Liya Xu
lx2hy
02/27/2014
Section: 103

CS 2150 Post-lab Report

Here is the test case I used for this lab (testfile4.txt):

> *a aa aaa aaaa aaaaa aaaaaa b bb bbb bbbb bbbbb bbbbbb c cc ccc cccc ccccc cccccc d dd ddd dddd ddddd dddddd e ee eee eeee eeeee eeeeee f ff fff ffff fffff ffffff g gg ggg gggg ggggg gggggg h hh hhh hhhh hhhhh hhhhhh i ii iii iiii iiiii iiiiii j jj jjj jjjj jjjjj jjjjjj k kk kkk kkkk kkkkk kkkkkk l ll lll llll lllll llllll m mm mmm mmmm mmmmm mmmmmm n nn nnn nnnn nnnnn nnnnnn o oo ooo oooo ooooo oooooo p pp ppp pppp ppppp pppppp q qq qqq qqqq qqqqq qqqqqq r rr rrr rrrr rrrrr rrrrrr s ss sss ssss sssss ssssss t tt ttt tttt ttttt tttttt u uu uuu uuuu uuuuu uuuuuu v vv vvv vvvv vvvvv vvvvvv w ww www wwww wwwww wwwwww x xx xxx xxxx xxxxx xxxxxx y yy yyy yyyy yyyyy yyyyyy z zz zzz zzzz zzzzz zzzzzz*

The text I generated is in alphabetical order, and the next one is always greater than the previous one. So the Binary Search Tree of this text will always grow to the right. This test case will show the large difference between an AVL tree and a Binary Search Tree because when you search for a word in the text, it would have to follow as many links as the depth-1 of the word. However for the AVL tree, for every node in the tree, the height of the left and right sub-trees differs at most by 1, it will always rotate to balance itself. So the links followed to search for a word would be much fewer.

In the following pages, there are the actual numerical results of both the AVL tree and the Binary Search Tree.

Testfile1

```
Please enter the name of a file of words: testfile1.txt
 we can't solve problems by using the same kind of thinking we used when
we created them -albert einstein
19 words in this text


BST:
Left links followed = 0
Right links followed = 0
Total number of nodes = 17
Avg. node depth = 3.52941

AVL Tree:
Left links followed = 0
Right links followed = 0
Total number of nodes = 17
Single Rotations = 2
Double Rotations = 2
Avg. node depth = 2.52941

    Enter word to lookup > same
    Word was found: same

BST:
Left links followed = 2
Right links followed = 2
Total number of nodes = 17
Avg. node depth = 3.52941

AVL Tree:
Left links followed = 1
Right links followed = 2
Total number of nodes = 17
Single Rotations = 2
Double Rotations = 2
Avg. node depth = 2.52941
```

Testfile2

```
Please enter the name of a file of words: testfile2.txt
 a bee caught dung everywhere flying greatly higher in mauve skies than we had ever
flown
16 words in this text


BST:
Left links followed = 0
Right links followed = 0
Total number of nodes = 16
Avg. node depth = 6.0625

AVL Tree:
Left links followed = 0
Right links followed = 0
Total number of nodes = 16
Single Rotations = 9
Double Rotations = 0
Avg. node depth = 2.5

        Enter word to lookup > greatly
        Word was found: greatly

BST:
Left links followed = 0
Right links followed = 6
Total number of nodes = 16
Avg. node depth = 6.0625

AVL Tree:
Left links followed = 1
Right links followed = 2
Total number of nodes = 16
Single Rotations = 9
Double Rotations = 0
Avg. node depth = 2.5
```

Testfile3

```
Please enter the name of a file of words: testfile3.txt
 zany cobwebs littered the clockwork orange landscape like misty works of surreal art
13 words in this text


BST:
Left links followed = 0
Right links followed = 0
Total number of nodes = 13
Avg. node depth = 3.23077

AVL Tree:
Left links followed = 0
Right links followed = 0
Total number of nodes = 13
Single Rotations = 1
Double Rotations = 2
Avg. node depth = 2.23077

        Enter word to lookup > misty
        Word was found: misty

BST:
Left links followed = 3
Right links followed = 2
Total number of nodes = 13
Avg. node depth = 3.23077

AVL Tree:
Left links followed = 2
Right links followed = 1
Total number of nodes = 13
Single Rotations = 1
Double Rotations = 2
Avg. node depth = 2.23077
```

Testfile4

a aa aaa aaaa aaaaa aaaaaa b bb bbb bbbb bbbbb bbbbbb c cc ccc cccc ccccc cccccc d dd ddd dddd ddddd dddddd e ee eee eeee eeeee eeeeee f ff fff ffff fffff ffffff g gg ggg gggg ggggg gggggg h hh hhh hhhh hhhhh hhhhhh i ii iii iiii iiiii iiiiii j jj jjj jjjj jjjjj jjjjjj k kk kkk kkkk kkkkk kkkkkk l ll lll llll lllll llllll m mm mmm mmmm mmmmm mmmmmm n nn nnn nnnn nnnnn nnnnnn o oo ooo oooo ooooo oooooo p pp ppp pppp ppppp pppppp q qq qqq qqqq qqqqq qqqqqq r rr rrr rrrr rrrrr rrrrrr s ss sss ssss sssss ssssss t tt ttt tttt ttttt tttttt u uu uuu uuuu uuuuu uuuuuu v vv vvv vvvv vvvvv vvvvvv w ww www wwww wwwww wwwwww x xx xxx xxxx xxxxx xxxxxx y yy yyy yyyy yyyyy yyyyyy z zz zzz zzzz zzzzz zzzzzz
156 words in this text

```
BST:
Left links followed = 0
Right links followed = 0
Total number of nodes = 156
Avg. node depth = 77.5

AVL Tree:
Left links followed = 0
Right links followed = 0
Total number of nodes = 156
Single Rotations = 148
Double Rotations = 0
Avg. node depth = 5.41667

    Enter word to lookup > qqqqq
    Word was found: qqqqq

BST:
Left links followed = 0
Right links followed = 100
Total number of nodes = 156
Avg. node depth = 77.5

AVL Tree:
Left links followed = 4
Right links followed = 3
Total number of nodes = 156
Single Rotations = 148
Double Rotations = 0
Avg. node depth = 5.41667
```

A characterization of situations where AVL trees are preferable to Binary Search trees is that when the text or data is sorted. This would make the Binary Search tree have only the left branches or the right branches, thus making the average depth of it much deeper than the AVL tree. The worst run time for search, insert and delete in the AVL tree is O(log n), and for Binary Search Tree is O(n). We can tell the difference from their runtimes that Binary Search Tree is much slower than the AVL tree.

However most of the time, the cost of the implementation of an AVL tree is high. Because every time you insert or delete a node in an AVL tree, it will self sort itself through a single rotation or a double rotation. The AVL tree also takes more memory to hold it because the program also has to remember the balance factor while the binary search tree doesn't need to. The time wasted is sometimes even more than the time it saves by having depth. In some cases, the data set being processed would contain limited amount of worst cases, so implementing AVL might not always be better than the binary search tree. In addition, since the difficulty of implementing a AVL tree is much higher than implement a binary search tree, it is harder for programmers to debug the program especially when it comes to a larger projects.