* 学习目标

  * 能够理解数据结构之树

  * 能够理解数据结构之二叉树和遍历方式

  * 能够理解广度优先遍历和深度优先遍历

  * 能够理解什么是二叉查找树（BST）

  * 能够理解数据结构之红黑树

---

  * 回顾

    * 自定义LinkedList：双向列表，队列，栈

    * 自定义ArrayList：动态扩容，数组复制

  * 能够理解数据结构之树

    * 树英文名字：Tree

    * 用来模拟具有树状结构性质的数据集合

    * 一棵倒挂的树:根朝上，叶朝下

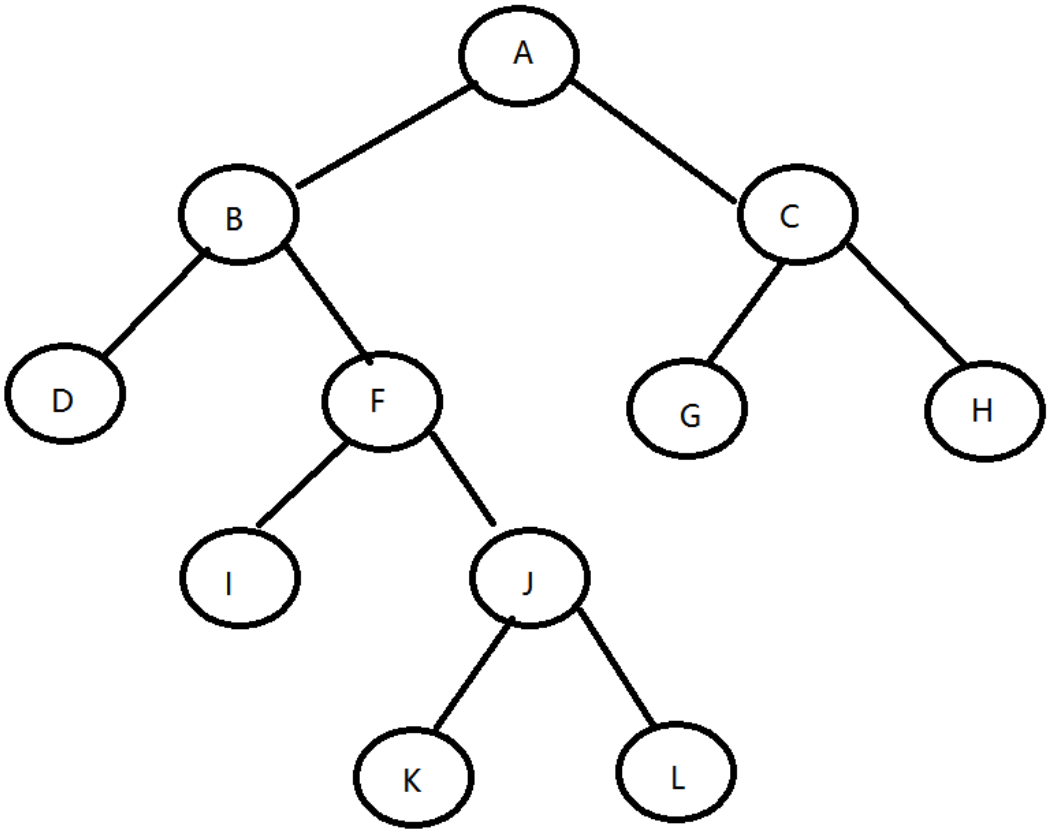
