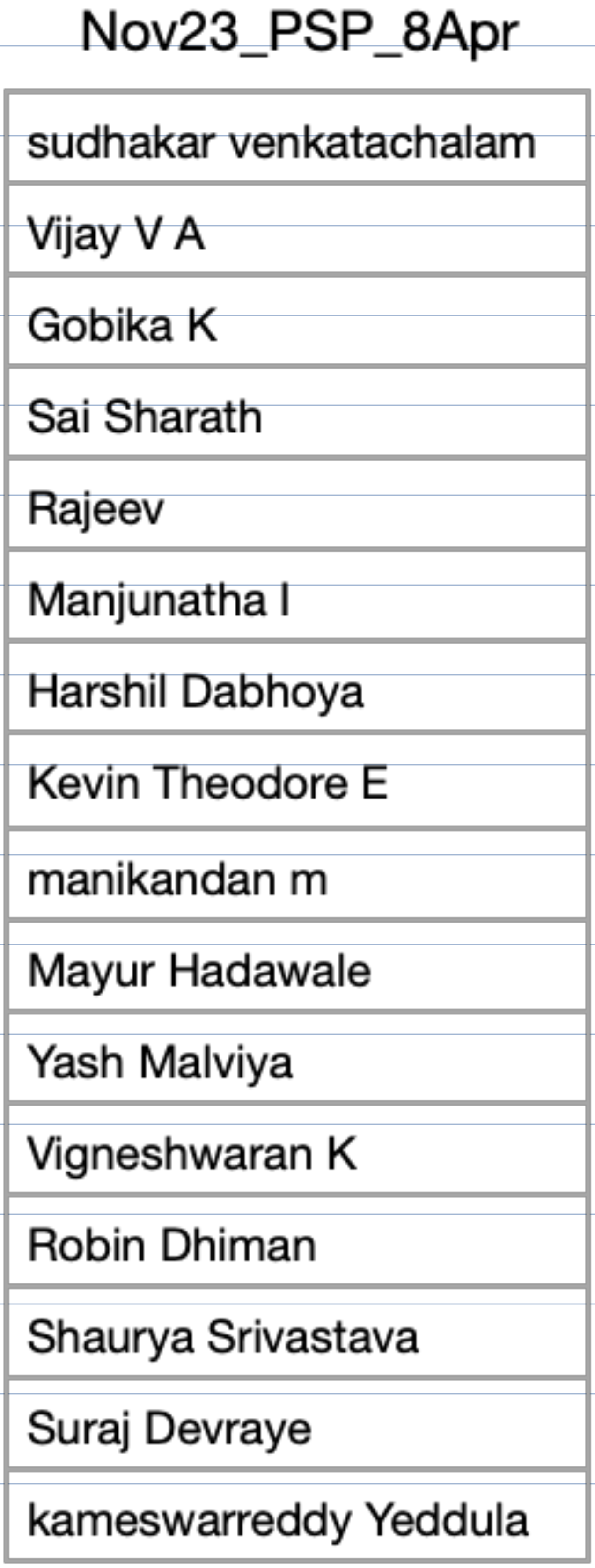
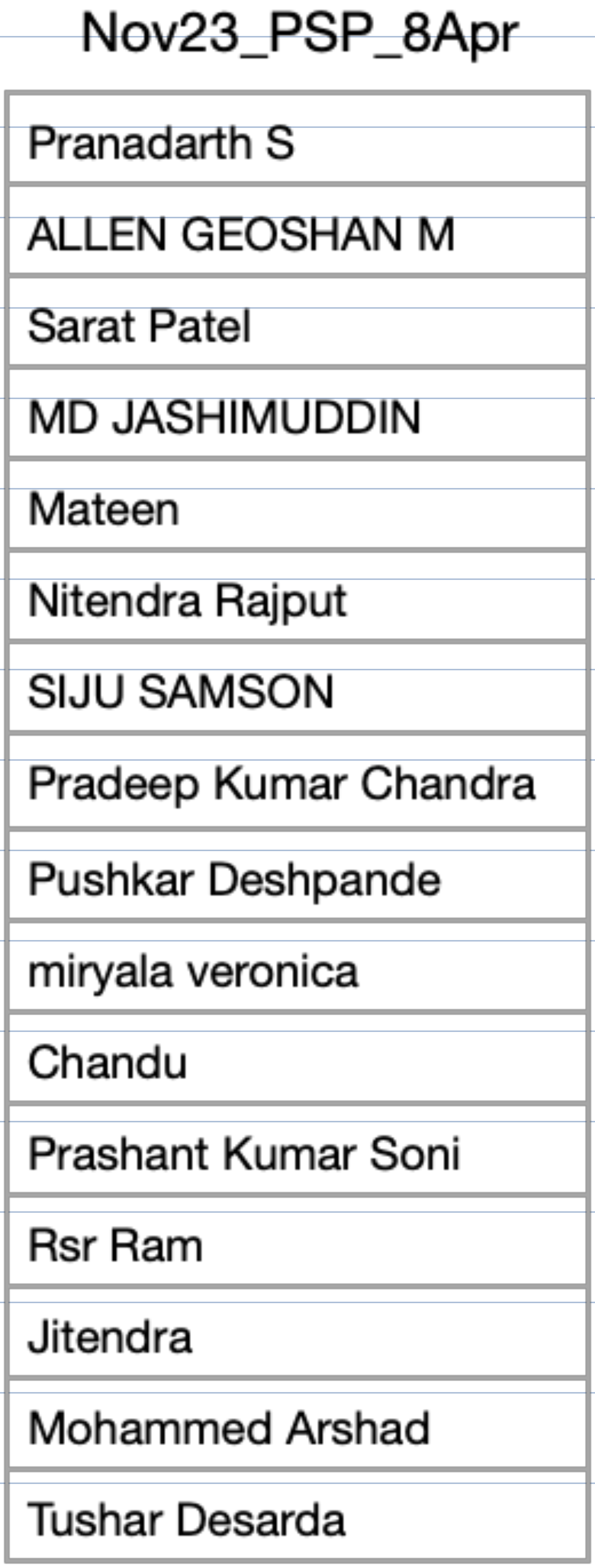
Batch PSP : 55.93 → 60



