**What is binary tree**

- Binary Tree is an Non-linear data structure unlike List , Stack and Queue

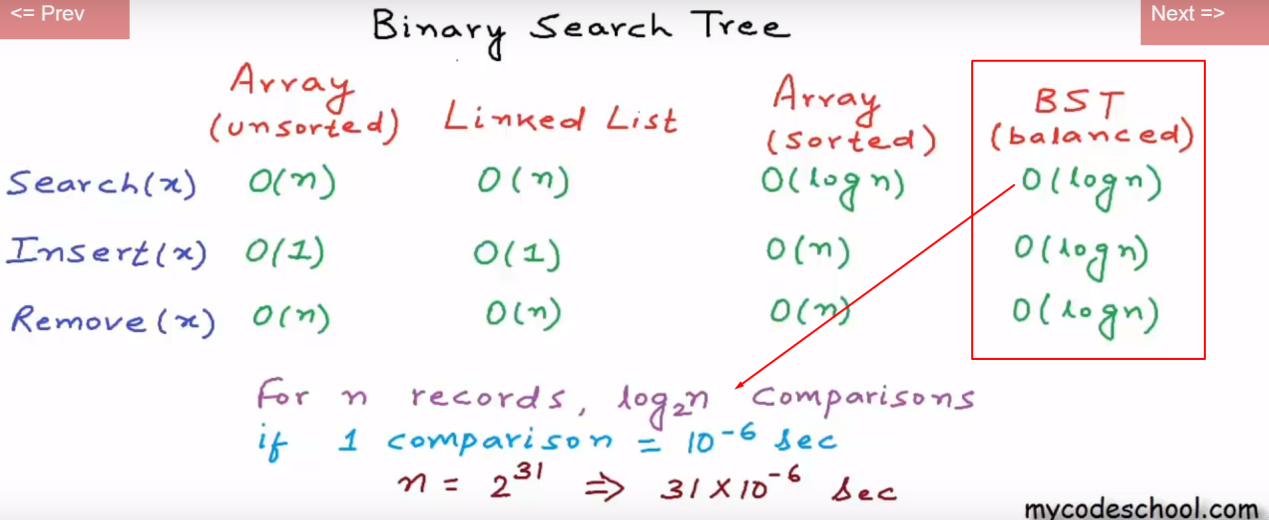
- In Binary Tree every node can only have max two child ,unlike normal tree can have any number of child , a complete binary tree is that every node always have 2 child.

**Advantage of binary tree**

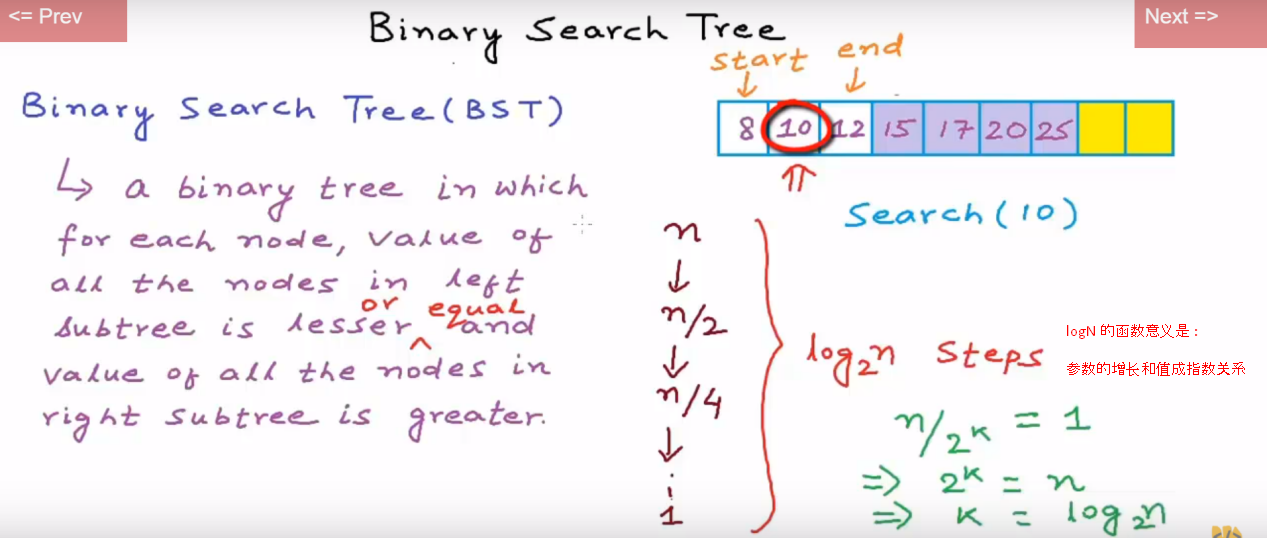
- it’s a good model to store naturally-hierarchical data ,example file system on computer

