作业三：Avl 树——输出给定区间内的所有元素

张竣凯
3210300361
数学与应用数学

2022 年 10 月 28 日

Write a function that takes as input a binary search tree, T, and two keys, k1 and k2, which are ordered so that k1 k2, and prints all elements X in the tree such that k1 Key(X) k2. Do not assume any information about the type of keys except that they can be ordered (consistently). Your program should run in O(K + logN) average time, where K is the number of keys printed. Bound the running time of your algorithm.

——摘自课本习题 4.37

1 设计思路

1.1 在 AVLTree.h 添加 printbetween 函数

```
1  void printbetween( AvlNode *t , const Comparable & k1, const Comparable & k2)
2  {
3      if( t == nullptr || t -> element < k1)
4          return;
5
6      else if( t -> element > k2 )
7          printbetween( t -> left, k1, k2 );
8
9      else
10         printbetween( t -> left, k1, k2 );
11         cout << t -> element << " ";
12         printbetween( t -> right, k1, k2 );
13 }
```

用于将给定区间内的所有元素打印出来，参数分别为 AvlNode *t、Comparable & k1 以及 Comparable & k2，初始的 t 指向根节点。

若 t 为空指针或 t 指向的元素小于 k1，则什么都不做就直接返回；

若 t 指向的元素大于 k2，则进行函数迭代，此时代入函数的新 t 为旧 t 的左节点；

若遇到除了以上提到的几种情况，则先进行函数迭代，此时代入函数的新 t 为旧 t 的左节点，接着打印出旧 t 指向的元素，最后再进行函数迭代，此时代入函数的新 t 为旧 t 的右节点。

1.2 在 main.cpp 中编写 test 函数

```
1  void test(int N)
2  {
3      clock_t start, finish;
4      int k1, k2;
5      AvlTree<int> T;
6      for( int i = 1; i <= N; i++)
7          T.insert(i);
8      k1=1;
9      k2=N/10;
10     for( ; k2 <= N; k2 += N/10)
11         cout << "当 k1 = " << k1 << ", k2 = " << k2 << " 时: " << endl;
12         cout << "区间内的数据: ";
13         start = clock();
14         T.printbetween( k1, k2 );
15         finish = clock();
16         cout << endl;
17         cout << "打印出区间内数据的所需时间为:"<< double(finish - start)*1000 /
18                                 CLOCKS_PER_SEC << "毫秒。" << endl;
19         cout << endl;
20     return;
21 }
```

测试用的代码大同小异，故编写 test 函数以减少代码复制，参数为题目中的 N。

该函数应用了 ctime 库记录函数实际运行所需的时间，同时也使用了 random 库来随机选取小于 N 的两个数 k1 与 k2，且 k1 k2。函数内部创建了一个 AvlTree T，储存元素类型为整型 int，并且通过 for 循环将 1 至 N 依次 insert 到 T 里，完成打印给定区间内的所有元素后，输出函数实际运行所需的时间。

1.3 在 main.cpp 中编写 main 函数

调用事先写好的 test 函数，且可控制 N 的大小

1.4 尽可能地使用引用 & 以及充分考虑必要的 const 限制

以便于减少内部复制和提高安全性

2 理论分析

AVL 树又称为高度平衡树，对于有 N 个节点的 AVL 树，其深度为 $logN$，于是在树中进行节点访问时，无论是平均或最坏情况下，其时间复杂度都为 $O(logN)$。

在从根节点开始查找到 k2 节点后，函数开始通过中序遍历访问 k1 到 k2 之间的 K 个节点，且在每个节点处的工作（输出节点的元素）时间复杂度为 $O(1)$，故访问 K 个节点的时间复杂度为 $O(K)$。

将以上两个分析结合在一起后，我们可以得到该算法的时间复杂度为 $O(logN) + O(K) = O(K + logN)$，符合题意。

3 数值结果分析

本人进行了固定 N=1000，改变 K 的测试，得到了表 1 的结果，并画出一条 K-t 的拟合曲线（图 1），显然该曲线近似为一条直线，符合先前的理论分析结果：$O(logN)+O(K) = O(K+logN)$。

表 1: 测试结果

实例	1	2	3	4	5	6	7	8	9	10
K	100	200	300	400	500	600	700	800	900	1000
t/ms	0.014	0.035	0.055	0.07	0.091	0.121	0.142	0.177	0.214	0.237

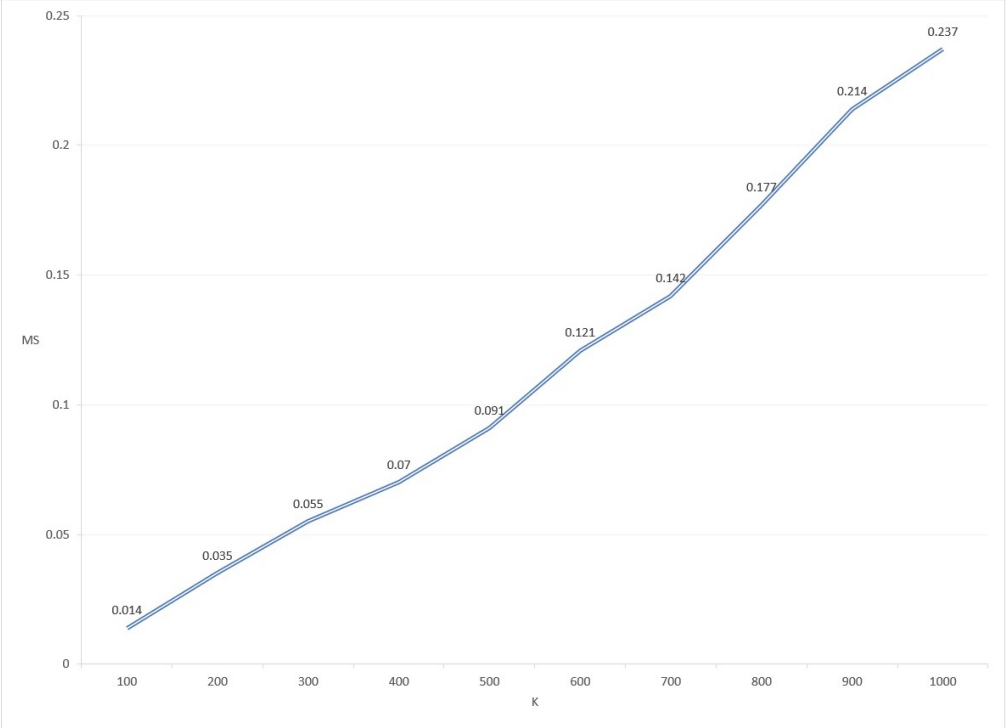

图 1: K-t 拟合曲线