บทที่ 11 Trees, Binary Trees

บทเรียนย่อย

11.1 Trees Concept

11.2 Trees Component

11.3 Trees Traversal

11.4 Binary Trees Concept

11.5 Binary Trees Operations and Implementation

11.6 Binary Search Trees Concept

11.7 Binary Search Trees Implementation

วัตถุประสงค์

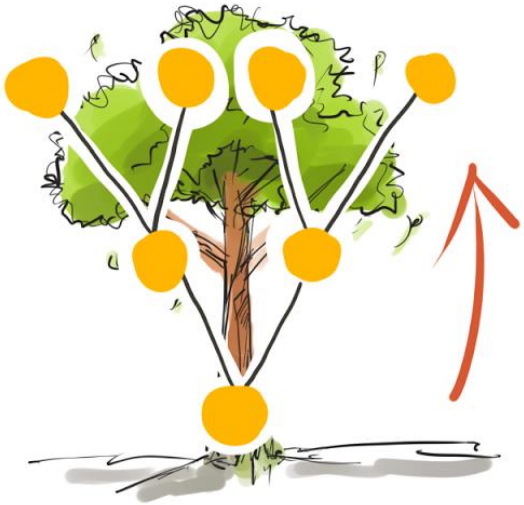
- นิสิตมีความรู้ และความเข้าใจเกี่ยวกับแนวคิด และองค์ประกอบสำคัญต่าง ๆ ในการ จัดการโครงสร้างข้อมูลในรูปแบบของ Tress
- นิสิตสามารถเขียนโปรแกรมเพื่อดำเนินการตามแนวคิดของ Trees
- นิสิตสามารถนำแนวคิดของ Trees มาประยุกต์ใช้งานในการพัฒนาโปรแกรม

บทที่ 11 Tree

บทเรียนย่อย

11.1 Trees Concept

โครงสร้างข้อมูลแบบต้นไม้ เป็นโครงสร้างโครงสร้างชนิดไม่เชิงเส้น (Non-linear) ที่มีการเชื่อมกันของ โหนด (node) แบบไม่เป็นวงกลม ไม่มีโหนดที่มีข้อมูลว่างเปล่า (null) และมีข้อมูลที่มีความสัมพันธ์ระหว่าง โหนด ที่มีความสัมพันธ์เป็นลำดับชั้น (Hierarchical Relationship)



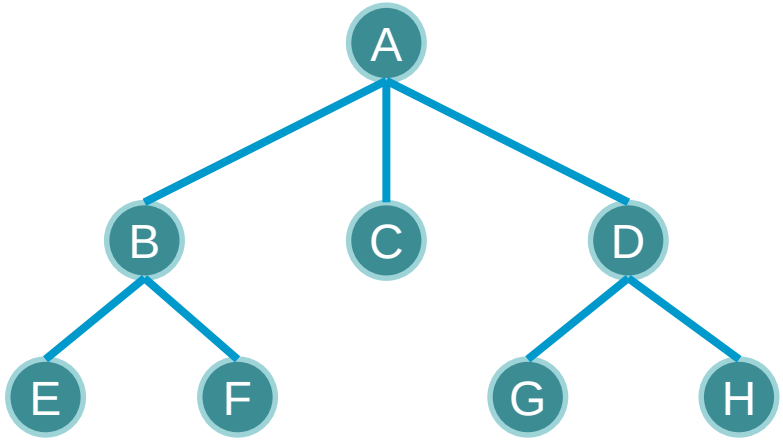
อ้างอิง : https://www.tamemo.com/post/22/data-structure-tree-มาปลูกต้นไม้กันไหม

บทที่ 11 Tree

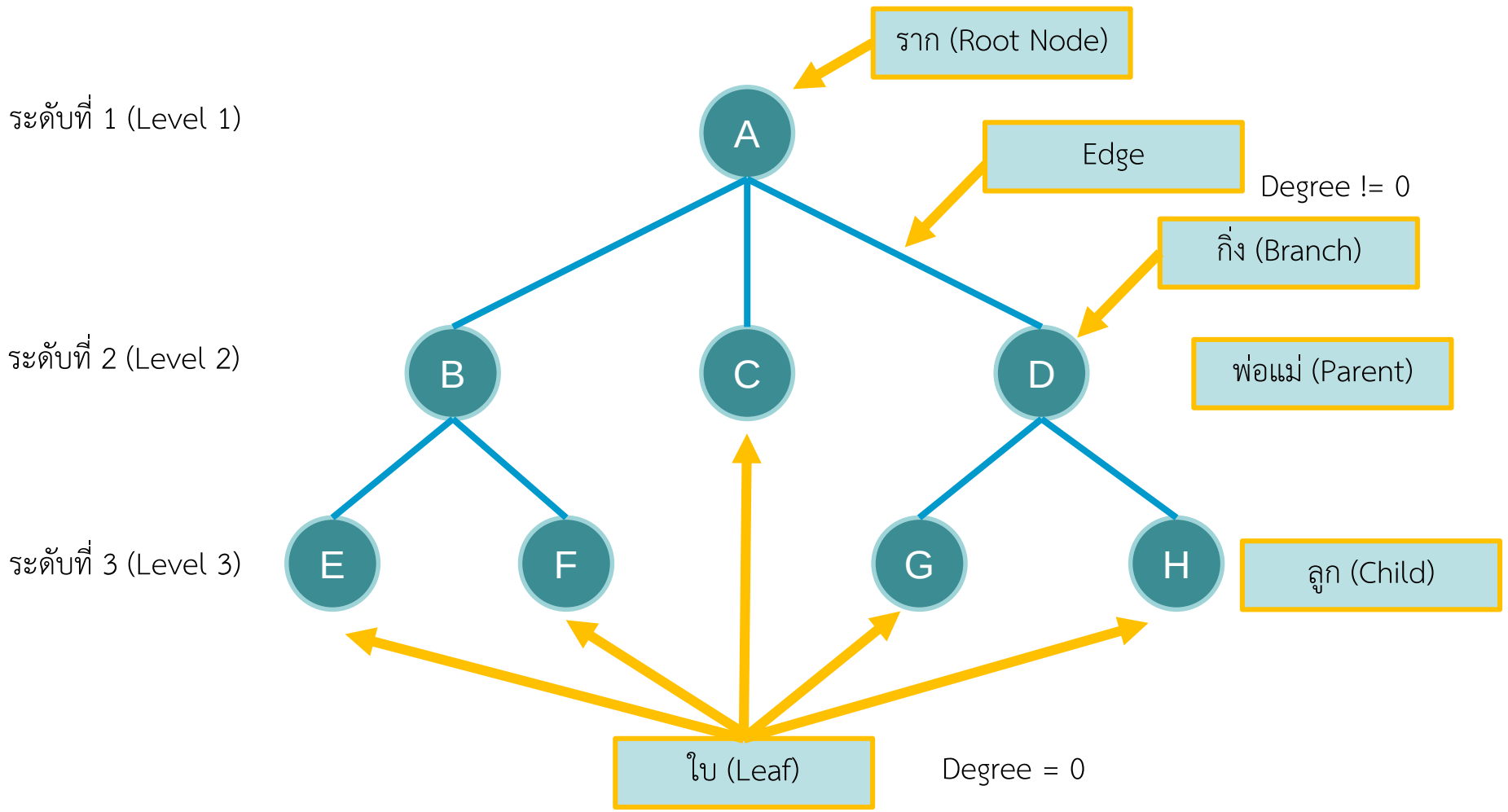
บทเรียนย่อย

11.2 Trees Component

โครงสร้างแบบต้นไม้ ประกอบไปด้วย โหนด (node) และ ลิงก์ (link) หรือ Edge ส่วน โหนด จะมีชื่อเรียกตามลำดับชั้น และความสำคัญ



ส่วนประกอบของโครงสร้างแบบต้นไม้



บทที่ 11 Tree

บทเรียนย่อย

11.3 Trees Traversal

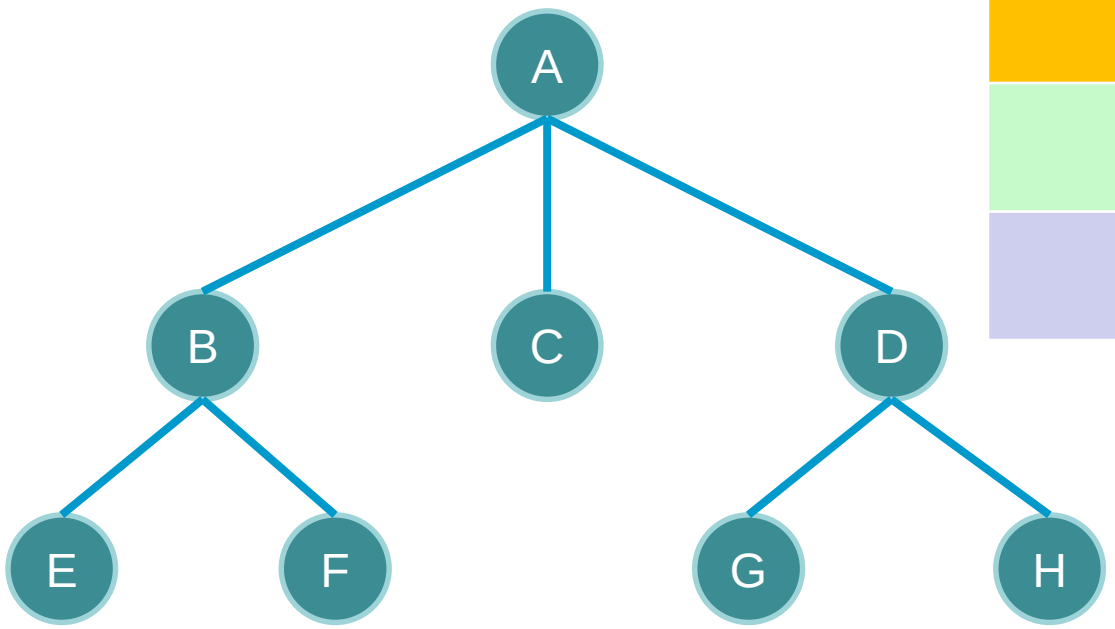
การสำรวจไปในต้นไม้ เนื่องจาก Tree เป็นโครงสร้างแบบ Non-linear ทำให้ไม่สามารถแสดงข้อมูลแบบเดิมได้ จึ่งจำเป็นต้องใช้วิธีการสำรวจ โดยวิธีการสำรวจในโครงสร้างต้นไม้หลัก ๆ มี 3 วิธี ดังนี้

- Pre-Order Traversal
- In-Order Traversal
- Post-Order Traversal

การสำรวจไปในต้นไม้

ลำดับ	Pre-Order	In-Order	Post-Order
1	ประมวลผล	ค้นหาทางซ้าย	ค้นหาทางซ้าย
2	ค้นหาทางซ้าย	ประมวลผล	ค้นหาทางขวา
3	ค้นหาทางขวา	ค้นหาทางขวา	ประมวลผล

Pre-Order Traversal



Pre-Order

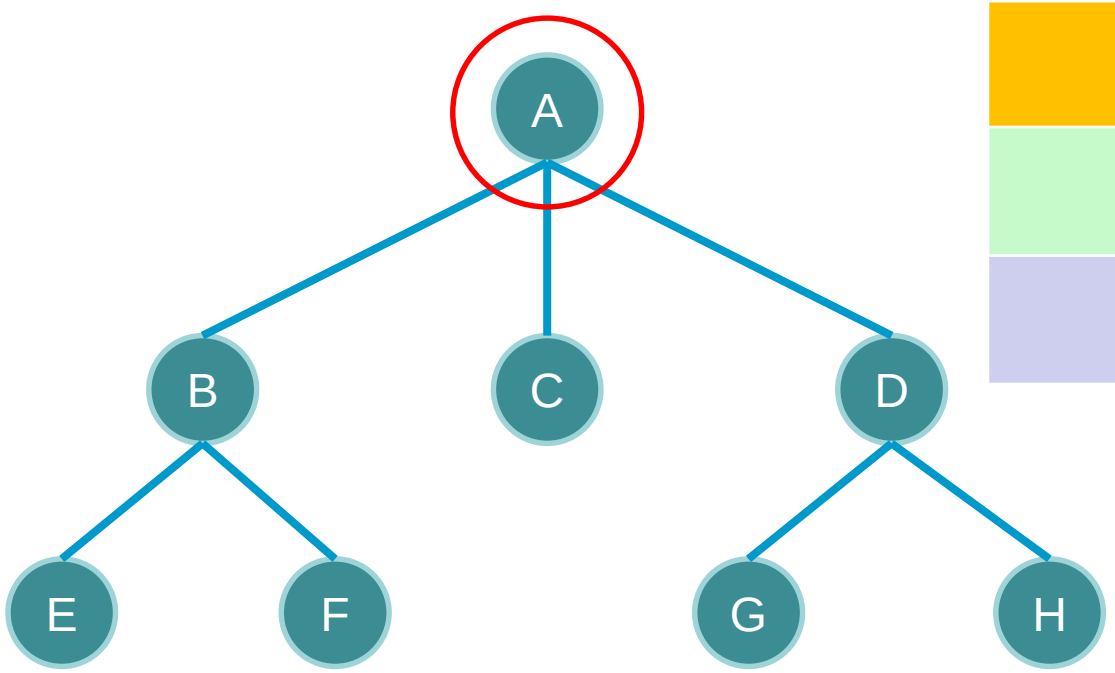
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ :

Pre-Order Traversal



Pre-Order

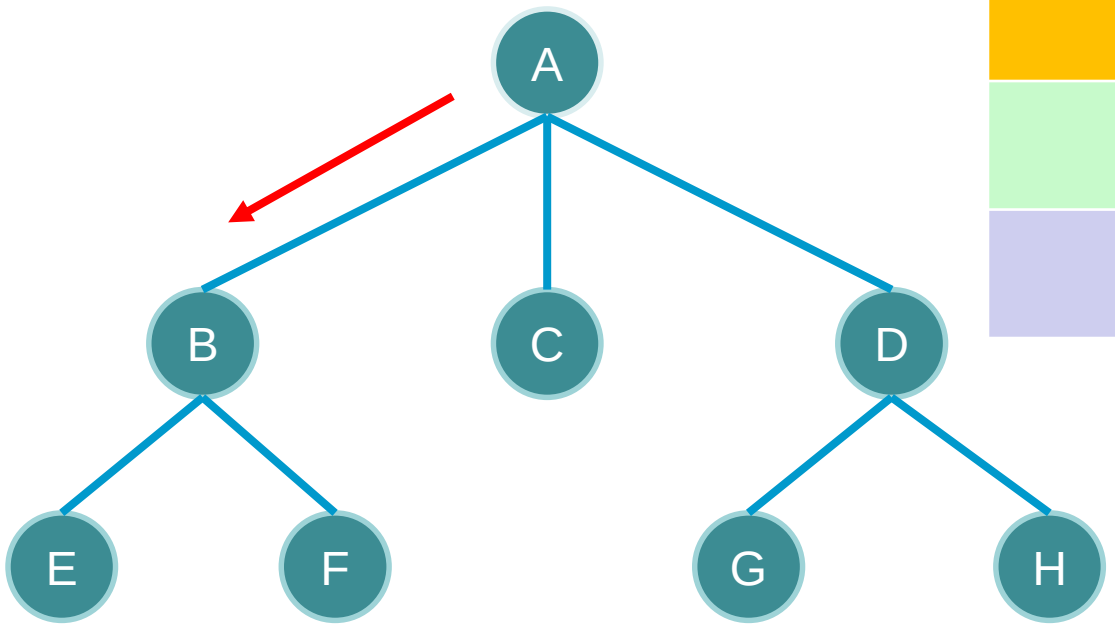
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ :

Pre-Order Traversal



Pre-Order

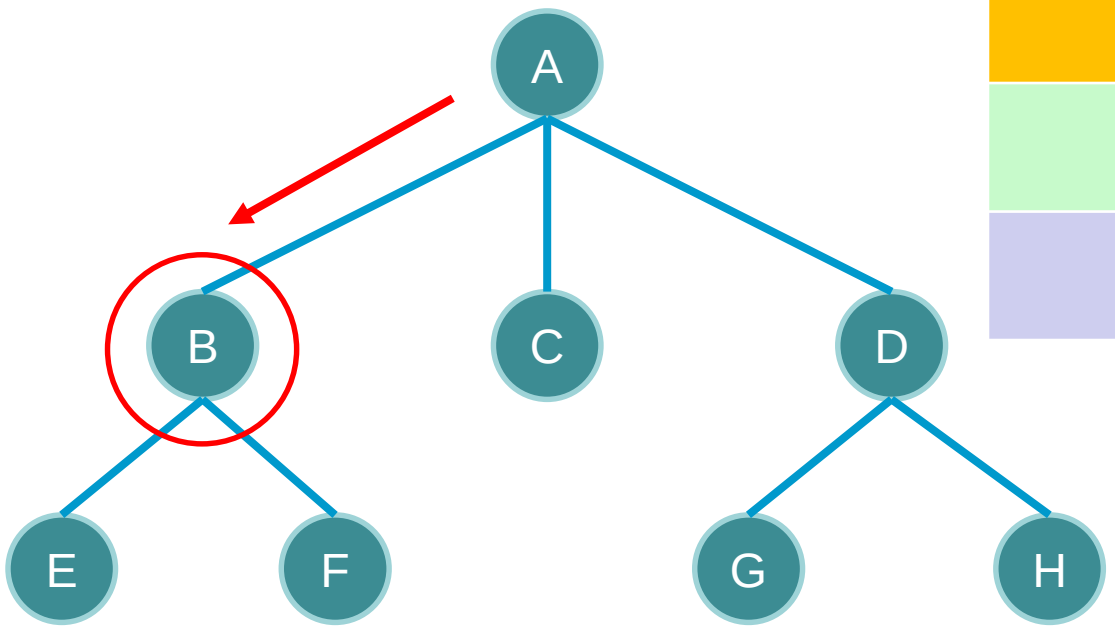
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A

Pre-Order Traversal



Pre-Order

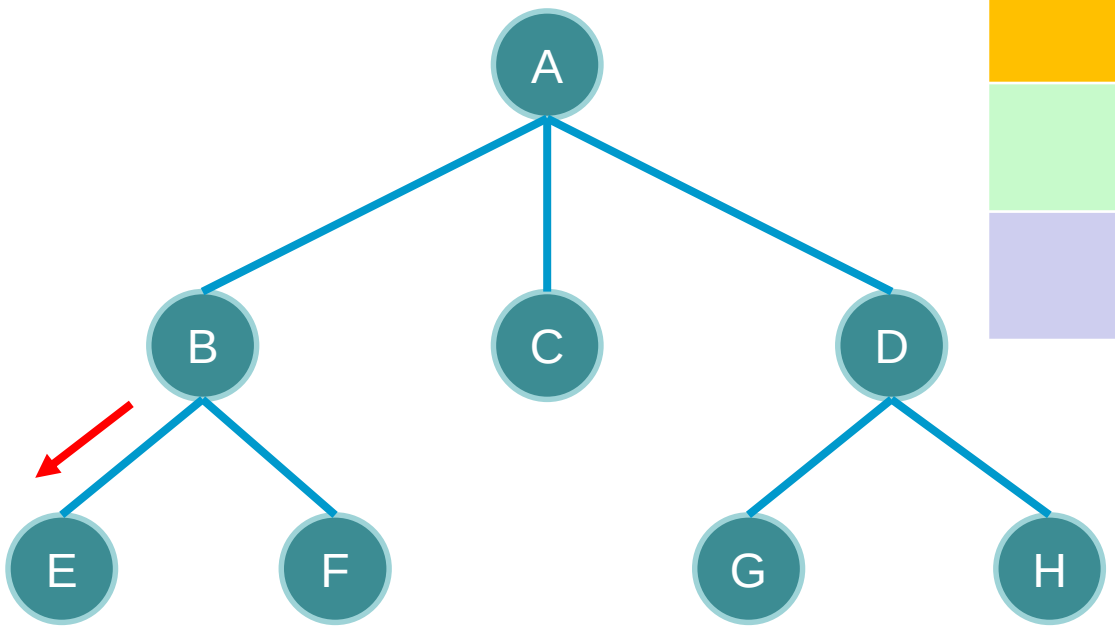
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A

