CS 225

Data Structures

Feb. 28 – AVL Trees

Wade Fagen-Ulmschneider

Course Logistics Update

CBTF exams will go on as-scheduled:

- Theory Exam 2 is ongoing
- Sample Exam available on PL

MPs and Lab assignments will be due on schedule:

- MP4 is released; due March 12, 2018
- lab_huffman is released later today

My office hours are cancelled today.

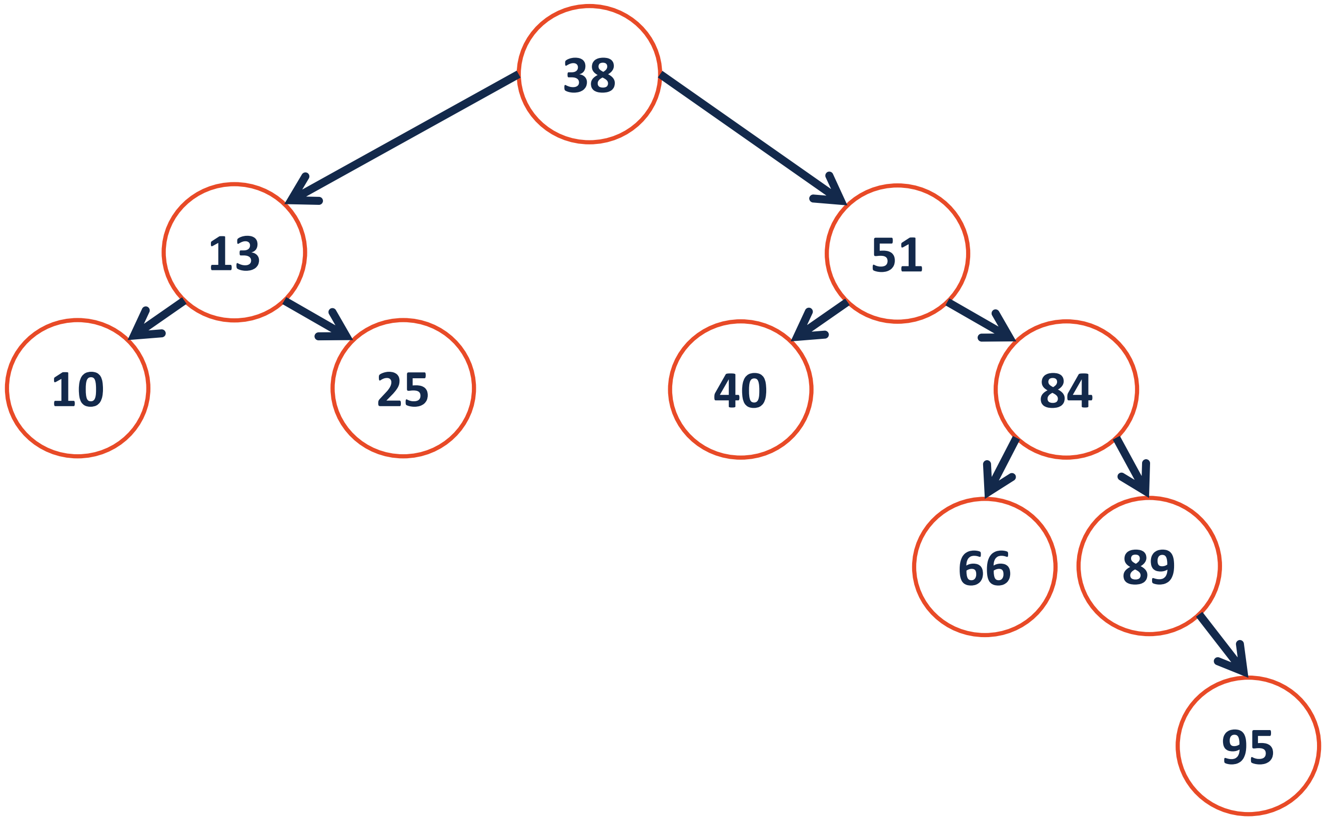
Lab Sections

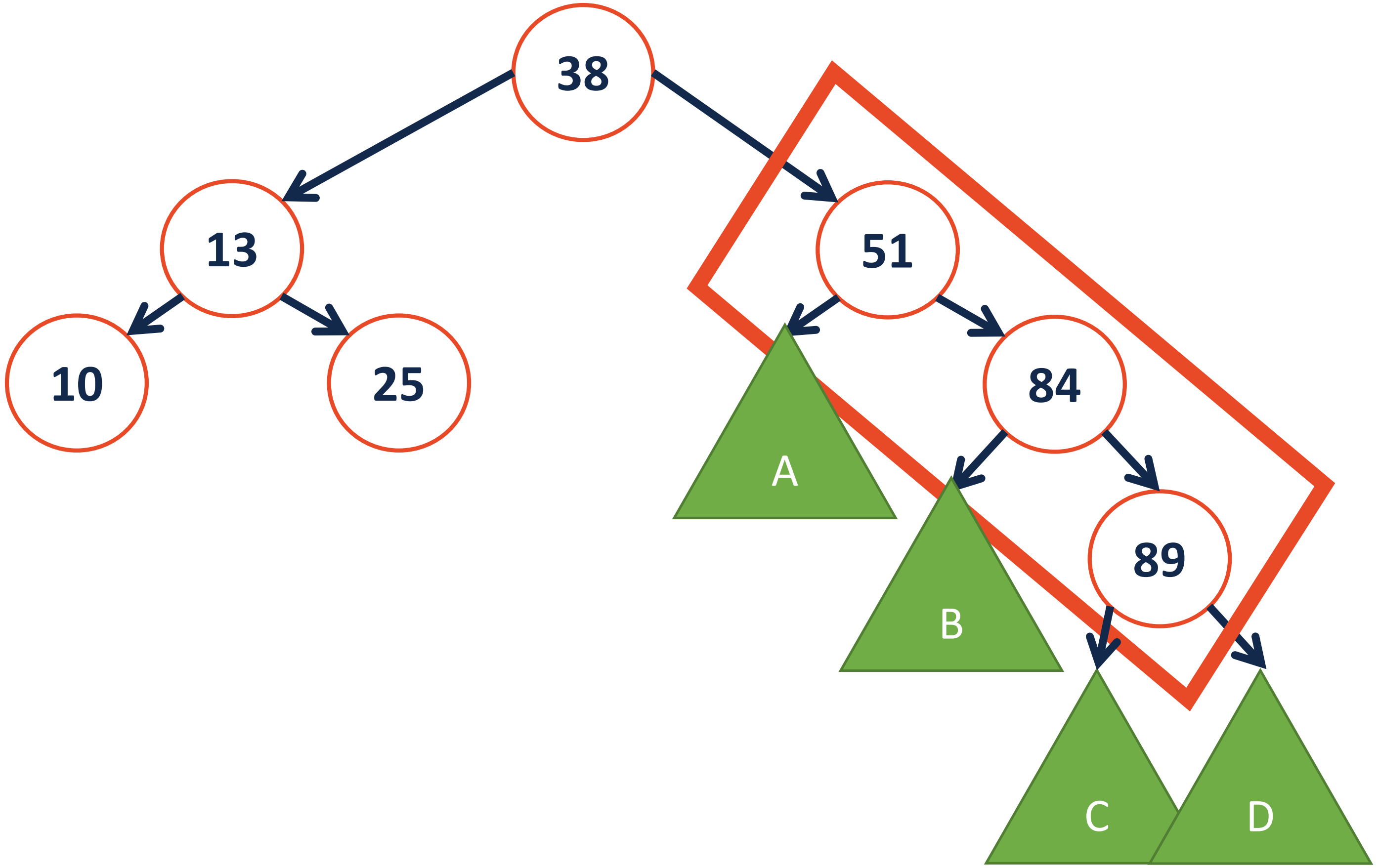
All lab sections are **not** meeting this week.

Instead, all CAs and non-striking TAs will hold open office hours (using the regular queue, held in the basement):

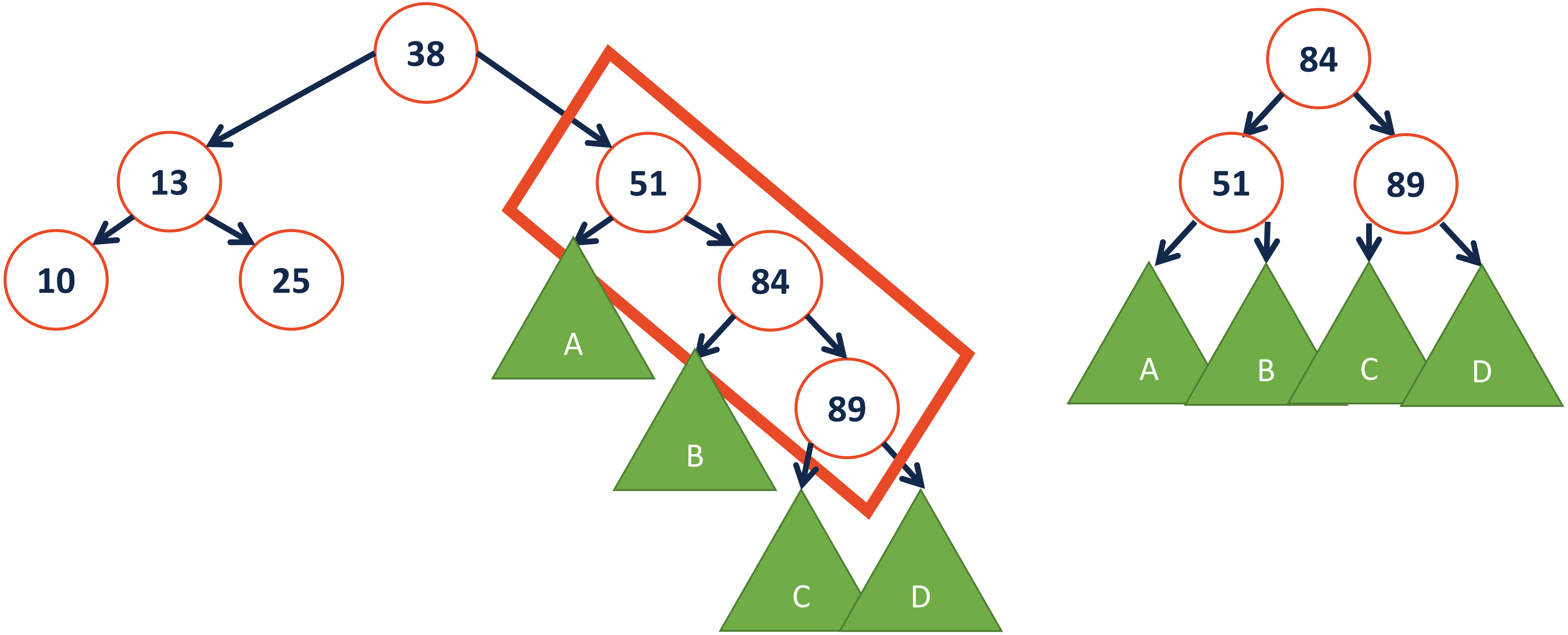
- Feel free to use the room to work with your peers on the lab. Staff will be available in open office hours in the basement of Siebel.
- *An intro video on Huffman trees will be provided.*

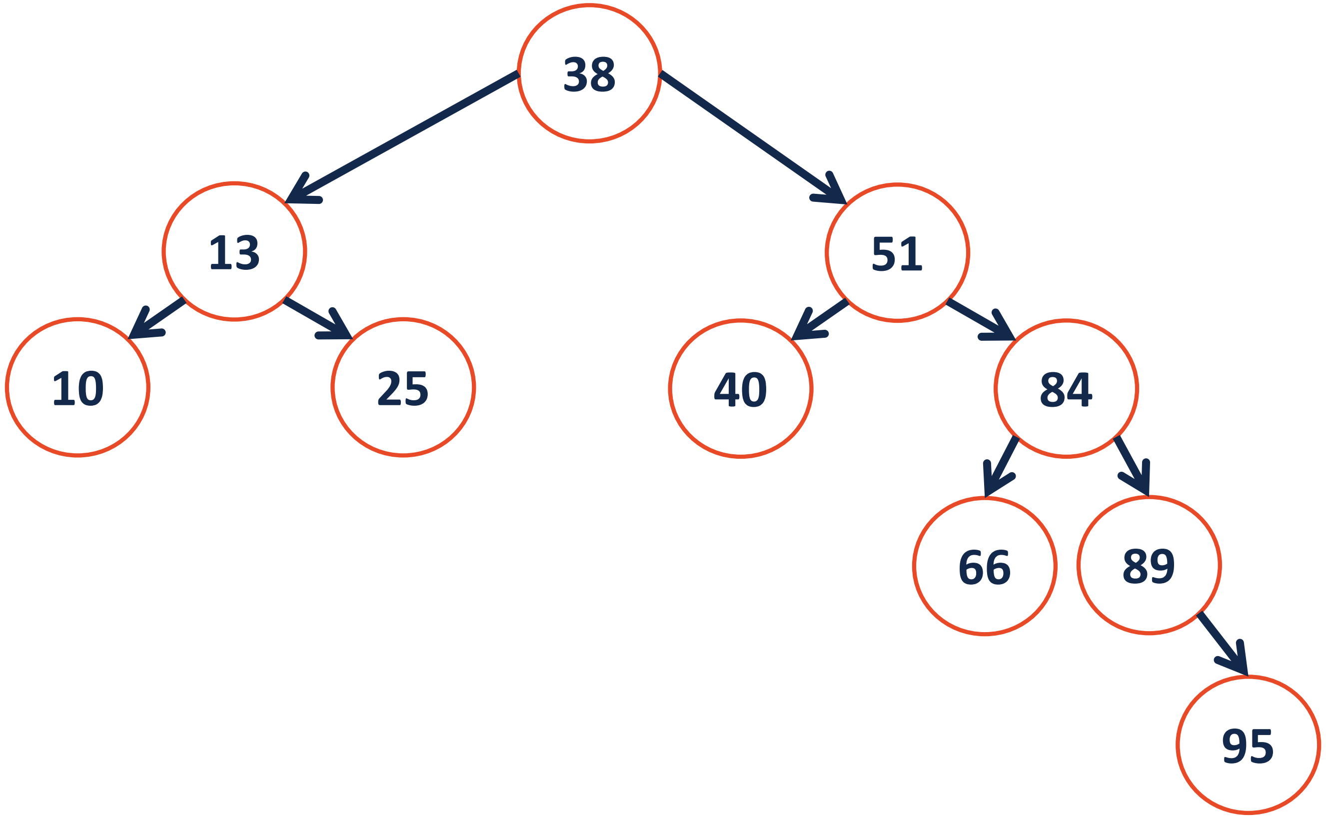
Left Rotation

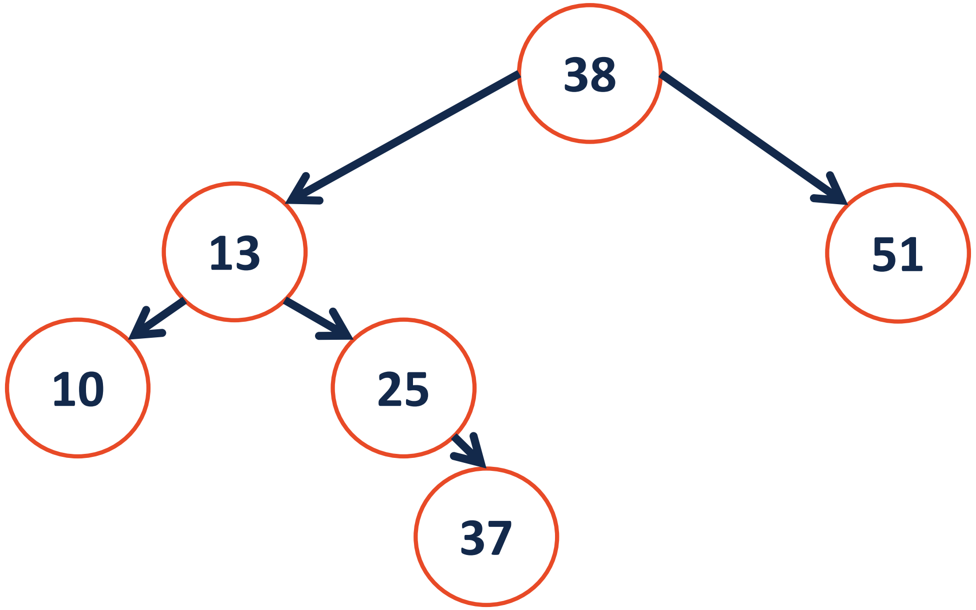


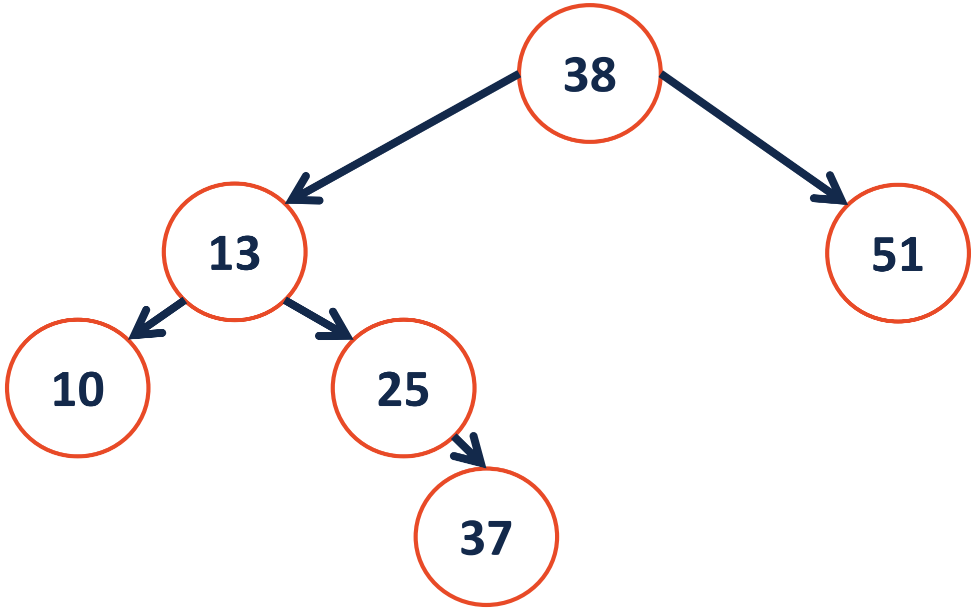


38
13
51
10
25
84
89
A
B
C
D


38
13
51
10
25
84
89
A
B
C
D
84
51
89
A
B
C
D


38
13
51
10
25
40
84
66
89
95


38
13
51
10
25
37


38
13
51
10
25
37

BST Rotation Summary

- Four kinds of rotations (L, R, LR, RL)
- All rotations are local (subtrees are not impacted)
- All rotations are constant time: O(1)
- BST property maintained

GOAL:

We call these trees:

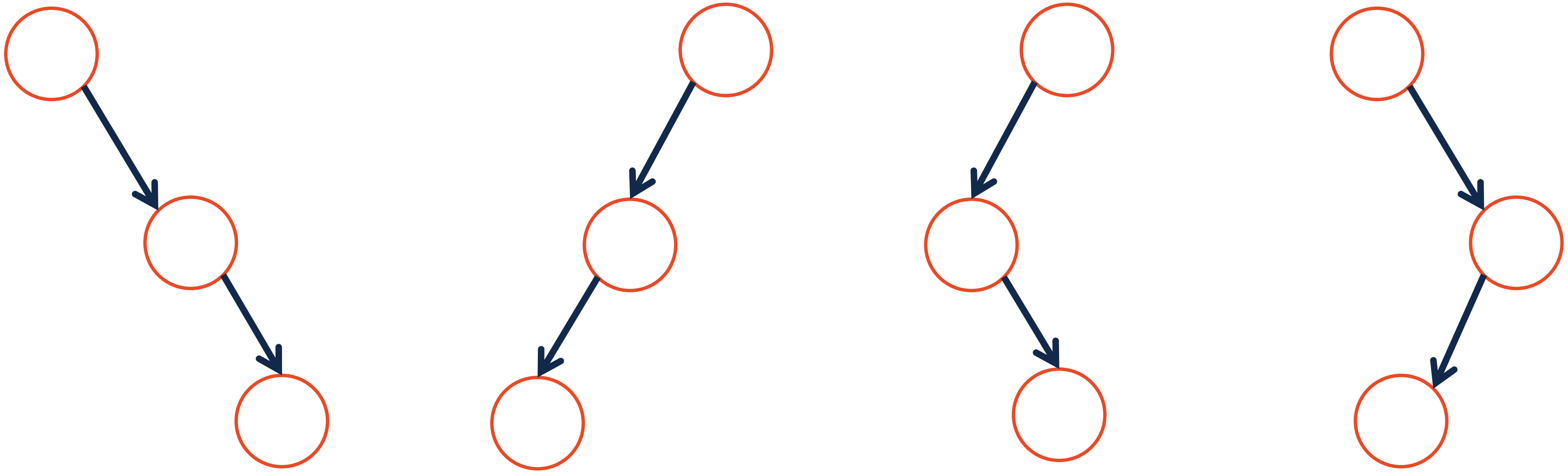
AVL Trees

Three issues for consideration:

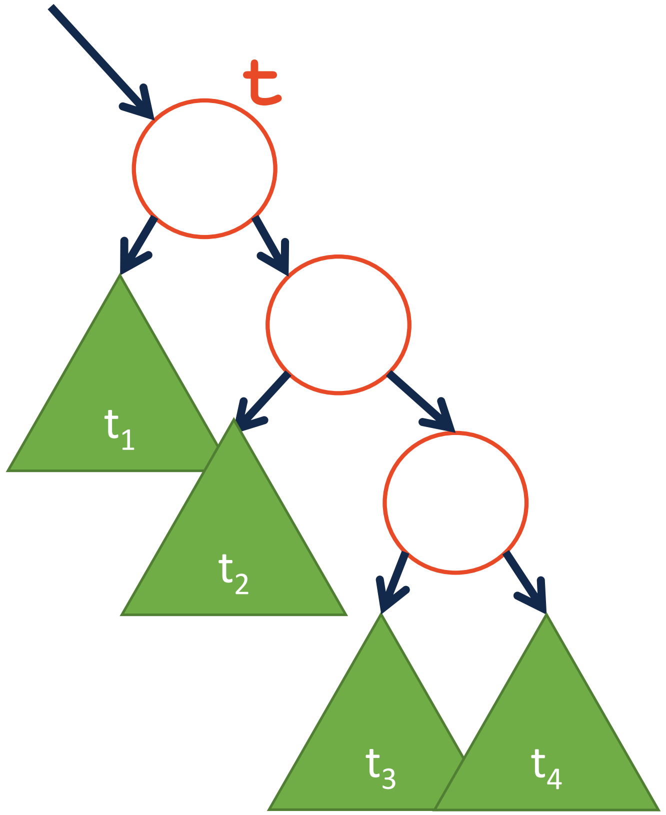
- Rotations
- Maintaining Height
- Detecting Imbalance

AVL Tree Rotations

Four templates for rotations:



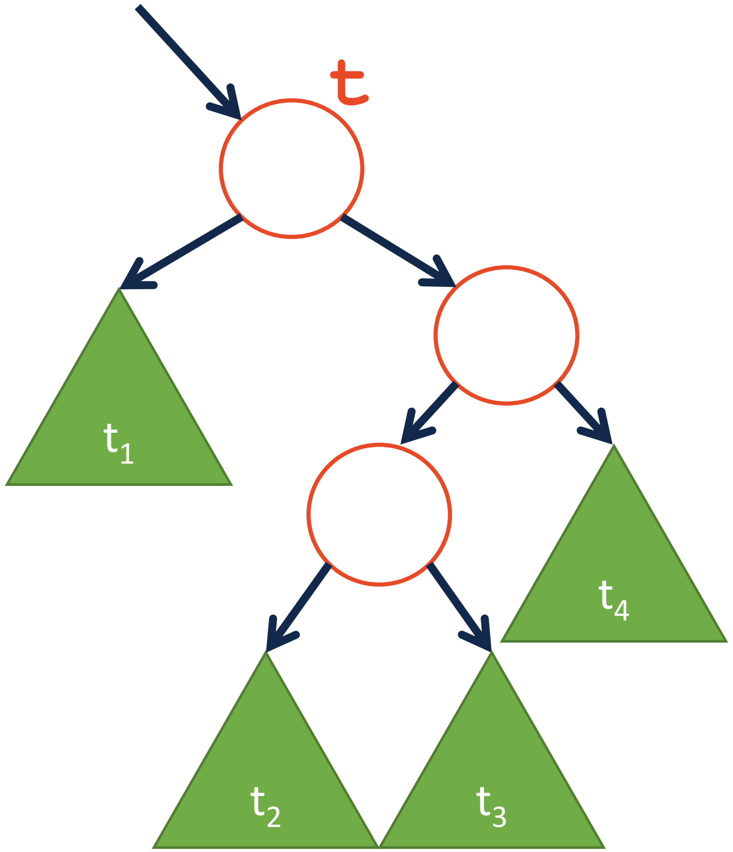
Finding the Rotation



Theorem:
If an insertion occurred in subtrees **t_3** or **t_4** and a subtree was detected at **t**, then a ___________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t->right** is ______.

Finding the Rotation



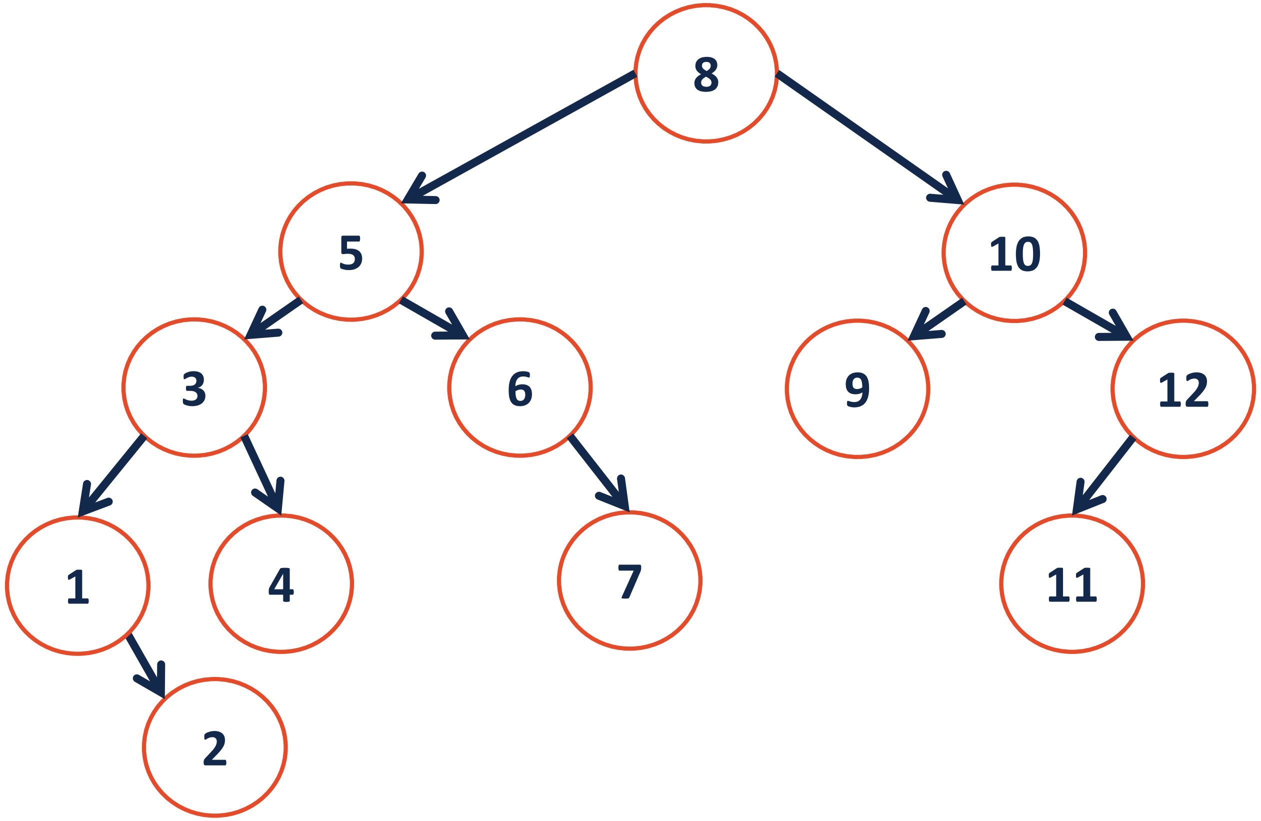
Theorem:
If an insertion occurred in subtrees **t_2** or **t_3** and a subtree was detected at **t**, then a ___________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t->right** is ______.

Insertion into an AVL Tree

`_insert(6.5)`

```
struct TreeNode {
  T key;
  unsigned height;
  TreeNode *left;
  TreeNode *right;
};
```

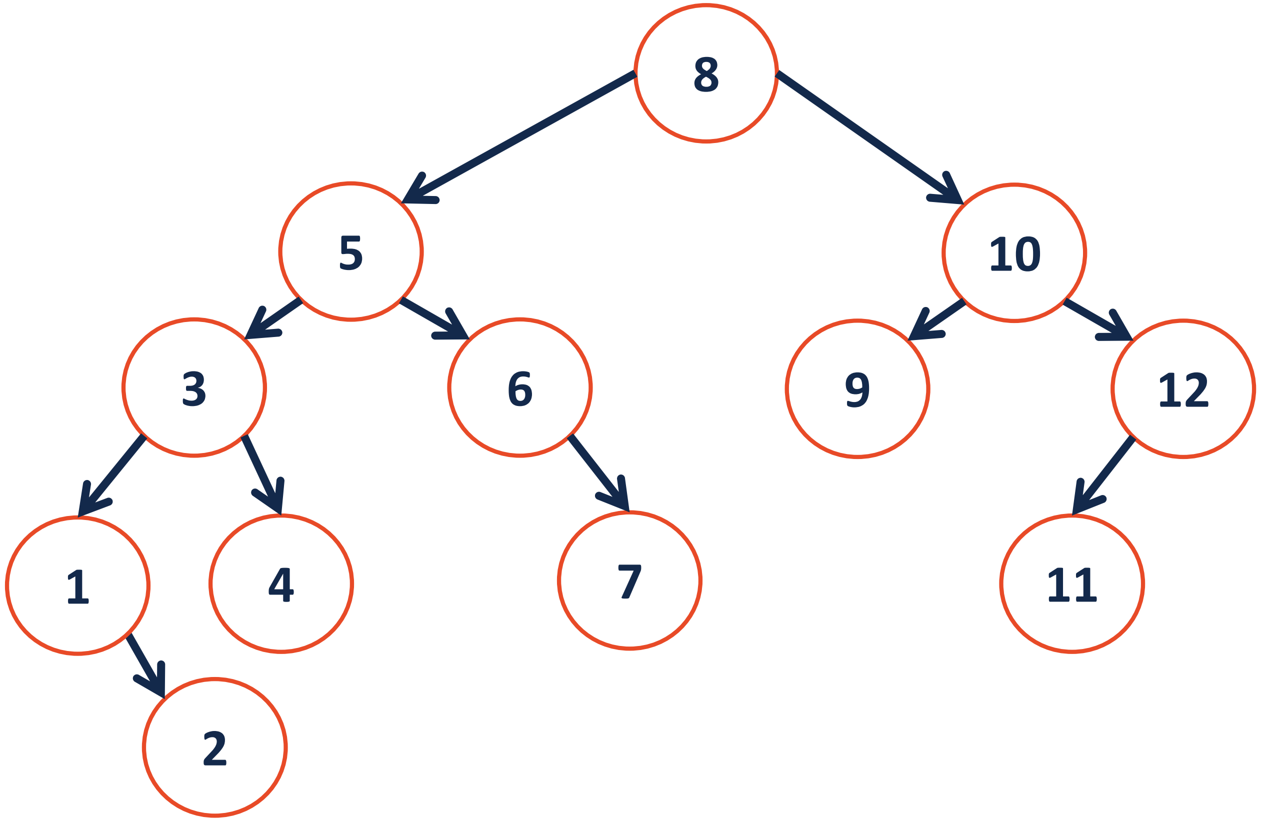


Insertion into an AVL Tree

`_insert(6.5)`

Insert (pseudo code):

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

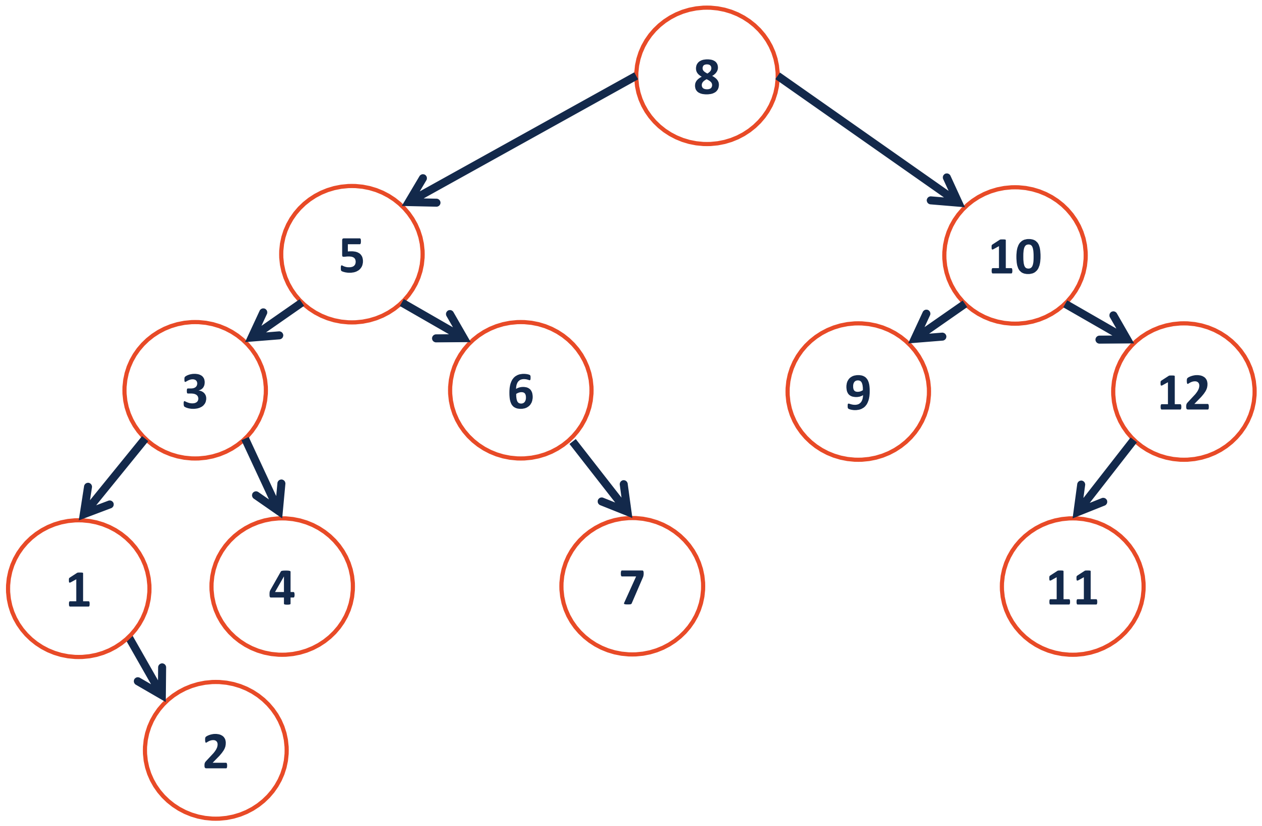



```
struct TreeNode {
  T key;
  unsigned height;
  TreeNode *left;
  TreeNode *right;
};
```

```
template <class T> void AVLTree<T>::_insert(const T & x, treeNode<T> * & t ) {
  if( t == NULL ) {
    t = new TreeNode<T>( x, 0, NULL, NULL);
  }

  else if( x < t->key ) {
    _insert( x, t->left );
    int balance = height(t->right) - height(t->left);
    int leftBalance = height(t->left->right) - height(t->left->left);
    if ( balance == -2 ) {
      if ( leftBalance == -1 ) { rotate_____________( t ); }
      else                     { rotate_____________( t ); }
    }
  }

  else if( x > t->key ) {
    _insert( x, t->right );
    int balance = height(t->right) - height(t->left);
    int rightBalance = height(t->right->right) - height(t->right->left);
    if( balance == 2 ) {
      if( rightBalance == 1 ) { rotate_____________( t ); }
      else                    { rotate_____________( t ); }
    }
  }

  t->height = 1 + max(height(t->left), height(t->right));
}
```

Height-Balanced Tree

Height balance: $b = \text{height}(T_R) - \text{height}(T_L)$


8
5
10
3
6
9
12
1
4
7
11
2

AVL Tree Analysis

We know: insert, remove and find runs in: ____________.

We will argue that: h = ___________.

AVL Tree Analysis

Definition of big-O:

…or, with pictures: