**实验三 红黑树和区间树**

陈俊驰 PB18000051

1. **实验内容及要求：**

本次实验包含两个部分，红黑树和区间树，具体要求如下。

1. 红黑树

在该实验中，要求实现红黑树的基本算法，分别对整数n=20,40,60,80,100，随机生成n个互异的正整数（K1，K2......Kn），以这n个正整数作为结点的关键字，向一棵初始空的红黑树中依次插入n个结点，统计算法运行所需时间并画出时间曲线。然后，随机删除红黑树中n/4个结点，统计删除操作所需要时间并画出时间曲线图。

对于输入，从input文件夹中的input.txt文件读入，随机生成的正整数，用于构建红黑树。在本次实验中，选择生成共100个随机数，然后分别读取前20~100个作为输入。

对于输出，对于五个规模，分别构建红黑树，在time1.txt文件输出构建红黑树的插入操作所花费的时间，并在inorder.txt中输出构建好的红黑树的中序遍历序列；删除n/4个结点后，在time2.txt文件中输出删除红黑树结点的时间，并在delete\_data.txt中输出删除的结点关键字以及删除后的中序遍历序列。

1. 区间树

在该实验中，实现区间树的基本算法，随机生成30个正整数区间，以这30个正整数区间的左端点作为关键字构建红黑树，向一棵初始空的红黑树中依次插入30个节点，然后随机选择其中3个区间进行删除。实现区间树的插入删除遍历和查找算法。

对于输入，从input文件夹中的input.txt文件读入，每一行两个随机数据表示区间的左右端点，读取每行数据作为结点的int域。其中，右端点应大于左端点，所有区间取自[0,25][30,50]且左端点互异，不能有与(25,30)有重叠。

对于输出，将构建好的区间树的中序遍历序列以每行为左右端点及max域的格式输出到inorder.txt中，然后删除3个区间并将删除的数据及删除后中序遍历输出到delete.txt中，最后随机生成三个区间，其中一个取自(25,30)，在区间树中进行搜索，将搜索到的重叠区间或者Null输出到search.txt。

1. **实验设备和环境：**

本次实验所使用的实验设备为笔记本电脑，型号为2018版联想拯救者，处理器为Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz，内存为8.00GB。

所使用的实验环境为Windows 10，所使用的IDE为Visual Studio 2019。

1. **实验方法和步骤：**

首先，根据实验文档的名称与结构要求建立根文件夹，文件夹名称为36-陈俊驰-PB18000051-project3，在根文件夹下建立实验报告文件和ex1、ex2子文件夹，在子文件夹中创立三个子文件夹，分别为input文件夹，用于存储输入文件；src文件夹，用于存储源程序，包含项目文件夹及cpp文件，由于所使用的IDE为Visual Studio，每个源程序对应一个项目文件夹，由于相对路径问题，在执行时需要打开该项目在VS中执行，而不能直接通过使用Debug文件夹中存储的exe文件；output文件夹，其中为每个算法在output文件夹下构建一个子文件夹用于存储结果数据和时间结果。

1. 红黑树

0.首先根据要求生成100个互异的正整数，其中search函数用于在数组中搜索当前生成的随机数是否已经存在，从而保证互异

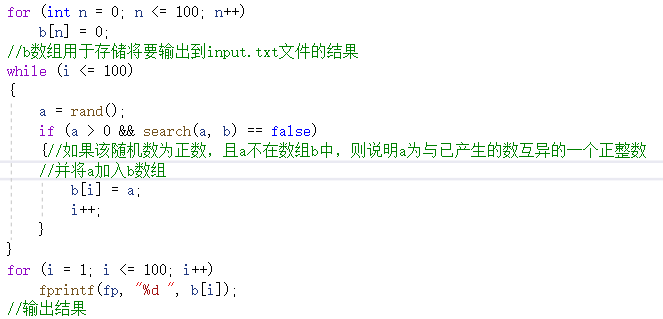


图1 生成100个互异的正整数

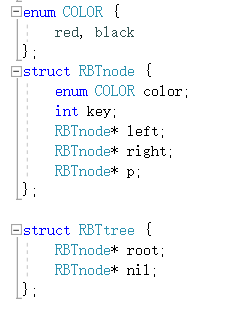
1.首先定义红黑树的结构体，RBTnode为红黑树的节点；RBTtree为红黑树，包括根与nil两部分。其中为了方便表示颜色，利用enum定义red和black。

图2 定义红黑树结构体

2.打开对应的文件并初始化相关变量数组，其中in代表输入，其余四个代表插入删除的输出。input数组用于存储输入的数据。

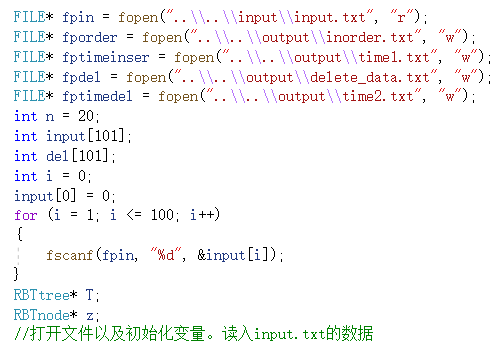


图3初始化变量打开文件

3.然后，对不同规模n进行如下的操作，首先初始化一个根节点即为T.nil的空树作为初始；然后通过创建并利用rb\_insert插入n个关键字值为读入数据的结点，并输出其时间及中序遍历序列；然后，利用del数组作为input中第i个数据是否被删除的标志，选中n/4个值，找到其结点并利用rb\_delete删除，并输出删除时间以及中序遍历序列。

