

# Splay Tree

1

## 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.

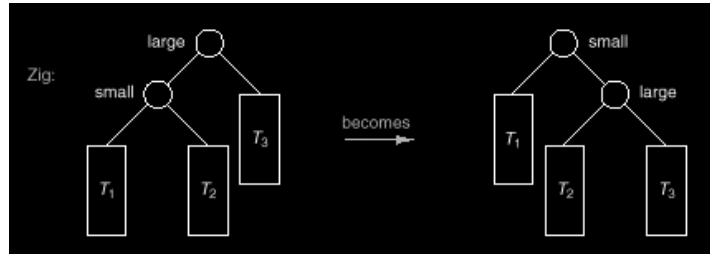


DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-2  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Idea of Zig and Zag

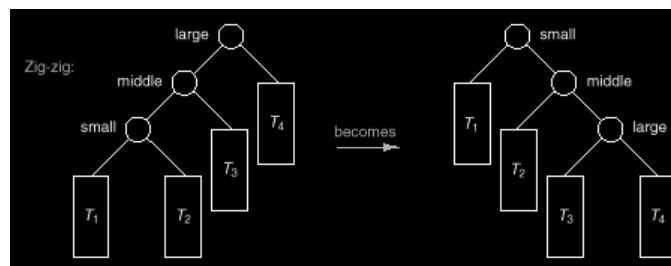
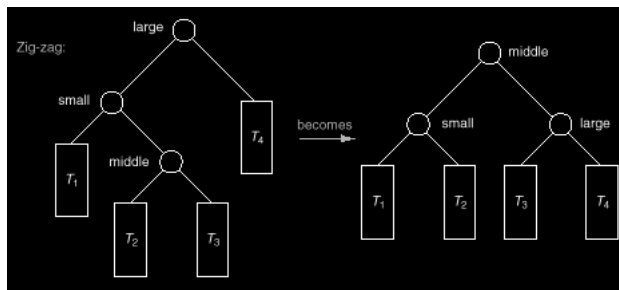
- Right-Rotation = Zig
- Left Rotation = Zag



DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-3  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Zig-Zag and Zig Zig



DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-4  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Basic Algorithm

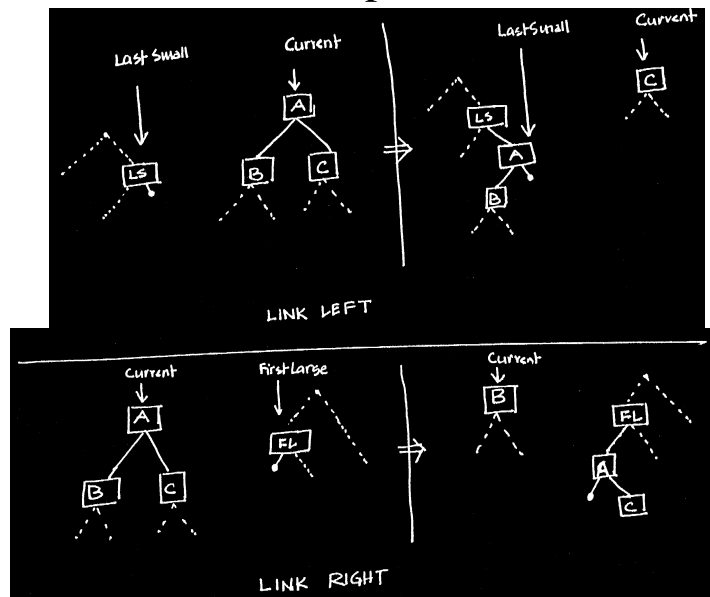
- Start from the root and keep on splitting the tree applying zig/zag and eventually bringing the target at the root.
- 
- Divide the tree in three parts:
  - Tree#1: all the nodes confirmed greater than the key.
  - Tree#3: all the nodes confirmed larger than the key.
  - Tree#2: if the key is there it must be inside this tree.



DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-5  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Some Operations



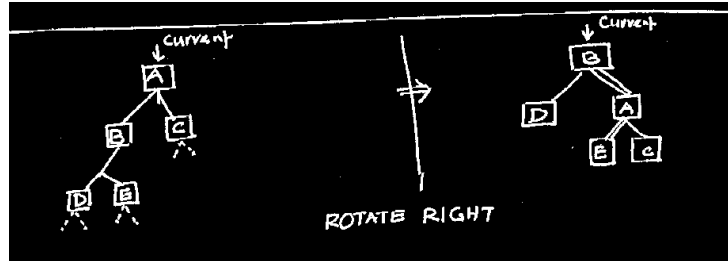
DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-6  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Some Operations (continued..)



