Data Structures (Spring 2020)

Binary Search Tree (5th Lab)

2020.04.17

Seoul National University

Database Systems Lab

Today’s Lab

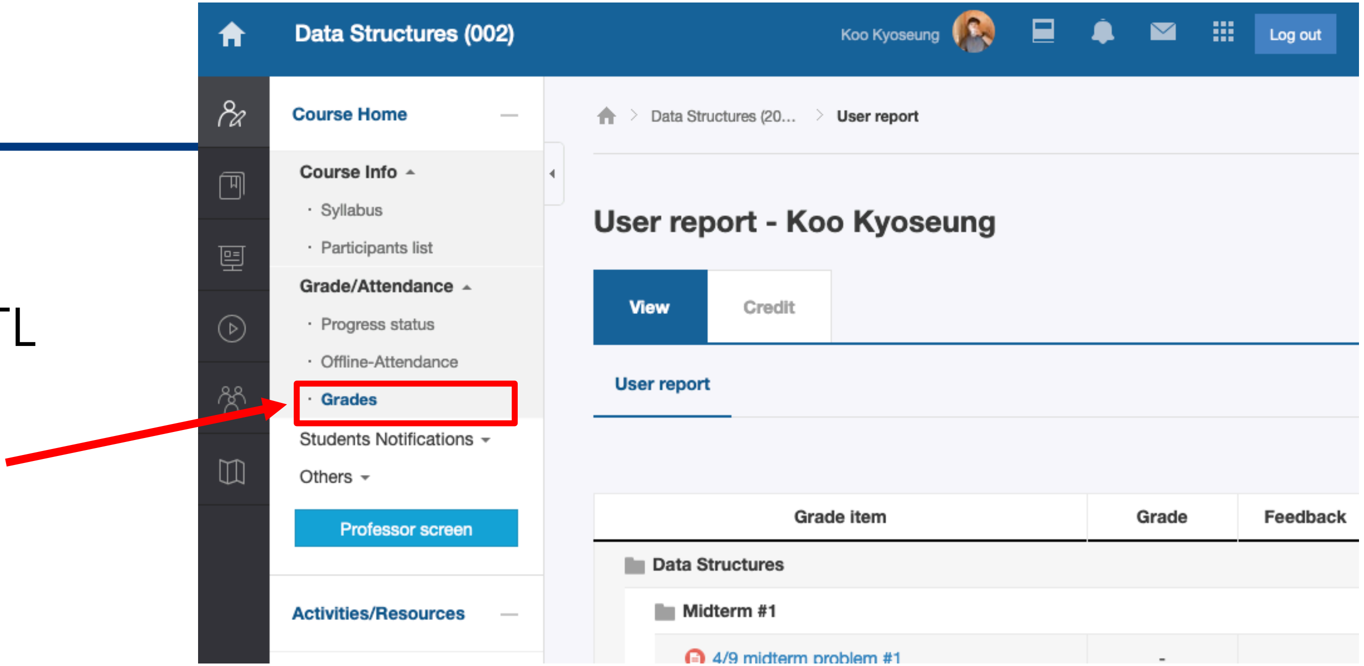
- Announcement
  - Midterm #1
  - Programming Assignment

- Binary Search Tree
  - Search()
  - Remove()

- Midterm #1 Claim

Announcement

- Midterm #1 score
  - Check out your score at eTL



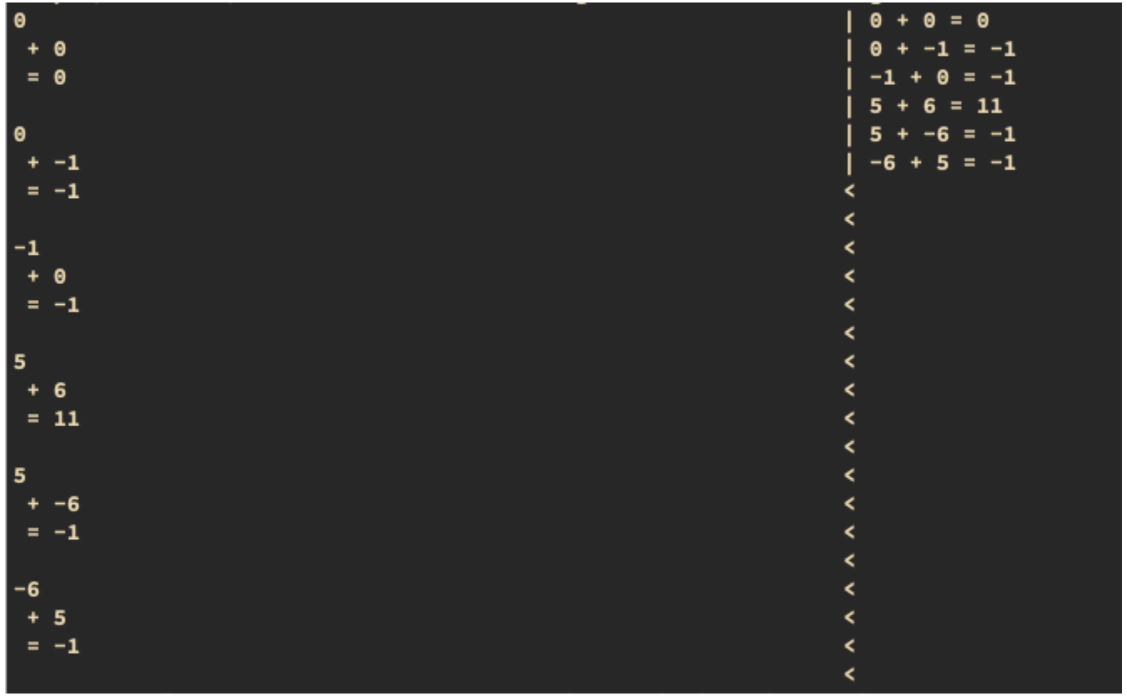
User report - Koo Kyoseung

Grade item	Grade	Feedback
Data Structures		
Midterm #1		
4/9 midterm problem #1	-	

- Claim sessions
  - This Lab class: Apr 17 Lab class (16:00 ~ 17:50)
  - Mail to TA (we will not provide the exam problems): ~ Apr 21
  - Visit our office (Engr. Bldg. 301, Room 418): 13:30~15:00 on Tue Apr 21

Announcement

- Programming Assignment
  - Please check that your output is same with the provided test case outputs
  - **From PA #2, we will not consider any cases like the below one**

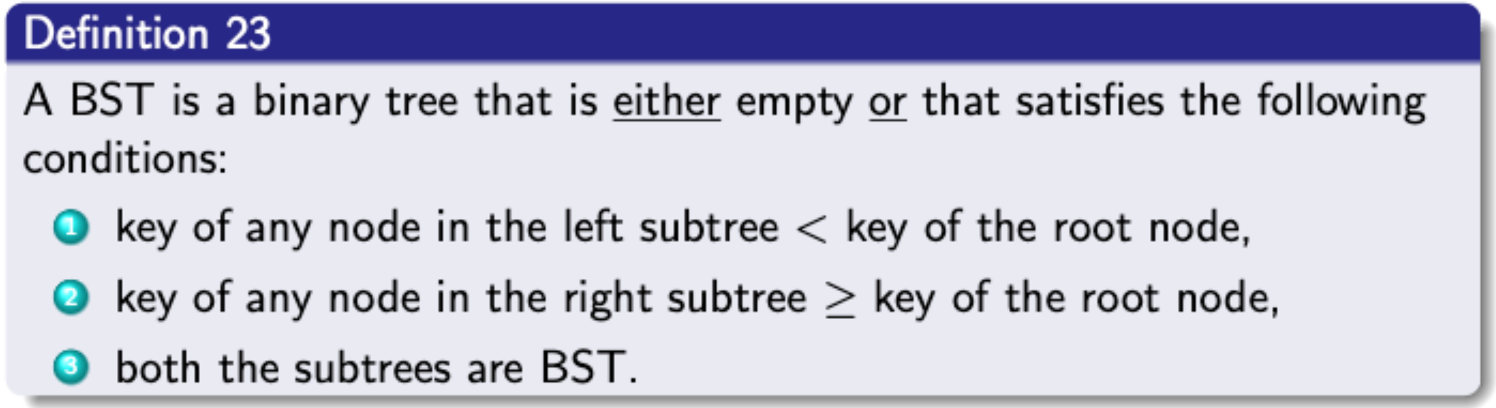


```
0                                                              | 0 + 0 = 0
 + 0                                                           | 0 + -1 = -1
 = 0                                                           | -1 + 0 = -1
                                                               | 5 + 6 = 11
0                                                              | 5 + -6 = -1
 + -1                                                          | -6 + 5 = -1
 = -1                                                          <
                                                               <
-1                                                             <
 + 0                                                           <
 = -1                                                          <
                                                               <
5                                                              <
 + 6                                                           <
 = 11                                                          <
                                                               <
5                                                              <
 + -6                                                          <
 = -1                                                          <
                                                               <
-6                                                             <
 + 5                                                           <
 = -1                                                          <
                                                               <
```

<-- Example with `` `$ diff` ``
Mistakes about using `System.out.println()`
Correct method: `System.out.print()`

Binary Search Tree

- We implemented insert method for Binary Search Tree
  - This tree satisfy the below properties
  - This BST uses their value as a key



Definition 23

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

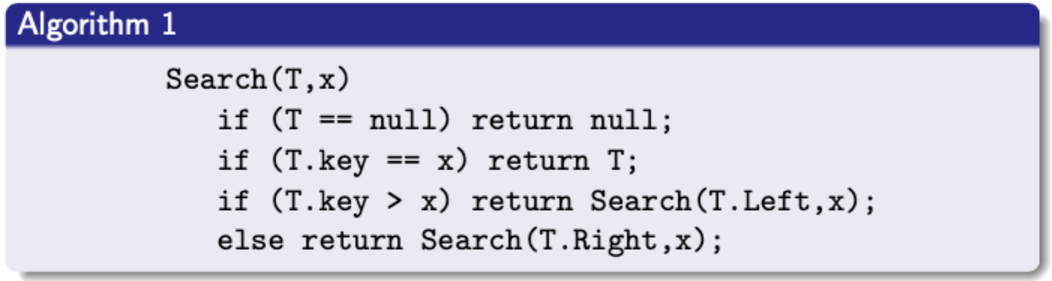
1. key of any node in the left subtree $<$ key of the root node,
2. key of any node in the right subtree $\geq$ key of the root node,
3. both the subtrees are BST.

Property of Binary Search Tree

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

Binary Search Tree

- Implement methods into Binary Search Tree
  - Write code into **TNodeImpl.java** based on previous lecture's code
  - **String findString(TNode<String> root, String value)**
  - **TNode<String> removeString(TNode<String> root, String value)**
  - **TNode<String> removeMin(TNode<String> root)**
  - **TNode<String> getMin(TNode<String> root)**

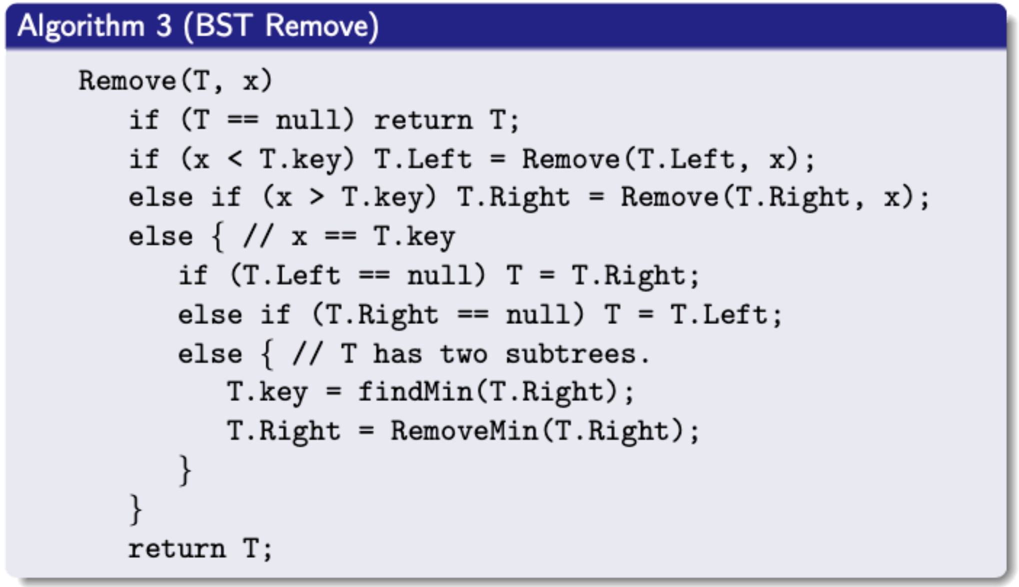


Algorithm 1

```
Search(T,x)
   if (T == null) return null;
   if (T.key == x) return T;
   if (T.key > x) return Search(T.Left,x);
   else return Search(T.Right,x);
```

^ BST Search

BST Remove -->



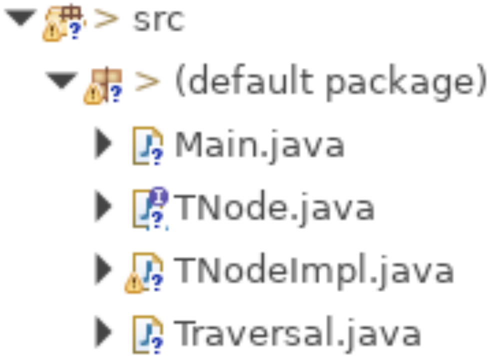
Algorithm 3 (BST Remove)

```
Remove(T, x)
   if (T == null) return T;
   if (x < T.key) T.Left = Remove(T.Left, x);
   else if (x > T.key) T.Right = Remove(T.Right, x);
   else { // x == T.key
      if (T.Left == null) T = T.Right;
      else if (T.Right == null) T = T.Left;
      else { // T has two subtrees.
         T.key = findMin(T.Right);
         T.Right = RemoveMin(T.Right);
      }
   }
   return T;
```

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

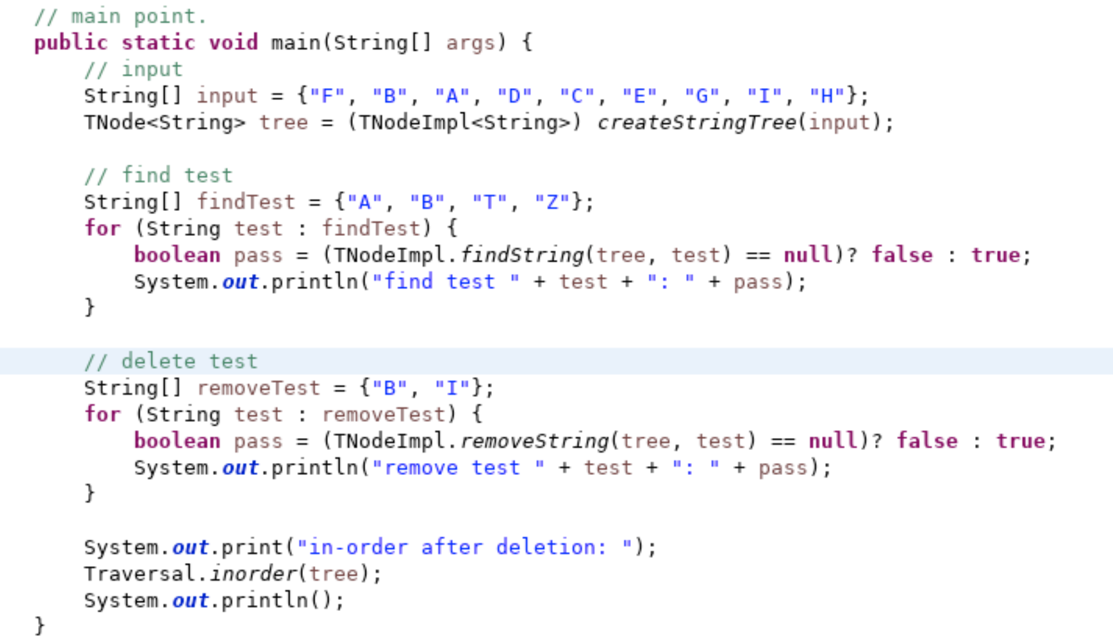
Exercises

- Fill the blank of codes
  - Write your methods into **`TNodeImpl.java`**
  - **`Replace Main.java with provided one`**



```
src
  (default package)
    Main.java          // Main (Replace with provided one)
    TNode.java         // Tree Node Interface
    TNodeImpl.java     // Tree Node Impl (TODO)
    Traversal.java     // Traversal
```

Project Structure

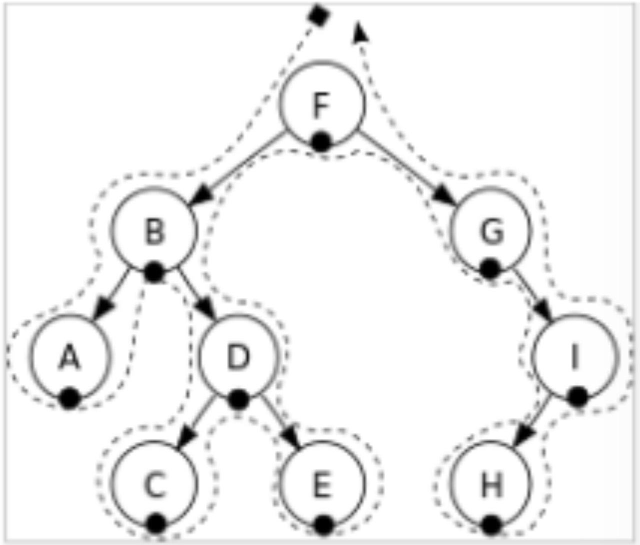


```java
// main point.
public static void main(String[] args) {
    // input
    String[] input = {"F", "B", "A", "D", "C", "E", "G", "I", "H"};
    TNode<String> tree = (TNodeImpl<String>) createStringTree(input);

    // find test
    String[] findTest = {"A", "B", "T", "Z"};
    for (String test : findTest) {
        boolean pass = (TNodeImpl.findString(tree, test) == null)? false : true;
        System.out.println("find test " + test + ": " + pass);
    }

    // delete test
    String[] removeTest = {"B", "I"};
    for (String test : removeTest) {
        boolean pass = (TNodeImpl.removeString(tree, test) == null)? false : true;
        System.out.println("remove test " + test + ": " + pass);
    }

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

Main.java

Exercises

- Result



In-order: A, B, C, D, E, F, G, H, I.

```
$ java Main
find test A: true
find test B: true
find test T: false
find test Z: false
remove test B: true
remove test I: true
in-order after deletion: A C D E F G H
```