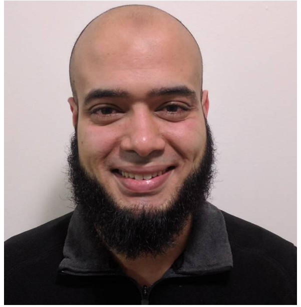
Data Structures

AVL Insertion

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Insertion

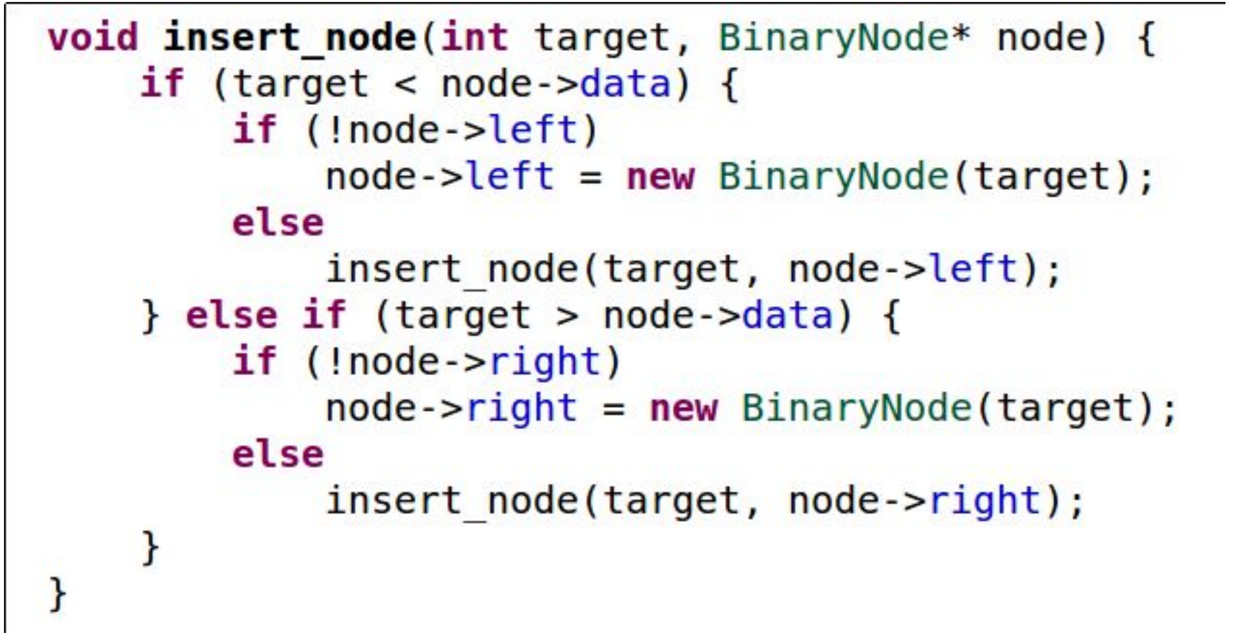
- To insert, we insert normally
- But, we need to fix any corruption (|BF| > 1) immediately
- To do so, in end of function, we update height then call balance which will check if |BF| > 1 or not
- We need also to update left & right as they might be changed
- We will follow the last homework code (rewritten BST with extra node struct)

Insertion: before

- This is our insertion (before change)
- Observe: it assumes tree nodes never rebalanced (changed)
- We need to balance

