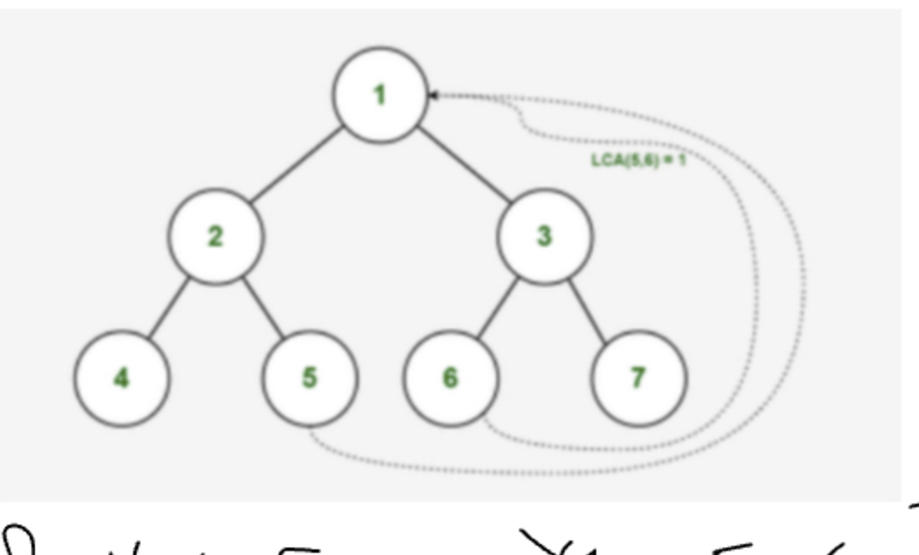
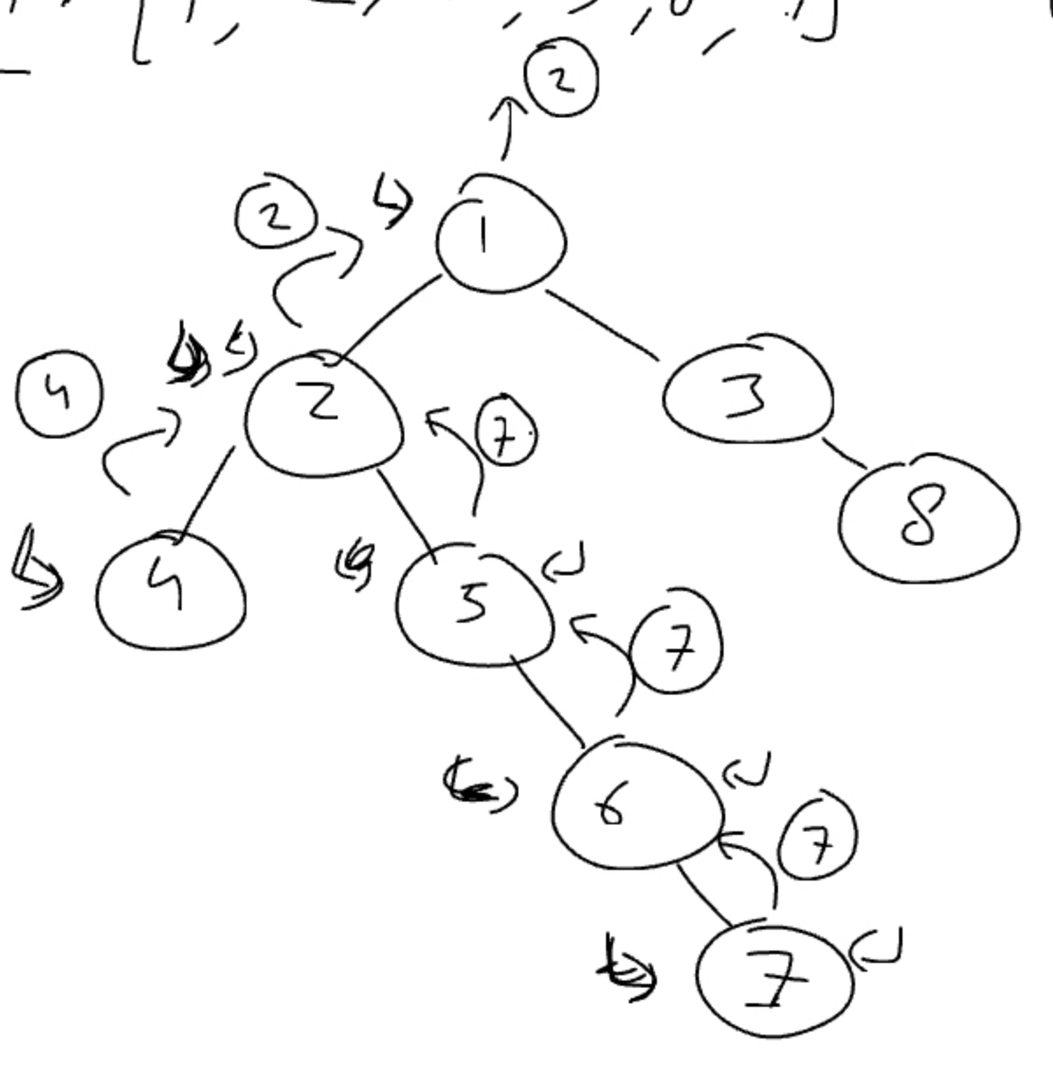


