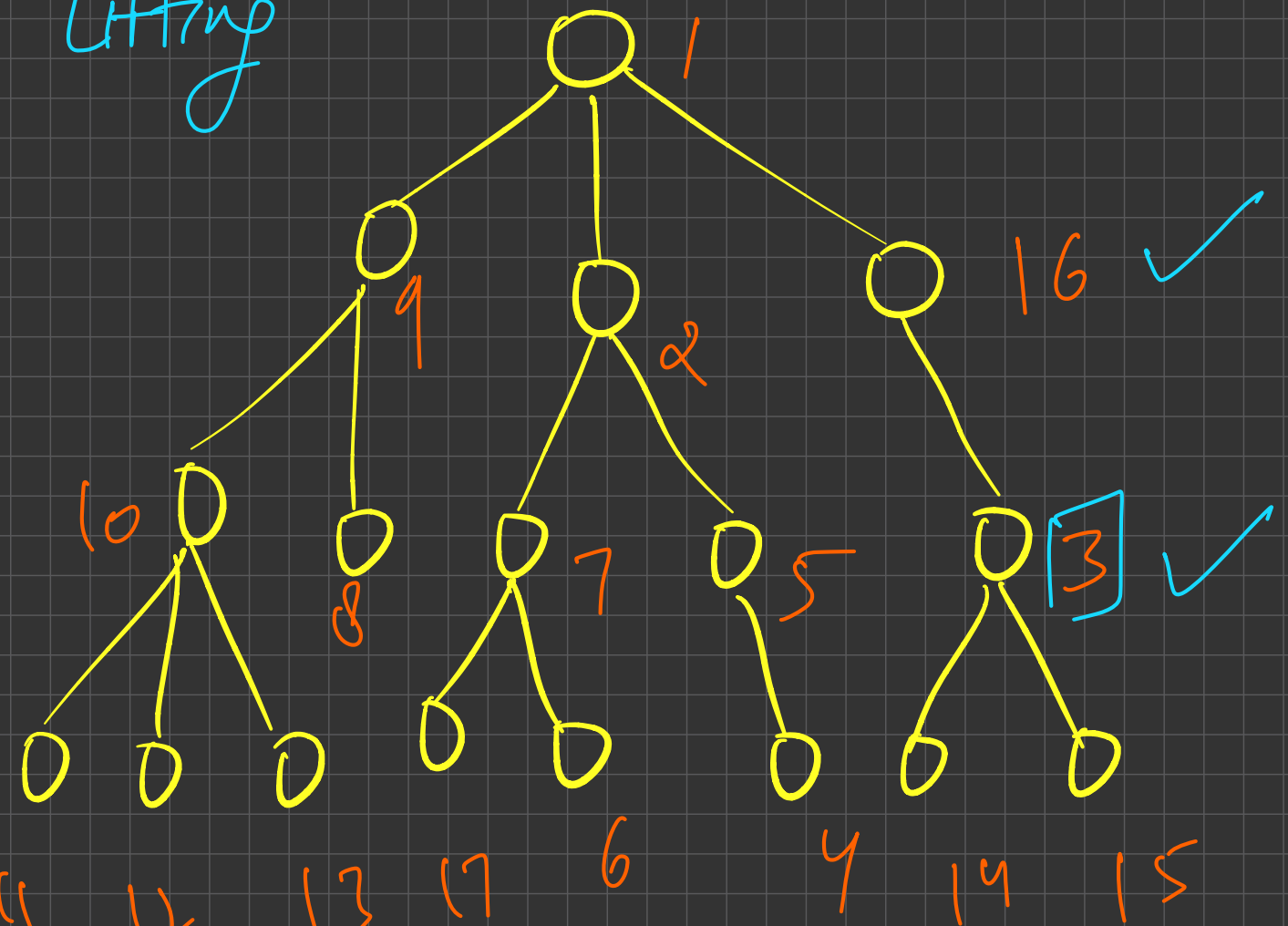
Binary Lifting + LCA

Trees 3

– Priyansh Agarwal

- Diameter of Tree
- In time — out time trick
- Properties of Trees
- Terminologies

Binary Lifting

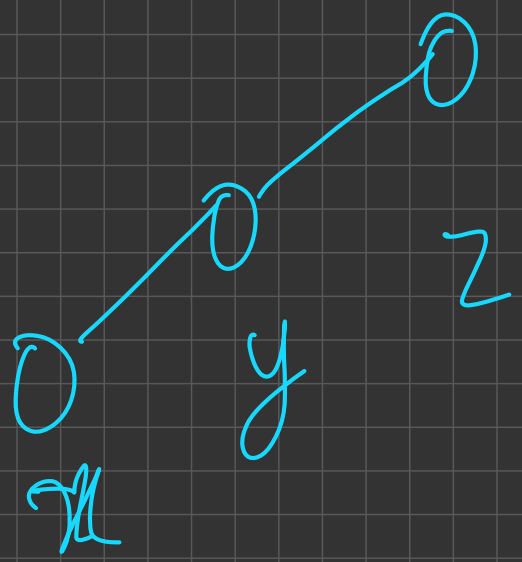


find the second parent of 14

$$p2[x] = p1[p1[x]]$$

$$\frac{y}{\frac{}{}}$$

$$\frac{}{y}$$



z

Find the 105th parent of X

```
for (int i = 0 ; i < 105 ; i++) {
    x = parent[x]
}
```

O(n)

kth parent of X

O(k)

find out the 30th parent of X

1 1 1 1 0

X → 2nd parent of X (Y)

Y → 4th parent of Y (A)

A → 8th parent of A (B)

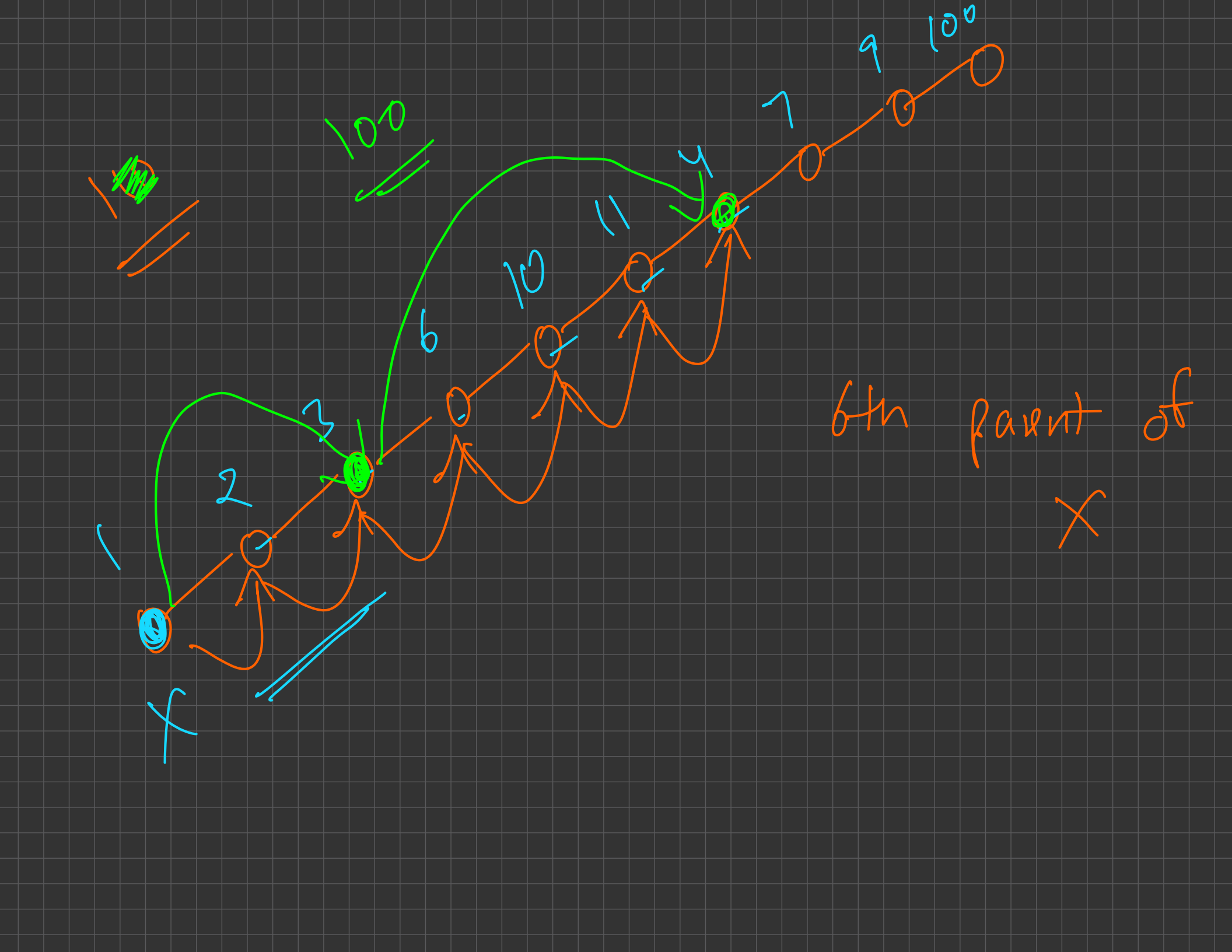
B → 16th parent of B (C)

