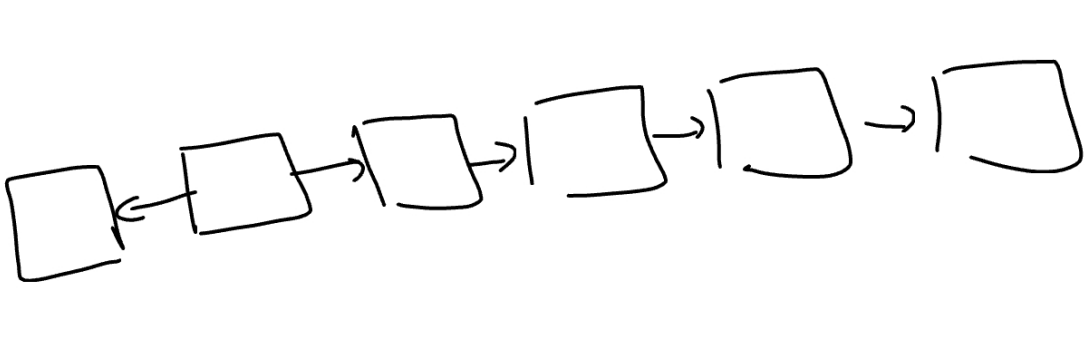
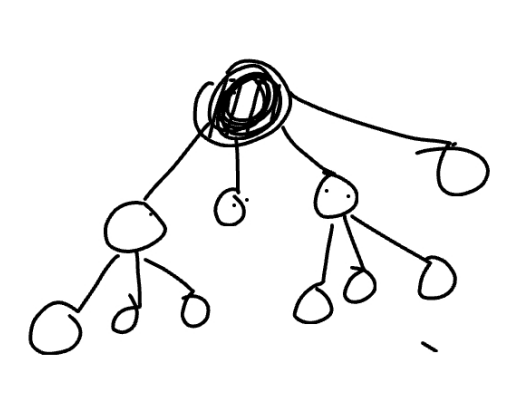




LinkedListNode

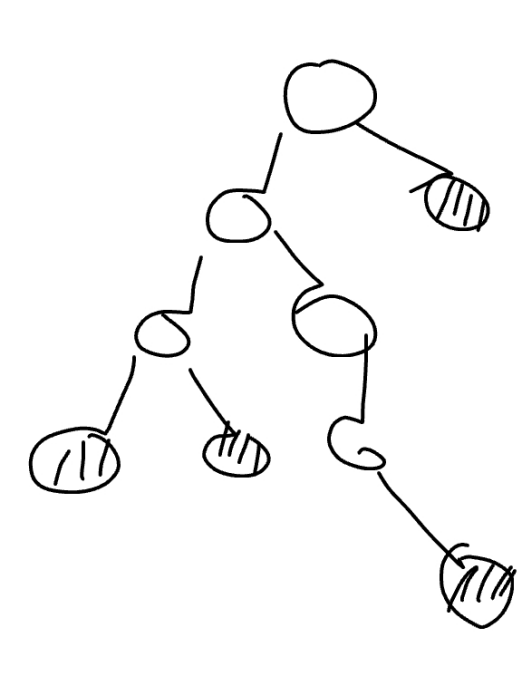
int
*next

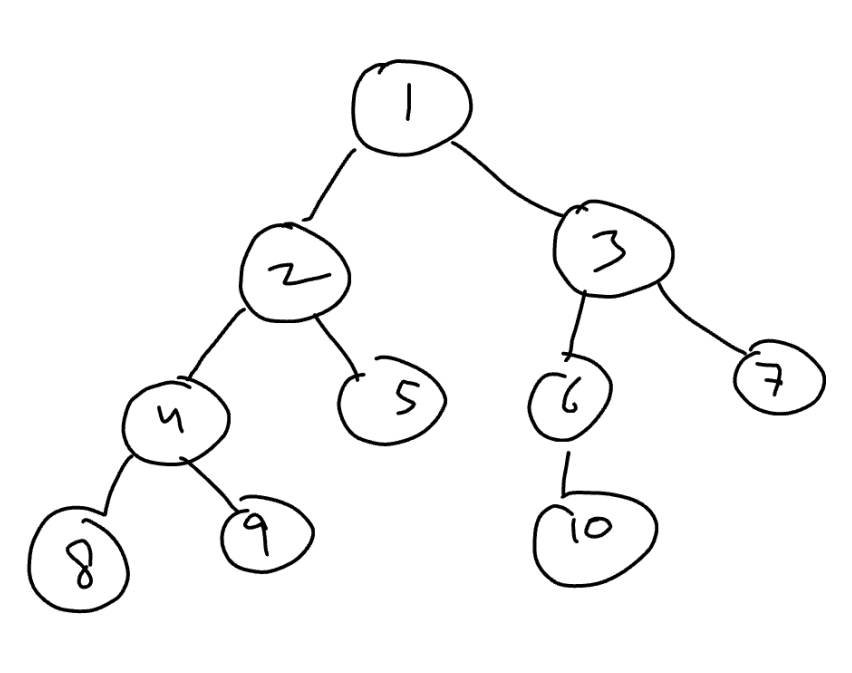
TreeNode

int
vector<TreeNode*>
children.

