Red-Black Tree Implementation

Homework #10
By
John Robertson
CS 303 Algorithms and Data Structures
November 3, 2019

1. **Problem Specification**

The purpose of this assignment is to implement a red-black tree and compare its performance to that of a hash table.

2. **Program Design**

This program, due to inheritance, has to inherit from
The following steps were required to develop this program:

- Implement the RBNode and RBTree classes
- Debug the above until they worked
- Put test cases in the file (derived from what I used for binary search tree, as well as a few red-black tree specific tests
- Make a minor change to my previous binary search tree class for the sake of completeness by adding a string representation for Node and putting that into inorder_tree_walk()
- Write the driver
- Debug the driver

3. **Testing Plan**

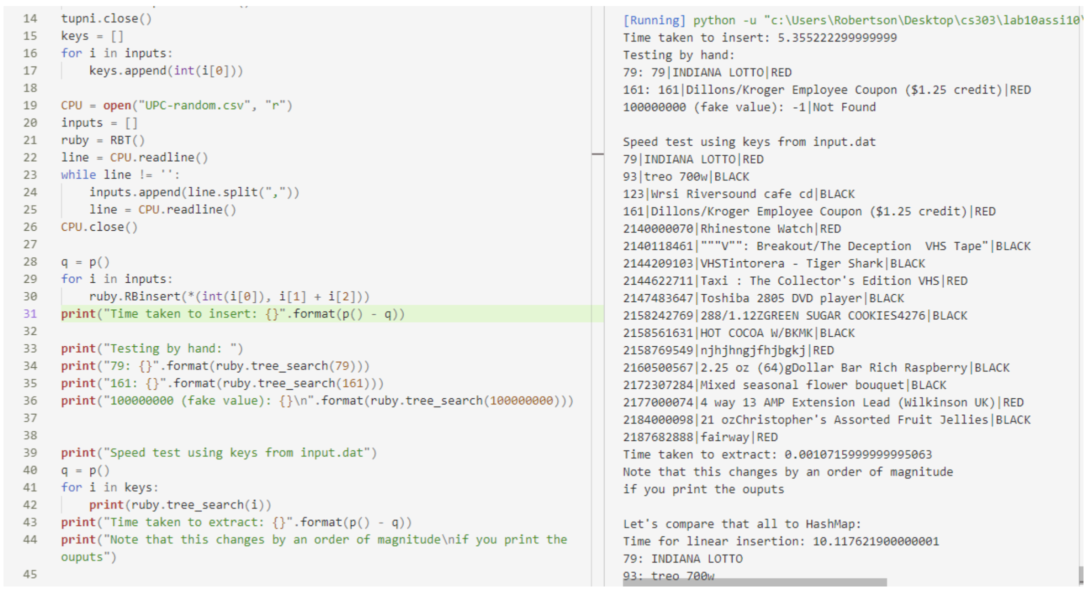
The red-black tree is tested on the same data that the original binary search tree was tested on; this is covered in the next section. Some more data are added to test the RBTs rotation aspect. Beyond that, debugging only required working out the details of whether it was inserting and rotating correctly.

4. **Test Cases**

RBT is tested on zero elements, a randomly populated list, a sorted list (worst case for a BST), a reverse sorted list (also BST worst case), and a few arrays that guarantee left and right rotations. Concerning the main driver and the given data files, only speed of insertion and extraction are looked for, as it is already known that these two functions both work.

5. **Analysis and Conclusions**

RBT Insertion Time	HashMap Quadratic Insertion Time	RBT Extraction Time	HashMap Quadratic Extraction Time
5.356 seconds	7.412 seconds	0.00107 seconds	0.00154 seconds



```python
tupni.close()
keys = []
for i in inputs:
    keys.append(int(i[0]))

CPU = open("UPC-random.csv", "r")
inputs = []
ruby = RBT()
line = CPU.readline()
while line != '':
    inputs.append(line.split(","))
    line = CPU.readline()
CPU.close()

q = p()
for i in inputs:
    ruby.RBinsert(*(int(i[0]), i[1] + i[2]))
print("Time taken to insert: {}".format(p() - q))

print("Testing by hand: ")
print("79: {}".format(ruby.tree_search(79)))
print("161: {}".format(ruby.tree_search(161)))
print("100000000 (fake value): {}\n".format(ruby.tree_search(100000000)))


print("Speed test using keys from input.dat")
q = p()
for i in keys:
    print(ruby.tree_search(i))
print("Time taken to extract: {}".format(p() - q))
print("Note that this changes by an order of magnitude\nif you print the
ouputs")

```

```
[Running] python -u "c:\Users\Robertson\Desktop\cs303\lab10assi10
Time taken to insert: 5.355222299999999
Testing by hand:
79: 79|INDIANA LOTTO|RED
161: 161|Dillons/Kroger Employee Coupon ($1.25 credit)|RED
100000000 (fake value): -1|Not Found

Speed test using keys from input.dat
79|INDIANA LOTTO|RED
93|treo 700w|BLACK
123|Wrsi Riversound cafe cd|BLACK
161|Dillons/Kroger Employee Coupon ($1.25 credit)|RED
2140000070|Rhinestone Watch|RED
2140118461|"""V"": Breakout/The Deception  VHS Tape"|BLACK
2144209103|VHSTintorera - Tiger Shark|BLACK
2144622711|Taxi : The Collector's Edition VHS|RED
2147483647|Toshiba 2805 DVD player|BLACK
2158242769|288/1.12ZGREEN SUGAR COOKIES4276|BLACK
2158561631|HOT COCOA W/BKMK|BLACK
2158769549|njhjhngjfhjbgkj|RED
2160500567|2.25 oz (64)gDollar Bar Rich Raspberry|BLACK
2172307284|Mixed seasonal flower bouquet|BLACK
2177000074|4 way 13 AMP Extension Lead (Wilkinson UK)|RED
2184000098|21 ozChristopher's Assorted Fruit Jellies|BLACK
2187682888|fairway|RED
Time taken to extract: 0.0010715999999995063
Note that this changes by an order of magnitude
if you print the ouputs

Let's compare that all to HashMap:
Time for linear insertion: 10.117621900000001
79: INDIANA LOTTO
93: treo 700w
```

I compared RBT in speed to HashMap's fastest function, quadratic probe (since I still do not have a working 7H(x)+1 probe). In terms of extraction, RBT is only marginally faster than HashMap; however, RBT is significantly faster than HashMap in terms of insertion time, considering how much of a difference two seconds is in computing. This must mean, at least against a hash map that is populated by a quadratic hash function, an RBT is faster when one will be inserting much data into it.

6. **References**

https://stackoverflow.com/q/576169/9295513
https://stackoverflow.com/q/15300550/9295513
Week 8's assignment
Week 9's assignment