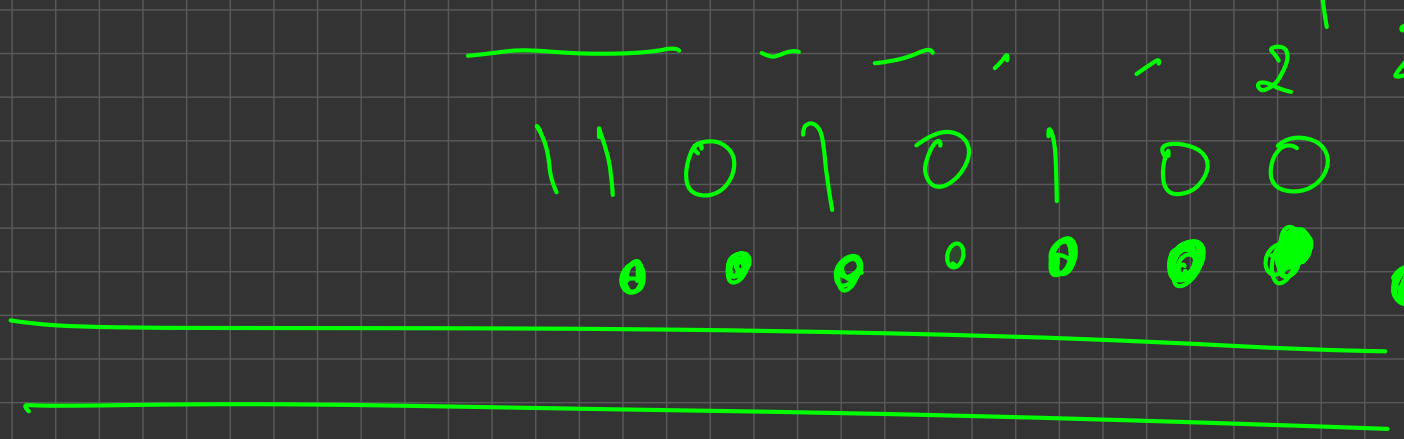
$X \rightarrow$ 1st parent, 2nd parent, 4th parent, 8th parent, 16th parent, 32nd parent

$\downarrow$

2^k th parent

such that $\boxed{2^k} \leq n$


100
100
1
2
3
4
5
6
7
8
9
10
11
x
6th parent of
x

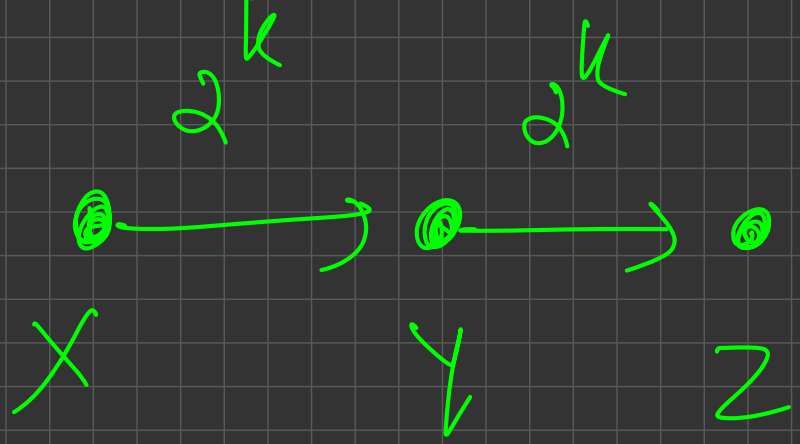


$X \xrightarrow{+1} Y \xrightarrow{+8} Z \xrightarrow{+32} A \xrightarrow{+128} B \xrightarrow{+256} C$

