Lab05: Binary Tree

26 กุมภาพันธ์ 2568

ตั้งแต่แล็บวันนี้เป็นต้นไป นักศึกษาจะเปลี่ยนมาเขียนโปรแกรมด้วยภาษาจาวาเป็นหลัก เนื่องจากการจัดการกับเมมโมรี่จะมีความง่ายมากกว่าในภาษาซี ซึ่งในเรื่องของ Tree และ Graph จะต้องการการจัดการกับเมมโมรี่ที่ซับซ้อนมากขึ้น

ปลูกต้นไม้ (Planting Tree)

เราจะมาฝึกสร้างต้นไม้กัน Data Structure แบบต้นไม้ (Tree) ก็มีความคล้ายกับ Stack, Queue ที่ด้านในนั้นจะประกอบไปด้วยโหนด (Node) หลาย ๆ โหนดด้วยกัน โดยวันนี้จะเป็นการสร้าง Binary Tree ซึ่งเป็นต้นไม้ที่เรียบง่ายที่สุดในบรรดาต้นไม้ทั้งหมด ซึ่งเป็นต้นไม้ที่โหนดพ่อแม่ (Parent Node) มีโหนดลูก (Child Node) ได้มากที่สุด**สอง**โหนด

```
typedef struct node{
    int val;
    struct node* leftChild;
    struct node* rightChild;
}NODE_T;
```

Figure 1: โค้ดการเขียนโหนดสำหรับ Binary Tree ในภาษาซี

ซึ่งโค้ดที่เทียบเท่ากับโค้ดภาษาซีด้านบนในภาษาจาวาจะเป็นดังนี้

```
class Node{
    int val;
    Node leftChild;
    Node rightChild;

    /* methods .... */
}
```

Figure 2: โค้ดการเขียนโหนดสำหรับ Binary Tree ในภาษาจาวา

วันนี้เราจะมาฝึกสร้าง Binary Tree แบบในไฟล์เลคเชอร์ที่อาจารย์ได้สอนไปเมื่อเช้านี้ โดยให้โครงสร้างของโหนดเป็นไปดังตัวอย่างด้านล่างต่อไปนี้

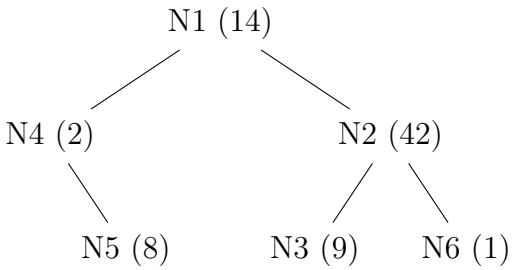
หากเราสร้างโหนดมาทั้งหมด 6 โหนด โดยให้ข้อมูลของโหนดนั้น ๆ จะมี 2 ข้อมูลหลักคือ **ชื่อ** และ **ค่าของโหนดนั้น ๆ**

1. **N1 14** สร้างโหนดที่มีชื่อว่า "N1" และโหนดนั้นมีค่าเป็น 14
2. **N2 42** สร้างโหนดที่มีชื่อว่า "N2" และโหนดนั้นมีค่าเป็น 42
3. **N3 9**
4. **N4 2**
5. **N5 8**
6. **N6 1**

หลังจากนั้นเราจะมีเส้นเชื่อม (Edge) ทั้งหมด 5 เส้น และลักษณะของเส้นเชื่อมเป็นดังต่อไปนี้ *(ลองไปนึก ๆ ดูว่าทำไมถึงได้ 5)*

- **{N1 N2 R}** ให้โหนด **N2** เป็นลูกทาง**ขวา** ของโหนด **N1** เราจะแทนสัญลักษณ์ว่า
- **{N2 N3 L}**
- **{N1 N4 L}**
- **{N2 N6 R}**
- **{N4 N5 R}**

หากนักศึกษาทำความเข้าใจกับคำสั่งด้านบนนี้ได้ นักศึกษาจะได้โมเดลของต้นไม้ดังต่อไปนี้



และเราจะสังเกตได้ว่า ต้นไม้ต้นนี้มีราก (Root) คือโหนดที่มีชื่อว่า **N1** นั่นเอง

เมื่อเราได้ต้นไม้มาแล้ว เราจะทำการท่องไปในต้นไม้ (Traverse) ซึ่งโดยทั่วไปแล้ว Binary Tree มีการ Traverse 3 แบบ นั่นก็คือ

1. Pre-order traversal
2. In-order traversal
3. Post-order traversal

โดยที่การทำ Pre-order traversal จะทำให้ได้ลำดับ **{14 2 8 42 9 1}** การทำ In-order traversal จะได้ลำดับ **{2 8 14 9 42 1}** และการทำ Post-order traversal จะได้ลำดับ **{8 2 9 1 42 14}** *(หากยังไม่เข้าใจ ลองไปทบทวนในไฟล์เลคเชอร์ของวันนี้ดู)*

ให้นักศึกษารับข้อมูลของโหนด (Node) ข้อมูลของเส้นเชื่อม (Edge) ตามรูปแบบด้านบน และให้พิมพ์ผลลัพธ์การท่องไปในต้นไม้ (Traversal) แบบ Pre-order, In-order และ Post-order ตามลำดับ (หากยังไม่เข้าใจรูปแบบ สามารถดูตัวอย่างข้อมูลนำเข้า-ส่งออกได้)

ข้อมูลนำเข้า (Input)

บรรทัดที่ 1	จำนวนเต็ม N แทนจำนวนโหนด (Node) โดยที่ $1 \leq N \leq 20$
บรรทัดที่ 2 ถึง $N+1$	ข้อมูลของแต่ละโหนด โดยให้อยู่ในรูปแบบของ `name value` คั่นด้วยช่องว่างหนึ่งช่อง
บรรทัดที่ $N+2$ ถึง $2N$	ข้อมูลของแต่ละเส้นเชื่อม โดยให้อยู่ในรูปแบบของ `parent child direction` โดยให้ `direction` มีค่าได้แค่ `L` หรือ `R` เท่านั้น
บรรทัดที่ $2N+1$	ข้อมูลของโหนดที่เป็นรากของต้นไม้ (Root) โดยให้พิมพ์ชื่อของโหนดที่ต้องการเป็นราก

ข้อมูลส่งออก (Output)

บรรทัดที่ 1	ผลลัพธ์ของการทำ Pre-order traversal
บรรทัดที่ 2	ผลลัพธ์ของการทำ In-order traversal
บรรทัดที่ 3	ผลลัพธ์ของการทำ Post-order traversal

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
6 N1 14 N2 42 N3 9 N4 2 N5 8 N6 1 N1 N2 R N2 N3 L N1 N4 L N2 N6 R N4 N5 R N1	14 2 8 42 9 1 2 8 14 9 42 1 8 2 9 1 42 14
5 Somsak 49 Somsri 40 Sompong 52 Somsong 56 SomO 11 Somsak Somsri L Somsak Somsong R Somsri Sompong L Somsong SomO R Somsak	49 40 56 52 11 56 40 49 52 11 56 40 11 52 49

หมายเหตุ: หลังจากนี้ขอให้นักศึกษาส่งไฟล์โค้ดที่เป็นไฟล์ประเภท `.java` ใน **LEB2** โดยขอให้**ตั้งชื่อตามนี้เท่านั้น**

`Lab05_PlantingTree.java`