Splay tree

Property: recently accessed elements
are quick to access again

- is a binary search tree
- and all operations are combined with

splaying operation

- – rearranges the tree so that the node of the (looked up) element is placed at the root of the tree
- – use ~ rotations

Whenever an element is looked up in the tree,
move that element to the root of the tree
insert x: as with a normal binary search tree.
Splay the newly inserted node x to the top of the tree
delete a node ***x***: as with a binary search tree
splay the parent of the removed node to the top of the tree

Splay operations

*Assume **p** is the parent of **x***

Zig-zig step: when ***x*** and ***p*** are either both right children or they are both left children.

- The tree is rotated around ***g*** and then around ***p*** . (➔ ***p*** is the new root of the subtree)

e.g.: *when **x** and **p** are both left children*

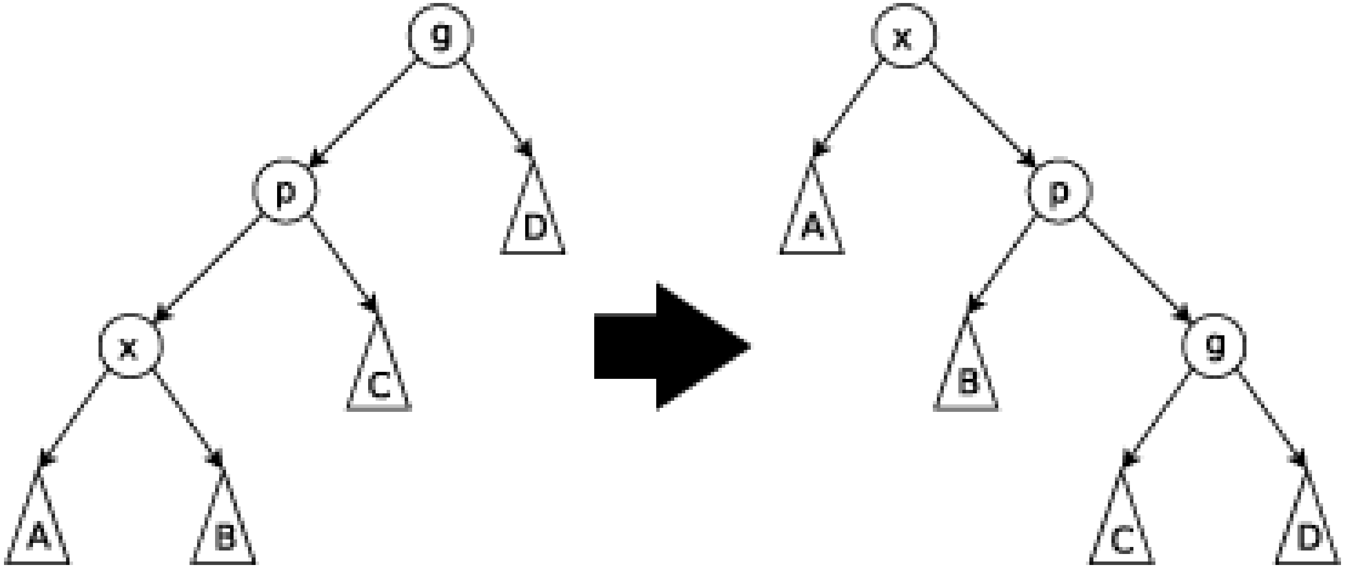


Image: *CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1182290*

Splay operations

Assume ***p*** *is the parent of* ***x***

Zig-zag step: when ***x*** is a right child and ***p*** is a left child
or ***x*** is left and ***p*** is right child