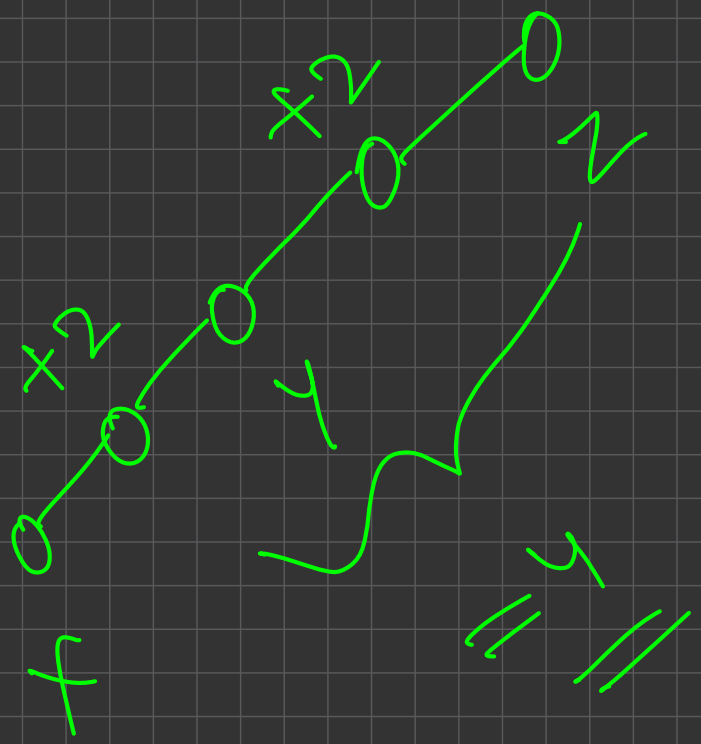
kth parent of X

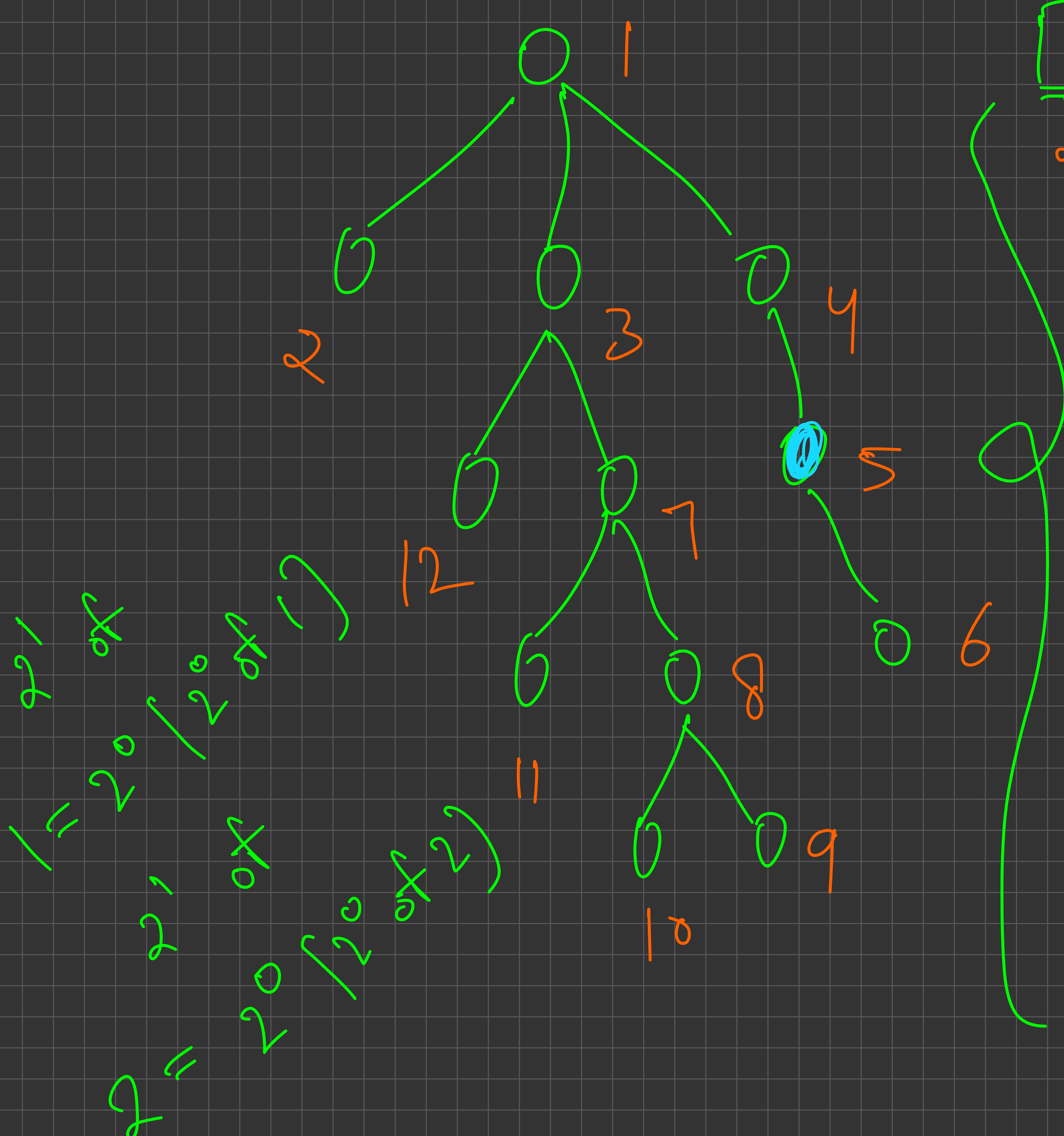
O(no. of set bits in k) < O(log k)

2^0 th parent of every node → 2^1 th parent of every node



2^{k+1} th parent of x





1	-1	-1
2	1	-1
3	1	-1
4	1	-1
5	4	1
6	5	4
7	3	
8	7	
9	8	
10	8	
11	7	
12	3	

1st 2nd 4th 8th --- $2^{\log(n)}$ th

parent

$O(n \log n)$

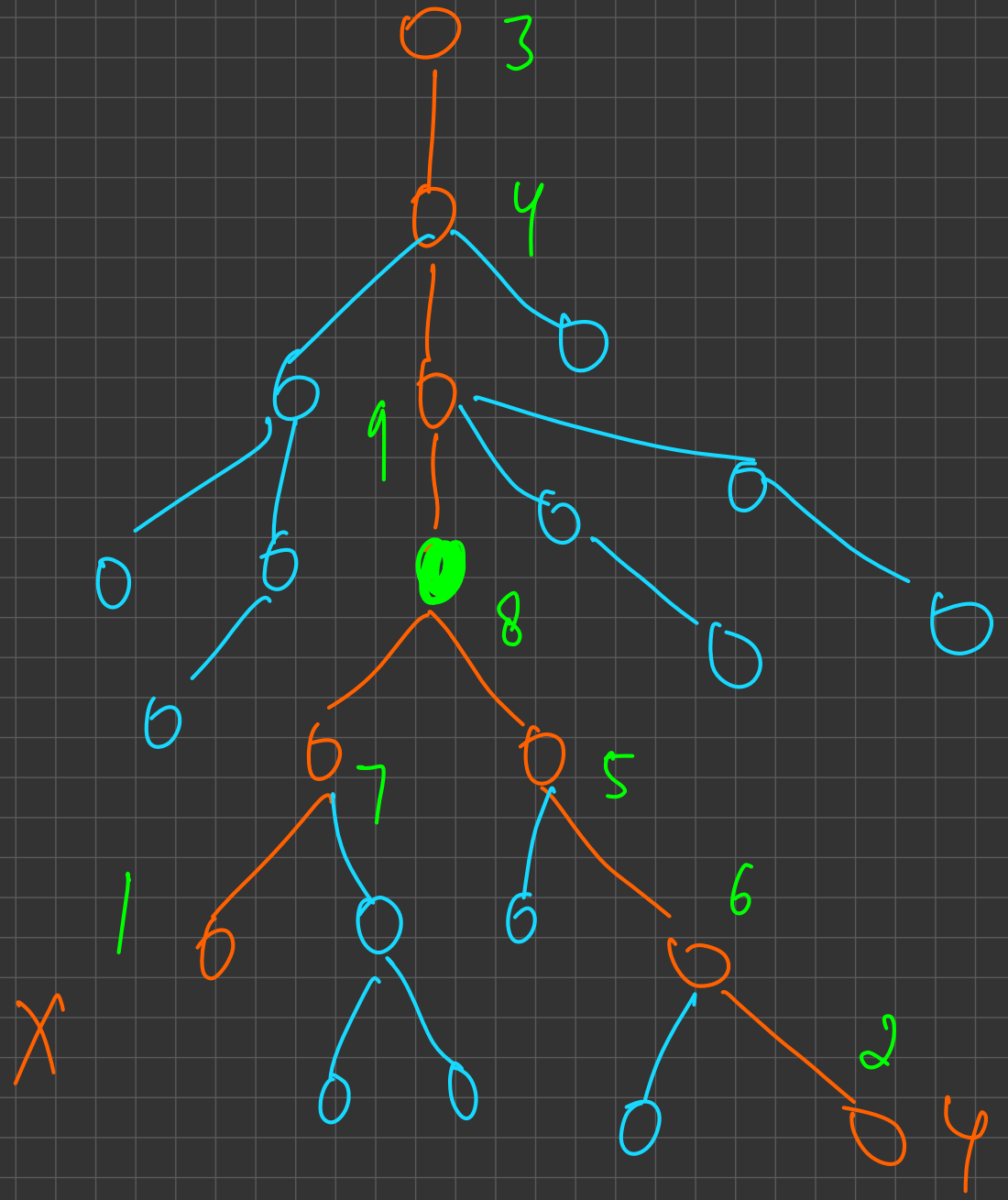
$$= 2^k\text{th parent of } \left[2^k\text{th parent of } x \right]$$

