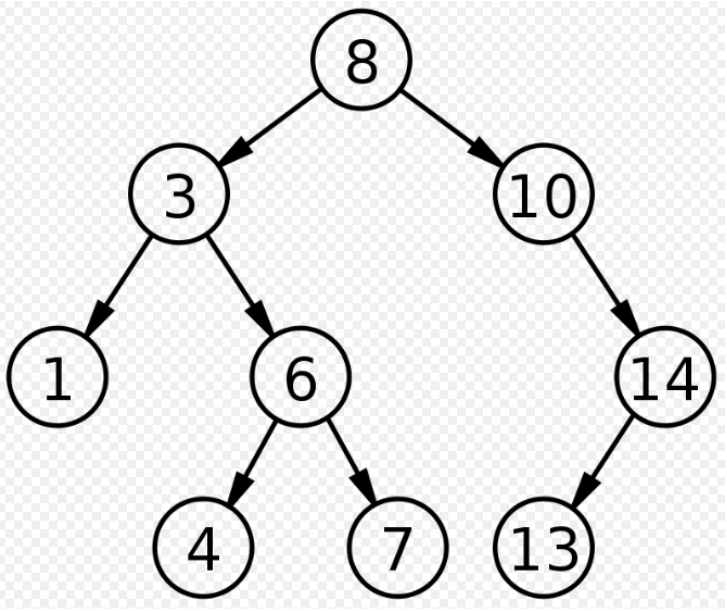
Binary Search Tree

Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

Operations

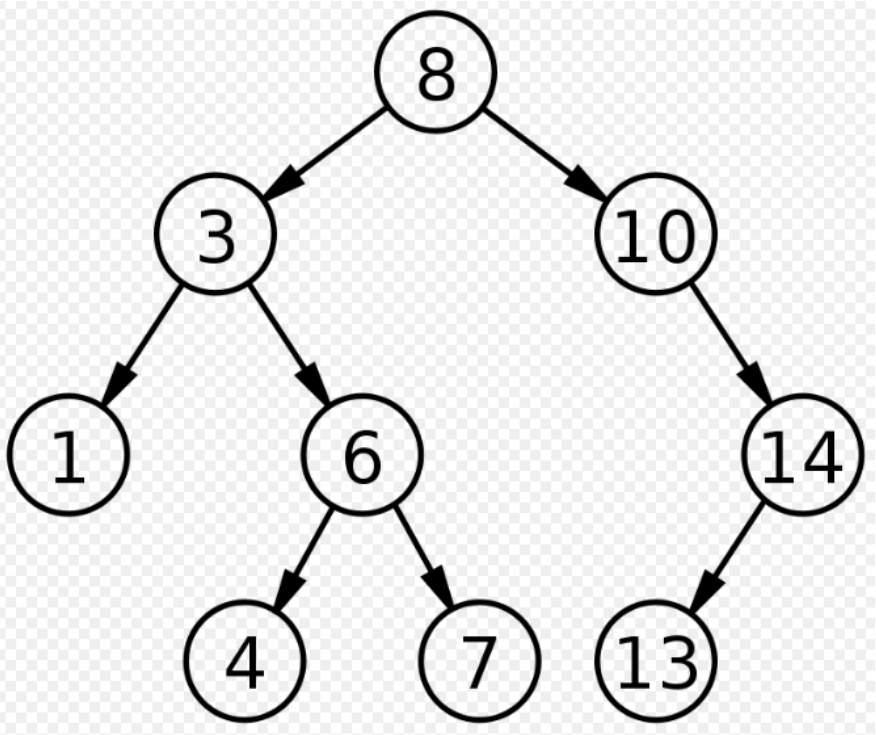
1. Insert
2. Search
3. Delete
4. Traversal (Pre-order, In-order, Post-Order)



