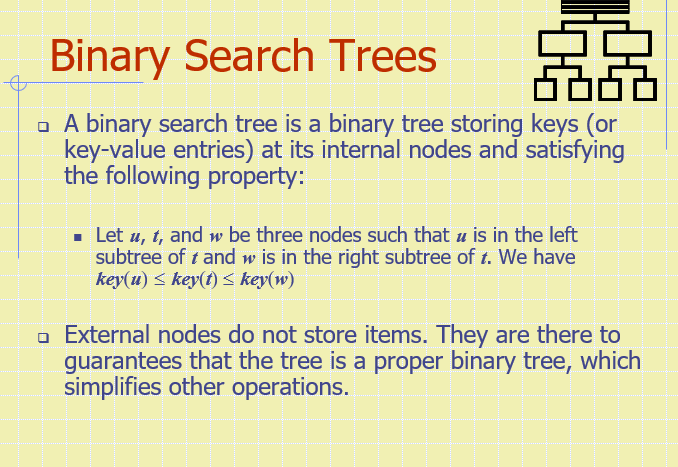
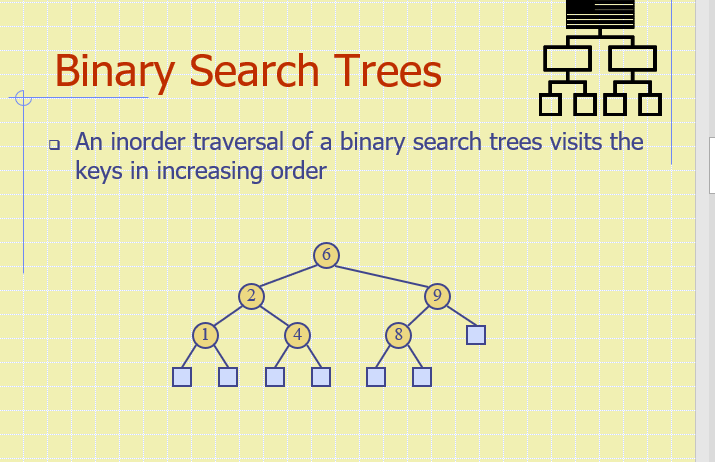
binary search tree就是一个binary tree存储着keys(或key-value entries)。

