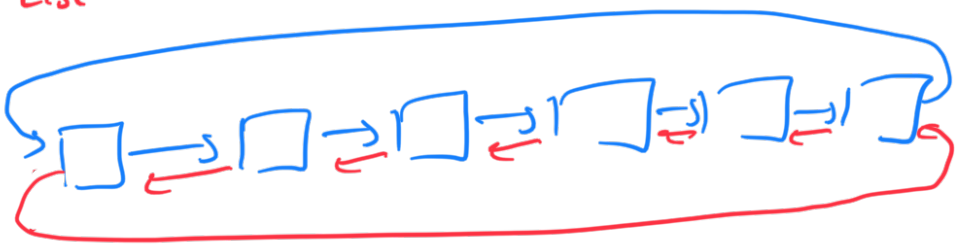
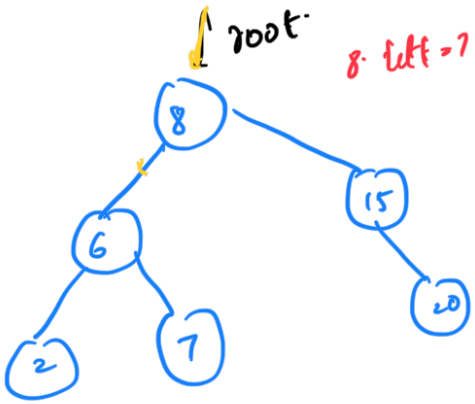
Trees – Followup

Circular Doubly Linked List.



fail.next = head
head.prev = tail

Question: Given a B.ST, convert this into a circular Doubly Linked List (Inorder List)

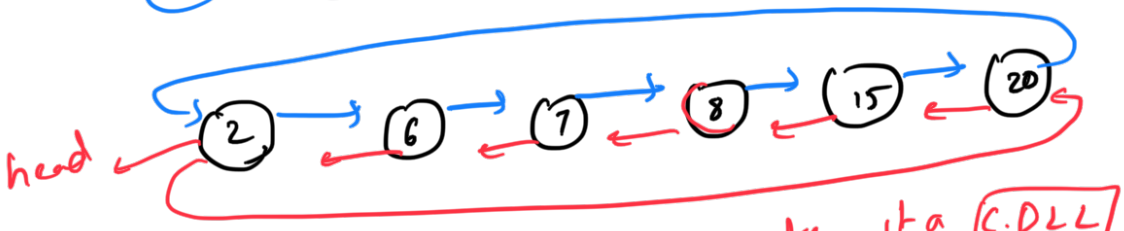


8.left = ?
8.right = 15

8.left = ??

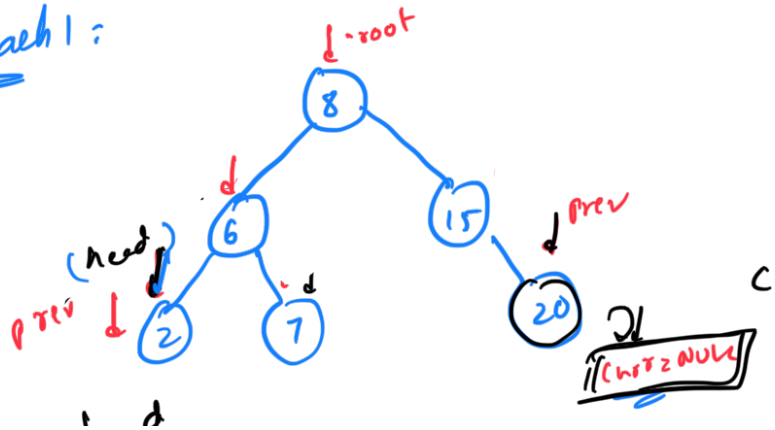
Left ⇒ prev
Right ⇒ next.

15.left = 8 [Prev]
15.right = 20 [Next]



⇒ Change the Pointers to make it a C.D.L.L

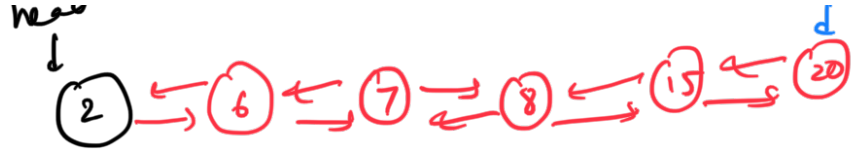
Approach 1:



head =
prev.

curr.left = prev
prev.right = curr

prev




```
head.left = prev;
prev.right = head;
```

```
head = NULL; prev = NULL;
inordu (root) {
if(root == NULL) return;
inordu (root.left);
if(head == NULL) {??
    head = root;
}
else {
    prev.right = root;
    root.left = prev;
}
prev = root;
inordu(root.right);
}
head.left = prev;
prev.right = head
```