Traversal

Preorder Traversal

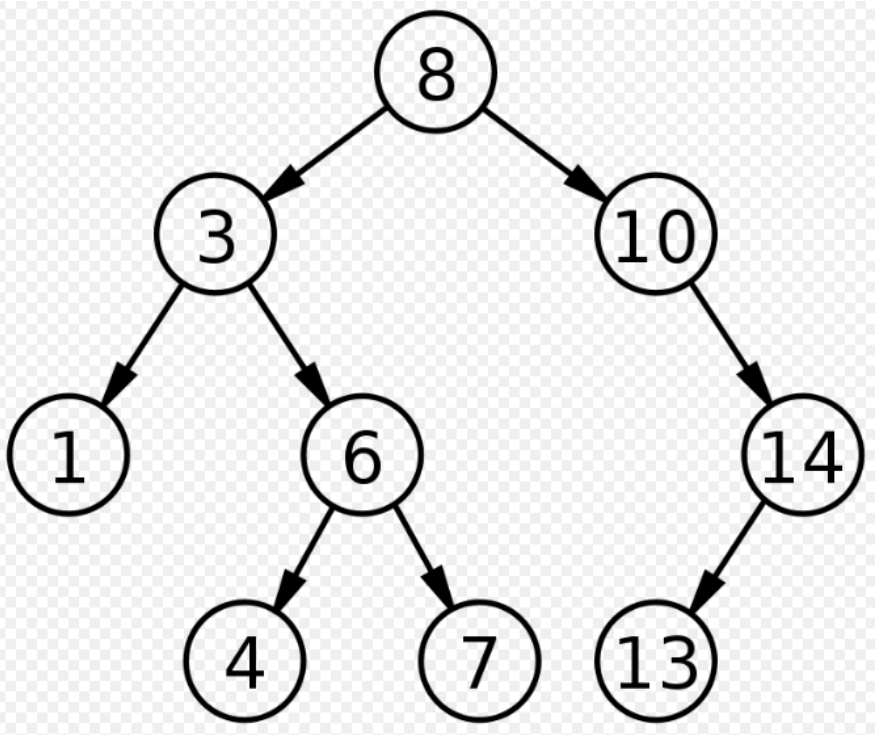
- Process data of root node
- Traverse left subtree
- Traverse right subtree



Traversal

Postorder Traversal

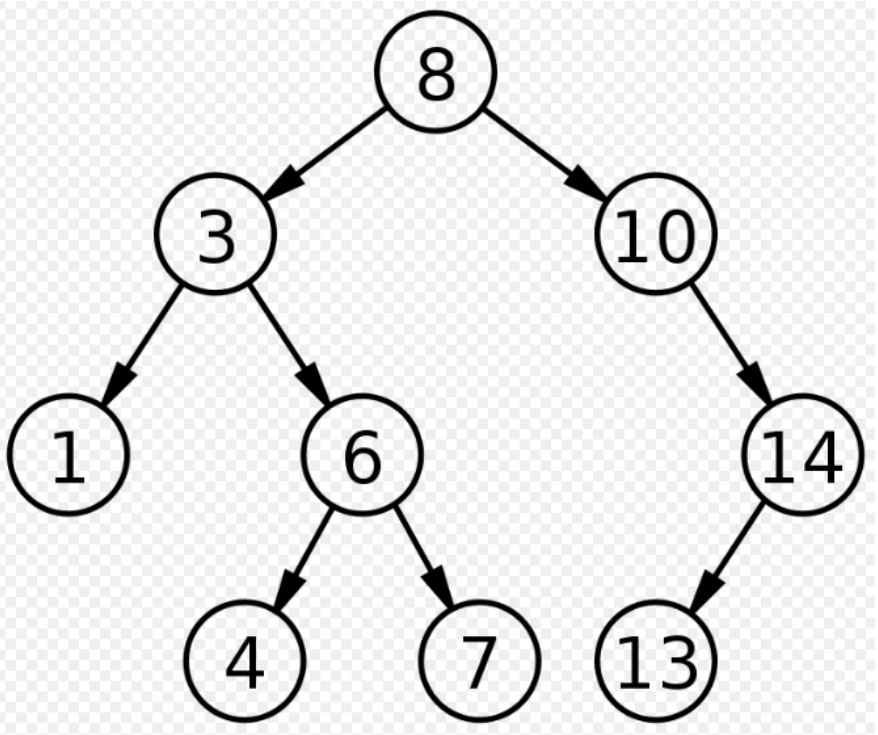
- Traverse left subtree
- Traverse right subtree
- Process data of root node



