VOLUNTEER AT EOH 2026!

For EOH 2026, we're holding workshops to teach visitors important skills. We're planning on holding sessions to teach TinkerCAD, Python, and HTML & CSS. We're looking for volunteers to help teach these workshops. Furthermore, we also have many other volunteering requiring no pervious experience!

Workshops Volunteering

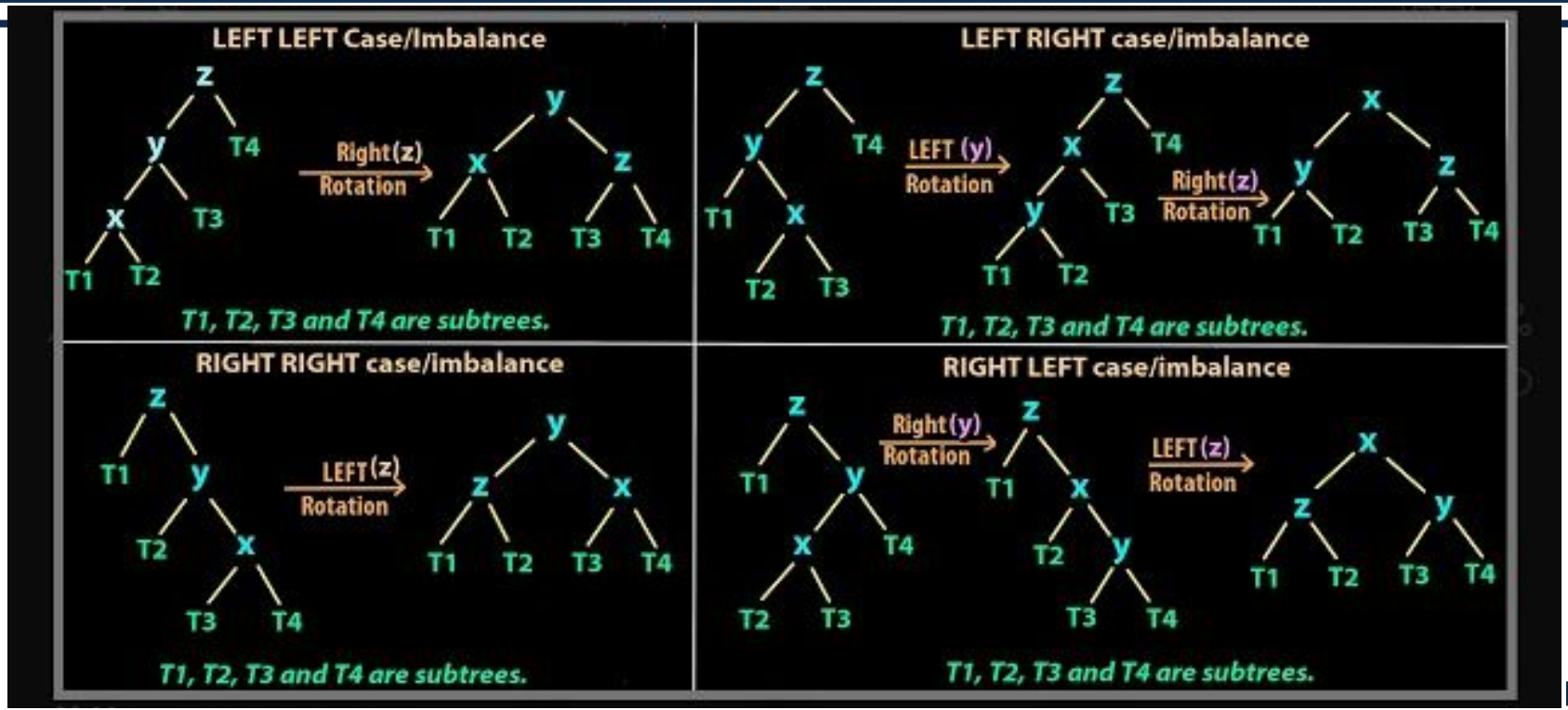
