Discrete Computational Structures

อ. ภูริวัจน์ วรวิชัยพัฒน์

อ้างอิงจากหนังสือ: Discrete Computational Structure, คทา ประดิษฐ์วงศ์

ทบทวน (Recap) ก่อนกลางภาค

การนับเบื้องต้น

- ตย: การคูณ, การเรียงสับเปลี่ยน, จัดหมู่

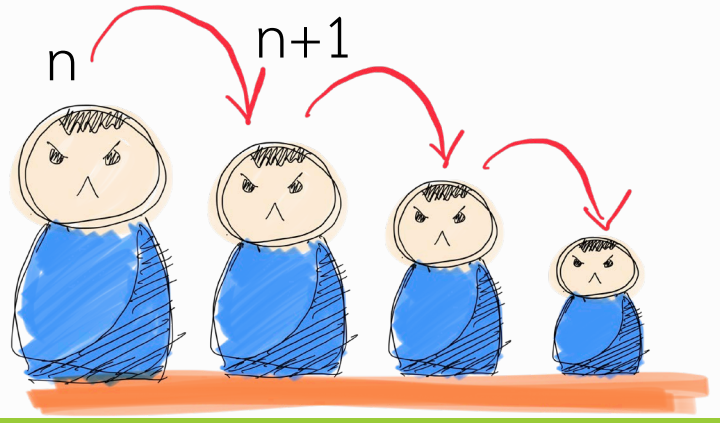
ฟังก์ชันก่อกำเนิด (Generating function)

ความสัมพันธ์เวียนเกิด (Recurrent relation)

เครื่องยนต์ สถานะจำกัด (Finite automata)

กราฟเบื้องต้น (Graphs)

- G (V, E), การเดิน, สะพาน, กราฟออยเลอร์

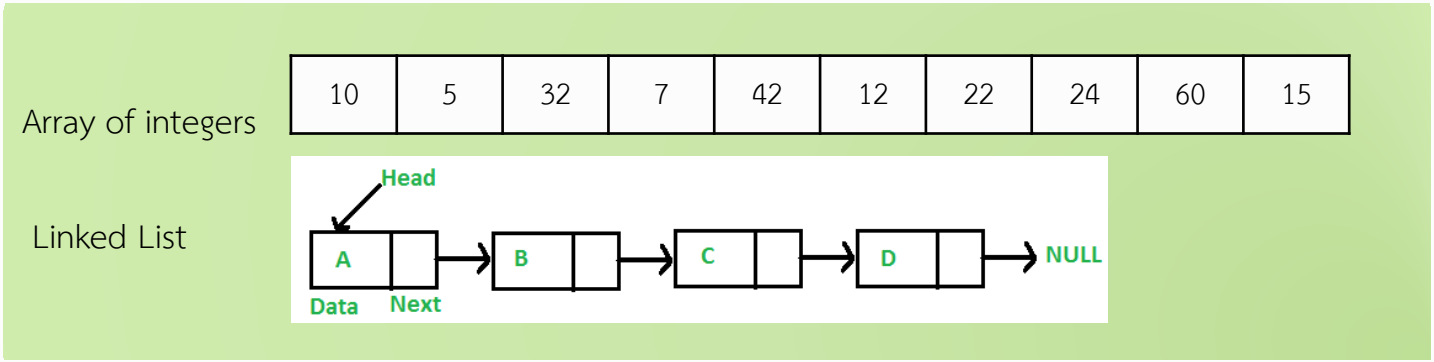


เนื้อหาปลายภาค - Overview

- **ต้นไม้, Tree** TODAY
- กราฟทิศทาง, Directed graph
- ข่ายงาน และการประยุกต์ใช้ข่ายงาน, Network
- การหาเส้นที่สั้นที่สุด Shortest path, Dijkstra’s algorithm
- การหาต้นไม้ทอดข้ามที่น้อยที่สุด Kruskal’s algorithm & Prim’s algorithm

ต้นไม้ - Tree

เมื่อเราเรียนโปรแกรมมิ่ง การจัดเก็บข้อมูลที่คุ้นเคยมักจะเป็นเส้นตรง (linear) เช่น Array, หรือ Linked List

