Splay Tree

- A Self-adjusting Tree
  - A Binary Search Tree
  - Not so well balanced explicitly.
- Adjustments
  - Every time a new node is inserted, it is positioned at the root.
  - Every time a node is retrieved, it becomes the root.
- Performance
  - A sequence of $m$ insertions or retrievals with splaying a binary search tree of size $n$ will never need more than $m(1 + 3 \log n) + \log n$ upward moves of a target node.
Idea of Zig and Zag

- Right-Rotation = Zig
- Left Rotation = Zag

Zig-Zag and Zig Zig
Basic Algorithm

- Start from the root and keep on splitting the tree applying zig/zag and eventually bringing the target at the root.

- Divide the tree in three parts:
  - Tree#1: all the nodes confirmed greater than the key.
  - Tree#3: all the nodes confirmed larger than the key.
  - Three#2: if the key is there it must be inside this tree.

Some Operations
Some Operations (continued..)

Algorithm

Case 1: CC
Final Two Steps..

The Last Step
Sentinel Binary Tree

- No Need to Check for NULL Pointer. When we are at "Sentinel" node, we know we are at the leaf.

- In Splay Tree we start by copying the "target" at the sentinel node.

- The two pointers of the sentinel keeps track of the low and high sub-trees.

```java
treeNode = TreeSplay(treeNode = root, KeyType target) {
    TreeNode = current;  /* the current position in the tree */
    TreeNode = child;    /* one of the children of current */
    TreeNode = lastSmall; /* largest key known to be less than the target */
    TreeNode = firstLarge; /* smallest key known to be greater than the target */
    extern TreeNode = sentinel;
    sentinel->entry.key = target; /* Establish sentinel for searching. */
    lastSmall = firstLarge = sentinel;
    for (current = root; !EQ(target, child->entry.key); current = current->right) {
        if (LT(current->entry.key, target)) { /* child = current->right; */
            current = LinkLeft(current, lastSmall);
            lastSmall = RotateRight(current, lastSmall);
        } else { /* child = current->left; */
            current = LinkRight(current, firstLarge);
            firstLarge = RotateLeft(current, firstLarge);
        }
    }
    if (current == sentinel) {
        print("Target has been inserted as root of the tree.");
        root = current = MakeNode(target, sentinel);
    }
```