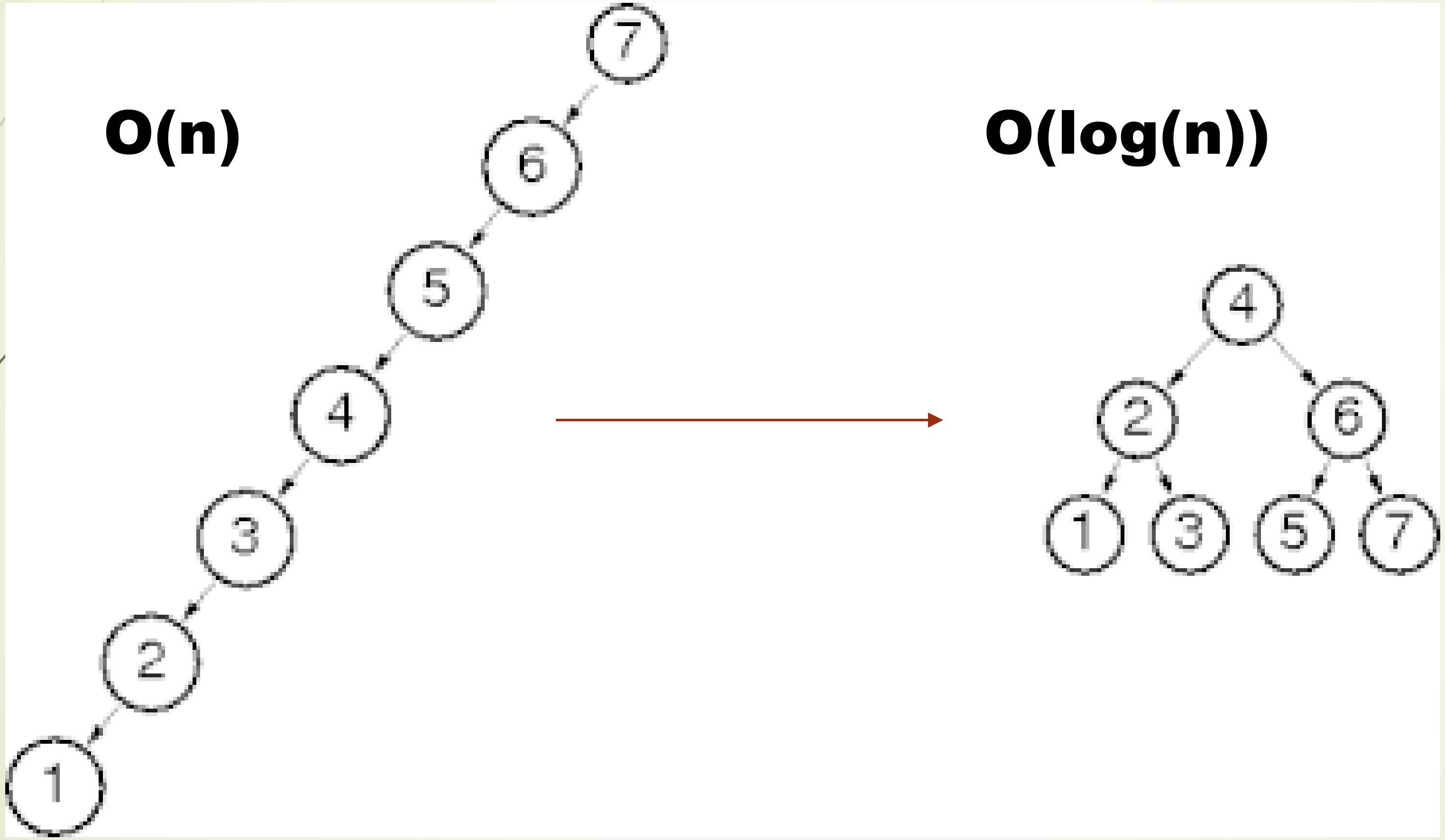
AVL Tree

Pittsford Sutherland Programming Club