Traversal

Inorder Traversal

- Traverse left subtree
- Process data of root node
- Traverse right subtree



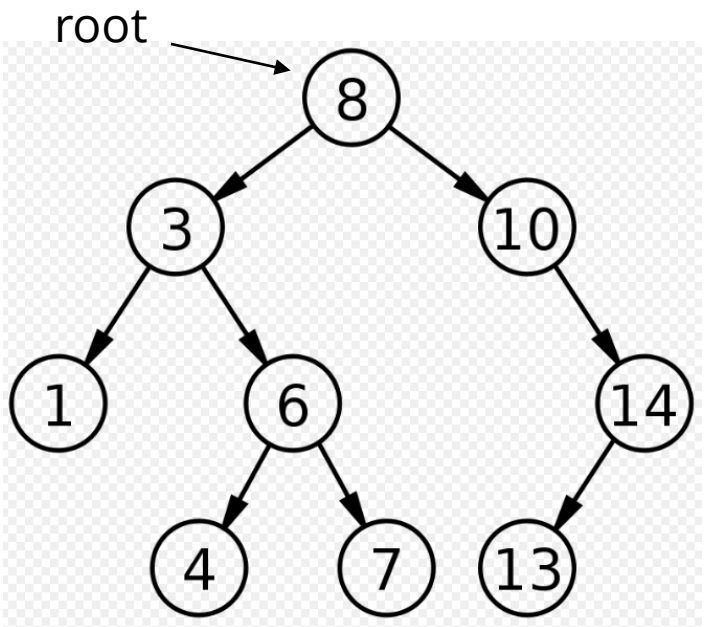
Insert Operation

Iterative Algorithm

```
TREE-INSERT(T, z)
 1  y = NIL
 2  x = T.root
 3  while x ≠ NIL
 4        y = x
 5        if z.key < x.key
 6              x = x.left
 7        else x = x.right
 8  z.p = y
 9  if y == NIL
10        T.root = z          // tree T was empty
11  elseif z.key < y.key
12        y.left = z
13  else y.right = z
```

Insert Operation

Recursive Algorithm




```
1.  If node == NULL
2.       return createNode(data)
3.  if (data < node->data)
4.       node->left  = insert(node->left, data);
5.  else if (data > node->data)
6.       node->right = insert(node->right, data);
7.  return node;
```

Search Operation

```
TREE-SEARCH(x, k)
1   if x == NIL or k == x.key
2         return x
3   if k < x.key
4         return TREE-SEARCH(x.left, k)
5   else return TREE-SEARCH(x.right, k)
```

Search Operation

```
ITERATIVE-TREE-SEARCH(x, k)
1   while x ≠ NIL and k ≠ x.key
2       if k < x.key
3           x = x.left
4       else x = x.right
5   return x
```