    * 树的特点：

      * 每个节点有零个或多个子节点

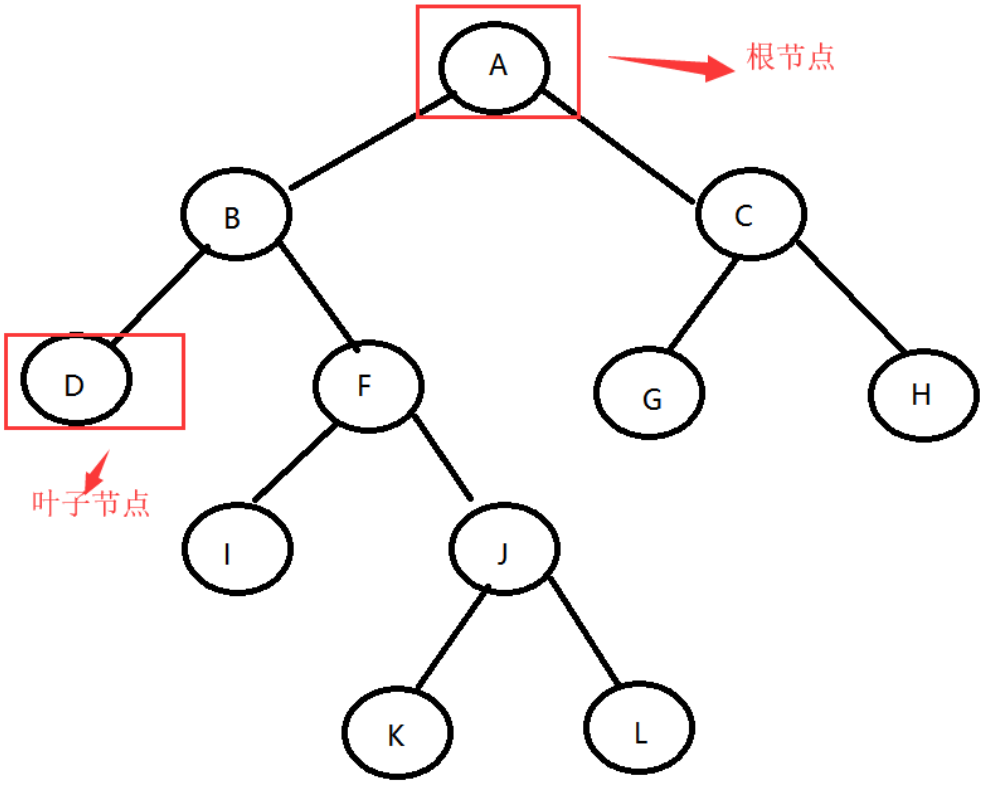
      * 没有父节点的节点称为根节点

      * 每一个非根节点有且只有一个父节点

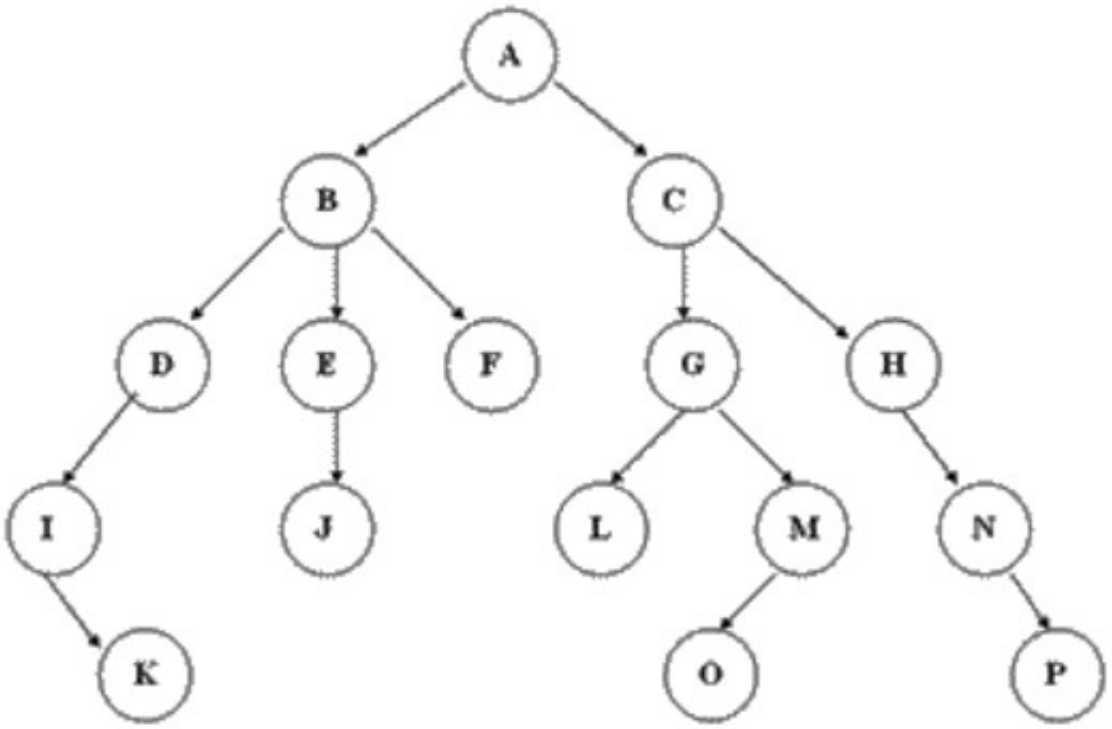
      * 除了根节点外，每个子节点可以分为多个不相交的子树



* 根节点和叶子节点



* 树的相关概念



* 节点的度：一个节点含有的子树的个数称为该节点的度。（上图 A->2 B->3 J->0）

* 树的度:一棵树中，最大的节点的度称为树的度（上图 B节点 3个度 最大）