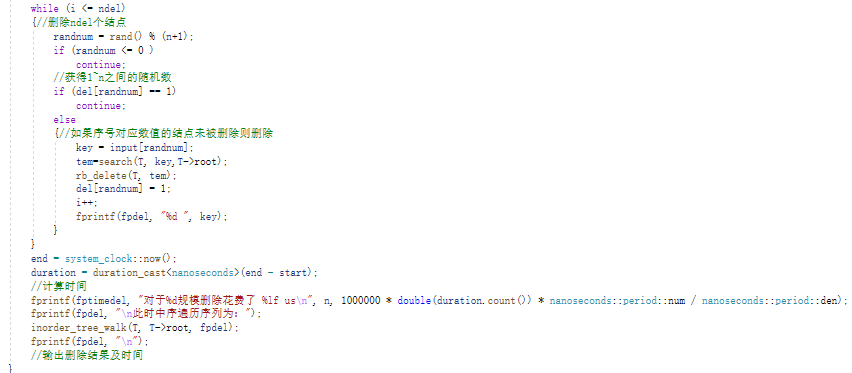
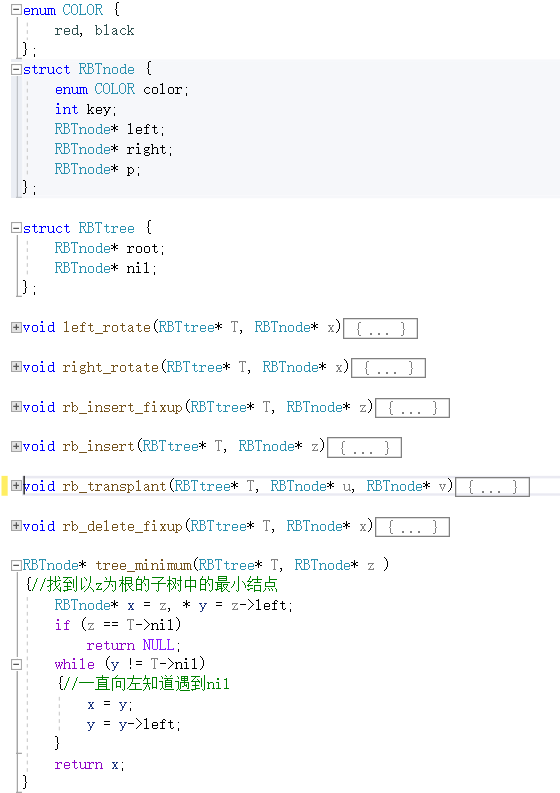


图4 对不同规模所进行的操作

4.部分展示如下的代码，其中插入删除rb\_insert和rb\_delete及相关代码由于过长就不具体展示，均按照课本所给出的伪代码的思想来书写。对于中序遍历，利用二叉树的规则来写。对于tree\_minimum和search由于红黑树本质上也是一种二叉搜索树，故利用二叉搜索树的性质来书写。其中RBTnode和RBTtree分别为红黑树结点和树的结构体定义。



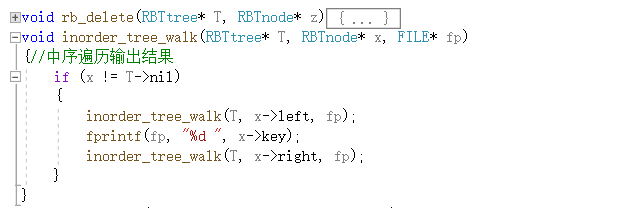


图5 插入删除及相关函数代码实现

1. 区间树

0.首先根据要求生成区间，其中通过search函数来保证不存在左端点，通过对于随机生成的左端点的值不同情况进行讨论，保证右端点要大于左端点且不与(25,30)有重叠。

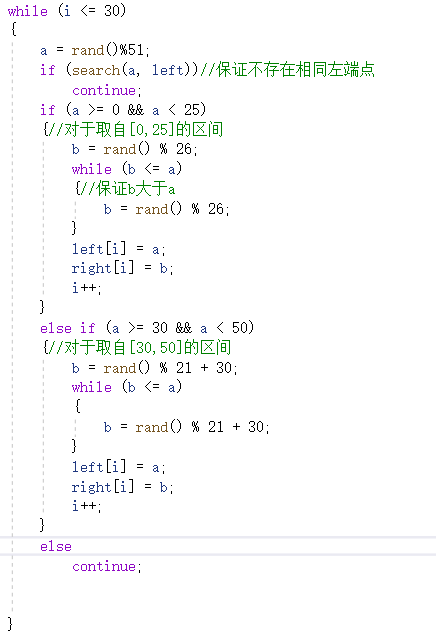


图6 生成随机的输入区间

1.首先，定义区间树的结构体，intnode表示区间树的结点；inttree表示区间树，包含root和nil；由于C语言本身不直接支持区间运算，故加入interval结构体用于定义区间，其包括区间上界high和下界low。其中为了方便表示颜色，利用enum定义red和black。

