CLASS 59

15 April 2023 12:10

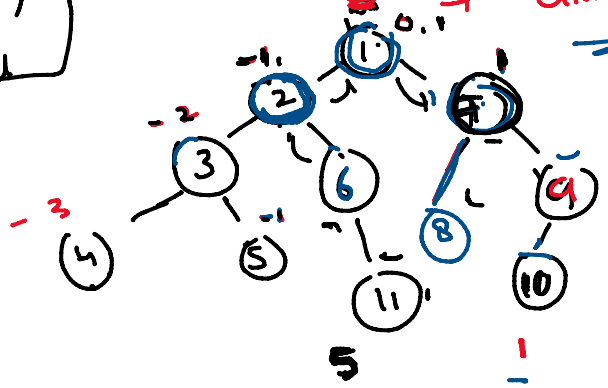
Tree

left view / right view

↳ along the row / Along the level

Top View

Along the column + First Node of every column



⇒ 4 3 2 1 7 9

0 → 1
-1 → 2
-2 → 3
-3 → 4
1 → 11
2 → 9

⇒ 1 2 3 4 7 9

1 2 3 4 11 9

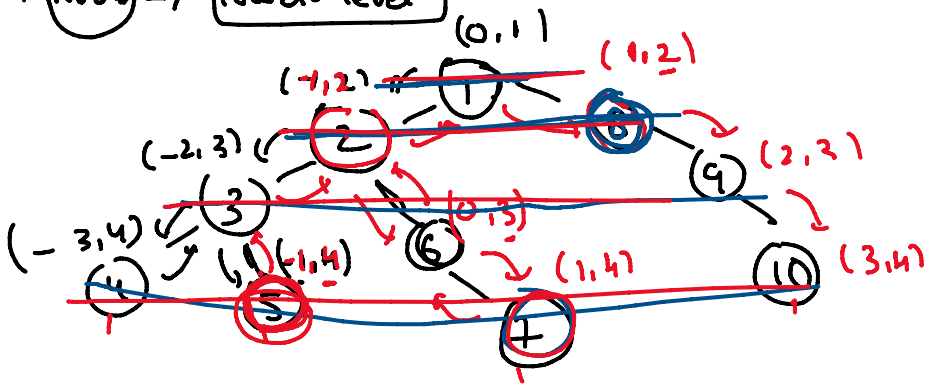
Vertical order Traversal

DFS , BFS

DFS ⇒ Higher depth then node at lower depth

Column → Node → lowest level

map < int , pair < int int > > mp



0 - {1,1}
-1 - {2,2}
-2 → {3,3}
-3 → {4,4}
1 → {2,8}
2 → {3,9}
3 → {4,10}

```
void topview ( Node* root, int col, int level, map<int, pair<int,int>>& mp)
{
    if ( root == NULL )
    {   return;
    }
    if( mp.find(col) != mp.end() )
    {   if ( mp[col].first > level )
        {   mp[col] = { level, root->data };
        }
    }
```

```
    }
    else
    {
        mp[col] = { level, root->data };
    }
    TopView( root->left, col-1, level+1, mp);
    "      ( root->right, col+1, "     , mp);
}
```

// Print map

BFS ⇒ level order wise

col → lower level Node

```
queue< Pair<int,Node*> > q;
q.push ({0, root});
map<int, int> mp;
while( !q.empty() )
{ auto f = q.front();
  if(mp.find(f.first) == mp.end())   Remove Bottom view
    { mp[f.first] = f.second->data;
    }
    q.pop();
  if( f.second->left)
  { q.push({ f.first-1, f.second->left});
  }
  if( f.second->right)
  { q.Push({ f.first+1, f.second->right});
  }
}
```