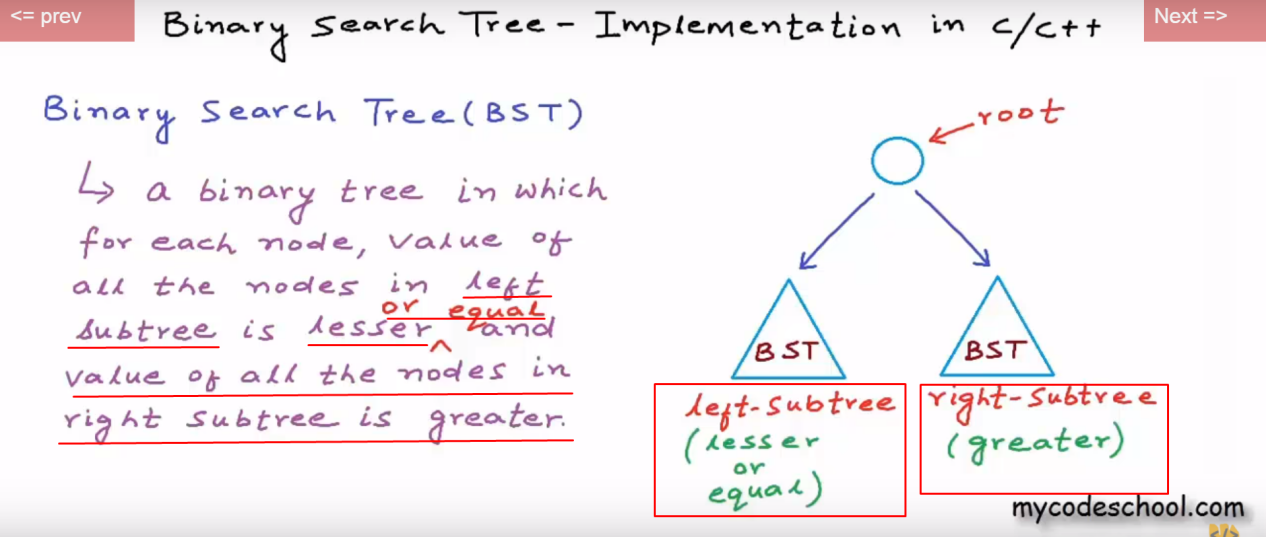
- good for quick search , delete and insertion