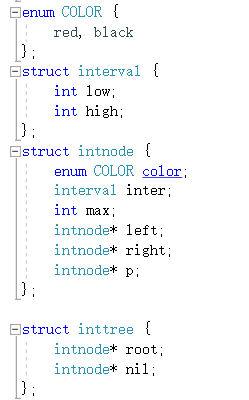


图7 区间树结构体定义

2.打开文件，并初始化变量数组，在low和high数组中对应的存储从input.txt文件中输入的区间

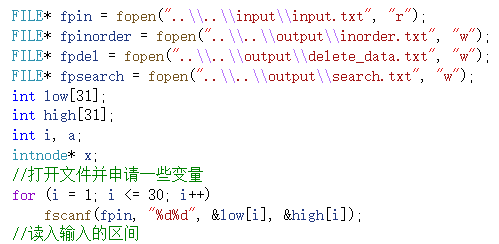


图8 初始化并读入数据

3.然后，对于所读入的数据，进行如下的操作，首先初始化一棵空的区间树T，其根节点为T.nil；声明一系列节点，并将输入的区间作为其区间域的low和high值初始化，然后调用rb\_insert插入，并输出中序遍历结果；然后随机选择并调用rb\_delete删除三个区间，并输出中序遍历结果及删除的节点值；最后随机生成3个区间，包括一个(25，30)内的区间，进行搜索并输出搜索的结果。



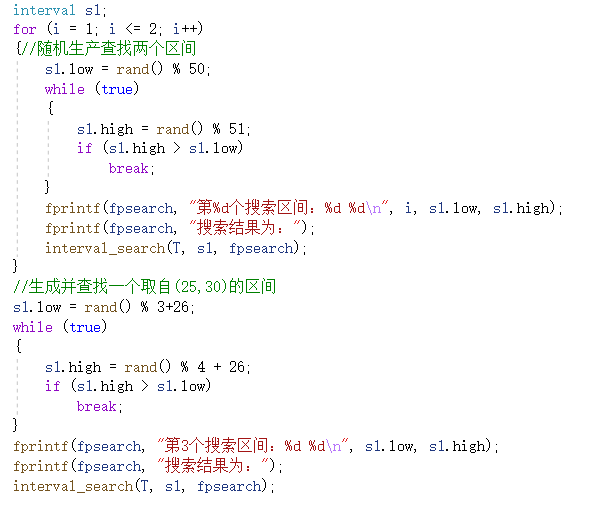


图9 区间树对于规模30的处理

4.由于原代码过长，故部分展示如下的代码。其中，由于区间树是在红黑树基础上进行扩展，故对于插入删除的基本算法保持不变，但需要加入对于max值的维护，在这里展示insert是如何维护max值的并展示一些自己加入的函数。主要加入的函数为maintain，将节点x处的max值修改为符合规则的值；overlap，利用定义判断是否重叠；check，利用中序遍历检查每一个结点的max值是否合规。对于insert，根据课本的思想，在第一阶段插入的过程中，变化将沿着插入位置向上传播，故我们在insert函数找寻路径的过程中对每一个祖先的max值进行修改；第二阶段，进行红黑性质恢复，仅有的结构改变是在于旋转，根据旋转后，y是x的父亲，先对x再对y维护其max值。对于delete，第一阶段中，为被删除节点的祖先和取代删除节点位置的结点的原祖先维护max值，第二阶段与插入同理。对于中序遍历，简单利用树的中序遍历思想即可。对于搜索，参照课本算法完成。

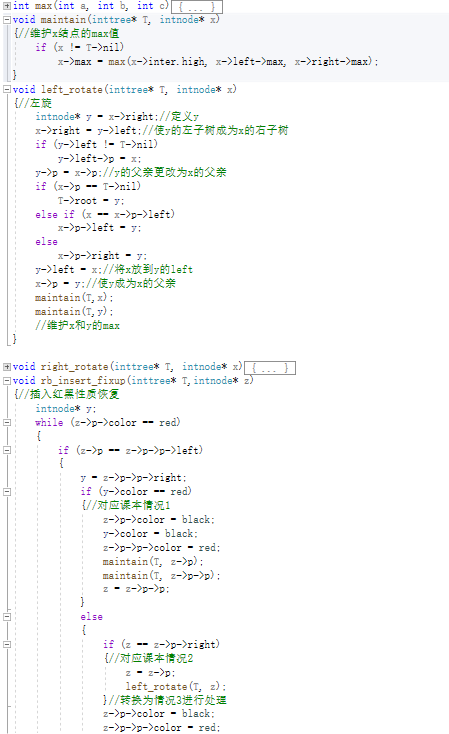






图10 区间树具体代码实现

对于红黑树，由于需要计时，且在最初几次运行时，包括编译、地址申请等因素在内，耗时不稳定，故重复多次运行取稳定的结果，以增加结果的可靠性。而对于区间树，运行一次即可。最终获得两个实验的输出结果，对于实验结果进行比较分析并撰写实验报告。

**四、实验结果与分析：**

1）红黑树

通过运行代码，获得inorder、time1、delete\_data、time2四个文件，通过IDE的调试检查生成的红黑树以及运行的输出结果，发现结果正确，成功实现了插入删除构建红黑树的要求。