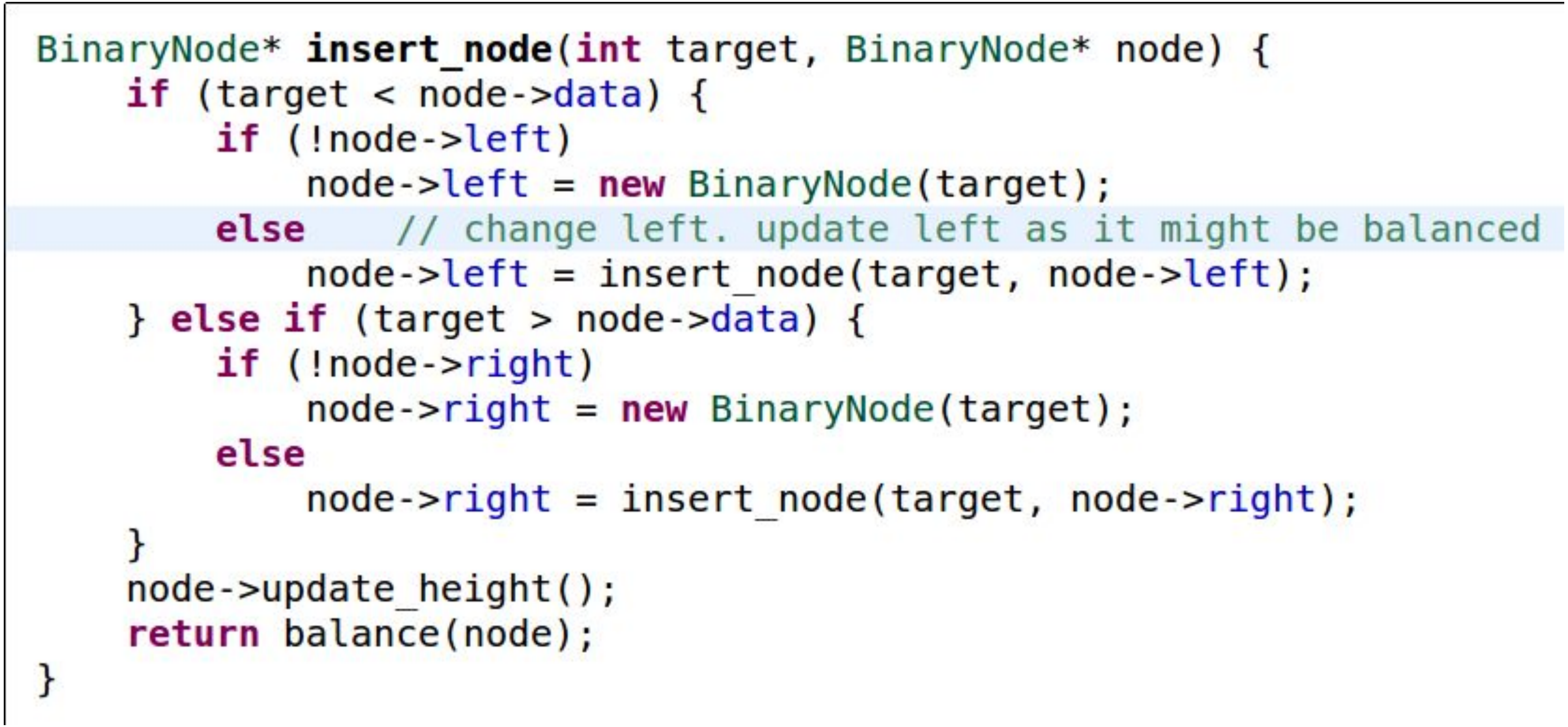
  And

  Make code flexible



```
void insert_node(int target, BinaryNode* node) {
    if (target < node->data) {
        if (!node->left)
            node->left = new BinaryNode(target);
        else
            insert_node(target, node->left);
    } else if (target > node->data) {
        if (!node->right)
            node->right = new BinaryNode(target);
        else
            insert_node(target, node->right);
    }
}
```

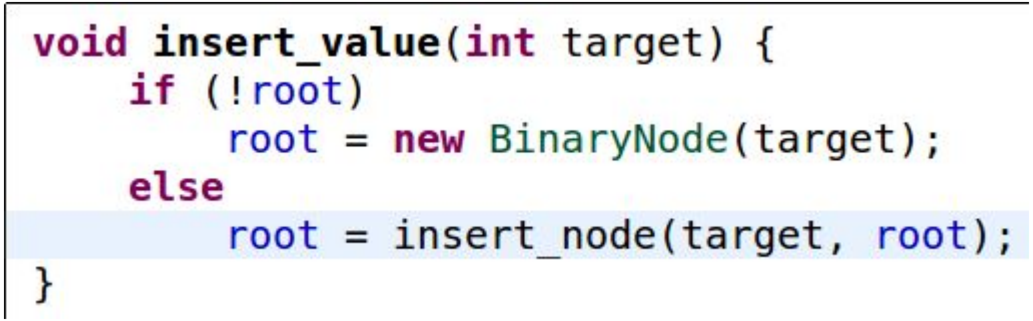
Insertion: after



```
BinaryNode* insert_node(int target, BinaryNode* node) {
    if (target < node->data) {
        if (!node->left)
            node->left = new BinaryNode(target);
        else    // change left. update left as it might be balanced
            node->left = insert_node(target, node->left);
    } else if (target > node->data) {
        if (!node->right)
            node->right = new BinaryNode(target);
        else
            node->right = insert_node(target, node->right);
    }
    node->update_height();
    return balance(node);
}
```