Anc. of 5 : [1, 2, 5]
Anc of 6 : [1, 3, 6]

[res = currPath]

currPath : [1, 2, ~~4~~, 5, 6, 7]

n1 = 4
n2 = 7



res = ~~x~~ 2

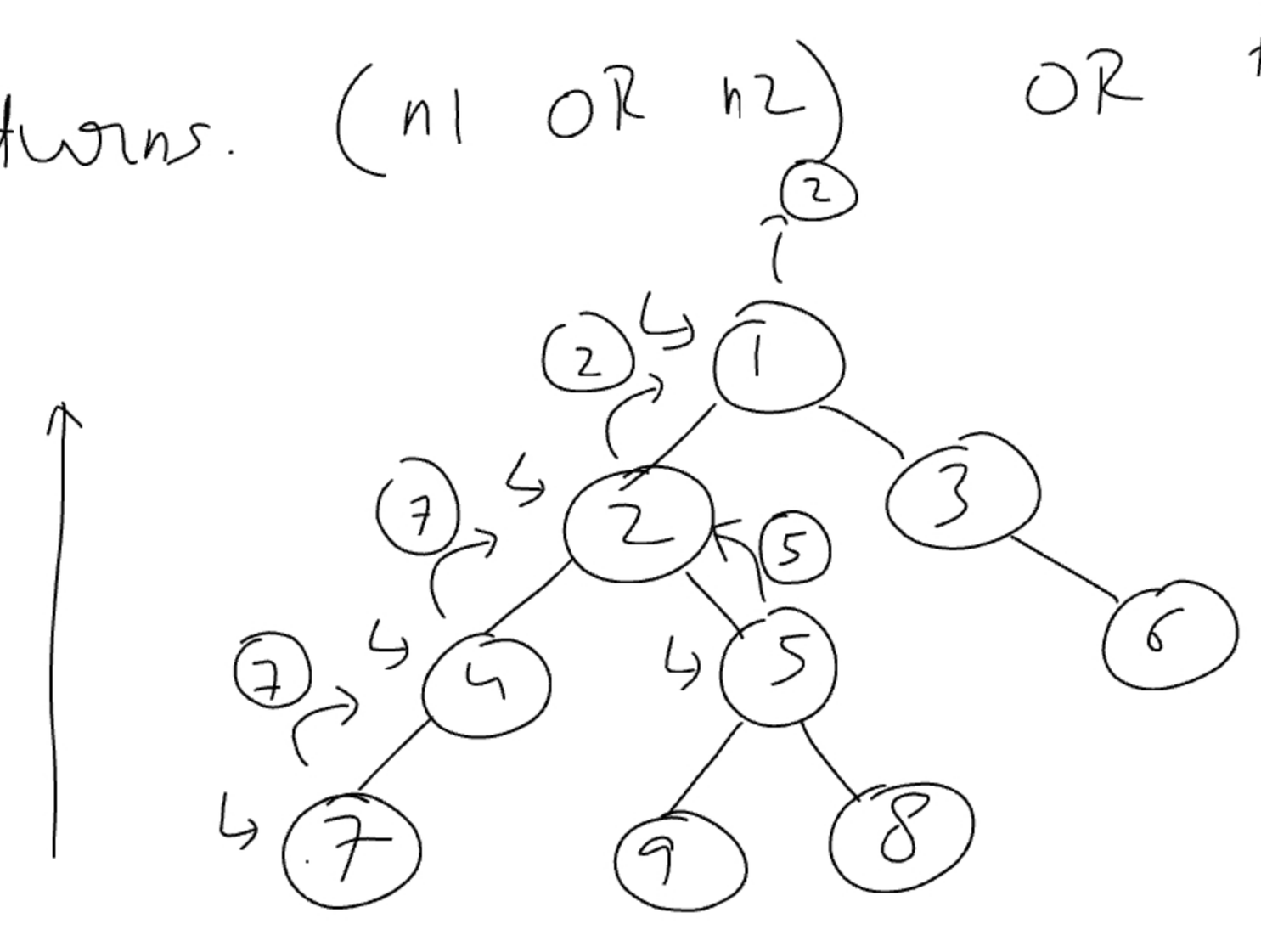
n1 = 4 n2 = 7
P1 = [1, 2, 4]
P2 = [1, 2, 5, 6, 7]