General Volunteering

AVL Tree Implementation

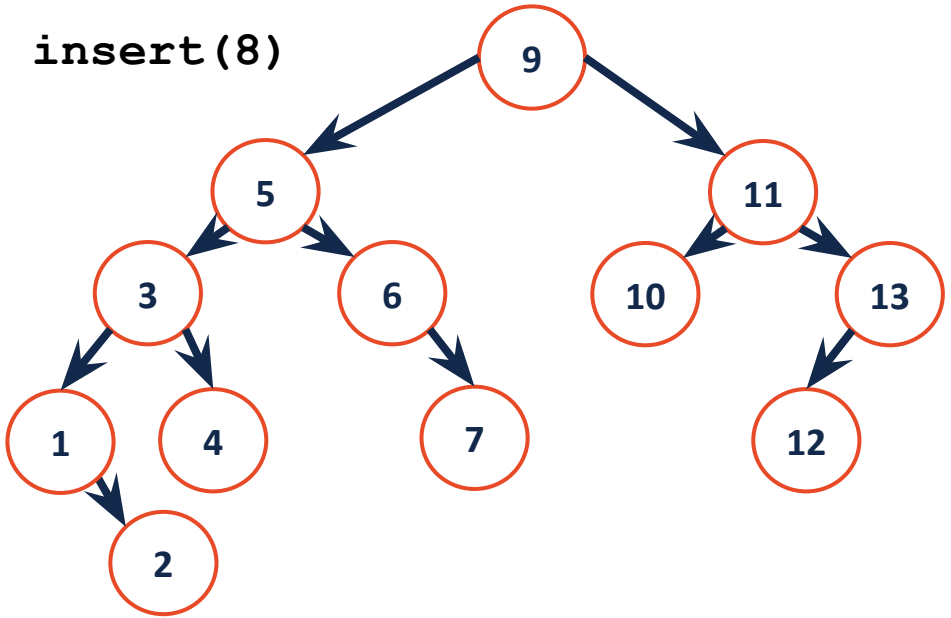
Learning Objectives

1. Implement AVL Tree Insert
2. Implement AVL Tree Rebalance
3. Implement AVL Tree Rotations

Rotations Summary



Insert in an AVL Tree

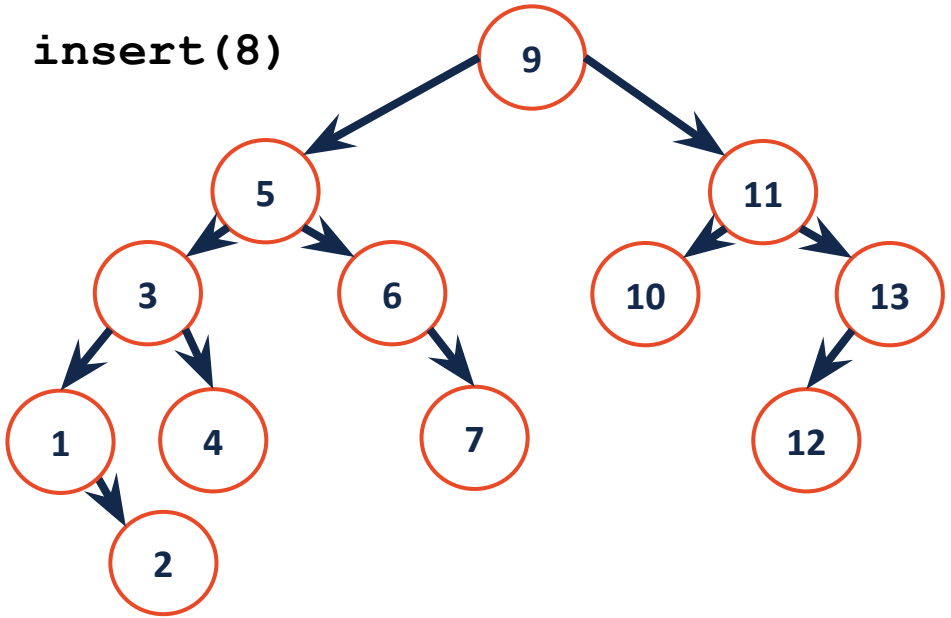



```
1  struct TreeNode {
2    T key;
3    unsigned height;
4    TreeNode *left;
5    TreeNode *right;
6  };
```