→ only 1a

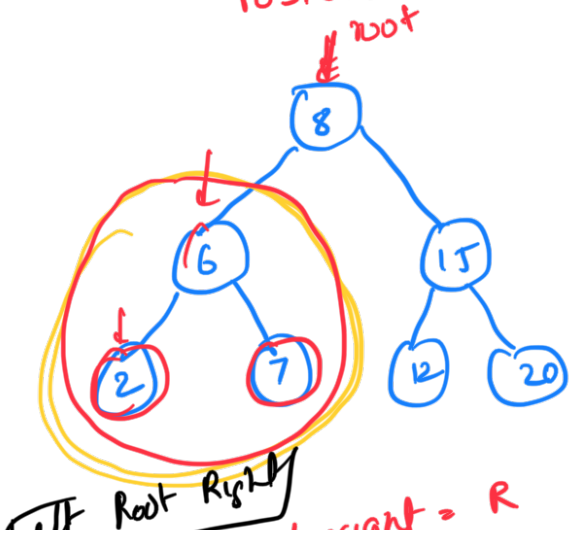
T.C: O(n)

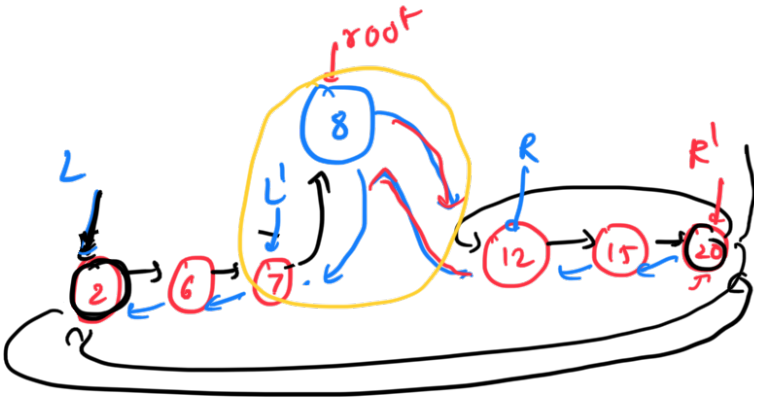
S.C: O(H) ← Recursion Stack

Approach2: → BSTtoDLL(root)

Postordu traversal

Left → Root → Right





7.right = 8

8.left = 7

root.right

(R).left = root

L' = L.left , R' = R.left

L'.right = root

root.left =

L' = L.left

R' = R.left

L'.right = root

root.left = L'

L.left = R'

R'.right = L

Inorder traversal

root

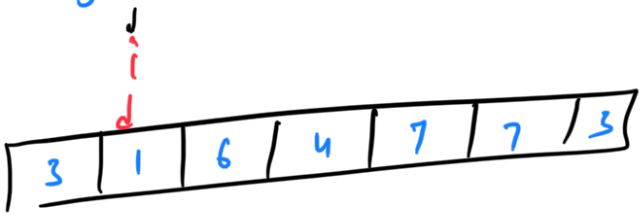
Edge Case

Problem

Question: write an iterator for inorder traversal B.T

1) hasNext() => if there is any node

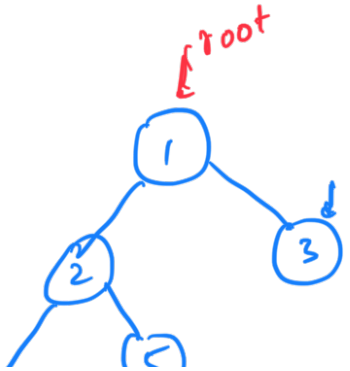
2) getNext() => Return next element of D.S




```
hasNext(){
    if(i<N)
        return true;
    else false
}
```

```
getNext(){
    return A[i];
    i++;
}
```

getNext(), hasNext()



get() : 6

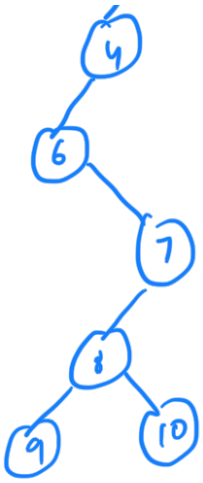
get() : 9

get() : 8

hasNext() : T

get() : 10

get() : 7



getNext(): 4

getNext(): 3

hasNext(): false.