$$2^k + 2^k = 2 \cdot (2^k) = 2^{k+1}$$

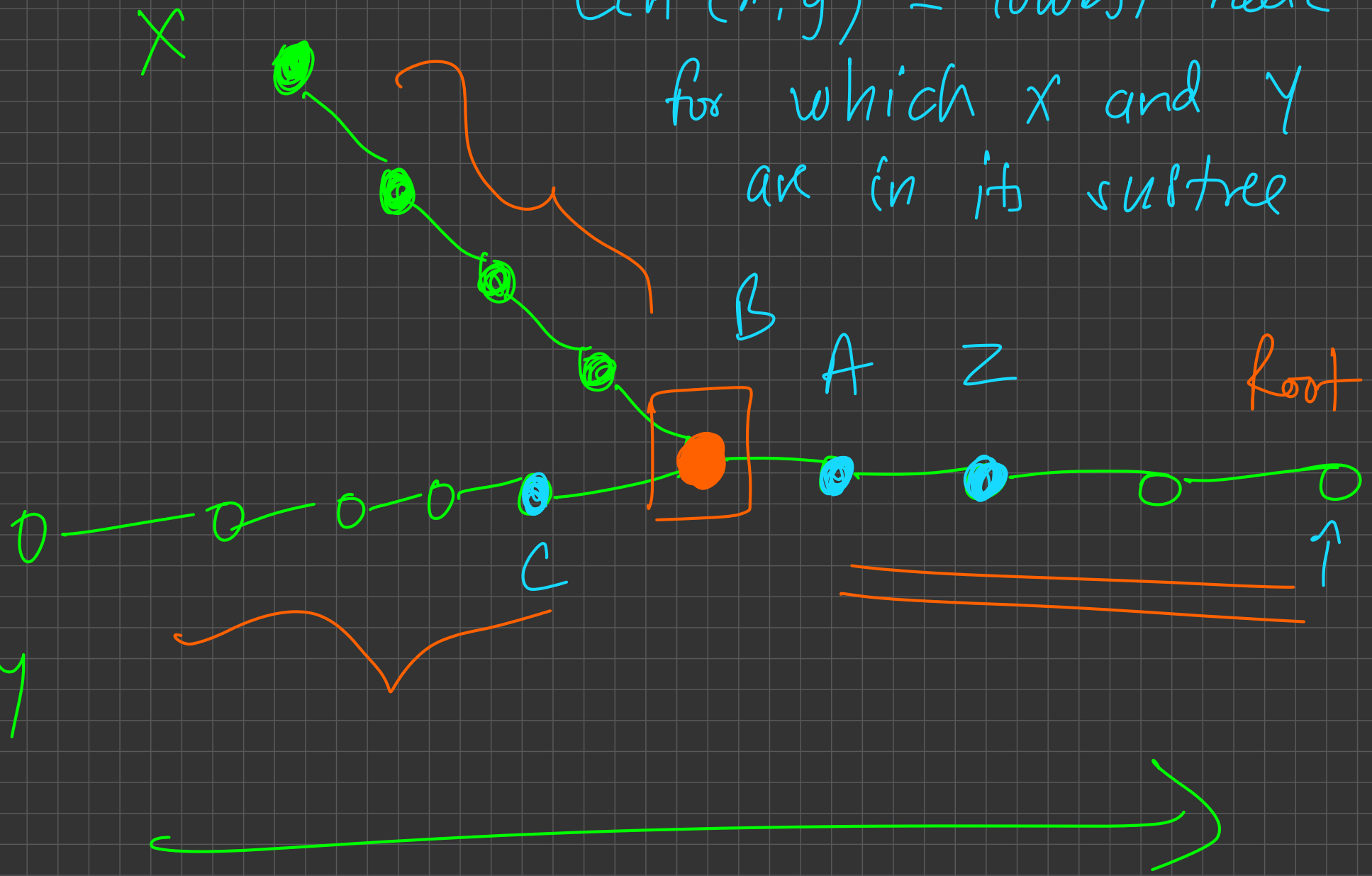
Binary Lifting

- Find kth Parent of any node in a Tree Problem Link
- Find LCA of 2 nodes Problem Link
  - O(logn) solution
  - Using in-time out-time trick -> O(logn)

Lowest Common Ancestor



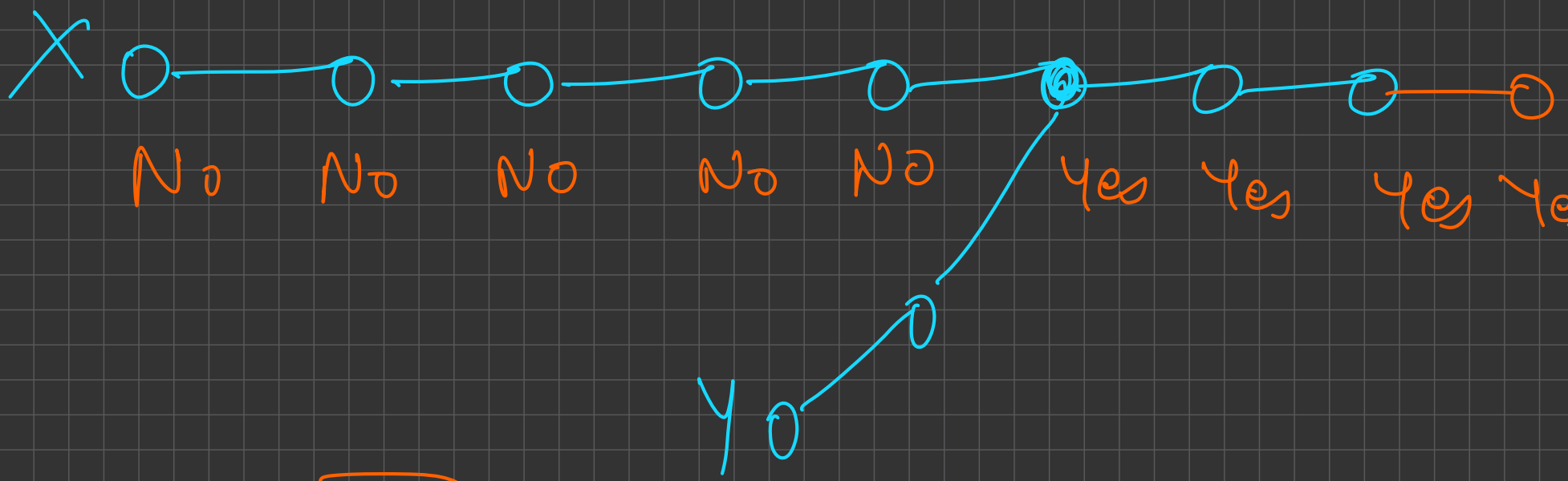
LCA (x, y) = lowest node for which x and y are in its subtree