对于5个规模，构建好的红黑树中序遍历如下，可见中序遍历序列符合二叉搜索树规则

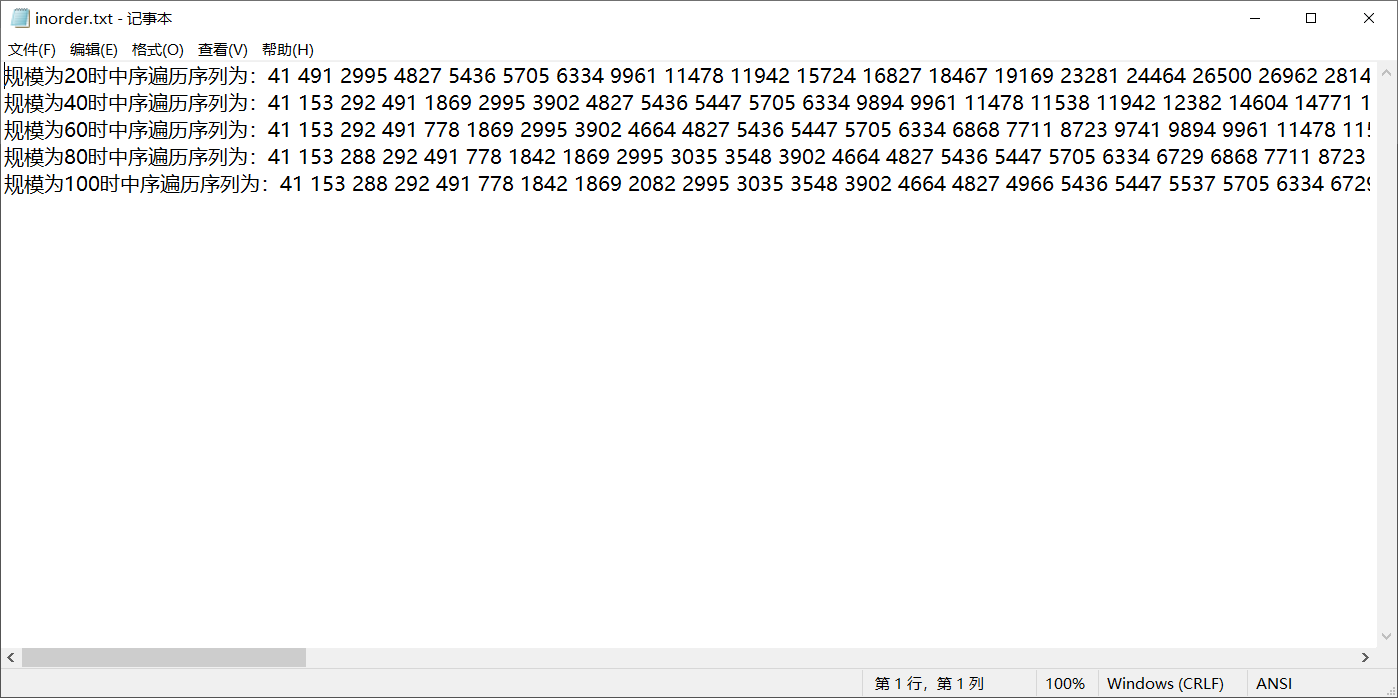


图11 构建红黑树结果中序遍历

其运行时间如下图

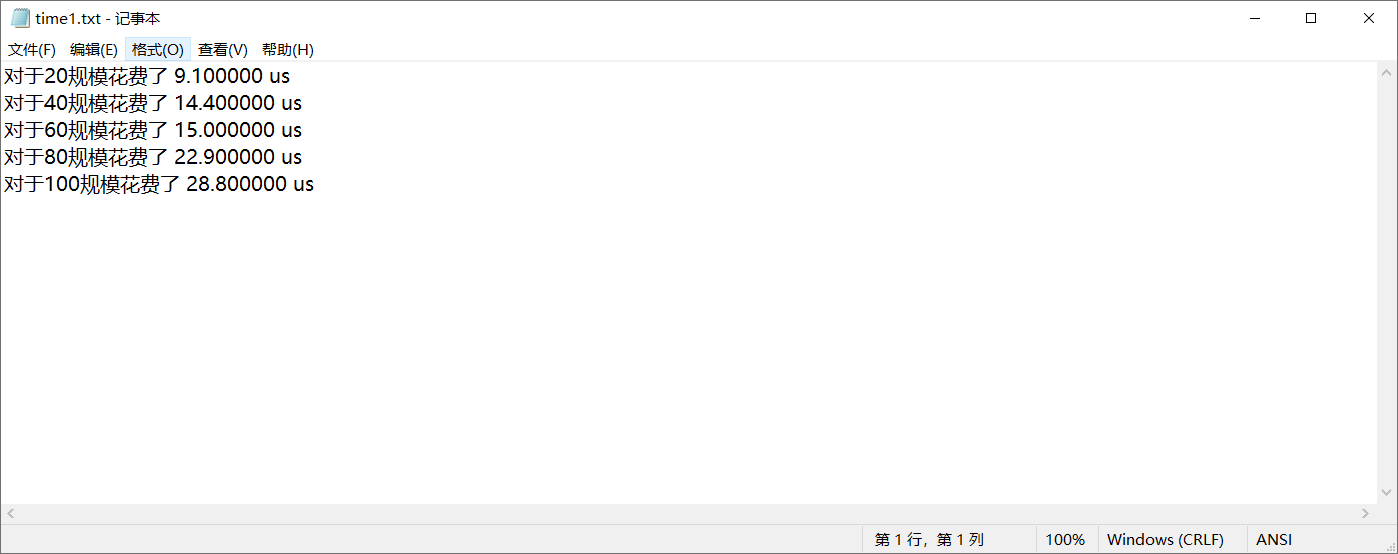


图12 构建红黑树插入操作时间

根据上述构建红黑树时的插入操作所花费时间可以获得如下所示的运行时间-规模曲线图

图13 构建红黑树插入操作时间-规模曲线图

其中横坐标为数据规模nlgn，纵坐标为运行时间单位us。通过程序拟合，发现其运行时间与规模的关系符合表达式t=0.0332nlgn+5.9296的函数，该函数为O(nlgn)。其中R²=0.9586，说明拟合程度一般，拟合结果不够精确。通过课本查询知，红黑树的插入操作插入一个节点运行时间为O(lgn)，由于我们将从一棵空树逐渐插入n个结点构成一棵红黑树，故总的运行时间应为O(lg1+lg2+...+lg(n-1))=O(lg((n-1)!))=O(nlgn)，实际复杂度与理论复杂度相同，运行结果较为准确。