"Print Mal

Bottom View → Deepest Node of each column

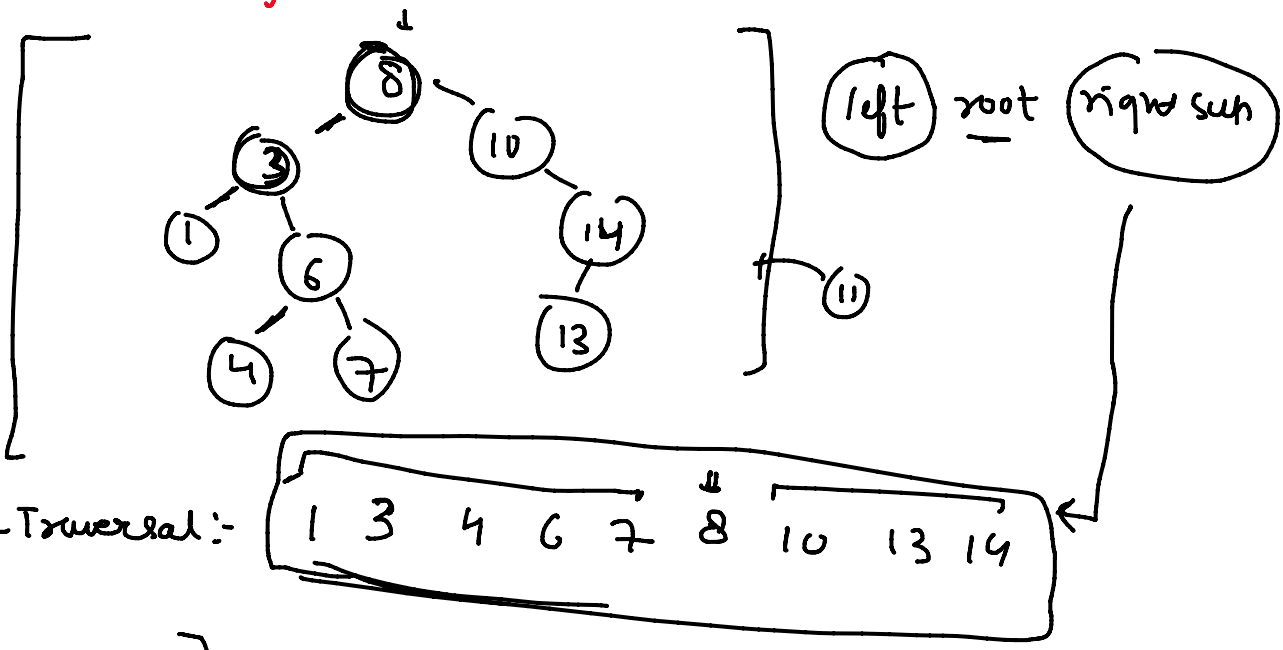
Difference?

Binary Tree

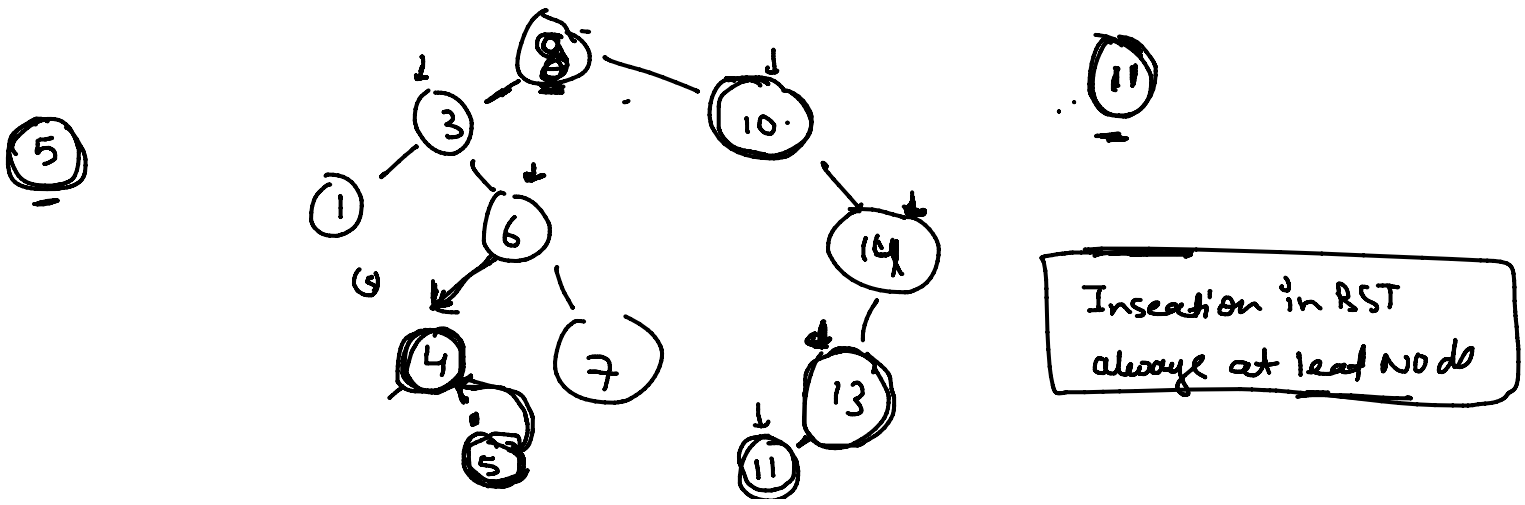
Binary Search Tree (BST)