n1 = 2 n2 = 4
P1 = [1, 2]
P2 = [1, 2, 4]

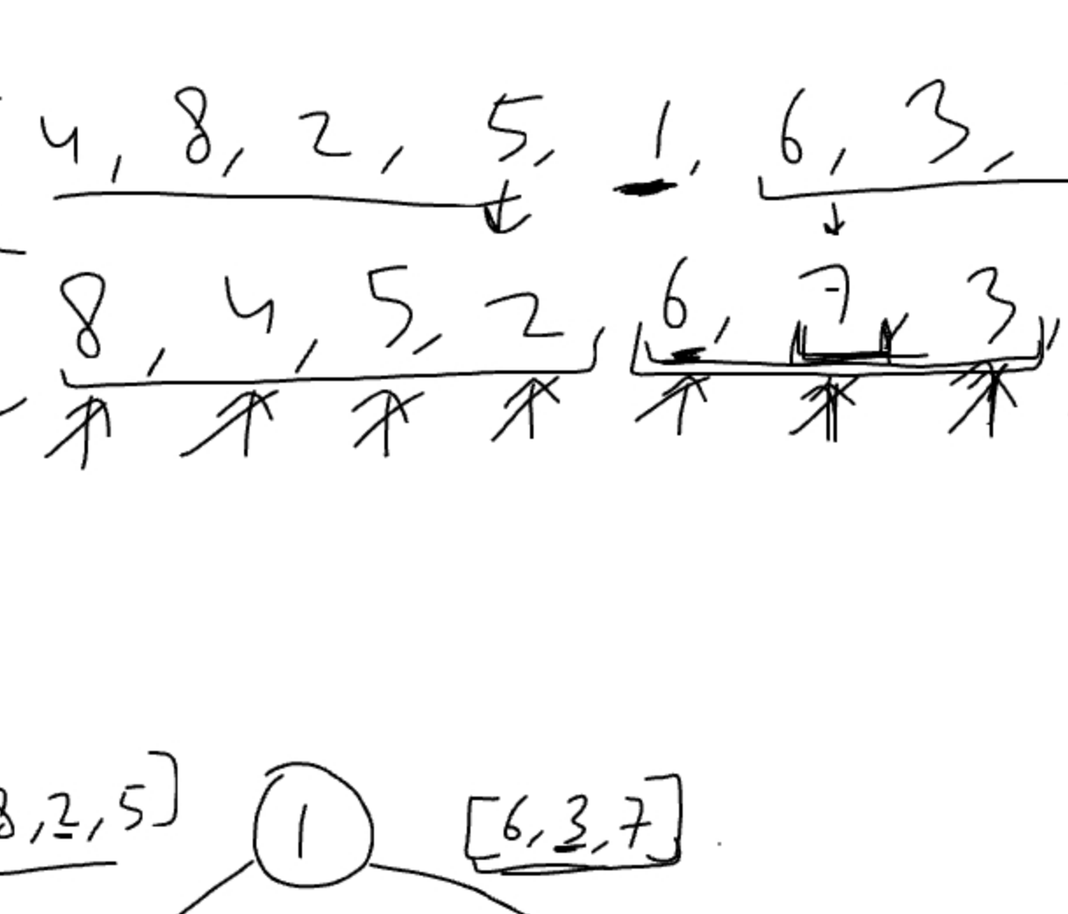
ans = ~~x~~ 2

func returns (n1 OR n2) OR the LCA if found.