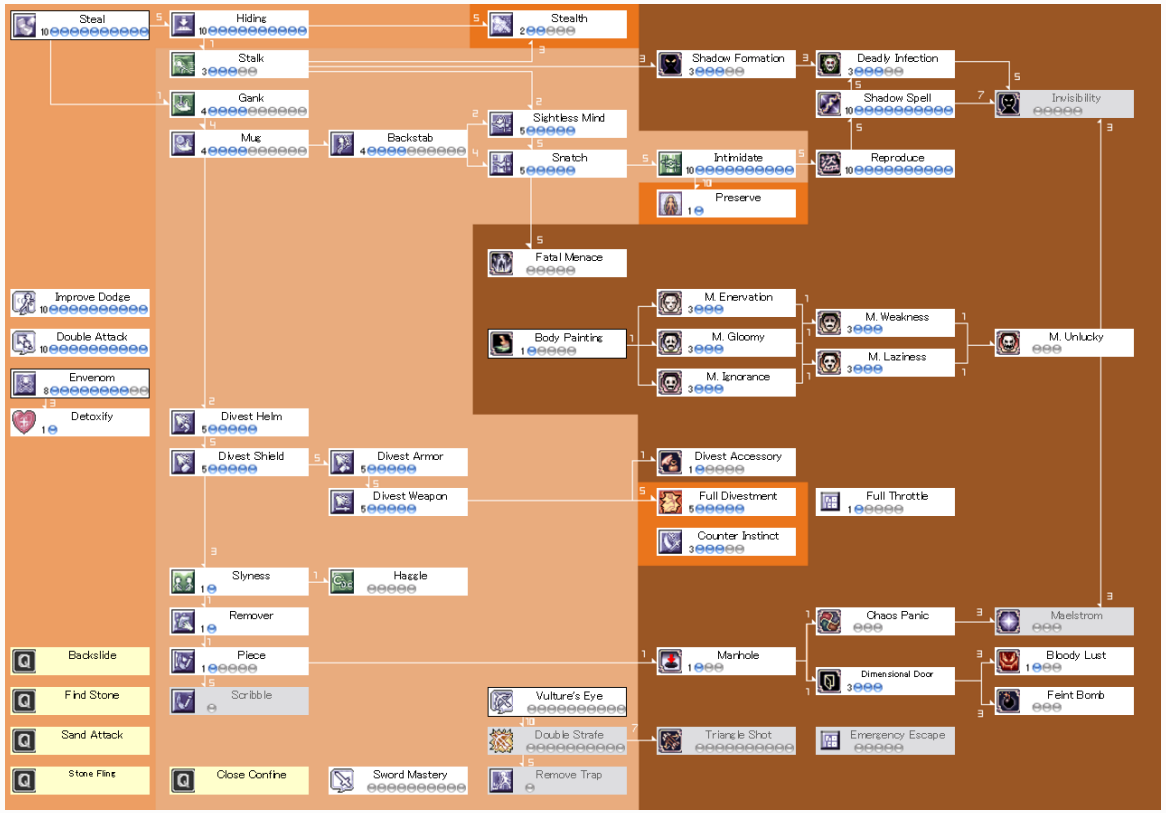


ต้นไม้ หรือ Tree

- เป็นกราฟการจัดเก็บข้อมูลที่ไม่เป็นเส้นตรง (Non-linear)
- นิยมใช้มากในทางวิทยาศาสตร์คอมพิวเตอร์
- ใช้ในการเก็บข้อมูล, การค้นหา, แสดงผล, เรียบเรียงแนวคิด หรือแนวทางการตัดสินใจ

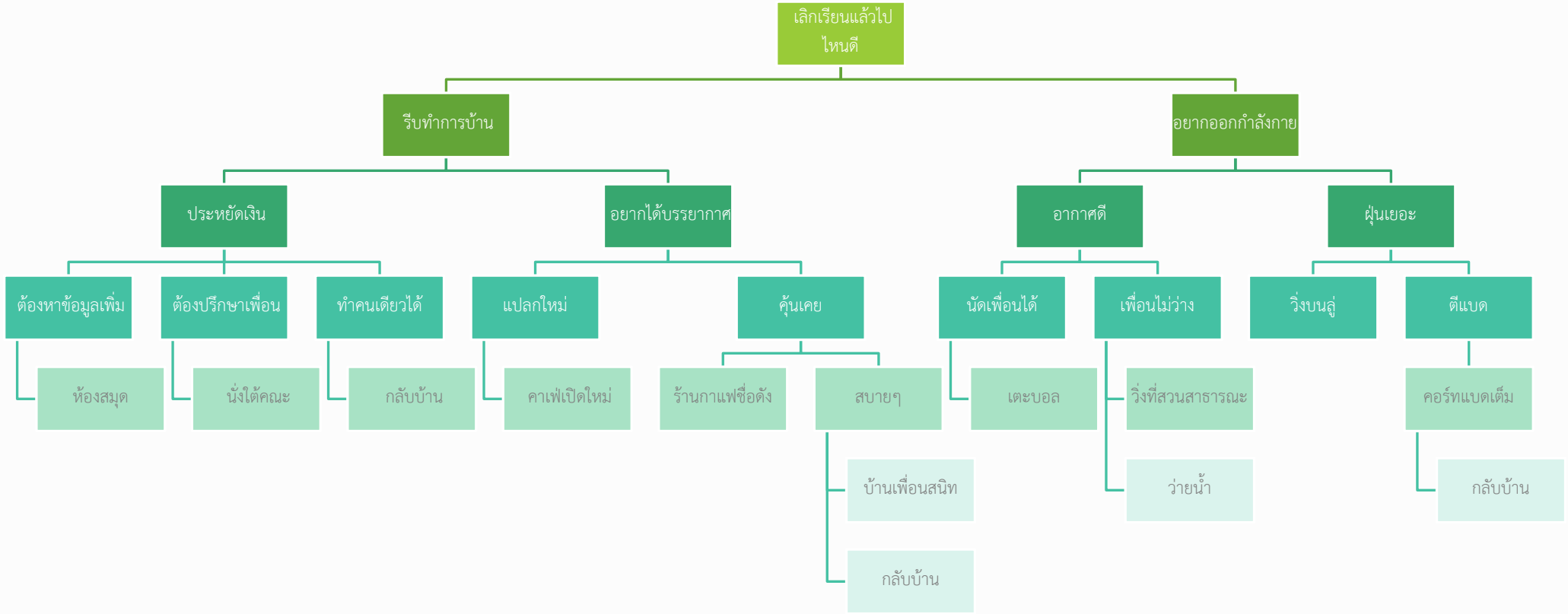
ตัวอย่างต้นไม้

ตัวอย่าง: แผนผังครอบครัว แผนผังระดับผู้บริหาร หรือ Skill tree



ตัวอย่างต้นไม้

ตัวอย่าง: ต้นไม้ตัดสินใจ, Decision tree



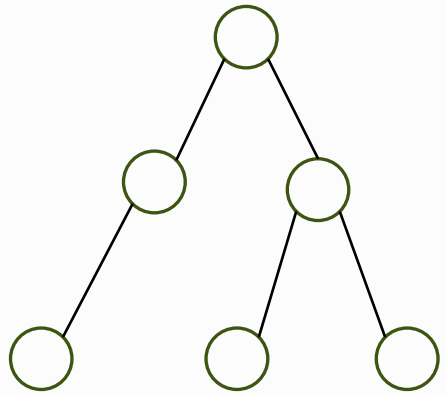
นิยามของต้นไม้ - Definition

1. ให้ a, b เป็นจุดยอดใดๆ, a ≠ b -> ในต้นไม้จะมีเส้นทางเดียวที่จาก a ไปสู่ b

2. ถ้ามีเส้นทางเชื่อมระหว่างทุกสองจุดใดๆ เพียงเส้นเดียวในกราฟ T ดังนั้น T เป็นต้นไม้