Approach 1 :

Store the Inorder Traversal in array

T.C : O(n)

S.C : O(n)

Approach 2 :

→ Convert the tree to a Linked List

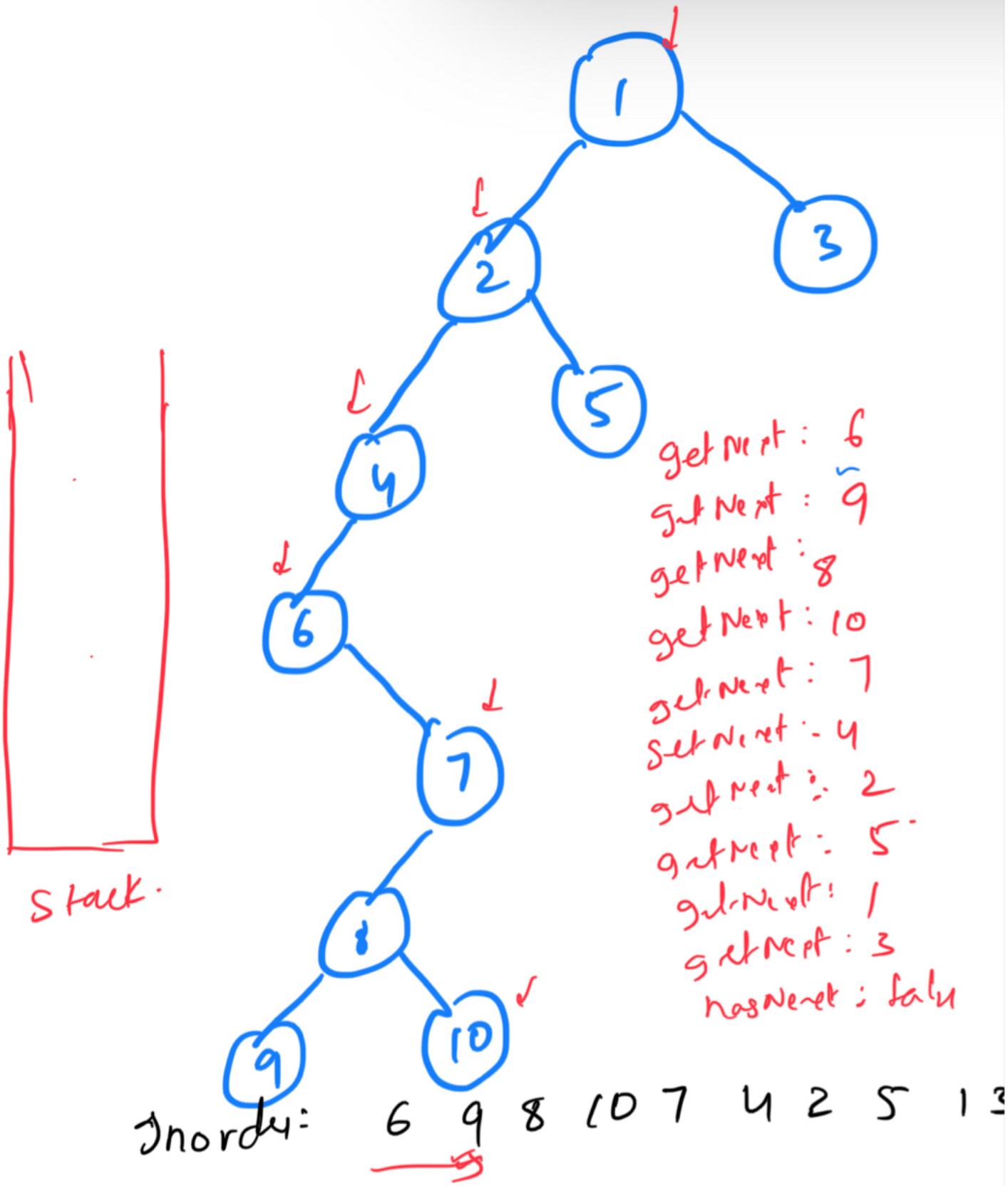
Losing structure of Tree

Approach 3 :

getNext() => Do 1 step in Inorder Traversal

↓ Recursion =>

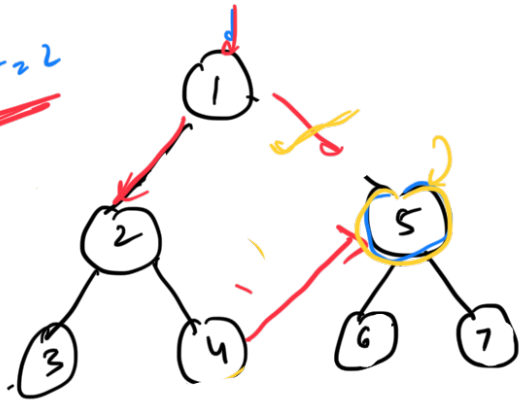
Iterative method for Inorder Traversal.


