Yingqing Huang
10/13/16
Section 106
yh3ra

2. Testfile4.txt:

apple banana cherry date elderberry fig grape hami jujube kumquat lemon mango nut orange peach strawberry tangerine watermelon

3. Why testfile4.txt works

My txt file is ordered from a to w, missing some characters. For binary search tree, it will be a long tree, which only has a single right child for each node, except for the leaf. On the other hand, for AVL tree, it creates a balanced tree. When I entered the word “watermelon”, which is the last word in the list, to look up, the right links followed for binary search tree is 17, while there is only 18 nodes, and for AVL Tree, the right links followed is only 4, both of them having left links followed 0. The average node path for binary search tree is 8.5, and AVL Tree is 2.556. From both data, it shows that AVL Tree is marked superior to binary search tree trees. It does not cost to much to achieve such better performance, just several rotations.

4. Numerical Result:

- testfile1.txt
  - BST
    - Total number of nodes = 17
    - Avg. node depth = 3.52941
  - AVL Tree
    - Total number of nodes = 17
    - Single Rotations = 2
    - Double Rotations = 2
    - Avg. node depth = 2.52941
  - Lookup word: thinking
    - BST
      - Left links followed = 2
      - Right links followed = 3
    - AVL
      - Left links followed = 1
      - Right links followed = 1
    - BST is much slower than AVL tree (just a single case)
- testfile2.txt
  - BST
    - Total number of nodes = 16
    - Avg. node depth = 6.0625
  - AVL
    - Total number of nodes = 16

        - Single Rotations = 9
        - Double Rotations = 0
        - Avg. node depth = 2.5
    - Lookup word: bee
        - BST
            - Left links followed = 0
            - Right links followed = 1
        - AVL
            - Left links followed = 2
            - Right links followed = 0
        - BST is a little bit faster than AVL tree (just a single case)
- testfile3.txt
    - BST
        - Total number of nodes = 13
        - Avg. node depth = 3.23077
    - AVL
        - Total number of nodes = 13
        - Single Rotations = 1
        - Double Rotations = 2
        - Avg. node depth = 2.23077
    - Lookup word: works
        - BST
            - Left links followed = 1
            - Right links followed = 3
        - AVL
            - Left links followed = 1
            - Right links followed = 2
        - BST is a little bit slower than AVL tree (just a single case)
- testfile4.txt
    - BST
        - Total number of nodes = 18
        - Avg. node depth = 8.5
    - AVL
        - Total number of nodes = 18
        - Single Rotations = 13
        - Double Rotations = 0
        - Avg. node depth = 2.55556
    - Lookup word: watermelon
        - BST

        - Left links followed = 0
        - Right links followed = 17
    - AVL
        - Left links followed = 0
        - Right links followed = 4
    - BST is much slower than AVL tree (just a single case)

5. A characterization of situations where AVL trees are preferable to BSTs.

When the binary search trees is not balanced, and the node we are trying to find is not root or top several nodes, AVL trees are preferable to binary search trees.

From the former cases, testfile2 and testfile4, they are both increased from “a” to bigger char. Therefore, for binary search trees, the length of the tree will be the same as the height of the tree. If we are trying to find the bottom node, the time is really long. However, for AVL trees, they are balanced and when search for a word, the worst case is to find the leaves, which the total links followed equals to the height of the tree. Therefore, for binary search trees O(n), and for AVL trees O(log(n)), whenever log(n) is smaller than n, AVL trees are preferable to binary search trees. In testfile2, binary search tree is a little bit faster than AVL trees. This is because word “bee” is at the second level of binary search tree, but at third level of AVL trees. However, if the word looked up is not first 4 words, binary search tree will be slower than AVL trees in this case.

6. A discussion of the costs incurred in an AVL implementation.

The cost appears to be those single and double rotations to let AVL tree become balance. Whenever AVL trees are updated (insertion and deletion), and the new node is not at leaf to make the whole tree balanced, the cost appears. AVL trees only take advantage in finding elements comparing to binary search trees.