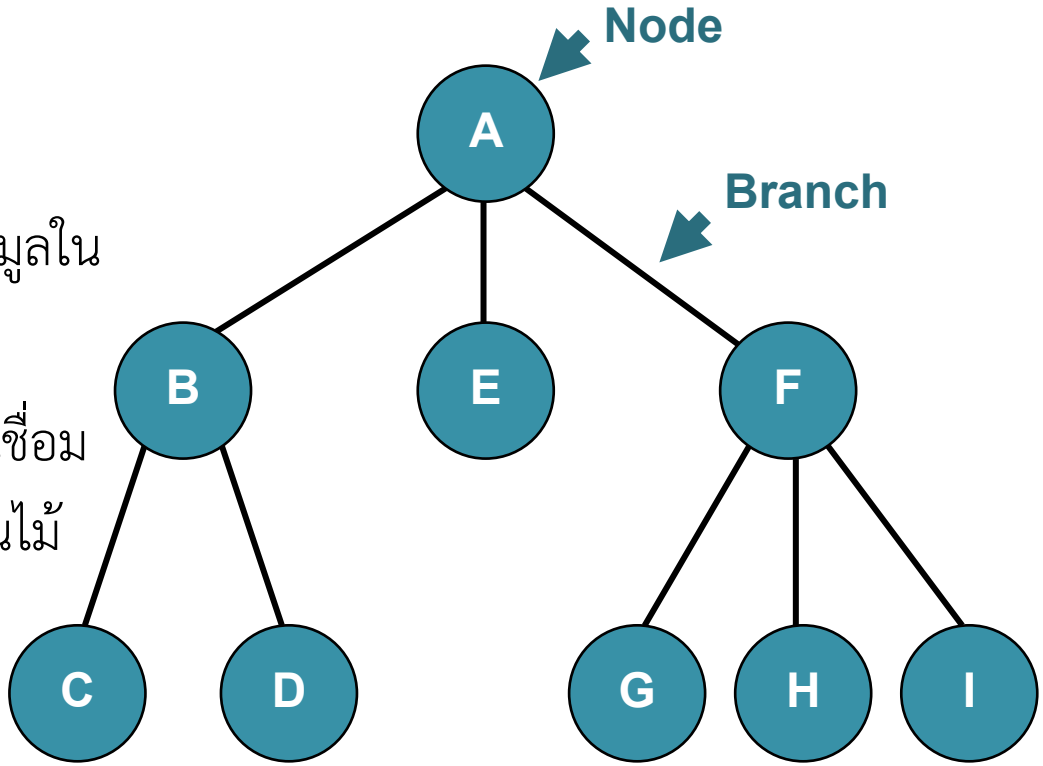
Chapter 4

Introduction to Trees &

Binary Search Trees

Trees

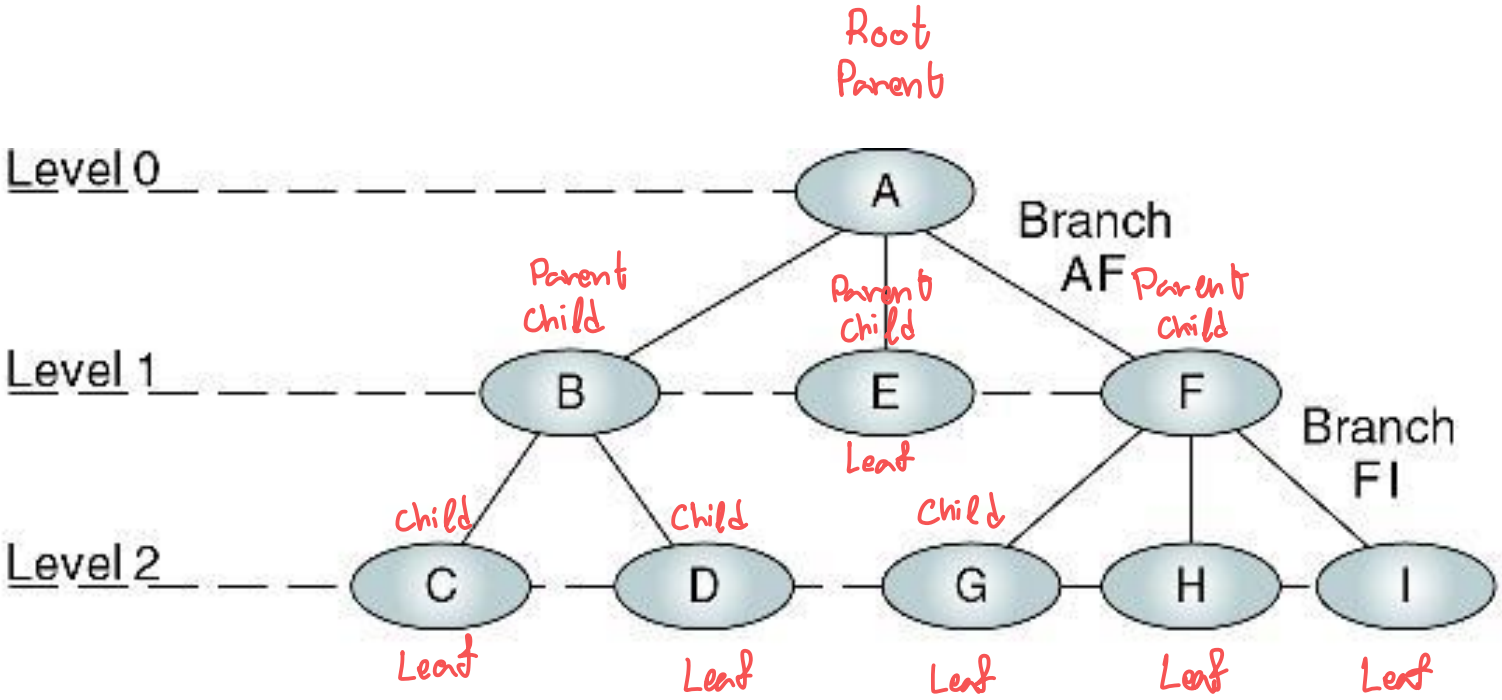
- เป็นโครงสร้างข้อมูล ที่ประกอบด้วย
  - โหนด (Node) เป็นกลุ่มข้อมูลในโครงสร้างต้นไม้
  - กิ่ง (Branch) เป็นส่วนที่ใช้เชื่อมโหนดต่างๆ ในโครงสร้างต้นไม้



