Balanced Search Trees

- Read section 6.3 (pages 218-225)
- How do **balanced** search trees differ from "general" search trees?
- What is the time complexity of search, insert and delete operations in a search tree in terms of the height of the tree?
- In a perfectly balanced search tree, what is the relation between the height of the tree and the number of items?
- **Self-balancing** trees are examples of transform-and-conquer. Which type of transform-and-conquer is that?
- What is the definition of **AVL** trees?
  - Explain the definition in simple terms.
  - What is the **balance factor** of a node?
  - Give an example of a tree that is not an AVL tree.
  - What operations do we call **rotations**? What are the different kinds of rotations?
  - Study figure 6.6 on how the rotation operations allow a tree to maintain its AVL status.
- Practice problems: 6.3.1, 6.3.2, 6.3.4, 6.3.5
- Describe **2-3 trees**.
  - What type of transform-and-conquer do these represent?
  - What kinds of nodes do these trees have? How do they each work?
  - What other requirement must 2-3 trees satisfy?
  - How do we search for an element in a 2-3 tree?
  - Insertion in such a tree goes as follows: Add the element to the deepest node you can. If that results in more than 2 keys in a node, lift the middle up and continue.
- Place the letters of the word ALGORITHM, one at a time, in an AVL tree as well as a 2-3 tree.
- Practice problems: 6.3.7