**Binary tree search compare to other data structure search**

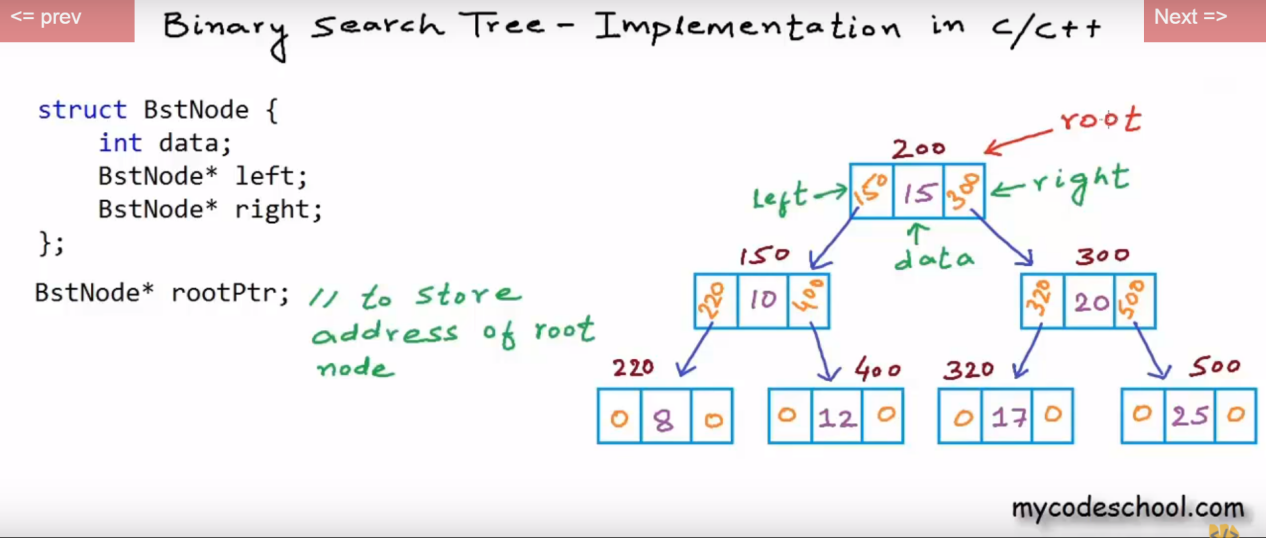


**Binary tree search using binary search(见算法介绍)**





**Implement binary search tree and set nodes**



**Depth first Traversal纵向遍历(相对breath first横向遍历同级节点) : Preorder , Inorder , Postorder**

Visit order

Preorder :

**<root>** <left> <right>

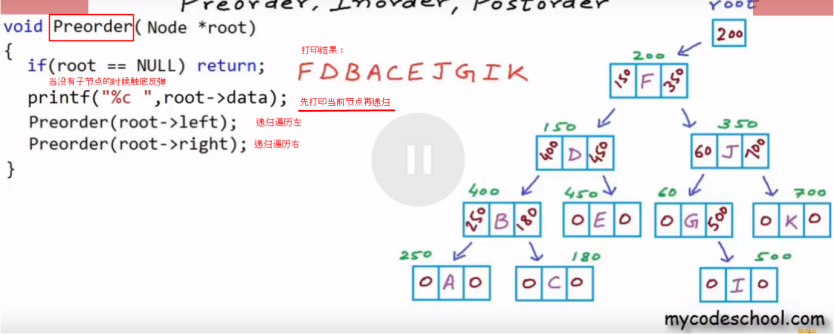
Inorder :

<left> **<root>** <right>

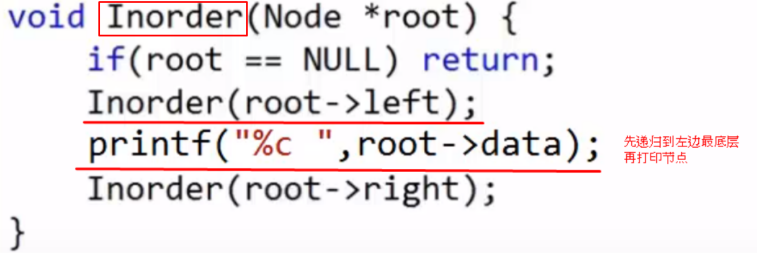
Postorder :