Terminology

- Root : โหนดแรกสุดของต้นไม้
- Parent : โหนดที่อยู่สูงกว่าโหนดที่พิจารณา 1 ระดับ
- Child : โหนดที่อยู่ต่ำกว่าโหนดที่พิจารณา 1 ระดับ
- Sibling : โหนดที่อยู่ระดับเดียวกันและมีพ่อเดียวกัน
- Leaf : โหนดที่ไม่มีลูก
- Ancestor : โหนดที่อยู่ในเส้นทางการเดินจากรูทมายังโหนดที่พิจารณา
- Descendent : โหนดที่อยู่ในเส้นทางการเดินจากโหนดที่พิจารณา ไปจนหมดต้นไม้
- Degree : จำนวนลูกของโหนดที่พิจารณา
- Level : ระยะทางจากรูทมายังโหนดที่พิจารณา
- Depth : ความสูงของต้นไม้ (= Level ของโหนดที่อยู่ห่างจากรูทมากที่สุด + 1)
- Subtree : ต้นไม้ย่อย
- Internal : ไม่ใช่ root และไม่ใช่ leaf

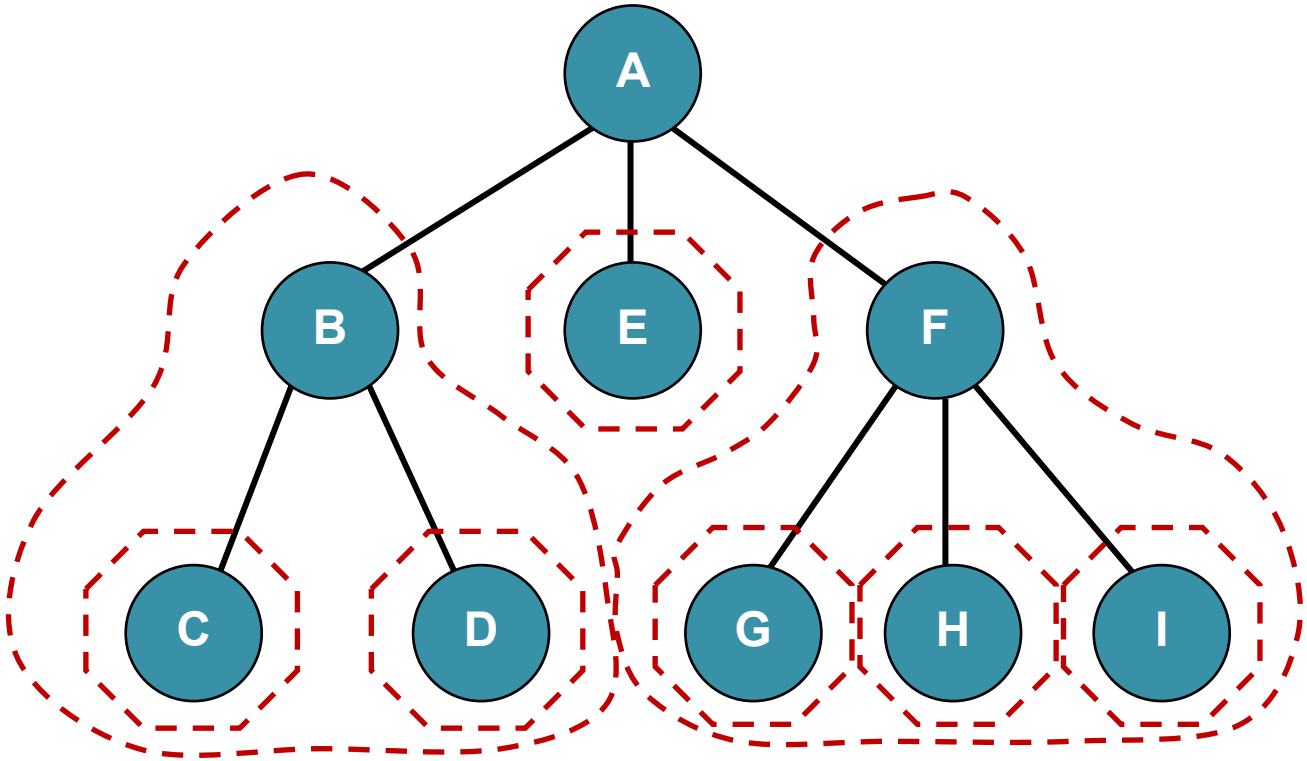
Terminology (cont.)