* 叶节点或终端节点：度为零的节点（上图 K O P）

* 父亲节点或父节点：若一个节点含有子节点，则这个节点称为其子节点的父节点（上图A是BC的父节点 B是DEF的父节点）

* 孩子节点或子节点：一个节点含有的子树的根节点称为该节点的子节点（上图BC是A的子节点 DEF是B的子节点）

* 兄弟节点：具有相同父节点的节点互称为兄弟节点（上图BC DEF LM是相互的兄弟节点 ）

* 节点的层次：从根开始定义起，根为第1层，根的子节点为第2层，以此类推（上图A为第1层 BC第2层 DEFGH第三层 ...）；

* 树的高度或深度 ：树中节点的最大层次（上图为5层）

* 能够理解数据结构之二叉树和遍历方式

* 每个节点最多含有两个子树的树称为二叉树

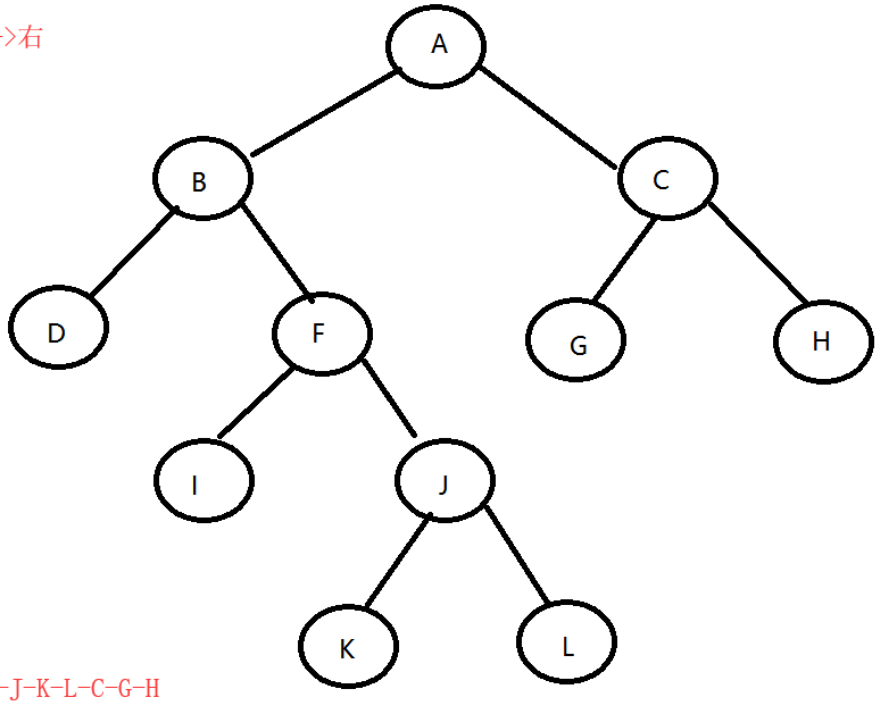
* 遍历方式：

 * 先序遍历：先根节点->遍历左子树->遍历右子树

 * 中序遍历：遍历左子树->根节点->遍历右子树

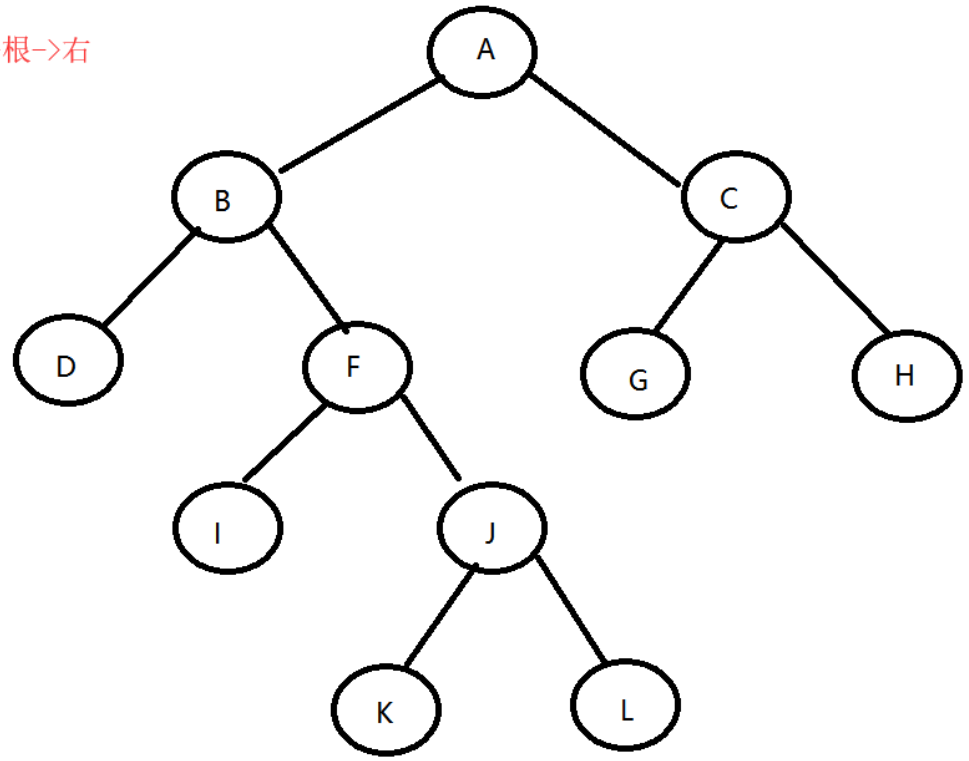
 * 后序遍历：遍历左子树->遍历右子树->根节点

先序：根->左->右



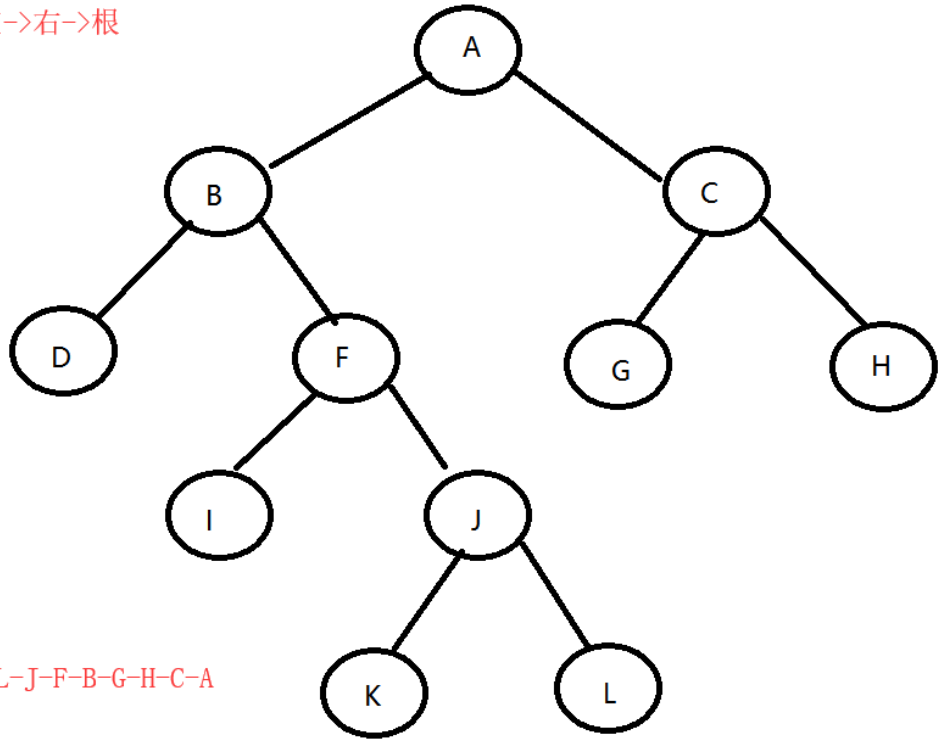
A-B-D-F-I-J-K-L-C-G-H