Insert in an AVL Tree

Insert (pseudo code):

1: Insert at proper place
2: Check for imbalance
3: Rotate, if necessary
4: Update height

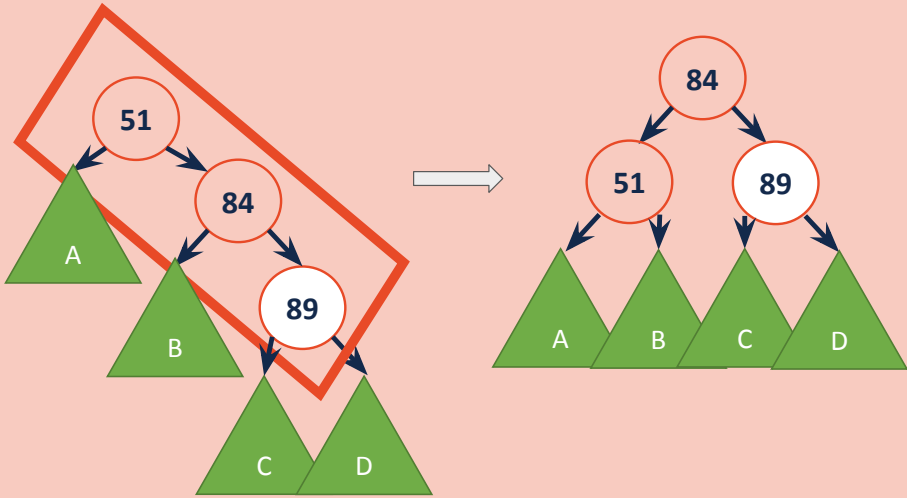



```
1  struct TreeNode {
2    T key;
3    unsigned height;
4    TreeNode *left;
5    TreeNode *right;
6  };
```

```
1  template <typename K, typename V>
2  void AVL<K, V>::_insert(const K & key, const V & data,
3  TreeNode *& cur) {
4    if (cur == NULL){
5      cur = new TreeNode(key, data);
6    } else if (key < cur->key) {
7      _insert( key, data, cur->left );
8    } else if (key > cur->key) {
9      _insert( key, data, cur->right );
10   }
11   _ensureBalance(cur);
12 }
13
```

```
1  template <typename K, typename V>
2  void AVL<K, V>::_ensureBalance(TreeNode *& cur) {
3    // Calculate the balance factor:
4    int balance = height(cur->right) - height(cur->left);
5
6    // Check if the node is currently not in balance:
7    if ( balance == -2 ) {
8      int l_balance =
9            height(cur->left->right) - height(cur->left->left);
10     if ( l_balance == -1 ) { _____________________________; }
11     else                   { _____________________________; }
12   } else if ( balance == 2 ) {
13     int r_balance =
14           height(cur->right->right) - height(cur->right->left);
15     if( r_balance == 1 ) { _____________________________; }
16     else                 { _____________________________; }
17   }
18
19   _updateHeight(cur);
20 };
21
```

```
1  template <typename K, typename V>
2  void AVL<K, V>::rotateLeft(TreeNode *& cur) {
3    // Modify Pointers
4
5
6
7
8
9
10
11
12
13
14
15
16   // Update Heights
17
18
19
20
21 };
```



```
1  template <typename K, typename V>
2  void AVL<K, V>::_remove(const K & key, TreeNode *& cur) {
3      // Base Case: If cur is null (element isn't there)
4
5      // If key is less than current key,
6          // remove in left subtree
7
8      // If key is greater than current key,
9          // remove in right subtree
10
11     // If key is equal to current key,
12         // 0 Child Case
13
14         // 1 Child Case
15
16         // 2 Child Case
17 }
```

Which node could I remove to cause multiple rotations?