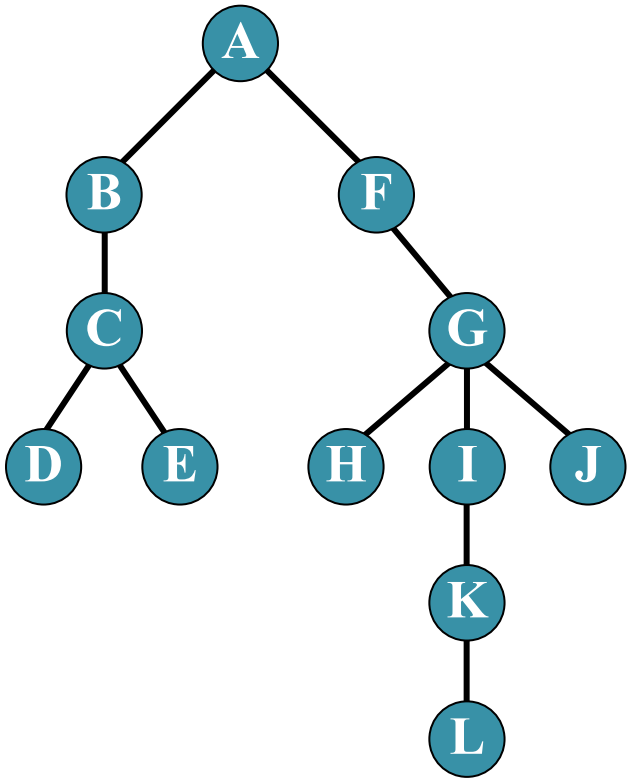
Root: A	Siblings: {B,E,F}, {C,D}, {G,H,I}
Parents: A, B, F	Leaves: C,D,E,G,H,I
Children: B, E, F, C, D, G, H, I	Internal nodes: B,F

Subtrees

- โครงสร้างต้นไม้ย่อยๆ ในต้นไม้ใหญ่ทั้งต้น



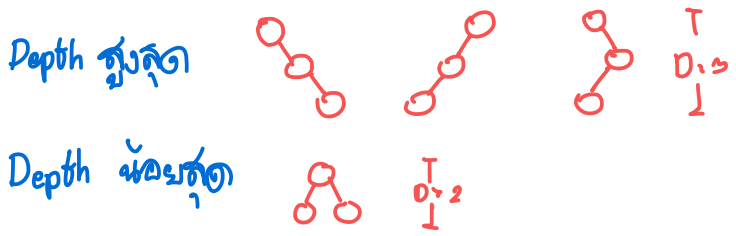
Terminology (cont.)



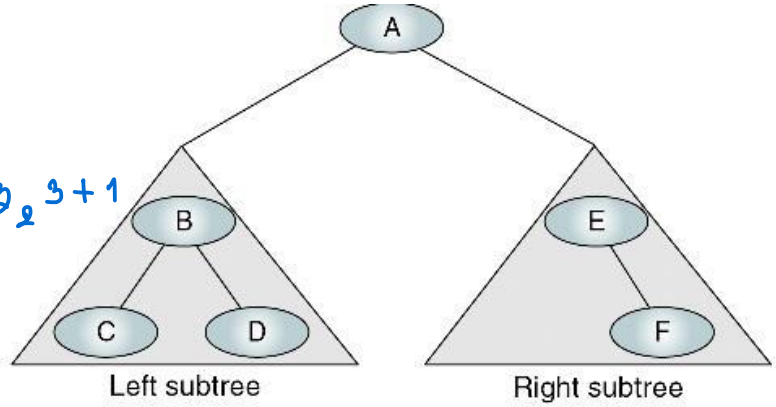
หาคำตอบต่อไปนี้

- Root : A
- Leaves : D E H L J
- Internal nodes : B C F G I K
- Ancestors of H : A F G
- Descendents of F : G H I J K L
- Siblings of I : H J
- Degrees of L : 0
- Parent of K : I
- Children of C : D E
- Depth : 6
- Height Levels of the tree : 5
- Level of J : 3
- Number of all subtrees : 11

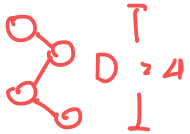
Binary Trees



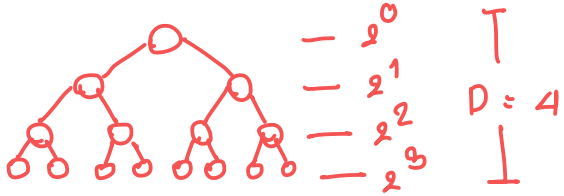
- เป็นโครงสร้างต้นไม้ ที่แต่ละโหนดจะมีโหนดลูกได้ไม่เกิน 2 ต้นไม้ย่อย
- ถ้าต้นไม้มีทั้งหมด N โหนด N = 3
  - Depth สูงสุด = N ; 3
  - Depth น้อยสุด = $\log_2 N + 1$; $\log_2 3 + 1$
- ถ้าต้นไม้มีความสูง D D = 4
  - จน.โหนดน้อยสุด = D ; 4
  - จน.โหนดมากสุด = $2^D - 1$; $2^4 - 1$



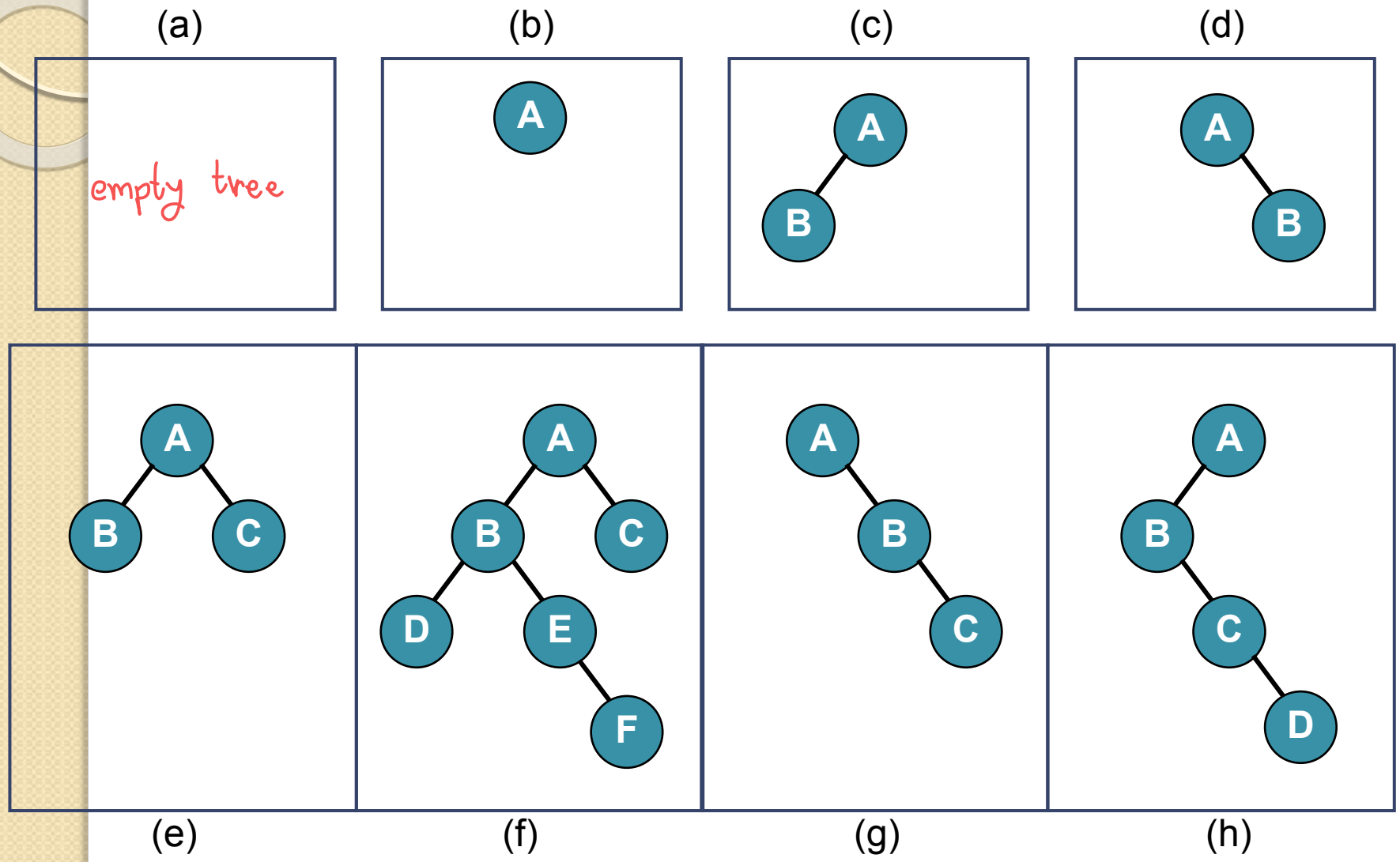
Node น้อยสุด



Node มากสุด

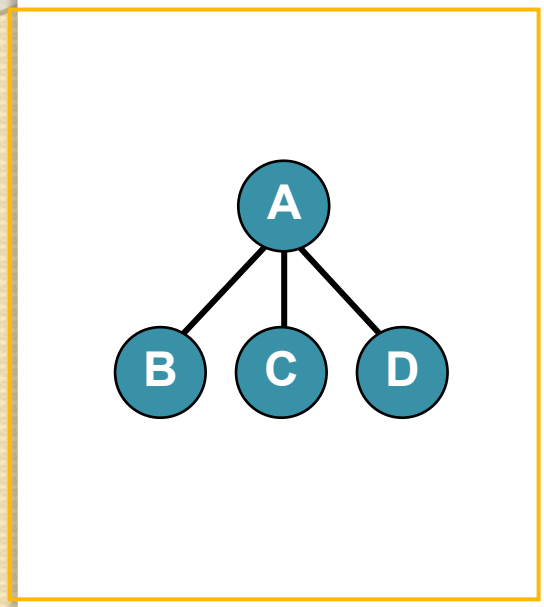


Binary Trees

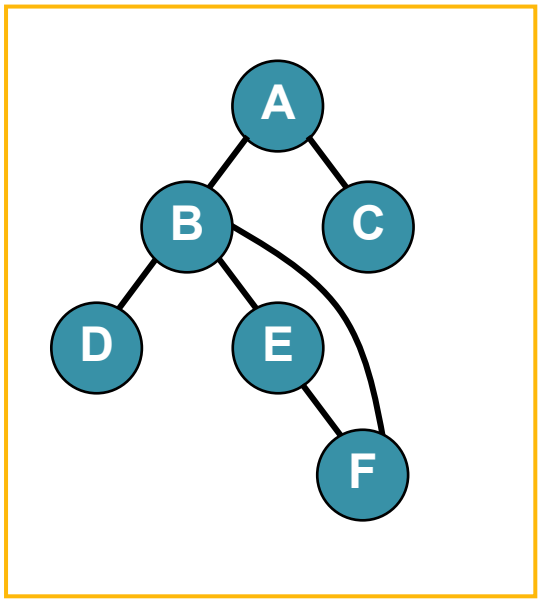


Structures which are not binary trees

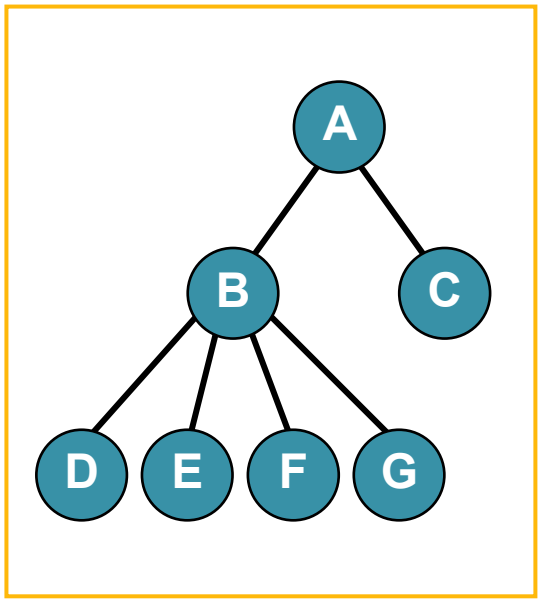
√ Tree
× Binary Tree



× Tree



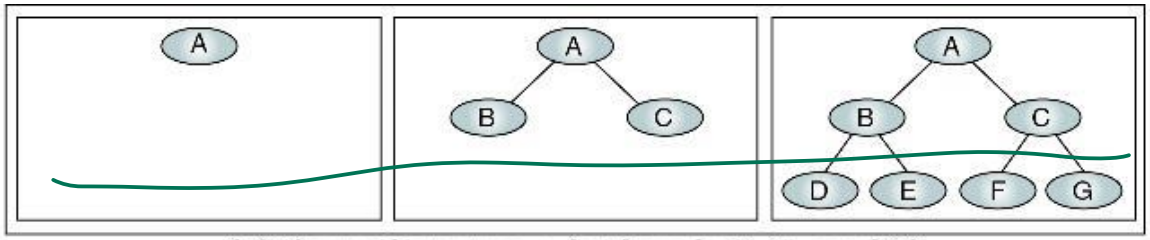
√ Tree
× Binary Tree



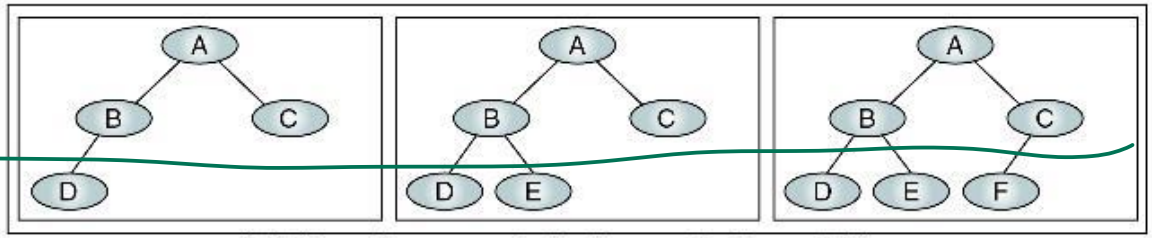
Complete and Nearly Complete Trees

- Complete Trees : มีจน.โหนดสูงสุด เมื่อมีความสูง D
- Nearly Complete Tree : มีความสูงน้อยสุด เมื่อมีโหนด N โหนด

ครบทุก level except lowest level



(a) Complete trees (at levels 0, 1, and 2)



(b) Nearly complete trees (at level 2)

D = 3

$N = 2^3 - 1 = 7$

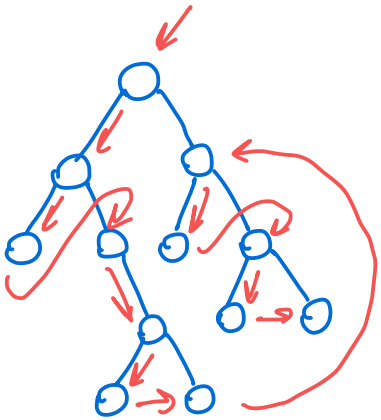
N = 4, 5, 6

$D = \log_2 4 + 1 = 3$

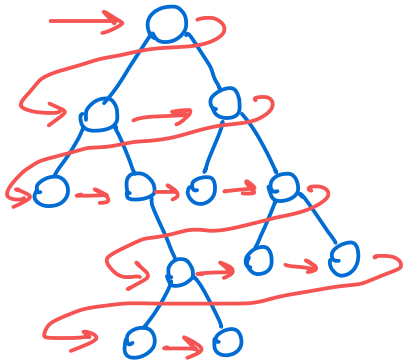
$D = \log_2 5 + 1 = 3$

Binary Tree Traversals

- Depth-first traversal : เริ่มท่องจากรูท ไปใน descendent ของลูกตัวแรกจนหมด ค่อยไปท่องในลูกตัวถัด
- Breadth-first traversal : เริ่มท่องจากรูทแบบแนวกว้าง ไปยังลูกทุกตัวก่อน แล้วค่อยท่องไปที่ระดับลูกของลูกไปเรื่อยๆ



Depth-first

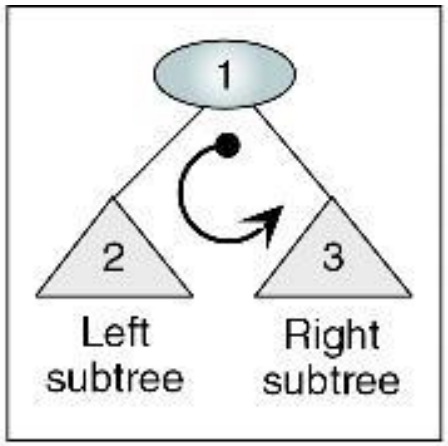


Breadth-first

Depth-first Traversals

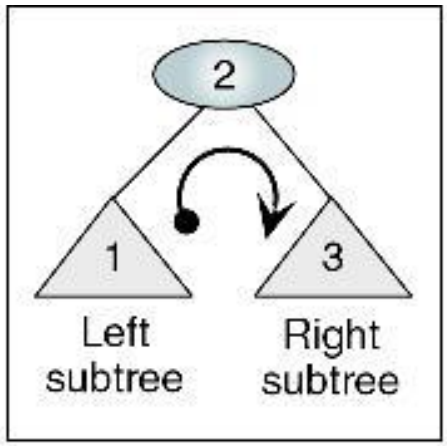
- เริ่มท่องจากรูท ไปใน descendent ของลูกตัวแรกจนหมด ค่อยไปท่องในลูกตัวถัด

Root → L → R



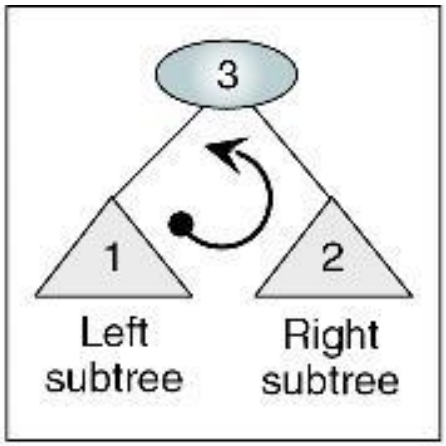
(a) Preorder traversal

L → Root → R



(b) Inorder traversal

L → R → Root



(c) Postorder traversal

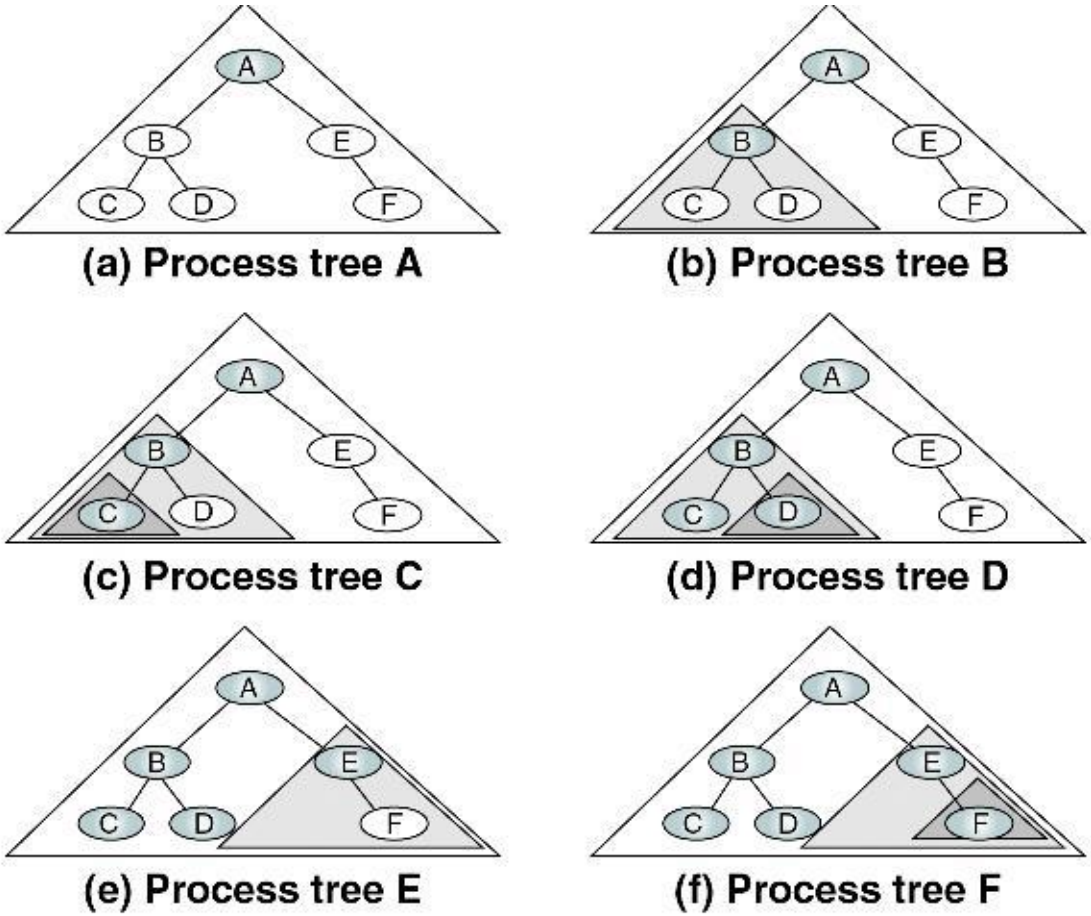
ยกตัวอย่าง การเข้าถึง root

Preorder Traversal

- ท่องตามลำดับ root -> left subtree -> right subtree

```
Algorithm preOrder (root)
Traverse a binary tree in node-left-right sequence.
   Pre  root is the entry node of a tree or subtree
   Post each node has been processed in order
1  if (root is not null)
   1   process (root)
   2   preOrder (leftSubtree)
   3   preOrder (rightSubtree)
2  end if
end preOrder
```

Preorder Traversal (cont.)



(a) Process tree A

(b) Process tree B

(c) Process tree C

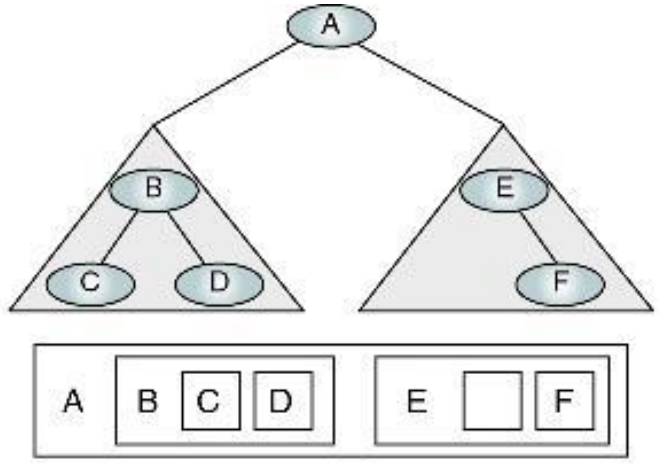
(d) Process tree D

(e) Process tree E

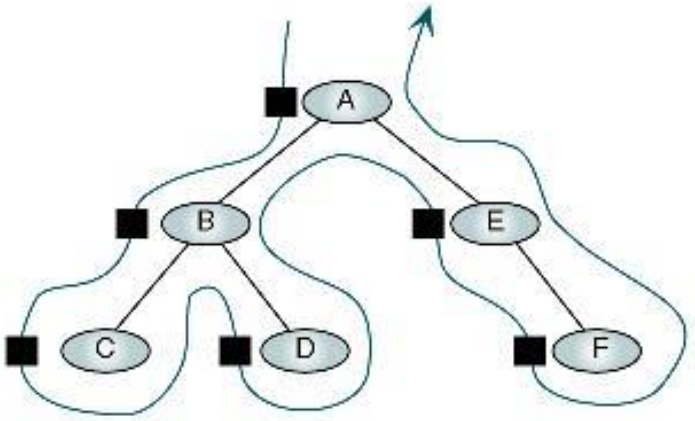
(f) Process tree F

Algorithmic Traversal of Binary Tree

Preorder Traversal (cont.)



(a) Processing order



(b) "Walking" order

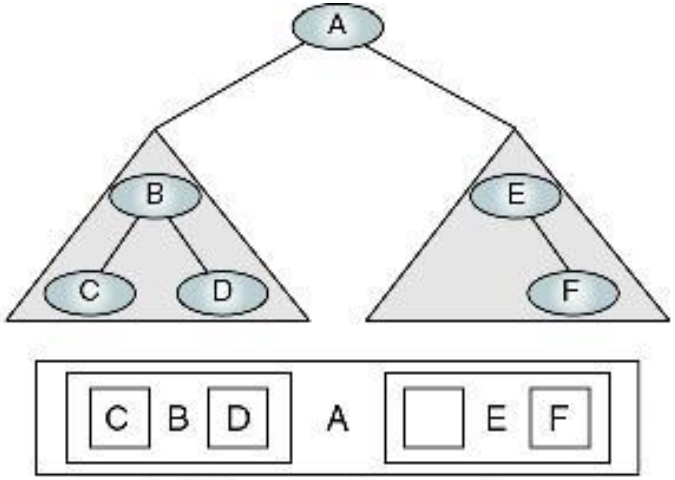
Preorder Traversal—A B C D E F

Inorder Traversal

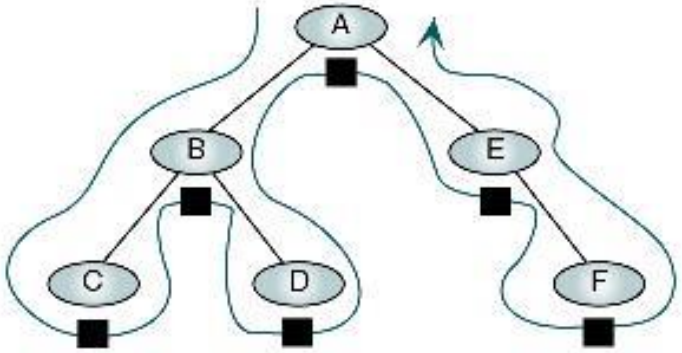
- ท่องตามลำดับ left subtree -> root -> right subtree

```
Algorithm inOrder (root)
Traverse a binary tree in left-node-right sequence.
   Pre  root is the entry node of a tree or subtree
   Post each node has been processed in order
 1 if (root is not null)
   1   inOrder (leftSubTree)
   2   process (root)
   3   inOrder (rightSubTree)
 2 end if
end inOrder
```

Inorder Traversal (cont.)



(a) Processing order



(b) "Walking" order

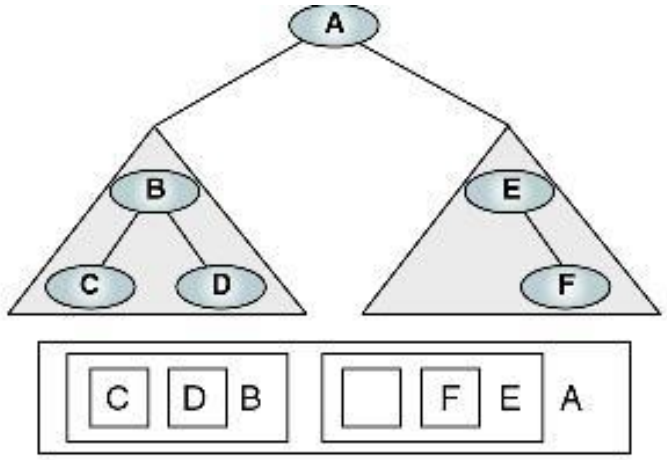
Inorder Traversal—C B D A E F

Postorder Traversal

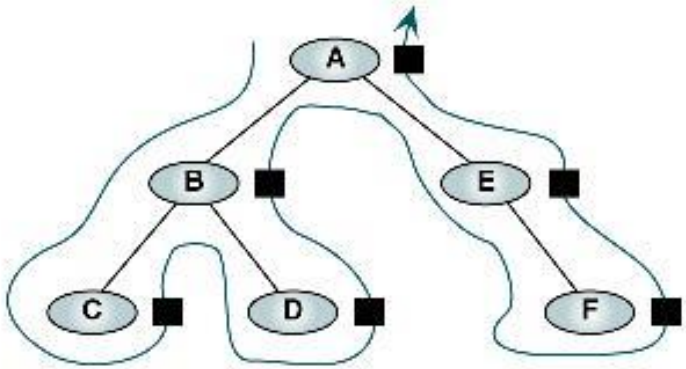
- ท่องตามลำดับ left subtree -> right subtree -> root

```
Algorithm postOrder (root)
Traverse a binary tree in left-right-node sequence.
   Pre  root is the entry node of a tree or subtree
   Post each node has been processed in order
1  if (root is not null)
   1   postOrder (left subtree)
   2   postOrder (right subtree)
   3   process (root)
2  end if
end postOrder
```

Postorder Traversal



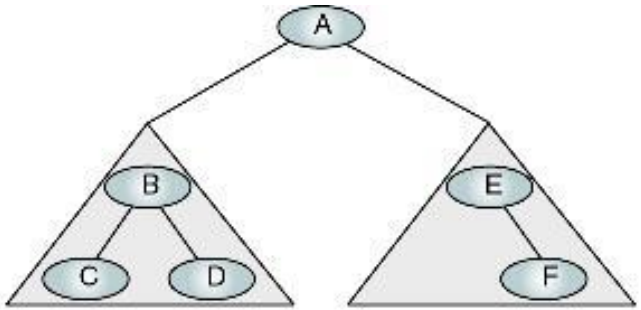
(a) Processing order

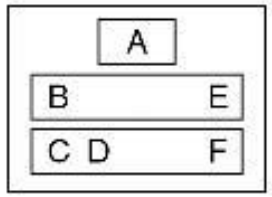


(b) "Walking" order

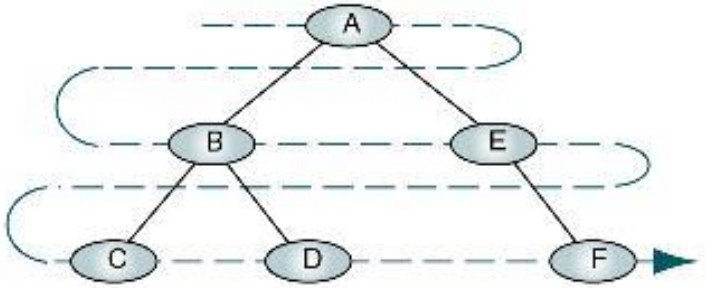
Postorder Traversal—C D B F E A

Breadth-first Tree





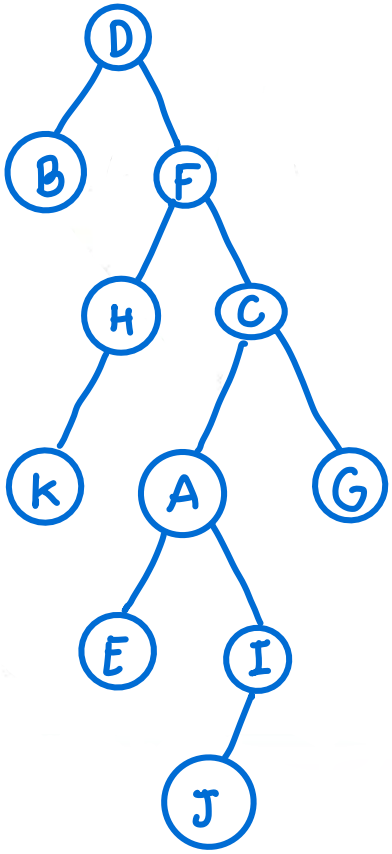
(a) Processing order



(b) "Walking" order

Breadth-first Traversal

Root : D



Depth - first :

- Preorder

  D B F H K C A E I J G

- Inorder

  B D K H F E A J I C G

- Postorder

  B K H E J I A G C F D

Breadth - first :

D B F H C K A G E I J

Quiz

- ให้หารูทของ Binary Tree เมื่อมีลำดับการท่องเข้าไปในต้นไม้ดังนี้
  - a) Tree with postorder traversal: FCBDG
  - b) Tree with preorder traversal: IBCDFEN
  - c) Tree with inorder traversal: CBIDFGE
- ให้วาดรูป Binary Tree เมื่อมีลำดับการท่องเข้าไปในต้นไม้ดังนี้
  - Preorder : JCBADEFIGH
  - Inorder : ABCEDFJGIH
- ให้วาดรูป Binary Tree ที่เป็นไปได้ เมื่อมีโหนดทั้งหมด 3 โหนด (A,B,C) (บอกด้วยว่าเป็น complete, nearly complete หรือไม่)
- ให้วาดรูป Nearly complete binary tree ที่มีการท่องไปในต้นไม้แบบ Breadth-first traversal เป็น JCBADEFIG