中序：左->根->右



D-B-I-F-K-J-L-A-G-C-H

后序：左->右->根



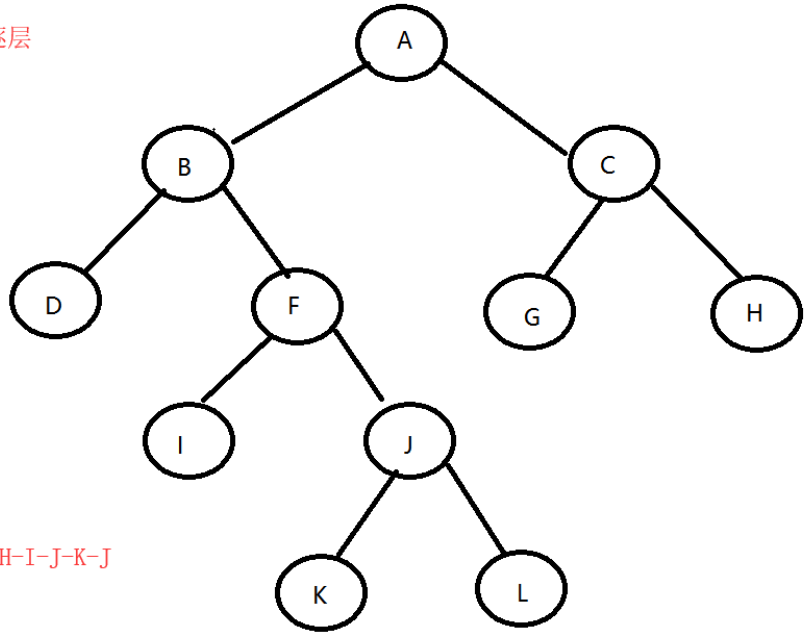
D-I-K-L-J-F-B-G-H-C-A

* 能够理解广度优先遍历和深度优先遍历

* 广度优先遍历：每一层节点依次访问，访问完一层进入下一层，而且每个节点只能访问一次。

* 假设每层节点从左到右访问

广度优先遍历：逐层

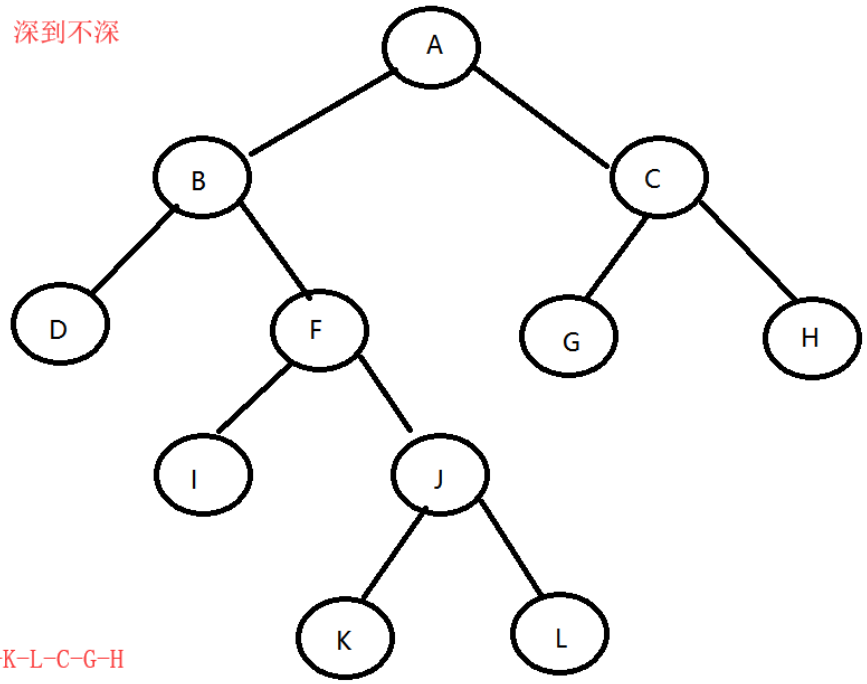


A-B-C-D-F-G-H-I-J-K-J

