**CS311 Yoshii For HW4ECs**

## To figure out the Balance Factor, you need the heights of both children first.

## Every time a node is added or deleted, the height and the balance factor of all ancestors will have to be updated. To do this, call a function climbUp that uses the UP link.

***AVL Tree Rotations (see the file ECRot)***

As soon as 2 or -2 is found during **climbUp**, you must do a **rotation**.

***Function FixIt Identifies which case it is and call the correct rotation function:***

**Case 1 : BFs are 2 and right 1**

**Case 2: BFs are -2 and left -1**

**Case 3: BFs are 2 and right -1**

**Case 4: BFs are -2 and left 1**

## Case 1: Single Left Rotation is required because it became right heavy

## The code to do this: Red is where (2) was found

* Vertex\* Pivot;
* Pivot = Red->Right; // Right child of the red node is the Pivot
* Red->Right = Pivot->Left;
* Pivot->Left = Red;
* The Pivot becomes the root of this subtree. You must also update:
* the height of Red and Pivot.
* the balance factors of Red and Pivot.
* the Up links of Pivot->Left, Red and Pivot.

## Case 2: Single Right Rotation is required because it became left heavy

## The code to do this: Red is where (-2) was found.

* Vertex\* Pivot;
* Pivot = Red->Left; // Left child of the red node is the Pivot
* Red->Left = Pivot->Right;
* Pivot->Right = Red ;
* The Pivot becomes the root of this sub-tree. You must also update:
* the height of Red and Pivot.
* the balance factors of Red and Pivot.
* the Up links of Pivot->Right, Red and Pivot.

## Case 3: Double Left Rotation is required because it became right heavy (with left heavy substree)

## The code to do this: Red was (2) and green was (-1) Do the right rotation for the right sub-tree

* Vertex\* Pivot;
* Pivot = Green->Left;
* Green->Left = Pivot->Right;
* Pivot->Right = Green ;
* The Pivot becomes the root of this sub-tree. You must also update:
* the height of Green and Pivot.
* the balance factors of Green and Pivot.
* the Up links of all nodes involved above

## Do the left rotation for the red tree.

* Vertex\* Pivot;
* Pivot = Red->Right; // Right child of the red node is the Pivot
* Red->Right = Pivot->Left;
* Pivot->Left = Red;
* The Pivot becomes the root of this subtree. You must also update:
* the height of Red and Pivot.
* the balance factors of Red and Pivot.
* the Up links of all nodes involved above

## Case 4: Double Right Rotation is required because it became left heavy (with right heavy substree)

## The code to do this: Red was (-2) and green was (1) Do the left rotation for the left sub-tree.

* Vertex\* Pivot;
* Pivot = Green->Right;
* Green->Right = Pivot->Left;
* Pivot->Left = Green;
* The Pivot becomes the root of this subtree. You must also update:
* the height of Green and Pivot.
* the balance factors of Green and Pivot.
* the Up links of all nodes involved above.

## Do the right rotation for the red tree.

* Vertex\* Pivot;
* Pivot = Red->Left;
* Red->Left = Pivot->Right;
* Pivot->Right = Red ;
* The Pivot becomes the root of this sub-tree. You must also update:
* the height of Red and Pivot.
* the balance factors of Red and Pivot
* the Up links of all nodes involved above.