**区间树上重叠区间的查找算法实验报告**

1. **实验目的**

通过实践，加深理解区间树的性质、特点，熟悉其插入旋转等相关操作，提高编程能力。

1. **实验内容**

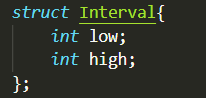
对实验三的红黑树算法进行修改，使其称为一棵区间树，并实现区间树上的重叠区间查找算法。

1. **实验步骤**

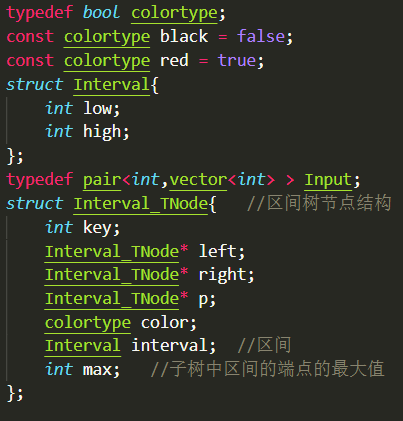
**（1）设计区间树的数据结构**

由于区间树是根据红黑树修改而来的，故仅需在实验三的红黑树数据结构的基础上做一些简单修改即可得到区间树的数据结构：仅需在红黑树结点的数据结构中增加interval域和max域即可得到区间树结点的数据结构。

interval域表示当前结点的区间，使用一个结构体表示：



max域表示以当前结点为根节点的子树所有区间的右端点中的最大值（包括当前结点）。区间树数据结构的其余部分与红黑树相同。

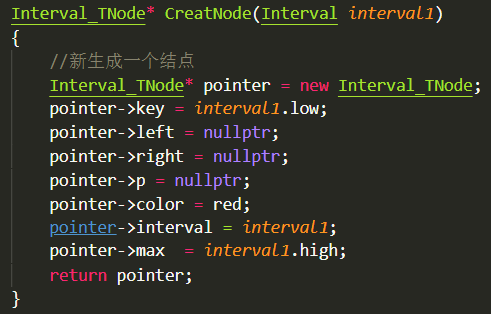




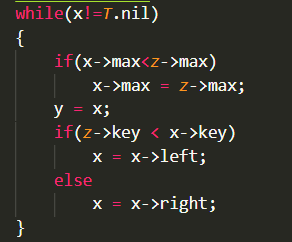
**（2）对结点中新增的域进行维护**

区间树的生成结点、插入结点以及调整操作与红黑树的相应操作均十分相似，只是在各个操作中均需要增加对interval域以及max域的维护操作。

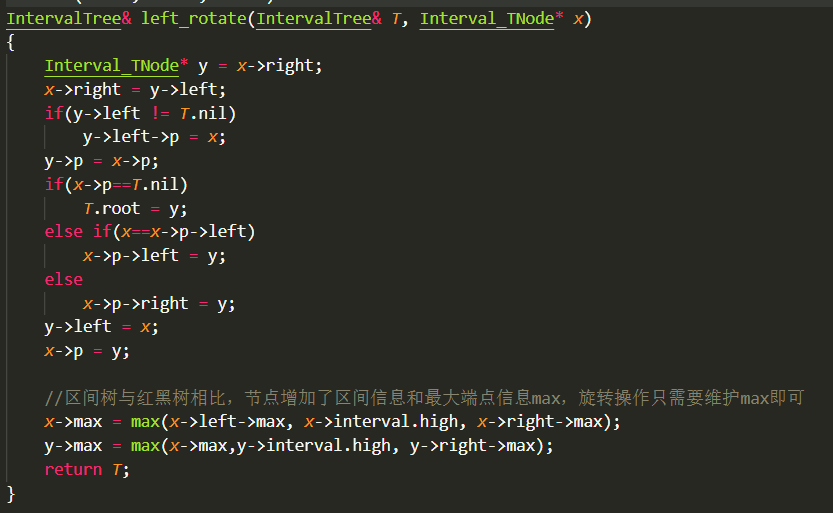
在CreatNode()函数中，要设置新生成结点的interval域，并且将max域设为interval域的右端点。因为总是在插入结点时才会生成新的节点，而节点总是插入到区间树的叶节点位置上，叶节点的左右孩子均为空，故其max域就是自身interval域的右端点。



在插入操作中不需要维护interval域，只需要维护max域，因为插入操作只涉及到结点的移动，所以interval域不会发生变化。可能有两处需要维护max域：一处是在自上而下寻找结点的插入位置时，由于新插入节点的max域可能很大，所以其祖先节点的max域可能也需要发生变化；

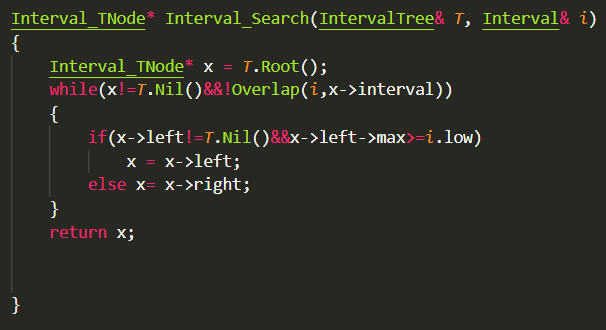


另一处是在调整区间树使其满足红黑树的性质时，由于调整过程中可能需要进行旋转，旋转过后原先父子节点的相对位置会发生变化，此时也需要维护max域。以左旋为例，在对x节点进行左旋时，只需要对x节点和它的右孩子结点y的max进行更新，在更新每个节点的max域时，只需用该节点和它左右孩子节点的max域的最大值进行更新即可。

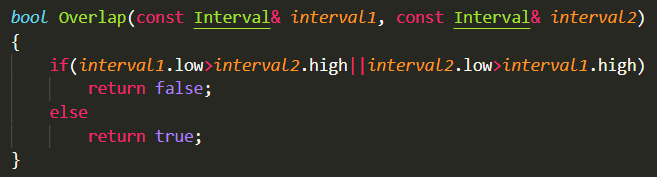


**（3）查找重叠区间**

在区间树T上查找一给定区间i时，从根节点x开始，逐步向下搜索。当找到一个重叠区间或者x指向T.nil时查找过程结束当x指向的节点不是T.Nil节点且节点x的区间与区间i重合时，说明已找到重叠区间，返回的是包含重叠区间的节点，否则说明找不到重叠区间。当x指向的节点不是T.Nil节点且节点x的区间与区间i不重合时，若x节点的左孩子不是T.Nil,并且左孩子的max域大于等于指定区间的左端点，那么令x指向左孩子，转向左子树中查找，否则转向右子树中查找。

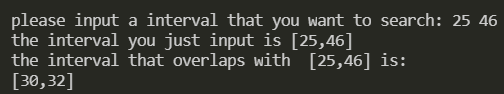


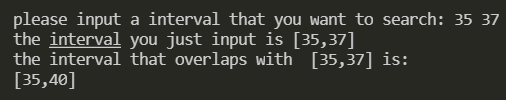
另外，在查找过程中，还需要判断两个给定区间是否重合，此操作通过Overlap()函数实现，由于两个区间interval1、interval2不重合时只存在两种情况，一种是interval1的左端点在interval2的右端点的右边，另一种是一种是interval2的左端点在interval1的右端点的右边，当两个区间的位置关系不满足这两种条件时，可以判断出这两个区间是重叠的。

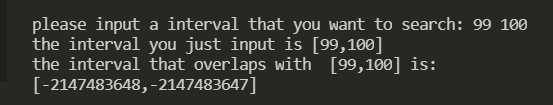


1. **实验结果**

输入三组区间[30,32][35,40]以及[99,100]，测试结果如下：







前两组区间均查找成功，最后一组区间查找失败，返回的是Nil节点的区间[-2147483648, -2147483647]，因为在初始化时将Nil节点的区间设为[INT\_MIN,INTMIN+1]。

**五、实验心得**

区间树的数据结构及操作如果从零写起比较复杂，但是如果参考红黑树的数据结构及操作，只需稍微进行一些改动即可。这启发我们在遇到比较复杂的问题时，可以首先查阅资料看是否有现成的数据结构及操作可供调用，如果有现成的库和数据结构可供调用或参考，那么我们就可以减少重复造轮子的时间，并且这些已经造好的轮子通常都是一些非常优秀的源代码可供我们学习。