1
2
3
4
5
6
7
8
9
10
Stack.
getNext : 6
getNext : 9
getNext : 8
getNext : 10
getNext : 7
getNext : 4
getNext : 2
getNext : 5
getNext : 1
getNext : 3
hasNext : false
Inorder: 6 9 8 10 7 4 2 5 1 3

Question: Flatten a Binary Tree

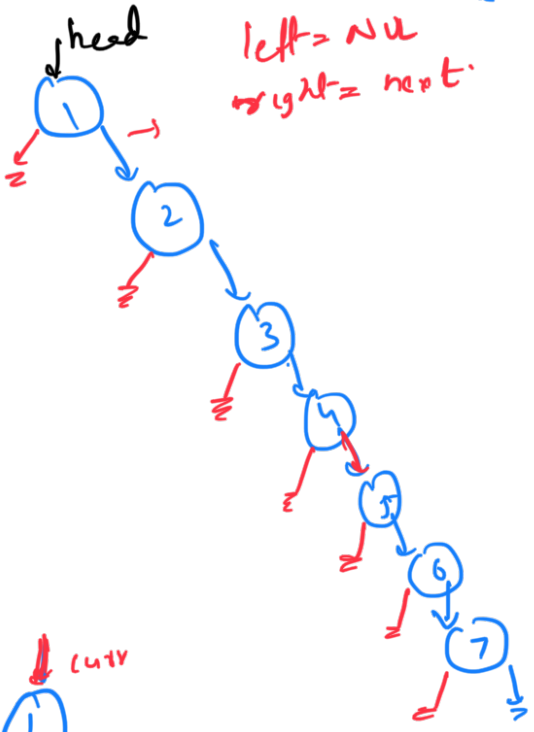
Leetcode

Root



1. right = 2

=>

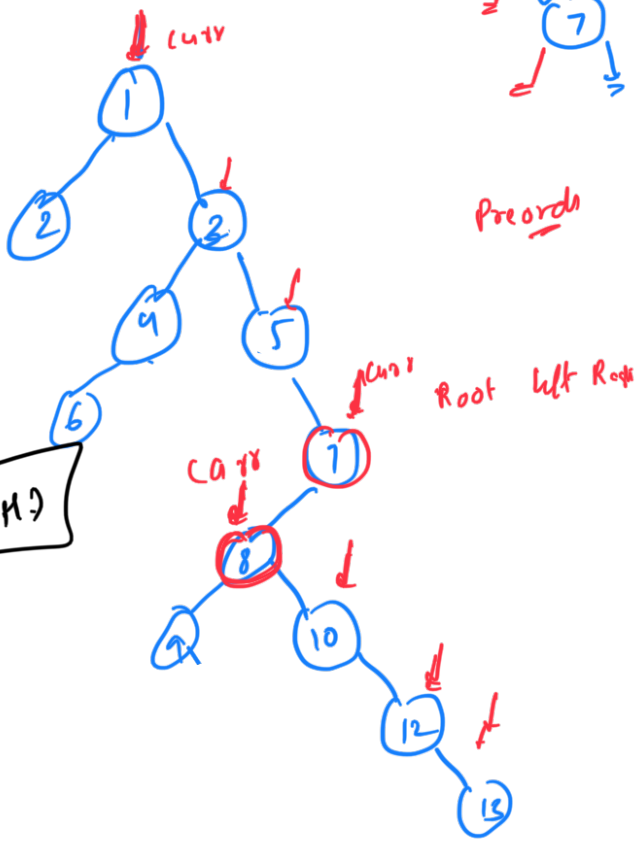


left = NULL
right = next.

1 2 3 4 5 6 7

Preorder:
{Root, Left, Right}

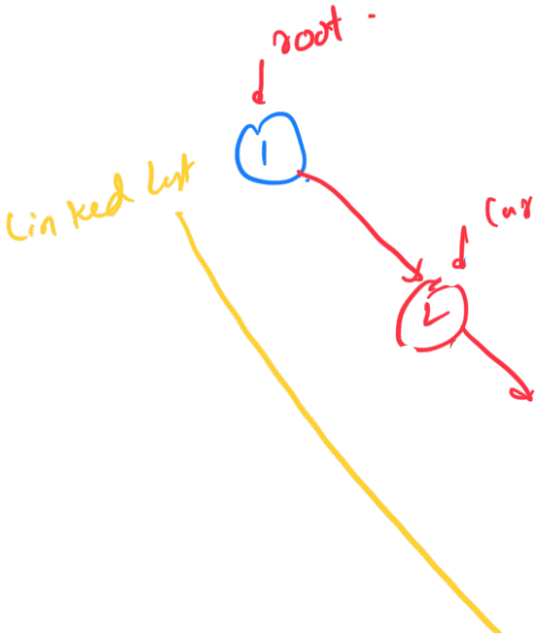
Pre: Root left Right



Preorder

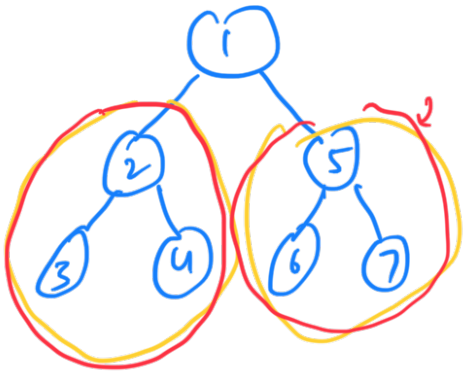
T.C: O(N)