LCA(x, y) = farthest node from root for which both x and y are in its subtree

= the lowest parent of x for which y is in its subtree

= the lowest parent of y for which x is in its subtree



check if kth parent of X has Y in it

subtree

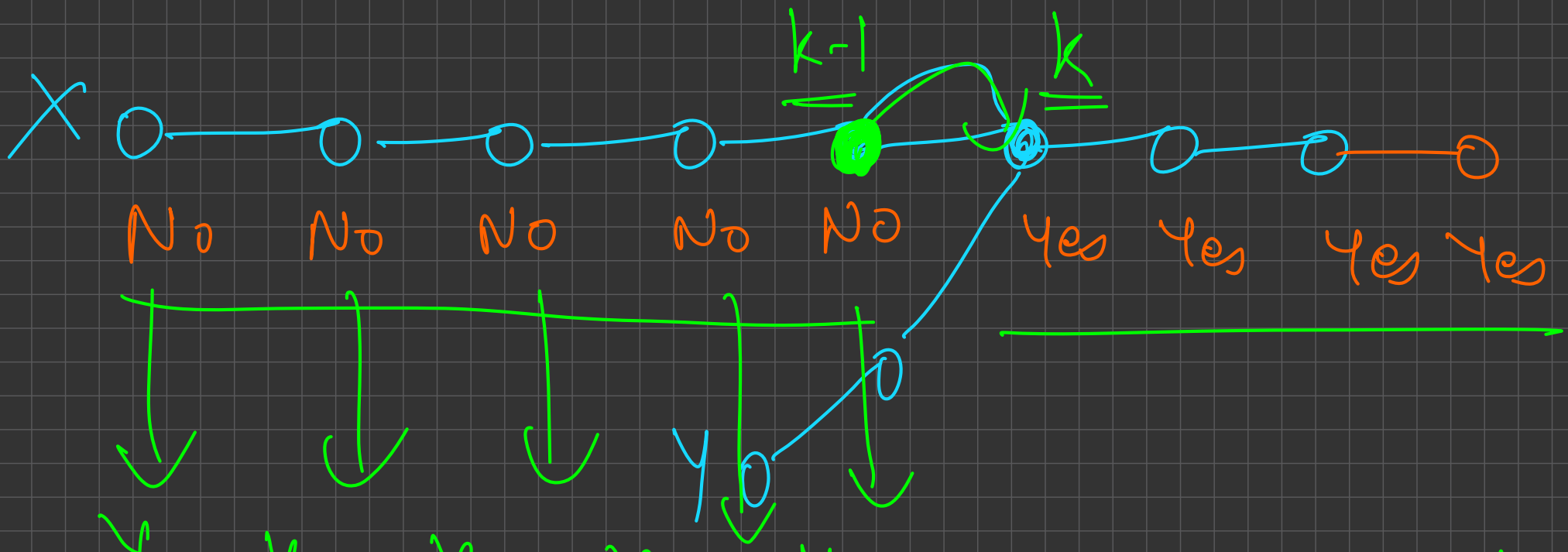
$\Rightarrow Z$

$\log n$ $\Leftarrow$ ① find out kth parent of X

$O(1)$ ② check if Z is an ancestor of Y

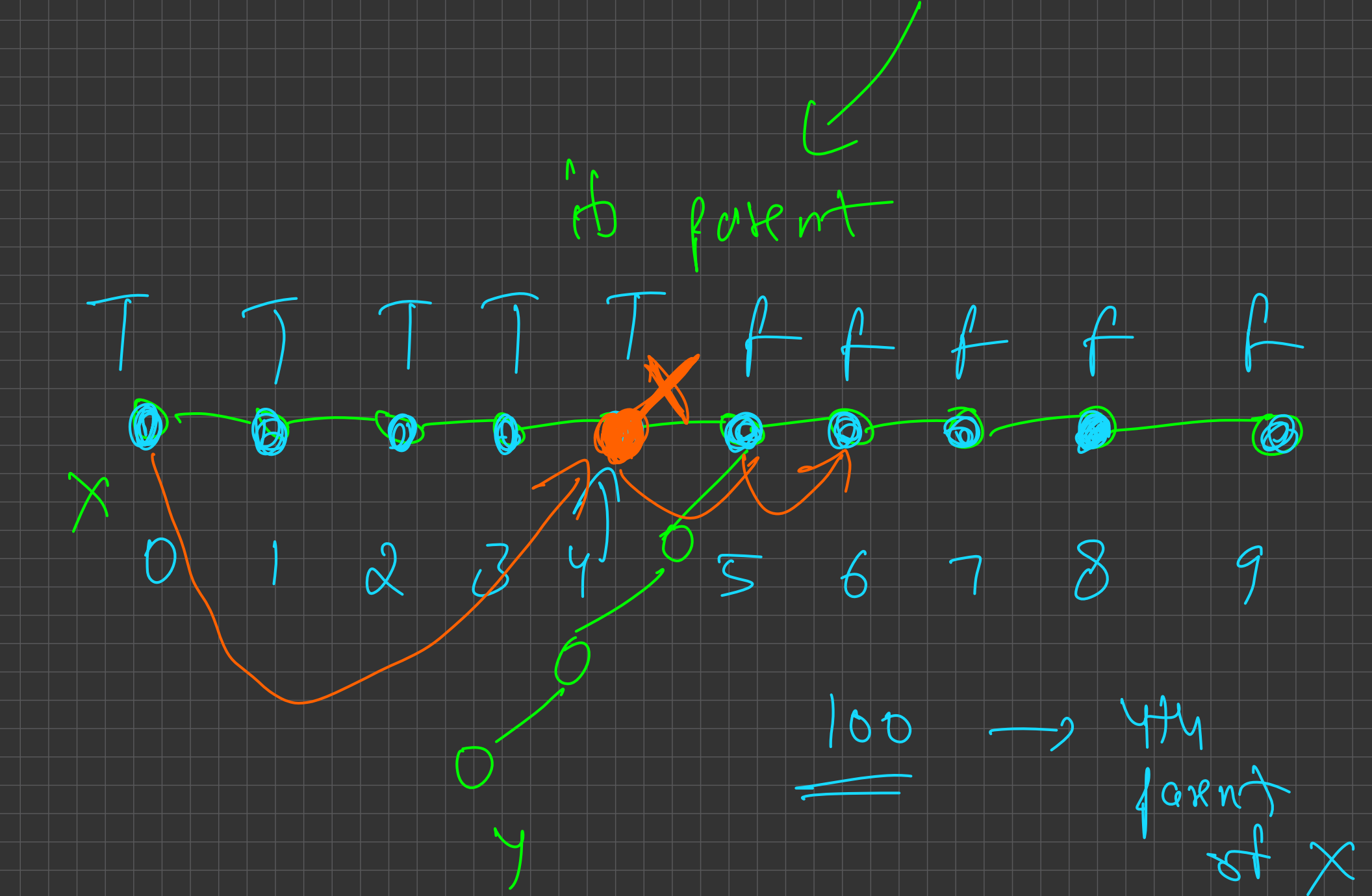
($\log n \cdot \log n$)

o ($\log^2 n$)



lca = lowest parent of X which is an ancestor of Y

= highest parent of X which is not an ancstr of Y



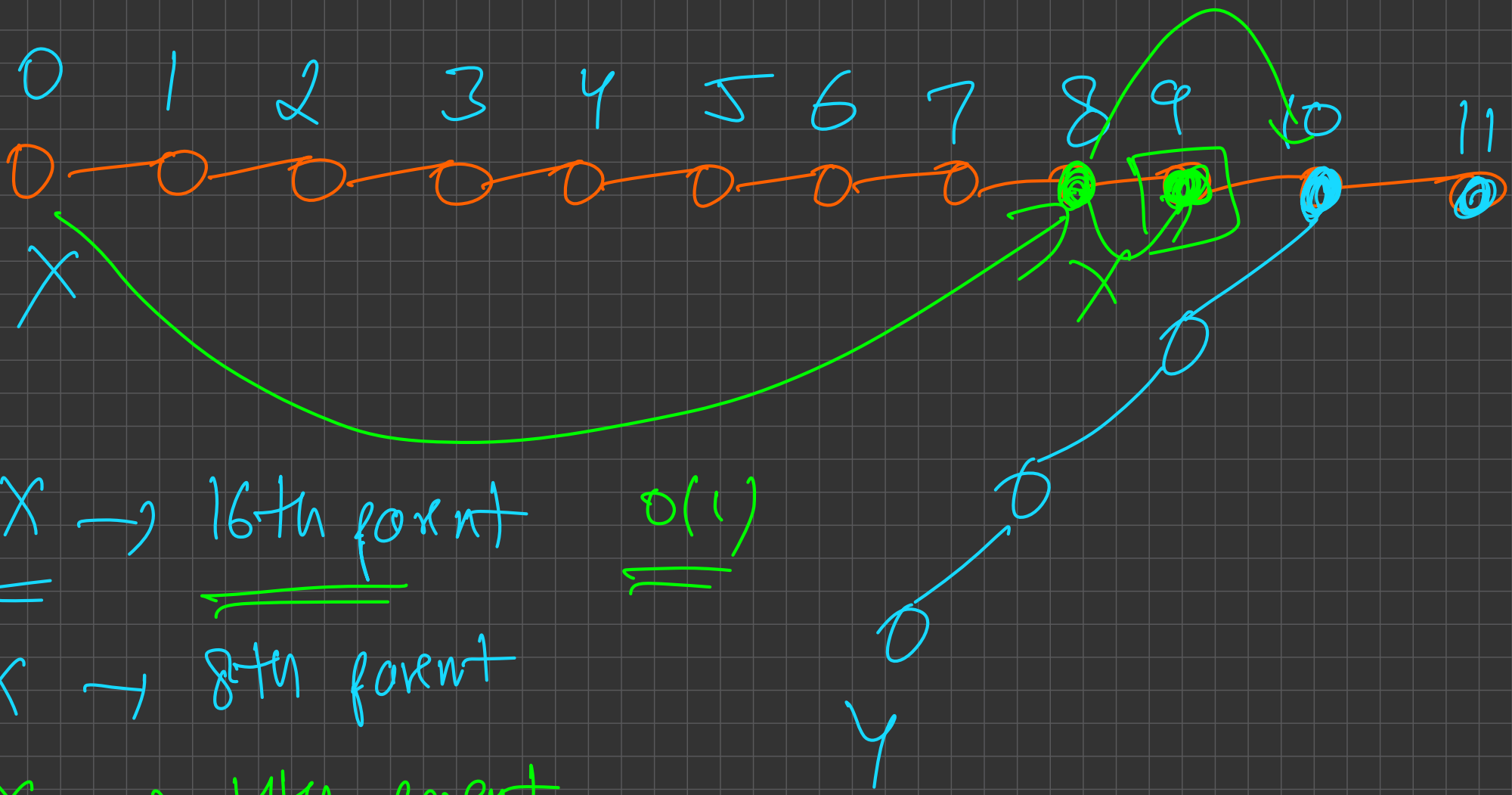
100 → 4th parent of X

8th parent of X → O(1)