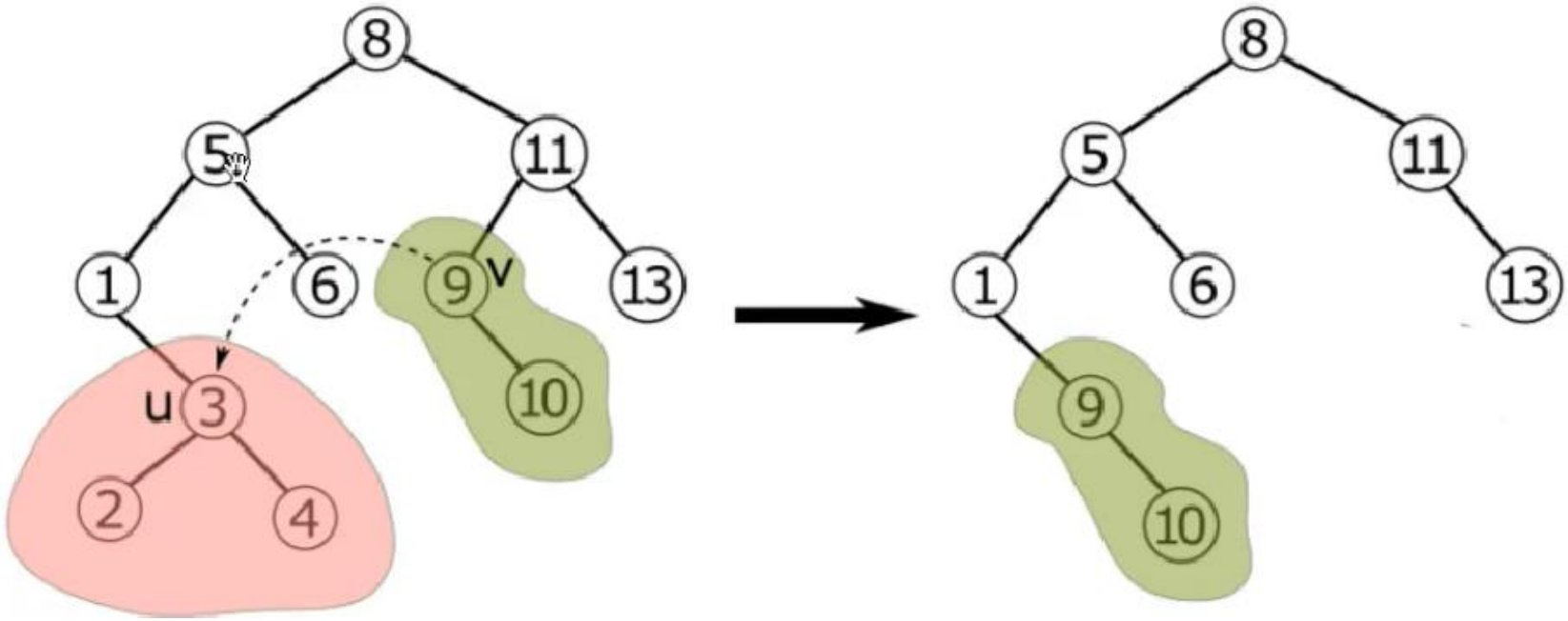
findMin Operation

```
TREE-MINIMUM(x)
1  while x.left ≠ NIL
2      x = x.left
3  return x
```

findMax Operation

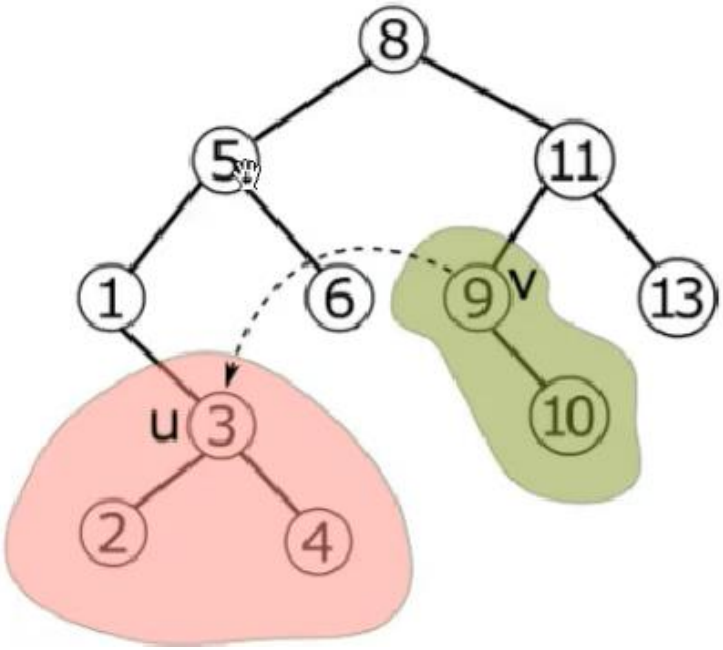
```
TREE-MAXIMUM(x)
1  while x.right ≠ NIL
2      x = x.right
3  return x
```

