CPE217 – Homework 6

Homework: Binary Search Tree Data Structure

Homework Due Date: 17 September 2024

Patiwet Wuttisarnwattana, Ph.D.

Department of Computer Engineering

- คำชี้แจงการส่งงาน
- ให้ทุกคนเข้าโมดูล Assignment (Link สีแดง) อ่านคำอธิบายการบ้านใน PDF และโหลด Java Starter Code เพื่อทำความเข้าใจโจทย์
- หลังจากนั้นให้ทุกคนทำการบ้านพร้อมกันกับกลุ่มของตัวเอง พร้อมปรึกษากับ Core Person ในกลุ่มตัวเองว่าจะส่งคำตอบสุดท้ายว่าเป็นอะไร
- ทุกคนสามารถตรวจคำตอบของตัวเองได้ ว่าทำมาถูกต้องหรือไม่โดยใช้โมดูล Quiz (Link สีเหลือง) แต่อาจารย์จะตรวจคะแนนจาก Core Person เท่านั้น นศ ทุกคนจะได้คะแนนเท่ากันทั้งกลุ่ม แม้ว่าคุณจะส่งโค้ดใน Link สีเหลืองแตกต่างกันก็ตามที
- นศ ต้องเติมโค้ดในพื้นที่ที่กำหนดให้เท่านั้น ห้ามประกาศตัวแปรระดับคลาสเพิ่ม ห้ามสร้างฟังก์ชันอื่น ๆ หรือเรียกใช้ฟังก์ชันพิเศษเพิ่ม (ไม่แน่ใจสามารถสอบถามได้ที่อาจารย์) ให้เติมโค้ดใน Template ของอาจารย์เท่านั้น ฝ่าฝืนหักคะแนน 25% แม้ผลลัพธ์สุดท้ายจะทำงานได้ถูกต้อง
- เมื่อ Core Person ส่งคำตอบแล้ว ให้ Core Person เข้าโมดูล Assignment (Link สีแดง) และใส่รหัสของเพื่อนในกลุ่มลงในช่องคำตอบ
- TA จะตรวจคำตอบในโมดูล Quiz และนำคะแนนมาลงในโมดูล Assignment เพื่อให้ทุกคนในกลุ่มได้คะแนนเท่ากันครับ
- โค้ดของคุณต้องมี Comment เพื่ออธิบายว่าโค้ดดังที่เห็นอยู่นี้ทำงานอะไร หรือ if นี้ใช้เพื่อแยกกรณีใดออกมา กลุ่มไหนที่ไม่มีคอมเม้นต์จะถูกหักคะแนน 50% การเขียนคอมเม้นต์ไม่ต้องเขียนละเอียดยิบ เขียนเท่าที่คุณต้องการให้ผู้ตรวจทราบก็พอ

นศ ที่จะส่งคำตอบ ท่านต้องให้คำมั่นปฏิญาณต่อคำพูดดังต่อไปนี้ หากไม่สามารถทำได้ ท่านจะไม่มีสิทธิ์ส่งงาน

- *ข้าพเจ้าและเพื่อนในกลุ่มเข้าใจและตระหนักดีว่า ในการทำการบ้านนี้ ข้าพเจ้าและเพื่อนในกลุ่มจะช่วยกันทำงานนี้ให้เสร็จสิ้นเอง โดยไม่ปรึกษาหรือแบ่งปันข้อมูลกับกลุ่มอื่น ๆ หรือบุคคลภายนอก*
- *หากข้าพเจ้าเป็นรุ่นพี่ที่กลับมาเรียนวิชานี้อีกครั้ง ข้าพเจ้าตระหนักดีว่า ข้าพเจ้าจะทำงานให้เสร็จสิ้นเองอีกครั้ง โดยไม่ดูคำตอบของปีก่อน ๆ*
- *ข้าพเจ้าจะไม่เผยแพร่เนื้อหาโจทย์การบ้านนี้ออกสู่สาธารณะโดยเด็ดขาด*
- *หากข้าพเจ้าไม่สามารถปฏิบัติตามคำมั่นนี้ได้ ข้าพเจ้ายินดีที่จะยอมรับคะแนน ศูนย์คะแนน ในทุก ๆ การบ้านโดยไม่โต้แย้ง*

การบ้านนี้ให้นักศึกษา implement Binary Search Tree (BST) โดยใช้ starter code ของอาจารย์ โดยมีคำอธิบายแต่ละส่วนดังต่อไปนี้

1. อาจารย์ได้สร้าง class Node ที่สามารถบรรจุ key ได้ค่า integer ได้ เสร็จเรียบร้อยแล้ว
    a. Node หนึ่ง Node สามารถชี้ไปยัง ลูกคนซ้าย (left child) และลูกคนขวา (right child) ได้ และจะชี้ไปยัง parent ด้วยก็ได้
    b. Constructor และอื่น ๆ ทำให้หมดแล้ว ขอให้ นศ ดูใน starter code
2. ในการแสดงผลต้นไม้
    a. VSCode เรียกใช้ command “chcp 65001” บน terminal ก่อนรันโปรแกรม
    b. Eclipse ตั้งค่าใช้ Support UTF-8
3. อาจารย์ได้ร่าง class Tree ไว้เป็น Template ซึ่ง คลาสนี้ จะทำหน้าที่ประมวลผล Node ตามคุณสมบัติของ BST
    a. ขอให้ นศ เริ่มทำการบ้านโดยการให้ class Tree นี้ ทำการสืบทอดคุณสมบัติ (inherit) class BTreePrinter ที่อาจารย์แนบมา เพื่อที่คลาสนี้จะได้สามารถแสดงแผนภาพต้นไม้ออกมาสวย ๆ ดังตัวอย่าง ผ่านฟังก์ชันที่ชื่อว่า printTree()
        - เมื่อ นศ ทำการสืบทอดคุณสมบัติของ class BTreePrinter แล้ว ขอให้ นศ Uncomment โค้ดของอาจารย์ที่ฟังก์ชัน printTree() แล้วเพิ่มเติมให้เหมาะสมและทำงานได้ตรงตาม Testcase
    b. ใน class Tree ขอให้ นศ สร้าง function ที่ชื่อว่า static void printNode(Node node) เพื่อทำการพิมพ์ค่า node.key ออกทาง console หาก node ที่รับเข้ามานั้นเป็น null ให้พิมพ์ออก console “Node not found!!!”
4. เป็นข้อตกลงของวิชานี้ว่า เมื่อไหร่ก็ตามที่อาจารย์เขียนฟังก์ชันลักษณะเป็น static นำหน้า นั่นจะเป็นสัญญาณให้ นศ รู้ว่า ฟังก์ชันนั้นจะต้องถูกพัฒนาถึงโดยมีการเรียกตัวเอง (Recursion)
5. ให้ class Tree มี method ดังต่อไปนี้
    a. public static Node findKey(Node node, int search_key) ทำหน้าที่หา node ที่บรรจุ search_key แบบ recursive ตามที่เรียนในห้อง
    b. public Node findKey(int search_key) ทำหน้าที่หา Node ที่บรรจุใน Tree ที่มี key ดังที่ระบุ โดยเรียกใช้บริการ static findKey() อีกทีหนึ่ง
    c. public static Node findMin(Node node) ทำหน้าที่หา node ที่มีค่า key น้อยที่สุด แบบ recursive ตามที่เรียนในห้อง
    d. public Node findMin() ทำหน้าที่หา Node ที่บรรจุใน Tree ที่มีค่า key น้อยที่สุด โดยเรียกใช้บริการ static findMin() อีกทีหนึ่ง
    e. public static Node findMax(Node node) ทำหน้าที่หา node ที่มีค่า key มากที่สุด แบบ recursive ตามที่เรียนในห้อง
    f. public Node findMax() ทำหน้าที่หา Node ที่บรรจุใน Tree ที่มีค่า key มากที่สุด โดยเรียกใช้บริการ static findMax() อีกทีหนึ่ง
    g. public static Node findClosestLeaf(Node node, int search_key) ทำหน้าที่หา node ที่มี null Node ที่สามารถนำ search_key ไปห้อยเป็น Node ใหม่ได้ แบบ recursive ตามที่เรียนในห้อง

h. public Node findClosestLeaf(int search_key) ทำหน้าที่หา node ที่มี null Node ที่อยู่ใน Tree ต้นปัจจุบันที่สามารถนำ search_key ไปห้อยเป็น Node ใหม่ได้ โดยเรียกใช้บริการ static findClosestLeaf () อีกทีหนึ่ง

i. public Node findClosest(int search_key) ทำหน้าที่หา Node ที่มี ใกล้เคียงกับ search_key มากที่สุด

- *เฉพาะฟังก์ชันนี้ในการบ้านนี้ กำหนดให้ใช้ while loop ทำ อย่าใช้ Recursive นะครับ*

j. public void insertKey(int key) ทำหน้าที่สร้าง Node ใหม่ที่บรรจุค่า key แล้วนำไปต่อใน BST ตามที่เรียนในห้อง (ให้ทำ findClosestLeaf แล้วเอา Key ใหม่ไปห้อย) ถ้าหาก key ที่บรรจุเข้ามาใหม่มีอยู่แล้วใน Node ใด Node หนึ่งของ Tree ให้พิมพ์ออกทางหน้าจอว่า “Duplicated key!!!” แล้วไม่ต้องทำอะไร

- *เฉพาะฟังก์ชันนี้ในการบ้านนี้ กำหนดให้เรียกใช้บริการ findClosestLead() นะครับ อย่าทำ insert แบบ Recursive*

k. public void printPreOrderDFT() และคู่หู static function ที่มีชื่อเดียวกัน ให้ทำหน้าที่พิมพ์ค่า key ของทุก ๆ Node ตามลำดับ Pre-Order Depth First Traversal ตามที่เรียนในห้อง

- ให้ pattern การพิมพ์ออกทาง console ให้เป็นไปดังตัวอย่างด้านล่าง เริ่มต้นด้วยคำว่า “PreOrder DFT node sequence [” ลงท้ายด้วย “]”

l. public void printInOrderDFT() และคู่หู static function ที่มีชื่อเดียวกัน ให้ทำหน้าที่พิมพ์ค่า key ของทุก ๆ Node ตามลำดับ In-Order Depth First Traversal ตามที่เรียนในห้อง

- ให้ pattern การพิมพ์ออกทาง console ให้เป็นไปดังตัวอย่างด้านล่าง เริ่มต้นด้วยคำว่า “InOrder DFT node sequence [” ลงท้ายด้วย “]”

m. public void printPostOrderDFT() และคู่หู static function ที่มีชื่อเดียวกัน ให้ทำหน้าที่พิมพ์ค่า key ของทุก ๆ Node ตามลำดับ Post-Order Depth First Traversal ตามที่เรียนในห้อง

- ให้ pattern การพิมพ์ออกทาง console ให้เป็นไปดังตัวอย่างด้านล่าง เริ่มต้นด้วยคำว่า “PostOrder DFT node sequence [” ลงท้ายด้วย “]”

n. public static int height(Node node) ทำหน้าที่หาว่า node นี้อยู่ที่ความสูงที่เท่าไหร่เมื่อเทียบกับลูกที่อยู่ลึกที่สุด ... *นศ จงจำไว้ว่า เมื่อไหร่ก็ตามที่อาจารย์เขียนฟังก์ชันลักษณะเป็น static นำหน้า จะเป็นสัญญาณให้ นศ รู้ว่าฟังก์ชันนี้ควรจะมีการเรียกตัวเอง (Recursion)*

o. public static int size(Node node) ทำหน้าที่หาว่า node กับลูก ๆ รวมกันแล้วมีจำนวนเท่าไหร่ (ตามความหมายของการหา Tree size) ... *นศ จงจำไว้ว่า เมื่อไหร่ก็ตามที่อาจารย์เขียนฟังก์ชันลักษณะเป็น static นำหน้า จะเป็นสัญญาณให้ นศ รู้ว่าฟังก์ชันนี้ควรจะมีการเรียกตัวเอง (Recursion)*

p. public static int depth(Node root, Node node) ทำหน้าที่หาว่า หากกำหนดให้ node นี้เป็นส่วนหนึ่งของ Tree แล้ว node นี้จะมีความลึกเป็นเท่าไหร่ เมื่อเทียบกับ root (ตามความหมายของการหา Node depth) ...

q. public int treeHeight() ทำหน้าที่คำนวณว่าต้นไม้ต้นนี้มีความสูงเท่าไหร่ จงเขียนโค้ดในหนึ่งบรรทัด (คำใบ้: เรียกใช้บริการ static height(Node node))

r. public int treeSize() ทำหน้าที่คำนวณว่าต้นไม้ต้นนี้มีบรรจุ Node ไปแล้วทั้งหมดกี่ Node จงเขียนโค้ดในหนึ่งบรรทัด (คำใบ้: เรียกใช้บริการ static size(Node node))

s. public int treeDepth() ทำหน้าที่คำนวณว่าต้นไม้ต้นนี้มีความลึกเท่าไหร่ จงเขียนโค้ดในหนึ่งบรรทัด (คำใบ้: เรียกใช้บริการ static depth(Node root, Node node))

t. public Node findKthSmallest(int k) และคู่หู static function ที่มีชื่อเดียวกัน ให้ทำหน้าที่หา Node ที่บรรจุใน Tree ที่มีค่า key เล็กเป็นอันดับที่ k (k=1 แปลว่า มีค่า key เล็กที่สุด, k=2 แปลว่า มีค่า key เล็กเป็นอันดับที่สอง)

u. public static Node findNext(Node node) ทำหน้าที่หาว่า Node ที่มีค่า key อยู่มากกว่าขึ้นไปอีกค่าหนึ่งของ node (input) คือ Node ใด

v. public static Node leftDescendant(Node node) ทำหน้าที่หา descendant Node ที่อยู่ด้านซ้ายสุดของ node (ซึ่งทำงานเหมือน findMin นั่นแหละ)

w. public static Node rightAncestor(Node node) ทำหน้าที่หา ancestor Node ที่อยู่ด้านขวาแรกของ node

x. public List rangeSearch(int x, int y) ทำหน้าที่ค้นหา Node ที่มี key อยู่ระหว่างค่า x กับค่า y โดยค่า x ≤ key ≤ y ซึ่ง Node ทั้งหมดที่เข้าเงื่อนไขนี้ให้คุณบรรจุเข้าไปใน List ผ่านฟังก์ชัน append() (เช่น list.append(node)) แล้วเสร็จสิ้นกระบวนการแล้วก็ return list นี้ออกไป

y. public void deleteKey(int key) และคู่หู static function ที่มีชื่อเดียวกัน ให้ทำหน้าที่ค้นหา Node ที่บรรจุอยู่แล้วใน BST แล้วทำการลบออก ตามที่เรียนในห้อง

- ถ้าหาก Node ที่คุณต้องการลบเป็น root Node ให้คุณทำการ implement ใน function นี้เลย
- แต่ถ้า Node ที่คุณต้องการลบไม่ใช่ root Node ให้คุณเรียกคู่หู static deleteKey() เพื่อทำการลบ node ต่อไป
- กรณีที่เป็น root จะมีกรณีที่ต้องพิจารณาอยู่ 6 กรณี เช่น Empty Tree, Single Node Root, Root with only left child, Root with only right child, Root with both children
- กรณีที่เป็น node ใด ๆ ให้ไปลบใน static deleteKey() โดยมีกรณีพิจารณาอยู่ 7 กรณี เช่น Leaf node on parent's left, Leaf node on parent's right, Node with left child on parent's left, ...

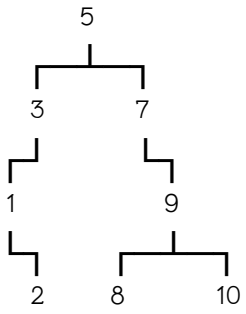
6. ตัวอย่างการทำงาน

Java code

```
public static void main(String[] args) {
        Tree tree = new Tree();
        tree.printTree();

        int[] keyList = {5, 3, 1, 2, 7, 9, 10, 8};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree();

        Node node = tree.findKey(3); Tree.printNode(node);
        node = tree.findKey(4); Tree.printNode(node);
        node = tree.findClosest(4); Tree.printNode(node);
        node = tree.findClosest(3); Tree.printNode(node);
        node = tree.findClosest(-999); Tree.printNode(node);
        node = tree.findClosest(999); Tree.printNode(node);
}
```

Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)

```
Empty tree!!!
```



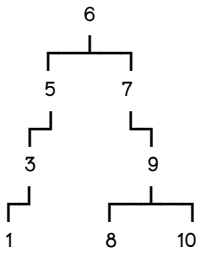

```
3
Node not found!!!
5
3
1
10
```

Java code

```
public static void main(String[] args) {
        Tree tree = new Tree();

        int[] keyList = {6, 7, 9, 5, 3, 9, 10, 8, 1};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree();

        List list = tree.rangeSearch(4, 8);
        list.printList();
}
```

Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)

```
Duplicated key!!!
```



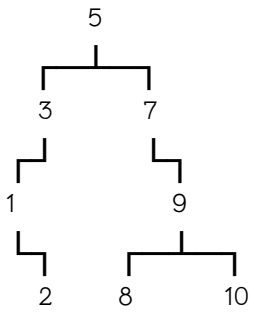

```
[Head] 5 6 7 8 [Tail]
```

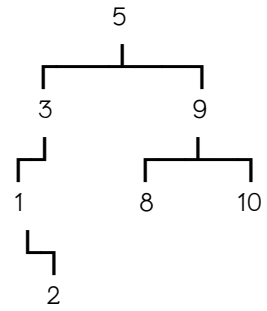
Java code

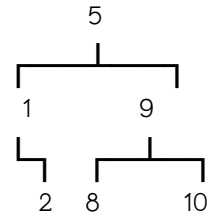
```
public static void main(String[] args) {
        Tree tree = new Tree();

        int[] keyList = {5, 3, 1, 2, 7, 9, 10, 8};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree(); System.out.println("-----------------");

        tree.deleteKey(7);
        tree.printTree(); System.out.println("-----------------");
        tree.deleteKey(3);
        tree.printTree();
```

```
        tree.deleteKey(9);
        tree.deleteKey(1);
        tree.deleteKey(5);
        tree.deleteKey(5); System.out.println("----------------");
        tree.printTree();

}
```

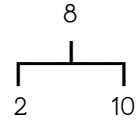
Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)







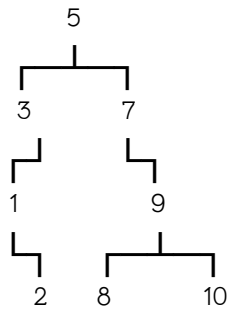
Key not found!!!



Java code

```
public static void main(String[] args) {
        Tree tree = new Tree();
        int[] keyList = {5, 3, 1, 2, 7, 9, 10, 8};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree();
        Node node = tree.findKey(4);
        Tree.printNode(node);
        node = tree.findClosest(4);
        Tree.printNode(node);
        node = Tree.findNext(node);
        Tree.printNode(node);
        node = Tree.findNext(node);
        Tree.printNode(node);
        node = Tree.findNext(node);
        Tree.printNode(node);
}
```

Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)

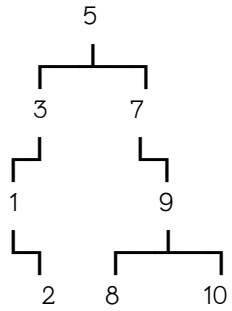



```
Node not found!!!
5
7
8
9
```

Java code

```
public static void main(String[] args) {
        Tree tree = new Tree();


        int[] keyList = {5, 3, 1, 2, 7, 9, 10, 8};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree();


        tree.printPreOrderDFT();
        tree.printInOrderDFT();
        tree.printPostOrderDFT();
}
```

Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)



PreOrder DFT node sequence [5 3 1 2 7 9 8 10]

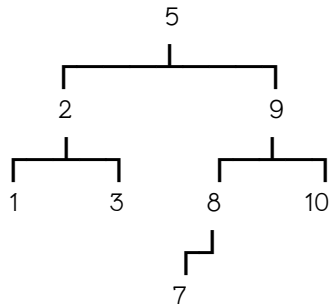
InOrder DFT node sequence [1 2 3 5 7 8 9 10]

PostOrder DFT node sequence [2 1 3 8 10 9 7 5]

Java code

```
public static void main(String[] args) {
        Tree tree = new Tree();


        int[] keyList = {5, 2, 3, 9, 1, 10, 8, 7};
        for (int i=0; i<keyList.length; i++)
            tree.insertKey(keyList[i]);
        tree.printTree();
        System.out.println(tree.treeDepth());
        System.out.println(tree.treeHeight());
        Node node = tree.findKey(9);
        System.out.println(Tree.depth(tree.root, node));
        System.out.println(Tree.height(node));


        node = tree.findMax();
        Tree.printNode(node);
        node = tree.findMin();
        Tree.printNode(node);
        node = tree.findKthSmallest(6);
        Tree.printNode(node);
        node = tree.findKthSmallest(3);
        Tree.printNode(node);
}
```

Output (แผนภาพต้นไม้ด้านล่างอาจแตกต่างกับผลลัพธ์จริงเล็กน้อย)




```
3
3
1
2
10
```

1 8 3

7. โปรดใช้ Starter code ที่อาจารย์แนบให้