Binary Search Tree

SCSJ2013 Data Structures & Algorithms

Nor Bahiah Hj Ahmad & Dayang Norhayati A. Jawawi

Faculty of Computing

Objectives

At the end of the class students are expected to:

- Identify characteristics of binary search tree
- Identify basic operations of a tree such as tree traversals, insert node, delete node, searching

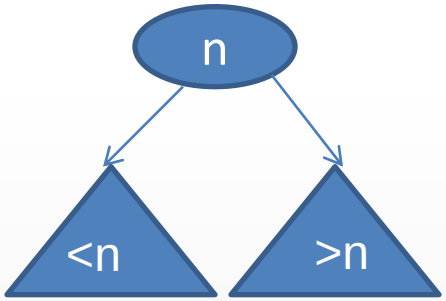
Binary Search Tree

What is binary search tree?

- **The tree has the properties, whereby every node, called n in the tree has value:**

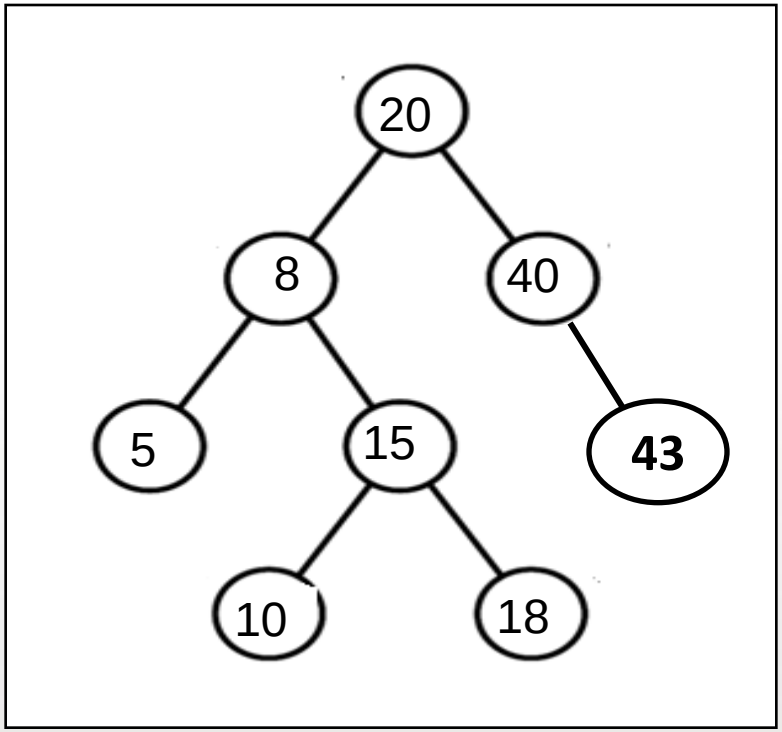
1. Value of n is greater than all values in n's left subtree.
2. Value of n is less than all values in n's right subtree.
3. Both left subtree and right subtree are also binary search trees

