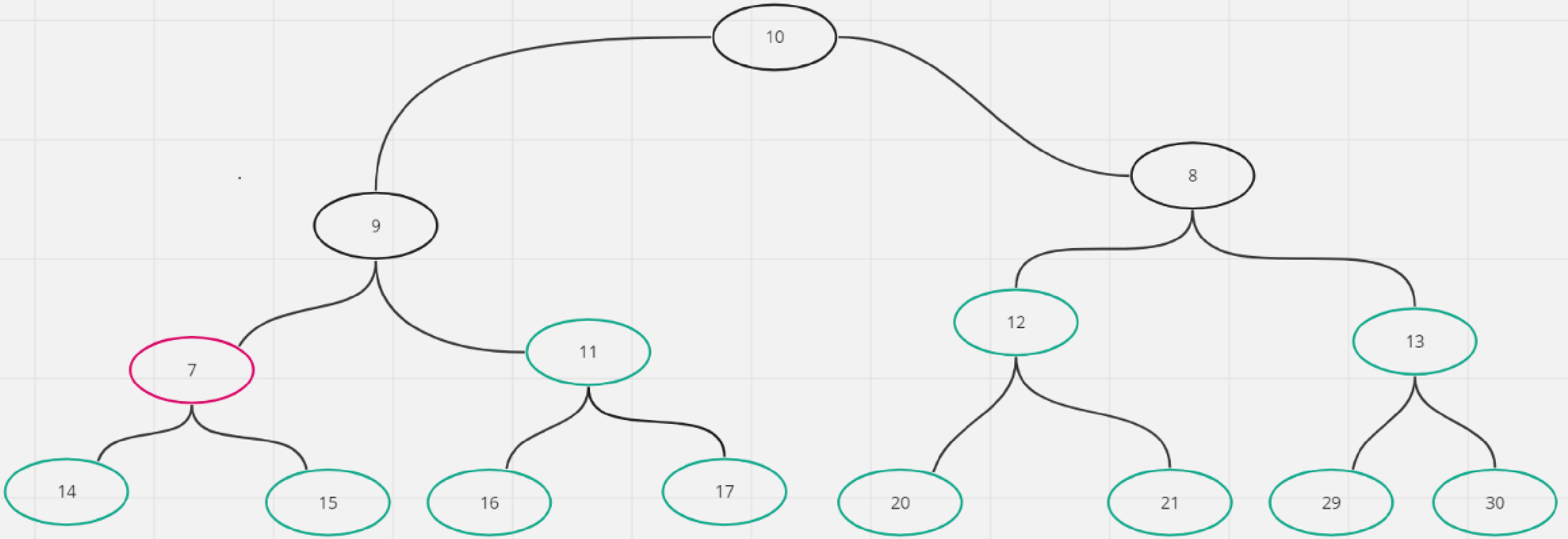

12
11
10
9
8
7
13
14
15
16
17
20
21
29
30

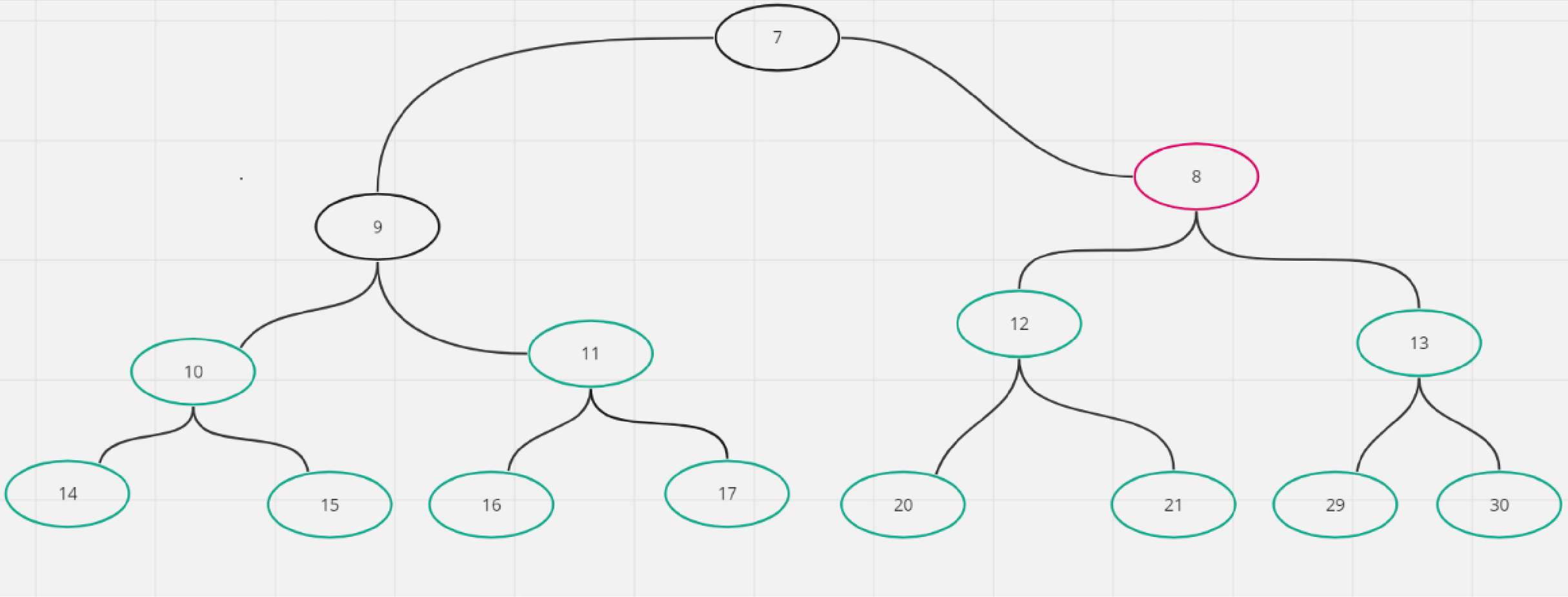

7
11
10
9
8
12
13
14
15
16
17
20
21
29
30

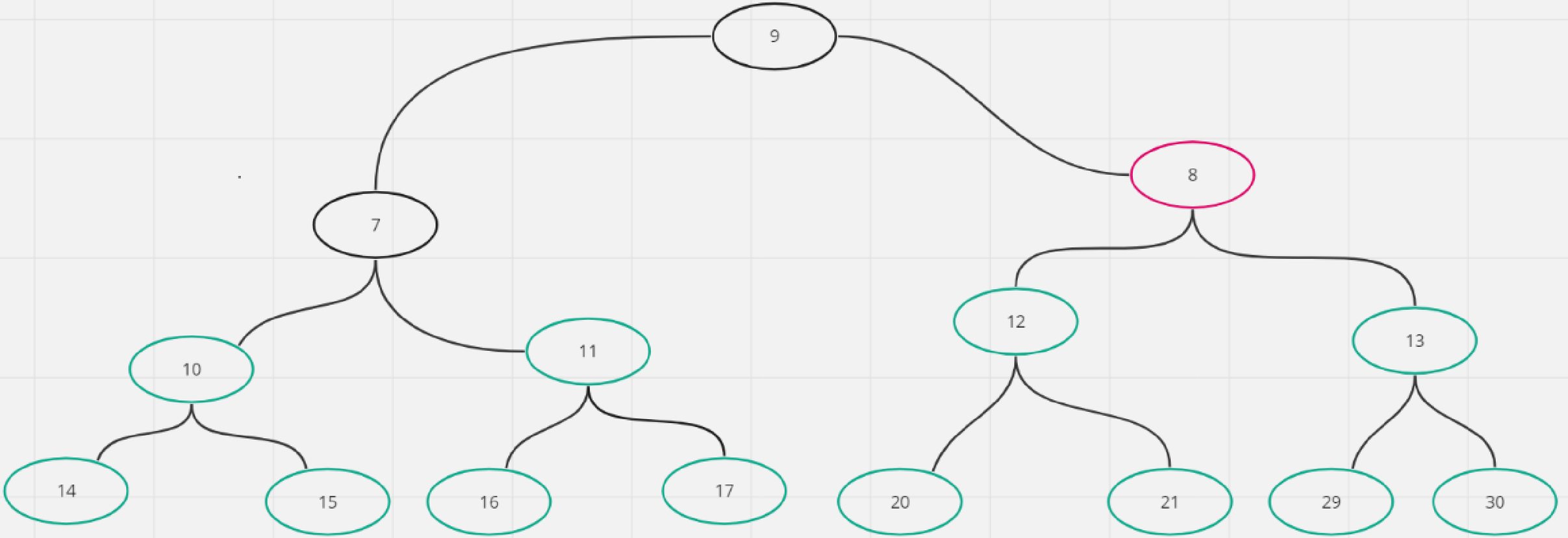

11
7
10
9
8
12
13
14
15