Nov23_PSP_8Apr

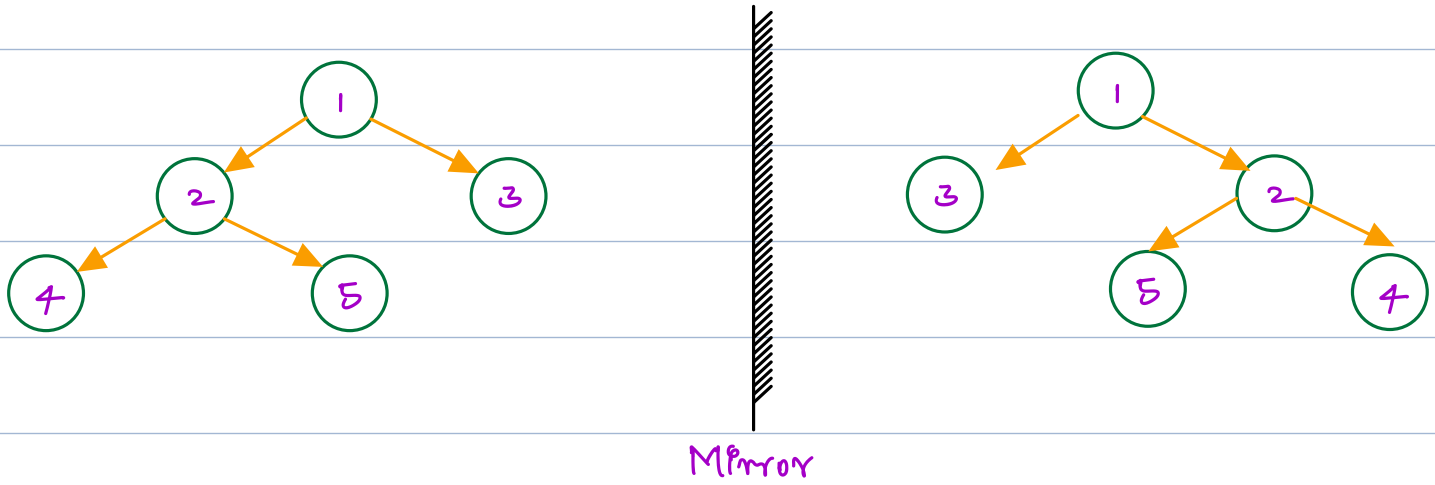
sudhakar venkatachalam
Vijay V A
Gobika K
Sai Sharath
Rajeev
Manjunatha I
Harshil Dabhoya
Kevin Theodore E
manikandan m
Mayur Hadawale
Yash Malviya
Vigneshwaran K
Robin Dhiman
Shaurya Srivastava
Suraj Devraye
kameswarreddy Yeddula