Insertion: caller

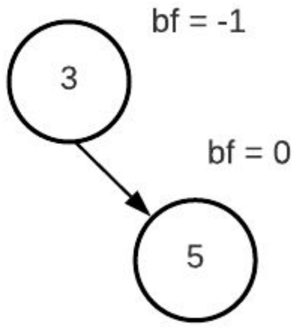
- The root can be changed from the caller, so consider that



```
void insert_value(int target) {
    if (!root)
        root = new BinaryNode(target);
    else
        root = insert_node(target, root);
}
```

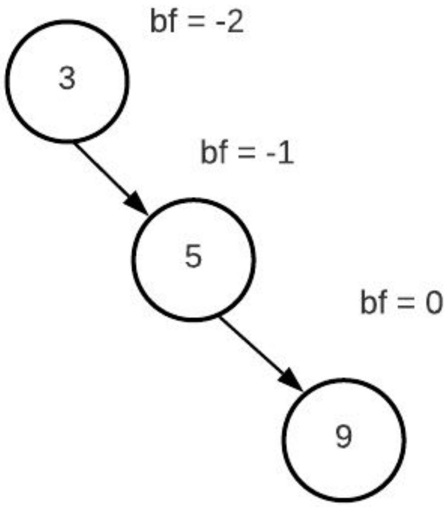
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Let’s insert the first 2 values
- No problems so far



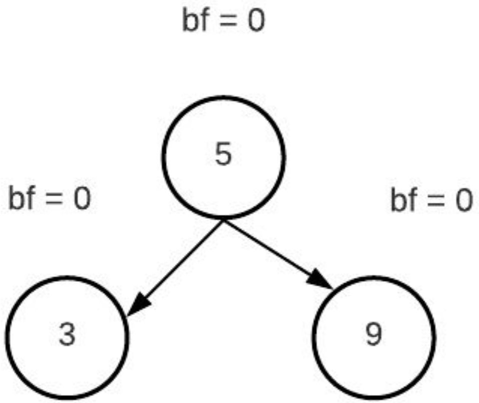
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- With 9, node(3) is imbalance
- Right-Right case
- Do Left rotation at 3



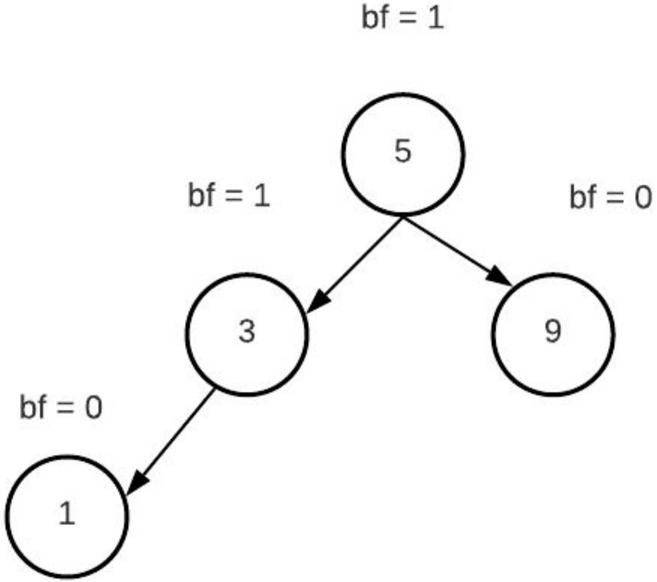
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Now fixed