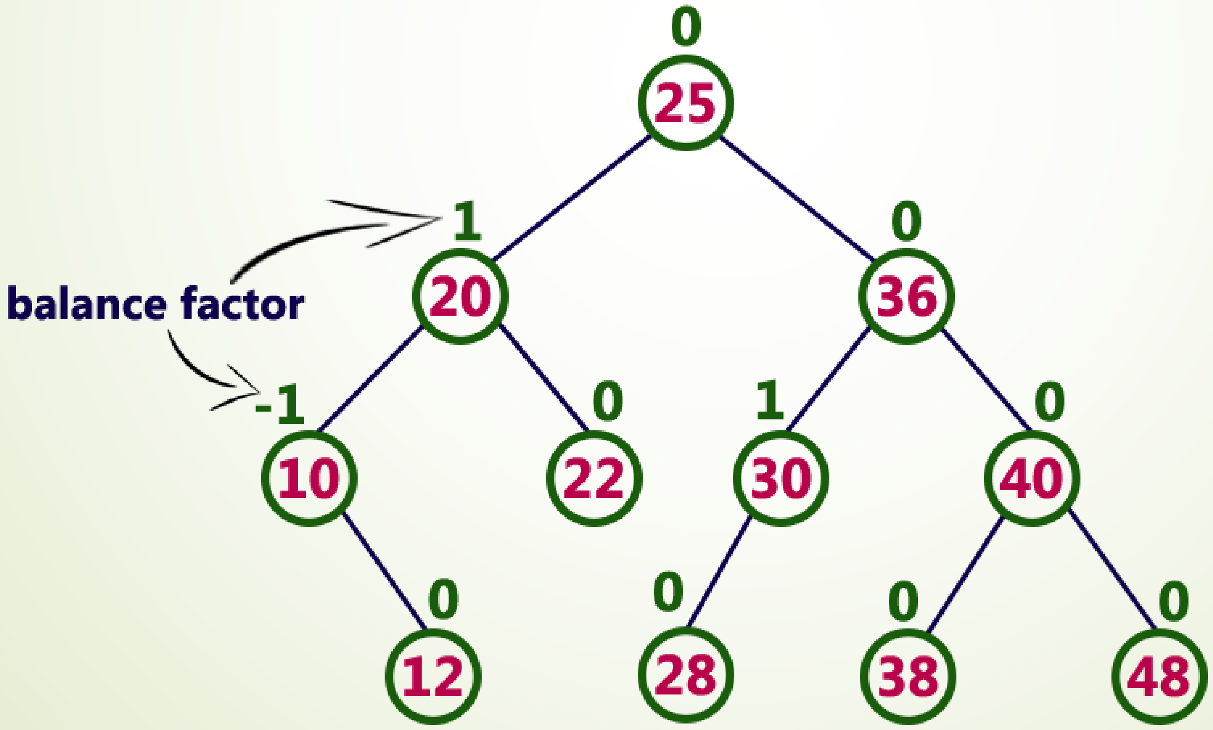
By Yizuo Chen, Yiyou Chen

What is “Balance”?