* 深度优先遍历： 对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次。

* 假设先走子节点的的左侧

深度优先遍历：深到不深



A-B-D-F-I-J-K-L-C-G-H

* 能够理解什么是二叉查找树（BST）

* 二叉查找树英文名字：Binary Search Tree

* 也叫做二叉搜索树，有序二叉树，排序二叉树

* 特点

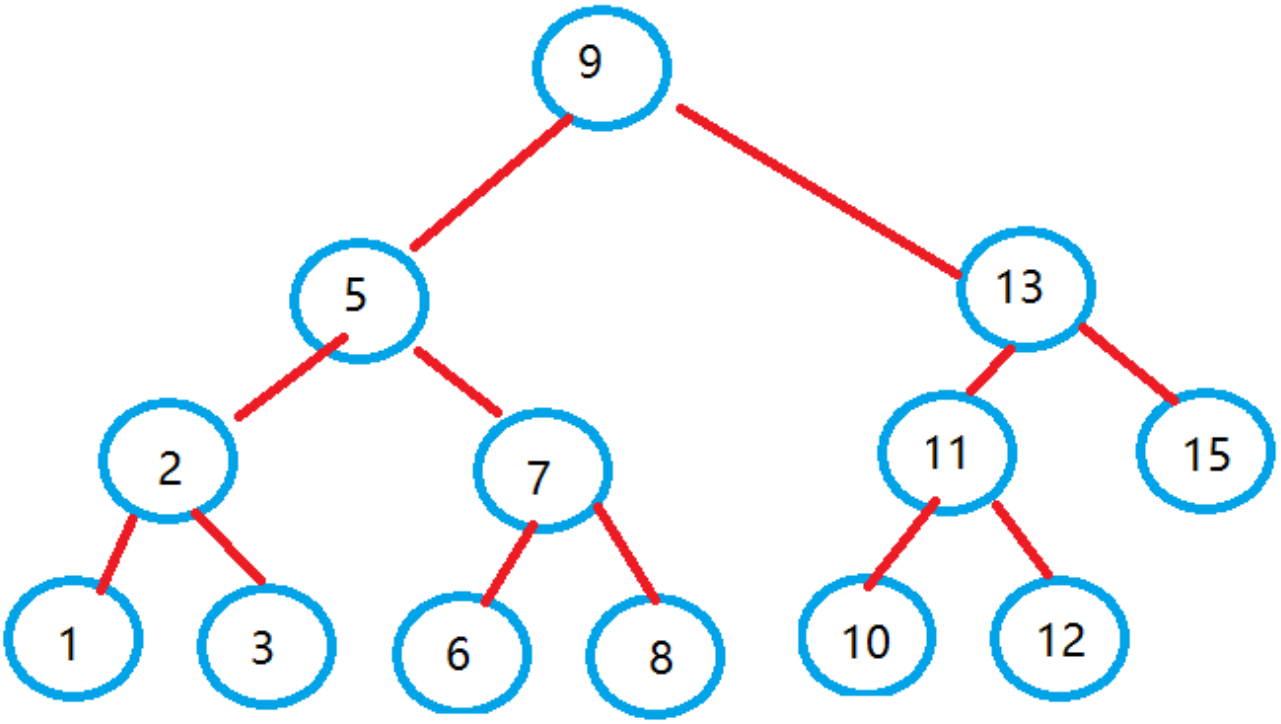
    * 若任意节点的左子树不空，则左子树上所有节点的值均小于它的根节点的值

    * 若任意节点的右子树不空，则右子树上所有节点的值均大于它的根节点的值