3. เส้นเชื่อมทุกเส้นในต้นไม้เป็นสะพานกราฟ

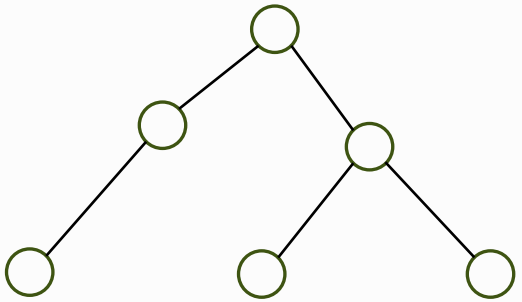
4. ต้นไม้ T ที่มีอันดับ n ต้นไม้นี้จะมีจำนวนเส้นเชื่อมเท่ากับ **n-1** เส้น เมื่อ n≥1

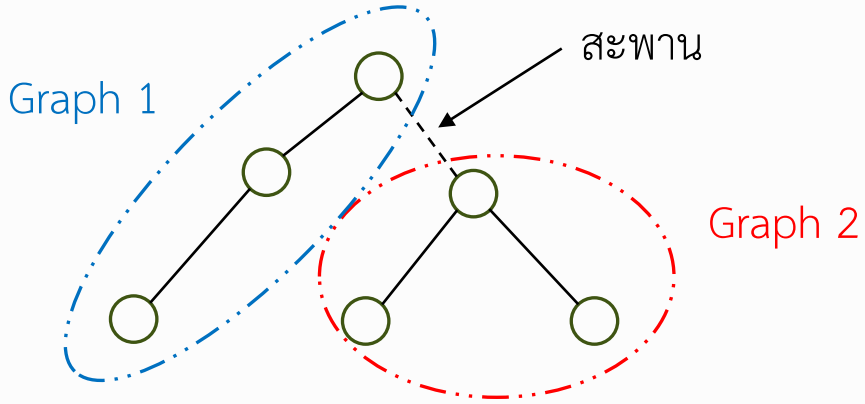


นิยามของต้นไม้ (ฉบับไม่เป็นทางการ)

1. ระหว่างจุดยอดสองจุดที่ไม่เป็นจุดเดียวกัน จะมีเส้นเชื่อมเดียวเสมอ

2. เส้นเชื่อมในต้นไม้ = สะพานกราฟ, ตัดเส้นเชื่อมออกทำให้เกิดกราฟ 2 อัน





3. จำนวนเส้นเชื่อม = จำนวน จุดยอด – 1, เมื่อ จุดยอด ≥1

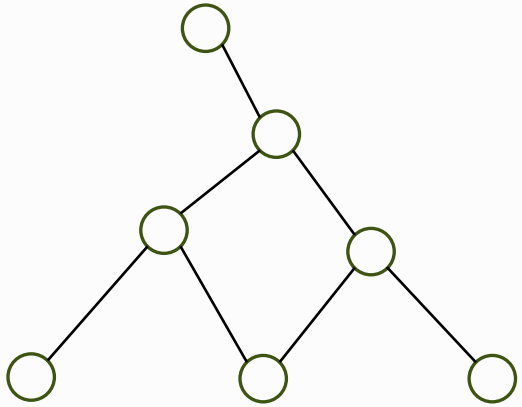
1. ระหว่างจุดยอดสองจุดจะมีเส้นเชื่อมเดียวเสมอ

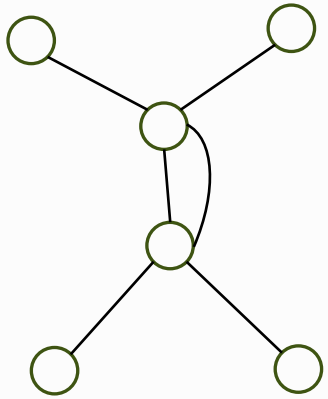
2. เส้นเชื่อมในต้นไม้ = สะพานกราฟ

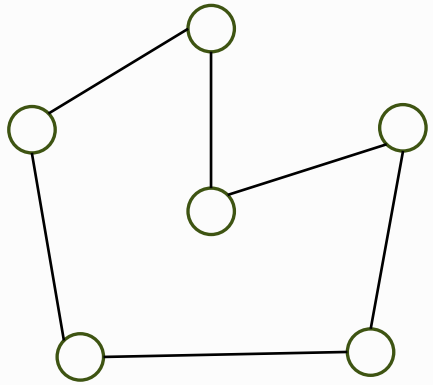
3. เส้นเชื่อม = จำนวน node – 1, เมื่อ n≥1

แบบฝึกหัด

จงแยกพิจารณารูปกราฟต่อไปนี้ และเลือกกราฟที่ตรงตามนิยามของต้นไม้ (เป็นต้นไม้)







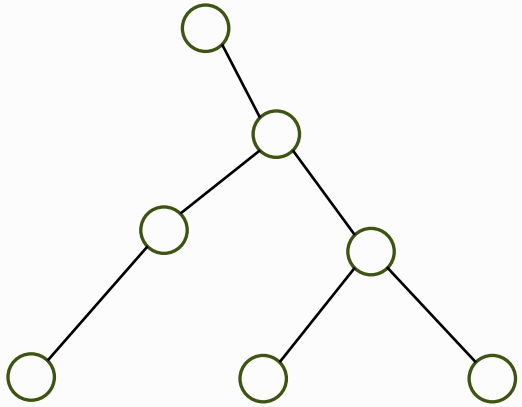
1. ระหว่างจุดยอดสองจุดจะมีเส้นเชื่อมเดียวเสมอ