Nov23_PSP_8Apr

Pranadarth S
ALLEN GEOSHAN M
Sarat Patel
MD JASHIMUDDIN
Mateen
Nitendra Rajput
SIJU SAMSON
Pradeep Kumar Chandra
Pushkar Deshpande
miryala veronica
Chandu
Prashant Kumar Soni
Rsr Ram
Jitendra
Mohammed Arshad
Tushar Desarda

Q → Given the root of a binary tree, write a function to invert the tree



Observation

∀node, swap your left and right children

pseudo code

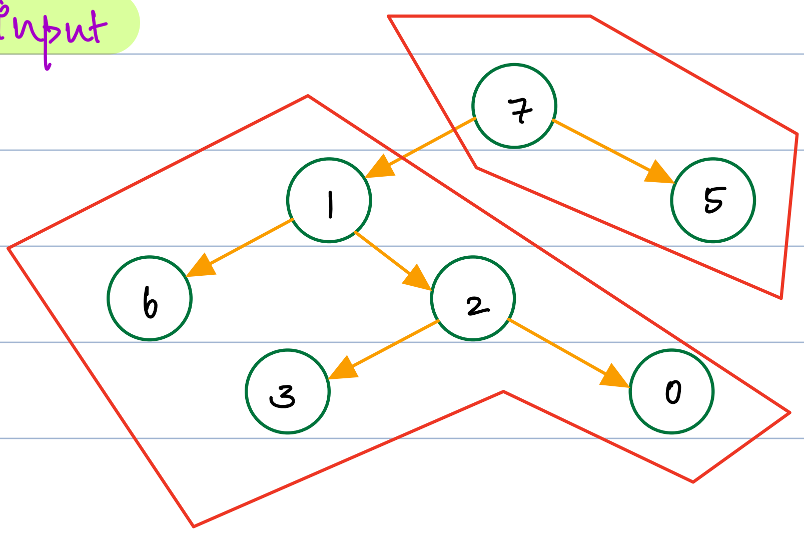
```
void invert Tree (root) {
        if (root == null) return None;
        Node temp = root.left;
        root.left = root.right;
        root.right = temp;
        invert Tree (root.left)
        invert Tree (root.right);
}
```

T.C = O(n)

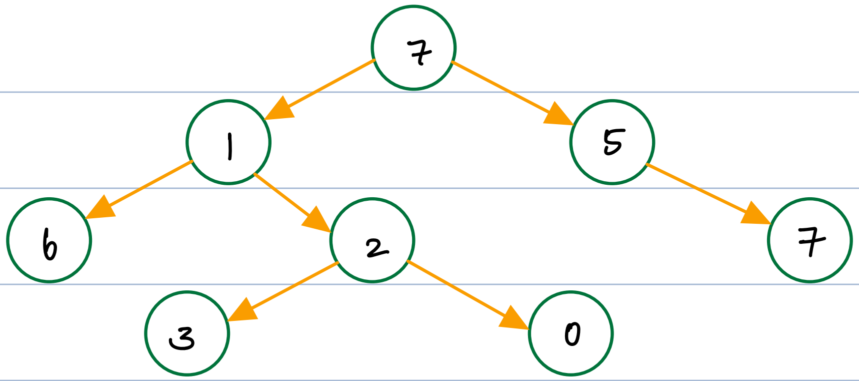
S.C = O(H)

Q → Given a binary tree, check if it is possible to divide the tree into 2 parts with equal sum of node values.