n1 = 7 n2 = 5

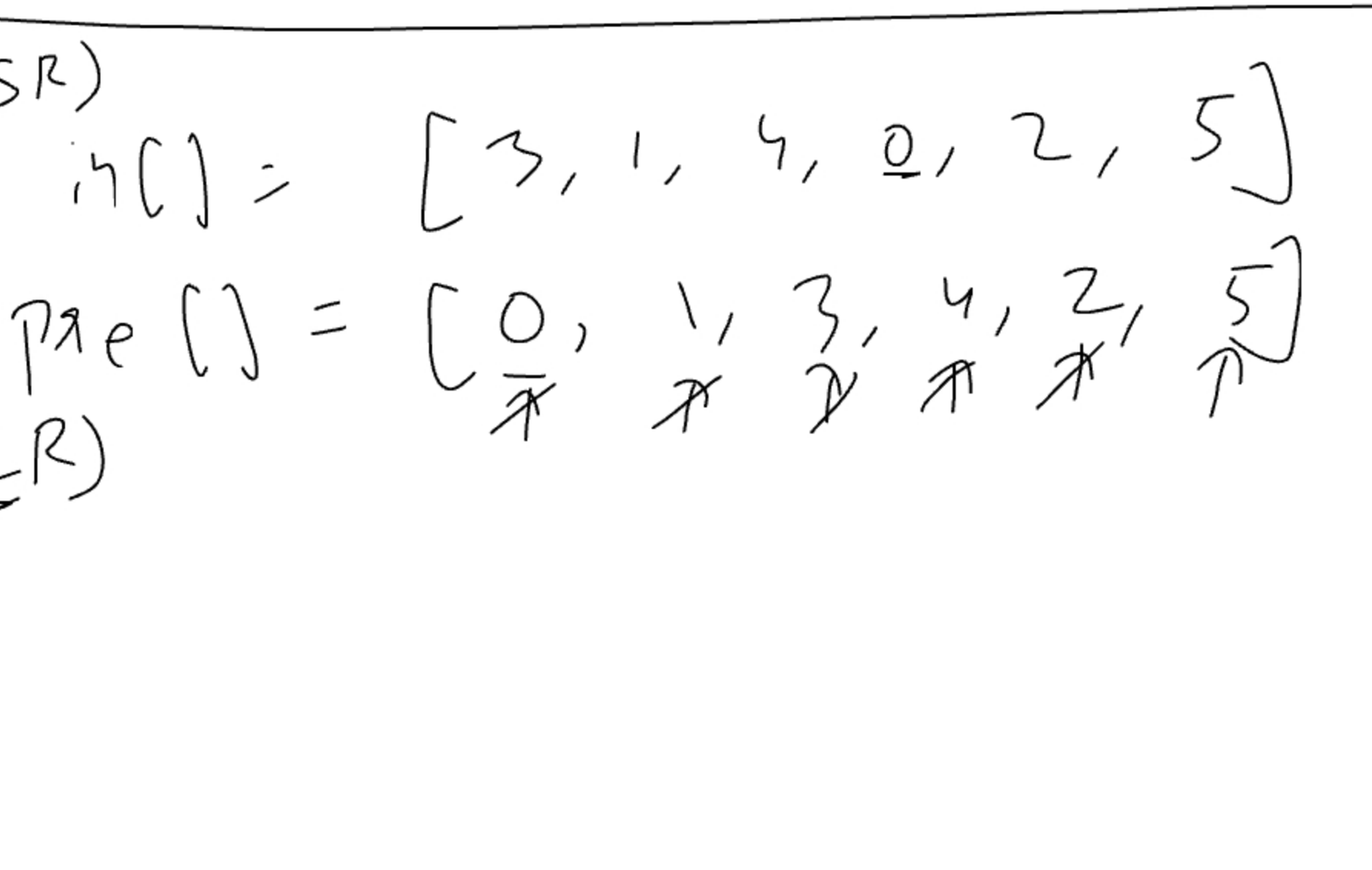


n1 = 2 n2 = 7.