此处的R²数值不够大、对理论复杂度符合程度不够高，主要有以下几方面原因，第一点，我们获得的运行时间-规模关系对较少，导致拟合不够准确；第二点，理论的时间复杂度是针对数据规模n无穷大时，由此可以推测，当数据规模很大时，运行时间应当符合O(nlgn)，而在实验中由于数据规模很小，各低阶项常数项对于运行时间贡献更大，这从拟合的函数nlgn系数仅为0.0332也可体现出，从而导致不够符合理论复杂度；最后一点，可以看出对于n=60时时间偏差最大，有可能存在其他程序干扰，导致计算机运算能力不稳定造成的。

对于五个规模的红黑树，删除操作所删除的结点及删除后的中序遍历序列如下图所示，可见删除后的中序遍历序列仍符合二叉搜索树的遍历规则

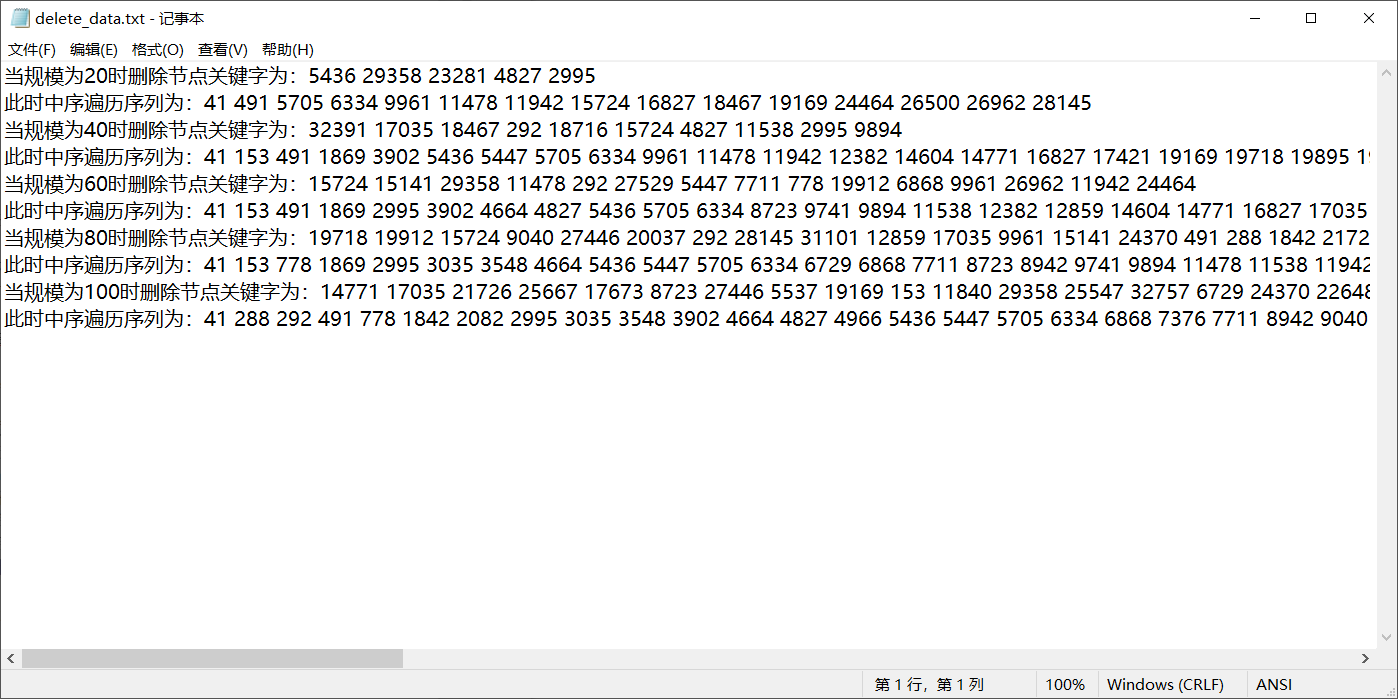


图14 删除及删除后结果示意图

其运行时间如下图

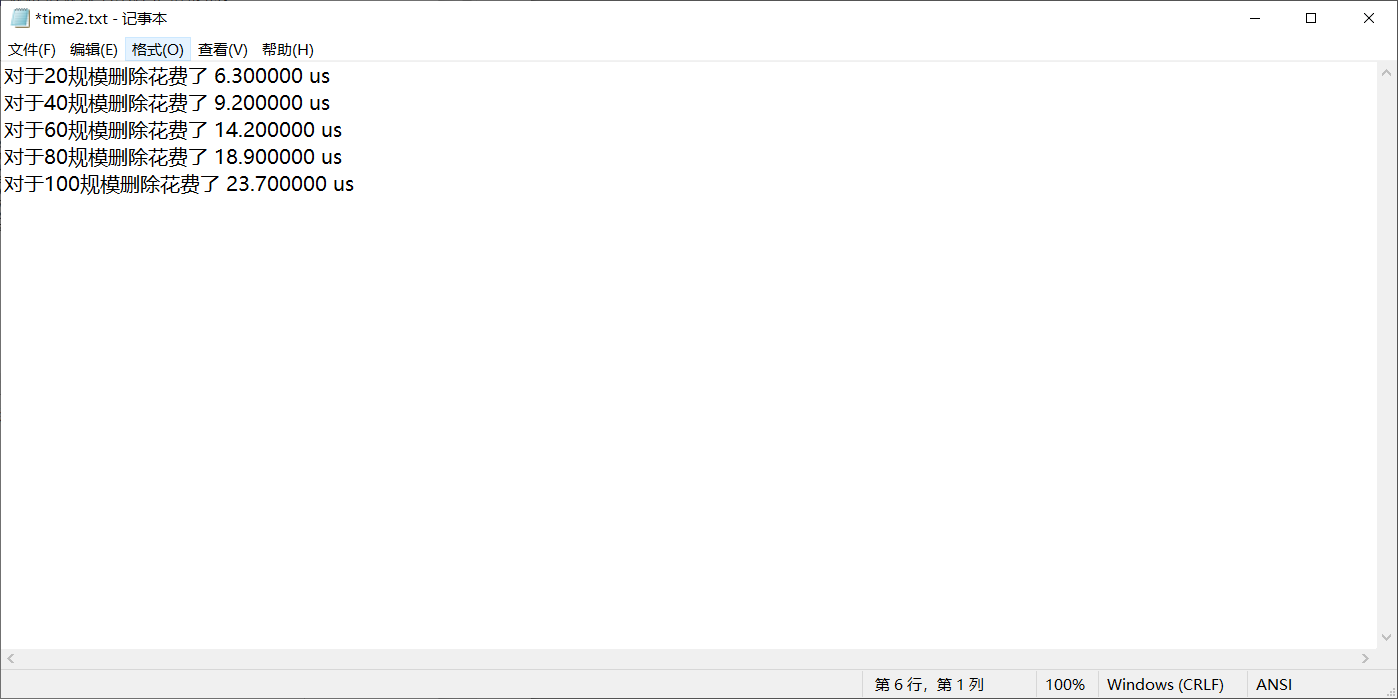


图15 删除操作运行时间

根据上述红黑树删除操作所花费时间可以获得如下所示的运行时间-规模曲线图

图16 删除操作运行时间-规模曲线图