Root left Right

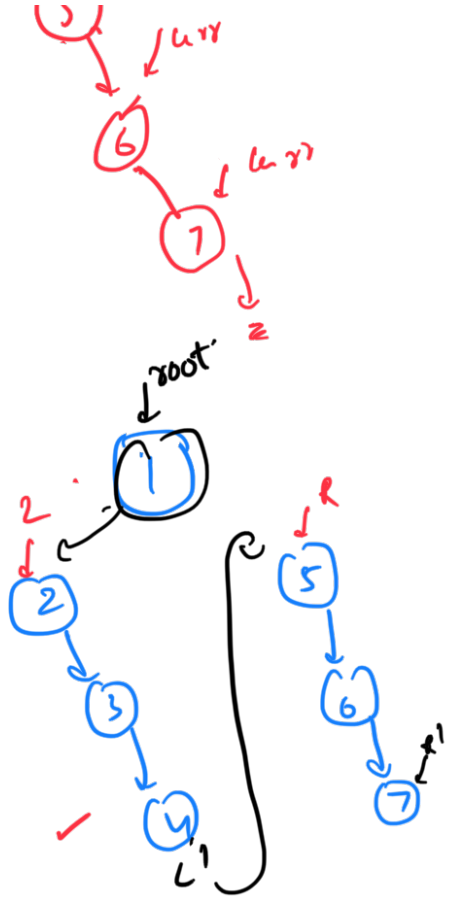


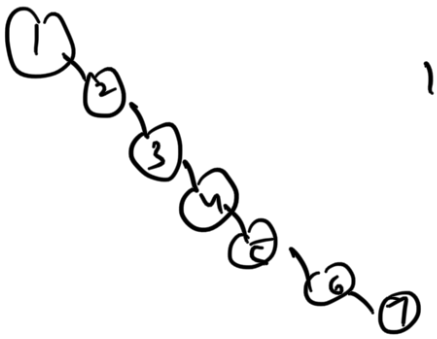
Linked List

Approach2 : Postorder



Postorder: Convert LST & RST into a Singly linked





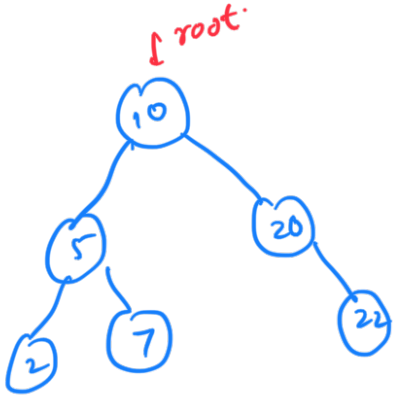
1. right = L
L'.right = R

head = Root
tail = R'.

Function will return head and tail

if (LST == NULL)
if (RST == NULL)

Question: 2 Sum B.S.T



K = 27

O(nlogn) 2 pointer

O(n)

2-pointer

Approach:

→ Inorder Traversal : Two pointer Approach : O(n)

T.C: O(n) / S.C: O(n)

Approach2:

Hashset:

T.C: O(n) S.C: O(n)

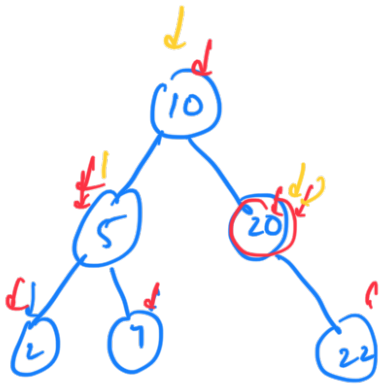
Inorder / Preorder / Postorder

& (k-n).

k = 27

5, 22

Approach3:



T.C: O(H)

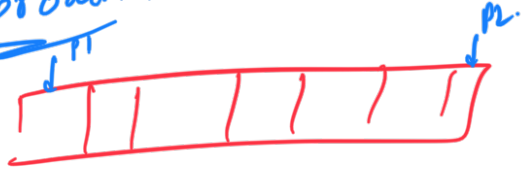
↓

Search

T.C: O(N x h)