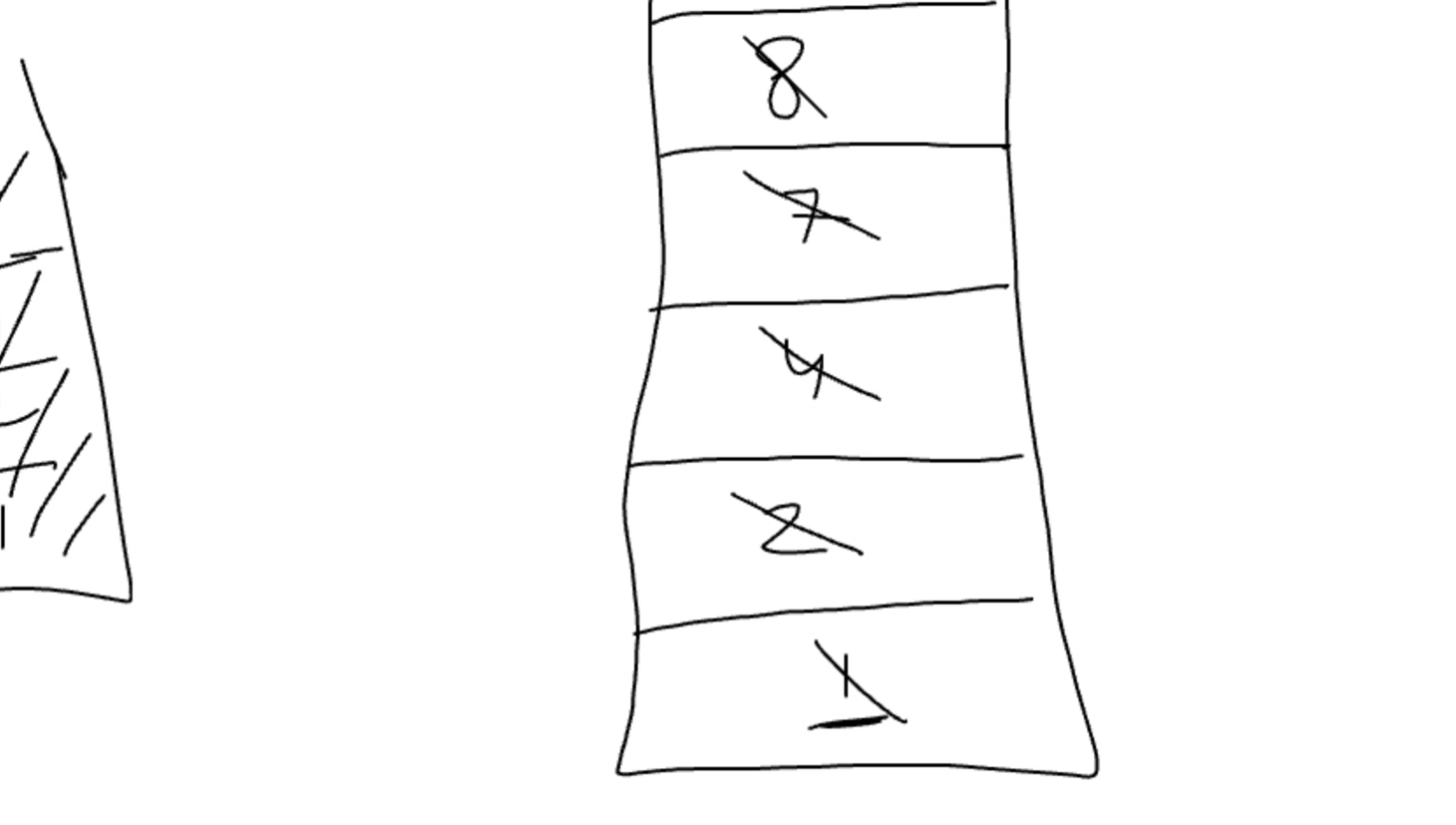


TC : O(n)
AS : O(H).