<left> <right> **<root>**

Preorder



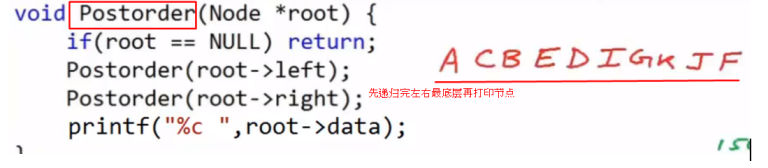
Inorder代码基本同理，区别在于先递归到左边最底层再打印节点



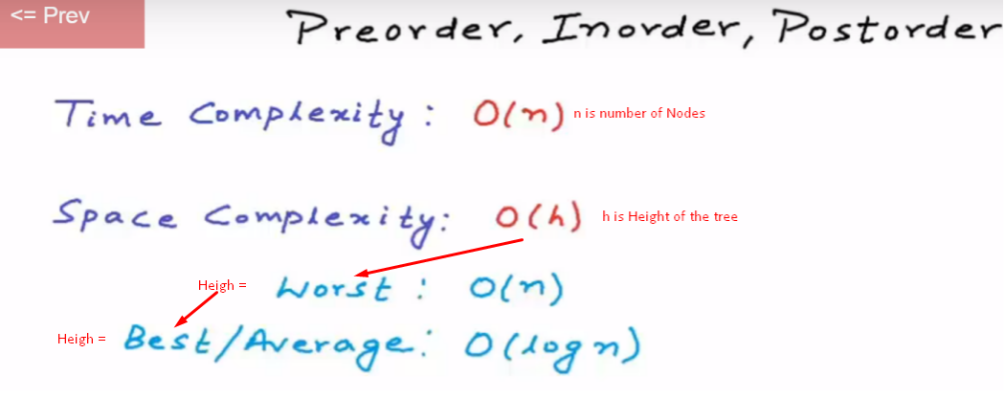
打印结果 :