The tree is rotated around ***p***, and then rotated around ***g*.**

e.g.: *when* ***x*** *is a right child and* ***p*** *is a left child*

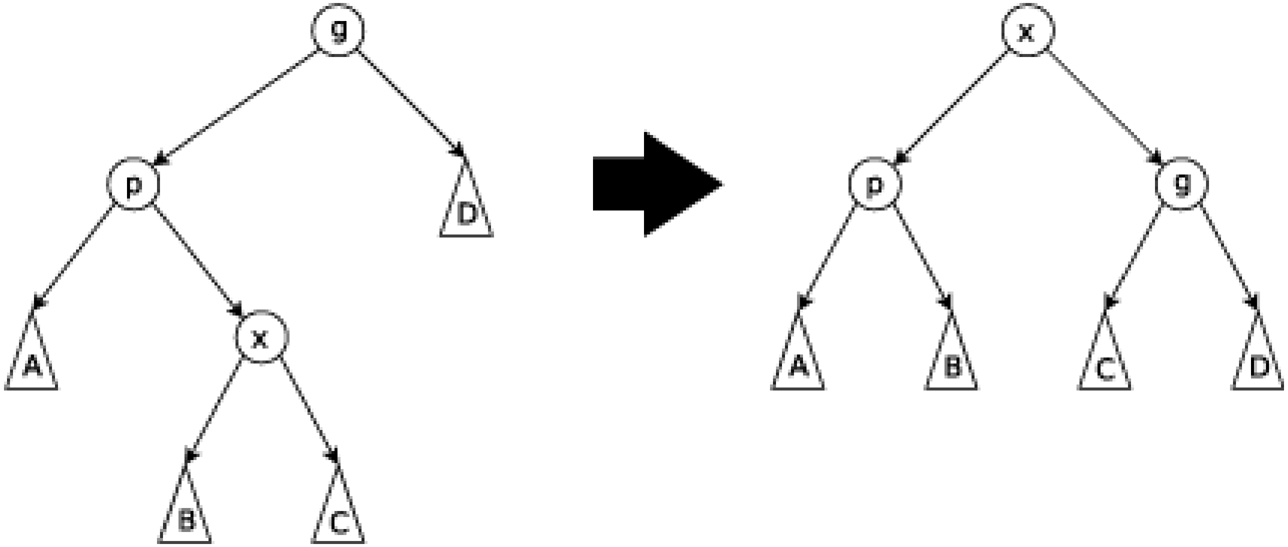


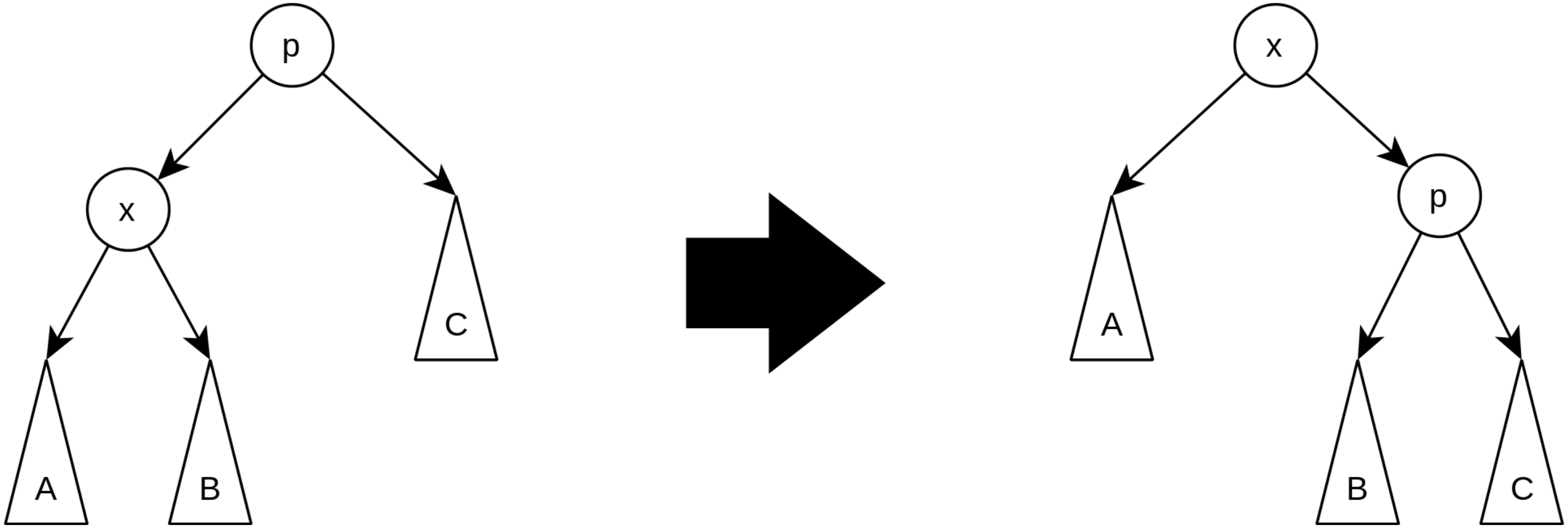
Image: *CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1182292*

Splay operations

*Assume **p** is the parent of **x***

Zig step: this step is done when ***p*** is the root.

e.g.: *when **x** is left child*



This operation is performed:

- only as the last step in a splay operation.
- only if when ***x*** has odd depth at the beginning of the operation.

Splay tree. Insert / delete

To **insert** a value ***x*** into a splay tree:

- insert x as with a normal binary search tree;
- then a splay operation of the inserted node is performed.

  As a result, the newly inserted node x becomes the root of the tree.

To **delete** a node ***x*** from a splay tree:

- use the same method as with a binary search tree;
- then splay the parent of the removed node to the top of the tree.

Remark: we can find other "close" variants in the literature

Operation splay

BSTNode:
    info: TComp
    left: ↑ BSTNode
    right: ↑ BSTNode
    parent: ↑ BSTNode
BinarySearchTree:
    root: ↑ BSTNode

```
Subalg. splay(x) {

  if x=NIL then return;

  while ([x].parent <> NIL)  execute
     if ([[x].parent].parent = NIL ) then
            if ([[x].parent].left = x) then  RightRotate([x].parent)
            else                             LeftRotate([x].parent)
            end_if
     else if ([[x].parent].left = x AND [[[x].parent].parent].left = [x].parent) then
                 RightRotate([[x].parent].parent);   RightRotate([x].parent)
     else if ([[x].parent].right = x AND [[[x].parent].parent].right = [x].parent) then
                 LeftRotate([[x].parent].parent);    LeftRotate([x].parent)
     else if ([[x].parent].left = x AND [[[x].parent].parent].right = [x].parent) then
                 RightRotate([x].parent);            LeftRotate([x].parent)
     else
                 LeftRotate([x].parent);             RightRotate([x].parent)
     end_if
       . . .
    end_if
  end_while

End_subalg.
```