    * 任意节点的左、右子树也分别为二叉查找树

    * 典型的二叉查找树

        * （1 2 3 5 6 7 8 9 10 11 12 13 15）

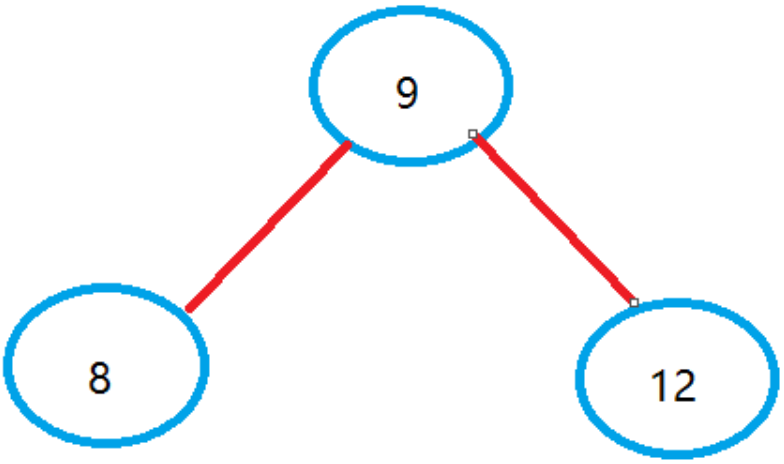


* 查找 3 和 10 的例子

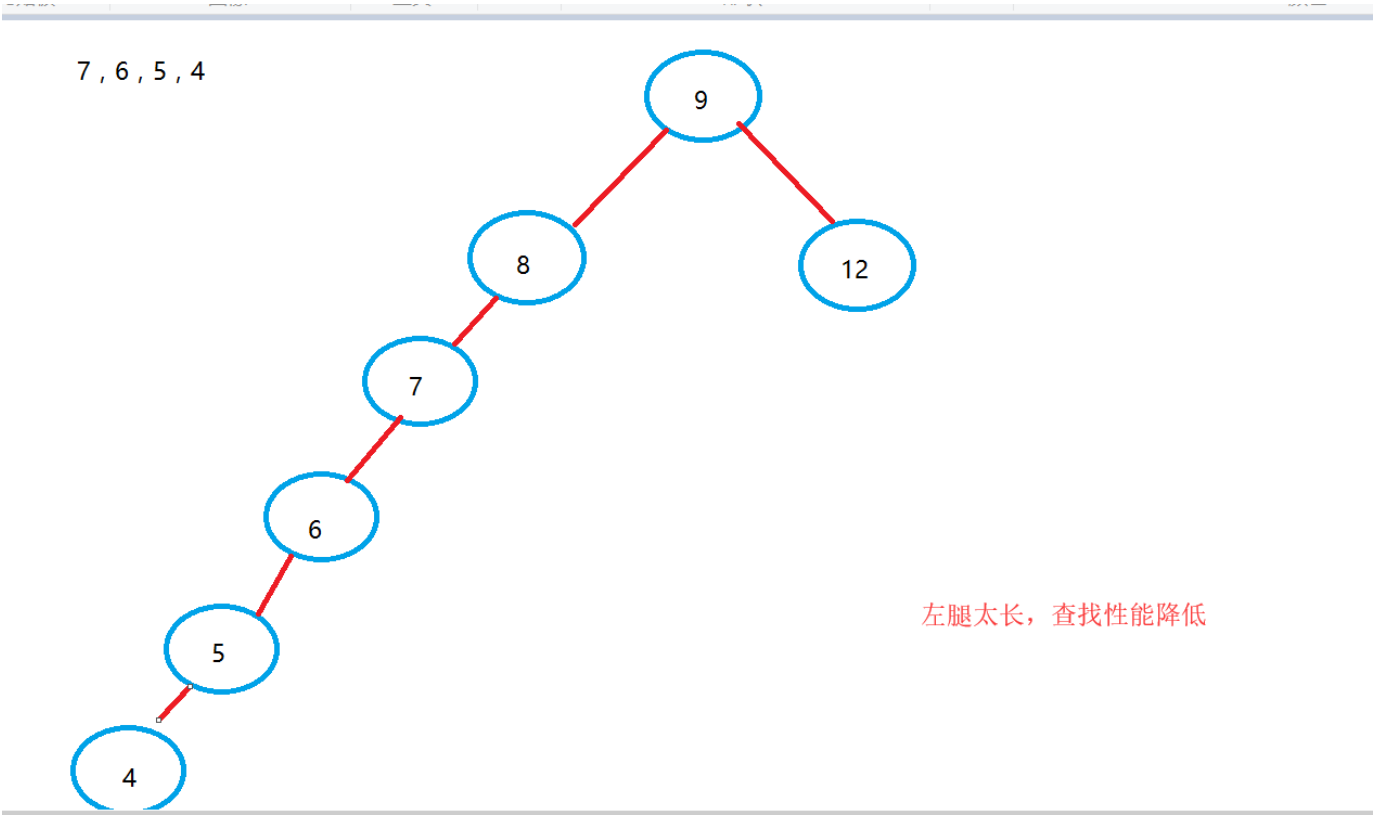
    * 极端例子演示

        * 假如初始化的树是这样的

.



* 再加入 7，6，5，4



* 怎么解决这个问题呢？可以使用红黑树存储。

* 能够理解数据结构之红黑树

* 根结点是黑的。（根黑）

