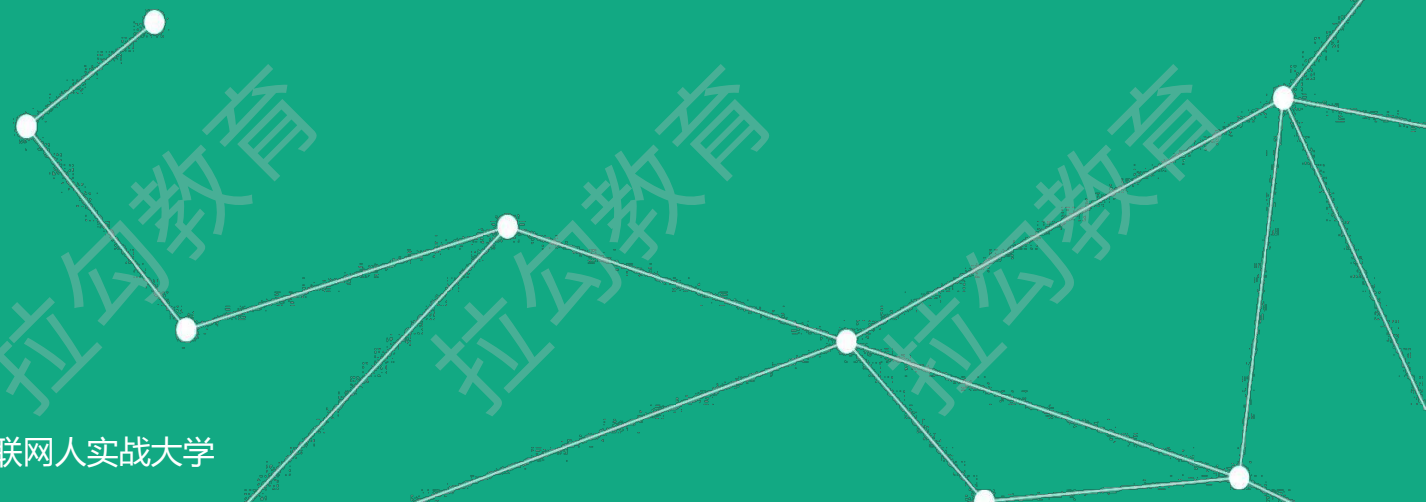
课时 11

平衡树的性能优化

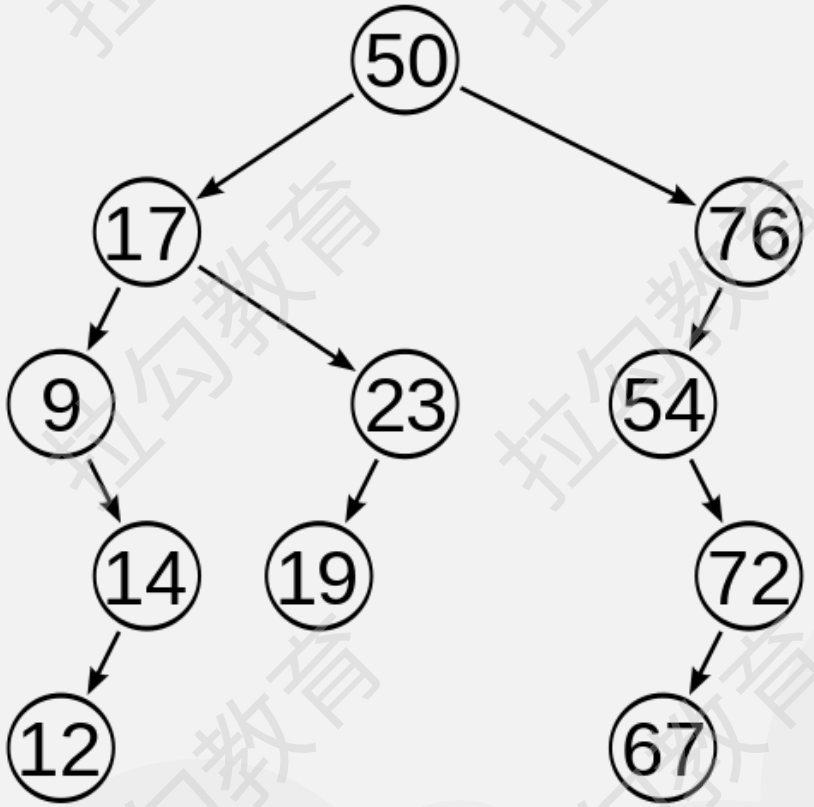
1. 二叉查找树 （BST）
2. 平衡树
3. Log-Structured 结构



二叉查找树（Binary Search Tree，简称 BST）

此图是一颗以 50 节点为根的二叉查找树，规范的来说：

- 二叉查找树是一棵二叉树，也就是说每一个节点至多有 2 个孩子，也就是 2 个子树
- 二叉查找树的任意一个节点都比它的左子树所有节点大，同时比右子树所有节点小



二叉查找树（Binary Search Tree，简称 BST）

为什么二叉查找树定义奇怪的特性，究竟有什么用？

其实都是为了方便搜索节点！

二叉查找树（Binary Search Tree，简称 BST）

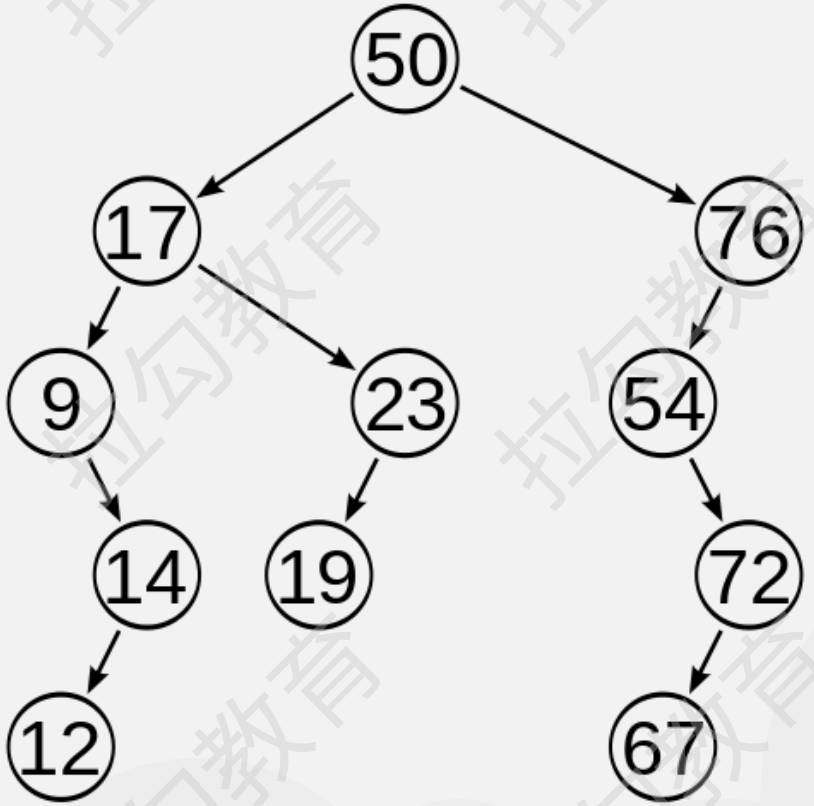
因为已经知道了所有左子树的节点都小于根节点

所有右子树的节点都大于根节点

当我们需要查找一个节点的时候，如果这个节点小于根，那我们肯定是去左子树中继续查找，因为它不可能出现在右子树中了，右子树的所有节点必须大于根

反过来说，如果想要查找的值大于根呢？

我们就只需要去右子树中查找即可



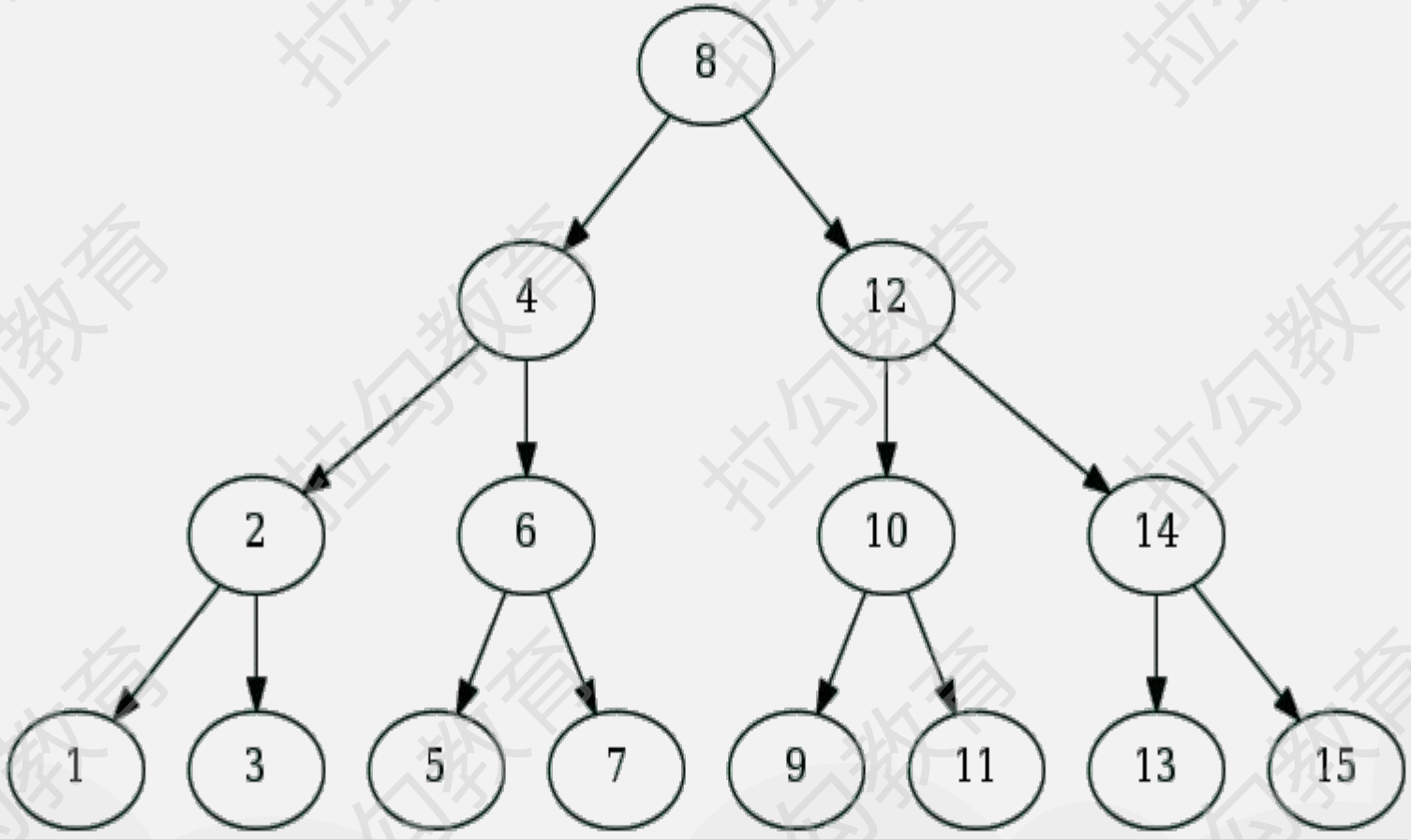
二叉查找树（Binary Search Tree，简称 BST）

```
TreeNode* Search(TreeNode* root, int key) {
    if (root == nullptr || root->key == key)
{
        return root;
    }

    if (root->key < key) {
      return Search(root->right, key);
    }

    return Search(root->left, key);
}
```

二叉查找树（Binary Search Tree，简称 BST）

高度是 O(log n)