DESIGN &  
ANALYSIS OF  
ALGORITHM

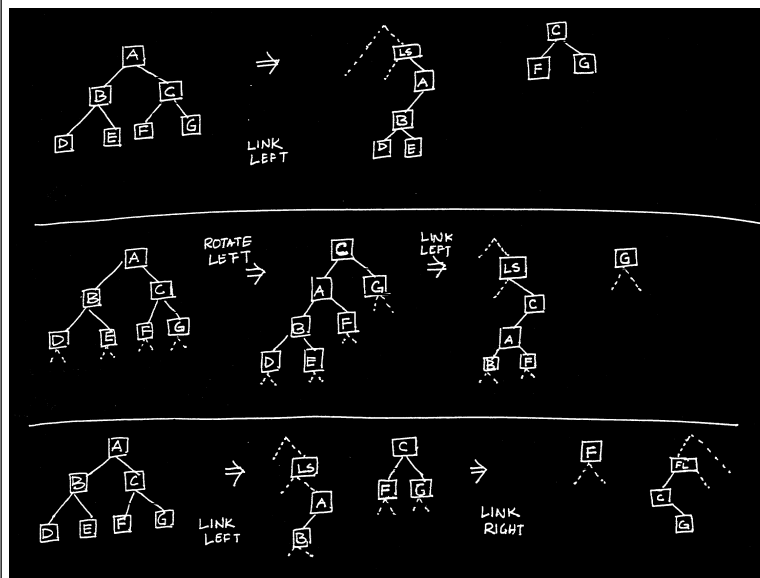


LECT-11, S-7  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## Algorithm



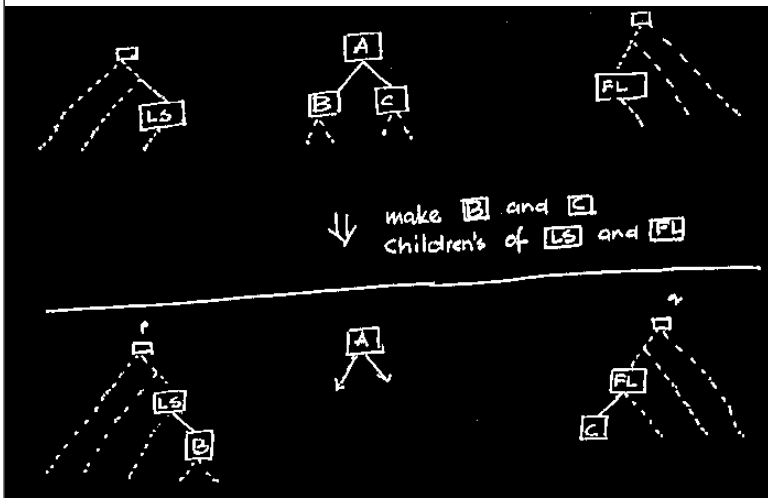
DESIGN &  
ANALYSIS OF  
ALGORITHM



Case  $I < C$

LECT-11, S-8  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

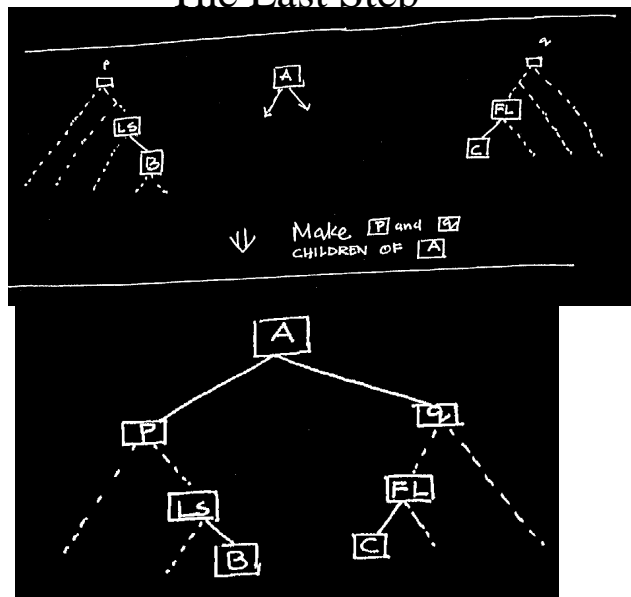
## Final Two Steps..



DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-9  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

## The Last Step

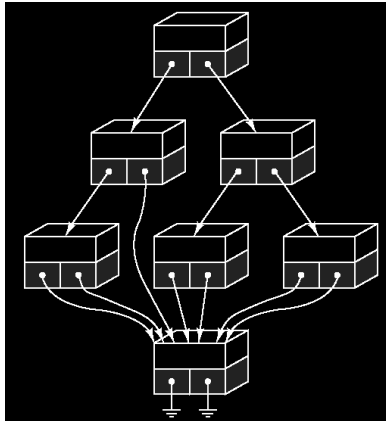


DESIGN &  
ANALYSIS OF  
ALGORITHM

LECT-11, S-10  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

# 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.



DESIGN & ANALYSIS OF ALGORITHM

LECT-11, S-11  
ALG00S, javed@kent.edu  
Javed I. Khan@1999

```

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;
}
    
```

```

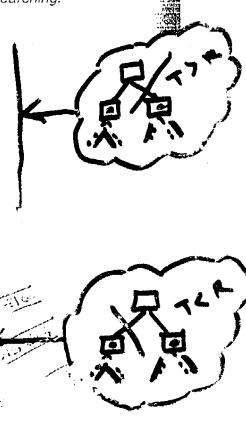
sentinel->entry.key = target; /* Establish sentinel for searching. */
lastsmall = firstlarge = sentinel;
    
```

```

for (current = root; NE(current->entry.key, target); )
    if (LT(current->entry.key, target)) {
        child = current->right;
        if (EQ(target, child->entry.key)) {
            current = LinkLeft(current, &lastsmall);
        } else if (GT(target, child->entry.key)) {
            current = RotateLeft(current);
            current = LinkLeft(current, &lastsmall);
        } else {
            current = LinkLeft(current, &lastsmall);
            current = LinkRight(current, &firstlarge);
        }
    } else {
        child = current->left;
        if (EQ(target, child->entry.key)) {
            current = LinkRight(current, &firstlarge);
        } else if (LT(target, child->entry.key)) {
            current = RotateRight(current);
            current = LinkRight(current, &firstlarge);
        } else {
            current = LinkRight(current, &firstlarge);
            current = LinkLeft(current, &lastsmall);
        }
    }
}
    
```

```

if (current == sentinel) {
    printf("Target has been inserted as root of the tree.");
    root = current = MakeNode(target, sentinel);
}
    
```



DESIGN & ANALYSIS OF ALGORITHM

LECT-11, S-12  
ALG00S, javed@kent.edu  
Javed I. Khan@1999