Special Type of Binary Tree

- → Each Node in left subtree is less than root Node value
- → " " " Right " " more " " " "

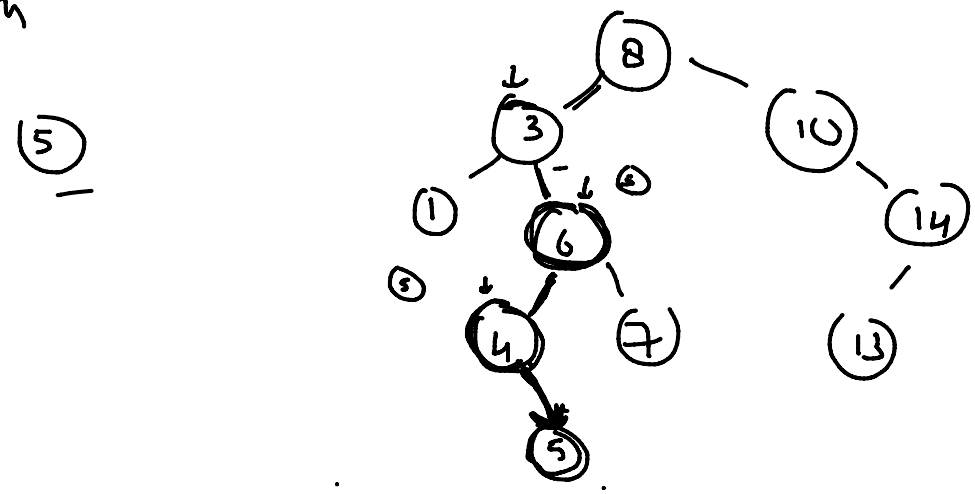


[Insertion in BST]

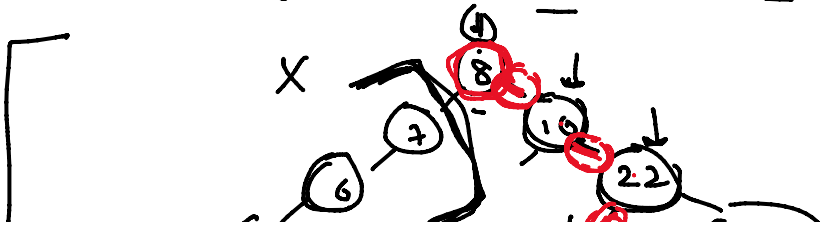


Insertion in BST always at leaf Node

```
Node * insertBST ( Node * root, int d)
{
    if (root == NULL)
    {
        Node * newNode = new Node (d);
        return newNode;
    }

    if ( d > root -> data )
    {
        root -> right = insertBST ( root -> right , d);
    }
    else
    {
        root -> left = insertBST (root -> left, d);
    }

    return root;
}
```



