CS 315 Week 13 (April 29 and May 1) summary and review questions

Topics covered:

- April 29: AVL tree insertion (single and double rotation) completed
- May 1: Rectangle area computation problem

Summary:

- AVL tree insertion involves three cases: single and double rotation. Case 1 is the simple one – insertion preserves the AVL tree property. So no need to change the structure of the tree. The height information about some of the nodes might have changed and this changed is done and it completes the process. In the other two cases, the AVL tree property may not hold at one or more nodes on the path from the root to the leaf that was inserted. Let N be the lowest node where this violation occurs. It is easy to see that there are at least two nodes on the path from N to the newly created leaf (not including N). Let the first two nodes in the path be N1 and N2. Thus, N1 is a child of N, and N2 is a child of N1. This leads to four cases each of which is named as follows:
    - N1 is a left child of N and N2 is a left child of N1. (LL case)
    - N1 is a left child of N and N2 is a right child of N1. (LR case)
    - N1 is a right child of N and N2 is a left child of N1. (RL case)
    - N1 is a right child of N and N2 is a right child of N1. (RR case)
- Cases 1 and 4 require a single rotation.
- Cases 2 and 3 require a double rotation.

Review questions:

1) Exhibit the AVL tree that results when the keys 1, 2, 3, …, 10 are inserted in this order. How many single and double rotations were required?

2) Show the Y-tree and the X-tree at every step for the following collection of rectangles:

    5, 10, 20, 20

    10, 20, 20, 20

    0, 25, 15, 30

3) What is the time complexity of the problem to compute the total length of the union of a collection of intervals? (Express it in terms of n = the number of intervals.)

4) Modify the insert algorithm for a binary search tree that has a successor link at each node. Thus, each node has, in addition to LEFT, RIGHT and KEY, another pointer SUC that points to the next bigger key in the tree. When inserting a new node with key k, make sure that this node is pointing to its successor, and its predecessor is updated to point to this node.