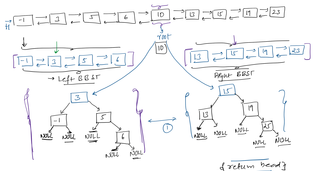
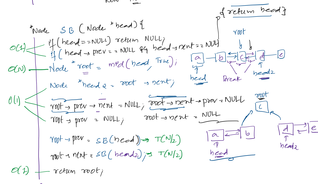
1) Given a SDLL convert into a BBST

Sorted Doubly linked list

If a BST is balanced, it is BBST.

{Are Balanced? — B-height difference between left & right subtree for every node ≤ 1}





```
*Node SB (Node *head) {
    if (head == NULL) return NULL;
    if (head → prev == NULL && head → next == NULL)
        return head;
    Node *root = mid(head, tail);
    Node *head2 = root → next;
    root → prev → next = NULL;
    root → next → prev = NULL;
    root → prev = SB(head) → T(N/2)
    root → next = SB(head2) → T(N/2)
    return root;
}
```

O(1) O(n) O(1) O(1) O(1)

Note: 1) root → prev can be NULL, root → next can be NULL } Define using this here & of condition

2) root → next can be NULL

Ex: When of there is a single [4]

Note: If single Node, we need to directly return that, if not it will go to an loop.

1) Recursive Relation:

$T(N) = 2T(N/2) + N$ $T(1) = 1$ → Merge Sort Recursive Relation

$T(N) = N \log N$ SC: $O(\log N)$

else 2) If we have sorted array getting mid → O(1)

If say we are using this array, do after

Recursive Relation: $T(N) = 2T(N/2) + 1$

$T(N) = O(N)$

Ex:

```
Given SDLL, sort all order in array
Node* BBST (vector<Node*> &data, int l, int r) {
    if (l > r) return NULL;
    int mid = (l+r)/2;
    Node *root = data[mid];
    root->left = BBST(data, l, mid-1);
    root->right = BBST(data, mid+1, r);
    return root;
}
```

l = r = mid

(l, mid-1) → return NULL

(mid+1, r) → return NULL

TC: $O(N + N)$ Inorder

SC: $O(N)$

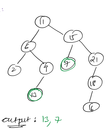
2a) Given a sorted array, construct BBST ✓

Given BST, since 2 nodes are swapped, find 2 nodes

Ex:

output: 24 & 13

Ex:

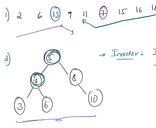


output: 13, 7

Simple: Inorder:

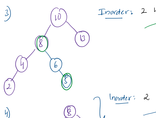
1) 2 6 13 7 11 15 16 18 21

2)



→ Inorder: 3 6 8 10

3)



Inorder: 2 4 6 10 13

4)

Inorder: 2 4 5 6 10 13

→ ans = INT_MAX

Note: While applying recursive traversal, we can get 1st peak ✓ check = true, } keep condition true } parent, root, root->data with prev, ascend

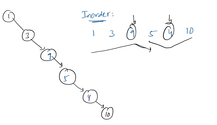
if (check == true) {
    ans = min(ans, root->data)
}

We can keep multiple together

↳ SC: $O(H)$ →

→ (Morris traversal)

5)

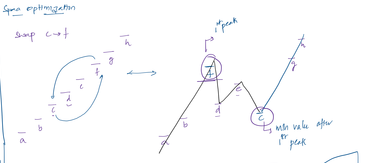


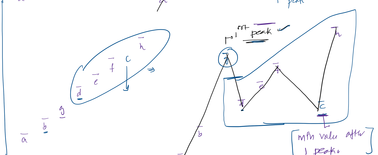
Inorder:

1 3 5 6 10

Space optimization

Swap c & f





Ex:

5 7 9 11 13 15 19 25 29 30

5 7 9 19 13 15 11 25 29 30

overall min: 11 → (19, 11)

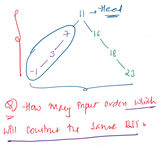
5 7 9 11 13 15 19 25 29 30

15 7 9 11 5 15 19 25 29 30

11 5

Given BST, How many different input nodes can get the same BST

Last: 11 7 3 -1 16 18 23



Input:

11 7 16 3 18 -1 23 ✓

11 16 7 3 -1 18 23 ✓

11 13 7 18 23 3 -1 ✓

11 16 18 7 23 3 -1 ✓

11 7 16 18 3 -1 23 ✓

11 7 3 16 -1 18 23 ✓

11 7 16 18 3 23 -1 ✓

3 7 -1 16 18 11 23 ✗

Obs: Relative order of left & right of input nodes are matched

Q) How many proper orders which will construct the same BST?

7 3 -1

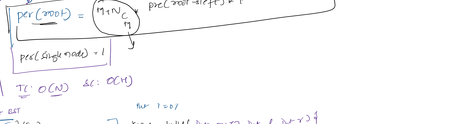
6 Empty Blocks → $^6C_3$

→ $^7C_3$ or $^7C_4$

In general:

→ $^{(M+N)}C_N$

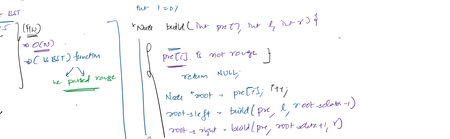
m = 3, N = 5 Total = $^8C_3$

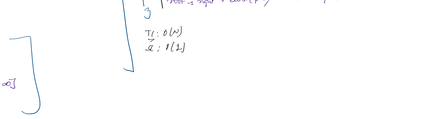


→ Total ways → $^8C_3 \times {}^4C_2$

m = 5, N = 5, → $^{10}C_5$

→ Total = $^{10}C_5 \times {}^4C_2 \times {}^4C_2$





$$\boxed{pre(root) = {}^{(M+N)}C_M \times pre(root \to left) \times pre(root \to right)}$$

$$\boxed{pre(single\ node) = 1}$$

TC: $O(N)$ SC: $O(H)$

6)

→ global value

pre: 5 3 2 4 8 15 13 19 25

root, left, right

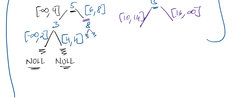
← O(N)

← O(N) ✓

→ optimization ✓ (in your own)

if root → left (-∞, 8)

→ (8, ∞)



(Hint)

← O(N)

← O(N log N) function

we passed range

```
*Node build(int pre[], int l, int r) {
    if pre[l] is not range {
        return NULL;
    }
    Node *root = pre[l]; l++;
    root->left = build(pre, l, root->data-1)
    root->right = build(pre, root->data+1, r)
}
```

for i => 0

TC: $O(N)$

SC: $O(1)$