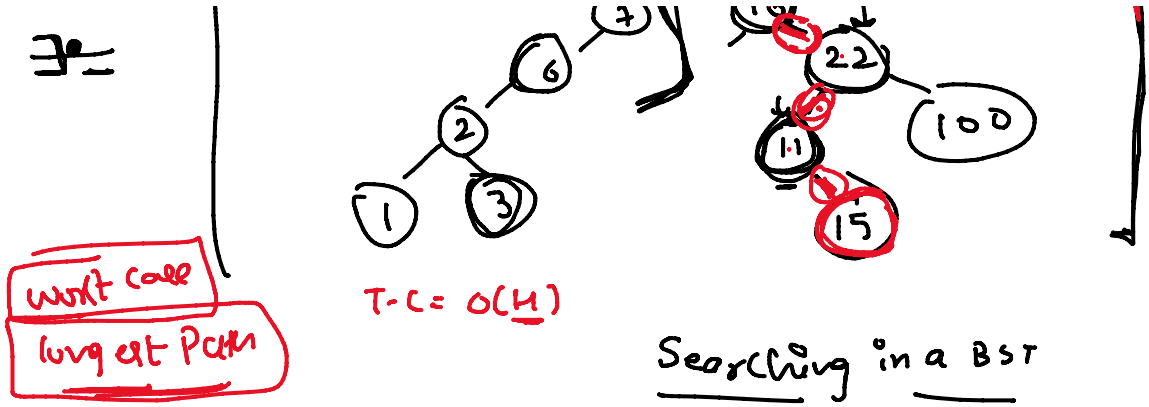
8 7 6 10 22 11 15



2 3 1 1 6 0

T.C = ? O(logn)

O(H)



work done
longest Path

T.C = O(H)

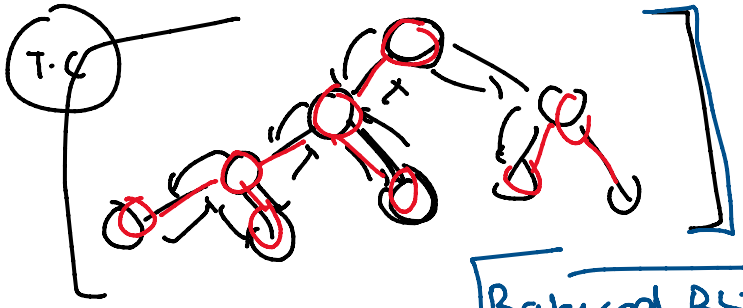
Normal BT

BST

O(H)

Searching in a BST

only comparison



B.T

T.C = O(n)

O(H)

O(n) = O(n+H)x

Balanced BST

$(lH - rH) \leq 1$

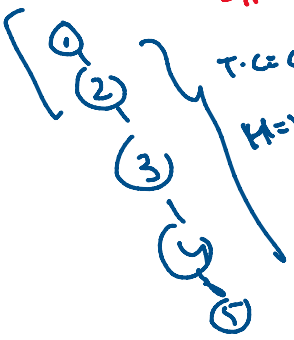
$H = \log_2(n)$

T.C = $O(\log_2 n)$

BST → O(H)

T.C = O(n)

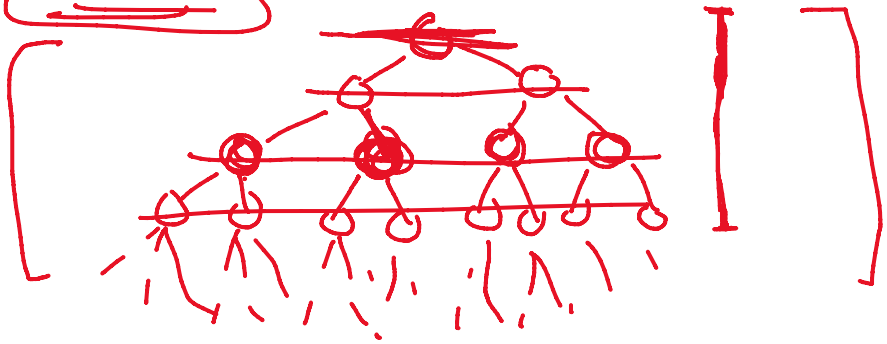
H = n



Balanced Binary Tree

Node - 2 child

Complete BT



$[\ 2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^h\ ] = n$

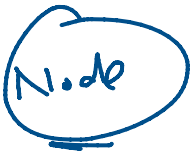
$l_w\left(2^h - 1\right) = n$

height of a BT = $\log_2(n+1) \sim O(\log_2(n))$

$2^h = n+1 \Rightarrow h = \log_2(n+1)$

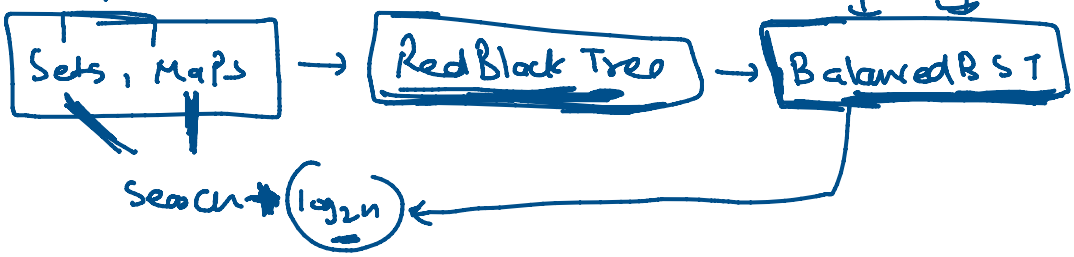
Balanced

Searching in BST



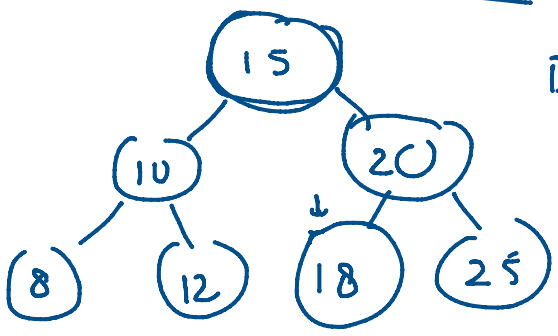

```
Node* search(Node* root, int d)
{
    if (root == NULL)
        return NULL;

    if (root->data == d)
        return root;

    if (root->data > d)
    {
        return search(root->left, d);
    }
    else
        return search(root->right, d);
}
```