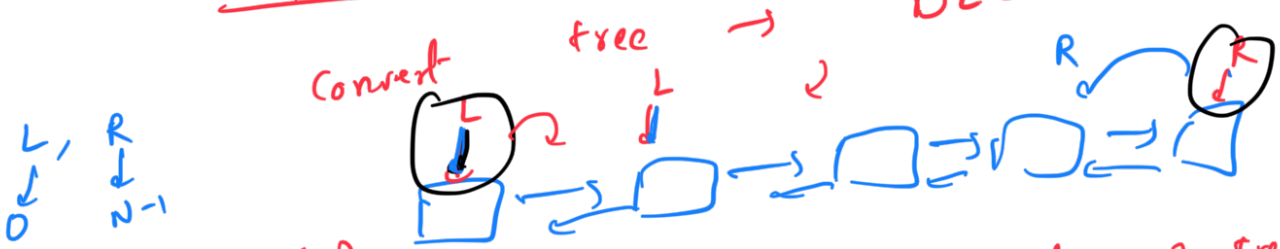
S.C: O(H) → [Recursion stack]

Approach4:



Sorted

DLL



T.C: O(n)

S.C: O(H)

Loosing the structure of tree.

Approach:

Inorder: Left Root Right

Reverse(Inorder): Right Root left.

Iterator

getNext()

ITERATOR

S.C : O(H)

Question: Check if a given Preorder, Postorder, Inorder are of the same tree.

Pre : 1 2 4 5 3

Ino : 4 2 5 1 3

Post : 4 5 2 3 1

Approach1:

→ Construct tree using Inorder and Preorder → [Tree]

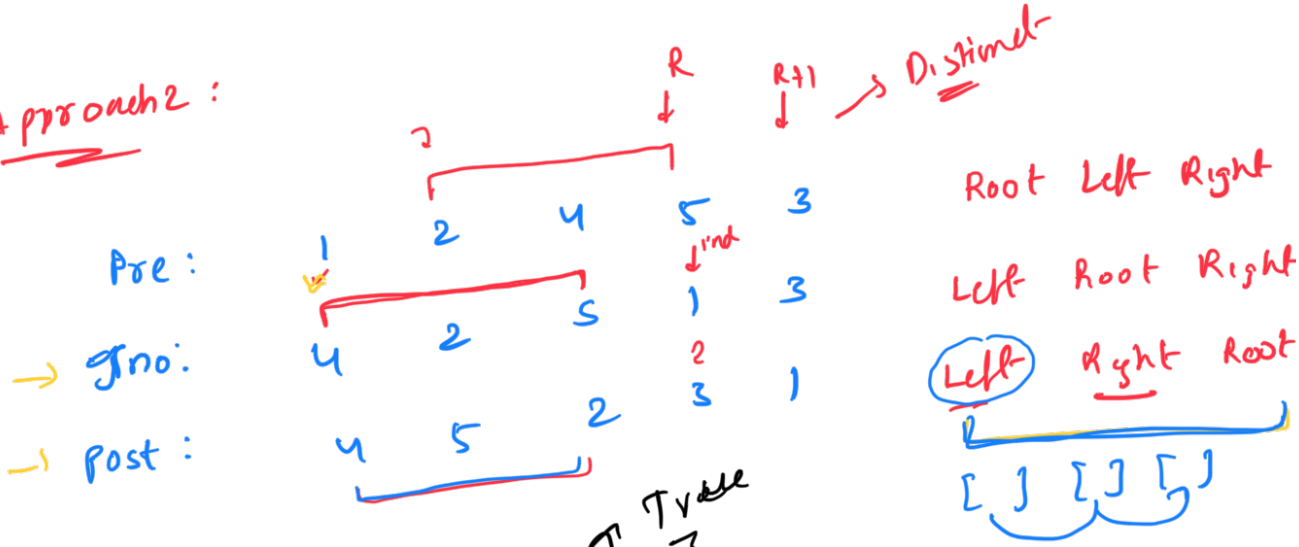
→ Find Postorder → O(N)