4th parent of $X \rightarrow O(1)$ $\cup$ X

2nd parent of $X \rightarrow O(1)$

1st parent of X



X → 16th parent O(1)

X → 8th parent

X → 4th parent

X → 2nd parent

X → 1st parent

0 1 2 3 4 5 6 7 8 9 10 11 12

T T T T T T T T T f f f f f

00

1000 → T ✓

100 → T

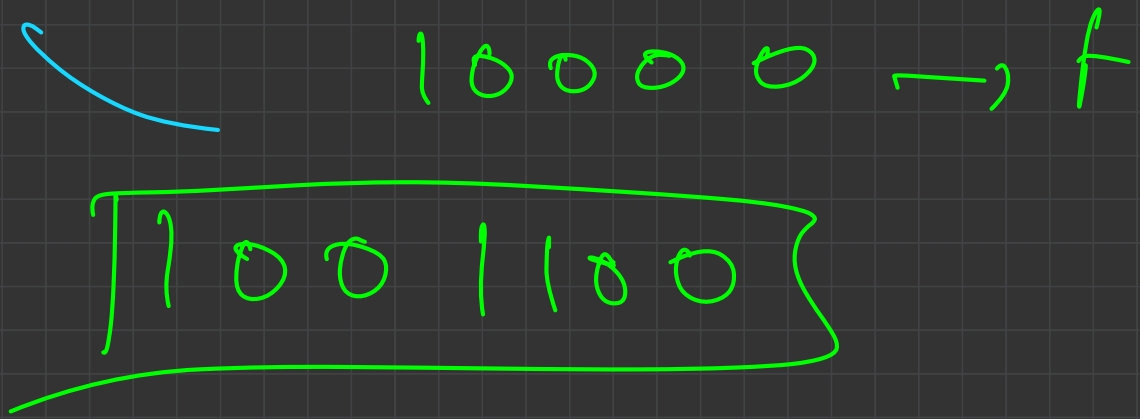
10 → f

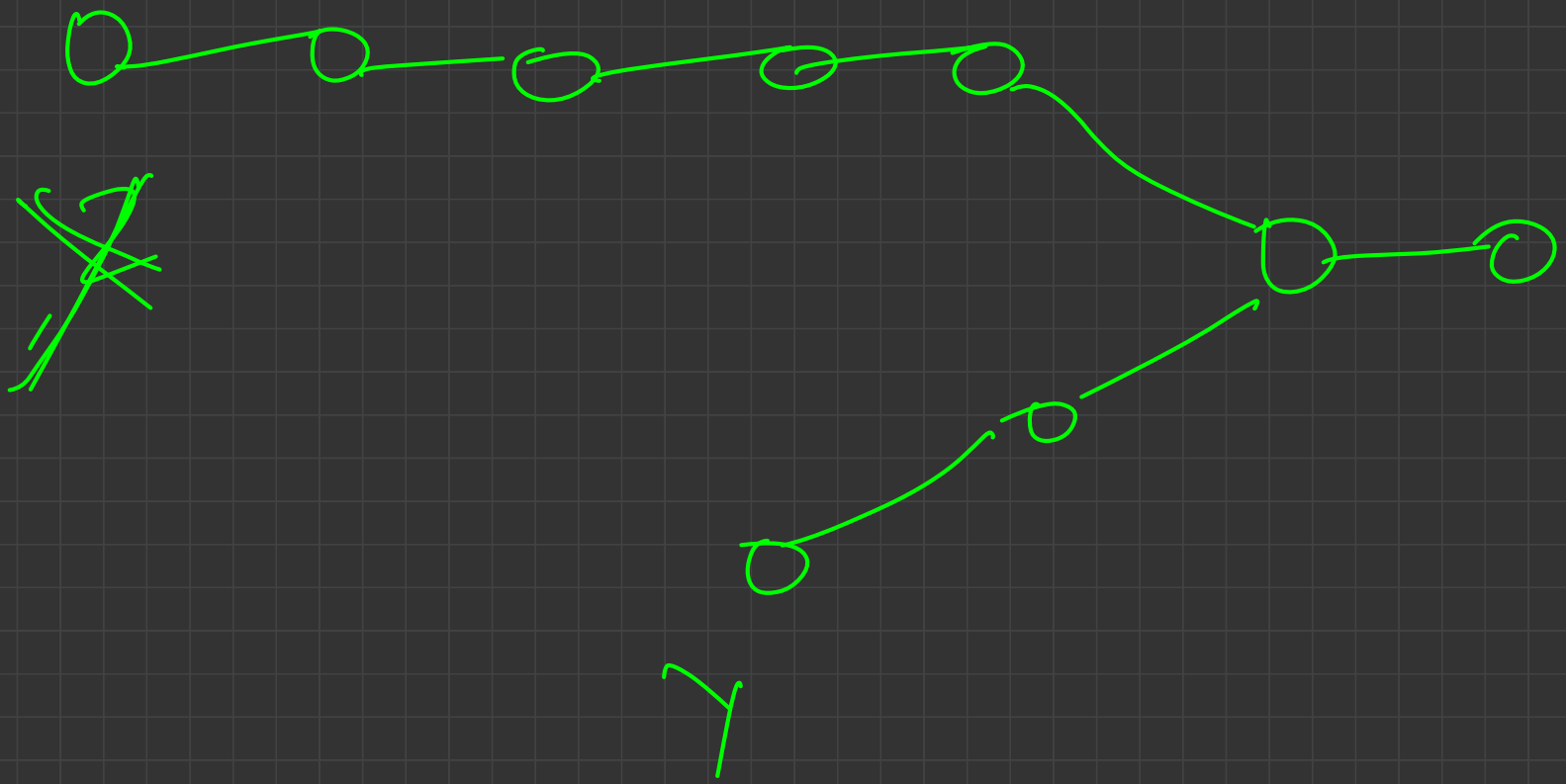
1 → f

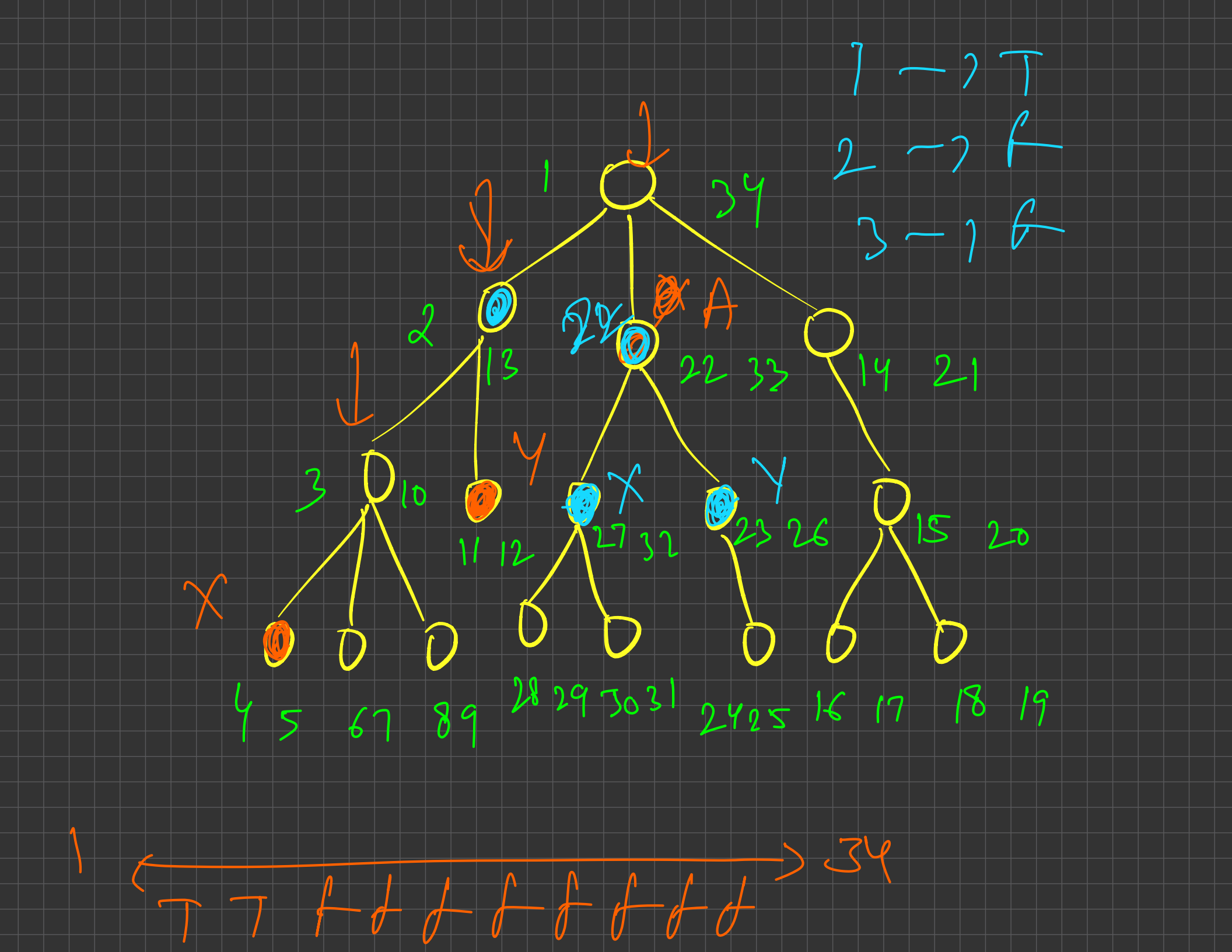
1000000 → f

1000000 → T ✓

100000 → f


10000 → f