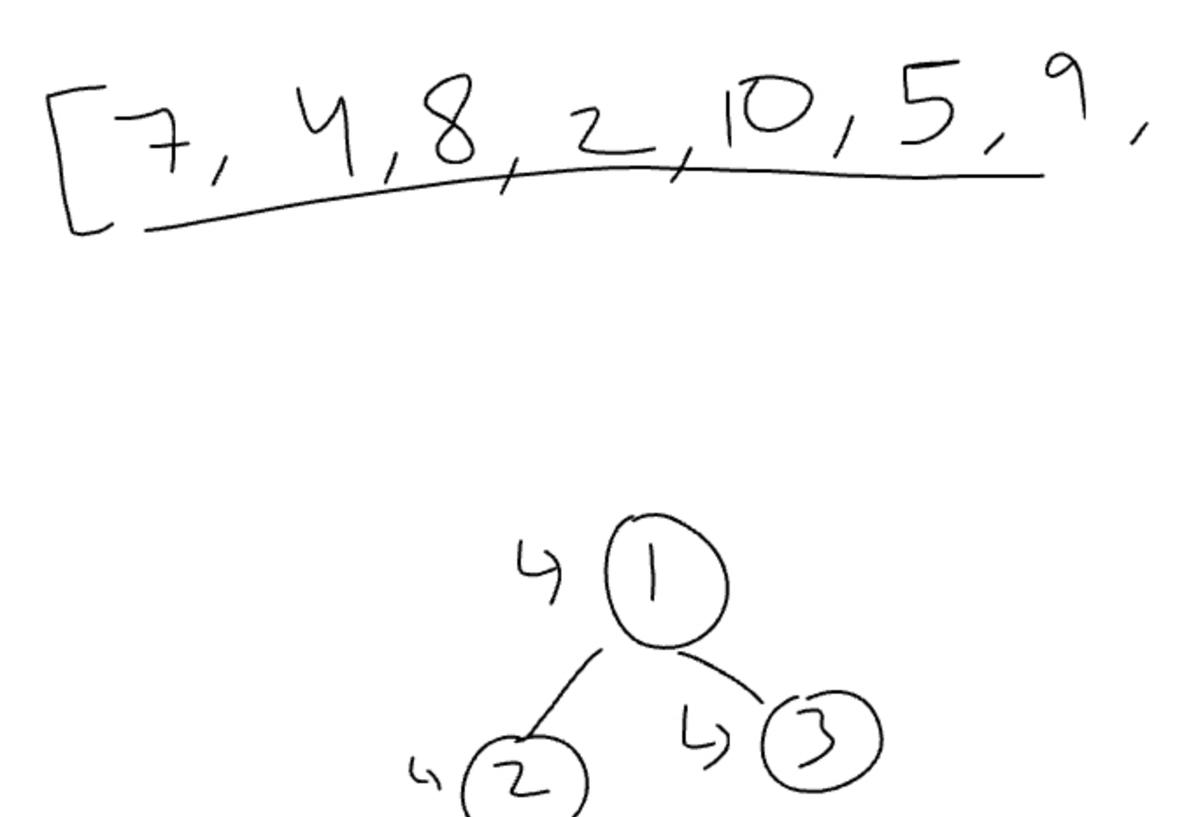
(LSR) inorder : [4, 8, 2, 5, 1, 6, 3, 7]
(LRS) postorder : [8, 4, 5, 2, 6, 7, 3, 1]



(LSR) in[] = [3, 1, 4, 0, 2, 5]
pre[] = [0, 1, 3, 4, 2, 5]
(SLR)



LSR.



[7, 4, 8, 2, 10, 5, 9, 1, 3, 6].

