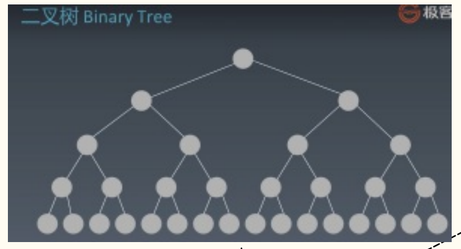
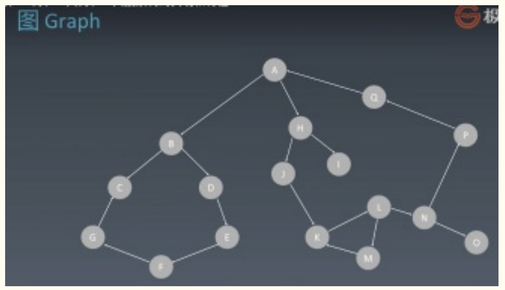




树和图的区别:
树没有环 节点有环的情况称为图

https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

树&图

Linked List 是特殊化的 Tree
Tree 是特殊化的 Graph

```java
public class TreeNode {
  public int val;
  public TreeNode left, right;
  public TreeNode(int val) {
    this.val = val;
    this.left = null;
    this.right = null;
  }
}
```

二叉树遍历

遍历方式:
- 递归遍历比较简单
- 循环遍历比较困难: 广度优先

前序遍历(Pre-order) : 根 - 左 - 右
中序遍历(In-order) : 左 - 根 - 右
后序遍历(Post-order) : 左 - 右 - 根

二叉搜索树

普通的没有任何状态的树，查找节点需要遍历整棵树

一般来说我们会把树变得有序，这样才有树存在的意义，否则就跟普通链表一样了

二叉搜索树又称：有序二叉树(Ordered Binary Tree)、排序二叉树(Sorted Binary Tree)，是指一颗空树或具有下列性质的二叉树:

1.左子树上**所有节点**的值均小于它的根节点的值

2.右子树上**所有节点**的值均大于它的根节点的值

3.以此类推：左、右子树叶分别为二叉查找树（这就是 重复性！）

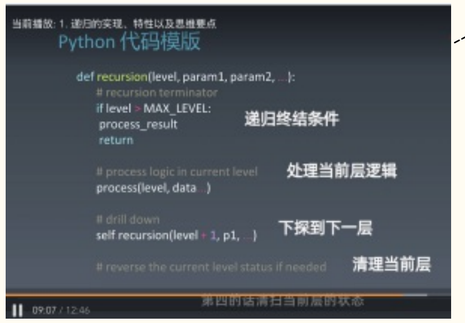
中序遍历：是升序遍历

常见操作

极端情况，树是一根棍子，退化为了链表，所以查找是 O(n)

1.查询 O(logn)
2.插入新节点(创建) O(logn)
3.删除 O(logn)

[二叉搜索树 Demo]



递归

树一般都是用递归，它的结点和树本身它的数据结构都是用递归来定义的

递归类似于循环，通过函数来进行的循环

递归像盗梦空间，向下进入不同的梦境中，向上又回到原来的一层
每一层的环境和周围的人都是一份拷贝，主角等几个人穿越不同层级的梦境（发生和携带变化）

对没有任何重复性的问题 那么它的复杂度是客观存在的 只能有多少复杂代码 就写多少复杂代码 否则可以递归完成

找重复性可以用高中的数学归纳法来列出 n=1, n=2, n=3… 然后分析
mutual exclusive, complete exhaustive

前中后序代码模版

```python
def preorder(self, root):
  if root:
    self.traverse_path.append(root.val)
    self.preorder(root.left)
    self.preorder(root.right)


def inorder(self, root):
  if root:
    self.inorder(root.left)
    self.traverse_path.append(root.val)
    self.inorder(root.right)


def postorder(self, root):
  if root:
    self.postorder(root.left)
    self.postorder(root.right)
    self.traverse_path.append(root.val)
```