DATABASE SYSTEMS LAB
SEOUL NATIONAL UNIVERSITY

Data Structures (Spring 2020)

Tree (4th Lab)

2020.04.10

Seoul National University
Database Systems Lab

Today's Lab

- Announcement
- Binary Search Tree
  - Implementation
  - Insert
  - Traversal

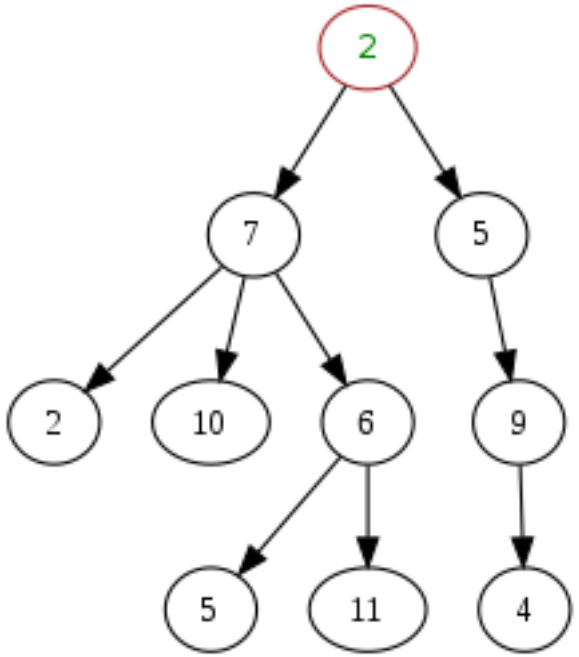
Announcement

- Online Exam
  - **Thanks for cooperation!**
  - We will do answering your submission as soon as possible

- Questions
  - Identification
  - Submission Time
  - Noise (such as Music)

Tree Implementation

- a **tree** is that simulates a hierarchical tree structure, with a root value and subtrees of children with a parent node, represented as a set of linked nodes.



From: https://en.wikipedia.org/wiki/Tree_(data_structure)

Binary Search Tree

Definition 23

A BST is a binary tree that is either empty or that satisfies the following conditions:

1. key of any node in the left subtree $<$ key of the root node,
2. key of any node in the right subtree $\geq$ key of the root node,
3. both the subtrees are BST.

Property of Binary Search Tree

Algorithm 2

```
Insert(T,x)
   if (T == null) return (T = new Node(x));
   else if (x < T.key) T.left = Insert(T.Left,x);
   else T.right = Insert(T.Right,x);
```

Insert node into Binary Search Tree

=> Find appropriate leaf node and insert node

From: Bongki Moon, "Lecture Notes on Data Structures: Trees"

Binary Search Tree

- Implement Binary Search Tree (only with insertion)

```java
interface TNode<E> {

    public TNode<E> getLeft();          // get left node
    public TNode<E> getRight();         // get right node
    public E getValue();                // get value

    public void setRight(TNode<E> value);   // set right node
    public void setLeft(TNode<E> value);    // set left node
    public void setValue(E value);          // set value

}
```

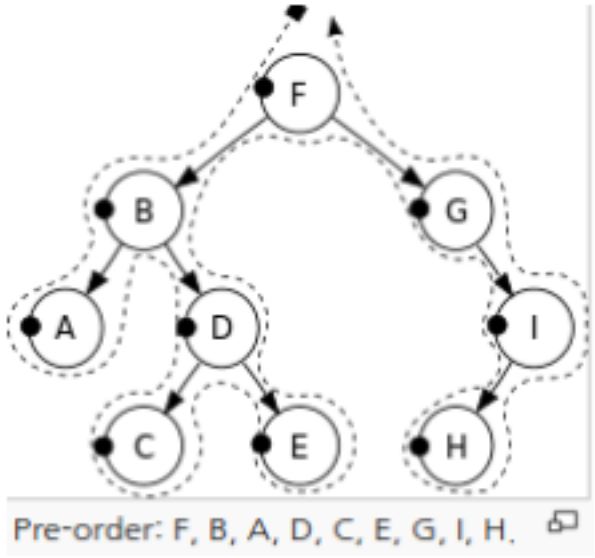
TNode.java

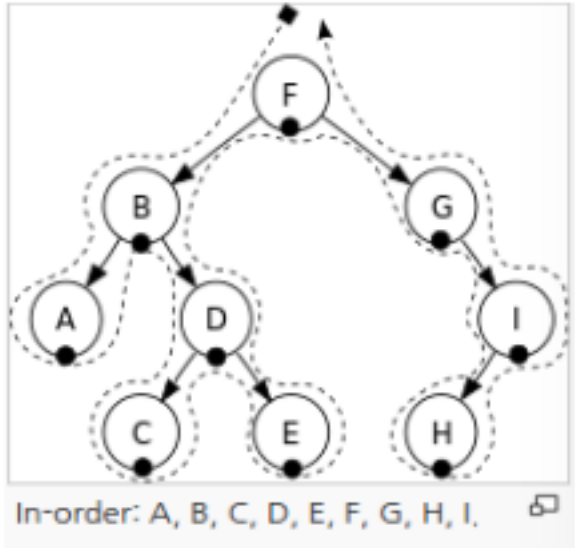
```java
// insert value into input tree named "root"
public static void insertStringToBST(TNode<String> root, String value) {
    // TODO: find leaf node and insert value (element)
```

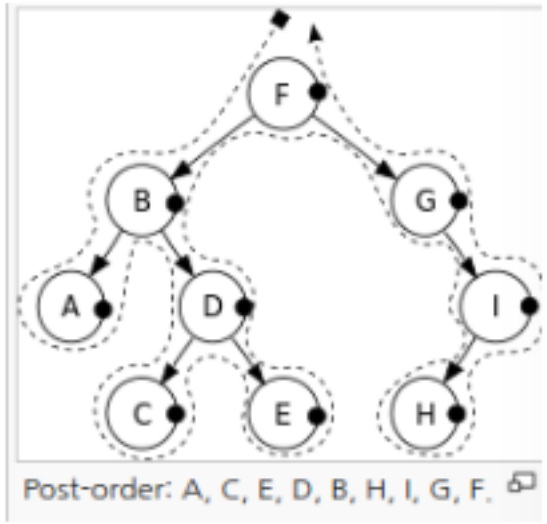
TNodeImpl.java

Tree Traversal

- Three types of traversal
  - Pre-order, In-order, Post-order

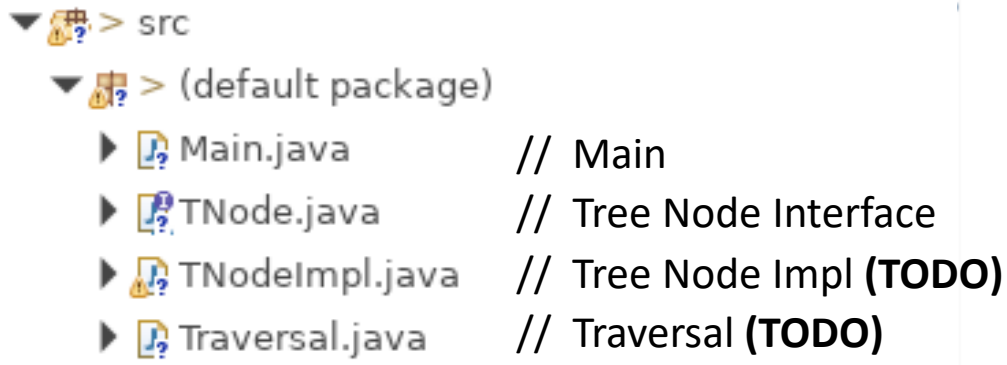






Exercises

- Fill the blank of codes
  - Write your code into "**// TODO:**" section (**TNodeImpl.java, Traversal.java**)
  - TNodeImpl (Tree Node) with insertion
  - Traversal (in-order, pre-order, post-order)



Project Structure

```java
public class Main {
    // main point.
    public static void main(String[] args) {
        TNode<String> tree = (TNodeImpl<String>) createStringTree(args);

        System.out.print("pre-order: ");
        Traversal.preorder(tree);
        System.out.println();

        System.out.print("in-order: ");
        Traversal.inorder(tree);
        System.out.println();

        System.out.print("post-order: ");
        Traversal.postorder(tree);
        System.out.println();

    }
```

Main.java

Exercises

- Result

```
$ java Main F B A D C E G I H
pre-order: F B A D C E G I H
in-order: A B C D E F G H I
post-order: A C E D B H I G F
```