Pre-Order Traversal



Pre-Order

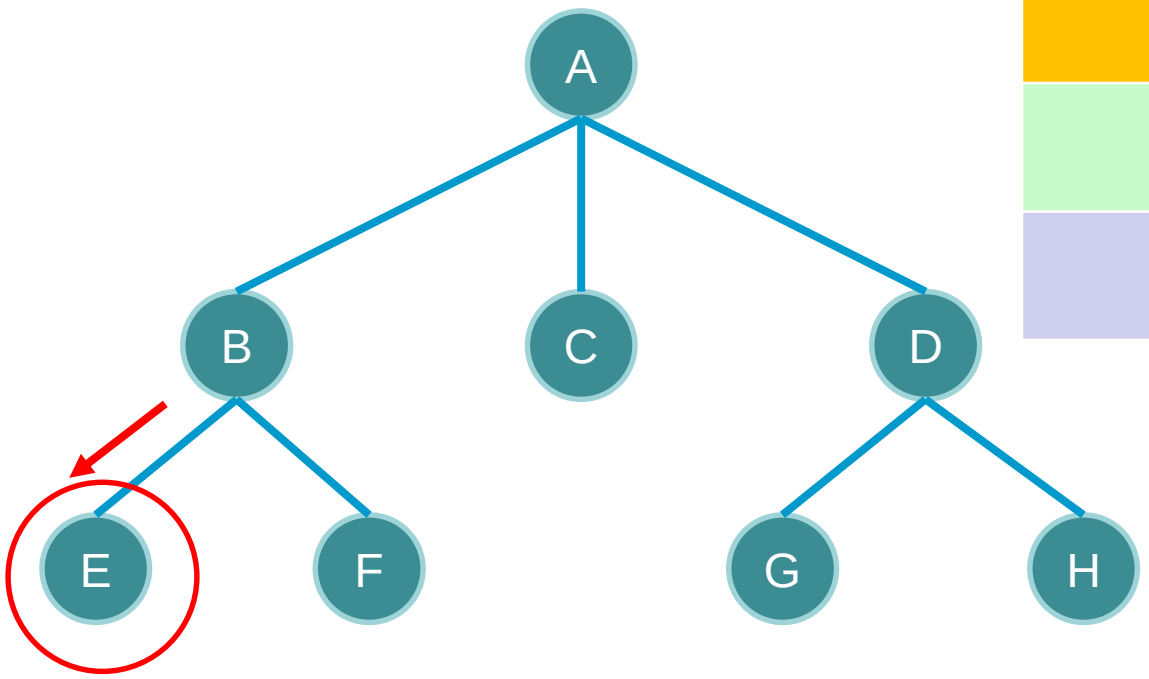
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B

Pre-Order Traversal



Pre-Order

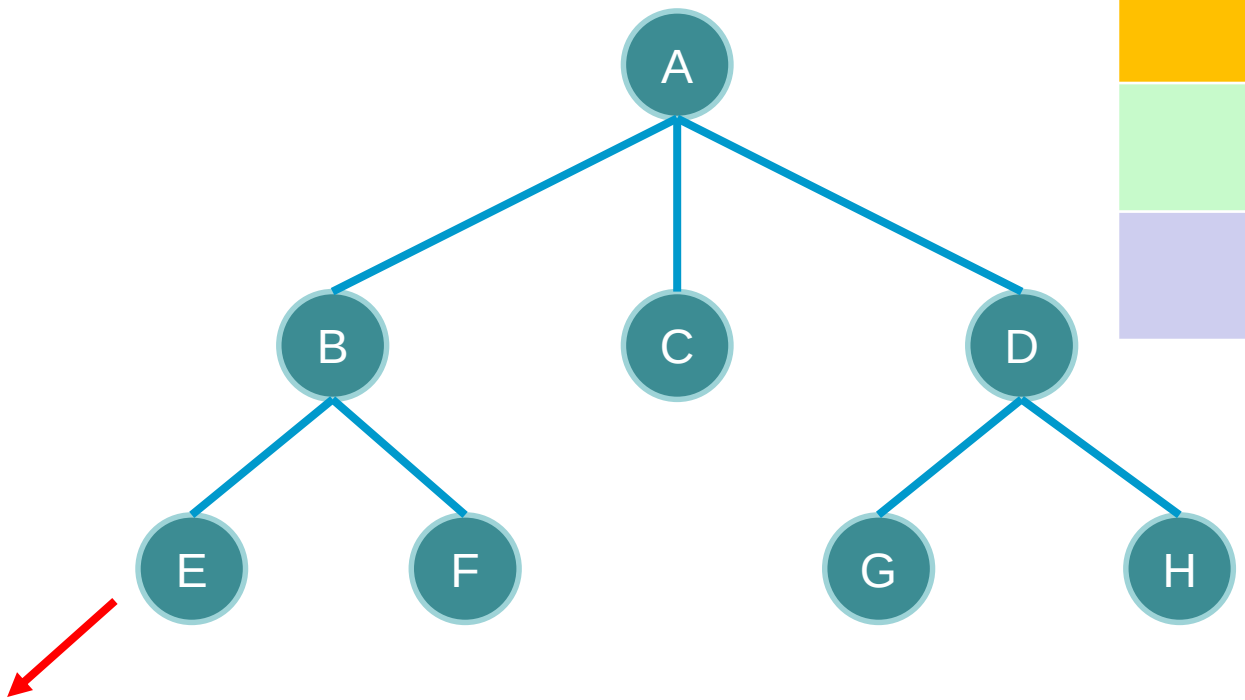
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B

Pre-Order Traversal



Pre-Order

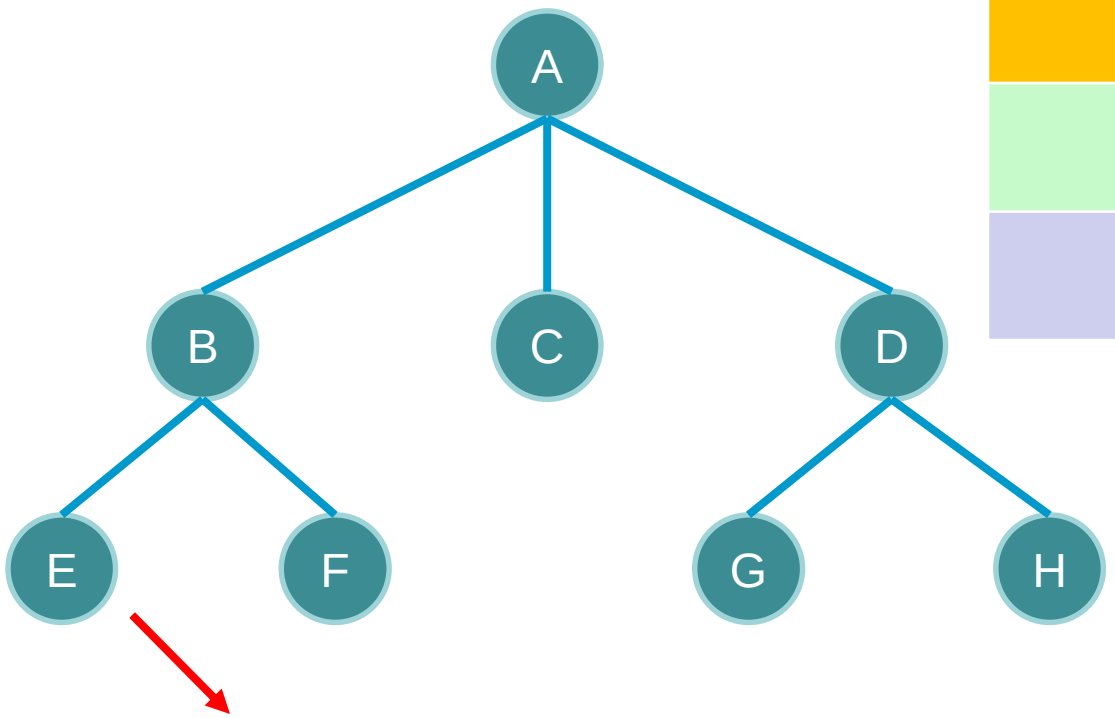
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E

Pre-Order Traversal



Pre-Order

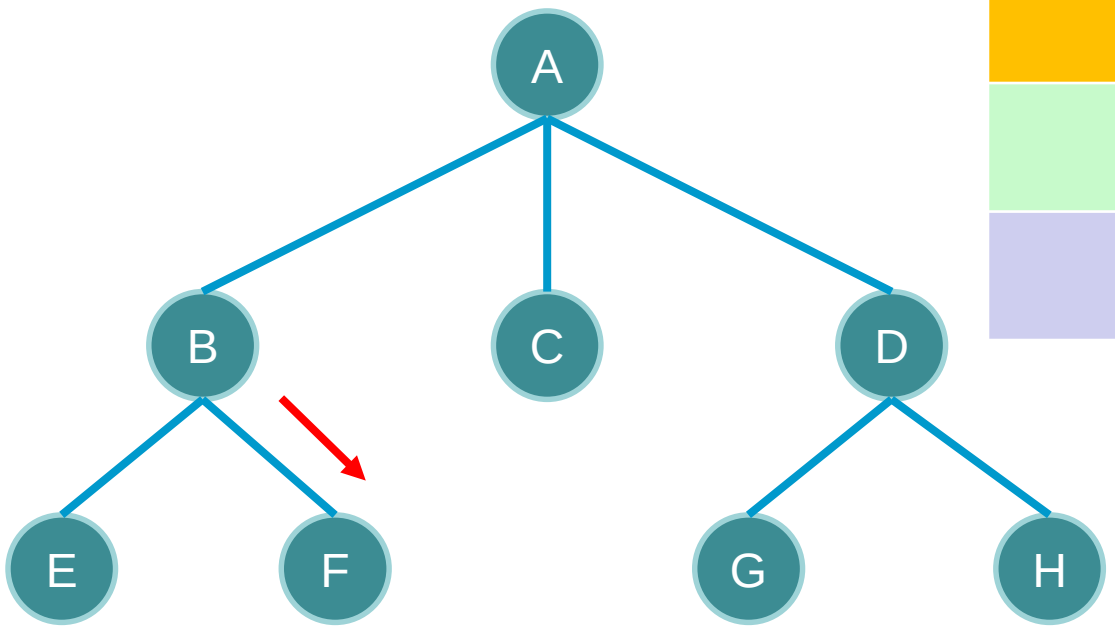
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E

Pre-Order Traversal



Pre-Order

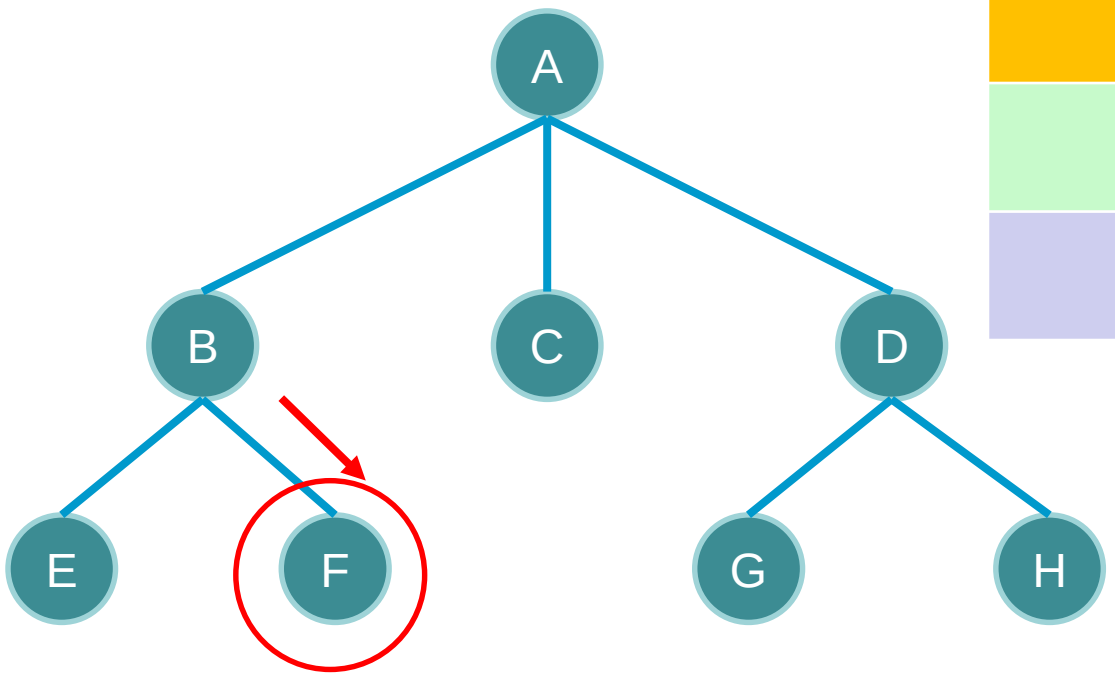
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E

Pre-Order Traversal



Pre-Order

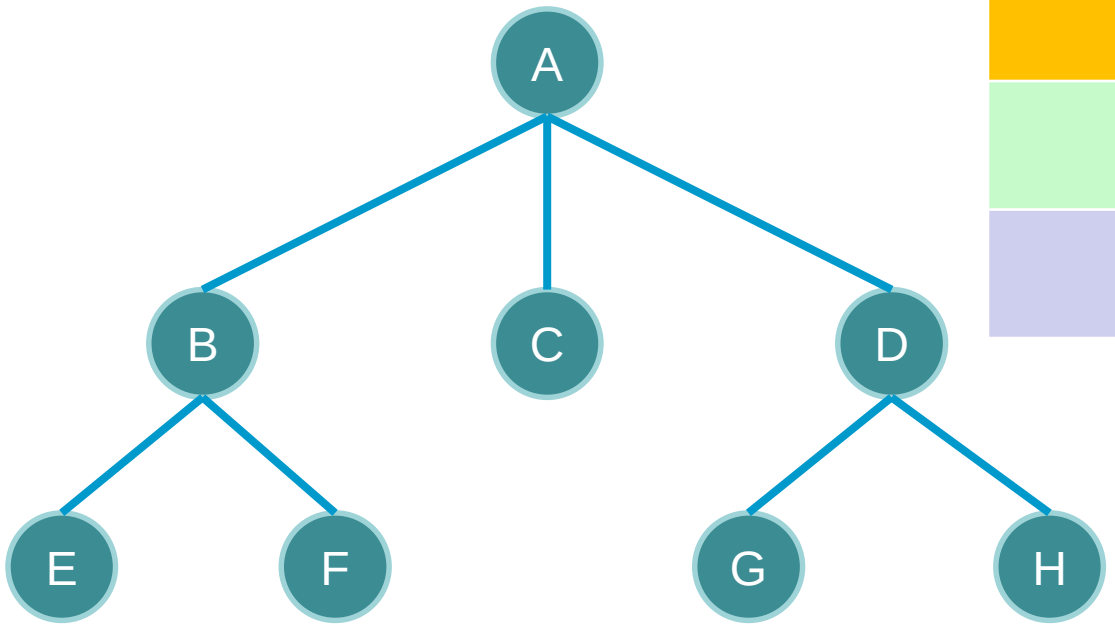
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E

Pre-Order Traversal



Pre-Order

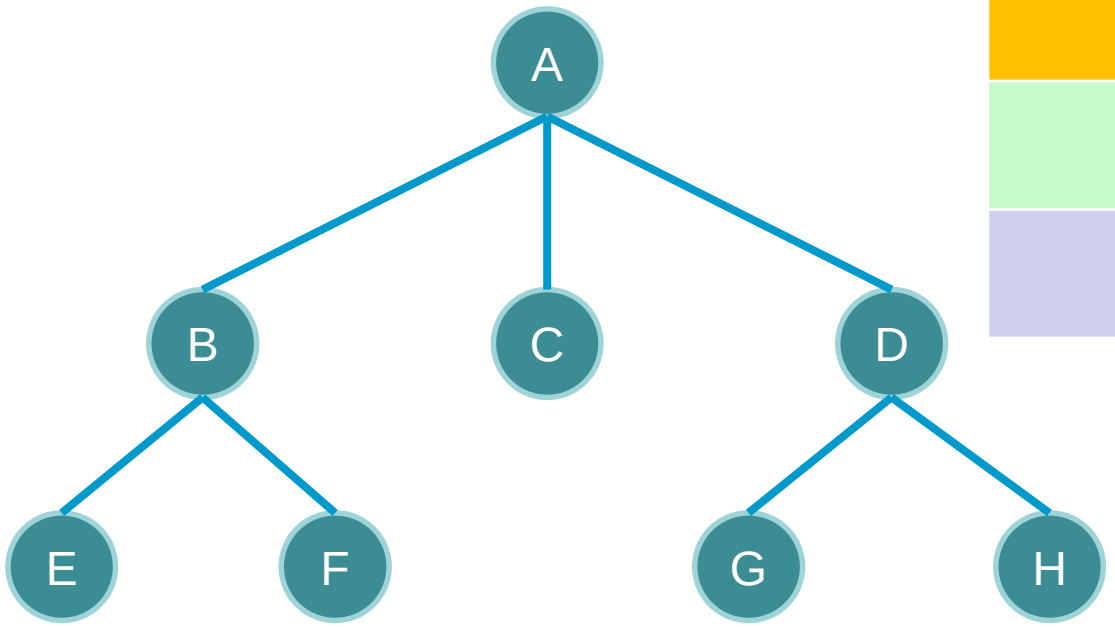
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E F

Pre-Order Traversal



Pre-Order

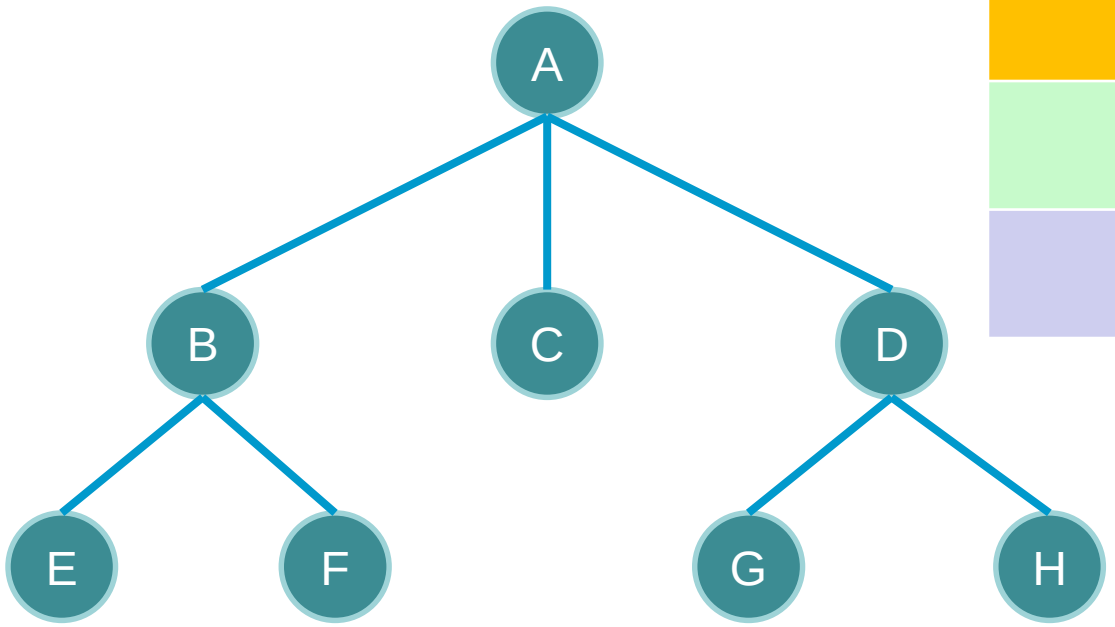
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E F C

Pre-Order Traversal



Pre-Order

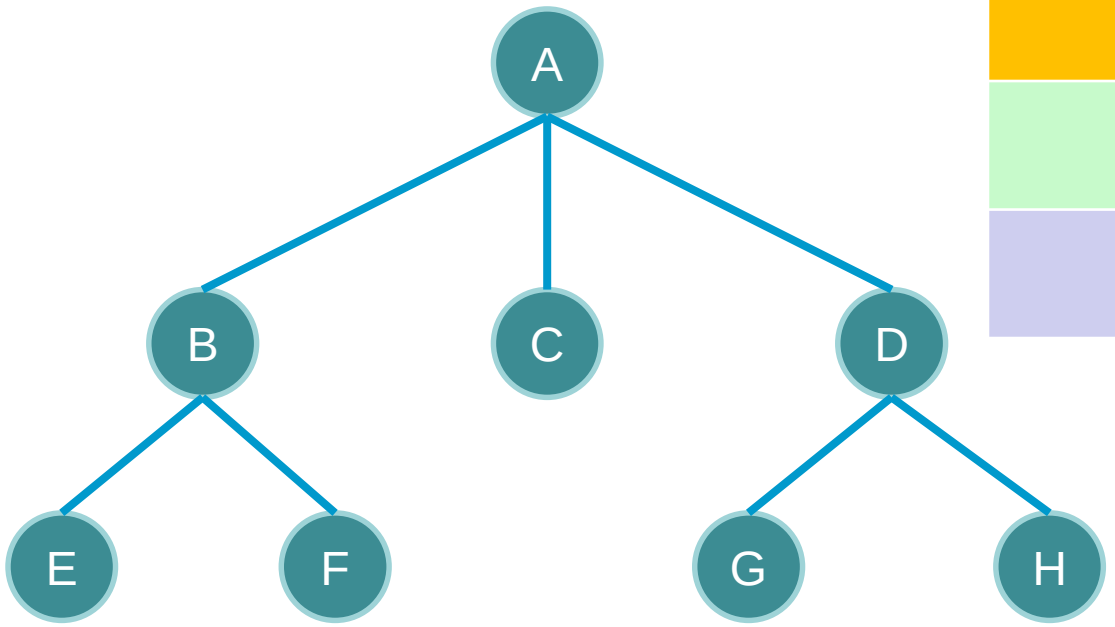
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E F C D

Pre-Order Traversal



Pre-Order

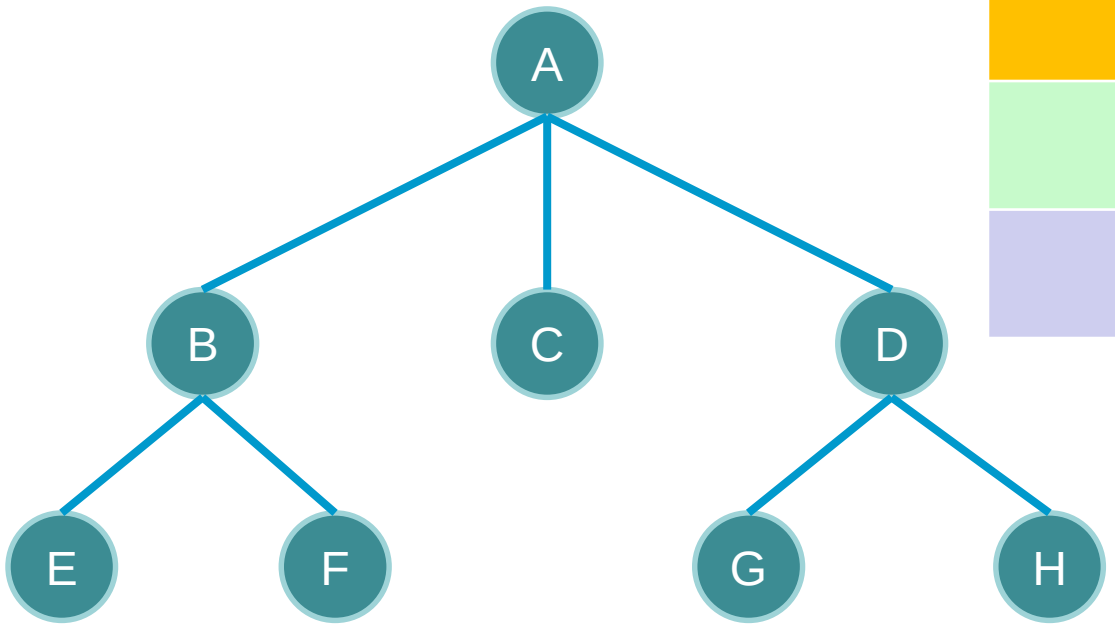
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E F C D G

Pre-Order Traversal



Pre-Order

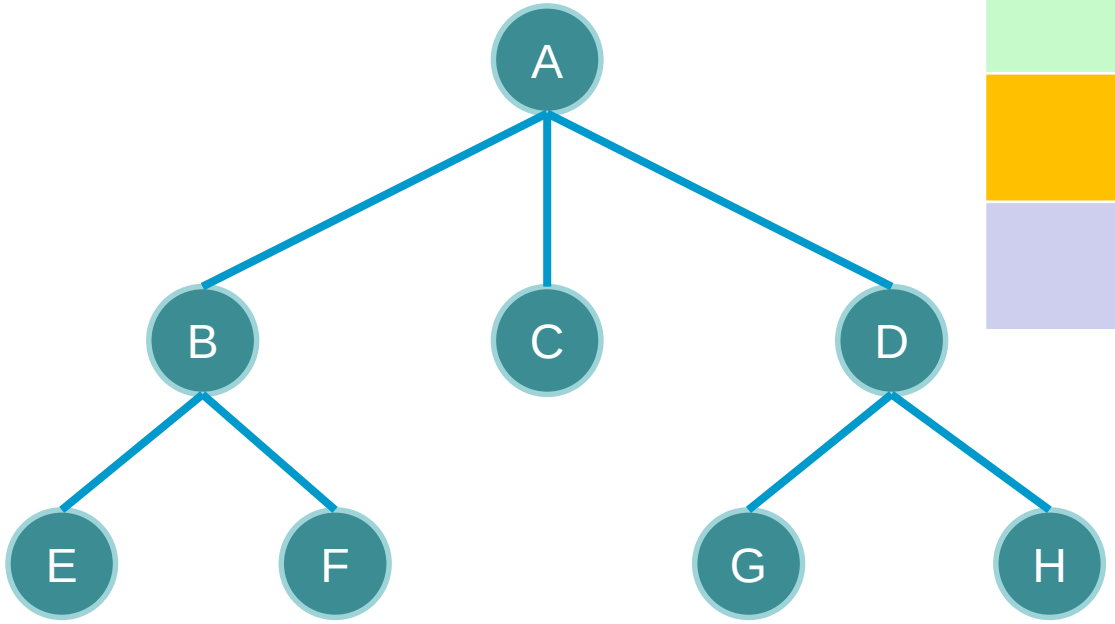
ประมวลผล

ค้นหาทางซ้าย

ค้นหาทางขวา

ผลลัพธ์ : A B E F C D G H

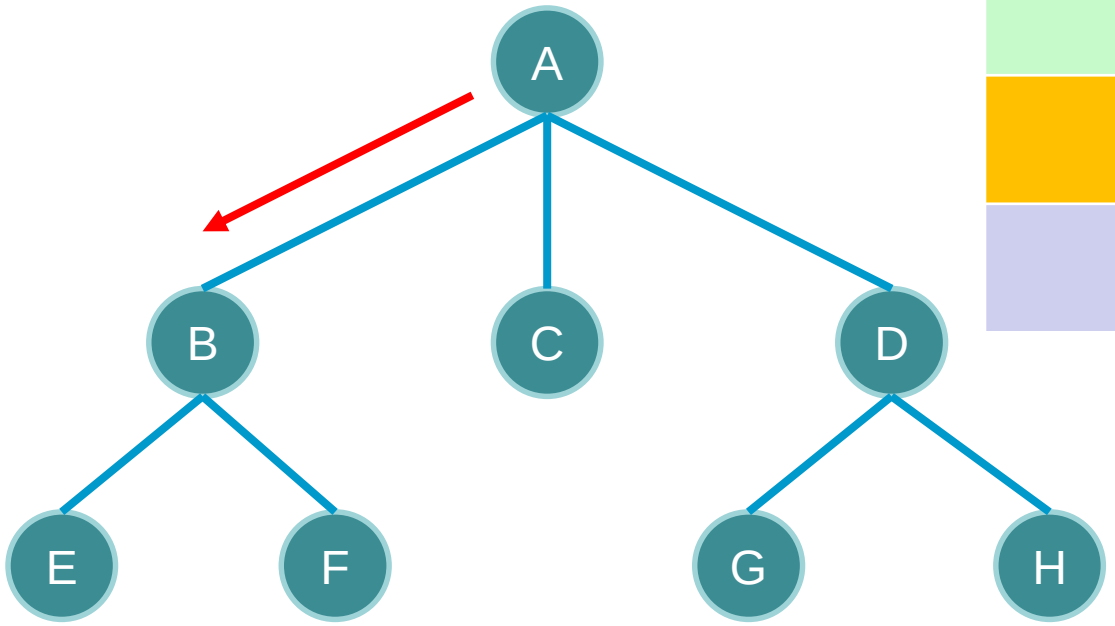
In-Order Traversal



In-Order
ค้นหาทางซ้าย
ประมวลผล
ค้นหาทางขวา

ผลลัพธ์ :

In-Order Traversal



In-Order

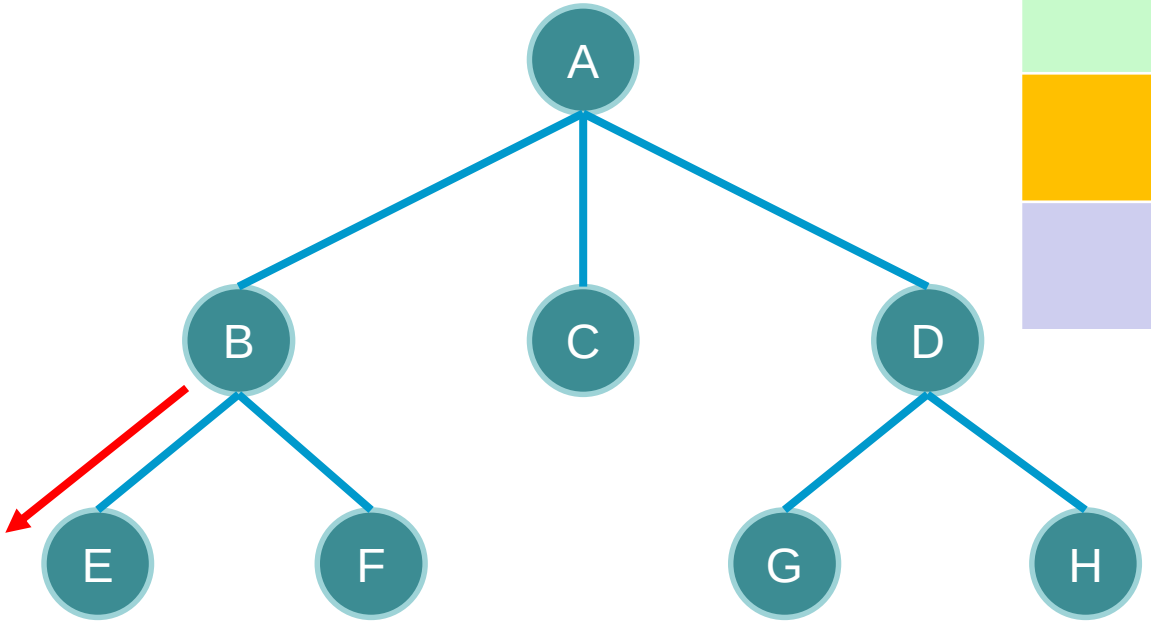
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ :

In-Order Traversal



In-Order

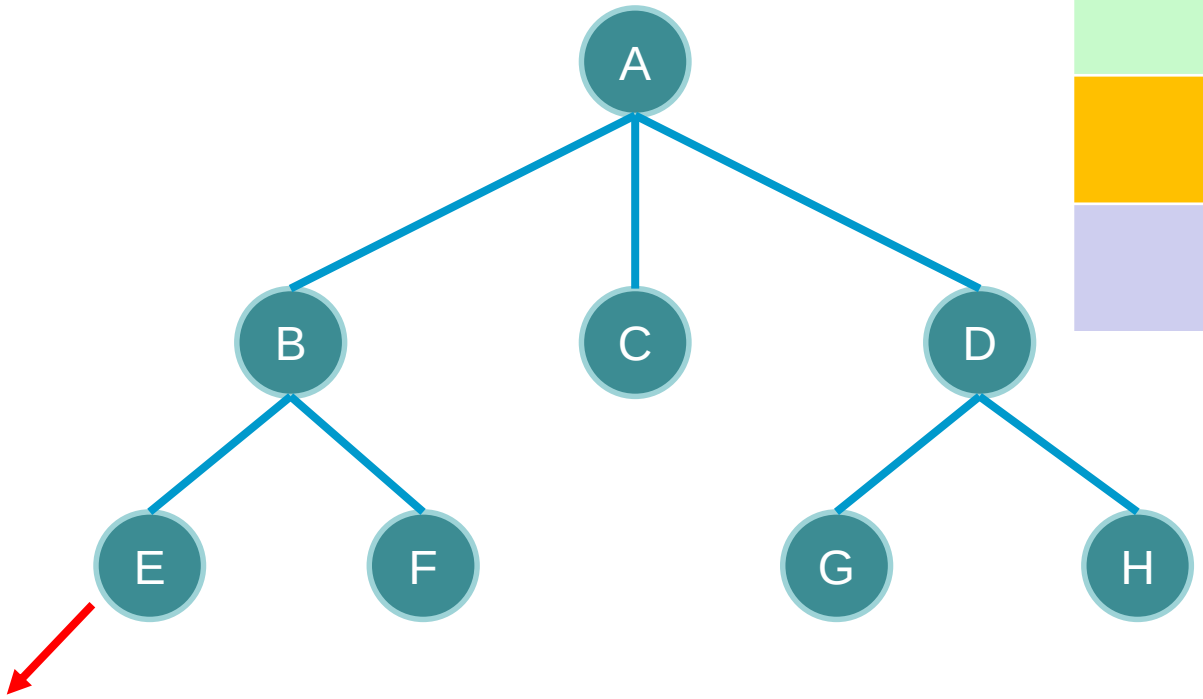
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ :

In-Order Traversal



In-Order

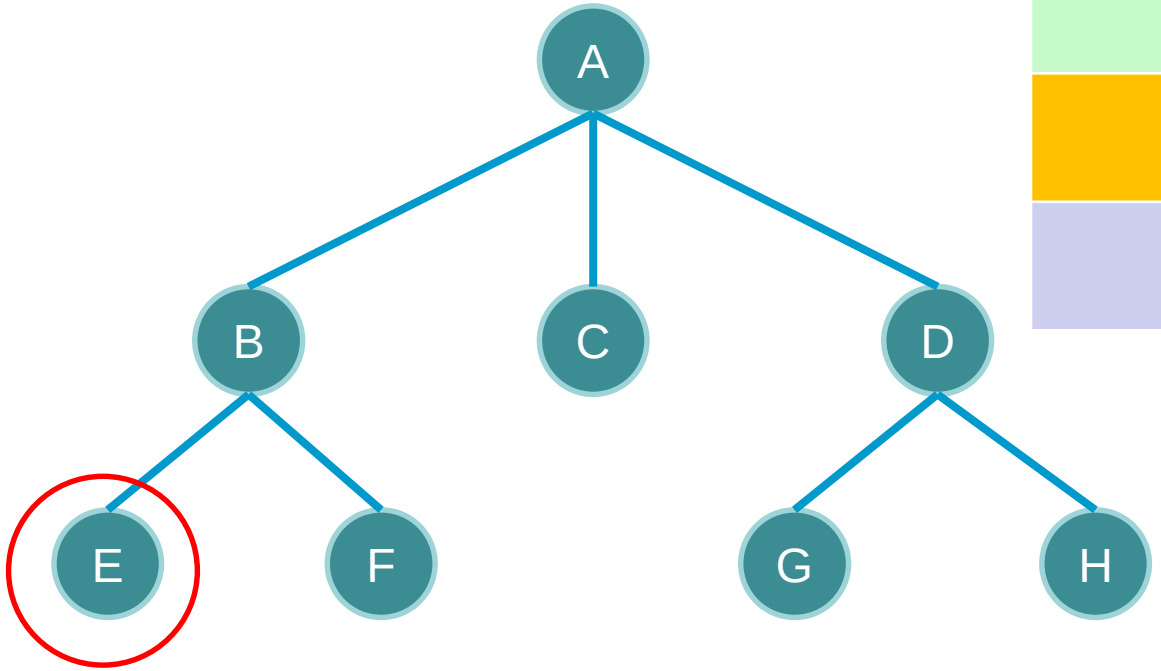
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ :

In-Order Traversal



In-Order

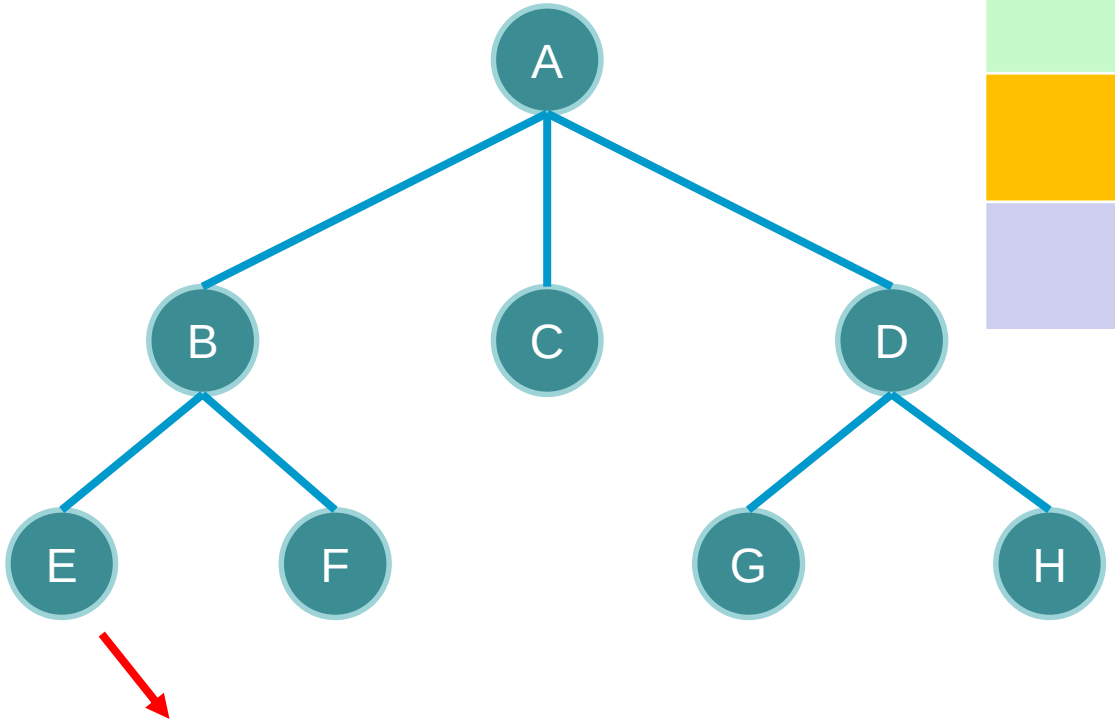
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ :

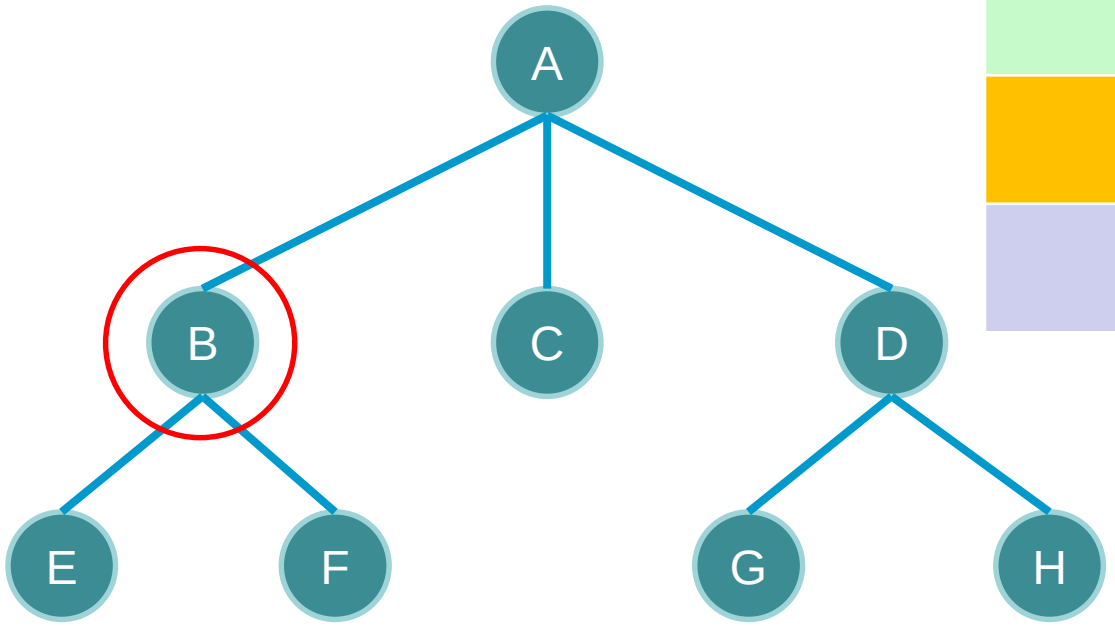
In-Order Traversal



In-Order
ค้นหาทางซ้าย
ประมวลผล
ค้นหาทางขวา

ผลลัพธ์ : E

In-Order Traversal



In-Order

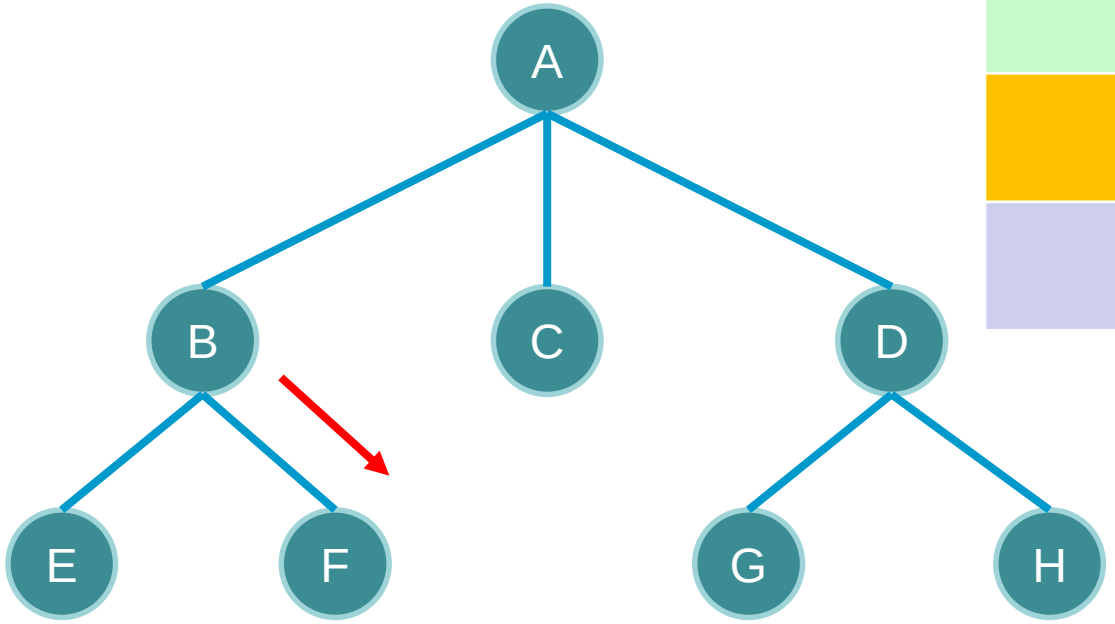
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E

In-Order Traversal



In-Order

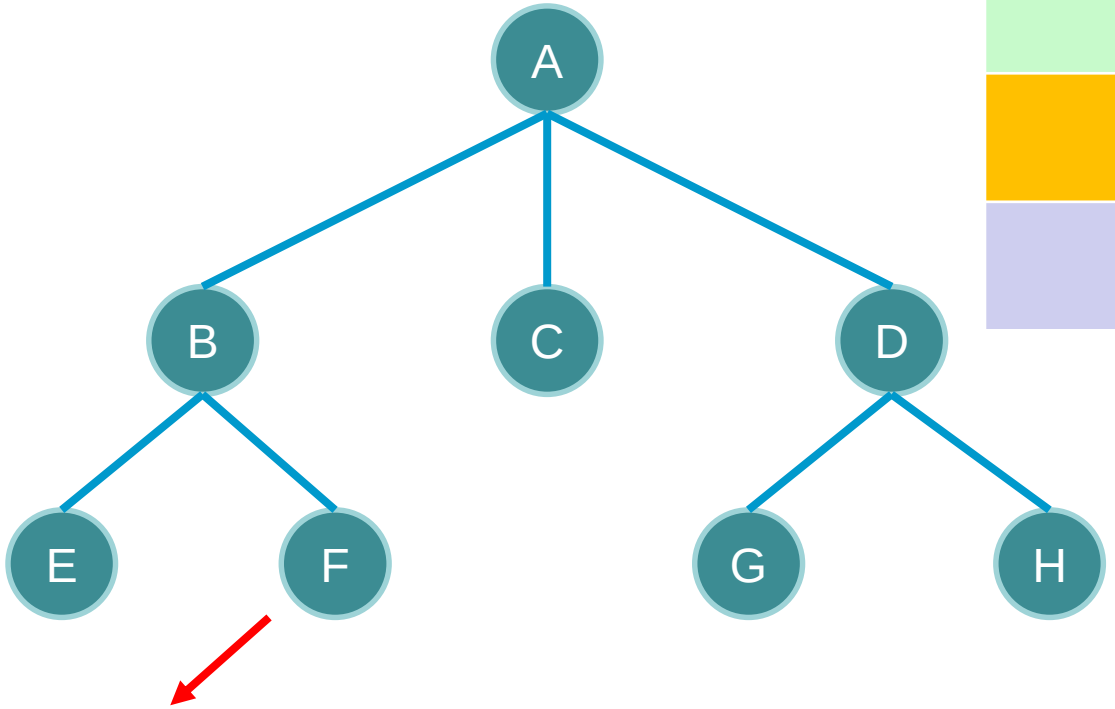
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B

In-Order Traversal



In-Order

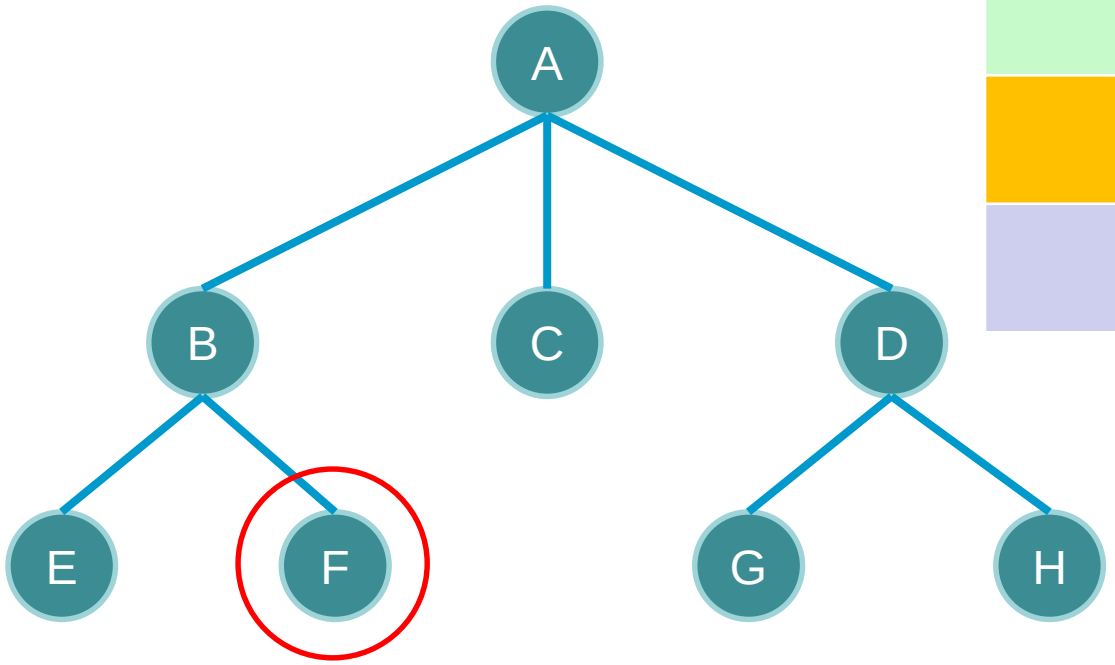
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B

In-Order Traversal



In-Order

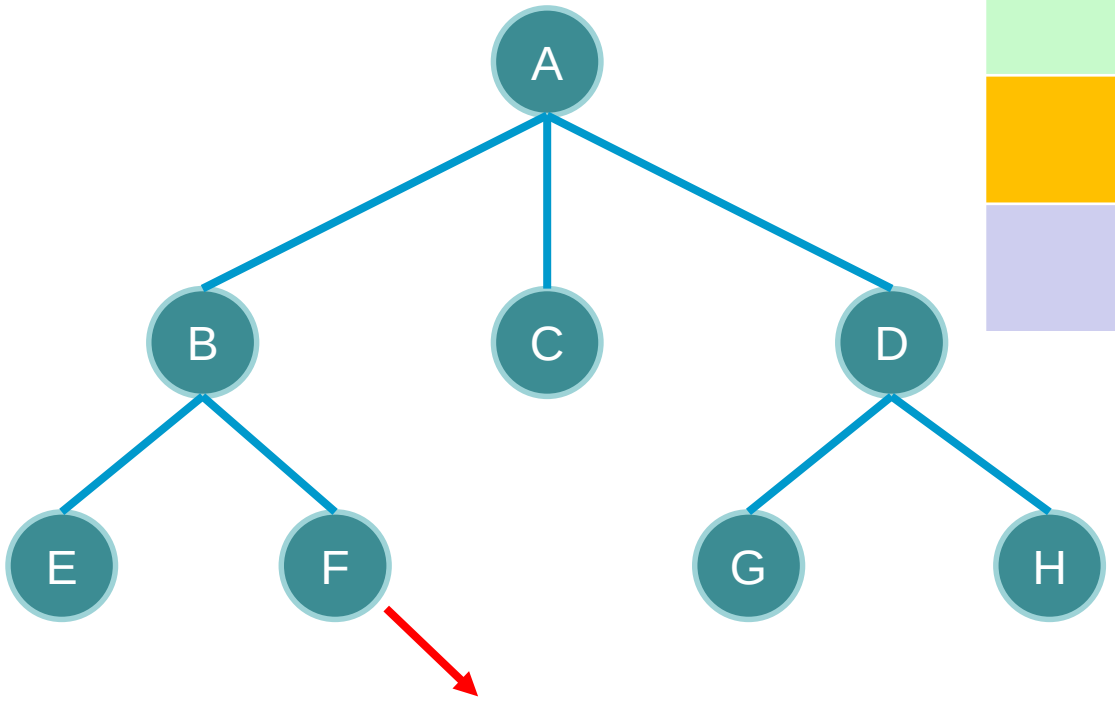
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B

In-Order Traversal



In-Order

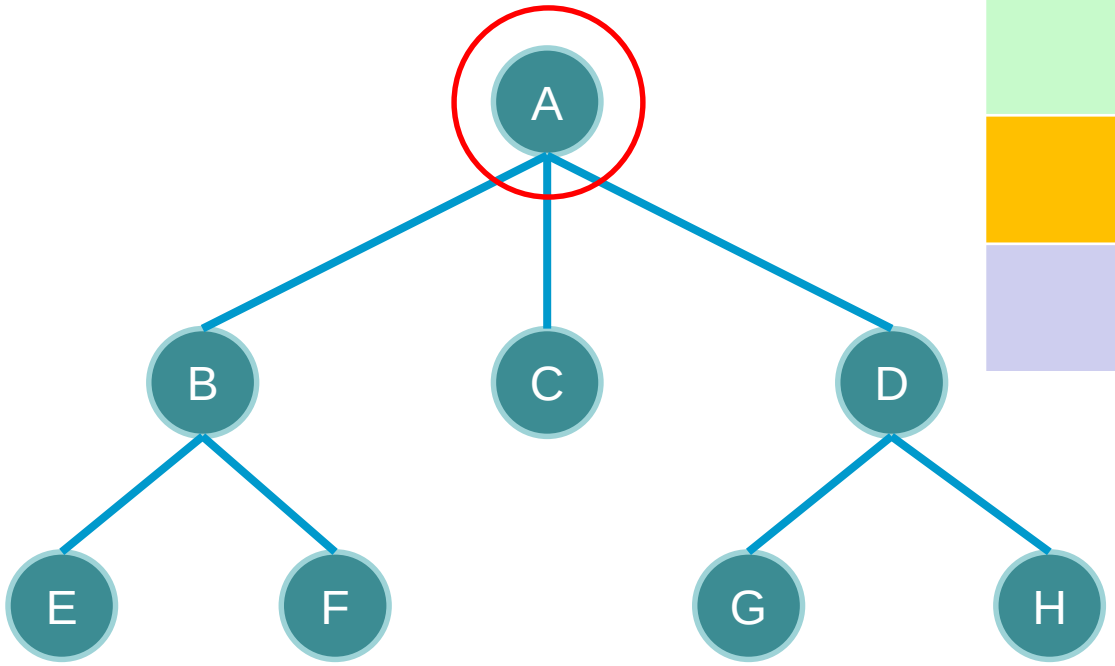
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F

In-Order Traversal



In-Order

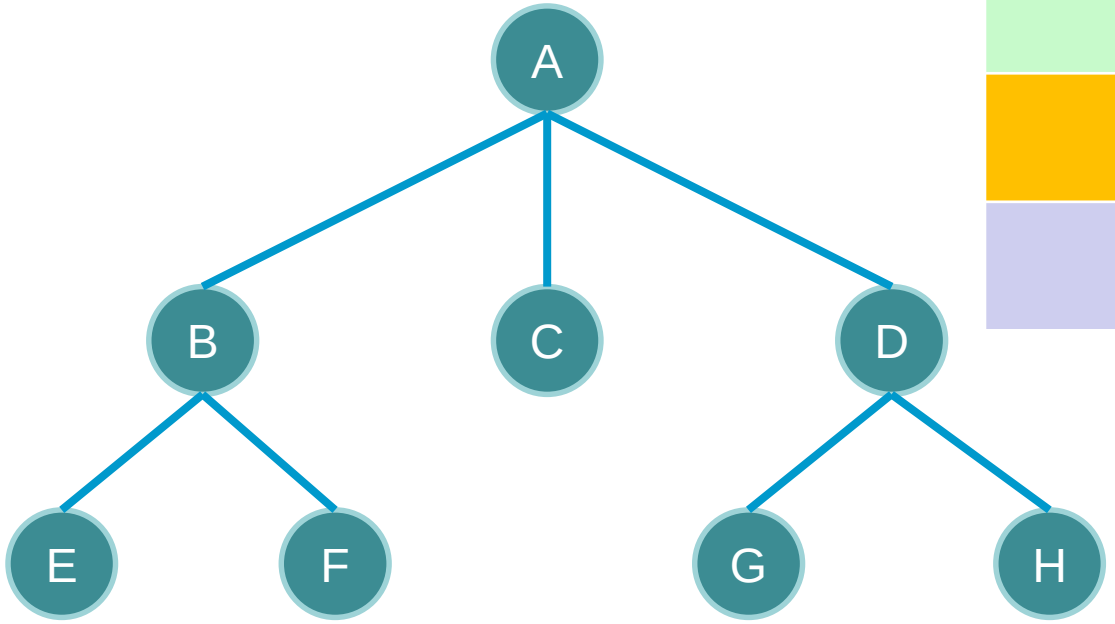
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F A

In-Order Traversal



In-Order

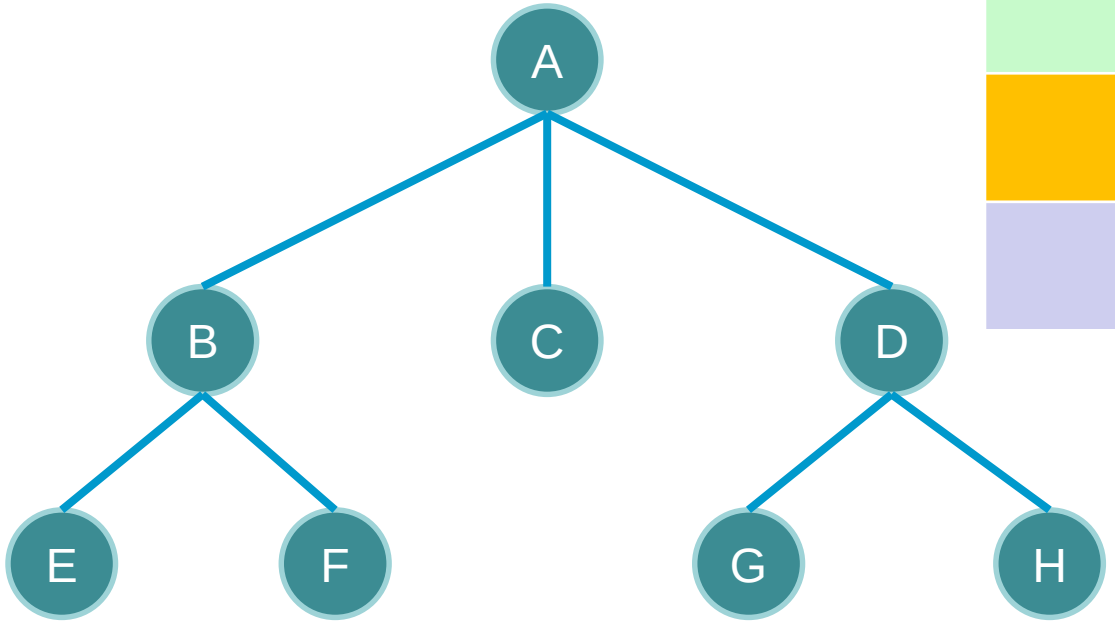
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F A C

In-Order Traversal



In-Order

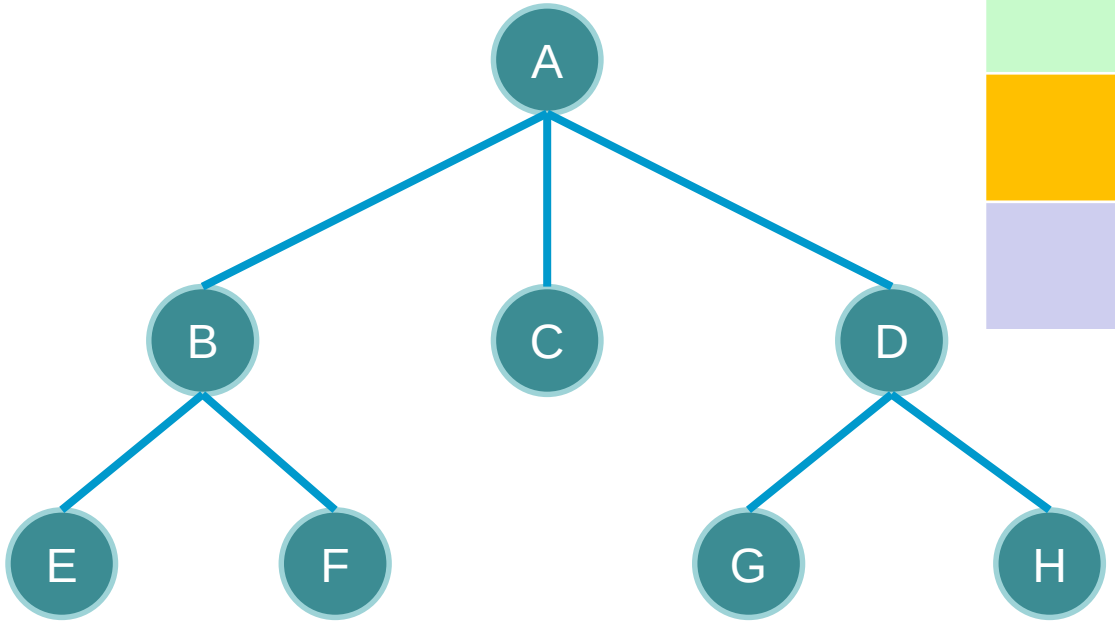
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F A C G

In-Order Traversal



In-Order

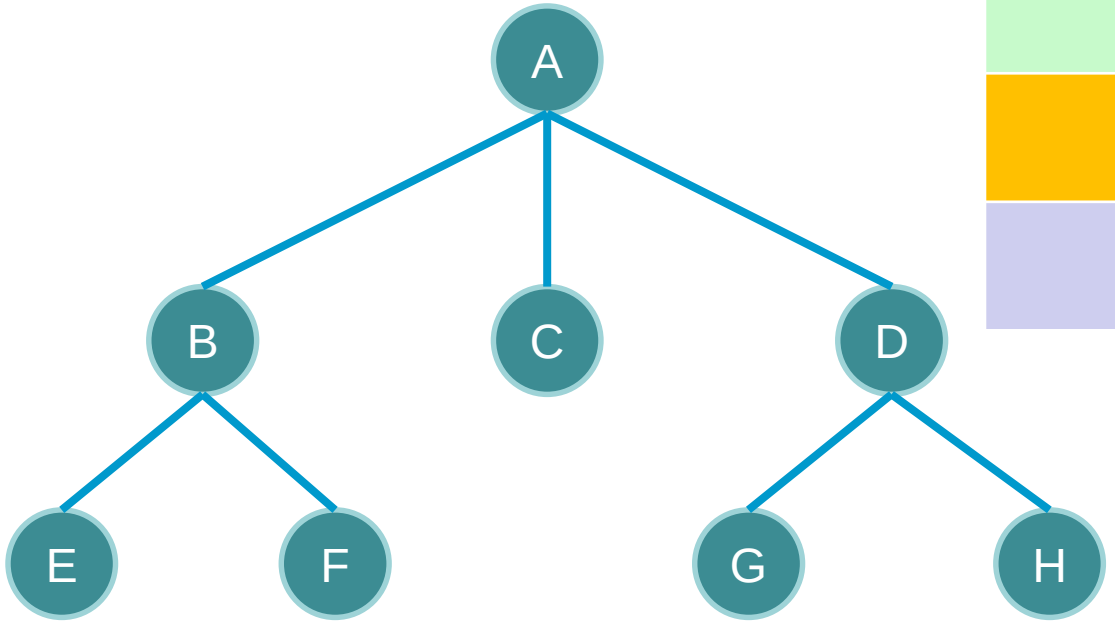
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F A C G D

In-Order Traversal



In-Order

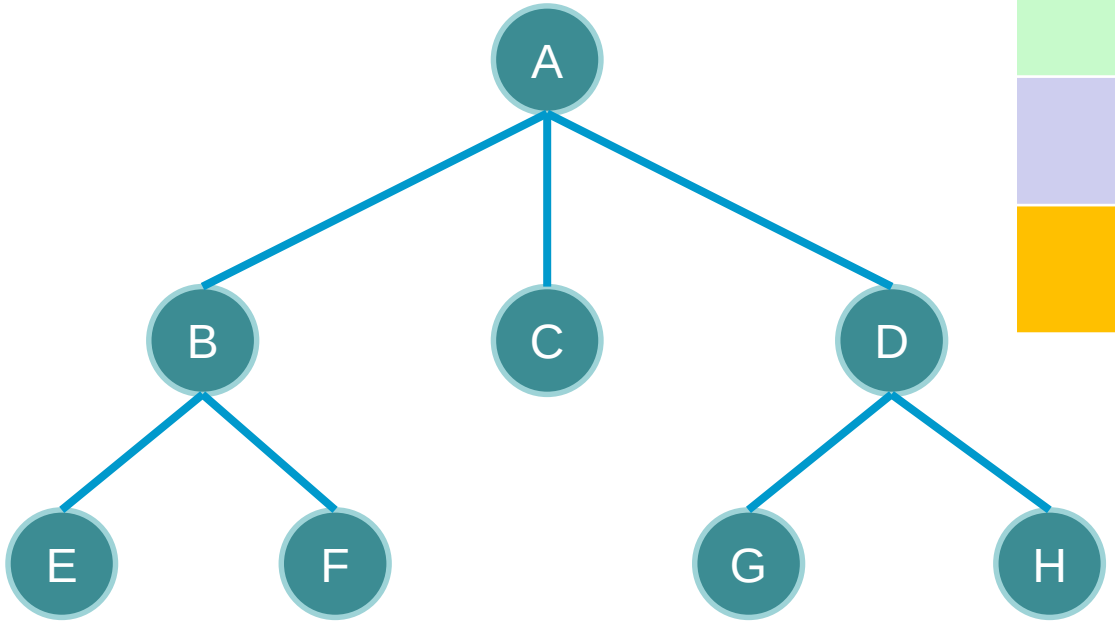
ค้นหาทางซ้าย

ประมวลผล

ค้นหาทางขวา

ผลลัพธ์ : E B F A C G D H

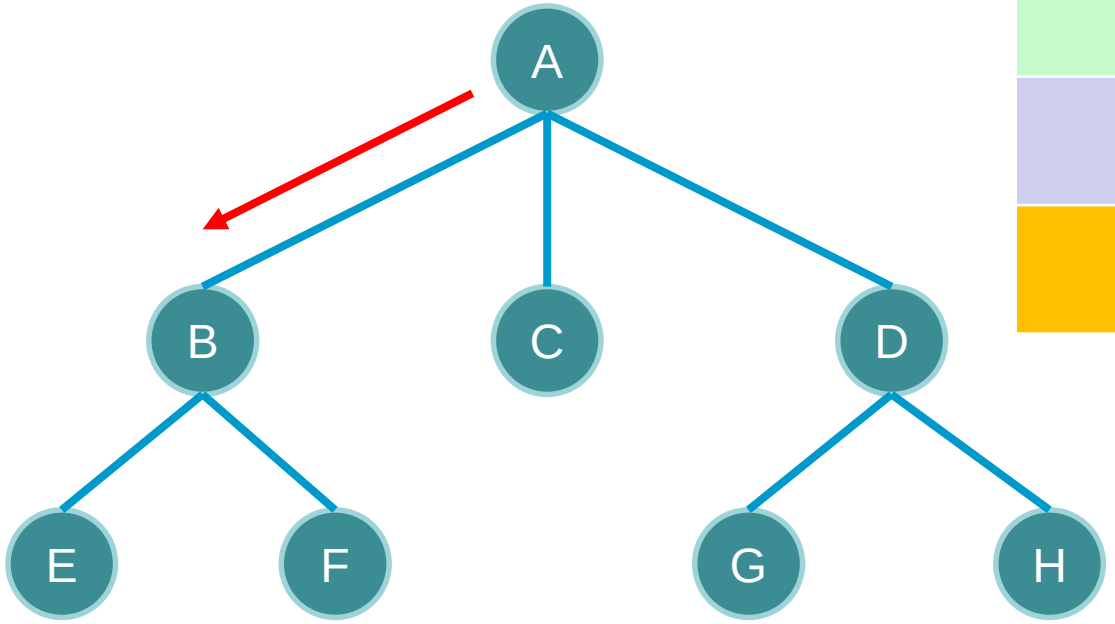
Post-Order Traversal



Post-Order
ค้นหาทางซ้าย
ค้นหาทางขวา
ประมวลผล

ผลลัพธ์ :

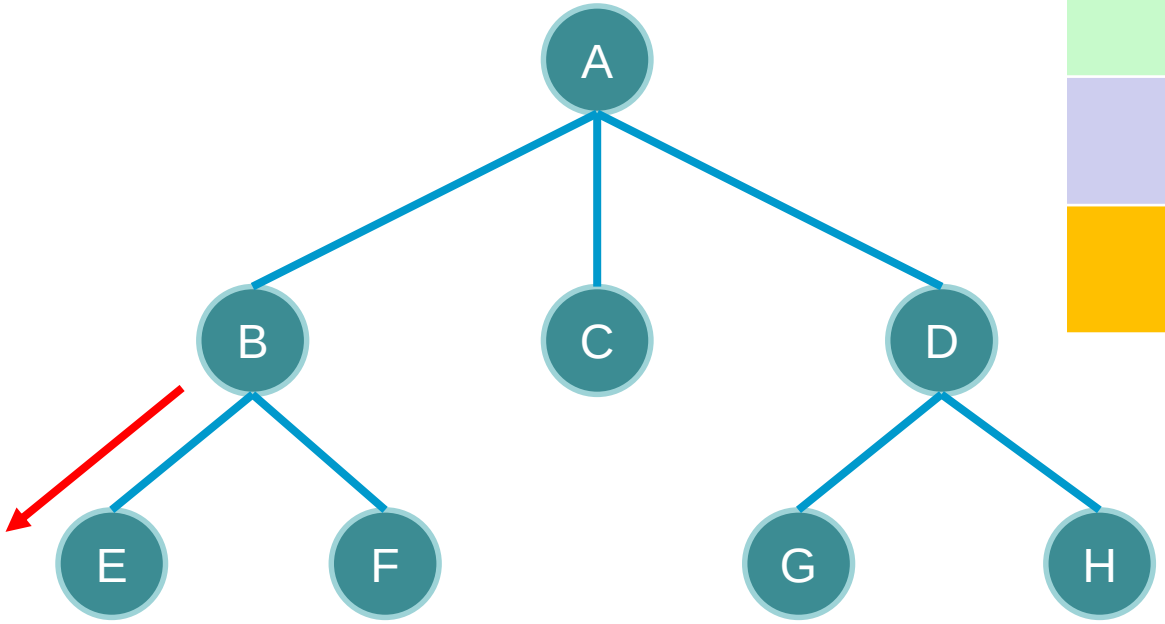
Post-Order Traversal



Post-Order
ค้นหาทางซ้าย
ค้นหาทางขวา
ประมวลผล

ผลลัพธ์ :

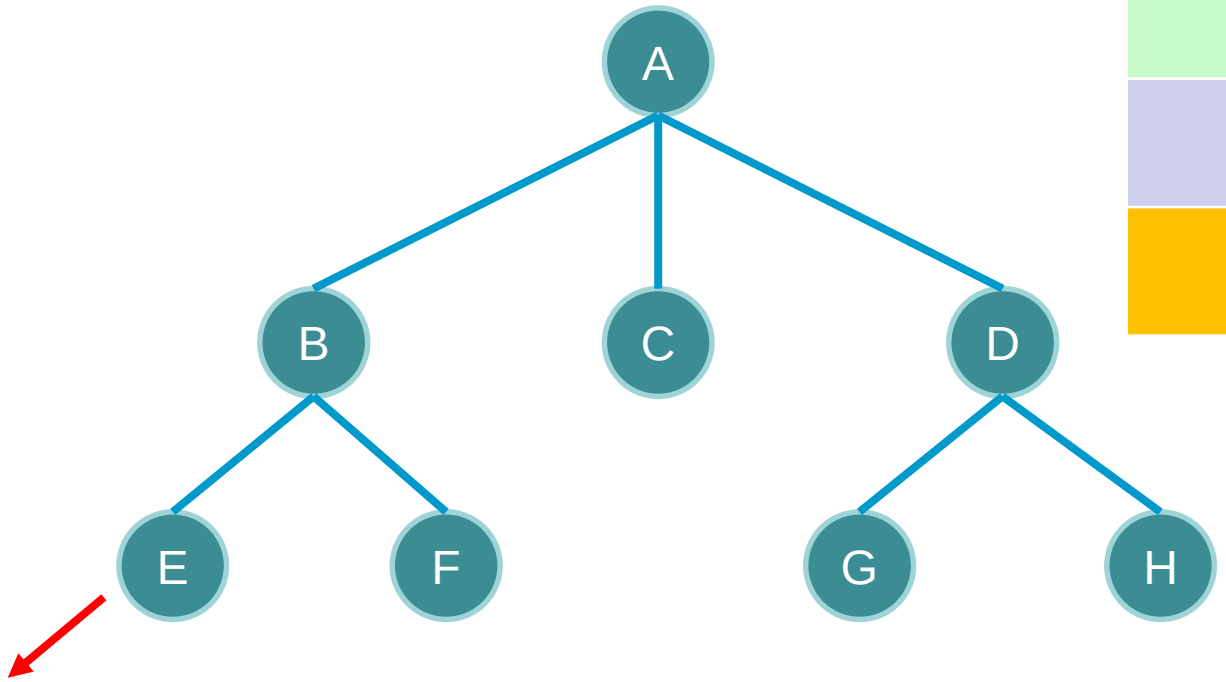
Post-Order Traversal



Post-Order
ค้นหาทางซ้าย
ค้นหาทางขวา
ประมวลผล

ผลลัพธ์ :

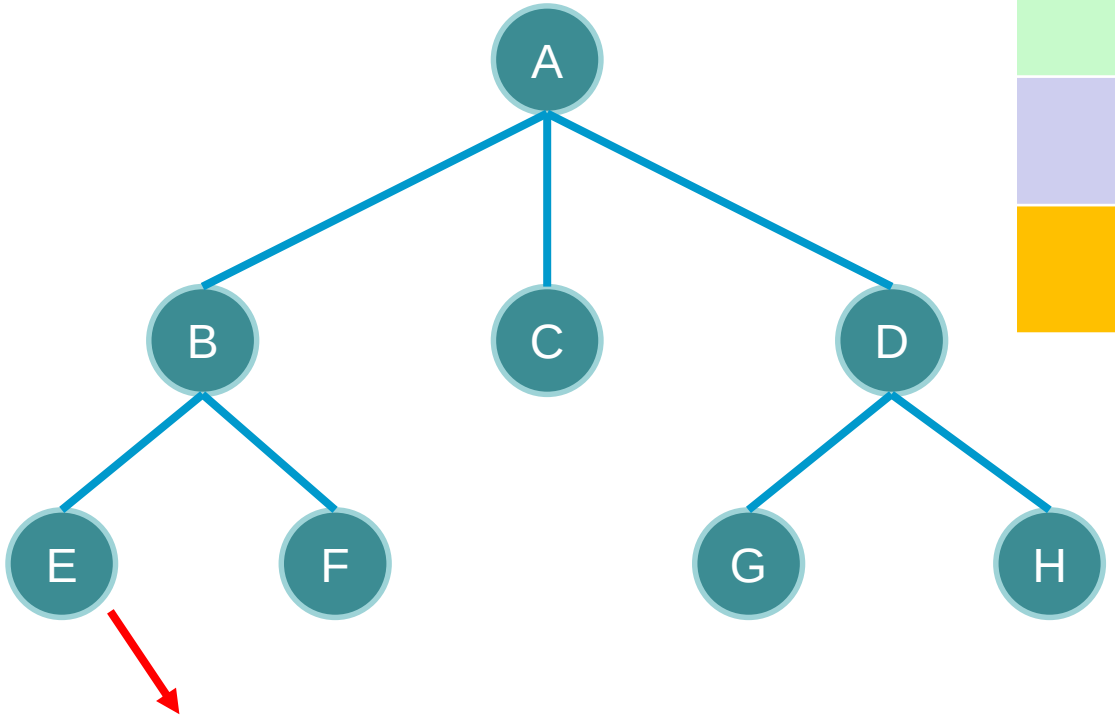
Post-Order Traversal



Post-Order
ค้นหาทางซ้าย
ค้นหาทางขวา
ประมวลผล

ผลลัพธ์ :

Post-Order Traversal



Post-Order

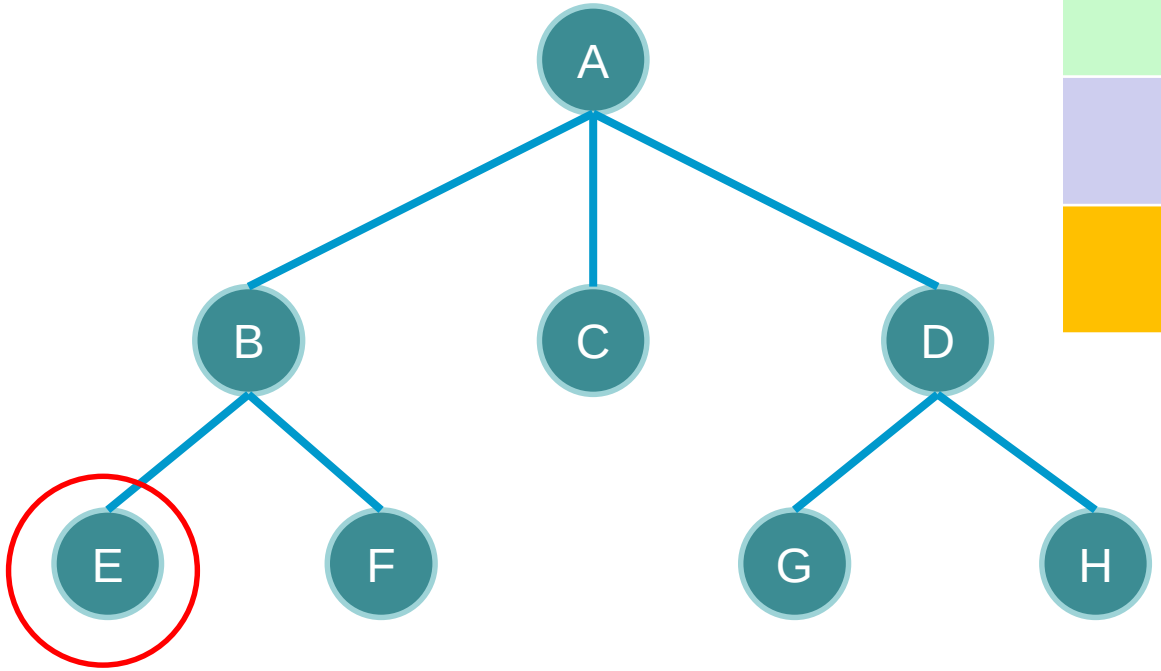
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ :

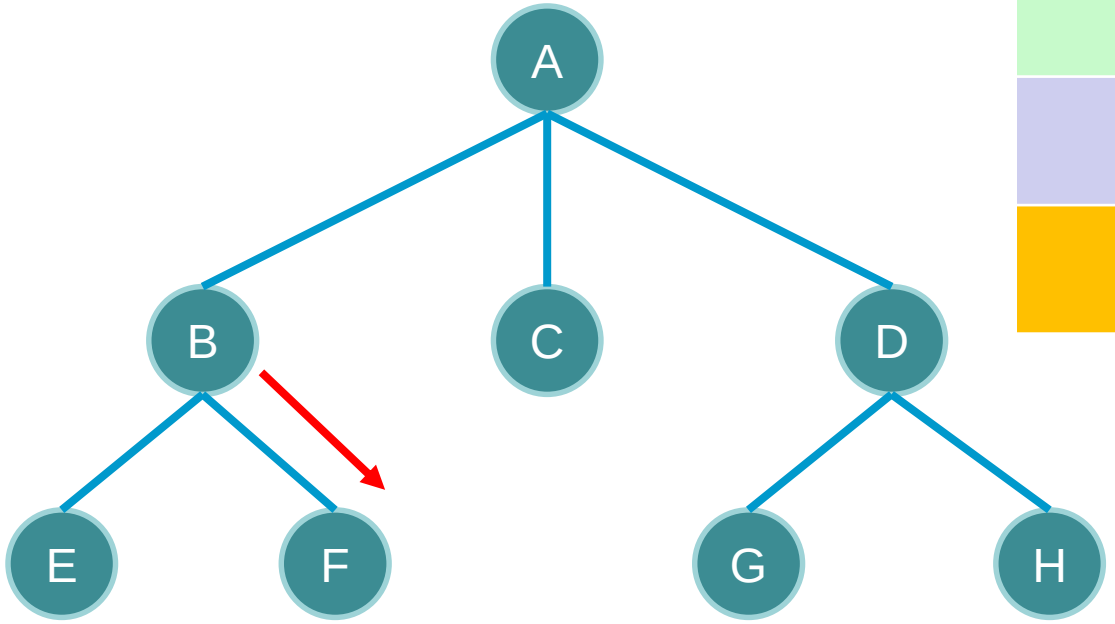
Post-Order Traversal



Post-Order
ค้นหาทางซ้าย
ค้นหาทางขวา
ประมวลผล

ผลลัพธ์ :

Post-Order Traversal



Post-Order

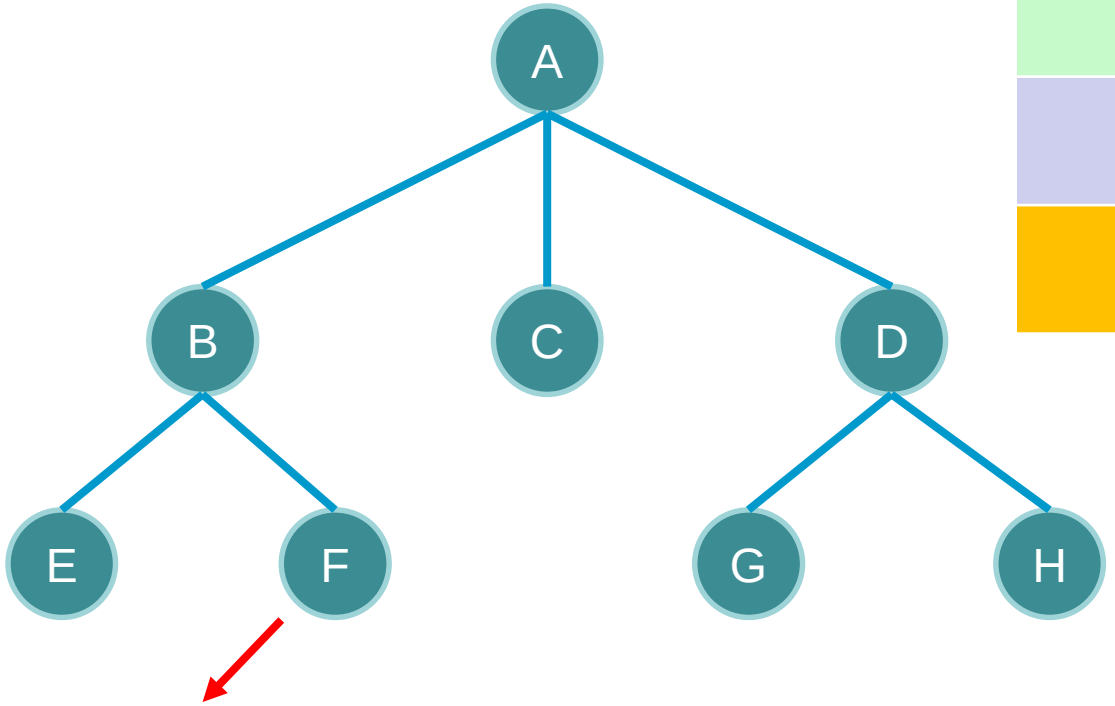
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E

Post-Order Traversal



Post-Order

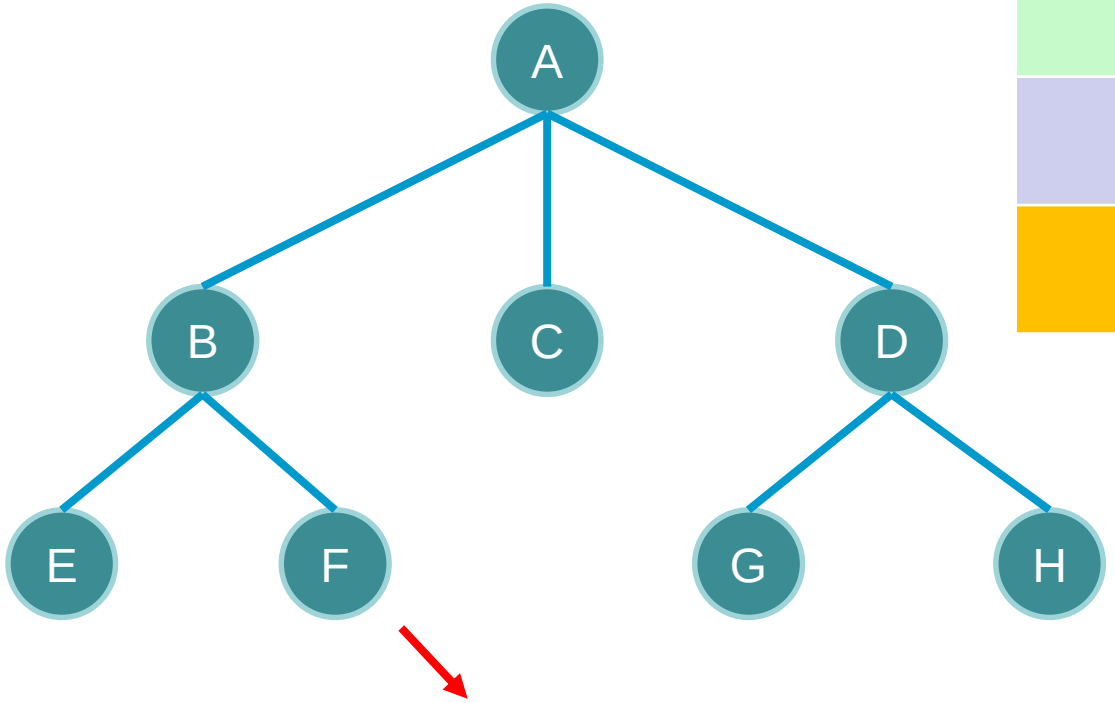
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E

Post-Order Traversal



Post-Order

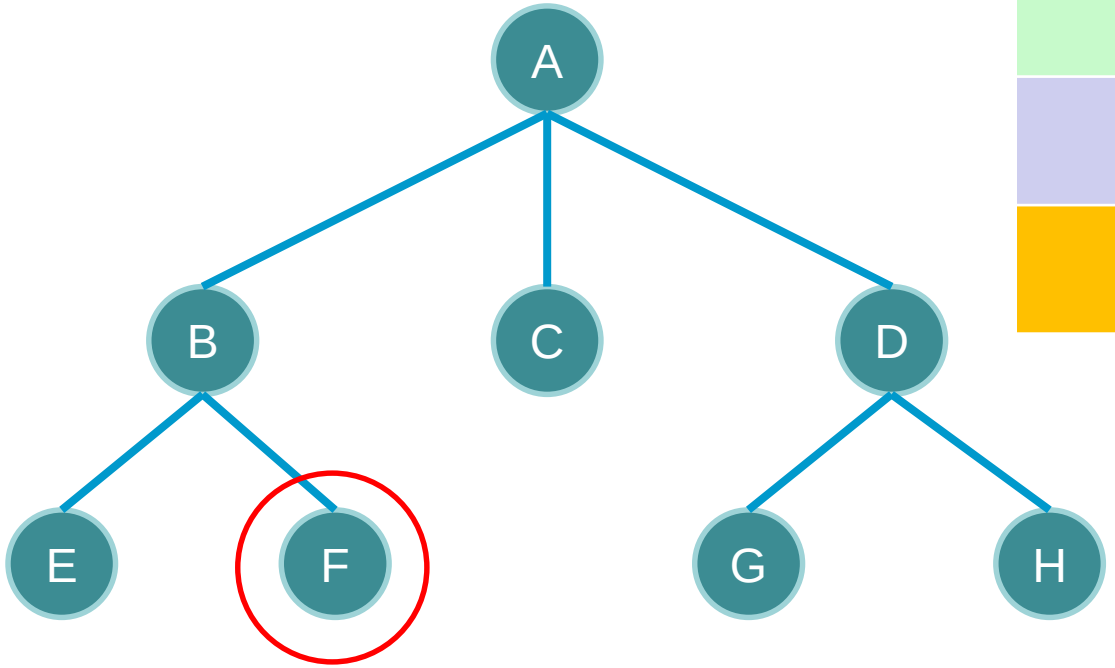
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E

Post-Order Traversal



Post-Order

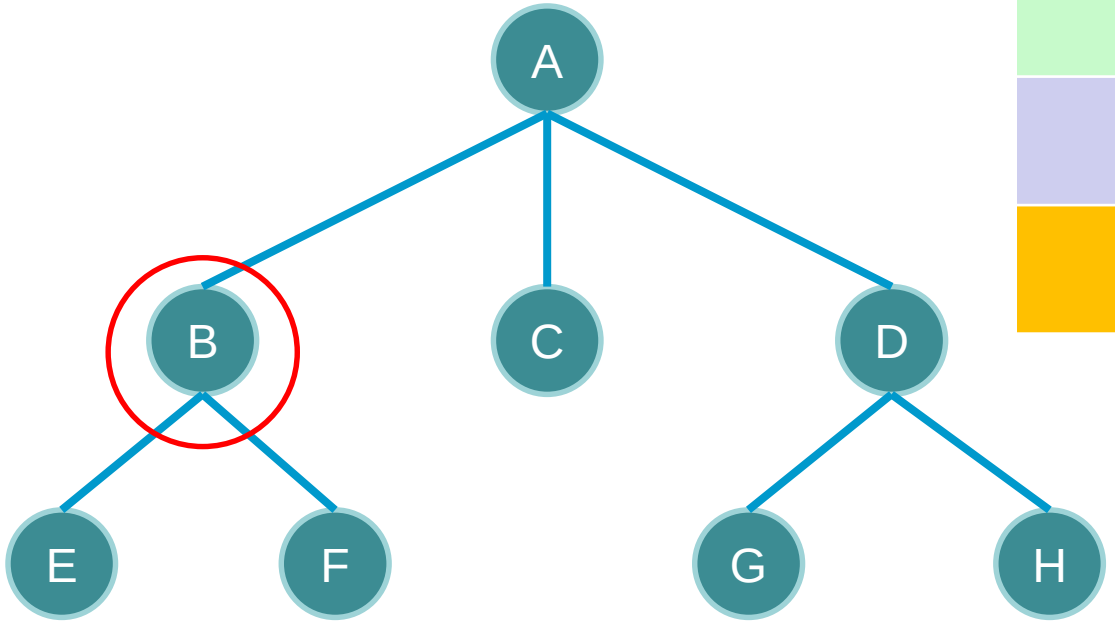
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E

Post-Order Traversal



Post-Order

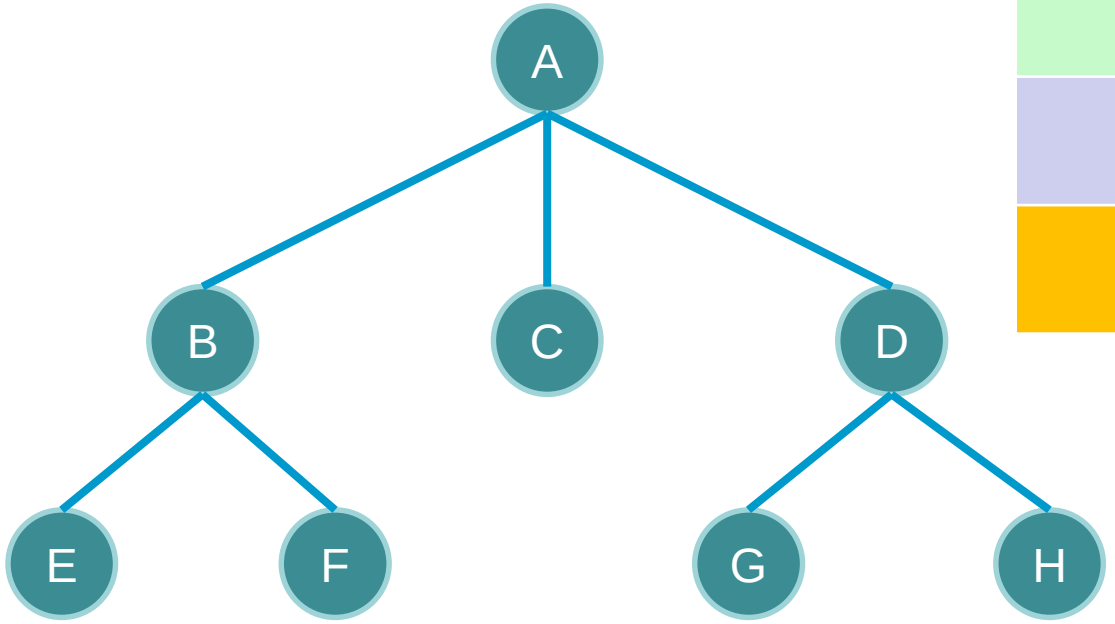
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F

Post-Order Traversal



Post-Order

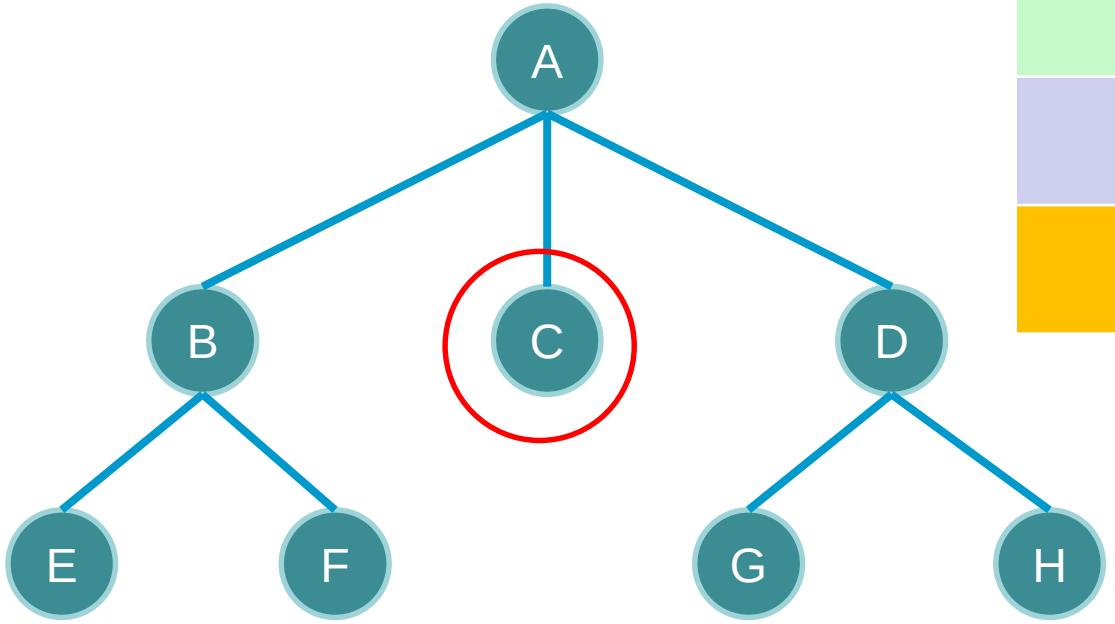
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B

Post-Order Traversal



Post-Order

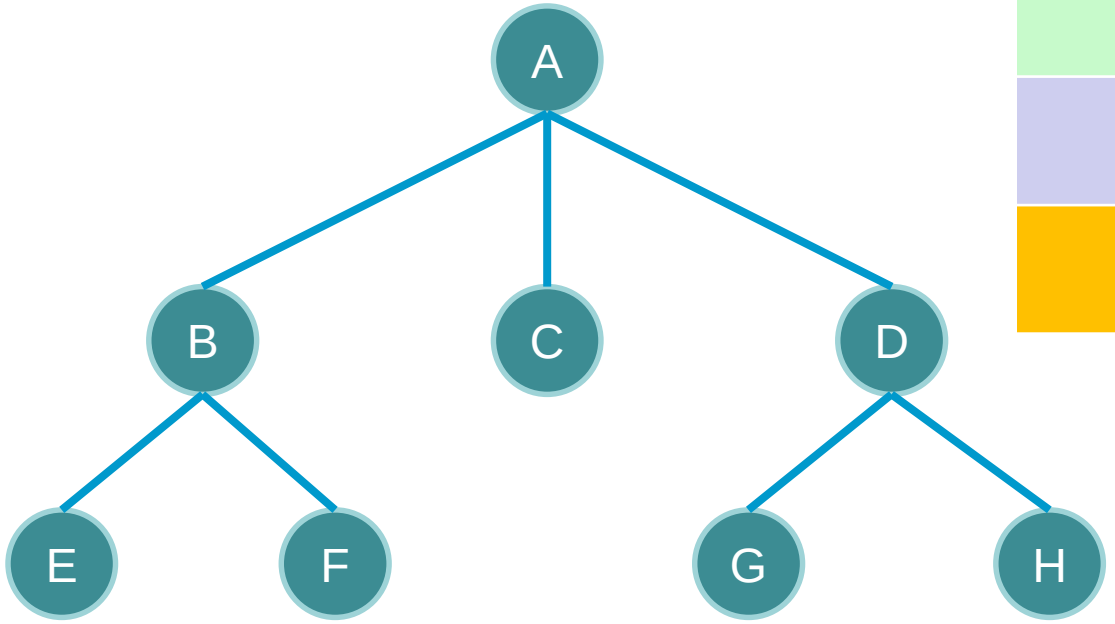
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B C

Post-Order Traversal



Post-Order

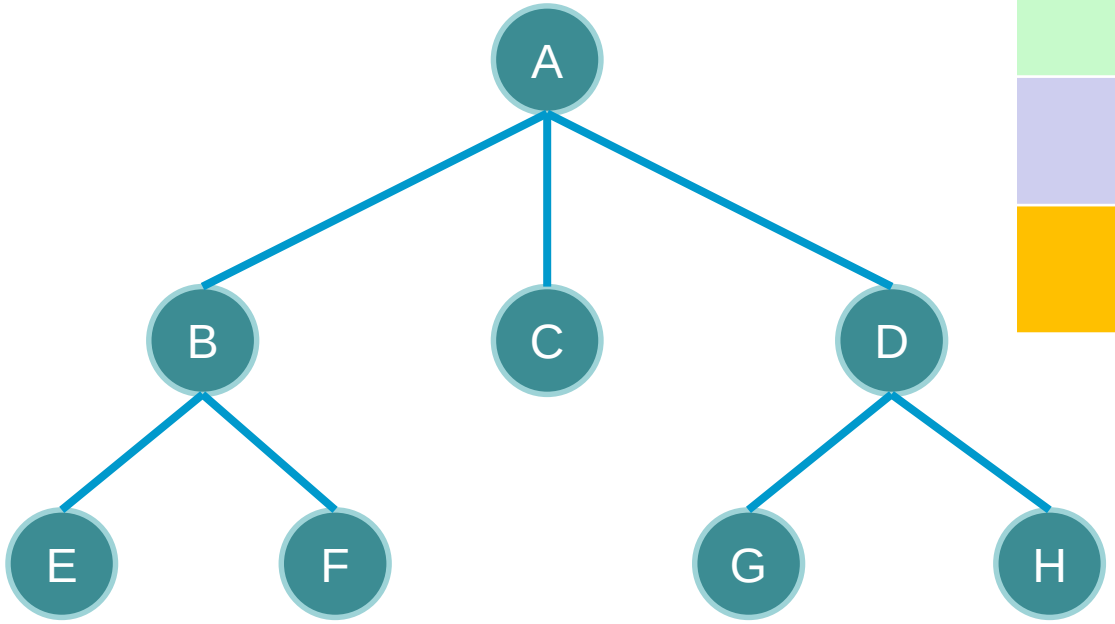
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B C G

Post-Order Traversal



Post-Order

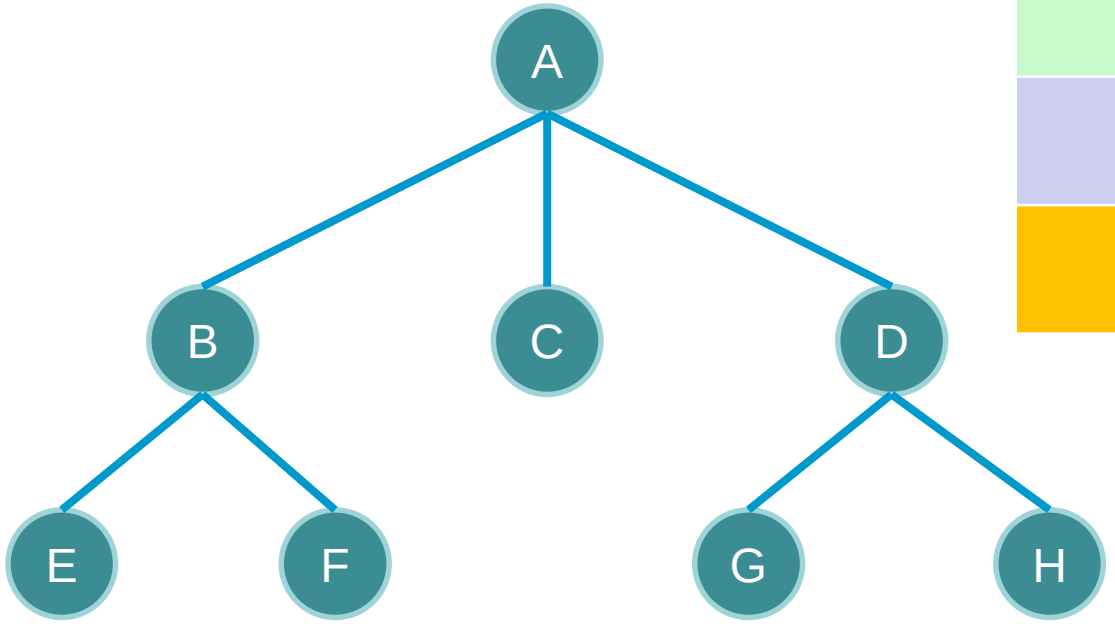
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B C G H

Post-Order Traversal



Post-Order

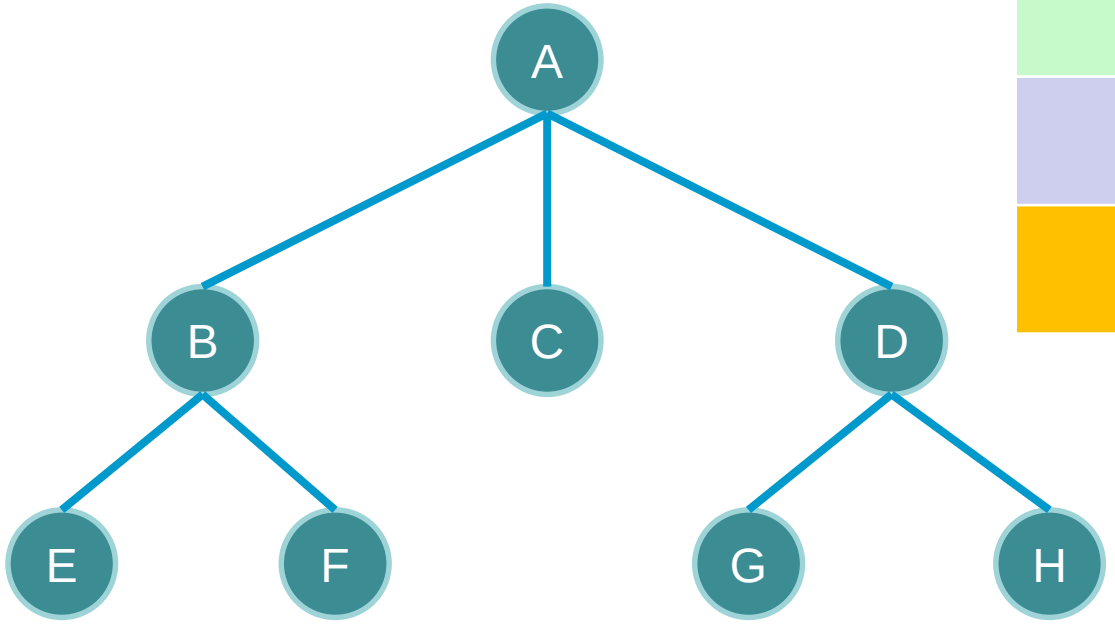
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B C G H D

Post-Order Traversal



Post-Order

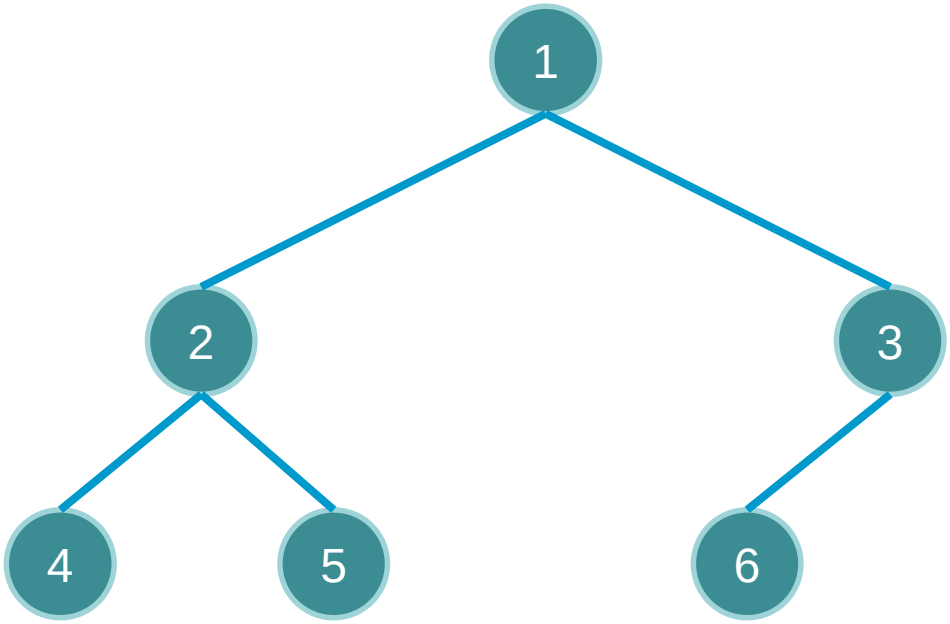
ค้นหาทางซ้าย

ค้นหาทางขวา

ประมวลผล

ผลลัพธ์ : E F B C G H D A

Example

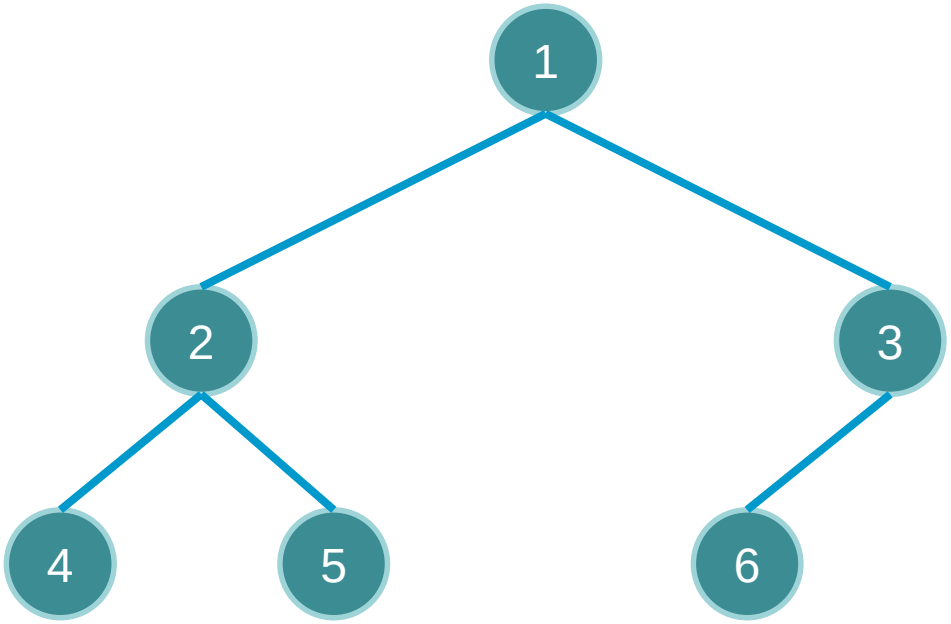


ผลลัพธ์แบบ Pre-Order :

ผลลัพธ์แบบ In-Order :

ผลลัพธ์แบบ Post-Order :

Example

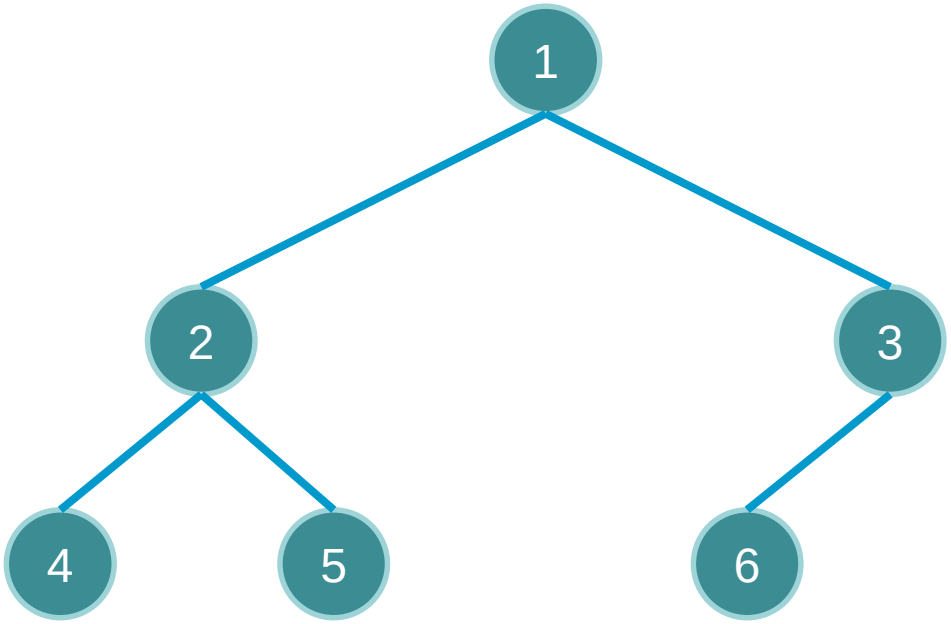


ผลลัพธ์แบบ Pre-Order : 1 2 4 5 3 6

ผลลัพธ์แบบ In-Order :

ผลลัพธ์แบบ Post-Order :

Example

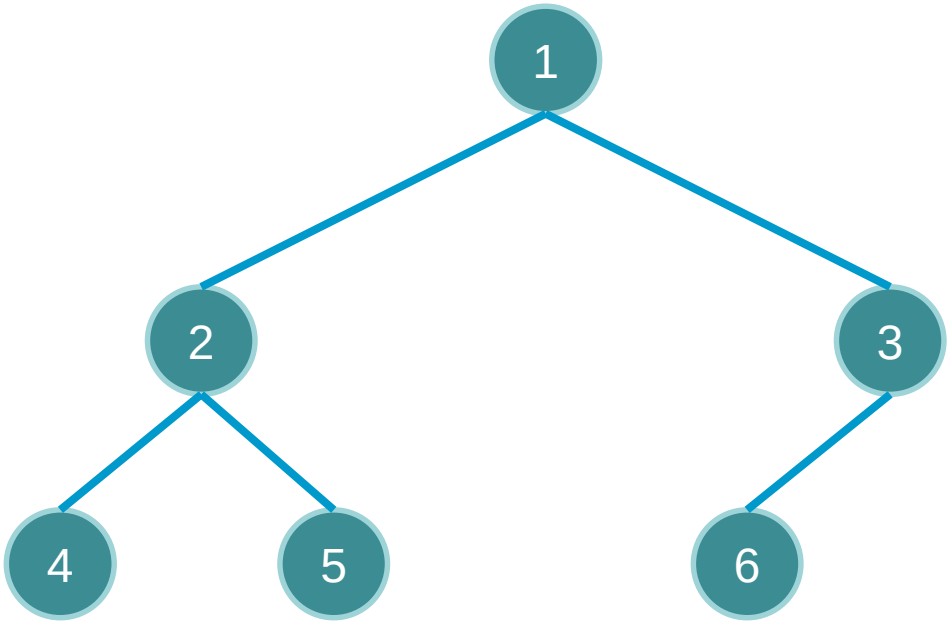


ผลลัพธ์แบบ Pre-Order : 1 2 4 5 3 6

ผลลัพธ์แบบ In-Order : 4 2 5 1 6 3

ผลลัพธ์แบบ Post-Order :

Example



ผลลัพธ์แบบ Pre-Order : 1 2 4 5 3 6

ผลลัพธ์แบบ In-Order : 4 2 5 1 6 3

ผลลัพธ์แบบ Post-Order : 4 5 2 6 3 1

บทที่ 11 Tree

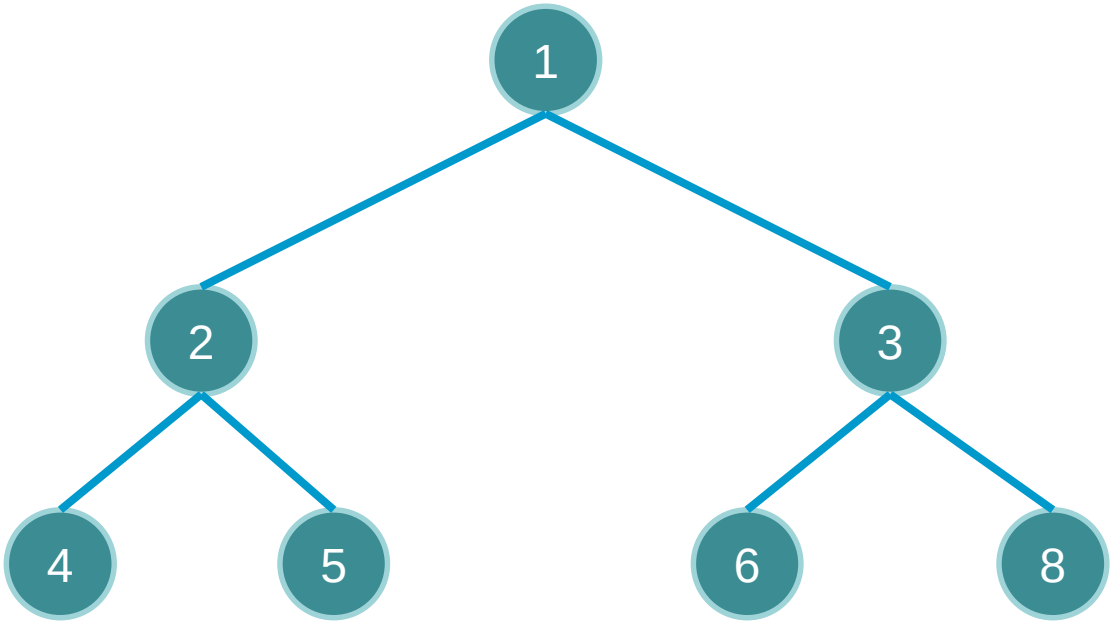
บทเรียนย่อย

11.4 Binary Trees Concept

โครงสร้างต้นไม้แบบทวิภาค นิยาม

1. Binary Tree ต้องมี Root Node
2. ในแต่ละโหนดต้องมี Edge แค่ ซ้าย (Left Subtree) กับ ขวา (Right Subtree)
3. Left Subtree และ Right Subtree ก็ต้องเป็น Binary Tree

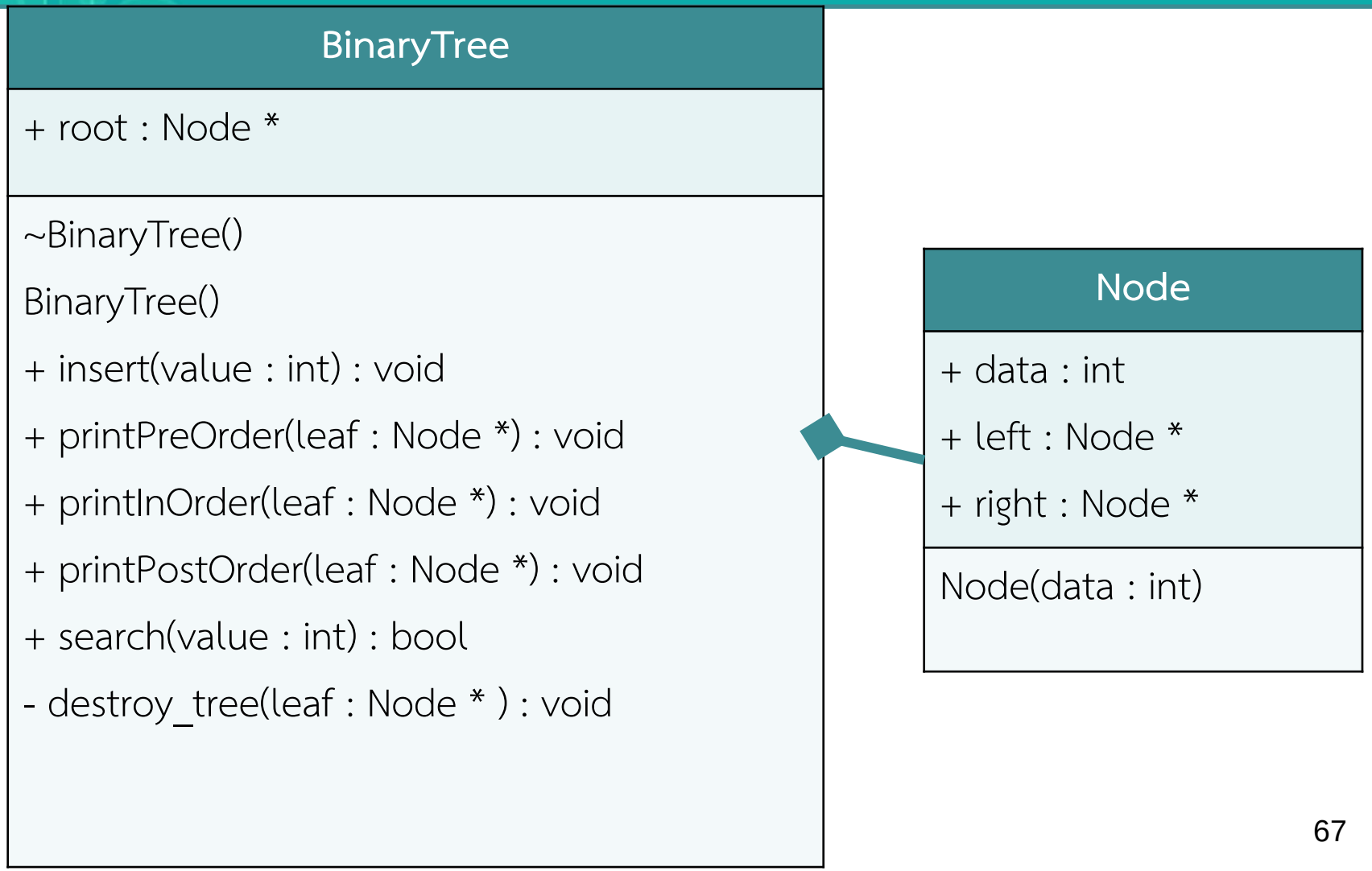
ตัวอย่าง Binary Tree



บทที่ 11 Tree

บทเรียนย่อย

11.5 Binary Trees Operations and Implementation



ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(5)

1. new Node

ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(5)

1. new Node
2. กำหนดให้ new Node มีค่า value เท่ากับค่าที่ระบุ

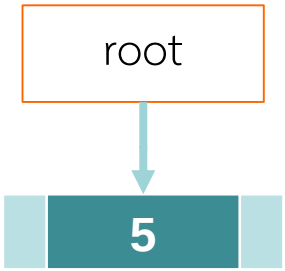
ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(5)

1. new Node
2. กำหนดให้ new Node มีค่า value เท่ากับค่าที่ระบุ
3. กำหนดให้ new Node เท่ากับ root

ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(5)



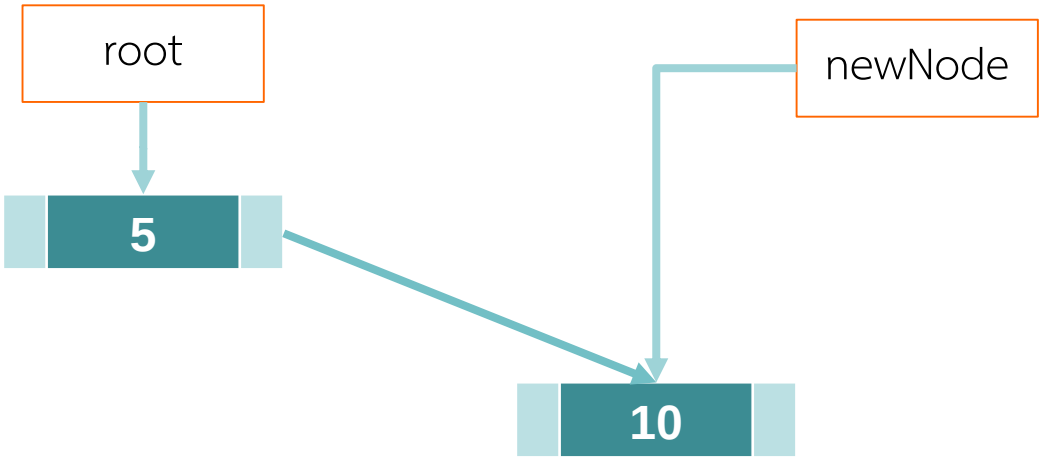
ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(10)

1. new Node
2. กำหนดให้ new Node มีค่า value เท่ากับค่าที่ระบุ
3. ถ้ามีค่าน้อยกว่า ให้ อยู่ข้างซ้าย และข้างซ้ายต้องเป็นค่าว่าง
4. ถ้ามีค่ามากกว่า ให้อยู่ ข้างขวา และข้างขวาต้องเป็นค่าว่าง

ตัวอย่างการทำงานในการ Insert ของ Binary Trees

insert(10)



1. new Node
2. กำหนดให้ new Node มีค่า value เท่ากับค่าที่ระบุ
3. ถ้ามีค่าน้อยกว่า ให้ อยู่ข้างซ้าย และข้างซ้ายต้องเป็นค่าว่าง
4. ถ้ามีค่ามากกว่า ให้อยู่ ข้างขวา และข้างขวาต้องเป็นค่าว่าง

ตัวอย่างการ code ในการ deconstruction ของ Binary Trees

```
~BinaryTree(){
      this->destroy_tree(this->root);
}
void destroy_tree(node *leaf) {
       if(leaf!=NULL) {
              destroy_tree(leaf->left);
              destroy_tree(leaf->right);
              delete leaf;
       }
}
```

ตัวอย่างการ code ในการ printInOrder ของ Binary Trees

```
void printInOrder(node *leaf)
{
    if (leaf == NULL)
        return;
    printInOrder(node->left);
    count << node->data;
    printInOrder(node->right);

}
```

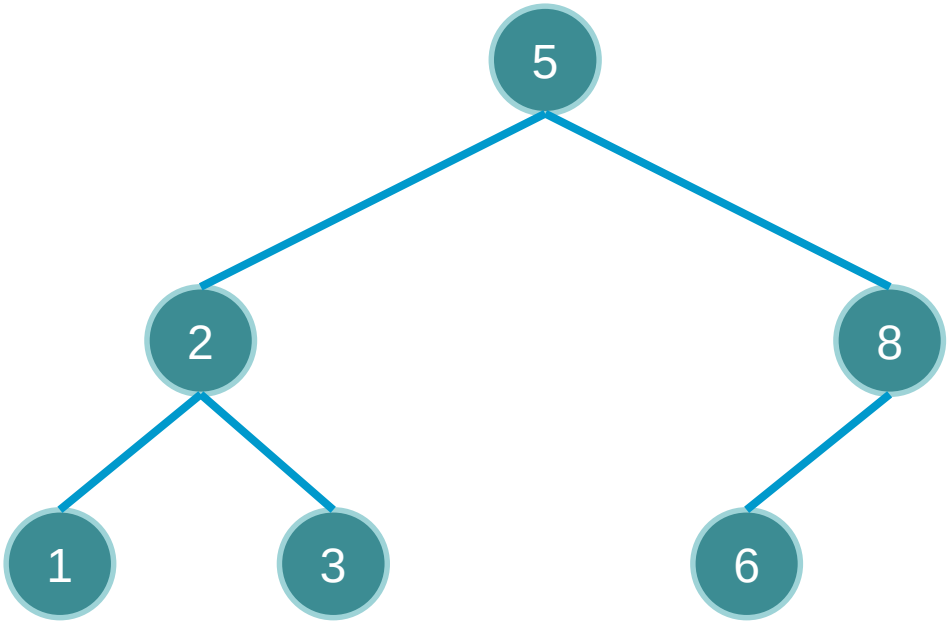
บทที่ 11 Tree

บทเรียนย่อย

11.6 Binary Search Trees Concept

การค้นหาข้อมูลแบบโครงสร้างต้นไม้ทวิภาค เป็นการค้นหาค่าใด ๆ ที่ต้องการจากโครงสร้างข้อมูลที่ได้จัดเก็บข้อมูลนั้นไว้ โดยในการจัดเก็บข้อมูลต้องเป็นโครงสร้างแบบต้นไม้ โดยที่มีข้อมูลน้อยกว่า อยู่ทางด้านซ้าย และมีข้อมูลที่มากกว่า อยู่ทางด้านขวา

ตัวอย่าง Tree ในรูปแบบของ Binary Search Trees

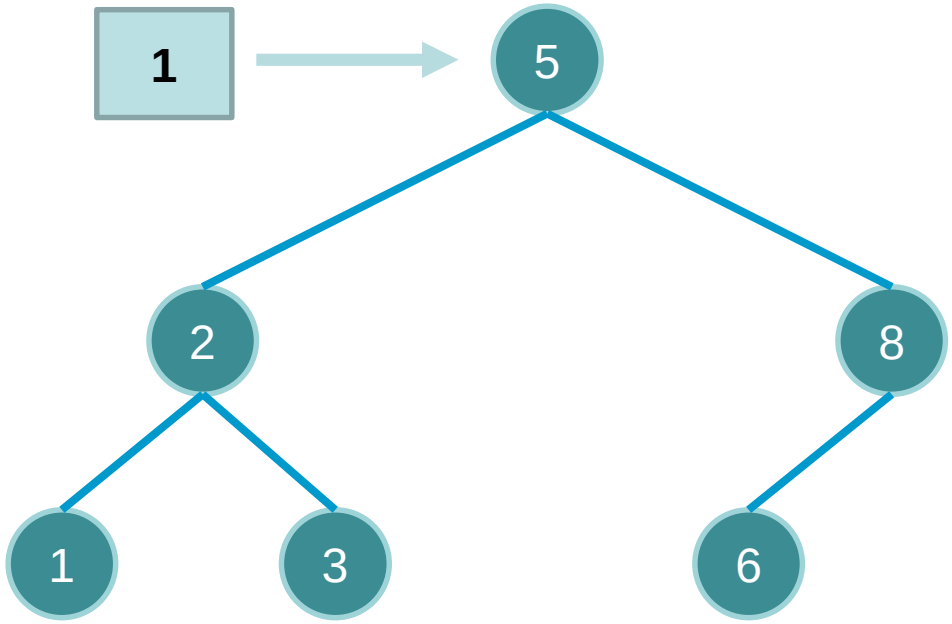


บทที่ 11 Tree

บทเรียนย่อย

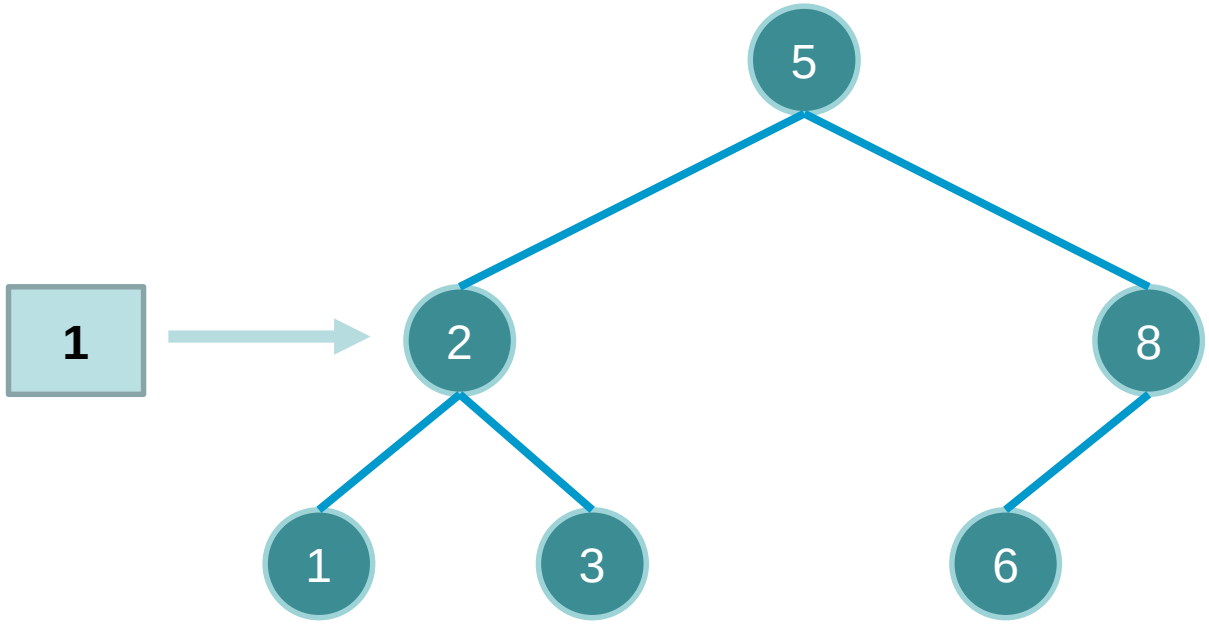
11.7 Binary Search Trees Implementation

search(1)



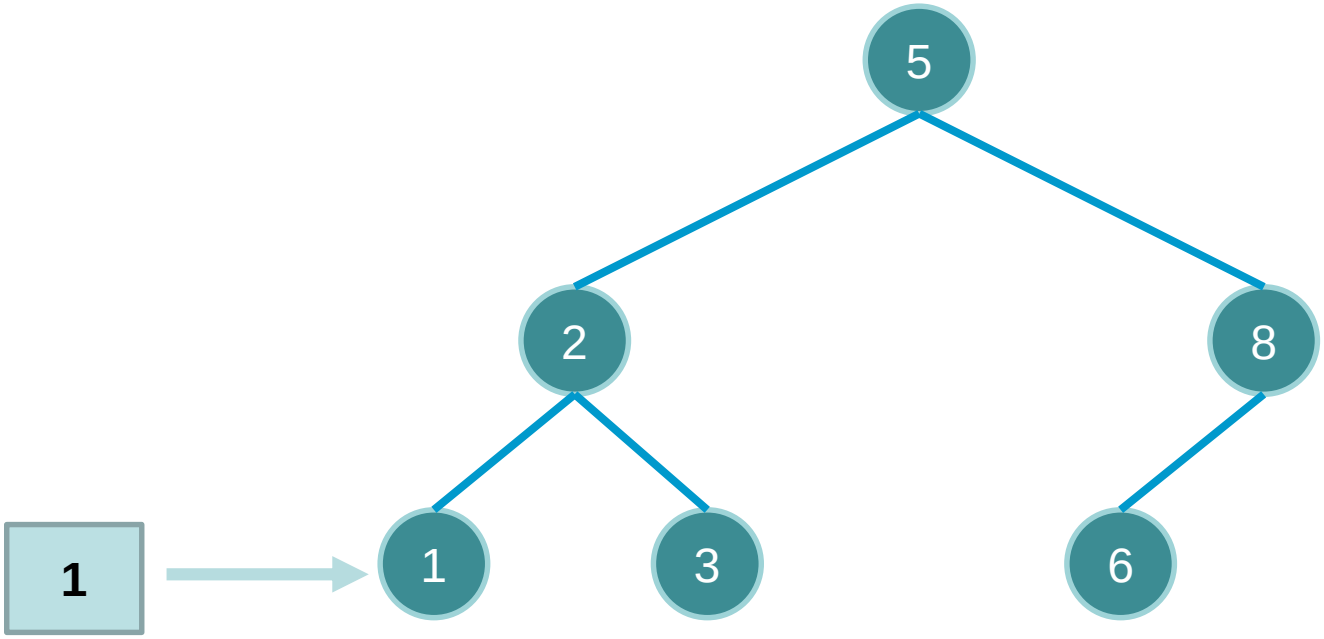
11.7 Binary Search Trees Implementation

search(1)



11.7 Binary Search Trees Implementation

search(1)



11.7 Binary Search Trees Implementation

```
bool search(int value){
        Node * Leaf = this->root;
        while(leaf!=NULL)  {
            if(value == leaf->data)      return true;
            if(value < leaf->data)
                        leaf = leaf->left;
            else
                        leaf = leaf->right;
      }
      return false
}
```

แบบฝึกหัด

1. สร้างคลาสชื่อ BinaryTree โดยมีความสามารถในการจัดการข้อมูล ดังนี้
   - Insert
   - Trees Traversal (PreOrder,InOrder,PostOrder)
   - Search แบบ Binary Search Tree

2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส BinaryTree ของตนเอง

3. สร้างในไดรเรกทอรี่ชื่อ example08 โดยตั้งชื่อไฟล์ binaryTree.cpp