Tree Balancing: AVL Trees

Dr. Yingwu Zhu
Recall in BST

- The insertion order of items determine the shape of BST
  - Balanced: search $T(n) = O(\log N)$
  - Unbalanced: $T(n) = O(n)$

- Key issue:
  - Need to keep a BST balanced!
  - Tree balancing techniques
AVL Tree Definition

- First, a BST
- Second, height-balance property: balance factor of each node is 0, 1, or -1
- Question: what is balance factor?
AVL Tree Definition

• First, a BST
• Second, **height-balance property**: balance factor of each node is 0, 1, or -1
• Question: what is **balance factor**?

\[
\text{BF} = \text{Height of the left subtree} - \text{height of the right subtree}
\]

Height: # of levels in a subtree/tree
Determine balance factor

```
      A
     / \  
    B   D
   / \  /  
  C   E G
   \  /  
    F  H
```
ADT: AVL Trees

- Data structure to implement

```
  Balance factor
     Data
     Left  Right
```
ADT: AVL Trees

• Basic operations
  – Constructor, search, traversal, empty
  – Insert: keep balanced!
  – Delete: keep balanced!
  – See P842 class declaration
  – Similar to BST
Example

Insert “DE”, what happens? Need rebalancing?
Basic Rebalancing Rotation

• Single rotation:
  – Right rotation: the inserted item is on the Left subtree of Left child of the nearest ancestor with BF of 2
  – Left rotation: the inserted item is on the Right subtree of Right child of the nearest ancestor with BF of -2

• Double rotation
  – Left-right rotation: the inserted item is on the Right subtree of Left child of the nearest ancestor with BF of 2
  – Right-left rotation: the inserted item is on the Left subtree of Right child of the nearest ancestor with BF of -2
How to perform rotations

• Rotations are carried out by resetting links

• Two steps:
  – Determine which rotation
  – Perform the rotation
Right Rotation

• **Key**: identify the nearest ancestor of inserted item with BF +2
• **A**: the nearest ancestor.
• **B**: left child
• **How?** (clockwise rotation)
  – **Step 1**: reset the link from parent of A to B (promote B)
  – **Step 2**: set the left link of A equal to the right link of B
  – **Step 3**: set the right link of B to A (demote A)
Example
Exercise #1

How about insert 4
Left Rotation

• How? (counter-clockwise rotation)
  – **Step1**: reset the link from parent of A to B (promote B)
  – **Step2**: set the right link of A to the left link of B
  – **Step3**: set the left link of B to A (demote A)
Example
Exercise #2

What if insert 18
Exercise #3

How about insert 1
Double Rotation

• **Left-right rotation**: the inserted item is on the **Right** subtree of **Left** child of the nearest ancestor with BF of 2

• **Right-left rotation**: the inserted item is on the **Left** subtree of **Right** child of the nearest ancestor with BF of -2
Double Rotations

• How to perform?
  – Step 1: Rotate child and grandchild nodes of the ancestor (grandchild < -- > child)
  – Step 2: Rotate the ancestor and its new child node
Example
Exercise #4

How about insert 7
Exercise #5

What if insert 15
Exercise #6

What if insert 14
Summary on Rebalancing

- AVL tree definition
- The key is you need to identify the nearest ancestor of inserted item
- Determine the rotation by definition
- Perform the rotation
- Implementation of AVL trees is not required in this class