16
17
20
21
29
30

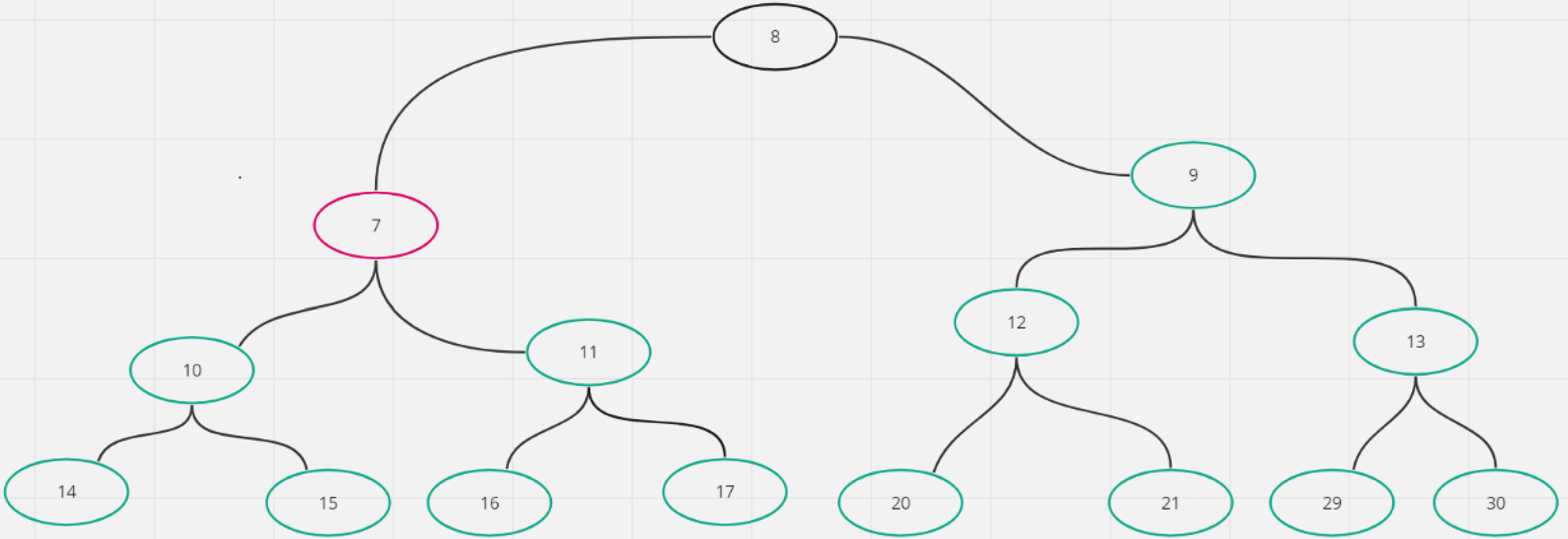

11
9
10
7
8
12
13
14
15
16
17
20
21
29
30

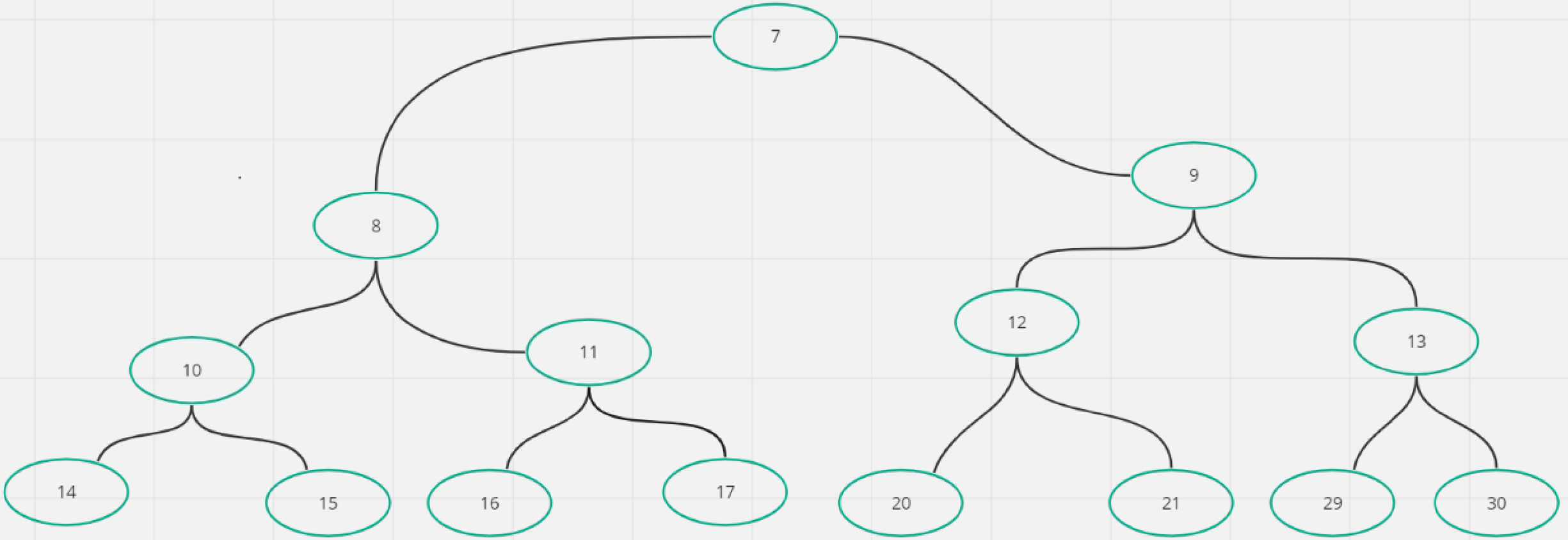

8
9
10
7
11
12
13
14
15
16
17
20
21
29
30


10
9
8
7
11
12
13
14
15
16
17
20
21
29
30


7
9
8
10
11
12
13
14
15
16
17
20
21
29
30


9
7
8
10
11
12
13
14
15
16
17
20
21
29
30


8
7
9
10
11
12
13
14
15
16
17
20
21
29
30


7
8
9
10
11
12
13
14
15
16
17
20
21
29
30

We solve all the issues, now let’s go to the implementation.