其中横坐标为数据规模nlgn，纵坐标为运行时间单位us。通过程序拟合，发现其运行时间与规模的关系符合表达式t=0.0308nlgn+3.2419的函数，该函数为O(nlgn)。其中R²=0.9974，说明拟合程度很高，拟合结果较为精确。通过课本查知，红黑树的删除操作，删除一个结点运行时间为O(lgn)，由于我们对于不同规模为n的数据删除掉n/4个结点，相较于原树的规模，n/4个结点导致红黑树规模减小不可忽略，故总的运行时间应为O(lgn+lg(n-1)+...+lg(3n/4+1))=O(lg(n!)-lg((3n/4)!))=O(lg(n!))=O(nlgn)，实际复杂度与理论复杂度相同，运行结果很准确，各数据都很好地符合了理论复杂度，偏差很小。

2）区间树

通过运行代码获得inorder.txt、delete\_data.txt和search.txt，经过检查各文件中的输出结果以及利用check函数去检查每一个结点max值，发现结果正确，成功实现了区间树的插入、删除、中序遍历、查找算法。

对于这30个区间插入后获得的区间树中序遍历序列结果如下，可以看出，其中序遍历的结果符合区间左端点从小到大的顺序，这也符合以左端点为key值二叉搜索树的中序遍历规律，max值通过check检查正确。