n 是这棵树的节点数量

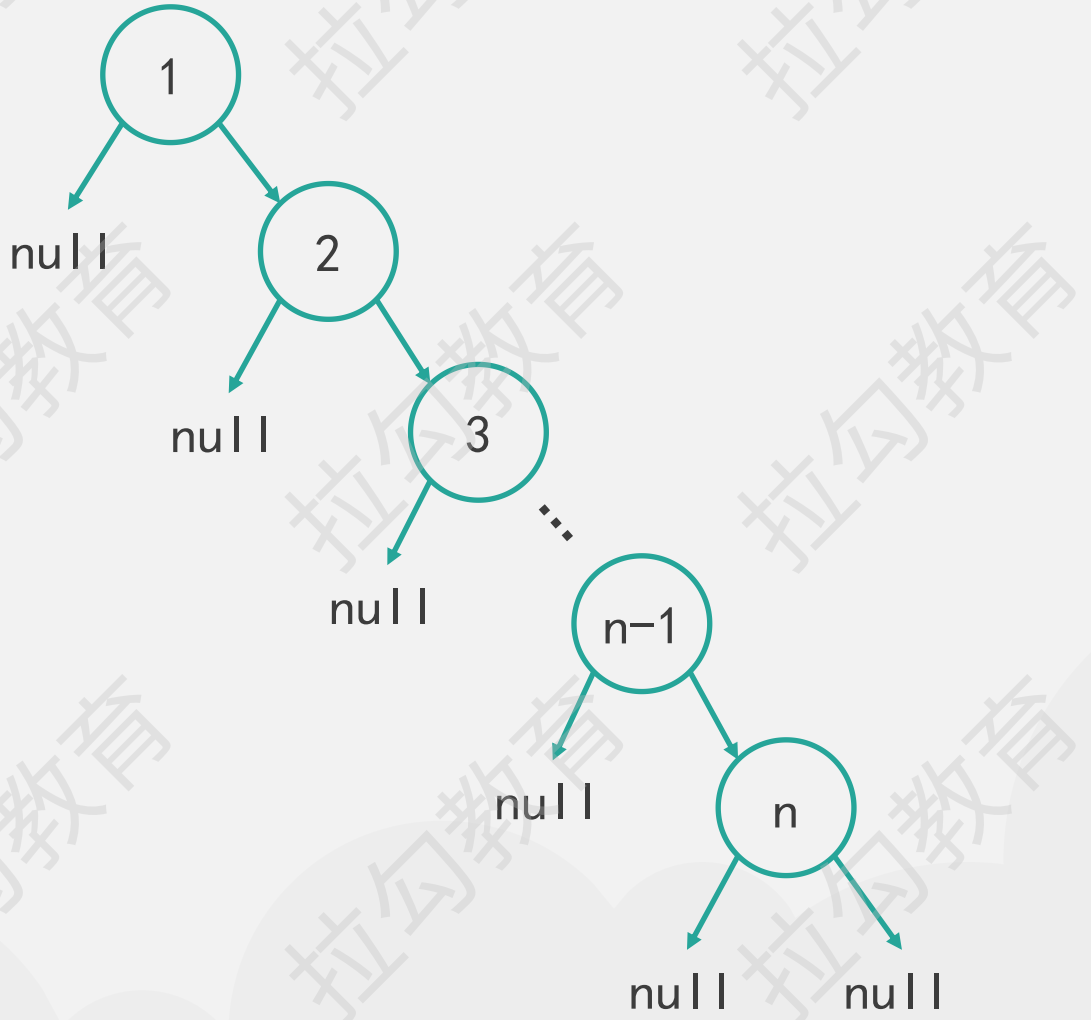


二叉查找树（Binary Search Tree，简称 BST）

这样的二叉树退化成了一个一维链表

最坏情况是需要从第一个节点查找到第 n 个节点

时间复杂度就成了 O(n)