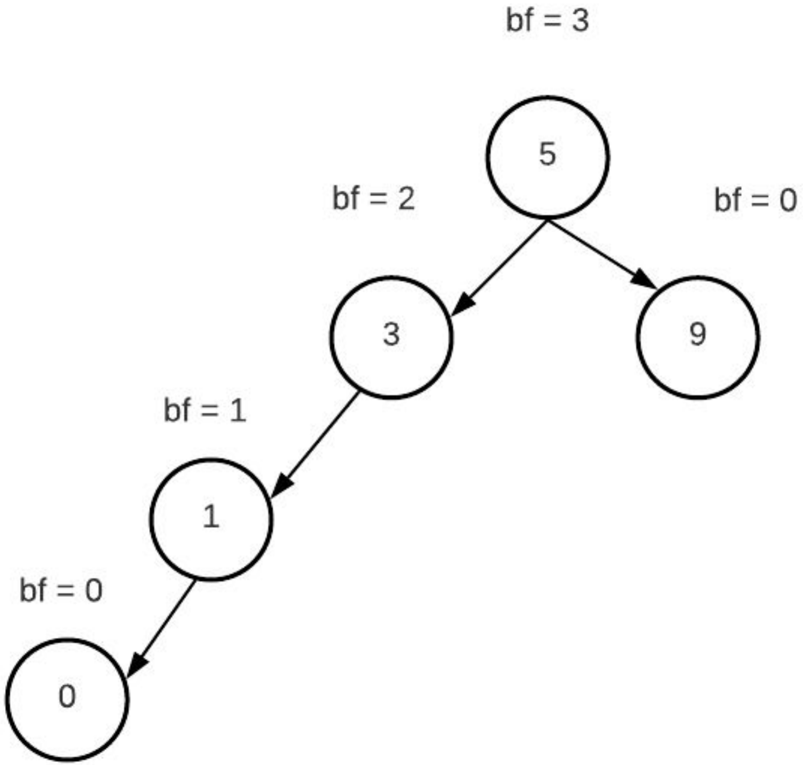
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Still good with 1



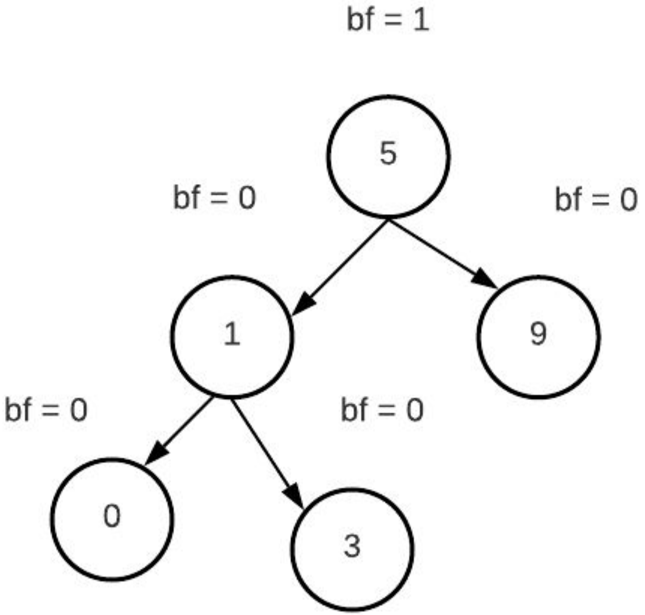
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- With 0, node(3) is imbalance
- Left-Left case
- Do right rotation at 3



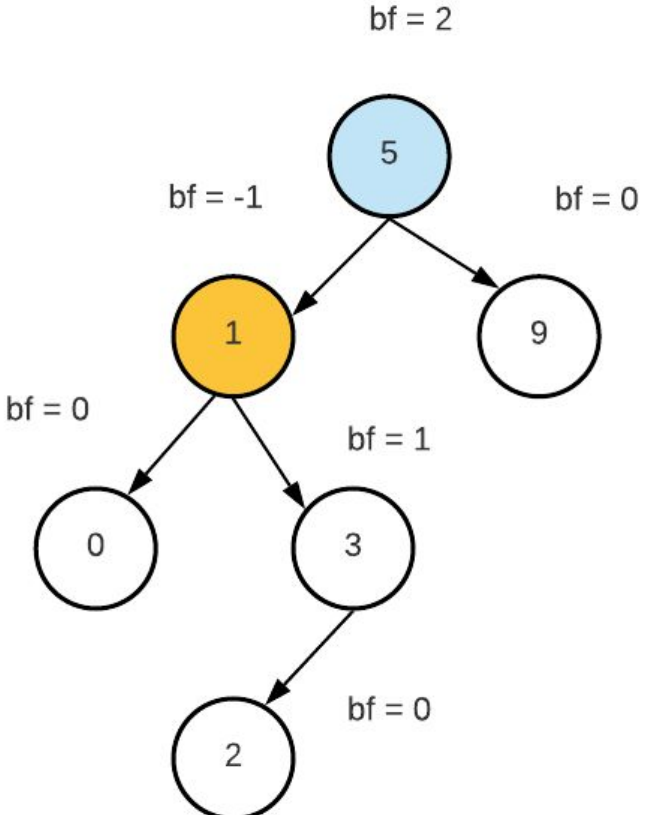
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Now fixed



