作业5报告

BST的实现：详见源代码

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

折半查找的实现：

文本

描述已自动生成

文本, 表格

中度可信度描述已自动生成

1. 生成两组数据，并建立对应的BST；

文本

描述已自动生成

将A初始化为大小为1024的2048中奇数有序序列，然后以A为输入建立BST结构，实例化为T1.

然后将A打乱顺序，作为输入建立BST结构，实例化为T2.

1. 生成测试样例，为1000个小于2048大于等于0的整数：

文本

描述已自动生成

1. 修改查找算法，返回值为搜索长度（比较次数），且为了区分成功与失败，成功返回正值，失败返回负值。

这是BST的搜索算法：

文本

描述已自动生成

折半搜索见上文。

1. 然后将test分别作为T1、T2、折半查找的测试用例，排好序的A数组作为折半查找的输入。计算失败成功次数、比较次数、平均比较次数。结果如下：

文本

中度可信度描述已自动生成

1. 结果分析：

共10000个小于2048的随机整数，从概率上来看，约有一半数据可以搜索成功，另一半失败。而实验中成功5019次，失败4981次，满足理论。

首先看T1。T1为顺序插入建立BST，所得到的树为一颗只有右子树的树，所有节点的左子树均为空，即树的高度为1024。在此上进行查找，同在排好序的一维数组上从头进行线性搜索无异，故而平均比较次数较大，从平均概率上来讲，应该平均比较512次左右。实验结果满足推论。

然后看T2。T2的建立时输入数组为一个随机序列，可以将T2看作一颗近似平衡的BST。1024=210，若T2完全平衡，则树深应为10，平均比较次数应小于等于10，但是结果比10略多（12，13），说明T2并非完全平衡。

最后看折半查找。又折半查找的复杂度分析可知，其所需最大比较次数为log21024 =10，成功情况所需次数可能小于10.实验结果符合该理论结论（成功9.02，失败10.06）。

结论：当BST近似平衡时，树深约为log2N，故一次搜索的时间复杂度为O(log2N),此时BST的时间效率跟二分查找算法相差不大，会稍慢一些（单纯考虑比较次数）。当BST严格平衡时，可以认为从平均性能来看，BST和二分查找算法差不多。但是若BST平衡性较差，如T1这种极端情况，那么BST的性能是差于二分查找算法的。

故而可以看出，搜索树的平衡程度，极大得影响了BST算法的性能，故而如何建立并保持一棵较为平衡的树，具有重要的现实意义。