满足以下性质，utw是三个node，u是left subtree，t是顶点，w是right subtree，那么key (u)<=keyt<=keyw

换句话说，左边的永远小于等于中间的永远小于等于右边的

external node不存储项目。它们可以保证树是一个合适的二叉树，从而简化了其他操作。

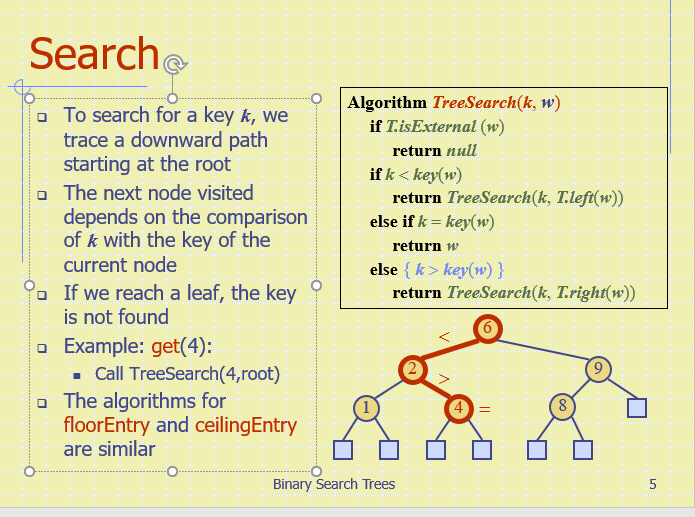




以inorrder traversal来遍历 binary search tree, 会visit key以增加Order

inorder,左node，自己，右Node

search

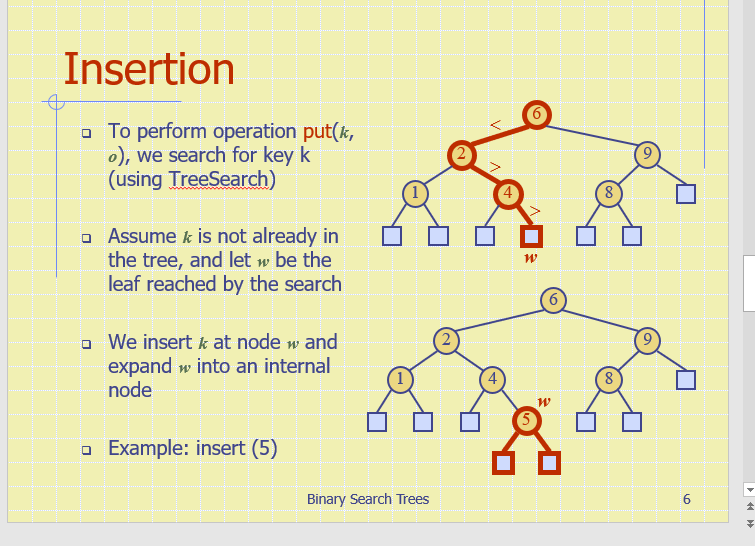


要搜索key k，我们要从root开始从上往下查，

从上到下时我们要看左Node右Node大还是小

如果我们到了leaf，就证明Key没找到

iunsertion



为了实现Put，我们要使用search

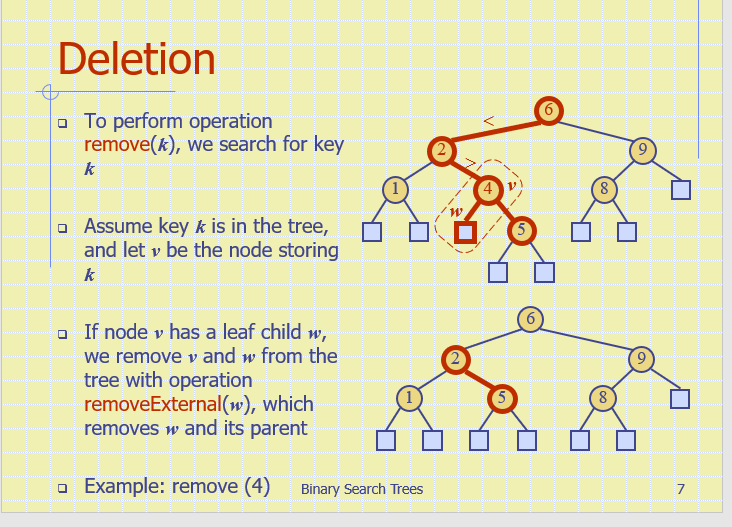
假设K不存在，并且我们已经到了空白leaf w

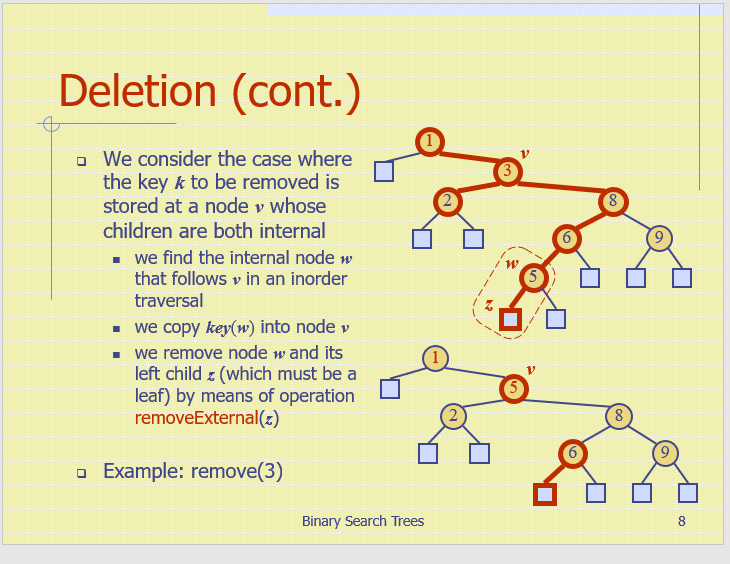
我们把k插入w，让后把w设置成Internal node

deletion,如果 要进行remove k,我们必须先搜寻，

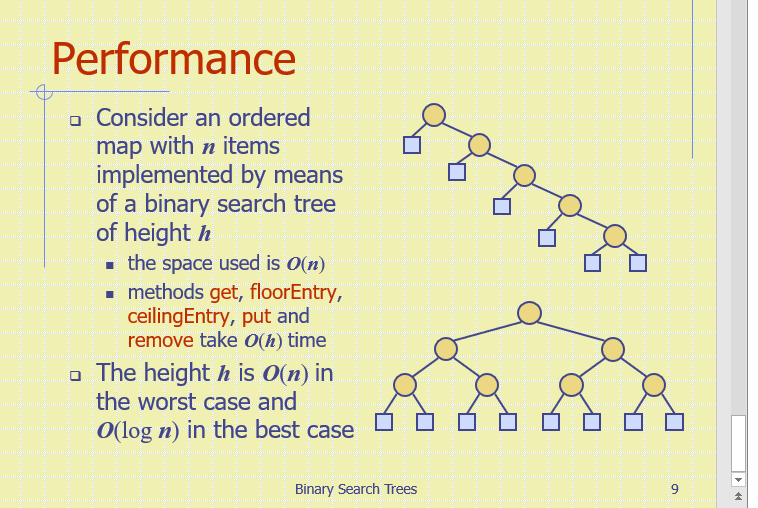
1.

如果K在tree中:假设v是·他现在位置，如果v有一个左leafw，我们把v和·w都删除，





如果我们要remove key k,k存储在node v，而他的两个都是Internal，我们找到follow v的那个w 用Inorder traversal, 注意注意是w follow v，所以是右边左下角， 我们把wcopy到v内，删除v，然后把w与他的左child删除，



space 是O(N),其他的都是O（h）,h是Onworst case,ologn best case