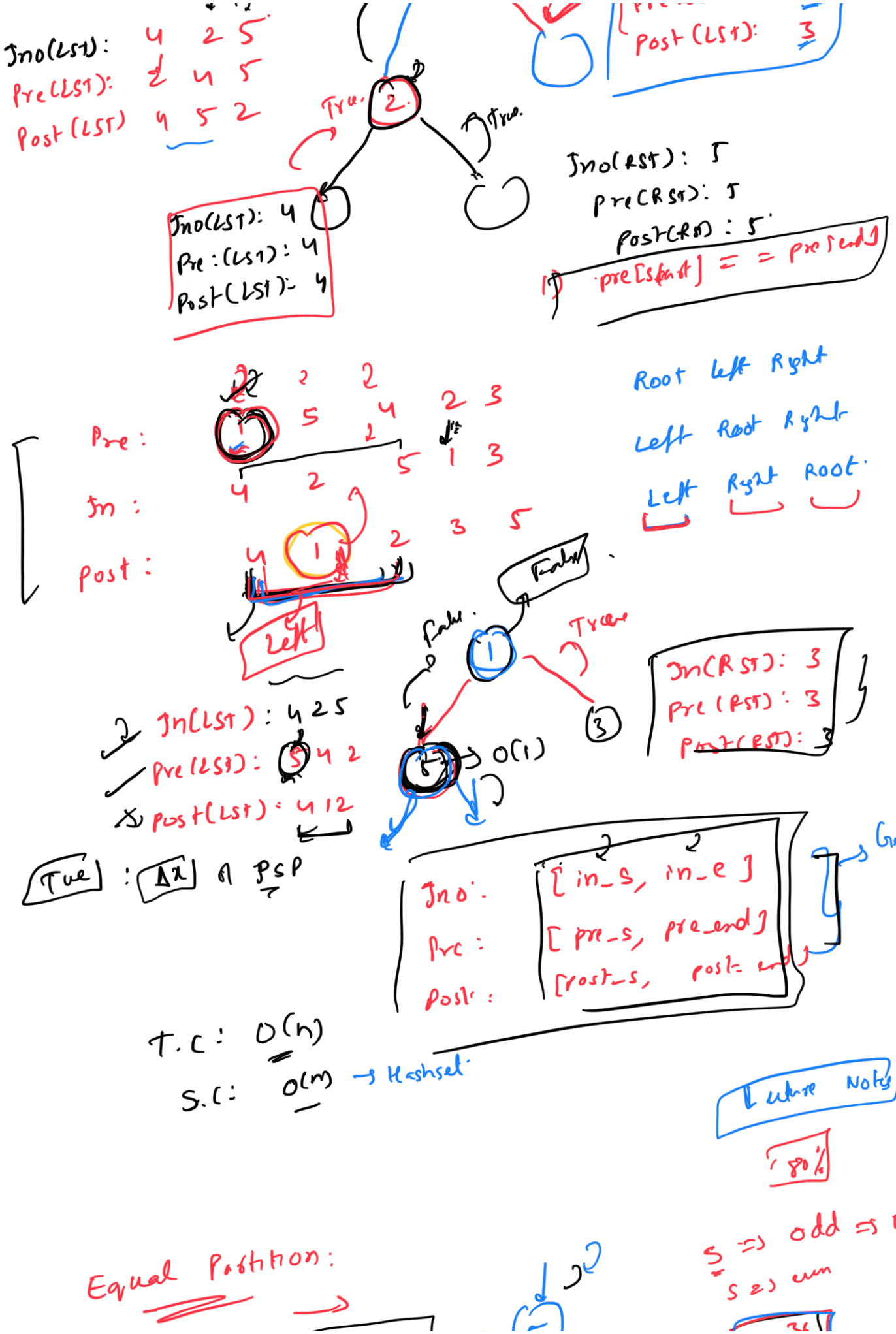
O(N) → [Hashing]