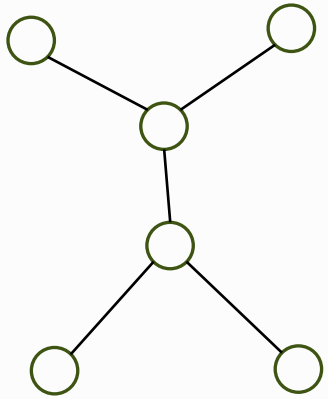
2. เส้นเชื่อมในต้นไม้ = สะพานกราฟ

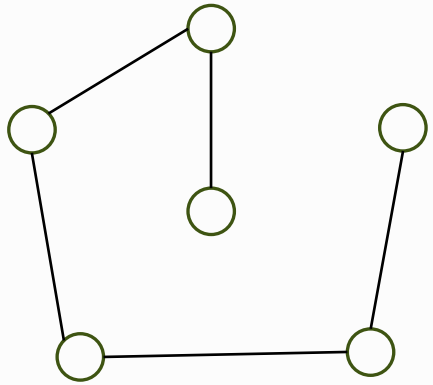
3. เส้นเชื่อม = จำนวน node – 1, เมื่อ n≥1

แบบฝึกหัด

จงแยกพิจารณารูปกราฟต่อไปนี้ และเลือกกราฟที่ตรงตามนิยามของต้นไม้ (เป็นต้นไม้)



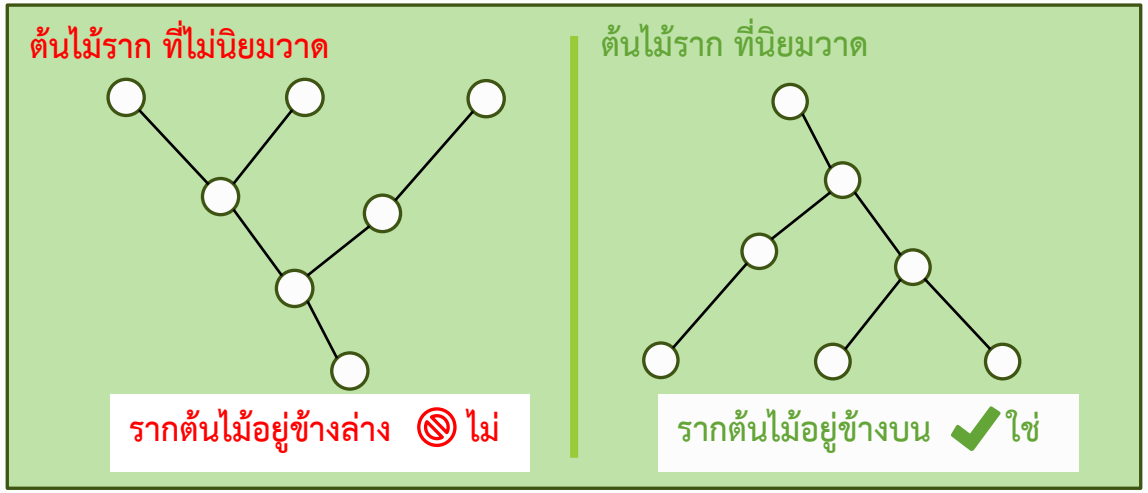


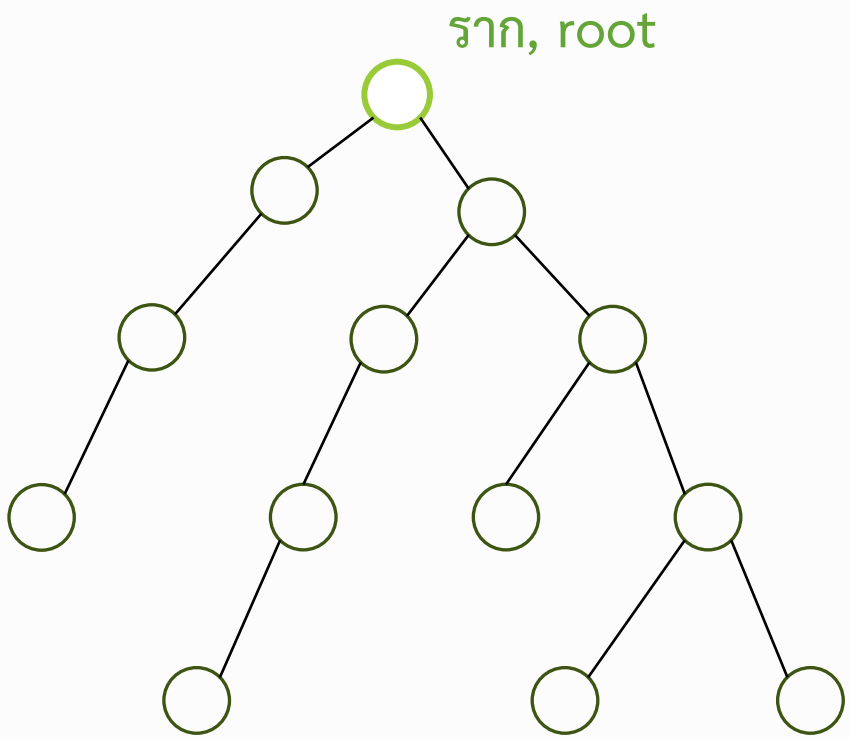


ต้นไม้ราก, Rooted tree

ต้นไม้รากคือต้นไม้ที่มีการกำหนดจุดในจุดหนึ่งให้เป็นราก

โดยทั่วไปการวาดต้นไม้รากนั้นจะวาดจากด้านบนลงมา ซึ่งแตกต่างจากต้นไม้จริงๆที่รากอยู่บนพื้น





คำศัพท์ที่เกี่ยวข้อง

- ปม, node

  จุดยอดภายในกราฟ

- ลูก, child

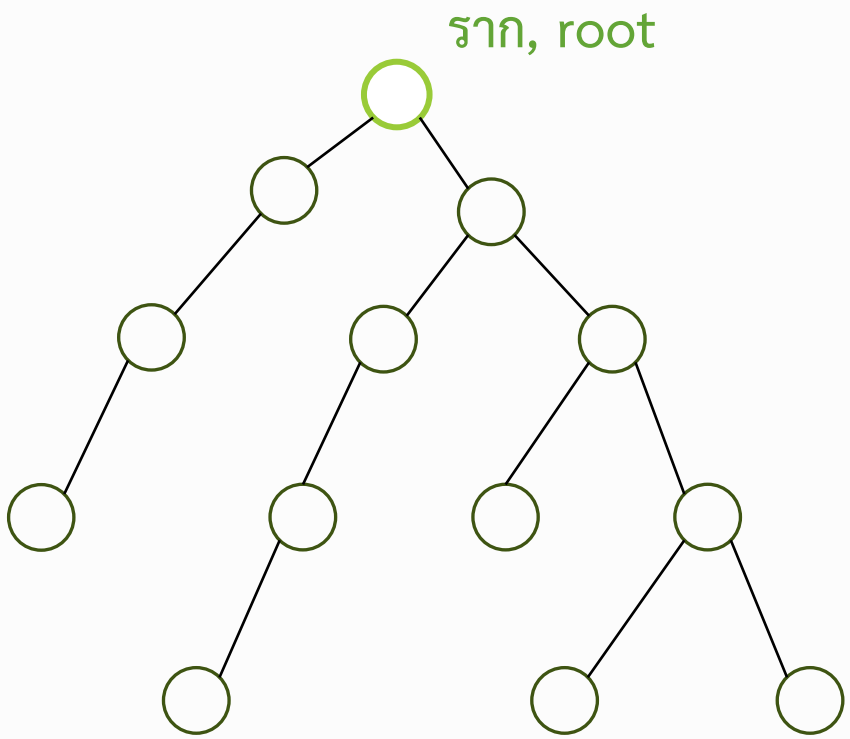
  ปมต่อออกมาที่กำลังพิจารณาไปด้านล่าง

- พ่อแม่, parent

  ปมที่อยู่ก่อนหน้าปมปัจจุบันไปหนึ่งขั้น

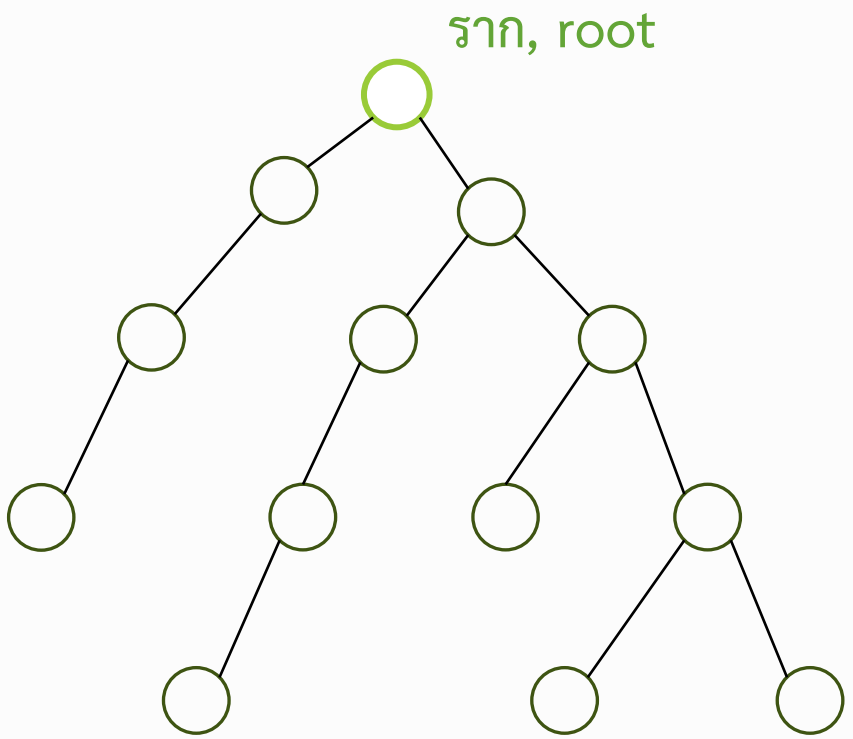
- พี่น้อง, siblings

  ปมที่มีพ่อกับแม่เหมือนกัน



คำศัพท์ที่เกี่ยวข้อง

- ราก, root
  ปมที่ไม่มีพ่อแม่
- ใบ, leaf
  ปมที่ไม่มีลูก
- ปมภายใน, internal node
  ปมที่อยู่ภายในต้นไม้ ไม่ใช่ทั้ง ราก และ ใบ
- ระดับ, level
  จำนวนเส้นเชื่อมจากรากมายังปม, รากมีระดับเป็น 0
- ความสูง, height
  จำนวนปมที่อยู่ในเส้นทางที่ยาวที่สุดจากรากถึงใบ



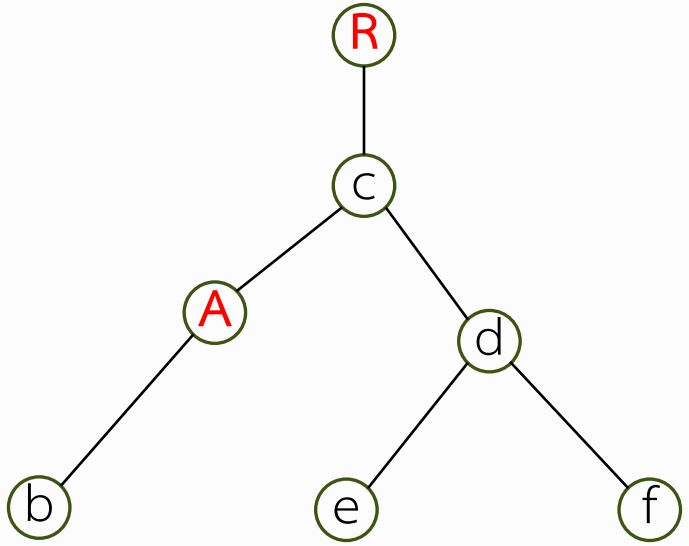
แบบฝึกหัด คำศัพท์ต้นไม้

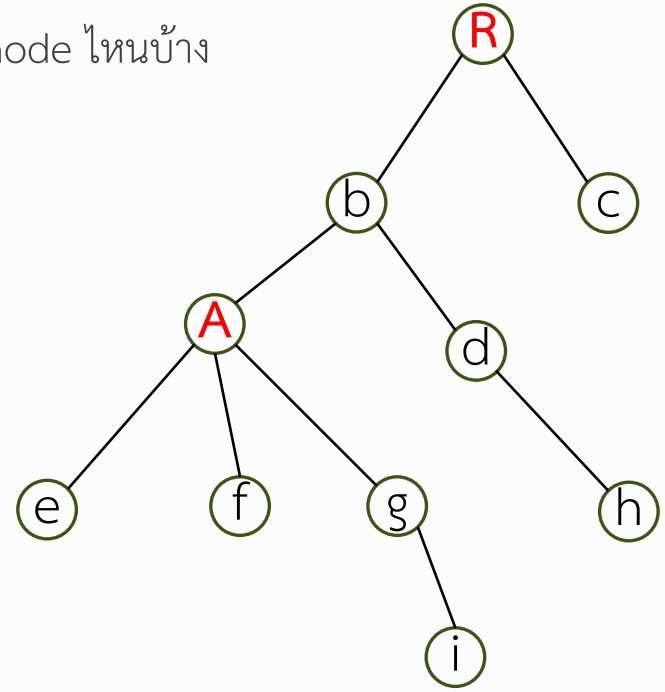
จากต้นไม้ด้านล่างนี้ ถ้าให้ R คือราก อยากทราบว่าต้นไม้แต่ละต้นมี

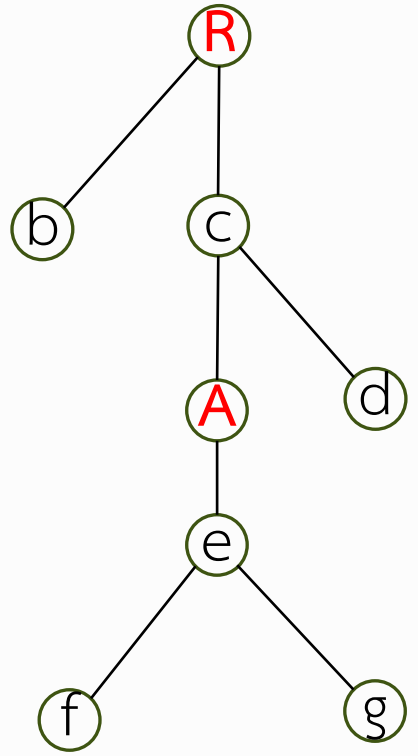
1.ใบ (leaf), 2.ปมภายใน (internal node), และ 3.ความสูง (height) เป็นเท่าไหร่?

ถ้า A คือ node ที่สนใจอยากทราบว่า A นั้นมี

1.ลูก (child), 2.พ่อแม่ (parent), และ 3.พี่น้อง (siblings) เป็น node ไหนบ้าง







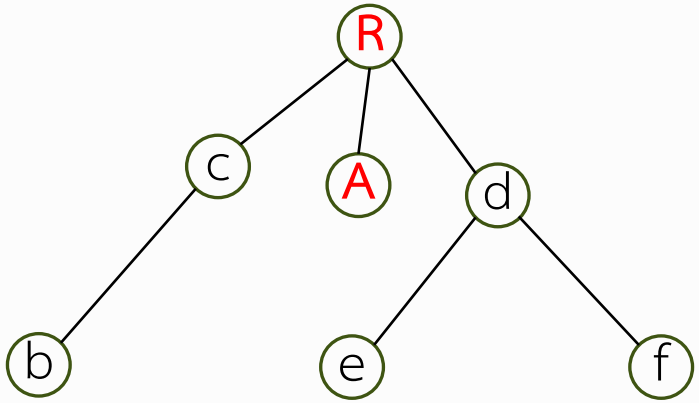
แบบฝึกหัด คำศัพท์ต้นไม้

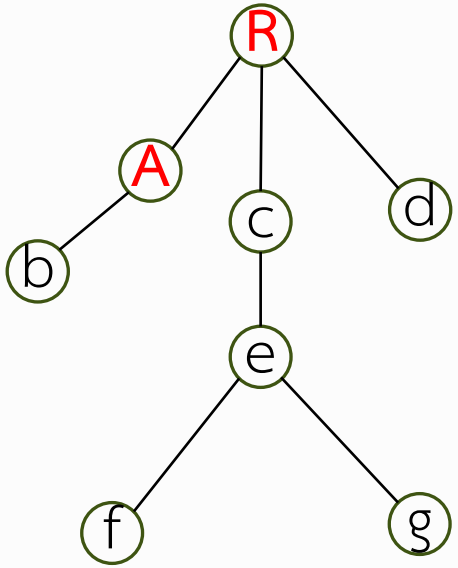
จากต้นไม้ด้านล่างนี้ ถ้าให้ R คือราก อยากทราบว่าต้นไม้แต่ละต้นมี

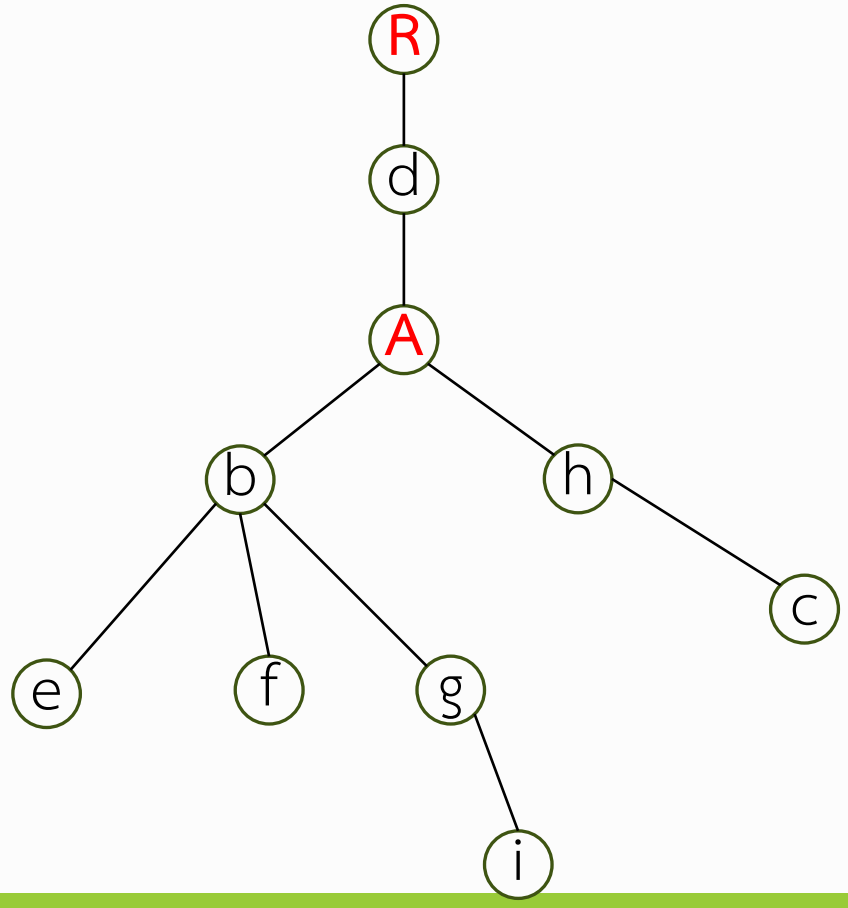
1.ใบ (leaf), 2.ปมภายใน (internal node), และ 3.ความสูง (height) เป็นเท่าไหร่?

ถ้า A คือ node ที่สนใจอยากทราบว่า A นั้นมี

1.ลูก (child), 2.พ่อแม่ (parent), และ 3.พี่น้อง (siblings) เป็น node ไหนบ้าง

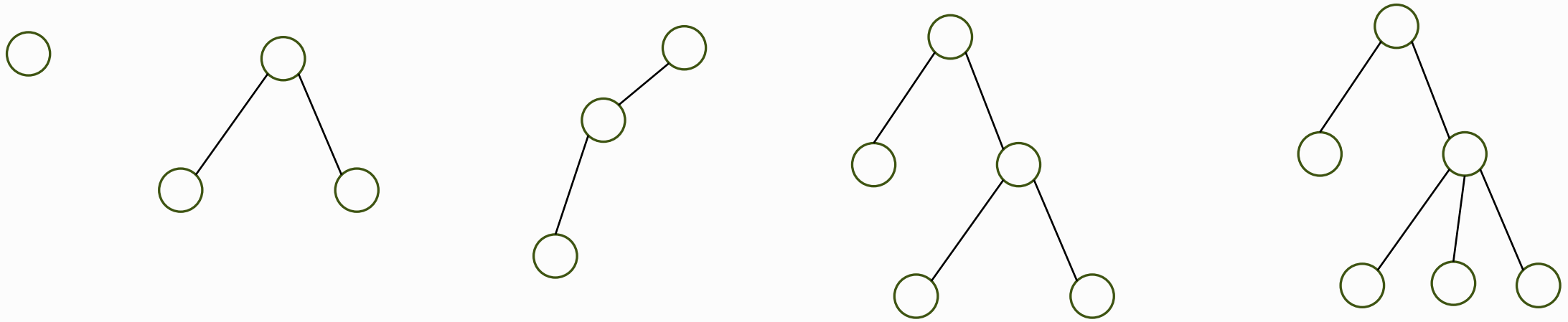






ต้นไม้ทวิภาค, Binary tree

ต้นไม้ทวิภาค คือ ต้นไม้ที่ทุกๆ node จะมีลูกได้มากสุดแค่ 2 nodes เท่านั้น โดยจะเรียกว่าลูกปมซ้าย (left child) และ ลูกปมขวา (right child), อย่างไรก็ตามต้นไม้ ว่าง ก็ถือว่าเป็นต้นไม้ทวิภาค



ไม่เป็น Binary tree

การท่องต้นไม้, Tree traversal

การท่องต้นไม้ทวิภาค (tree traversal) คือ การที่ไปเยี่ยมทุกปมในต้นไม้ การเยี่ยมหมายถึงการดึงข้อมูลหรืออ่านข้อมูลของปมนั้นๆ ซึ่งเป็นการทำงานปกติบนต้นไม้ทวิภาค การท่องต้นไม้จะต้องเริ่มต้นที่รากเสมอ ซึ่งจะมีทางเลือกสองแบบ:

1. การเยี่ยมปม

2. การท่องในต้นไม้ย่อย

โดยการท่องต้นไม้สามารถแบ่งได้อีกทั้งหมด 3 วิธี

1. การท่องต้นไม้ทวิภาคก่อนลำดับ (**pre**order traversal)