1001100


X
Y

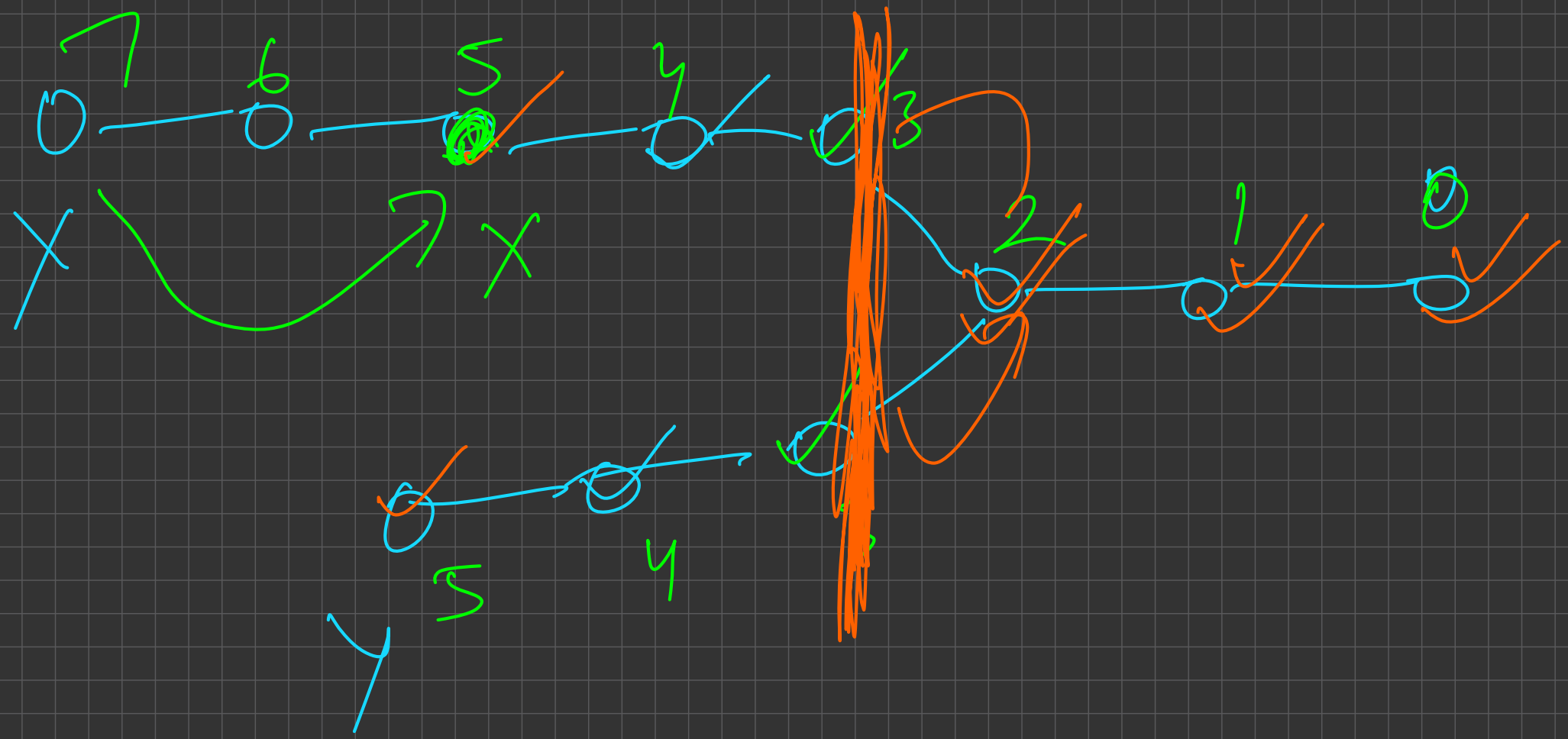

1 → T
2 → F
3 → G
1
34
2
13
24
A
22
33
14
21
3
10
Y
11
12
X
Y
27
32
23
26
15
20
X
4
5
6
7
8
9
28
29
30
31
24
25
16
17
18
19
1
34
T T F F F F F F F F

T T T T T T f f f f f

In time → O(log2n)

↓

Bit manipulation → O(log n)



level(y)

5

level(x)

7

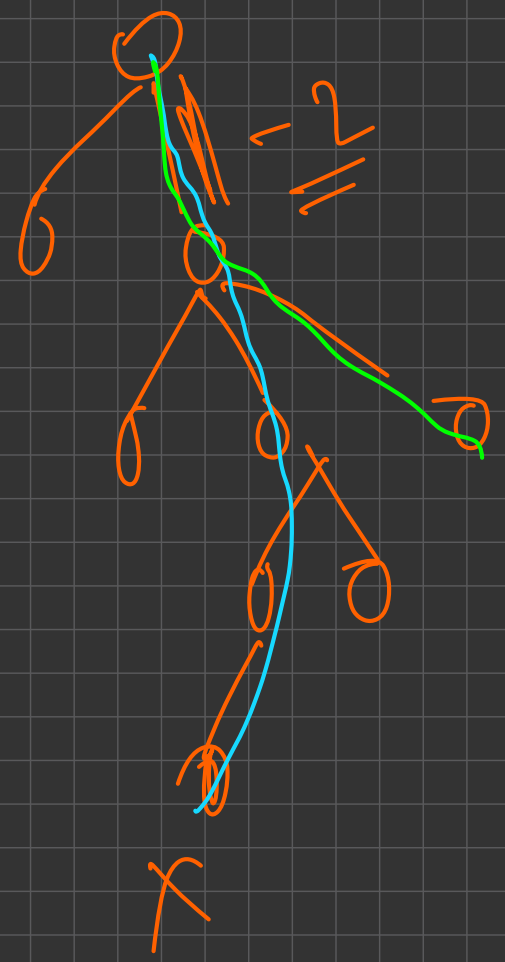
Bring X and Y to the same level

find out the biggest kth parent of X for which it is not equal to kth parent of Y

= Z

return 1st parent of Z

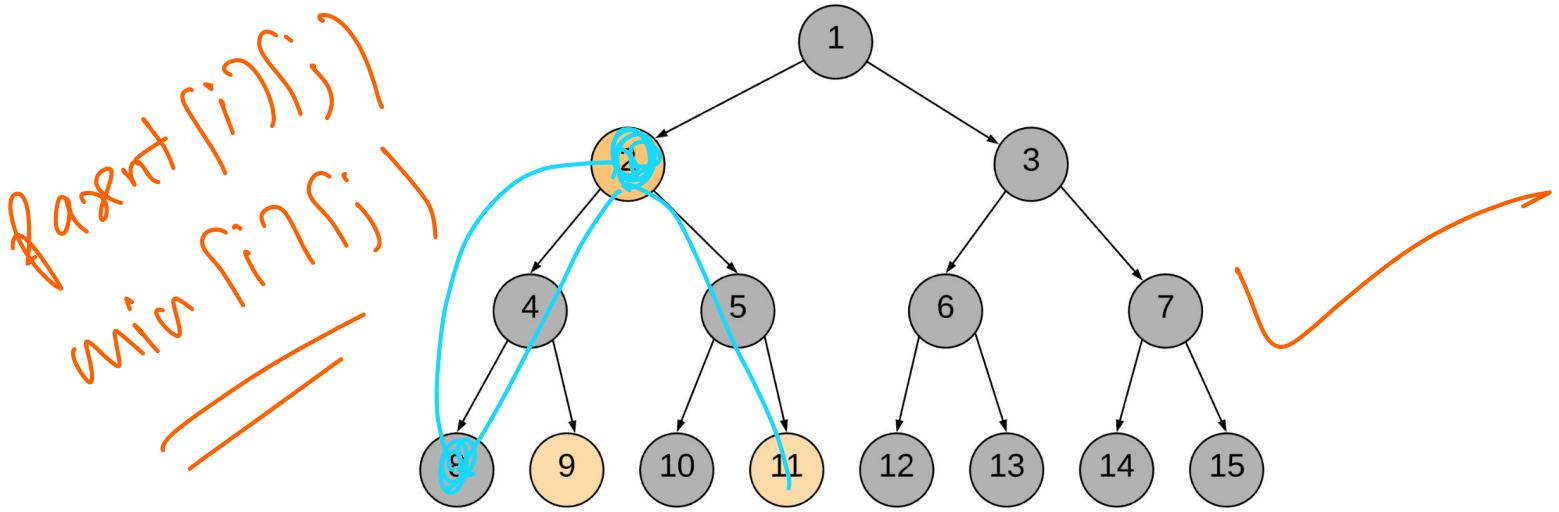
$n = [\text{level}[x] - \text{level}[y]]$ parent of x



$= \text{level}[x] + \text{level}[y] - 2 \cdot \text{level}[\text{lca}]$

Distance between any 2 nodes Link

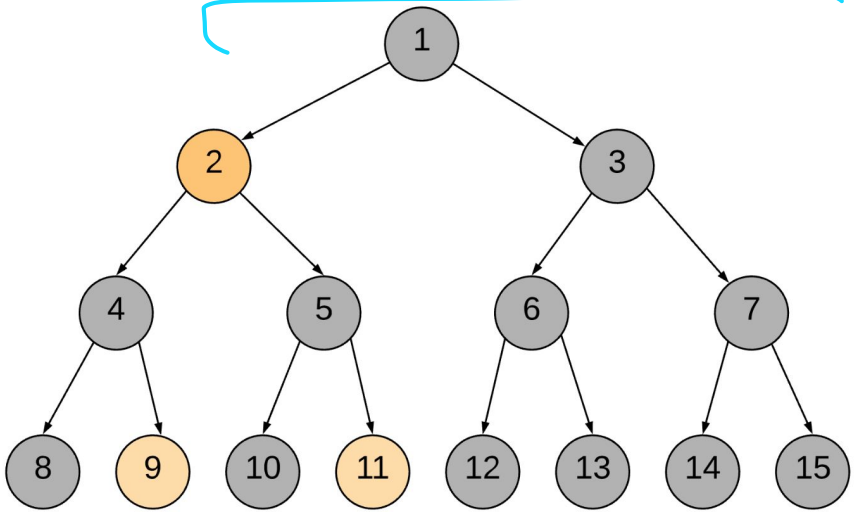
$$dist(A, B) = level_A + Level_B - 2 * Level_{LCA}$$



Lowest Common Ancestor for **Node 9** and **Node 11** is **Node 2**

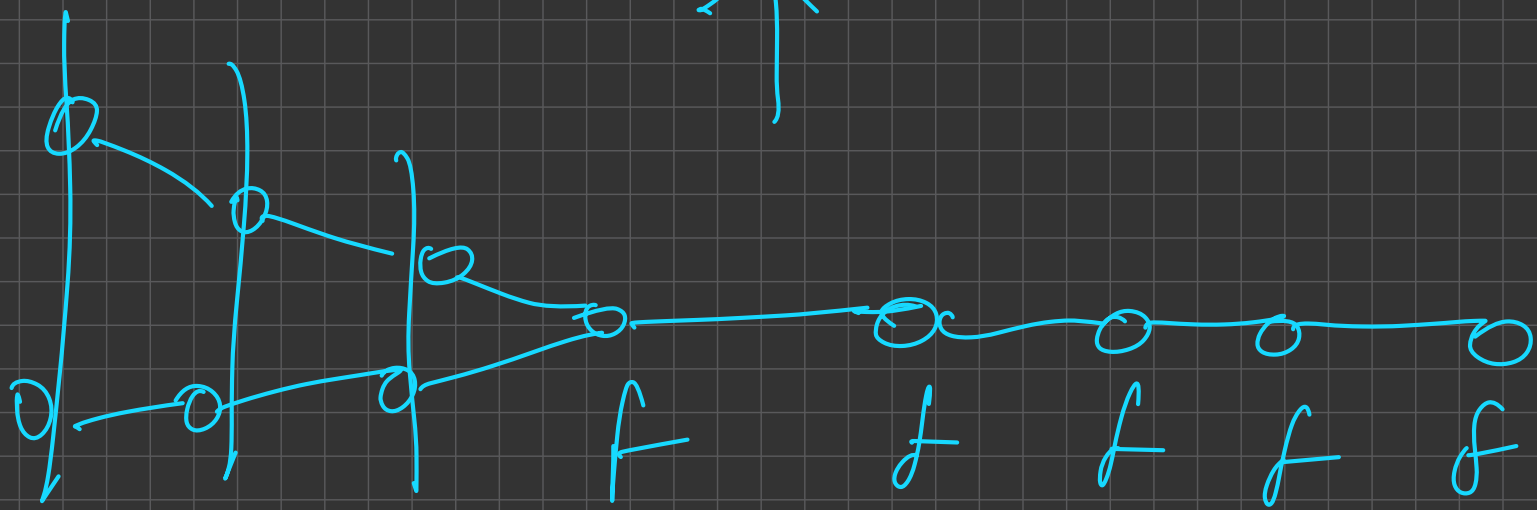
Find minimum value on Path

$$\min(A, B) = \min(\min(A, LCA), \min(B, LCA))$$



Lowest Common Ancestor for **Node 9** and **Node 11** is **Node 2**

T T T T T T T f f f f f f



f f f f f