平衡树

平衡树，就是说一棵树的数据结构能够维护好自己的高度和形状

让形状保持平衡的同时，高度也会得到控制

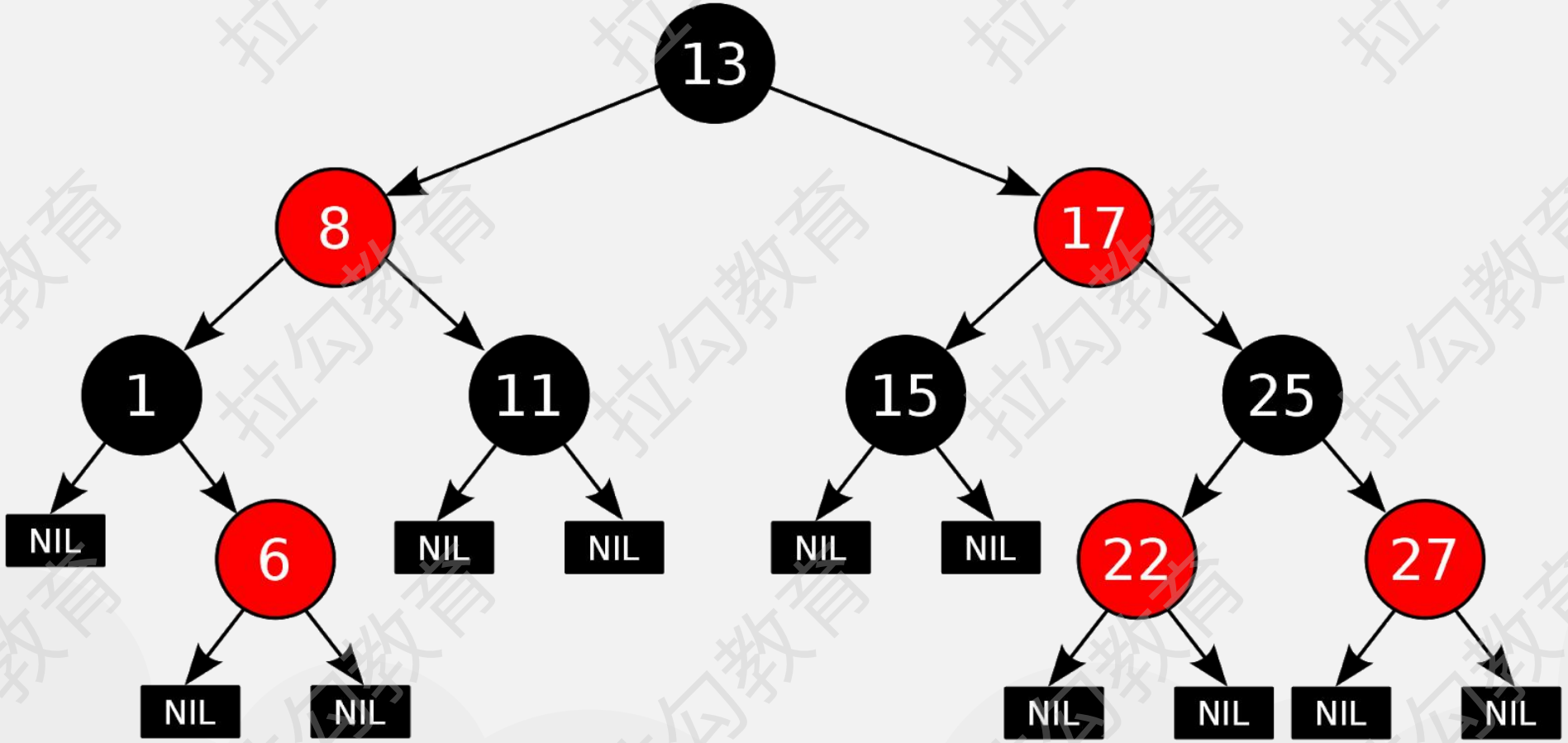
定义比较粗糙，属于一个愿景

平衡树

红黑树被定义成：

- 一棵二叉树，每一个节点要么是红色，要么是黑色
- 根节点一定是黑色
- 红节点不能有红孩子
- 每条从根节点到底部的路径都经过同样数量的黑节点

平衡树

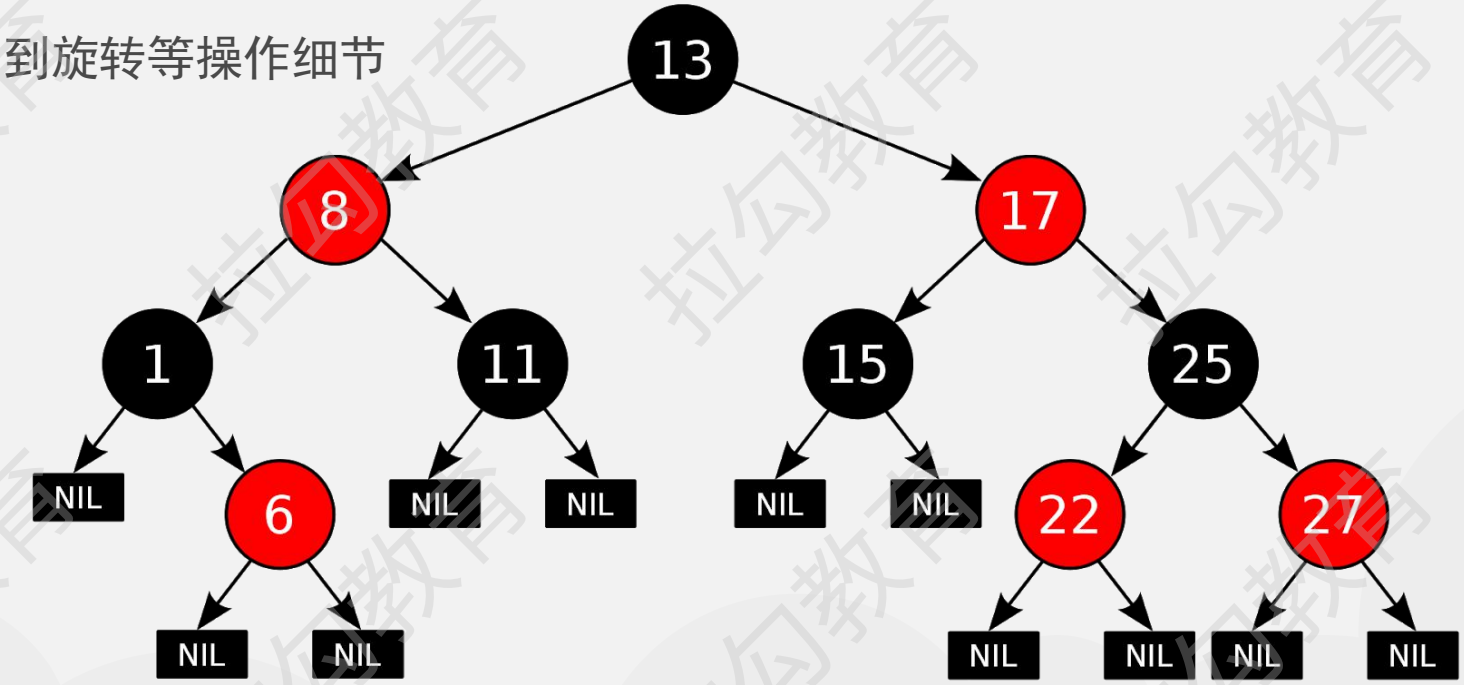


平衡树

满足这些定义的红黑树可以被数学证明树的高度为 0(log n)

要实现一棵红黑树是非常难的

其中有许多节点插入 / 删除需要用到旋转等操作细节



平衡树

B 树的每个节点可以存储多个数值，但是要求：

- 所有叶子节点的深度一样
- 非叶子节点只能存储 b ― 1 到 2b ― 1 个值
- 根节点最多存储 2b ― 1 个值

红黑树可以被理解成 order 为 4 的一种特殊 B 树，称为 2-3-4 树

之所以称为 2-3-4 树 是因为每个有子树的节点都只可能有两个，或者 3 个，或者 4 个子节点

类似想法的平衡树实现还有很多，比如 AVL 树，由它的发明者们的名字首字母命名

分别是 Adelson-Velsky-Landis，它的定义很简单，任意一个节点的左右子树高度最多相差 1

Log-Structured 结构

在计算机存储数据结构的发展中

Log-Structured 结构的诞生为许多文件系统或者是数据库打下了坚实的基础

例如

Google 的三驾马车之一，Bigtable 文件系统的底层存储数据结构采用的就是 Log-Structured 结构

还有大家所熟知的 MongoDB 和 HBase 这类的 NoSQL 数据库

它们的底层存储数据结构其实也是 Log-Structured 结构

Log-Structured 结构

假设一个视频网站需要一个统计视频观看次数的功能，如果给你来设计的话，会采用哪种数据结构呢？

可以运用哈希表这个数据结构，以视频的 URL 作为键、观看次数作为值，保存在哈希表里面所有保存在哈希表里面的初始值都为 0，表示并无任何人观看，而每次有人观看了一个视频之后就将这个视频所对应的值取出然后加 1

Log-Structured 结构

这种操作的瓶颈其实是在更新操作，也就是写操作上

那有没有方法可以不用顾及写操作的高并发问题，同时也可以最终获得一个准确的结果呢？

答案就是使用 Log-Structured 结构

Log-Structured 结构

Log-Structured 结构，有时候也会被称作是 Append-only Sequence of Data

因为所有的写操作都会不停地添加进这个数据结构中，而不会更新原来已有的值

这也是 Log-Structured 结构的一大特性

Log-Structured 结构

假设现在网站总共有三个视频，URL 分别就是 A、B 和 C

A: 1	A: 1	C: 1	A: 1	A: 1	B: 1	C: 1	A: 1
B: 1	B: 1	C: 1	C: 1	A: 1	C: 1	A: 1	A: 1

Log-Structured 结构

Log-Structured 结构其实在应用里会有很多的问题

比如说，一个数组不可能在内存中无限地增长下去，我们要如何处理呢？

如果每次想要知道结果，就必须遍历一遍这样的数组，时间复杂度会非常高