数据结构实验报告

* 1. 测试四种树的搜索插入删除效率，首先测试每一种树的对案例的处理效率，再分别比较树与树之间的效率。其中递增插入N个数采用的是插入从1到N的N个整数，递减插入是从N到1插入N个整数，递增和递减删除与插入操作相似；搜索1000个数时，随机生成1000个1到N之间的数，执行搜索操作。
  2. 建立四种树的数据结构，再进行测试，其中由于二叉搜索树BST运行时间较长，所以对案例的测试只进行一次，其余均进行5次后取平均值。
  3. 按随机顺序插入或删除N个数时，采取的是“洗牌”的方式，将1到N之间的所有整数进行“洗牌”，生成随机序列，并且保证不同的随机序列生成的概率是相同的。

1. 二叉搜索树BST
   1. 按递增顺序插入N个整数，随机查找1000个数，按相同顺序删除

插入操作折线图：

搜索操作时间：0.018ms

删除操作折线图：

* 1. 按递增顺序插入N个整数，随机查找1000个数，按相反顺序删除

插入操作折线图：

搜索操作时间：0.0122ms

删除操作折线图:

* 1. 按随机顺序插入N个整数，随机查找1000个数，按随机顺序删除

插入操作折线图：

搜索操作时间: 0.0082ms

删除操作折线图：

1. 二叉平衡树AVL
   1. 按递增顺序插入N个整数，随机查找1000个数，按相同顺序删除

插入操作折线图：

搜索1000个数时间：0.0082ms

删除操作折线图：

* 1. 按递增顺序插入N个整数，随机查找1000个数，按相反顺序删除

插入操作的时间折线图:

搜索时间：0.0152ms

删除操作折线图：

* 1. 按随机顺序插入N个整数，随机查找1000个数，按随机顺序删除

随机插入操作折线图：

随机搜索时间：0.0103ms

随机删除时间折线图：

1. B-树
   1. 按递增顺序插入N个整数，随机查找1000个数，按相同顺序删除

搜索操作折线图：

搜索时间：0.0072ms

删除操作折线图：

* 1. 按递增顺序插入N个整数，随机查找1000个数，按相反顺序删除

搜索操作折线图：

搜索时间：0.0093ms

删除操作折线图：

* 1. 按随机顺序插入N个整数，随机查找1000个数，按随机顺序删除

搜索操作折线图：

搜索时间：0.0081ms

删除操作折线图：

1. 红黑树
   1. 按递增顺序插入N个整数，随机查找1000个数，按相同顺序删除

插入操作折线图：

搜索时间：0.0083ms

删除操作折线图：

* 1. 按递增顺序插入N个整数，随机查找1000个数，按相反顺序删除

插入操作折线图：

搜索时间：0.0078ms

删除操作折线图：

* 1. 按随机顺序插入N个整数，随机查找1000个数，按随机顺序删除

插入操作折线图：

搜索时间：0.0092ms

删除操作折线图：

1. 效率比较
   1. 随机插入数据，插入的序列是随机分布的，观察它们所消耗的时间

下面是去除时间较长的BST的图表

由所测数据可以得到结论：BST进行插入的效率最低，500000次插入时间大概在30分钟左右，其次是AVL，再次是RBTree，插入效率最高的是B-Tree

* 1. 查找：随机查找1000个数，比较时间

BST:0.0082ms

AVL:0.0105ms

B-Tree:0.0082ms

RBTree:0.0101ms

可知查找操作四种树效率都很高，其中B树和二叉搜索树BST的检索效率较高

* 1. 随机执行删除操作：删除500000个数，分析时间

去除B-Tree删除时间后，RBTree、AVL与BST删除时间比较

随机插入后的随机删除操作只有B-Tree效率最低，RBTree其次，AVL树再次，BST树效率最好

1. 反思
   1. 实验中的代码效率可能会影响结果的判断，尽量减少多余的代码，采用更精确的时间测量方式（对于BST树的递增插入，由于时间复杂度过高，所以实验中采用了精确度不高的clock函数计时）。
   2. 实验可能因为代码实现方式引起误差，例如实现搜索和插入过程时使用递归或是循环遍历，由于在设计代码时未能保证实现方式统一，所以会引起误差。
   3. 实验代码实现和实验数据保存在相应文件夹内