[7

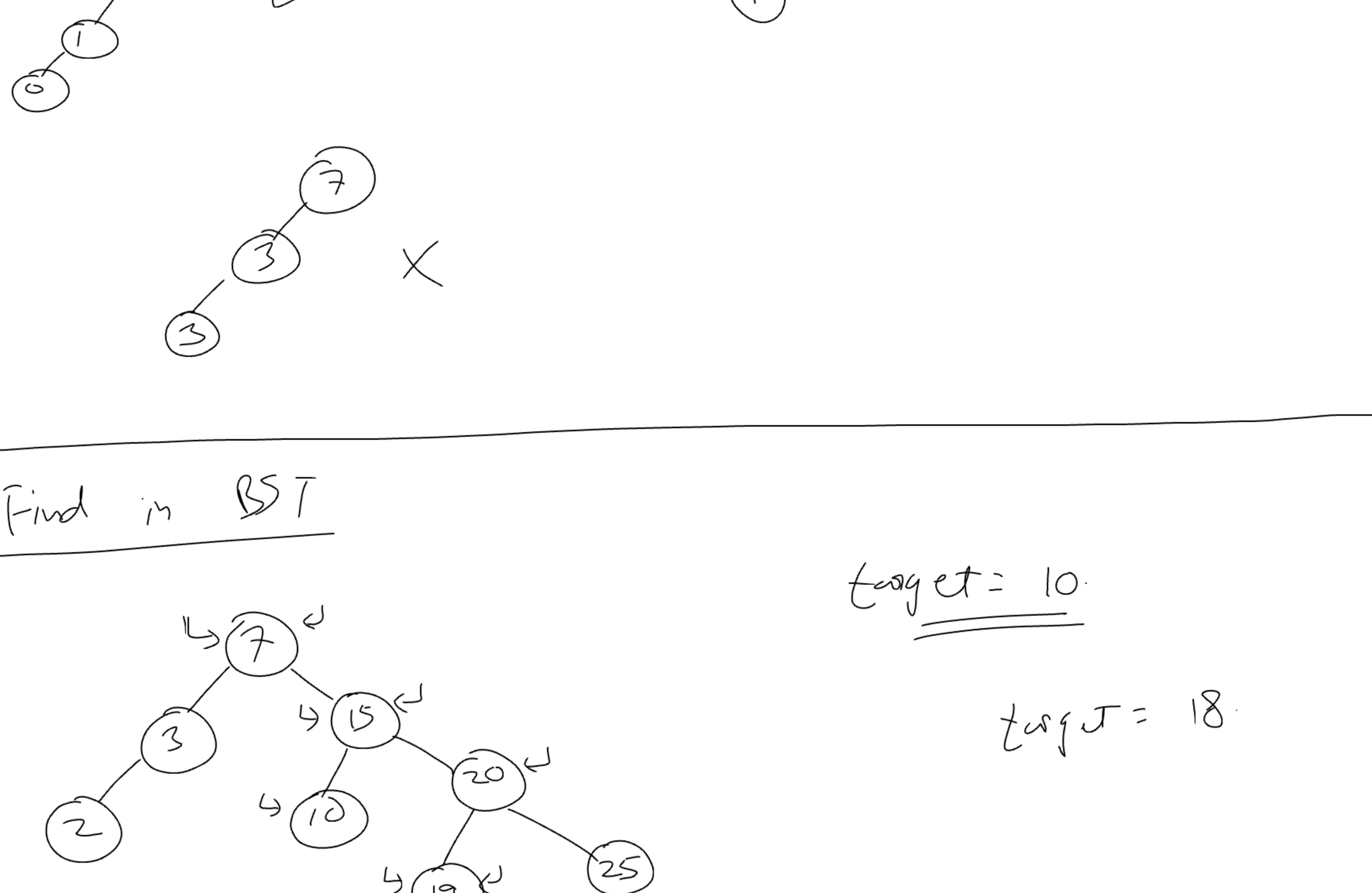
TC : O(n)
AS : O(h).

SLR

LRS ≠ SRL.