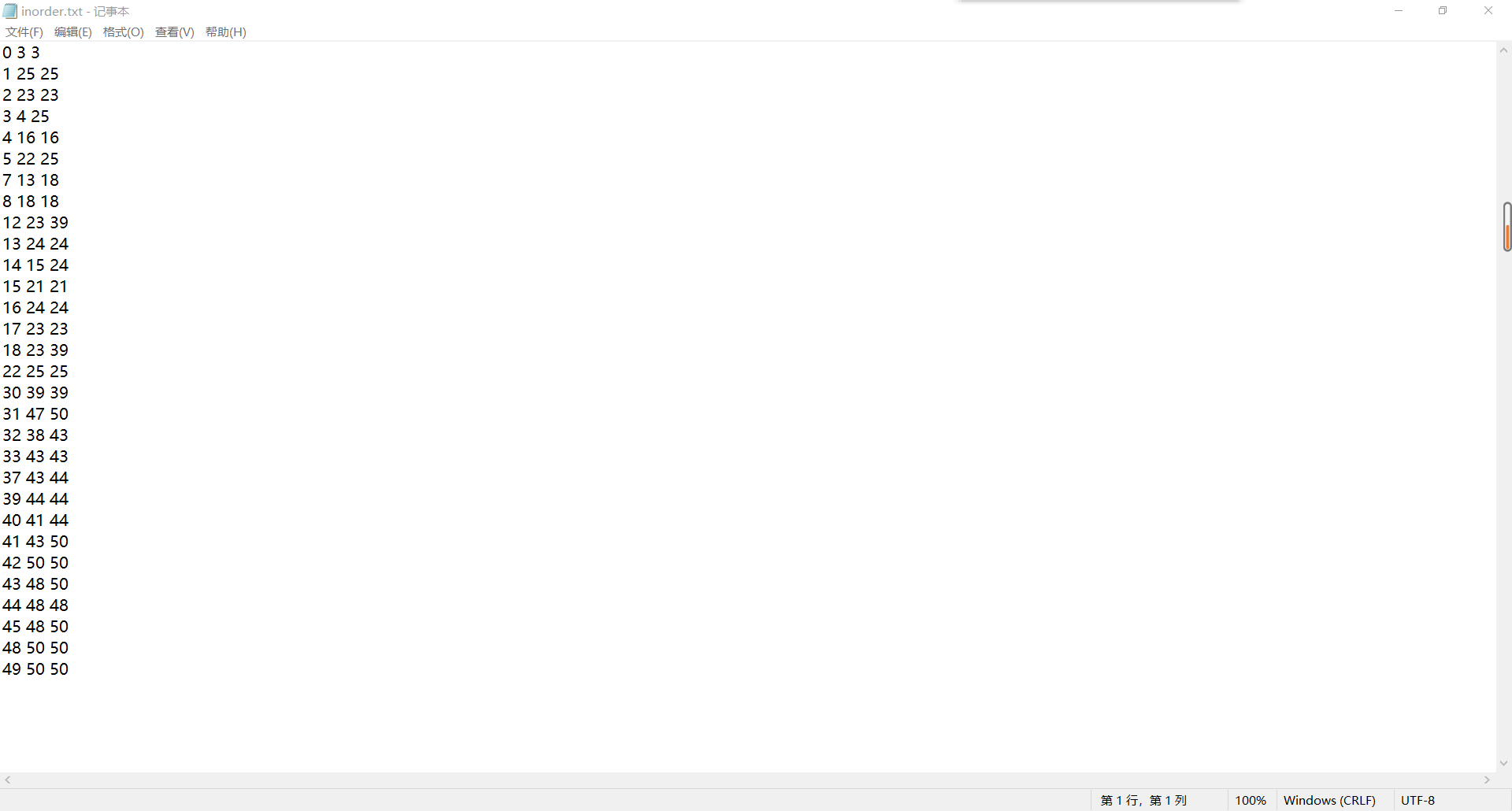


图17 区间树构建结果

删除的三个随机的区间以及删除后的中序遍历结果如下，同理，通过检查序列以及max值结果正确。相较于构建时的中序遍历序列，不存在的区间恰好为删除的三组数据。

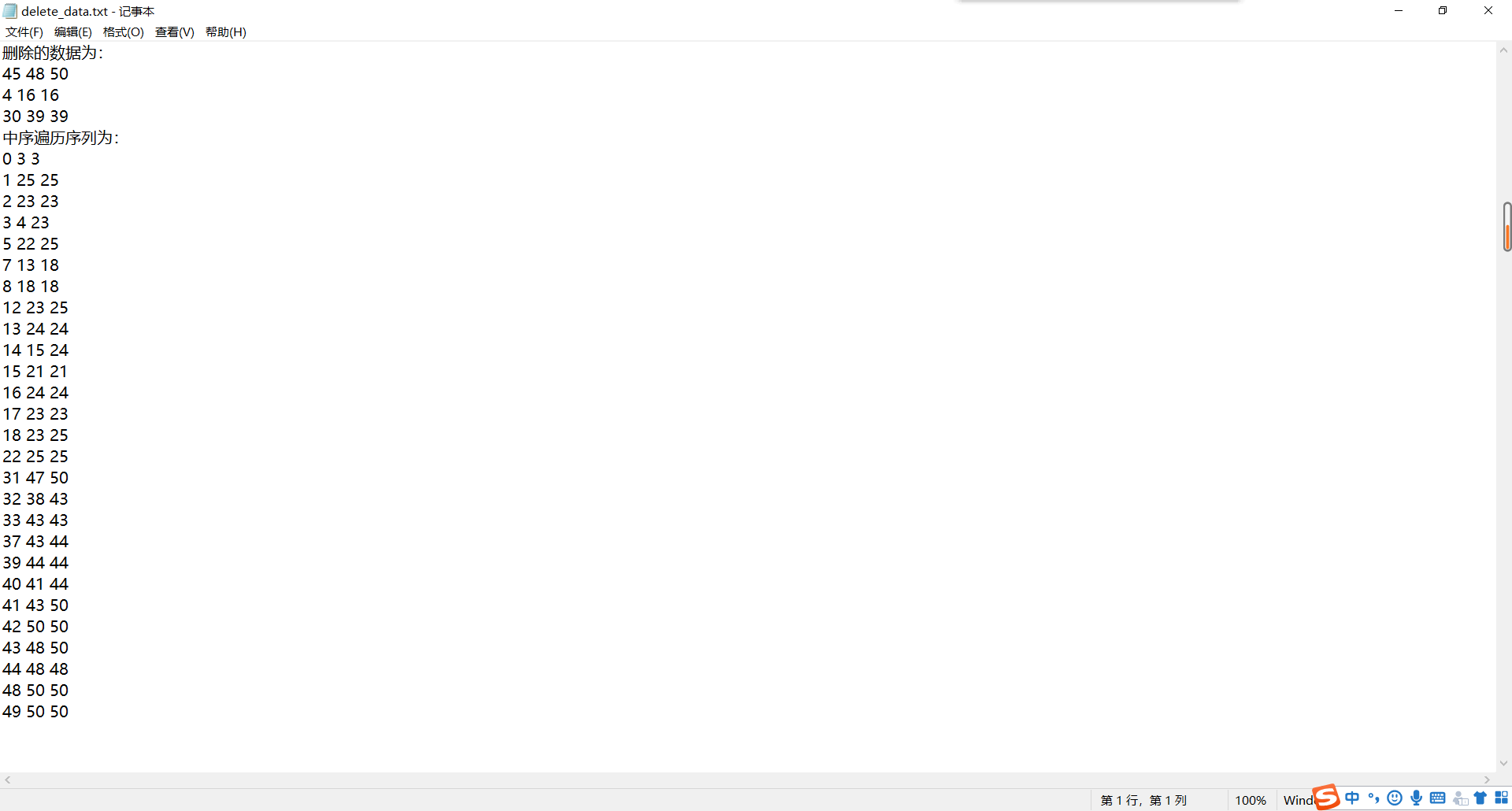


图18 删除节点及结果

三个随机生成的搜索区间及其搜索结果如下，其中前两个搜索区间的结果均与该区间重叠符合要求，第三个搜索区间根据定义知必定不存在一个区间与之重接，故搜索结果为Null，也符合要求。

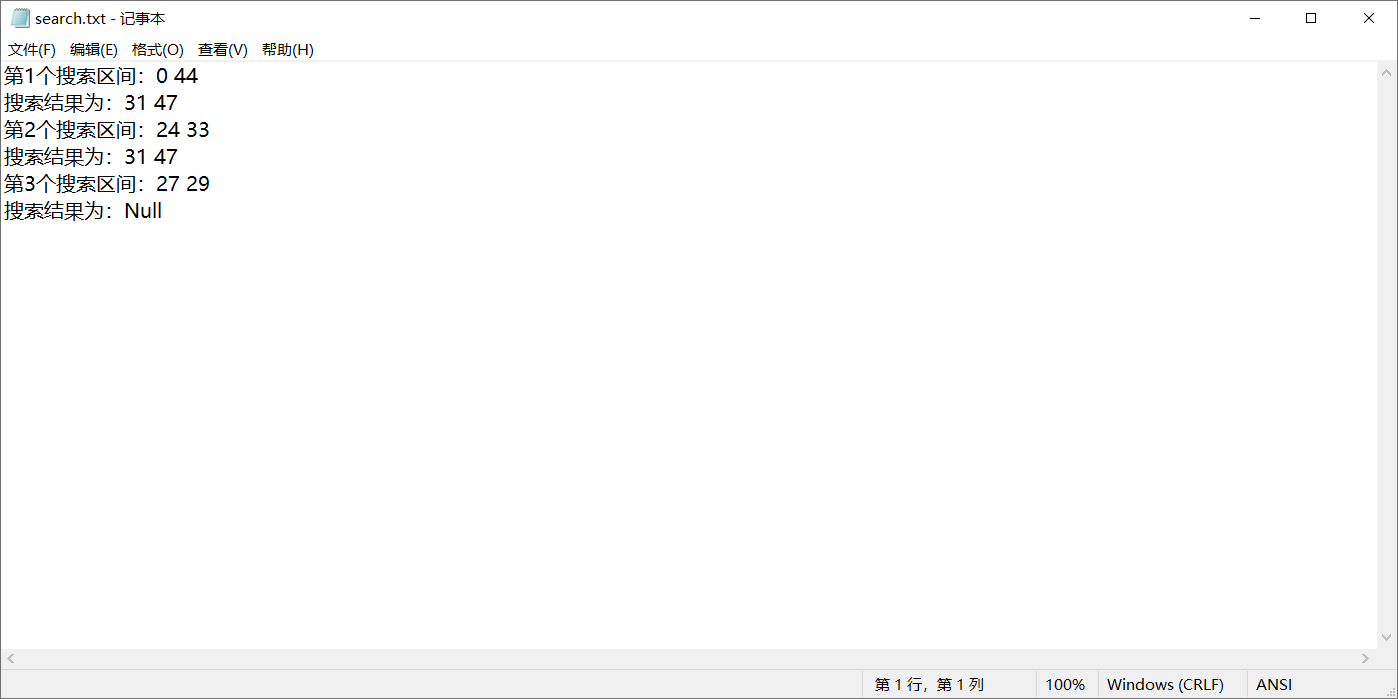


图19 搜索结果

**五、反省与总结：**

本次实验中，复习了红黑树以及区间树的相关知识，并在实际编程中，对于这两种数据结构及其相应操作进行了运用与实践，加深了对于课本相关知识的理解。在红黑树实验中，通过对于伪代码的阅读及翻译为C语言代码，并参考课本的文字说明，对于红黑树的掌握有了很大的帮助；在区间树实验中，最初完全不知晓数据扩张应该在哪里对max值进行维护，通过参考网上的一些资料，并详细阅读红黑树扩张定理的证明，详细理解了对于插入删除的操作与恢复阶段，分别对于max值应当如何处理，更加具体实际的掌握了数据扩张的方法。此外，通过实验过程中的学习，加深了对于树、递归与迭代转换等基础知识的掌握。

在实验过程中，出现了由于对于红黑树边界问题、T.nil理解不足导致出现了各项操作中访问不存在结点的情况，通过与伪代码比较、查找相关文字说明等方式解决。此外，还出现了由于条件判断导致的循环不合理终止的问题，最终通过反复检查以及利用断点debug等方式解决。在今后的实验中，应当更加细心认真，避免出现马虎导致的错误，此外，对于不了解不清楚的知识应当抓紧时间学习巩固。