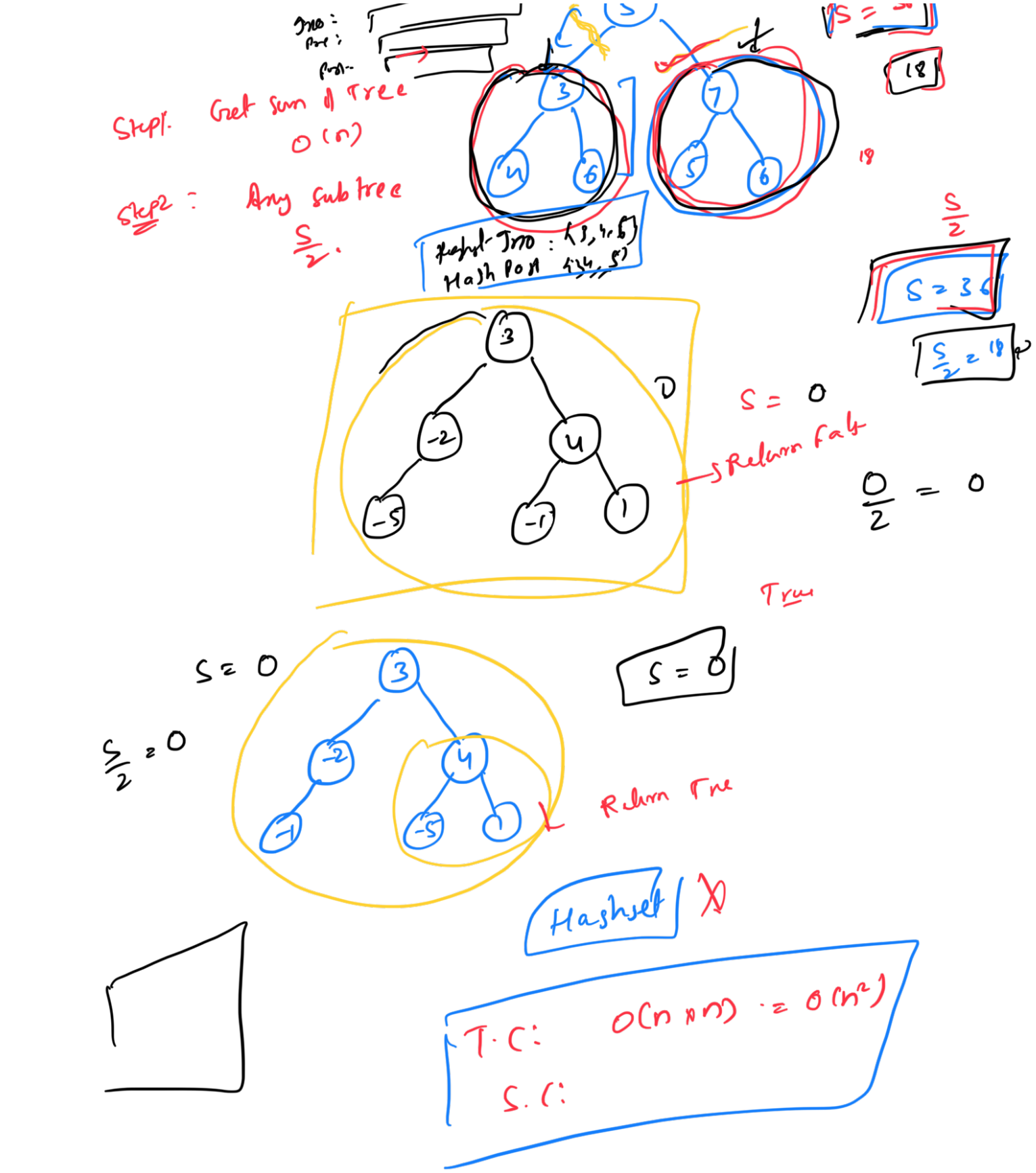
T.C : O(n)

S.C : O(n)

Approach2:




In o(LST): 4 2 5
Pre(LST): 2 4 5
Post (LST) 4 5 2
Post (LST): 3
True
True
In o(LST): 4
Pre :(LST): 4
Post(LST): 4
In o(RST): 5
pre(RST): 5
Post(RST): 5
pre[start] == pre[end]
Pre :
In :
Post :
Root Left Right
Left Root Right
Left Right Root
Left
False
False
True
In(LST): 4 2 5
Pre (LST): 5 4 2
Post(LST): 4 1 2
In(RST): 3
Pre (RST): 3
Post(RST): 3
O(1)
In o:
Pre :
Post :
[in_s, in_e]
[pre_s, pre_end]
[post_s, post_end]
T.C: O(n)
S.C: O(n) → Hashset
Future Notes
80%
Equal Partition:
S => odd => NO
S => even


Step1: Get sum of Tree
O(n)
Step2: Any subtree
S/2.
18
S/2
S = 36
S/2 = 18
S = 0
Return false
0/2 = 0
True
S = 0
S = 0
S/2 = 0
Return True
HashSet
T.C: O(n x n) = O(n²)
S.C:

Check if inorder, preorder and postorder are of the same array

```
bool check(vector<int> &pre, int pre_s, int pre_e, vector<int> &ino,
int ino_s, int ino_e, vector<int> &pos, int pos_s, int pos_e){
    if(pre_s > pre_e) return true;
    if(pre_s == pre_e){
        return (pre[pre_s] == ino[ino_s] && ino[ino_s] ==
pos[pos_s]);
    }

    int ino_ind = -1;
    for(int i = ino_s; i <= ino_e; i++)
        if(ino[i] == pre[pre_s]){
            ino_ind = i;
            break;
        }
    int pos_ind = -1;
    for(int i = pos_s; i <= pos_e; i++)
        if(pos[i] == pre[pre_s]){
            pos_ind = i;
            break;
        }
    if(ino_ind == -1 || pos_ind == -1) return false;

    int len = ino_ind - ino_s;
    bool isLeft  = check(pre, pre_s+1, pre_s + len, ino, ino_s,
ino_ind - 1, pos, pos_s, pos_s + len - 1);
    bool isRight = check(pre, pre_s+len+1, pre_e, ino, ino_ind + 1,
ino_e, pos, pos_s + len, pos_e);
    return isLeft && isRight;
}
```