Binary Search Trees

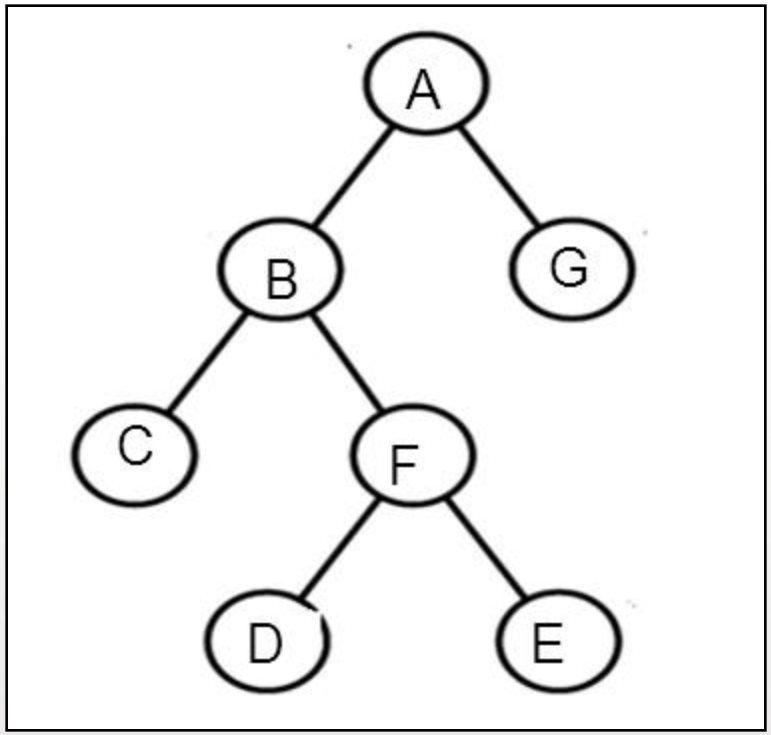


All key in the left sub-tree is less than n

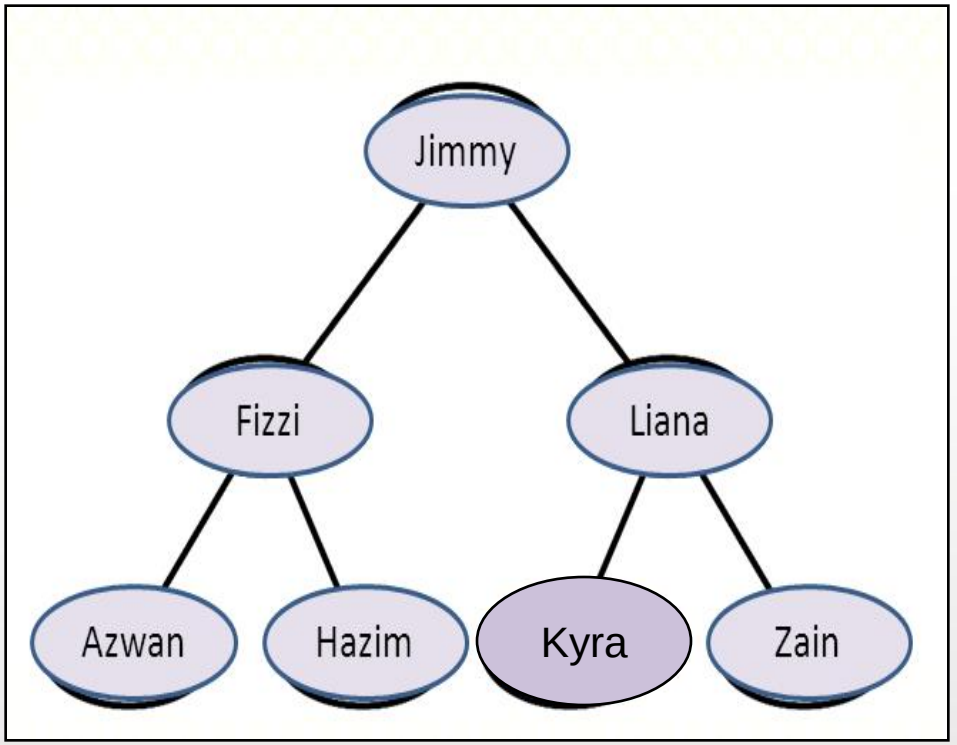
All key in the right sub-tree are greater than n



Binary Search Tree



Not a Binary Search Tree



Binary Search Tree

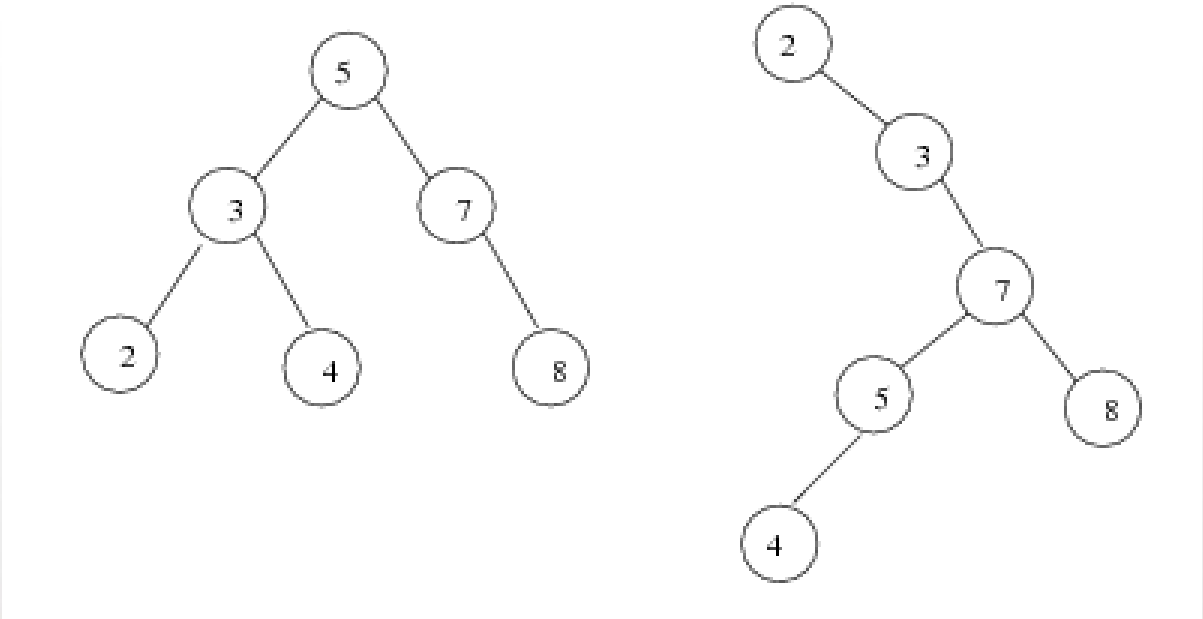
Disadvantages of Binary Search Tree

The shape of the tree depends on the order of insertions, and it can be degenerated.

When inserting or searching for an element, the key of each visited node has to be compared with the key of the element to be inserted/found.

Keys in the tree may be long and the run time may increase.

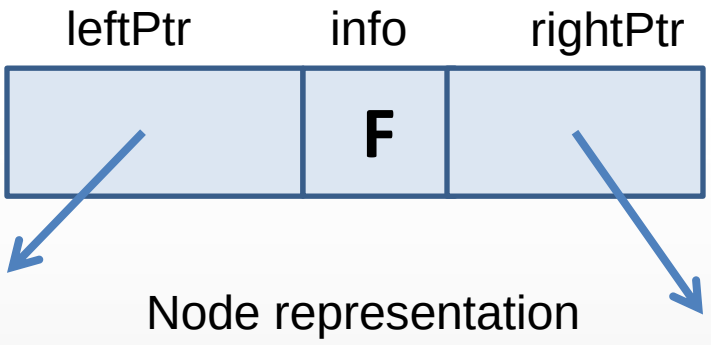
Binary Search Tree



Two binary search trees representing the same set:

- Average depth of a node is O(log n);
- Maximum depth of a node is O(n)

Pointer-based ADT Binary Tree

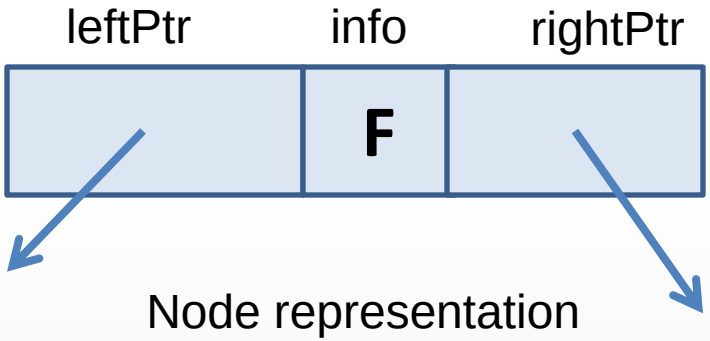


Node representation

A pointer-based implementation of a binary tree

- Elements in a binary tree is represented by using nodes.
- Nodes store the information in a tree.
- Each node in the tree must contain at least 3 fields containing:
  - item
  - Pointer to left subtree
  - Pointer to right subtree
- Need 2 declarations in a tree implementation:
  1. Node declaration
  2. Tree declaration

Node Implementation




```
typedef char ItemType;
struct TreeNode
{
    itemType info;
    TreeNode *left;
    TreeNode *right;
}
```

Where:

info, the node store char value.

left, pointer to left subtree

right, pointer to right subtree

Tree Implementation

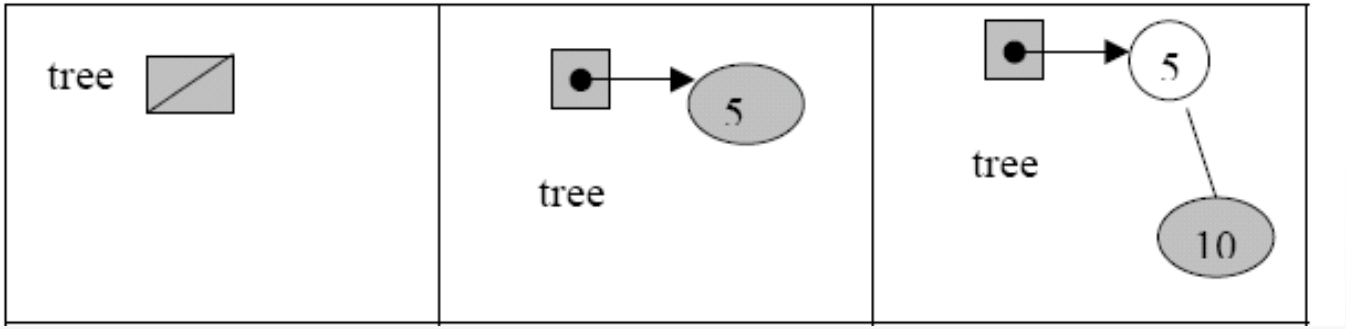
```
class TreeType {
public:
  TreeType(){ head = NULL; };
  bool IsEmpty() { return head == NULL; }
  void RetrieveItem(ItemType&,bool& found);
  void InsertItem(ItemType);
  void DeleteItem(ItemType);
  void PrintTree() const;
private:
  TreeNode * root;
};
```

The tree can be accessed using root, which is a pointer to root of the tree.

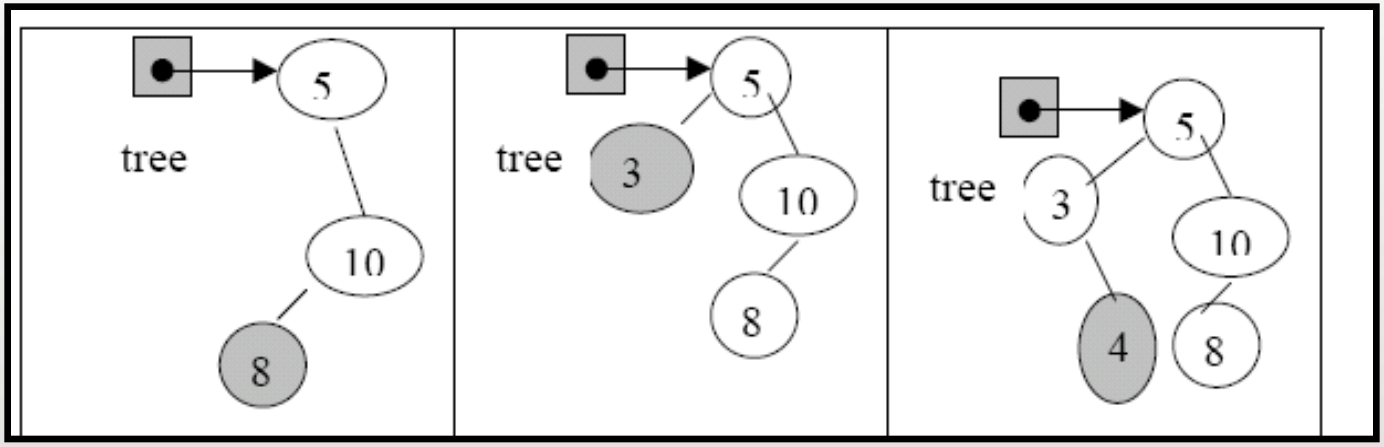
Insert Node Into a Binary Search Tree

- The insert operation will insert a node to a tree and the new node will become leaf node.
- Before the node can be inserted into a BST, the position of the new node must be determined. This is to ensure that after the insertion, the BST characteristics is still maintained.

Insert 5, 10, 8, 3 , 4 and 15 in a BST

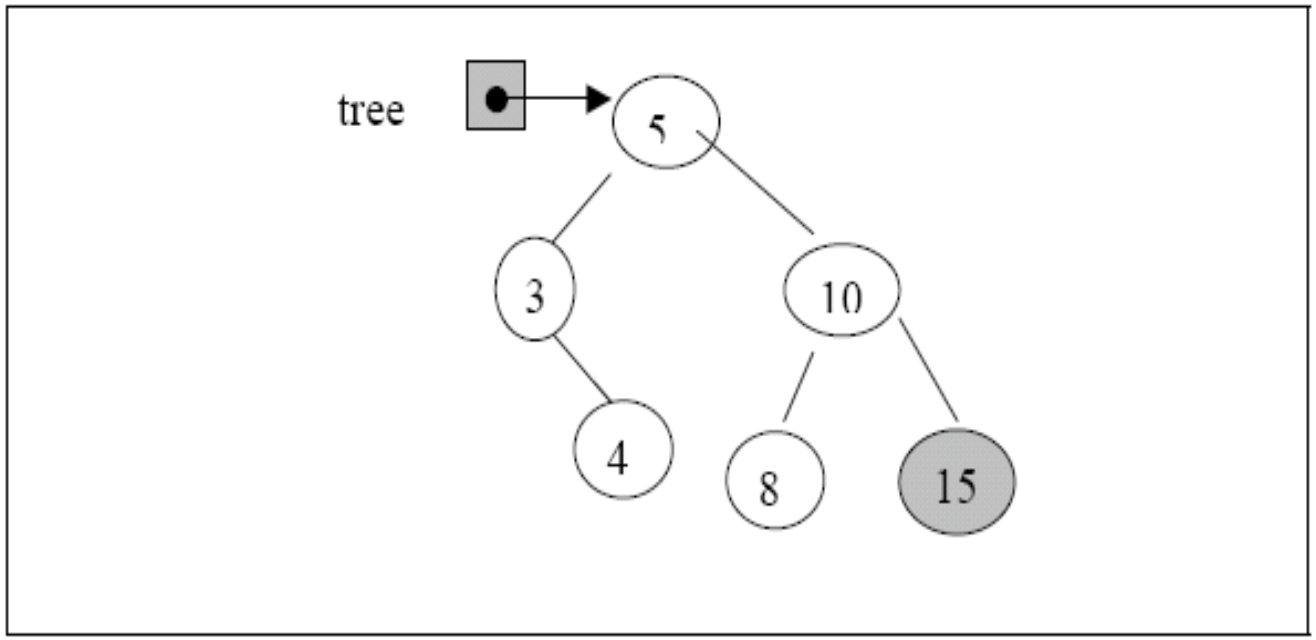


Empty tree Insert 5 Insert 10



Insert 8 Insert 3 Insert 4

Insert 5, 10, 8, 3 , 4, 15 to a tree



Finally, Insert the last node; 15.

Time complexity = O(height of the tree)

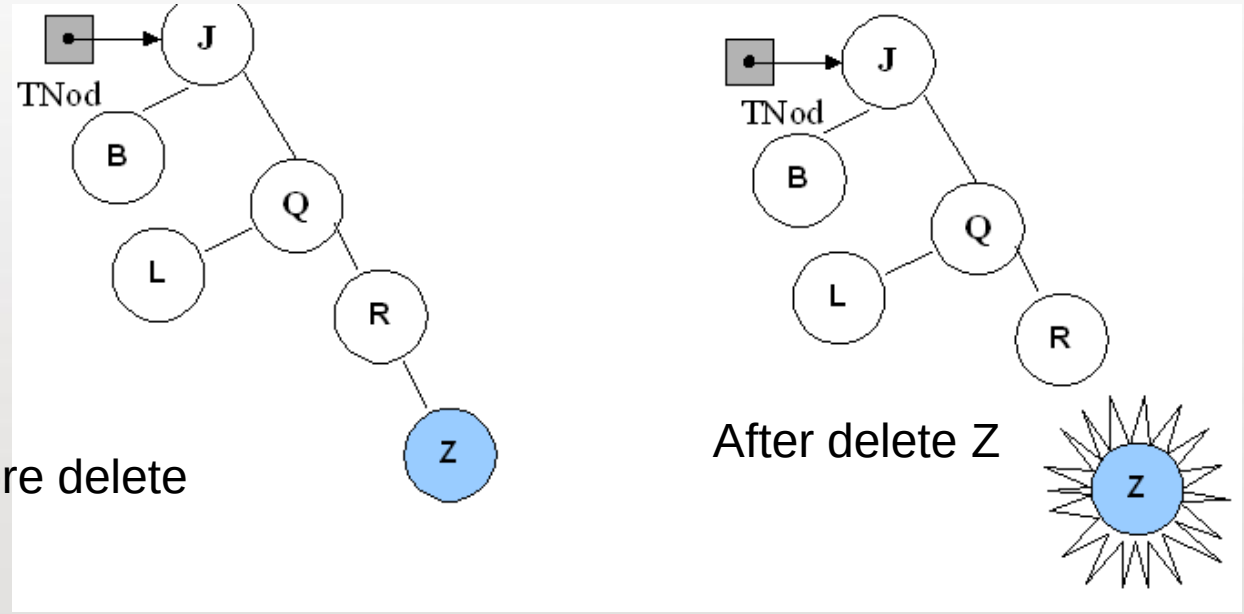
Delete a Node from a Tree

- When a node is deleted, the children of the deleted node must be taken care of to ensure that the property of the search tree is maintained.
- There are 3 possible cases to delete a node in a tree:
  1. Delete a leaf
  2. Delete a node with one child
  3. Delete a node that has two children

Delete a Leaf Node

The node to be deleted is a leaf

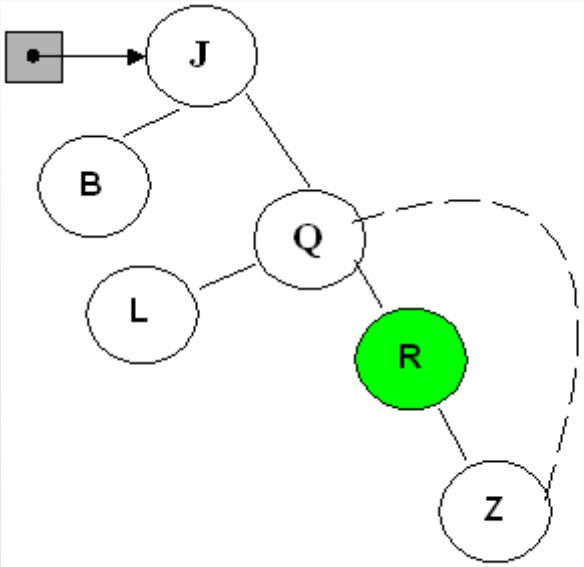
- **Set the pointer in N's parent to NULL and delete it immediately**
- **Example : Delete leaf Node: Z**



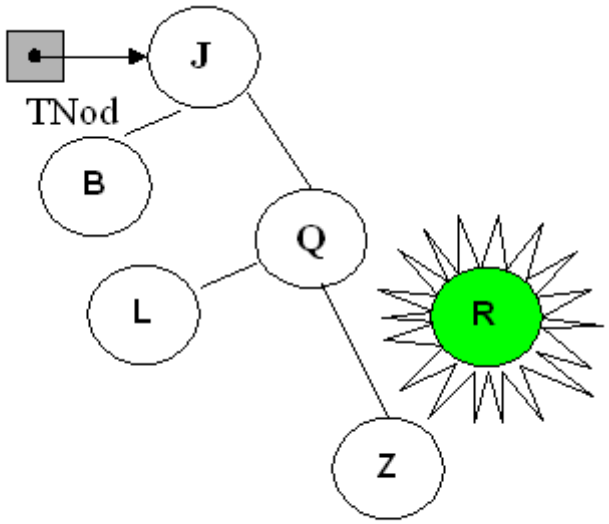
Delete a Node With One Child

Delete node R.

Adjust a pointer from the parent to bypass that node



Before delete R



After delete R

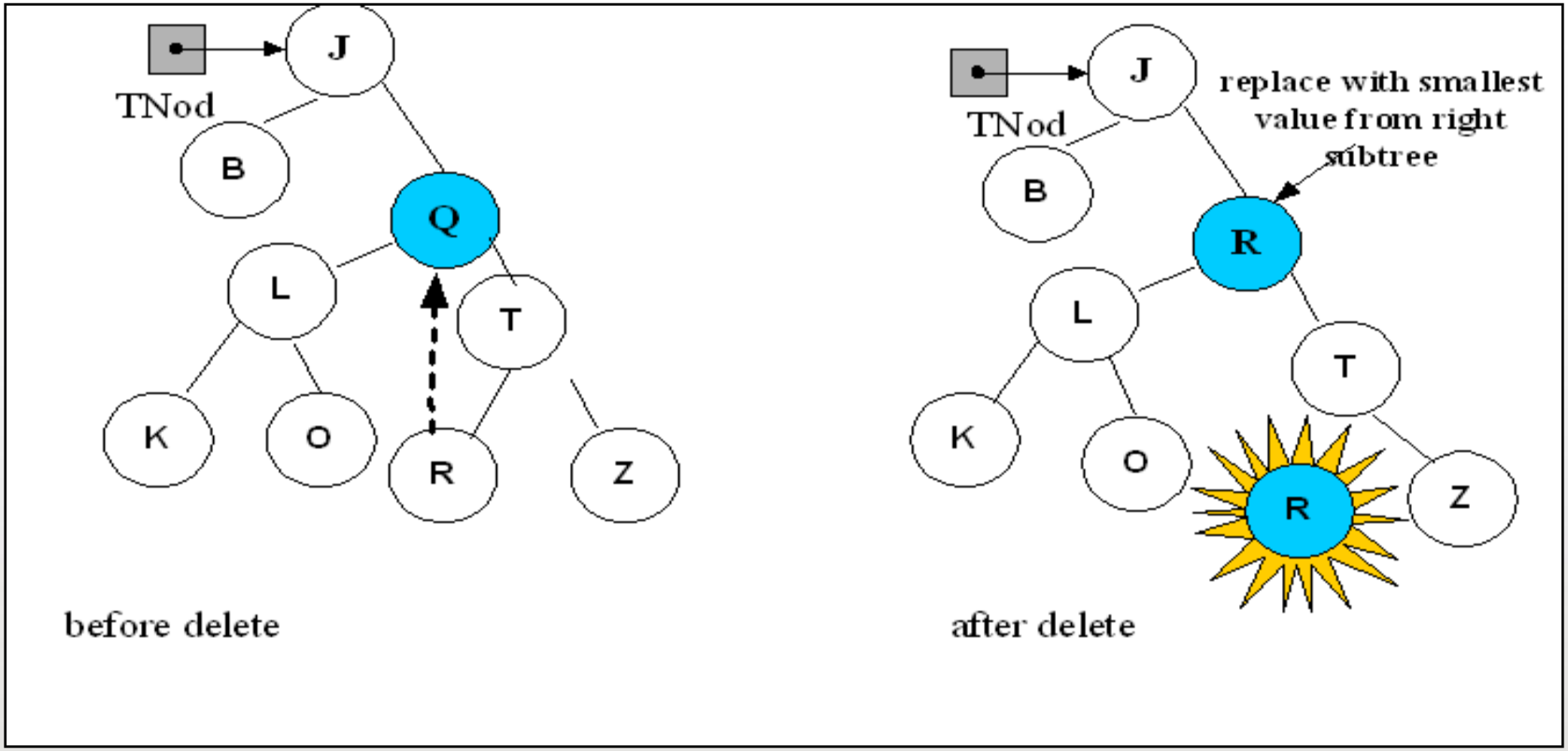
Delete a Node With Two Children

To delete a node N that has two children.

- Locate another node *M* that is easier to delete
  - M is the leftmost node in N's right subtree
  - M will have no more than one child
  - M's search key is called the inorder successor of N's search key
- Copy the item that is in M to N
- Remove the node M from the tree

Delete a Node with 2 Children

Delete Q that has 2 children



Summary

Binary search trees come in many shapes. The shape of the tree determines the efficiency of its operations. It is important to have balanced tree in order to ensure the efficiency of tree operations.

The height of a binary search tree with n nodes can range from a minimum of $O\log_2(n + 1)$ to a maximum of n.


Thank
You