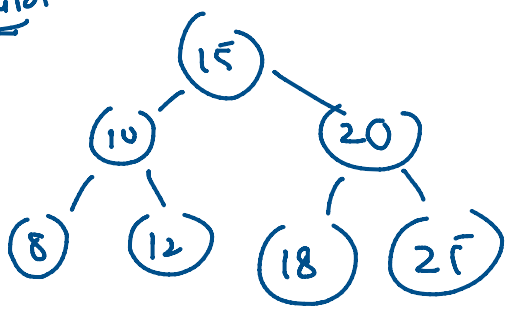
insertion (or) $\log_2(n)$

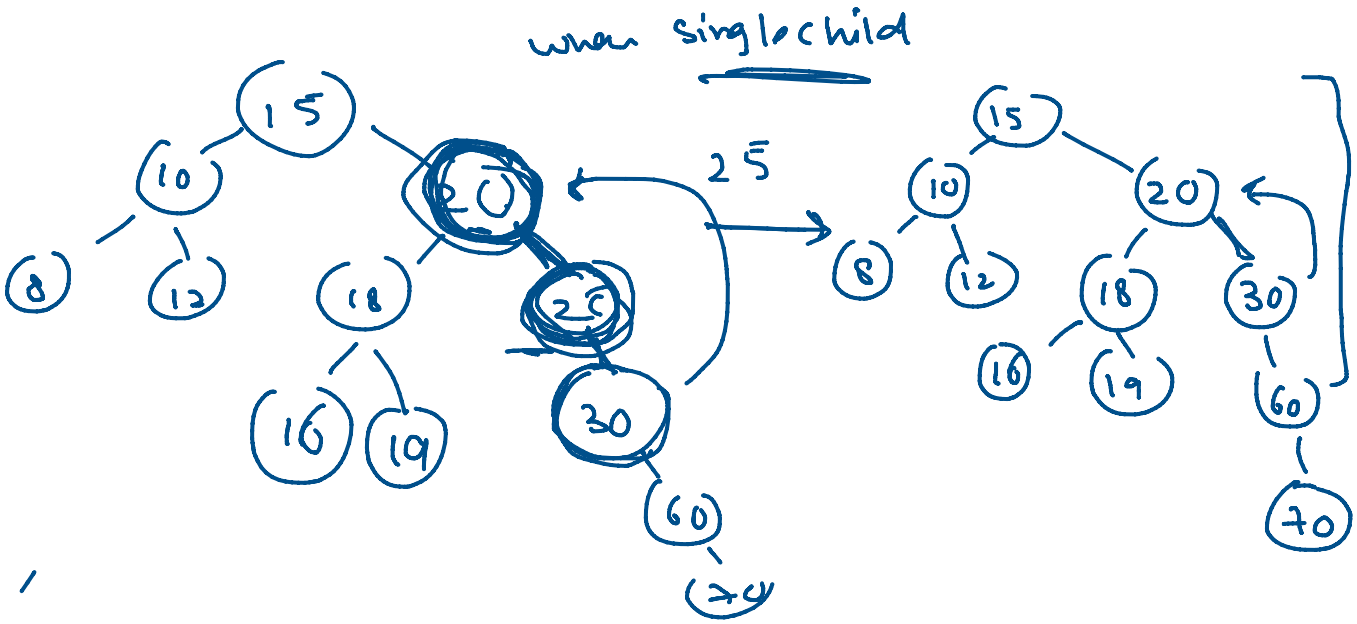


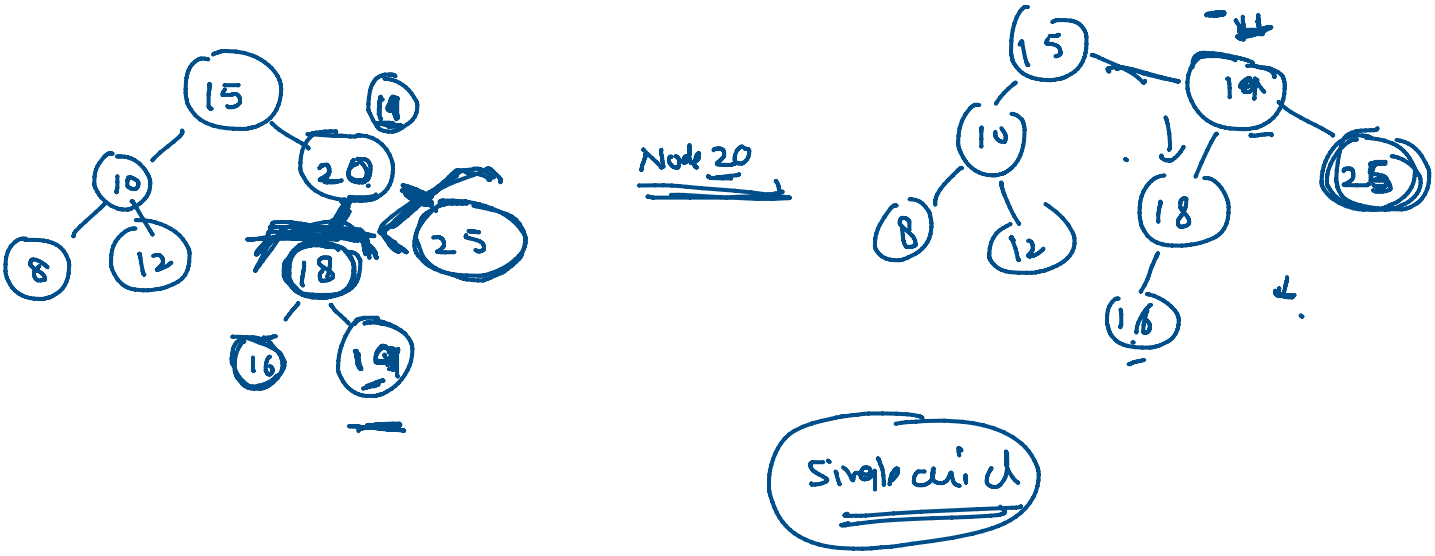
Hash table

Deletion in BST









Single child