Input



True



False

Observation

If sum of 2 halfs of tree should be equal, then sum of one half of tree = $\frac{\text{total sum}}{2}$

① Get the total sum of tree nodes

② If total sum is odd ; return false

③ Check for sub tree with sum = total_sum / 2

#pseudo code

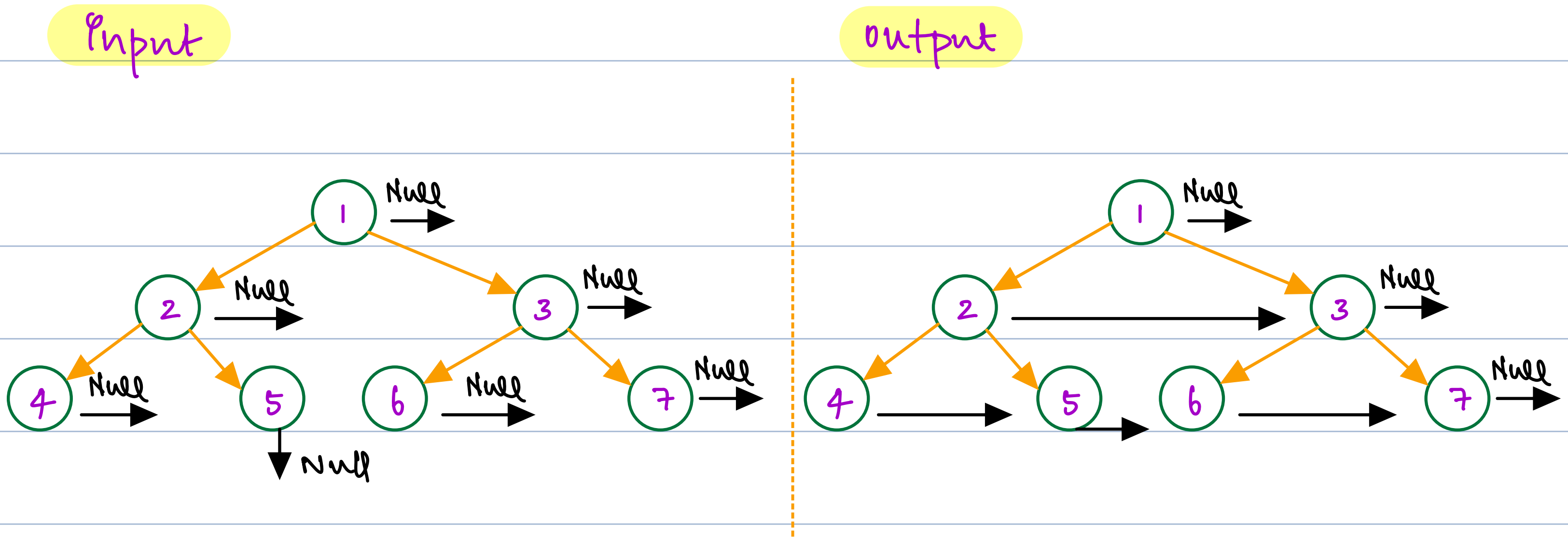
```
//step 1
int  getTreeSum (root) {
    if (root == null) return 0;
    return root.val +
                getTreeSum (root.left) +
                getTreeSum (root.right)
}

//step 2
  int totalsum = getTreeSum (root)
  if (totalsum %2 != 0) return false
  // global variable
  ans = False
  check (root, totalsum/2);

//step 3
  int check (root, targetsum) {
        if (root == null)
              return 0;
        int L = check (root.left, targetsum)
        int R = check (root.right, targetsum)
        int s = L+R+ root.val;
        if (s == targetSum) ans = true;
        return s;
}
```