Transplant Operation

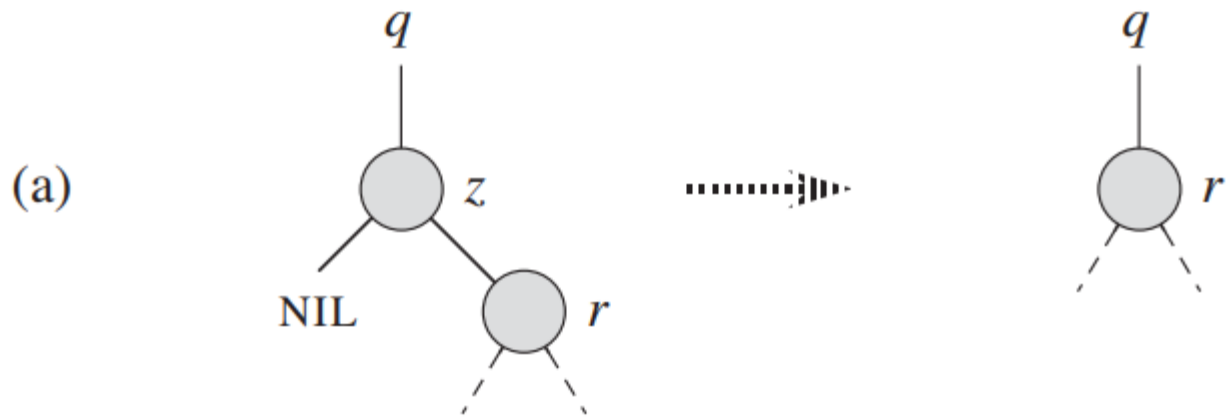


Transplant Operation

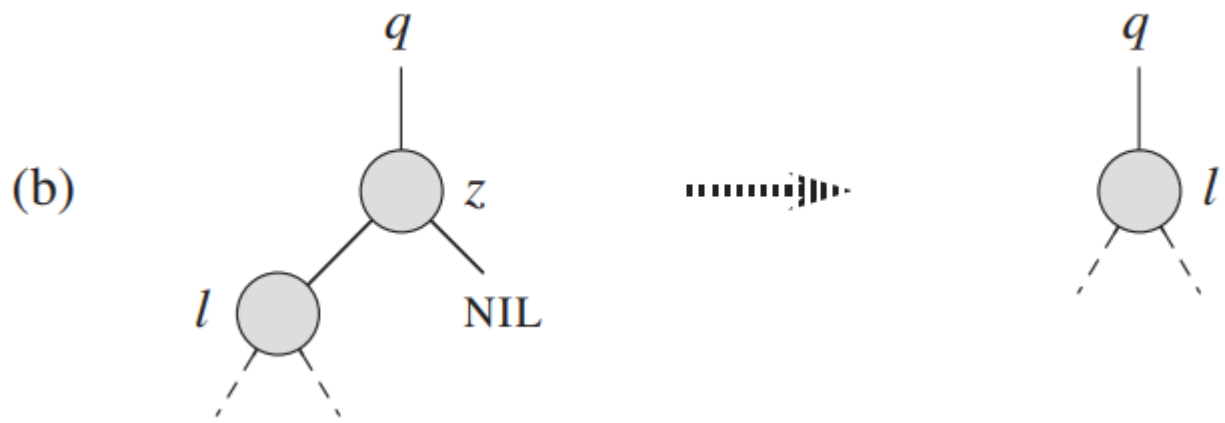
```
TRANSPLANT(T, u, v)
1   if u.p == NIL
2       T.root = v
3   elseif u == u.p.left
4       u.p.left = v
5   else u.p.right = v
6   if v ≠ NIL
7       v.p = u.p
```



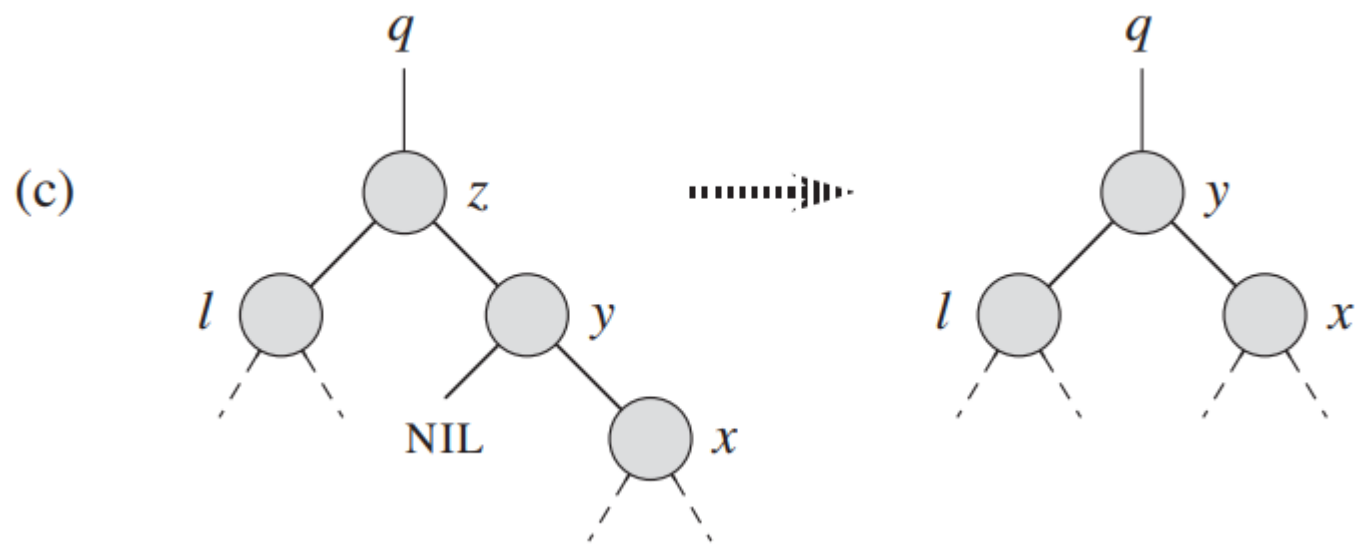
delete Operation



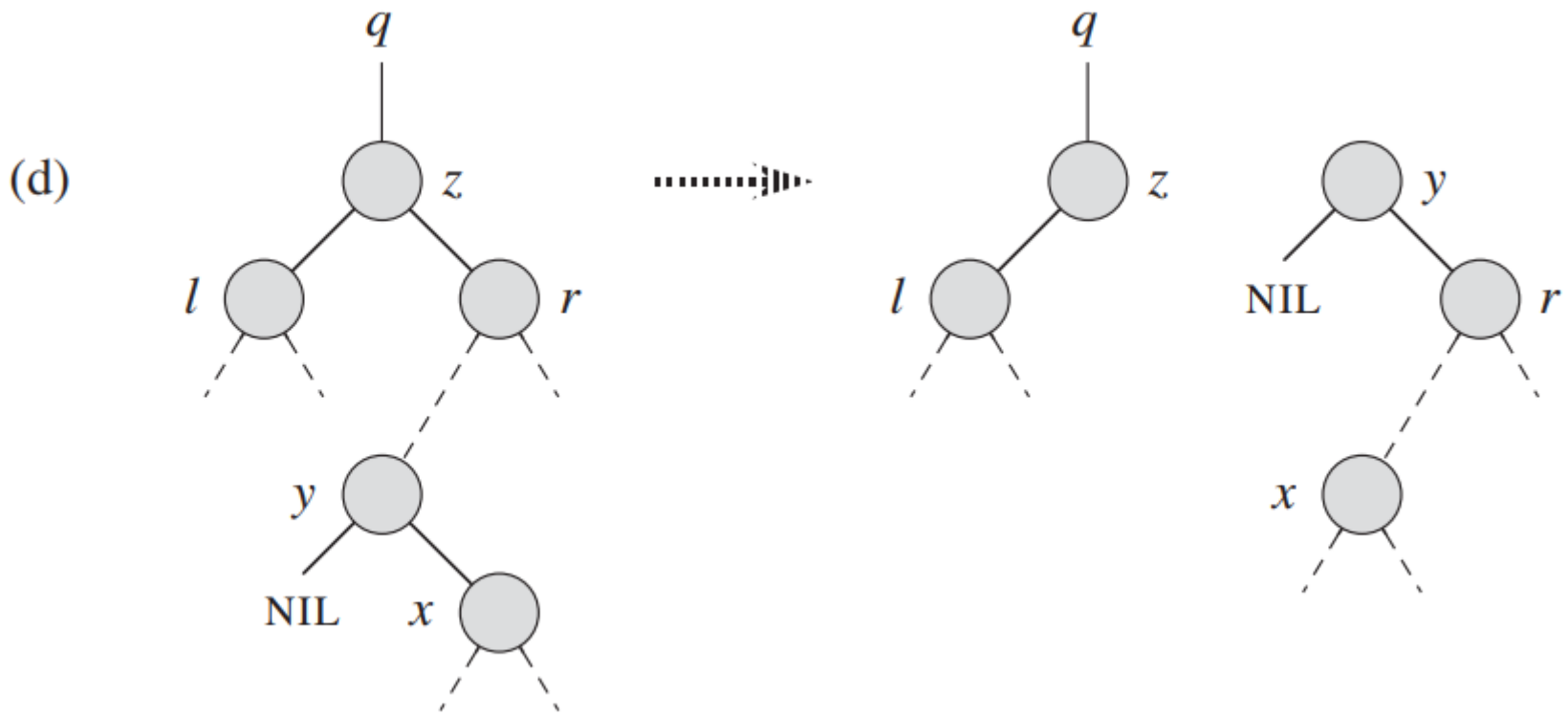
delete Operation



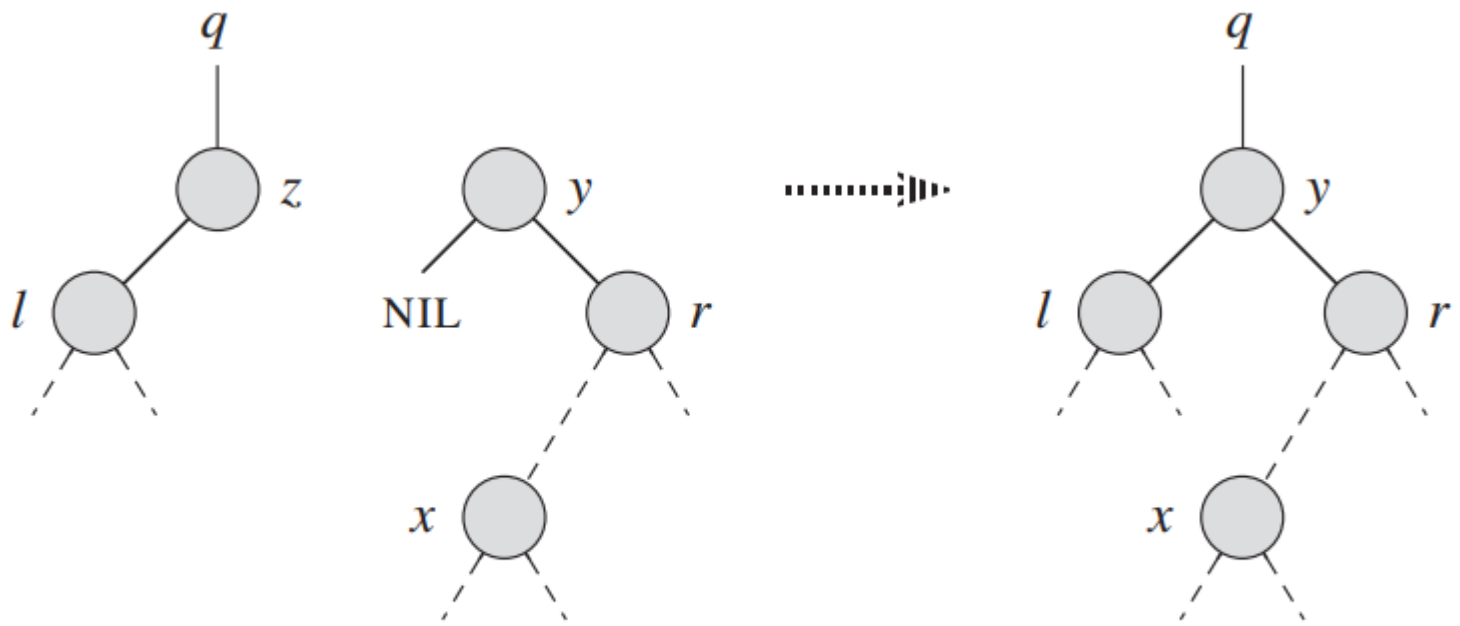
delete Operation



delete Operation



delete Operation



delete Operation

```
TREE-DELETE(T, z)
 1  if z.left == NIL
 2      TRANSPLANT(T, z, z.right)
 3  elseif z.right == NIL
 4      TRANSPLANT(T, z, z.left)
 5  else y = TREE-MINIMUM(z.right)
 6      if y.p ≠ z
 7          TRANSPLANT(T, y, y.right)
 8          y.right = z.right
 9          y.right.p = y
10      TRANSPLANT(T, z, y)
11      y.left = z.left
12      y.left.p = y
```