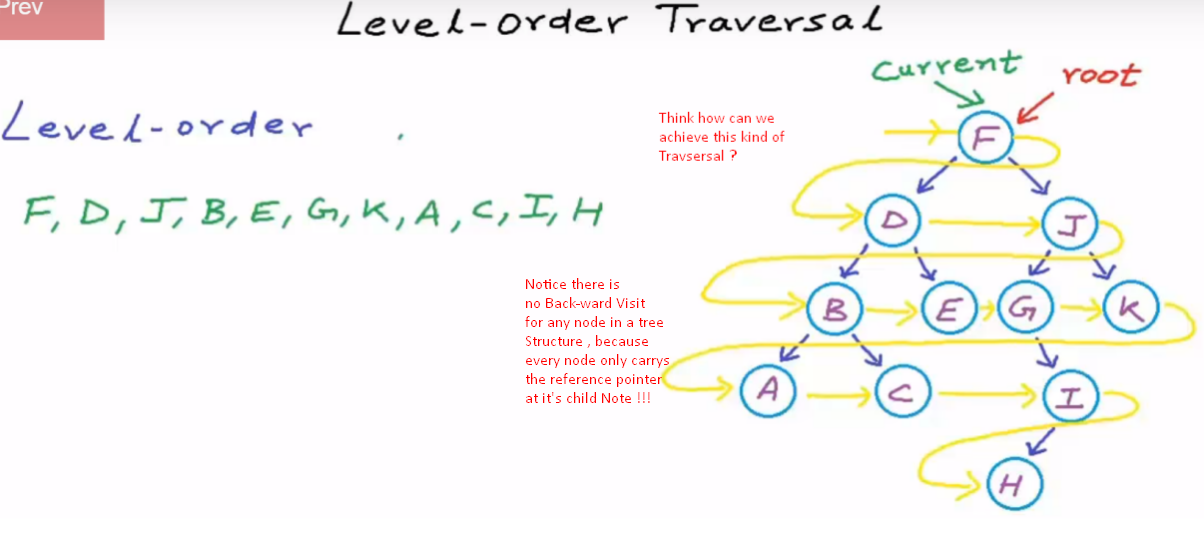
Postorder 代码则是先递归完毕左右最底层再打印节点



Evaluate Time and space Complexity of Preorder Inorder and Postorder Traversal

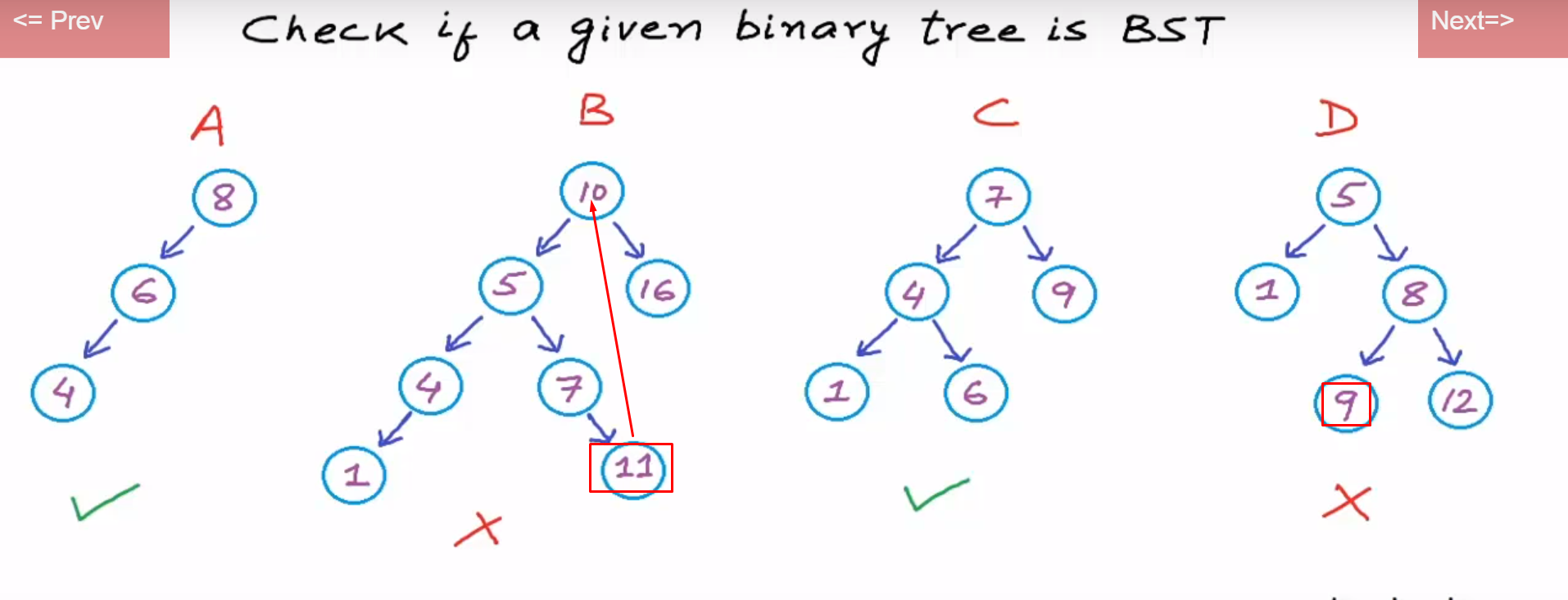


**Level Order Traversal (Same as Breath first Traversal横向遍历)**



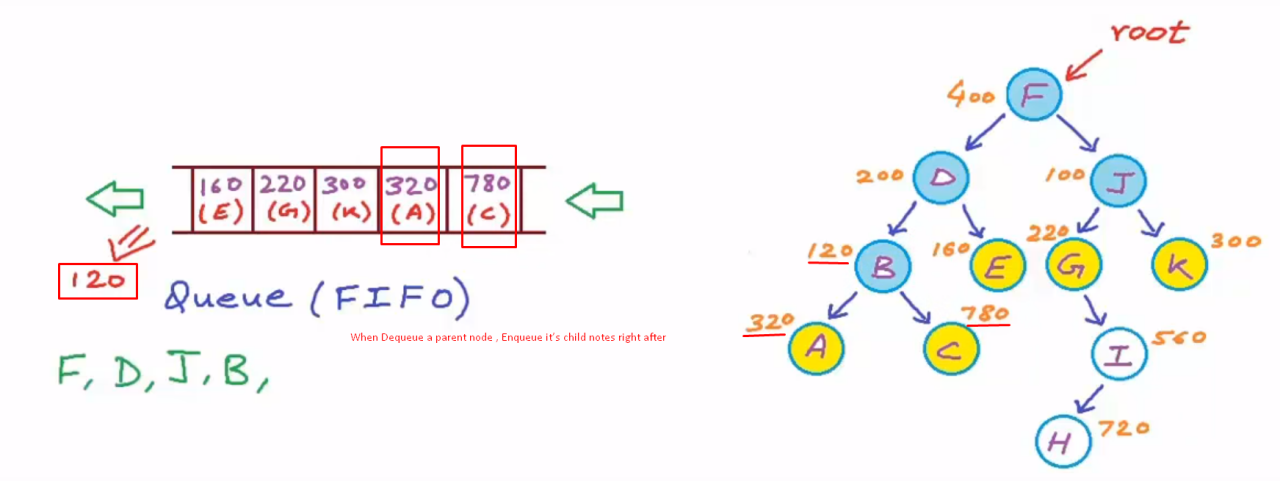
- Notice there is no Back-ward Visit for any node in a tree structure , because every node only carrys the reference pointer at it’s child Note

- a binary tree in which for each node *,* value of all the nodes in left subtree is lesser or equal and value of all the nodes in right subtree is greater.

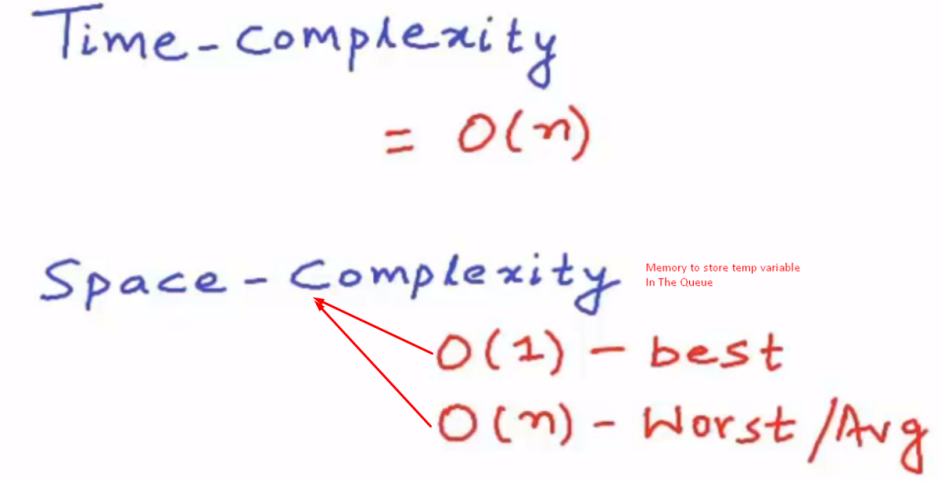


- and tree level starts at 0

So solution is using Queue data structure (FIFO) , When Dequeue a parent node , Enqueue it’s child notes right after



Evaluate Time and Space complexity of Level-order/breadth first Traversal



**Computing the tree height**

using post-order scan : scanning from <left> <right> <root>

代码: //用stack草图理解解释，见草稿本

height( t ): //t为root节点个数,height()为t节点所在高度

{

if(t==null)

heightval=-1;

else

{

heightLeft=height(t.left) //回调, .left理解为左边的节点

heightRight=height(t.right) //回调, .right理解为右边的节点

heightval=1+(height>height ? heightLeft : heightRight); //判断取值

//以上三行是post-order scan的体现 : scanning from <left> <right> <root>

}

return heightval;

}