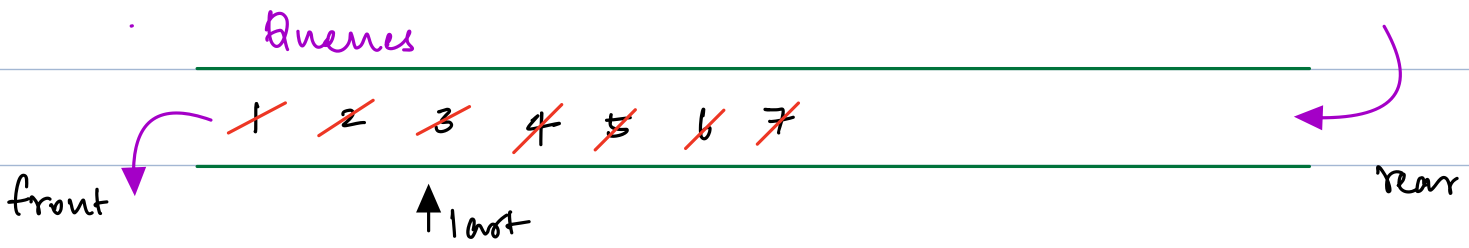
return ans;

T.C = O(n)   S.C = O(H)

Q → Given a perfect binary tree with next pointers in all nodes, initially pointing to null.

Update the next pointer to point to next node in same level ∀ nodes.





Observation

1. If (node == last) update last, node.next = null
2. else node.next = queue.front()

pseudo code

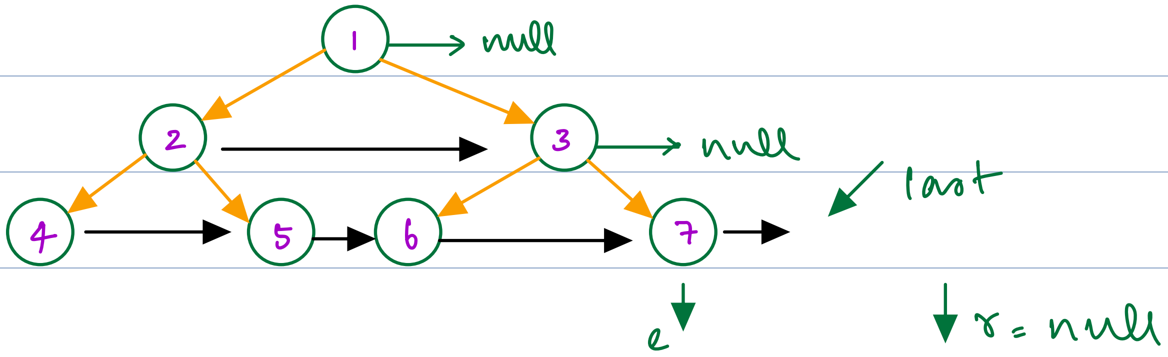
```
// Initialize my queue
queue.enqueue (root);
last = root;
while (!q.isEmpty()) {
        curr = queue.dequeue();
        if (curr.left) queue.enqueue (curr.left);
        if (curr.right) queue.enqueue (curr.right)
        if (curr != last)
            curr.next = queue.front()
        else
            if (!queue.isEmpty) last = queue.rear
}
```

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

↑ solve in O(1)

Observation

Concepts of Linked list, Queues & level order by simply using reference



pseudo code

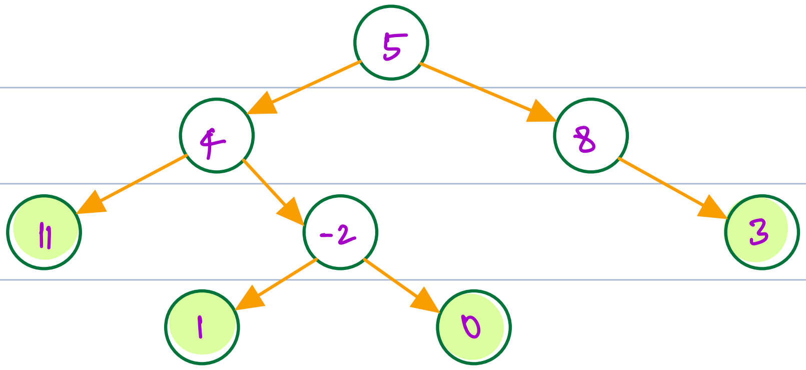
```
r = root;    last = root;
curr = root;
while (r != null) {
    if (root . left)
        curr.next = r.left;
        curr = curr.next
    if (root .right)
        curr.next = root.right;
        curr = curr.next;
    if ( r == last) {
        r = r.next;
```

T.C = O(n)

S.C = O(1)

```
        last.next = null;
        last = curr;
    }

    if (r != last)
        r = r.next;
}
```

Q → Given a binary tree & an integer k, check if there exist a root to leaf path sum = k.



K = 8 → True

K = 9 → False

Observation

1. Keep track of path sum for each node
2. If given node is leaf node check if path sum = k

pseudocode

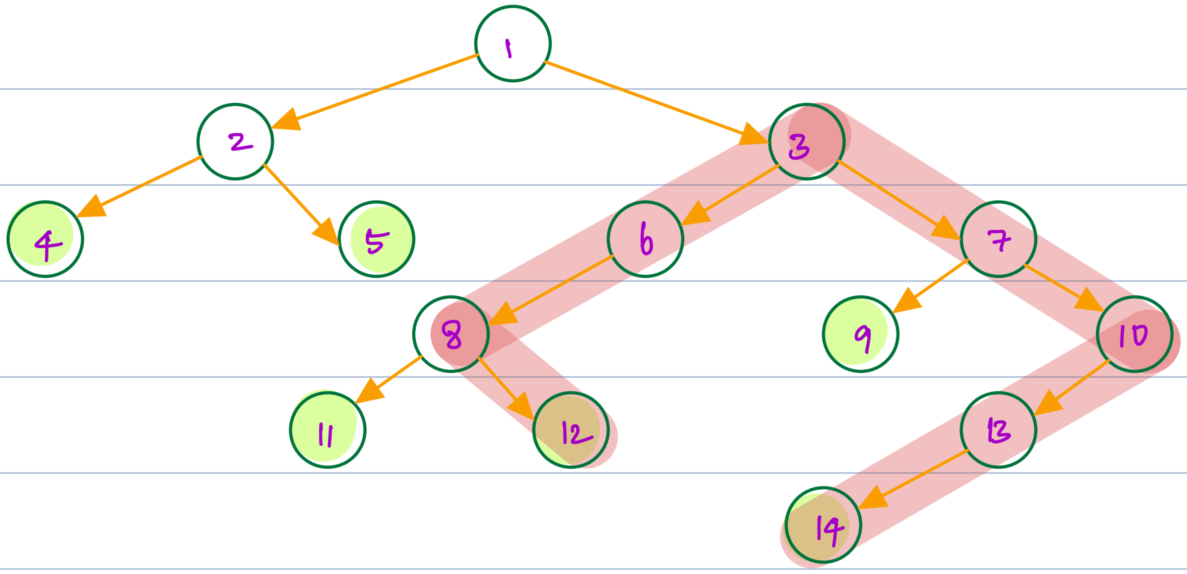
```
boolean  checkPathsum (root, k , pathsum) {
    if (root == null) return false
    pathsum = pathsum + root.val;
    if (root.left == null && root.right == null)
        if (pathsum == k) return true;
    }
    return  checkPathsum (root.left , k, pathsum)
            || checkPathsum (root.right, k, pathsum)
}
```

T.C = O(n)   S.C = O(H)

Q → Given a binary tree, find the longest path between any 2 nodes in the tree.

Diameter of Binary Tree → # nodes in the longest path between any 2 leaf nodes

what is distance from given node to farthest leaf node? Height



Observation

nodes in given Path = H LST + H RST +1

pseudo code

```
ans = INT_MIN
int get Diameter ( root) {
        if (root == null) return -1;
        H L = getDiameter (root. left)
        H R = get Diameter (root. right)
        ans = max (ans, HL +HR +1);
        return max (HL, HR) +1
}
```

T. C = O(N)   S. C = O(H)