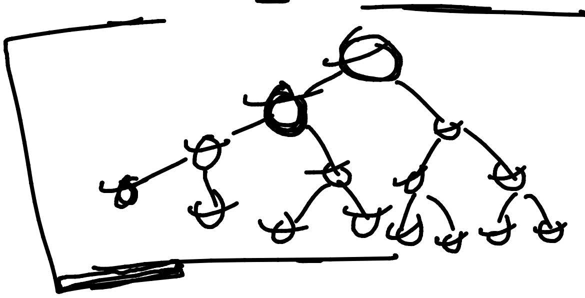
CLASS 58

15 April 2023 09:07

Class 58

Height Balanced Tree

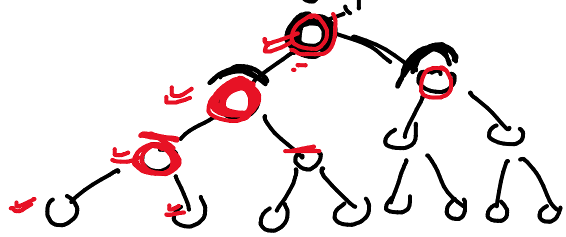


$|l_H - R_H| \le 1$

Tree (root)

Recursion

root ↓ $(l_H - R_H) \le 1$ HBT (root)



at every Node 2 condition follow

- left and right subTree are HBT

→ at a Node $|L_H - R_H| \le 1$

$n * O(n) = O(n^2)$

Node

T.C = $O(n^2)$

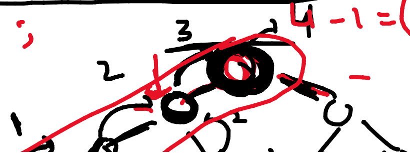
```
bool check( Node * root )
{
    if(root == NULL)
        return True;

    int lH = height(root -> left);      O(n)
    int rH = height (root -> right);

    bool hbt = check (root -> left) and check(root -> right);

    if (!hbt)
        return false;

    if( abs( lH - RH) < = 1)
        return true;

    return false;
```

(bool , Height)

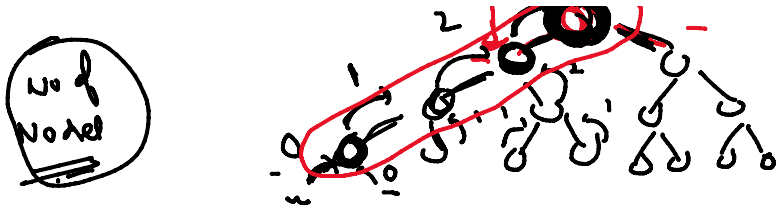
$4 - 1 = 3$

height O(n)

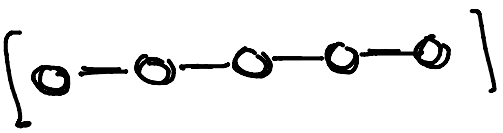
Path = No of edges = No of nodes - 1



No of



Pair = [No of edges] = [No of nodes - 1]



```
Pair <int, bool> check (Node *root)
{       if (root == NULL)
        { return {0, 1};
        }
```

T.C. = O(n)

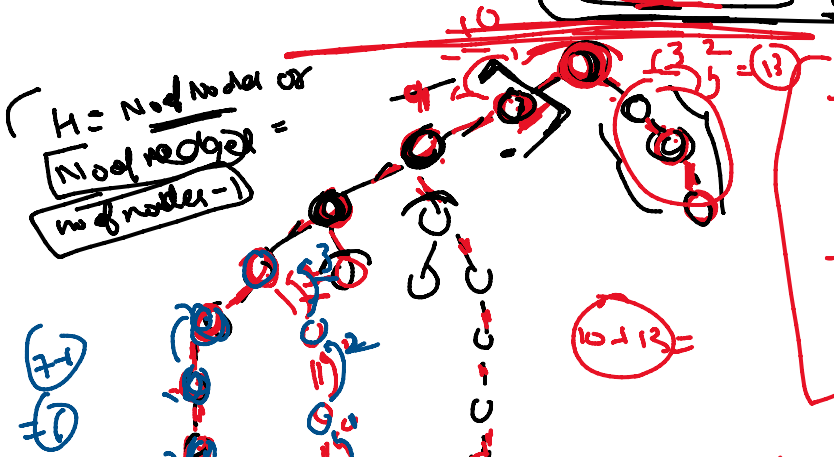
No of nodes

```
Pair <int, bool> left = check (root → left);
Pair <int, bool> right = check (root → right);
if (left.second and right.second)
{       if (abs (left.first - right.first) <= 1)
        {       return {h, 1};
        }
        else
        return {h, 0};
}
return {h, 0};
```

→ int h = 1 + max (left.first, right.first);

Diameter of Binary Tree

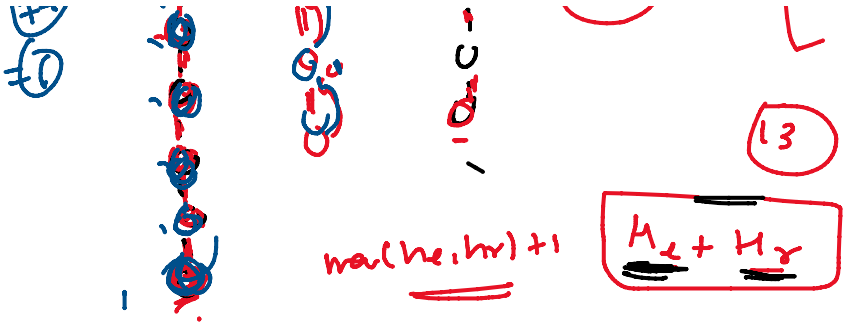
longest distance b/w 2 points



→ Diameter can pass through root Node

→ Diameter can be either in left subtree or right subtree

height(lsub) + h(rsub) + 2



13

$\max(h_l, h_r) + 1$

$H_l + H_r$

height(lsub) + h(rsub) + 2

int op1 = $H_l + H_r$ = 10 + 3 = 13

int op2 = Diameter (left);

int op3 = Diameter (right);

return max (op1, max (op2, op3))

Diameter = Distance b/w 2 Nodes;

⇒ No of edges

```
int diameter (Node* root)
{
    if (root == NULL)
        return 0;
    int op1 = height (root->left) + height (root->right);
    int op2 = diameter (root->left);
    int op3 = diameter (root->right);
    return max (op1, max (op2, op3));
}
```

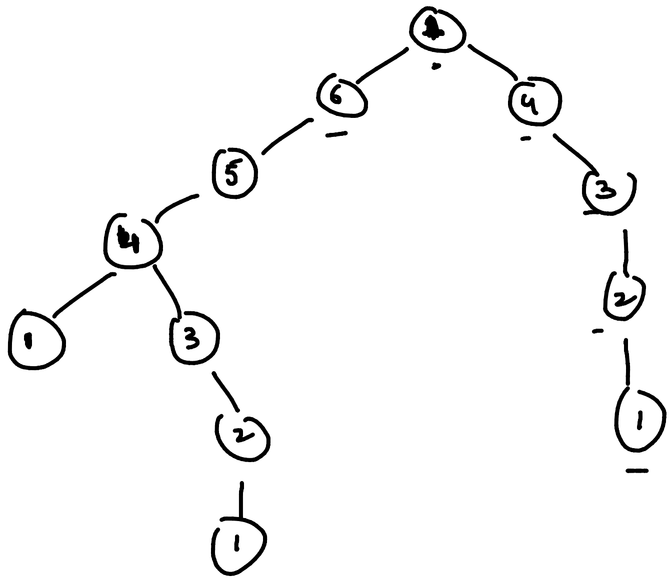
height diameter

```
Pair<int, int> diameter (Node* root)
{
    if (root == NULL)
    {
        return {0, 0};
    }
    auto left = diameter (root->left);
    auto right = diameter (root->right);
    int op1 = left.first + right.first;
    int op2 = left.second;
```

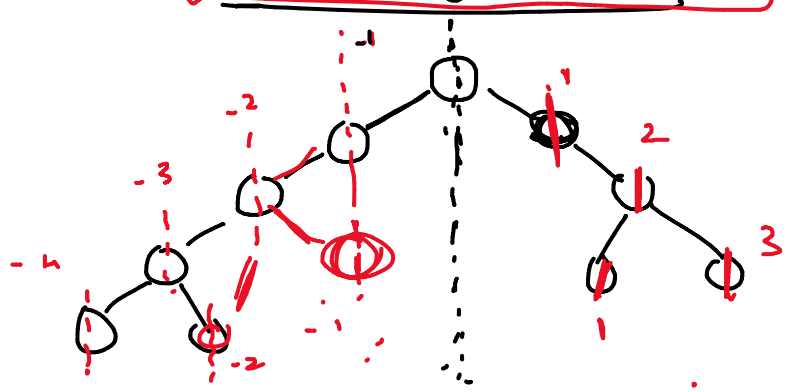
```
int op2 = left.second;
int op3 = right.second;

int d = max(op1, max(op2, op3));
int h = 1+ max(left.first, right.first);
} return {h,d};
```

```
diameter(root).second;
```

input data



Vertical order traversal



-4 -1)

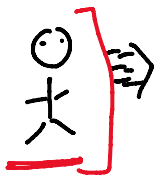
-4 -1
-3
-2

```
void vot( Node *root, map<int, vector<int>>& ma, int colno)
{
    if(root == NULL)
        return;
    [ma[colno].push_back(root->data);]
    vot(root->left, ma, colno-1);
    vot(root->right, ma, colno+1);
}
```

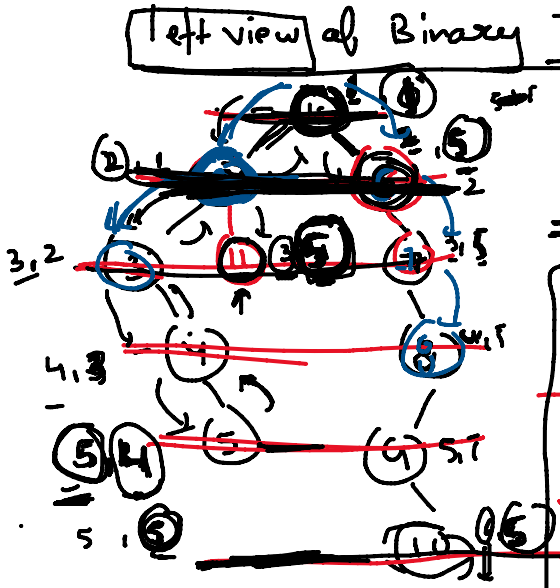
(-4, root) (-4, root)

```
for(auto i: ma)
{   cout << i.first << " ";
    for(auto j: i.second)
    {   cout << j;
    }
}
```

3



DFS



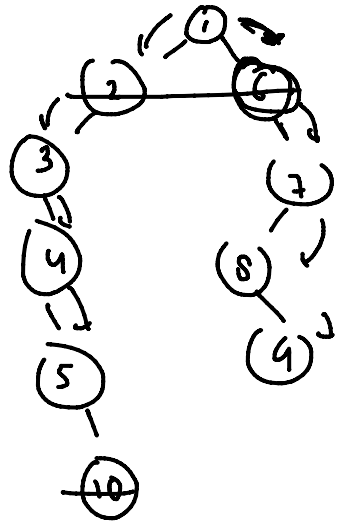
BFS

```
while(!qv.empty())
{   int n = qv.size();
    for(i=0; i<n; i++)
    {   if(i==0)
        {   cout << qv.front();
        }
        qv.pop();
        if(   )
        {
```

```
void leftview(Node* root, int level, int& maxlevel)
{   if(root == NULL)
        return;
    if(maxlevel < curlevel)
    {   cout << root->data;
```

```
        maxlevel = curlevel;
    }
    leftview ( root->left , curlevel+1, maxlevel);
    left-view ( root->right , " , " );
}
```

change order for right view



→ 1 6 7 8 9 10