AVL Tree

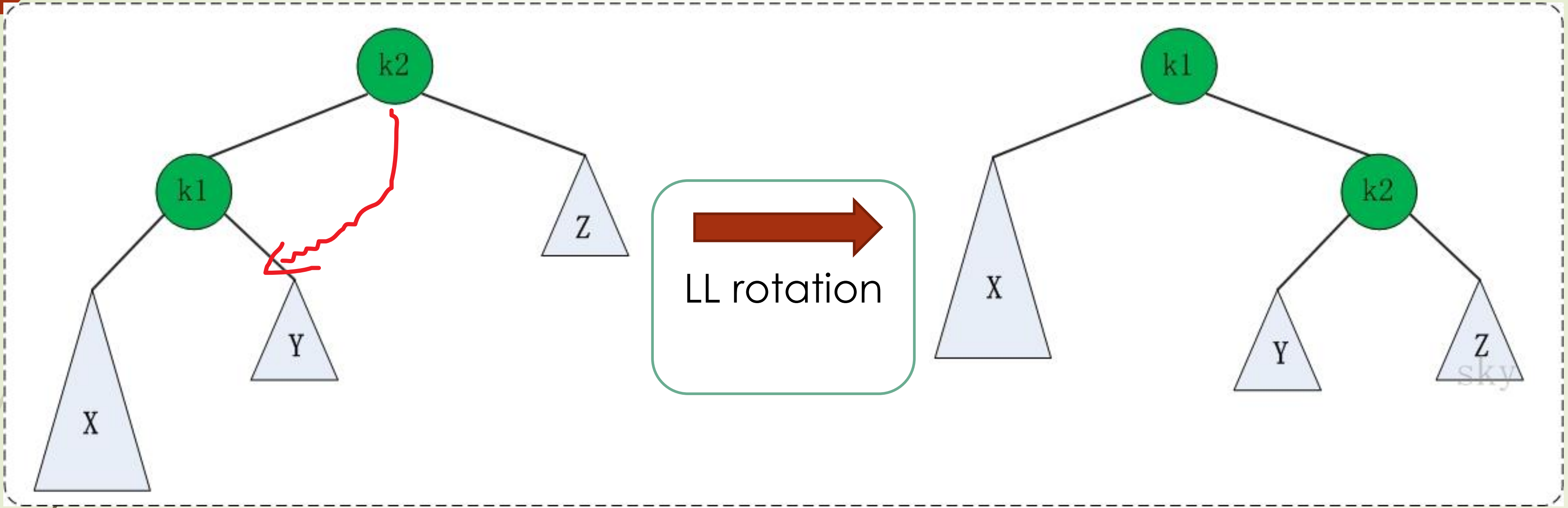
- A binary search tree with a balance condition.
- The height of left subtree and the height of right subtree differs at most 1.
- The height of an empty tree is defined to be -1.
- Balance factor saves the difference between the height of two subtrees



How to keep it “Balanced”?

Use Rotations!

Left-Left(LL) Rotation

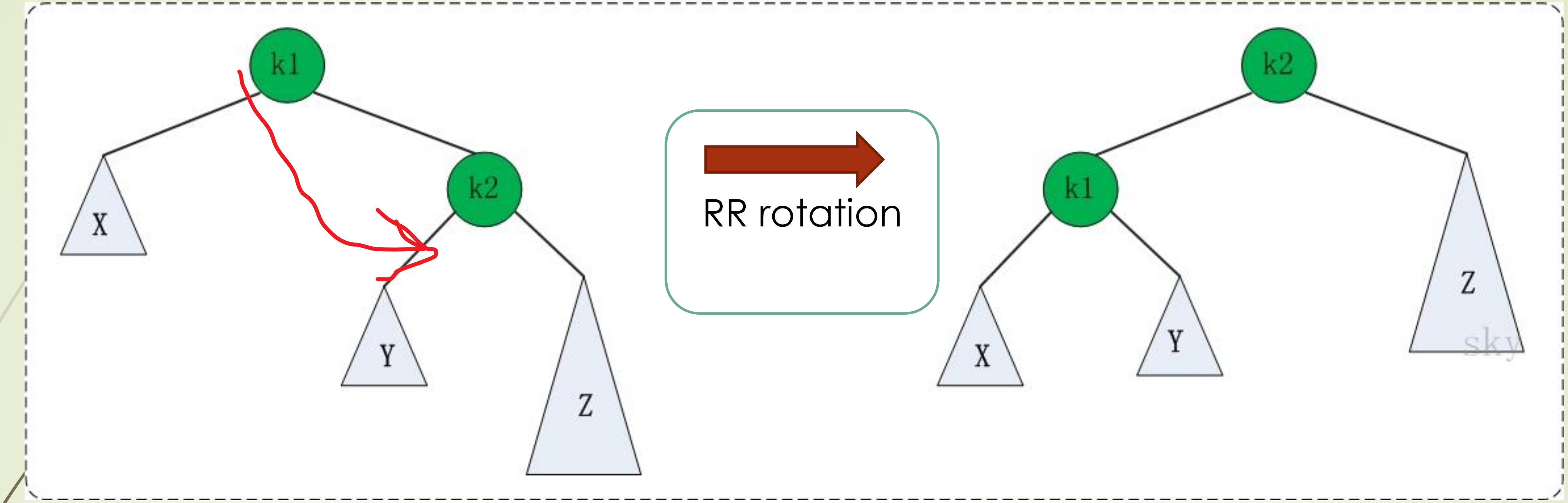


Condition: When root's left subtree is greater than its right subtree, and its left son's left subtree is greater than its left son' right subtree. (H(k1) > H(Z), H(X) > H(Y)))

How it works: We want to rotate k2 to be the right son of k1 and make k1 the root in order to make the tree more balanced.

We cannot directly make k2 the right son of k1 since k1 has already had its right subtree Y. According to the property of binary search tree, all the elements in subtree Y must be less than k2, so we can make Y be the left subtree of k2. Also, all the elements in Y, k2, and Z are greater than k1, so we can put all these elements in the right subtree of k1, and remain X unchanged.

Right-Right(RR) Rotation

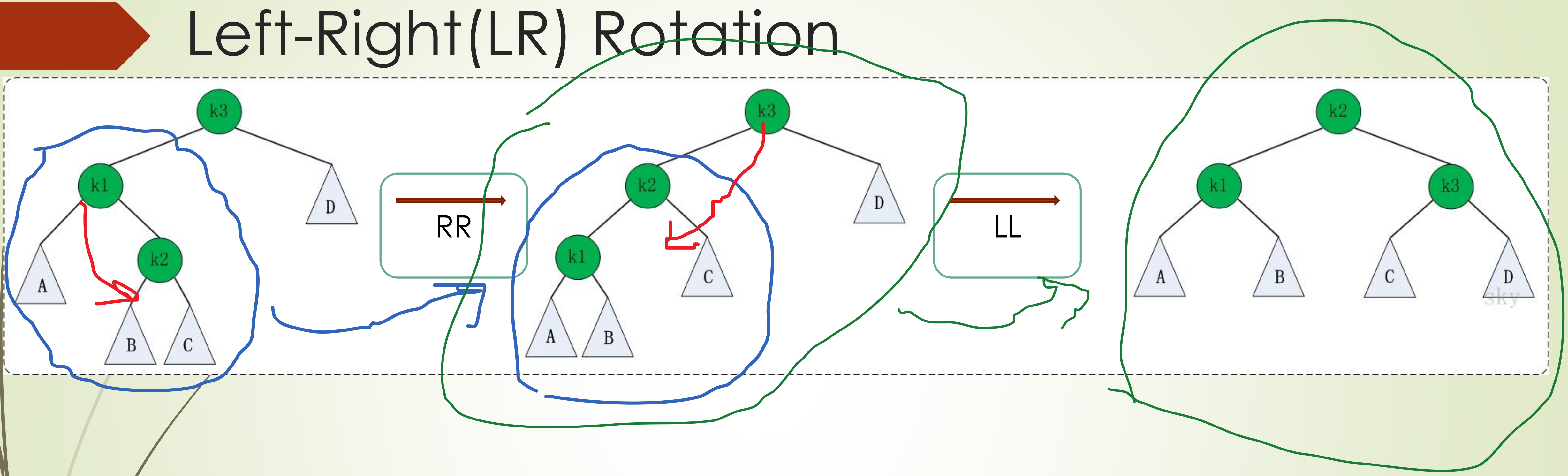


Condition: When root's right subtree is greater than its left subtree, and its right son's right subtree is greater than its right son' left subtree. (H(k2) > H(X), H(Z) > H(Y)))

How it works: We want to rotate k1 to be the left son of k2 and make k2 the root in order to make the tree more balanced.

We cannot directly make k1 the left son of k2 since k2 has already had its left subtree Y. According to the property of binary search tree, all the elements in subtree Y must be greater than k1, so we can make Y be the right subtree of k1. Also, all the elements in Y, k1, and X are less than k2, so we can put all these elements in the left subtree of k2, and remain Z unchanged.

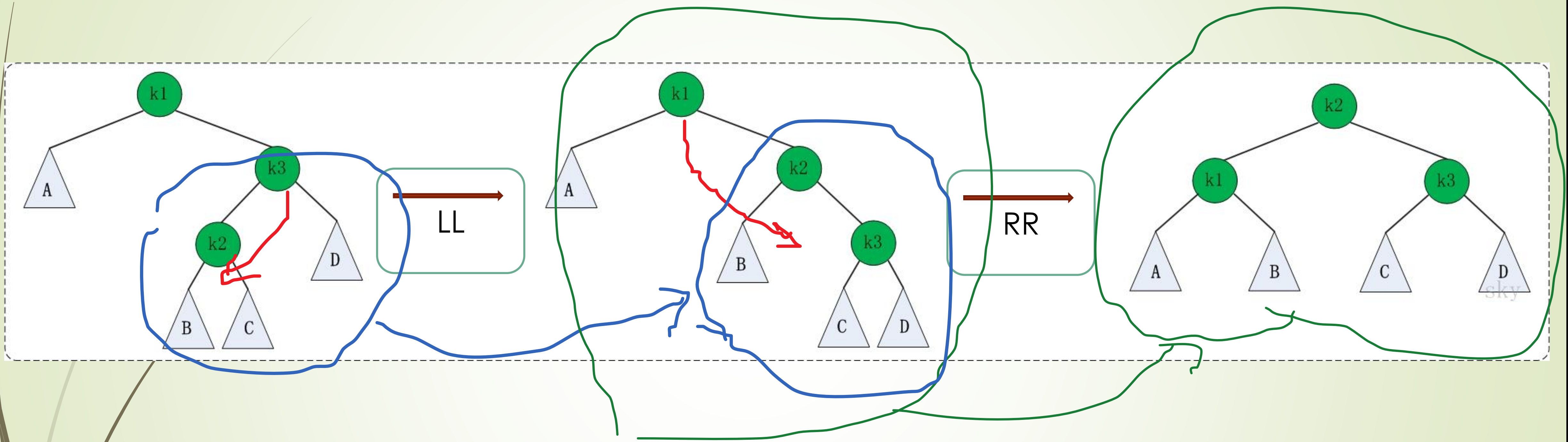
Left-Right(LR) Rotation



Condition: When root's left subtree is greater than its right subtree, and its left son's right subtree is greater than its left son' left subtree. (H(k2) > H(D), H(k2) > H(A)))

How it works: First make root's left subtree balance by using RR rotation.
Then make whole tree balance by using LL rotation.

Right-Left(RL) Rotation



Condition: When root's right subtree is greater than its left subtree, and its right son's left subtree is greater than its right son' right subtree. (H(k3) > H(A), H(k2) > H(D)))

How it works: First make root's right subtree balance by using LL rotation.
Then make whole tree balance by using RR rotation.

Build an AVL Tree

```
Struct NODE{
        Tree right;
        Tree Left;
        int element;
        int height; // -1 if it's an empty node, otherwise the height;
}
```

ALL the operations(Insert, remove, search) are the same as the binary search tree; Only need to add the balance process to each operation. Make sure to look at the C++ code.