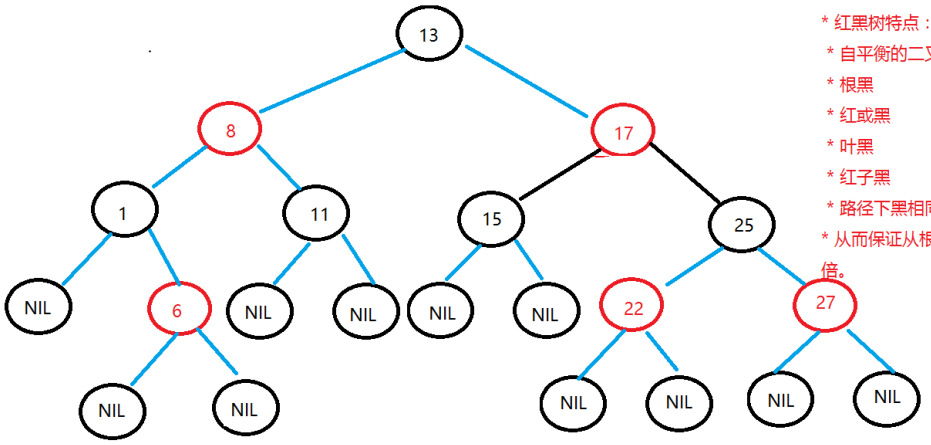
* 每个结点要么是红的要么是黑的。（红或黑）

* 每个叶结点（叶结点即指树尾端NIL指针或NULL结点）都是黑的。（叶黑）

* 如果一个结点是红的，那么它的两个儿子都是黑的。（红子黑）

* 对于任意结点而言，其到叶结点树尾端NIL指针的每条路径都包含相同数目的黑结点。（路径下黑相同）

* 以上规则保证从根节点到叶子节点，最长的路径不超过最短路径的2倍



* 红黑树特点：
* 自平衡的二叉查找树
* 根黑
* 红或黑
* 叶黑
* 红子黑
* 路径下黑相同
* 从而保证从根节点到叶子节点，最长的路径不超过最短路径的2倍。