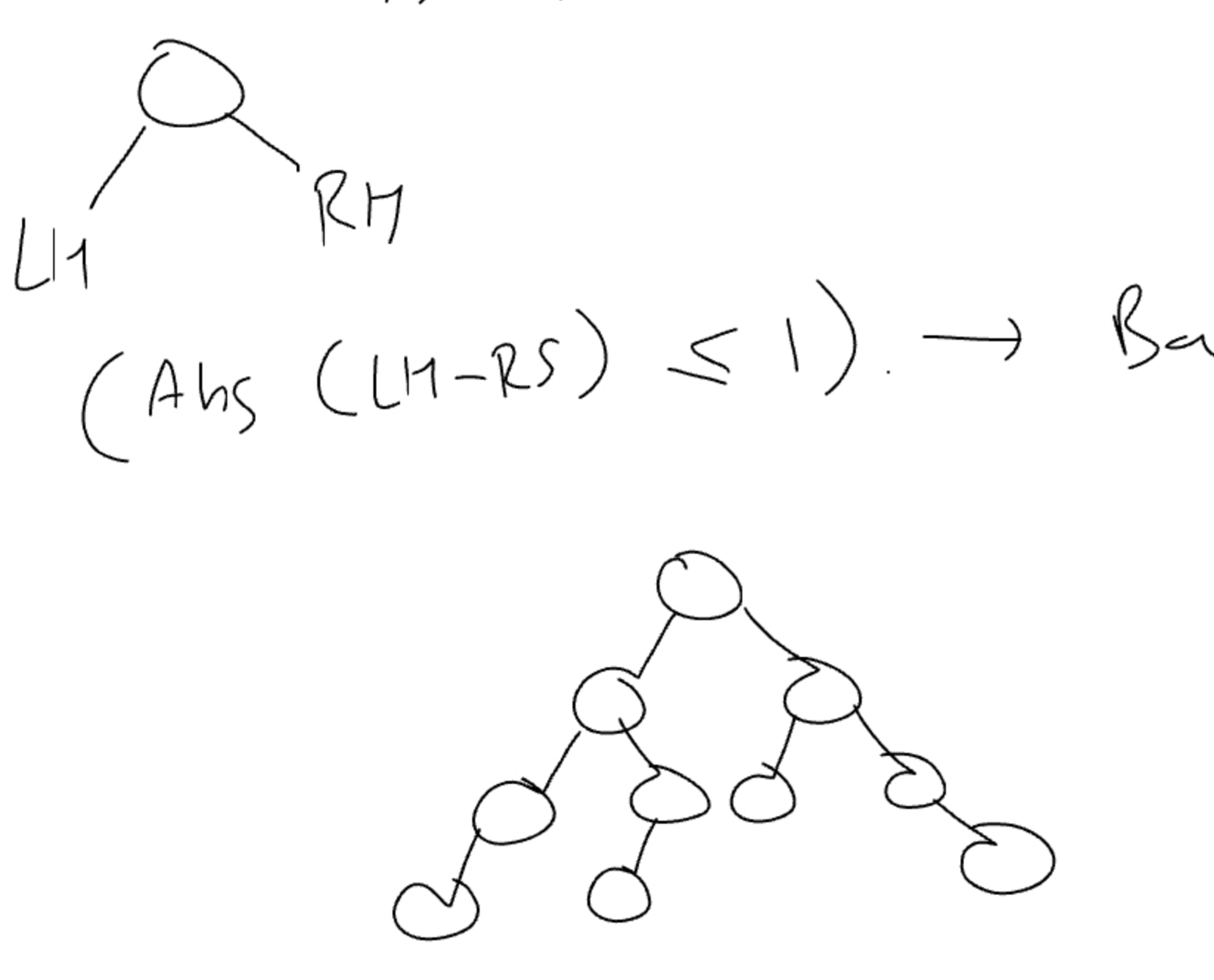
[4, 2, 7, 6, 5, 1, 3]

BST



Find in BST

RSL.



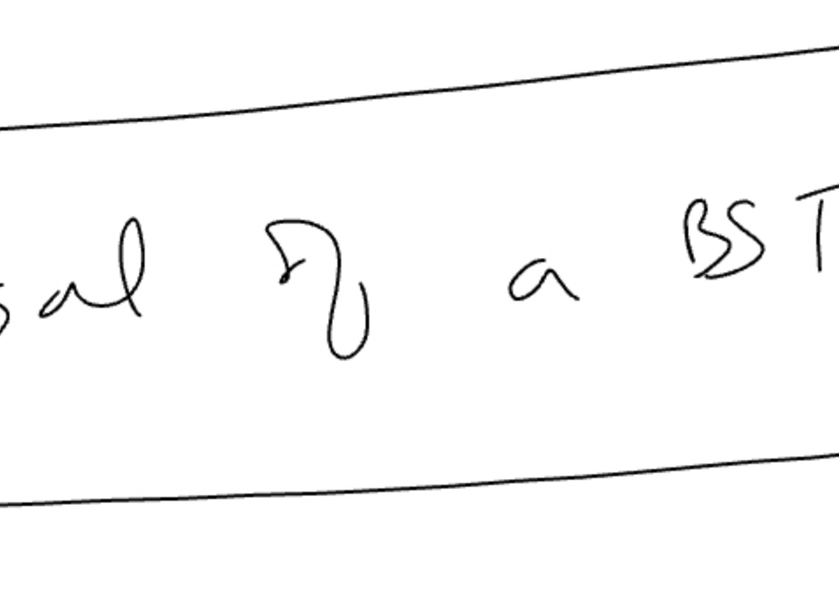
~~25, 20, 19, 15, 10, 7, 3~~, 2

target = 10

target = 18

TC : O(H)
AS : O(H) / O(1)
Rec Ito

(Abs (LH-RS) ≤ 1) → Balanced Binary Tree.



$H \approx \log_2(n)$

Ordered Hashmaps → Balanced BSTs.

inorder traversal of a BST is always sorted.

LSR

```
int func (int n, vector<int> arr){
    if (n ≤ 1) return n;
    ... do something ...
    return func(n-1, arr) + fun(n-2, arr);
}
```