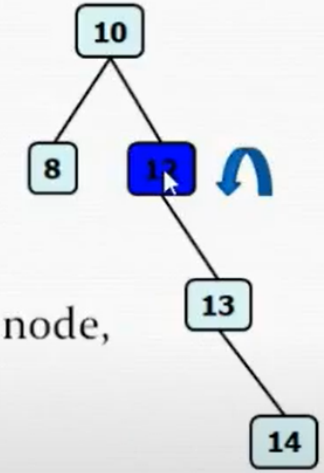
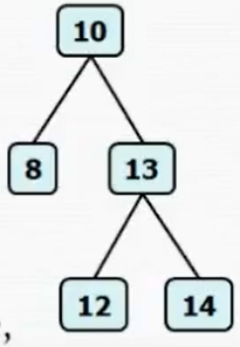
Right-Left Case

- Consider the AVL tree
- Insert 13 in the tree
- Travel upwards from new node to find first imbalance node i.e. 12
- If new node is right-right to imbalance node, its right-right case
- Two steps to balance:
  - Do a rotate right at 12->right(i.e. 14)
  - Do a rotate left at 12



Right-Left Case

- Consider the AVL tree
- Insert 13 in the tree
- Travel upwards from new node to find first imbalance node i.e. 12
- If new node is right-right to imbalance node, its right-right case
- Two steps to balance:
  - Do a rotate right at 12->right(i.e. 14)
  - Do a rotate left at 12



Computing Height

- Height of node: number of edges between node and farthest leaf
- We can calculate height of any node recursively
- height(x) = 1 + max(height(left), height(right))
- Height of leaf nodes are 0.

Compute Height : Implementation

```
int height(tree) {
   if (tree == NULL) {
      return -1;  //For NULL tree, eases computation
   }
   else {
      return (1 + max(heightBST(tree->left),
                      heightBST(tree->right)));
   }
}
```

Time Complexity = **O(n)**

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Implementation

- We alter the node structure to have one more field **ht**
- Updated node structure therefore is

```
typedef struct Node {
        int data;
        struct Node * left;
        struct Node * right;
        int ht;
} * Nptr;
```

- "ht" of leaf nodes are 0. Thus every new node has ht=0

Alka Jindal

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Implementation

(cont.)

```
//Algorithm
Nptr insertAVL(Nptr tree, int data) {
    1. insert similar to BST
    2. This insertion might have caused change in height of
ancestors of new node, thus update "height" of ancestors
    3. Check balance factor of every ancestor as we move up in
the AVL Tree
    4. if balance factor is greater than 1, test for left-left or left-
right case and fix accordingly
    5. if balance factor is less than -1, test for right-right or
right-left case and fix accordingly
}
```

Alka Jindal

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Implementation

(cont.)

```
int getHt(Nptr tree) {
   if (tree == NULL)
      return -1;
   else
      return (tree->ht);
}

int balanceFactor(Nptr tree) {
   if (tree == NULL)
      return 0;
   else
      return (getHt(tree->left) - getHt(tree->right));
}
```

Alka Jindal

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Implementation

(cont.)

```
Nptr leftRotate(Nptr tree) {
    Nptr X = tree->right;
    Nptr Y = X->left;

    tree->right = Y;
    X->left = tree;

    tree->ht = 1 + max(getHt(tree->left), getHt(tree->right));
    X->ht = 1 + max(getHt(X->left),getHt(X->right));
    return X;
}
```

Alka Jindal

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Implementation

(cont.)

```
Nptr rightRotate(Nptr tree) {
    Nptr X = tree->left;
    Nptr Y = X->right;

    tree->left = Y;
    X->right = tree;

    tree->ht = 1 + max(getHt(tree->left), getHt(tree->right));
    X->ht = 1 + max(getHt(X->left),getHt(X->right));
    return X;
}
```

Alka Jindal

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

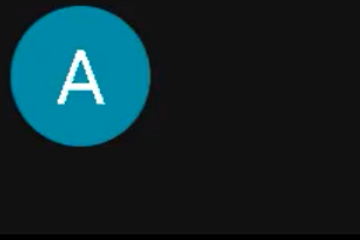
Insertion in AVL : Implementation

(cont.)

```
Nptr insertAVL(Nptr tree, int data) {
   //insertBST function, thus skipping writing
   //update height of current node
   tree->ht= 1 + max(getHt(tree->left), getHt(tree->right));
   int bal = balanceFactor(tree);

   if (bal > 1 && data < tree->left->data)  //left-left case
      return rightRotate(tree);

   else if (bal > 1 && data > tree->left->data) {  //left-right case
      tree->left = leftRotate(tree->left);
      return rightRotate(tree);
   }
```



Insertion in AVL : Implementation

(cont.)

```
    else if (bal<-1 && data < tree->right->data) {  //right-left case
        tree->right = rightRotate(tree->right);
        return leftRotate(tree);
    }

    else if (bal < -1 && data > tree->right->data)   //right-right case
        return leftRotate(tree);
    else
        return tree;
}
```

Microsoft PowerPoint - [PowerPoint Slide Show - [7_TreeCLASS]]

Insertion in AVL : Analysis

- Insert the new key as a new leaf just as in ordinary binary search tree: O(logN)
- Then trace the path from the new leaf towards the root, for each node x encountered: O(logN)
  - Check height difference: O(1)
  - If satisfies AVL property, proceed to next node: O(1)
  - If not, perform single/double rotation: O(1)

Alka Jindal