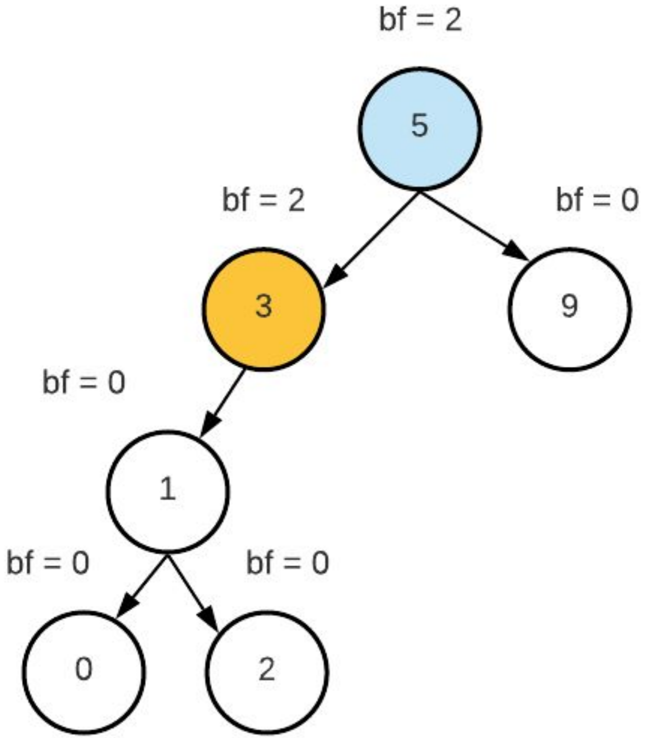
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- With 2, node(5) is imbalance
- Left-Right case
  - Bf: 2 -1
- First perform left_rotation(1)
  - This pushes 1 down and 3 up
  - A=0, B=2, C=Null
    - So B’s parent will change from 3 to 1
- **Then** perform right_rotation(5)



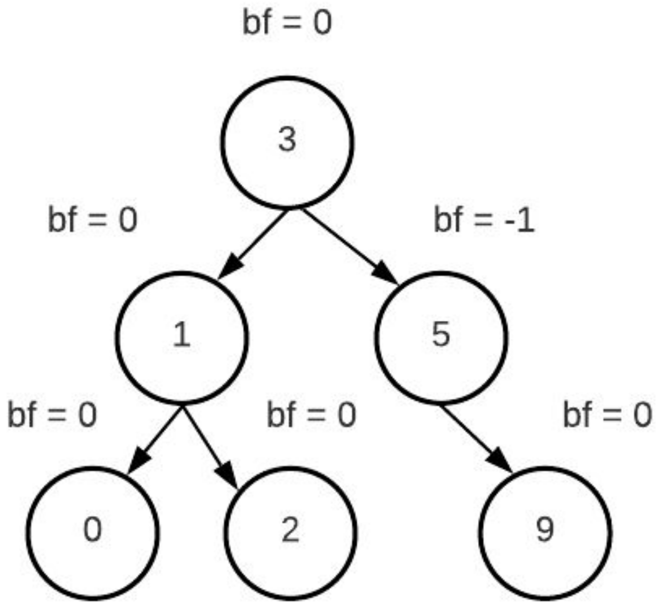
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Now 3 is left-left case
    - Observe: bf(3) = 2
    - Don’t let that confuse u
- **Remaining** perform right_rotation(5)
    - This pushes 5 down and 3 up
    - A=1, B=null, C=9



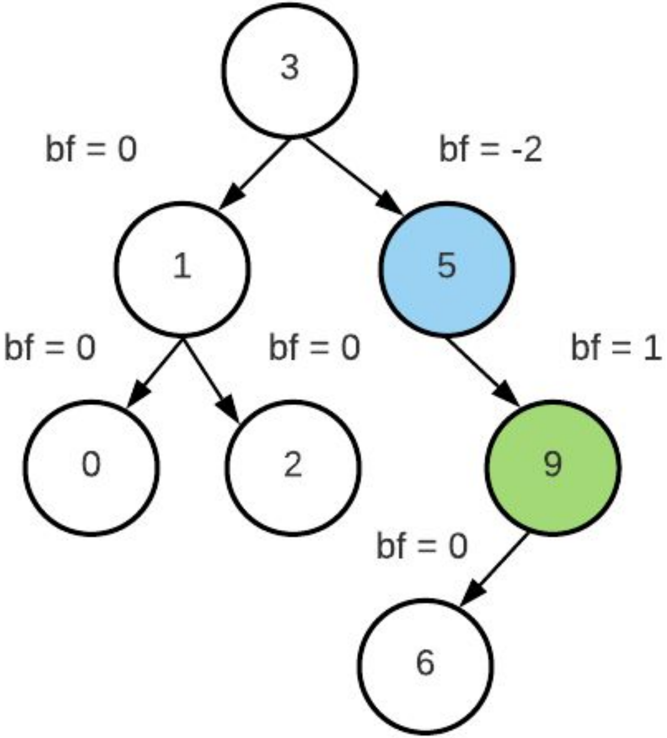
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Now fixed



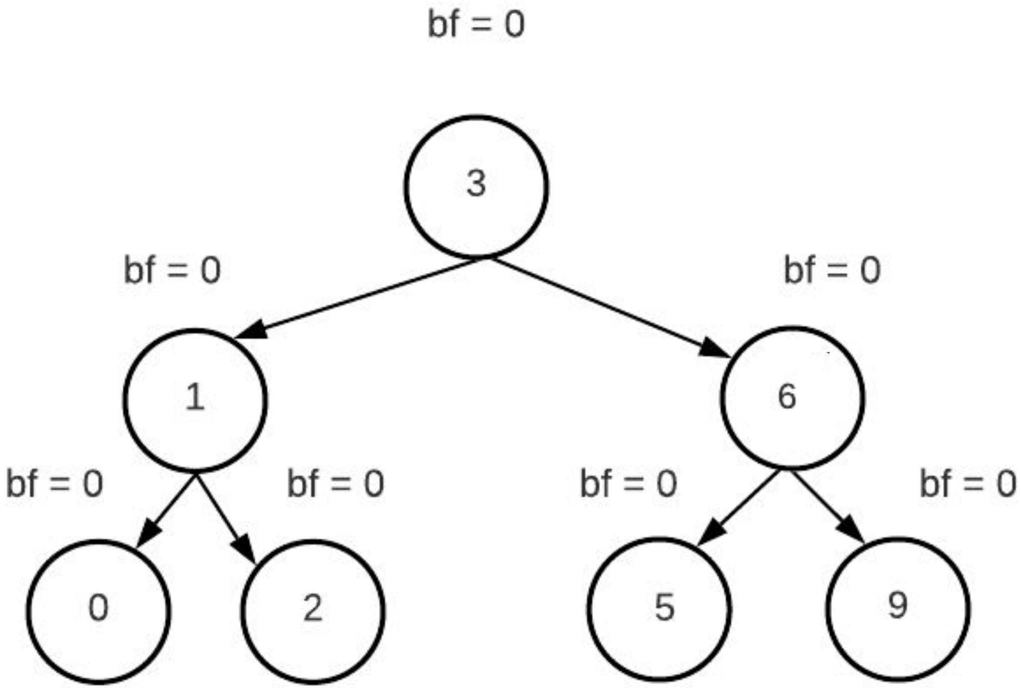
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- After inserting 6, we have right-left case
  - Bf: -2, 1
- Right-rotation(9) to convert to right-right
- Then Left-Rotation(5)
- As no children A/B/C this is visually direct



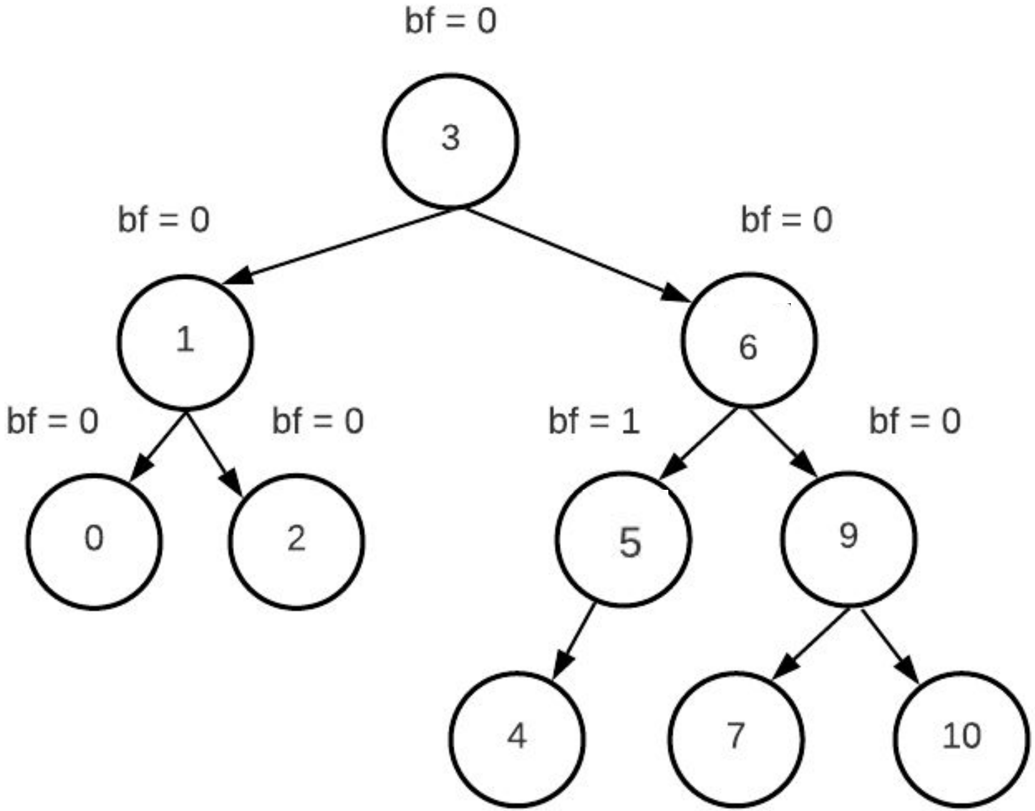
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Now fixed



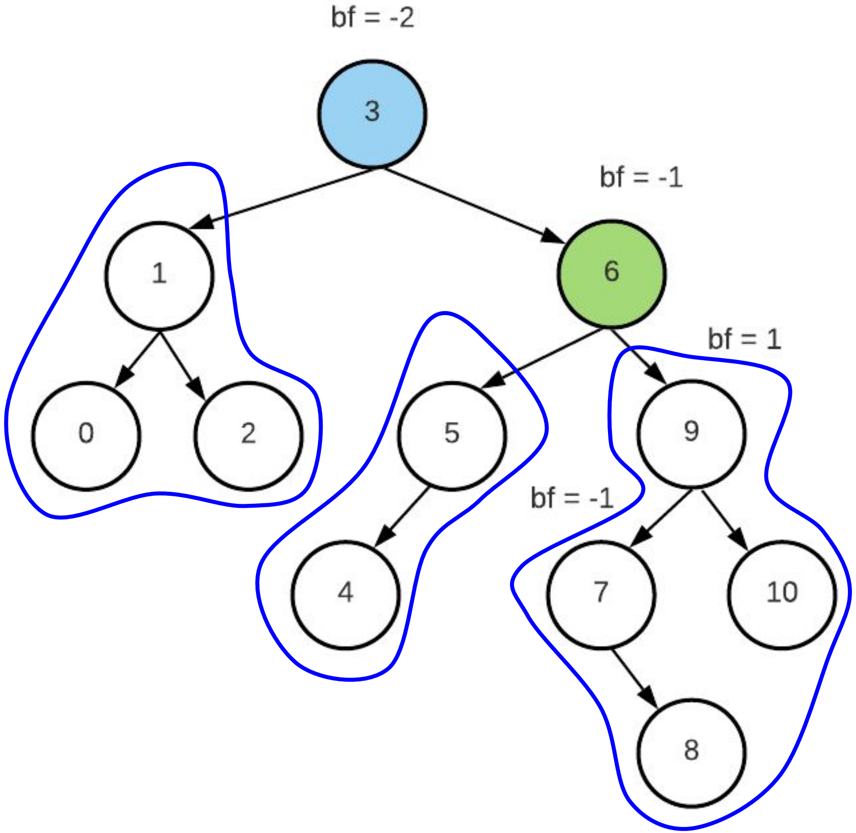
Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- Inserting next 3 values cause no problem

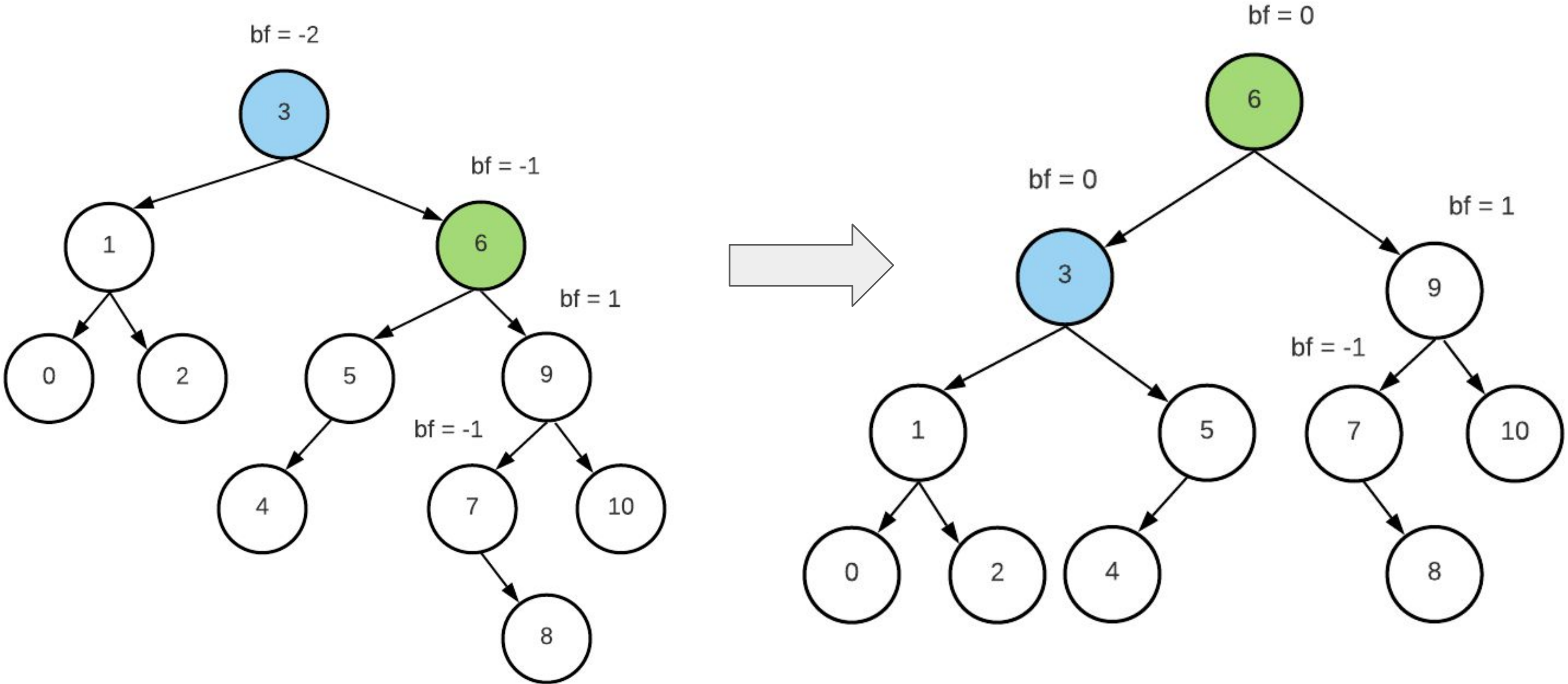


Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8

- After inserting 8, we have right-right case
  - Bf: -2, -1
- Left-rotation(3)
  - Gets 3 down and 6 up
  - A = subtree(1)
  - B = subtree(5)
    - It will change from left of 6 to right of 3
  - C = subtree(9)



Insert: 3, 5, 9, 1, 0, 2, 6, 10, 7, 4, 8



“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”