Binary Tree Node*

int
*left
*right

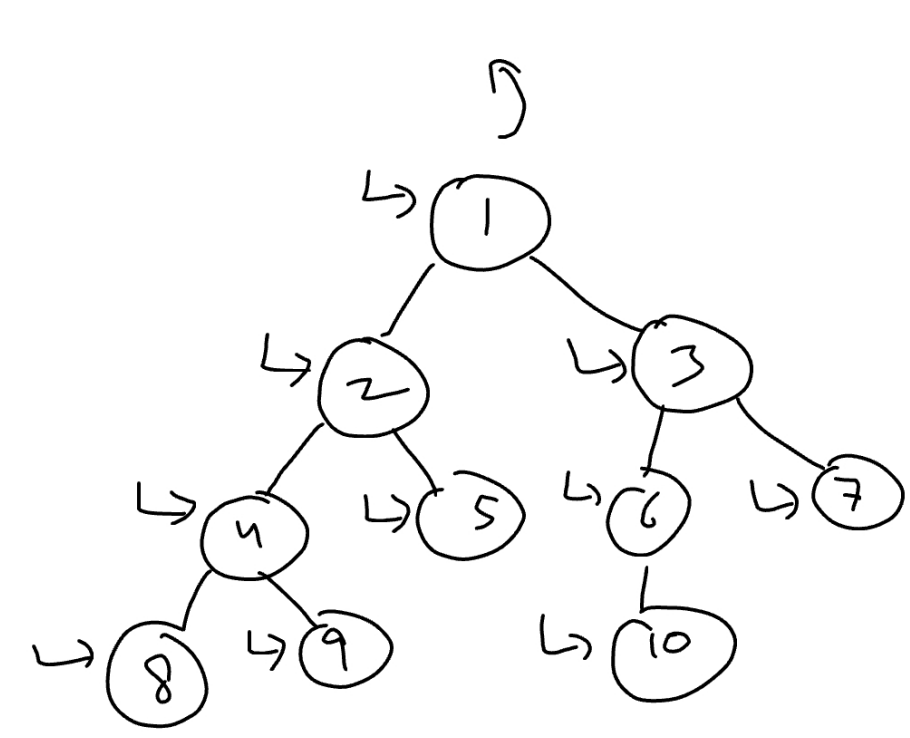



1
2
3
4
5
6
7
8
9
10

Preorder.

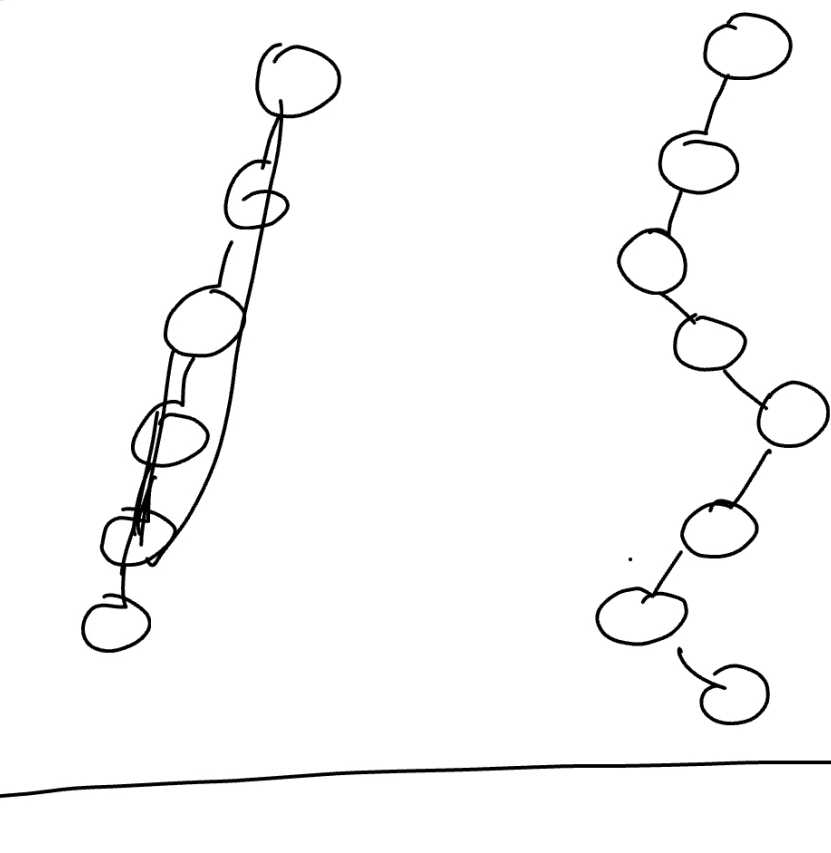
SC : Input space + AS + o/p space.

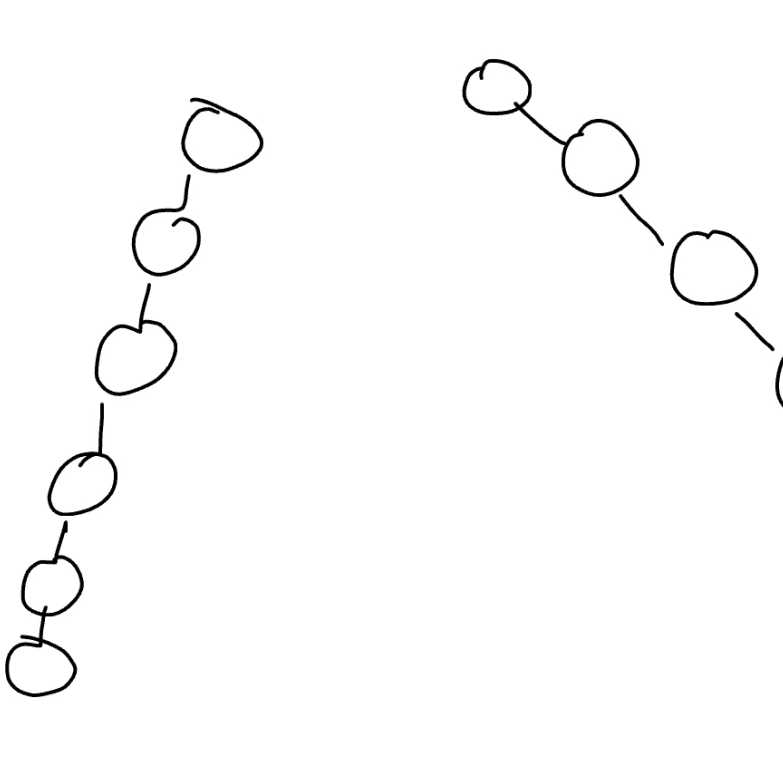
→ root
→ left subtree
→ right subtree


1
2
3
4
5
6
7
8
9
10

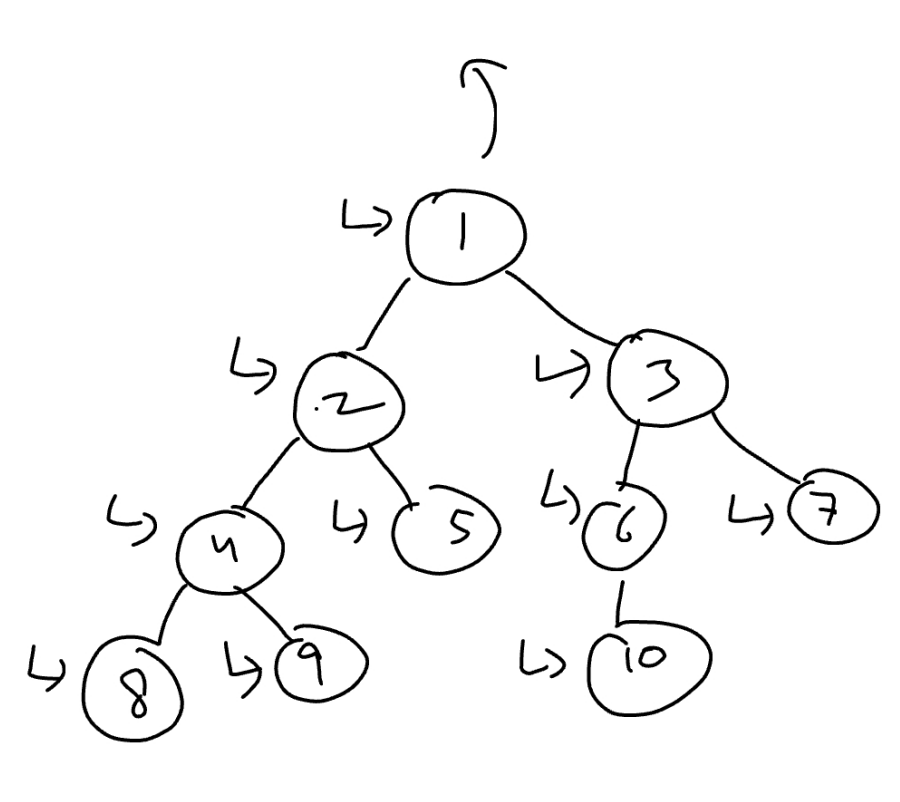
[1, 2, 4, 8, 9, 5, 3, 6, 10, 7]

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



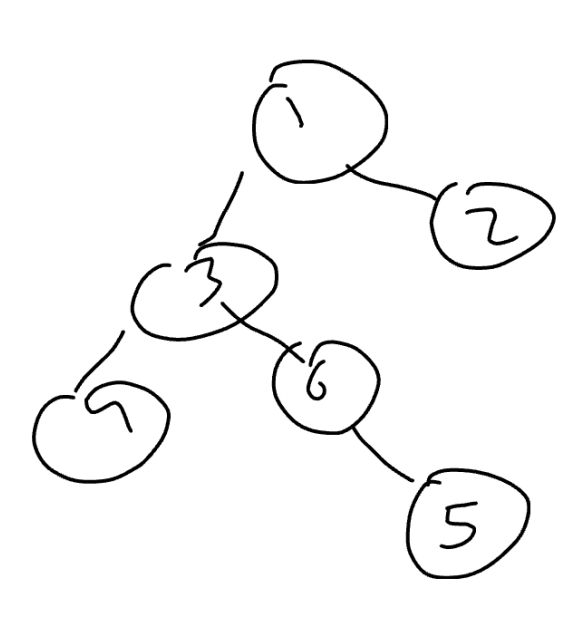


Postorder.


1
2
3
4
5
6
7
8
9
10

→ left subtree
→ right subtree
→ root

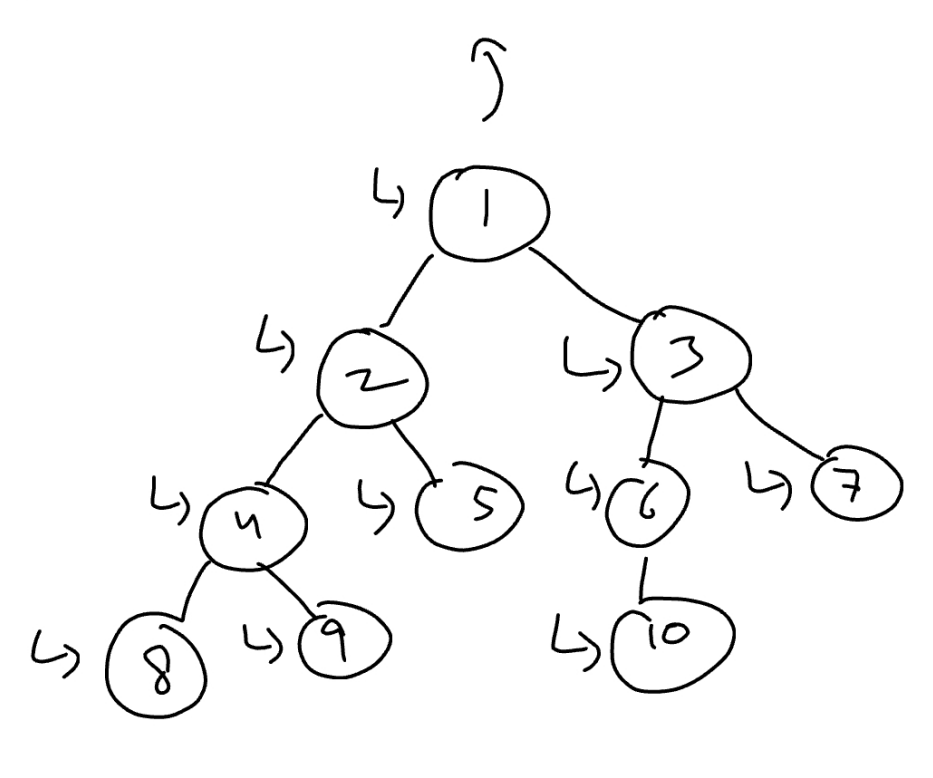
[8, 9, 4, 5, 2, 10, 6, 7, 3, 1]



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

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

Inorder.

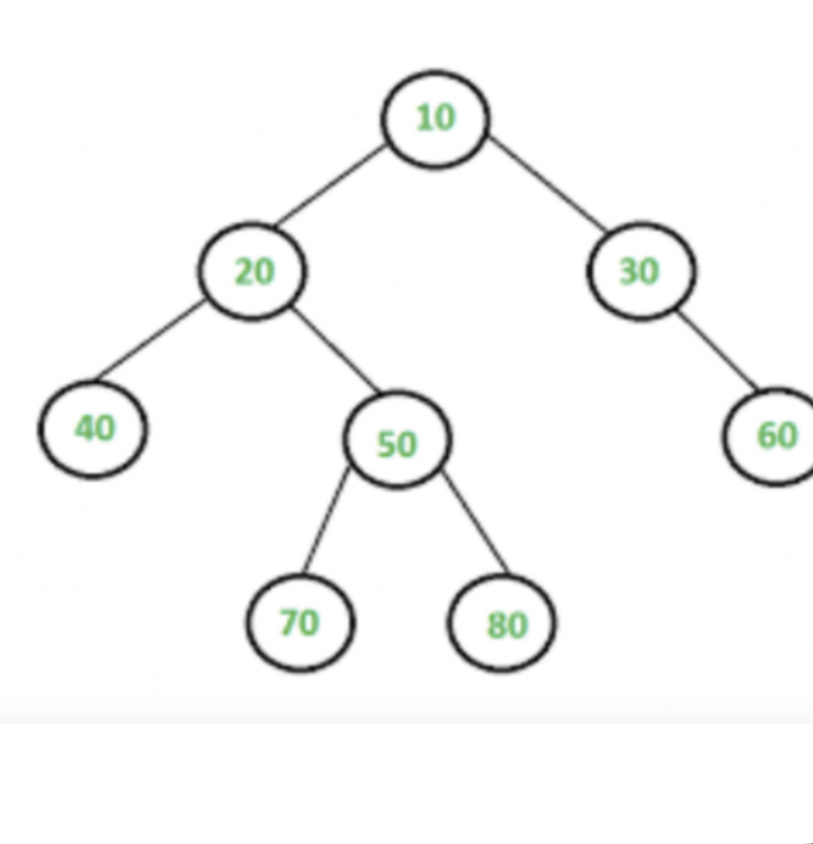

1
2
3
4
5
6
7
8
9
10

→ left subtree
→ root
→ right subtree.

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

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

Level-order Traversal


10
20
30
40
50
60
70
80

[10, 20, 30, 40, 50, 60, 70, 80].

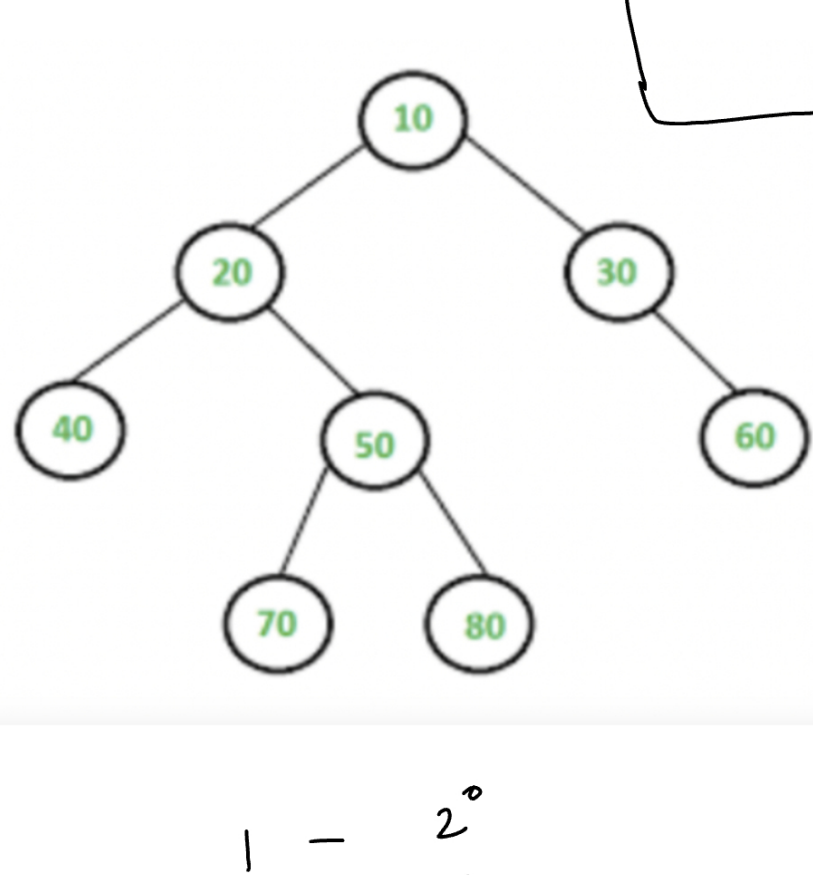
[10, 20, 30, 40, 50, 60, 70, 80

Hypothetical DS.

~~10~~, ~~20~~, ~~30~~, ~~40~~, ~~50~~ ~~60~~

Queue.

1) Pop from Queue - curr.
2) Process the curr node
3) Push the child nodes of curr into the Queue.

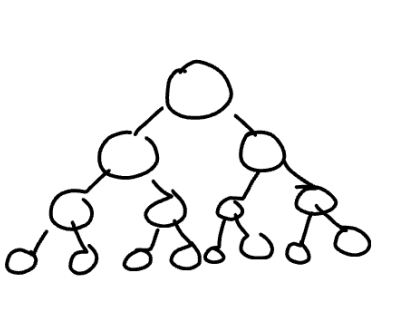

10
20
30
40
50
60
70
80

Queue.
~~10~~ ~~20~~ 30 ~~40~~ ~~50~~
~~60~~ ~~70~~ ~~80~~

[10, 20, 30, 40, 50, 60, 70, 80].

1 — 2^0
2 — 2^1
4 — 2^2
4 ↳ 8
8 ↳ 16
16 32 2^H
⋮

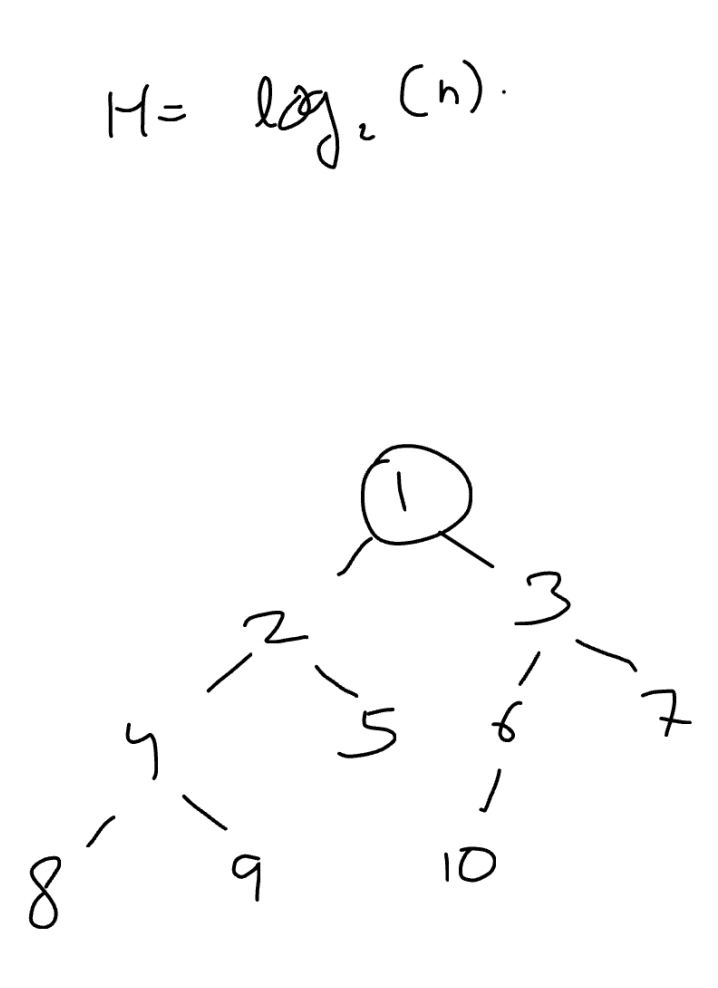
x
2x
4x
8x



$2^{H-1} \Rightarrow 2^{\log n - 1} = O(n).$

$H = \log_2(n).$

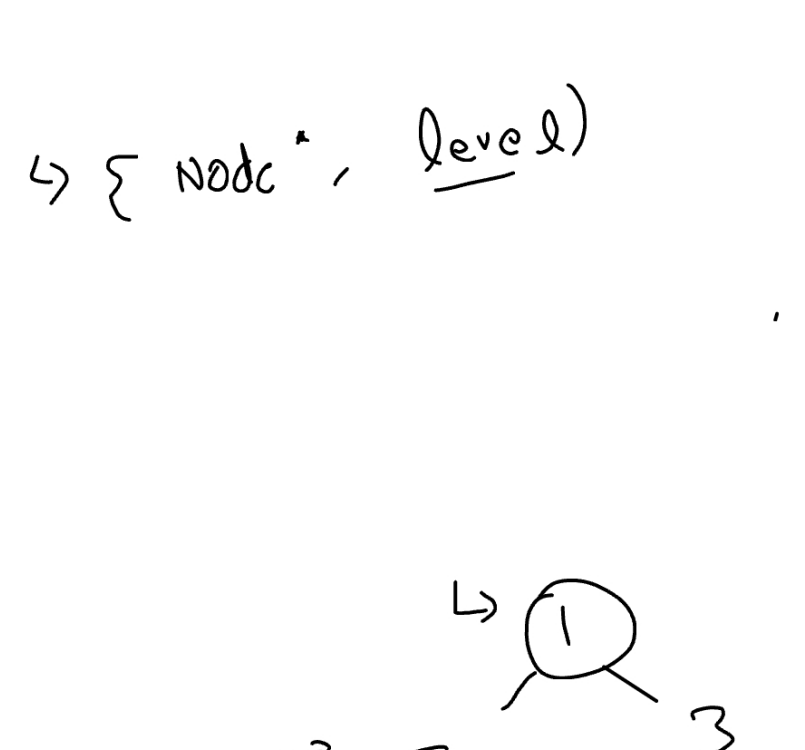
1
⋮
n/16
n/8
n/4
n/2


1
2
3
4
5
6
7
8
9
10

↳ { Node*, level)

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

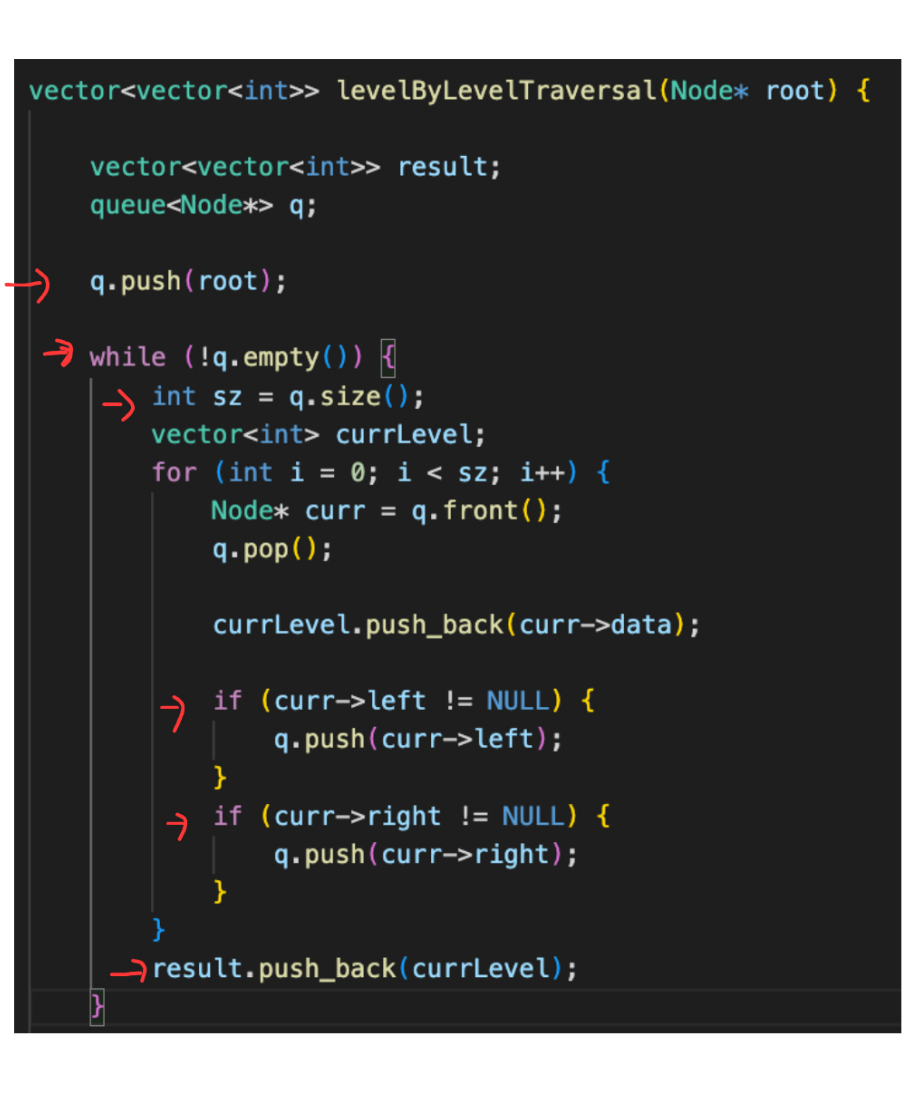
~~(1, 0)~~
~~(2, 1)~~
~~(3, 1)~~
(4, 2)
(5, 2)


1
2
3
4
5
6
7
8
9
10

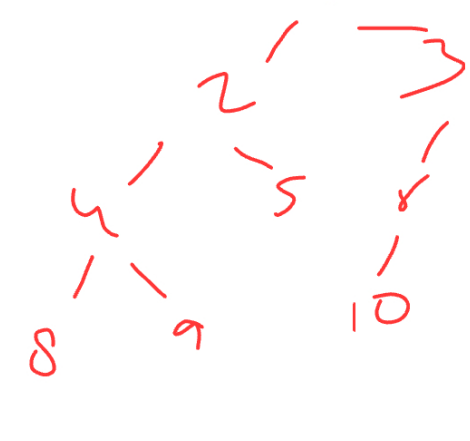
Queue.

~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, ~~7~~, ~~8~~, ~~9~~, ~~10~~

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



```
vector<vector<int>> levelByLevelTraversal(Node* root) {

    vector<vector<int>> result;
    queue<Node*> q;

    q.push(root);

    while (!q.empty()) {
        int sz = q.size();
        vector<int> currLevel;
        for (int i = 0; i < sz; i++) {
            Node* curr = q.front();
            q.pop();

            currLevel.push_back(curr->data);

            if (curr->left != NULL) {
                q.push(curr->left);
            }
            if (curr->right != NULL) {
                q.push(curr->right);
            }
        }
        result.push_back(currLevel);
    }
```

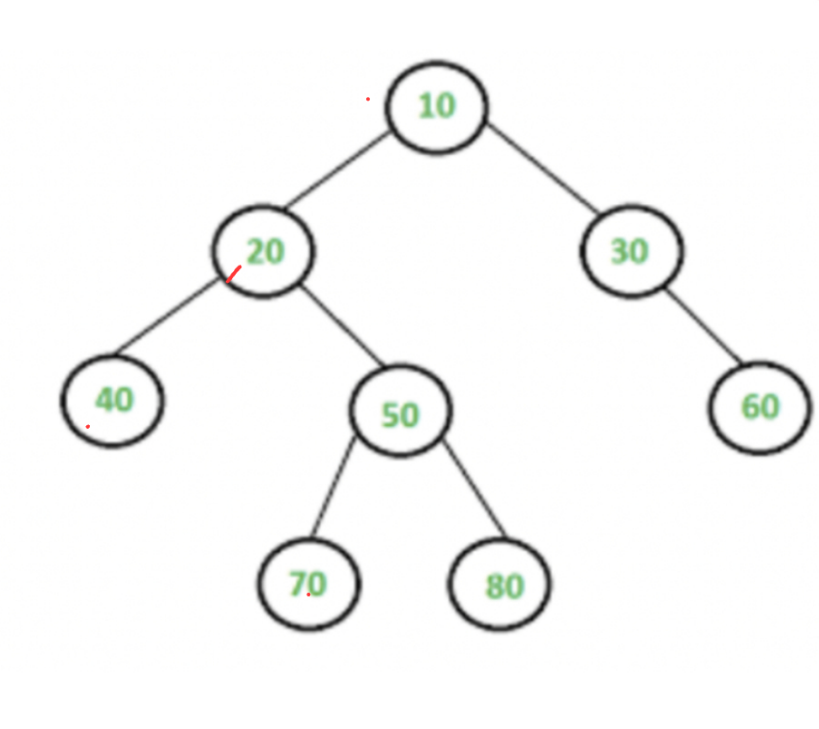

1
2
3
4
5
6
7
8
9
10

sz = ~~1~~ ~~2~~ ~~4~~ 3

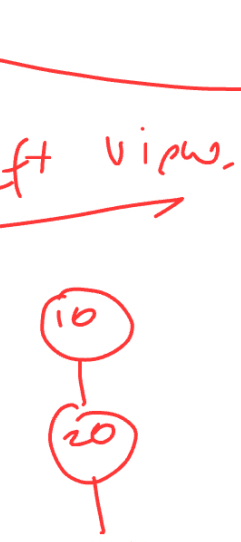
Queue.

~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, ~~7~~, ~~8~~, ~~9~~, ~~10~~

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

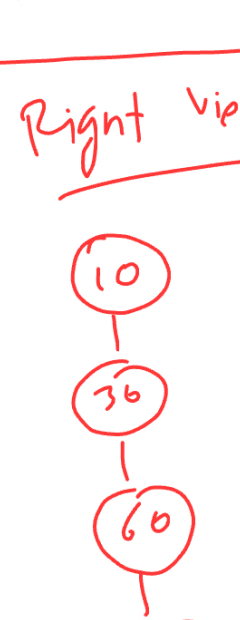

10
20
30
40
50
60
70
80

Left view.

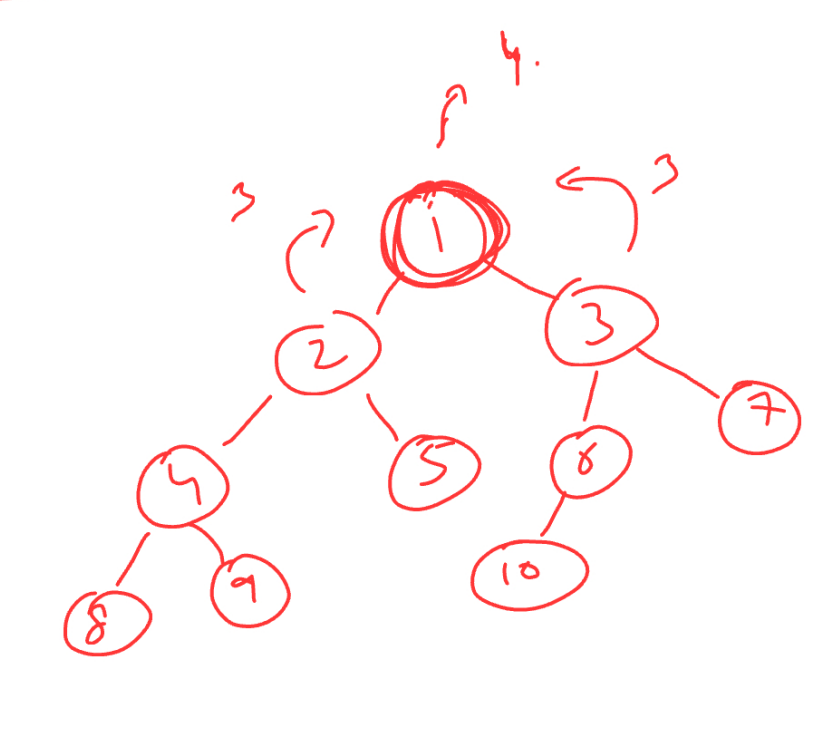


[10, 20, 40, 70]

Right view.



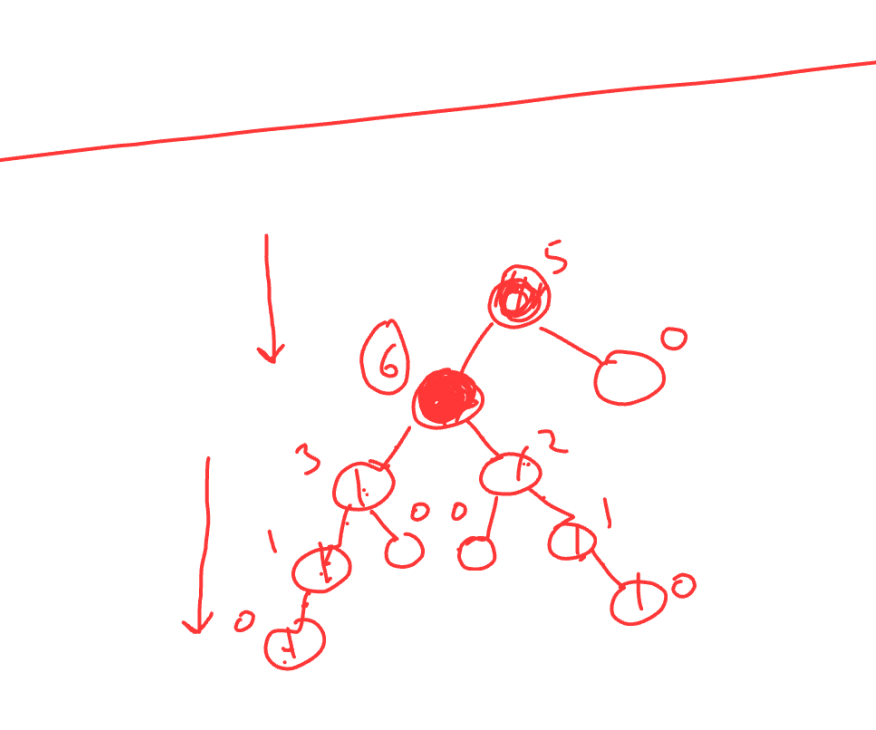
[10, 30, 60, 80].


4
3
3
1
2
3
4
5
6
7
8
9
10

lh, rh

H = max(lh, rh) + 1

Height(root) = max(Height(root -> left), Height(root -> right)) + 1


5
6
3
2
1
1
0
0
0
0
0
0

Diam = ~~5~~ 6

lh + rh.

⇒ Consider the possibility of each node being the correct diverging node.

Brute Force.

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

DP Memoization: