Today
- Quiz 9
- AVL Trees

Assignments
- HW 7 due
- HW 8 out
- HW 9 (optional) out
General case for single rotation (**left–left case**)

- Insertion in the **left** subtree of the **left** child
- Assumed that the tree is balanced before the rotation
- And becomes unbalanced after the rotation

Requires a "**right**" rotation:

After insert and rotation, done with rotations

- Subtree now rooted at $k_1$ is back to original height
- Since subtree is now balanced, no other rotations needed
Exercise: Fill in the following helper function:

```c
// return new subtree root after rotation
Node* rotate_right(Node* k2)
{
    Node* k1 = k2->left;
    k2->left = k1->right;
    k1->right = k2;
    return k1;
}
```
General case for single rotation (*right–right case*)

- Insertion in the *right* subtree of the *right* child
- The “mirror” image of the left–left case

Requires a “*left*” rotation:

![Diagram showing single rotation (right–right case)]
**Exercise:** Fill in the following helper function:

```c
// return new subtree root after rotation
Node* rotate_left(Node* k2)
{
    Node* k1 = k2->right;
    k2->right = k1->left;
    k1->left = k2;
    return k1;
}
```
A single (right or left) rotation might *not* rebalance the tree ... 

- In this case, the right rotation didn’t help!!!
- This is because we inserted into the *left* child’s *right* subtree
To deal with this, do two rotations (right-left case)

Start with a left rotation (at \( k_2 \)):

Followed by a right rotation (at \( k_3 \)): 
The “mirror” image – right-left case

Rotate left then right:
Basic insertion algorithm:
1. Recursively traverse path for insertion
2. Insert node
3. During backtracking, update heights and determine balance factors
4. Rebalance (rotate) if node’s balance factor less than $-1$ or greater than 1

Only need to perform (single or double) rotation once on insert
- After rotation performed, rebalancing is complete
- Height changes are “local” to insertion path
- Because inserting on a path doesn’t change height of other paths

Cost of insertion
- Cost for a single or double rotation is $O(1)$
- Traversing “down” the path is $O(\log n)$ since tree is balanced
- Traversing “up” the path for rotations and height update is $O(\log n)$
- Thus insertion is $O(\log n)$

Hints for HW-8: Where each node stores its own height
1. The work is focused on insert and erase (trivially in height() function)
2. Start with getting insert to work, then do erase
3. Start by modifying insert to:
(a) work recursively
(b) get height to be updated
(c) call stubbed out `rebalance` function (just returning `st_root`)

4. Then work on the `rebalance` function

5. Then move on to `erase`
   (a) recursively find node to remove
   (b) find inorder successor interatively
   (c) copy key and value to `st_root`
   (d) call `erase` from `st_root->right` to remove inorder successor (via its key)
   (e) adjust/extend `rebalance` as needed (small issues not accounted for)
Basic idea of “backtracking”

- Backtracking occurs after a recursive call ...
Making sense of the rebalance step ...

- calls the \texttt{rotate\_left} and \texttt{rotate\_right} helpers
- updates heights due to the rotations
- trick is to determine which rotation to call and when (based on heights)

```cpp
Node* rebalance(Node* st_root)
{
    if (!st_root) // nothing to balance
        return st_root;

    Node* lptr = st_root->left;
    Node* rptr = st_root->right;

    // left but no right pointer (special case)
    if (lptr && !rptr && lptr->height > 1) {
        ... check if need double rotation ...
        ... do rotate right ...
        ... adjust height ...
    }

    // right but no left pointer (special case)
    else if (... similar to above ...) {
        ... similar to above ...
    }

    // left and right pointer, left "heavy"
    else if (...) {
        ... similar to cases above ...
    }

    // left and right pointer, right "heavy"
    else if (...) {
        ... similar to cases above ...
    }

    return st_root;
}
```
Unlike for insertion, deletion may require multiple rebalance steps

- may need to keep applying rotations as we navigate back “up” the path