Code Project 2——实验报告

PB14204015 张晗

1. **问题介绍：红黑树数据结构**

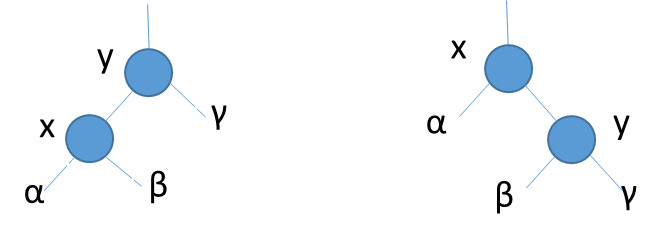
红黑树是一棵二叉搜索树，每一个节点多了一个颜色属性，颜色可以是红色或者黑色。红黑树由于其特殊的机制使得没有一条路径的长度会比其他路径长出两倍。

* 1. 红黑树的性质：
     1. 每个节点是红色的或者是黑色的
     2. 根节点是黑色的
     3. 每个叶节点（NIL）是黑色的
     4. 如果一个节点是红色的，则它的两个子节点是黑色的
     5. 对每个节点，从该节点到其所有后代叶节点的简单路径上，均包含相同数目的黑色节点

1. **红黑树性质的维护**
   1. 旋转操作

旋转操作是一种能满足二叉搜索树性质的搜索树局部操作。在下图中：

左旋操作：右旋操作：



* 1. 插入操作

首先进行一般的二叉搜索树的插入操作，并给新增节点颜色设置为红色。

这样红黑树的性质之4可能被破坏。

于是要运行insert-fix 操作。分以下几种情况：

* + 1. 新节点的叔节点是红色，则父节点和叔节点赋值黑色，祖父节点设为红色，

这样仍有可能违反性质4，新节点的指针上移。

* + 1. 叔节点是黑色，新节点是右子数，则父节点为轴左旋转化为case3
    2. 叔节点是黑色，新节点是左子数，则以祖父节点为轴右旋，问题解决
  1. 扩展红黑树的性质维护——区间数

区间数是扩展的红黑树，节点多了一个max属性，key为区间低端点。max指的是以该节点为跟的子树的区间上限。为了维护这个性质，我们采用递归的方法。

x.max = max(x.high, x.left.max, x.right.max)。

1. **红黑树的操作举例**
   1. 求最大

沿着根节点到叶节点一直向右找，直到遇到哨兵节点，这个节点是最大

* 1. 求最小

沿着根节点到叶节点一直向左找，直到遇到哨兵节点，这个节点最小

* 1. 求前驱，后继

以后继为例，前驱算法思想相同

分情况讨论：

如果节点的右子树非空，则求右子树的最小值，即是后继。

如果节点右子树为空，则沿着路径向上寻找，直到某个节点是其父节点的左子树。

* 1. 遍历

遍历采用递归算法，分为前序遍历，中序遍历和后序遍历。

不同点在于访问当前节点的顺序不同。

1. **编程实现一个红黑树和区间数的demo**

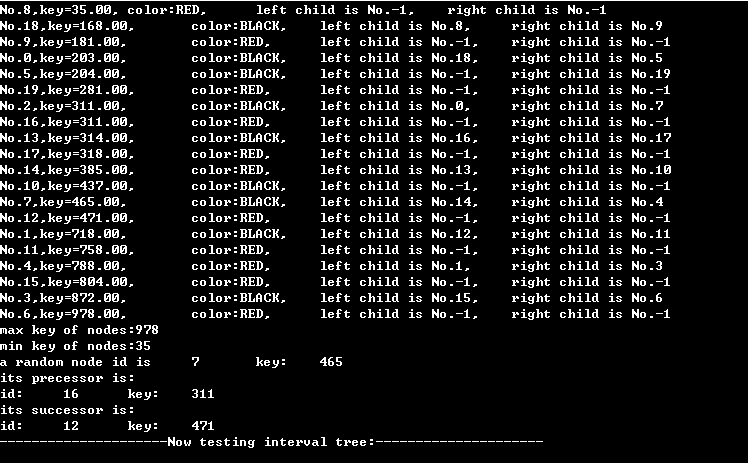
类图：

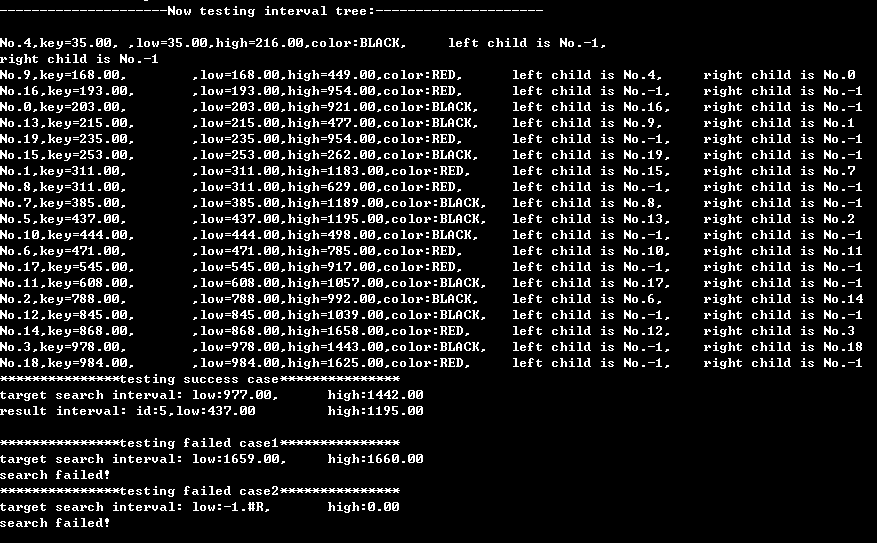


其中rbNode 是红黑树节点，rbTree是普通红黑树，intvTree是区间树。

在主函数里面测试如下几项内容：

1. **测试**
   1. 红黑树general创建功能
   2. 红黑树中序遍历，实现一个有序打印的功能
   3. 区间树创建
   4. 区间树搜索
      1. 搜索成功的case，这个成功的保证是，随机选取树的一个节点，将其高低界限分别减一，创建一个新区间，这个区间必然能被搜到。由于区间树的搜索结果有时候并不唯一，所以并不能保证能搜到我们随机选取的这个节点
      2. 搜索失败的case，失败原因是I’.max<i.min 即所有的节点下限都要高于这个上限
      3. 搜索失败的case，失败原因是I’.min>i.max 即所有的节点上限都要低于这个节点的下限
2. **测试结果**





符合我们的预期。