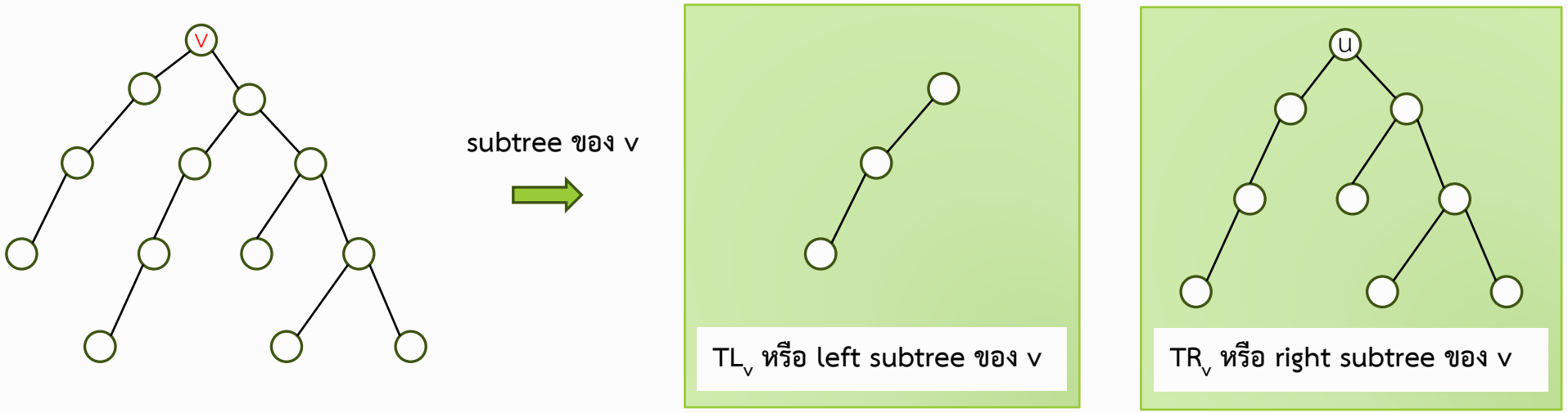
2. การท่องต้นไม้ทวิภาคตามลำดับ (**in**order traversal)

3. การท่องต้นไม้ทวิภาคหลังลำดับ (**post**order traversal)

การท่องต้นไม้ – สัญลักษณ์/คำศัพท์, Tree traversal terms

สัญลักษณ์ที่จะใช้ในการท่องต้นไม้คืออยู่สองสัญลักษณ์หลักๆคือ: TL_v และ TR_v; ถ้าให้ให้ T เป็นต้นไม้ ให้ v เป็นรากของต้นไม้ โดย TL_v หมายถึงต้นไม้ทางซ้าย (left subtree) ของ v และ TR_v หมายถึงต้นไม้ทางขวา (right subtree) ของ v

แล้ว subtree คือ? -> ต้นไม้ที่เป็นส่วนหนึ่งอยู่ภายในต้นไม้อีกที ยกตัวอย่างเช่น:



การท่องต้นไม้ก่อนลำดับ (preorder)

$Trav_p(T)$

Trav -> Traverse

p -> preorder

T ในที่นี้คือ ต้นไม้ T

อัลกอริทึมการท่องต้นไม้ก่อนลำดับจะแทนด้วยสัญลักษณ์ $Trav_p(T)$

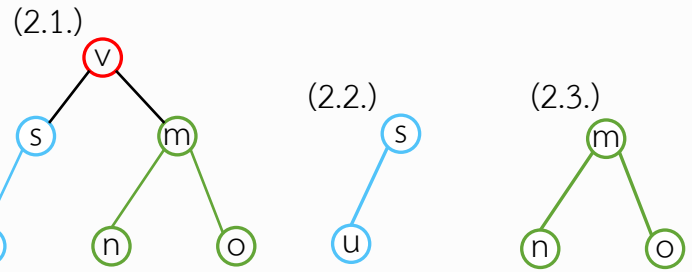
1. ถ้าต้นไม้ T เป็นต้นไม้ว่างจะไม่มีการท่อง

2. ถ้าต้นไม้ T ไม่เป็นต้นไม้ว่างจะทำกระบวนการต่อไปนี้

   2.1 เยี่ยม v

   2.2 $Trav_p(TL_v)$

   2.3 $Trav_p(TR_v)$



คำสั่ง $Trav_p(T)$ จะได้ v, $Trav_p(TL_v)$, $Trav_p(TR_v)$

v, s, $Trav_p(TL_s)$, $Trav_p(TR_s)$,

v, s, u, $Trav_p(TL_u)$, $Trav_p(TR_u)$,

v, s, u,

m, $Trav_p(TL_m)$, $Trav_p(TR_m)$

m, n, $Trav_p(TL_n)$, $Trav_p(TR_n)$, o, $Trav_p(TL_o)$, $Trav_p(TR_o)$

m, n,

o,

ผลลัพธ์: v, s, u, m, n, o

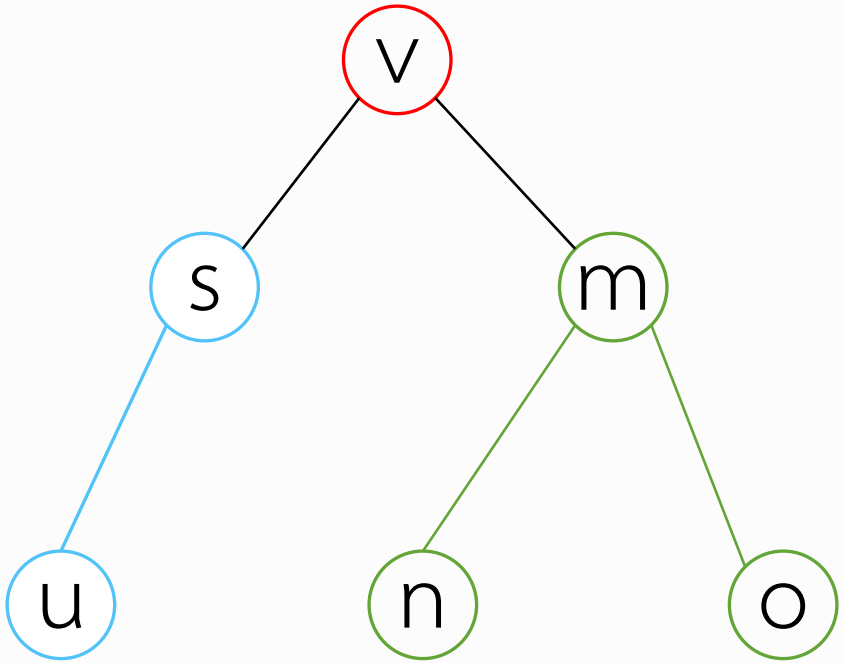
การท่องต้นไม้ก่อนลำดับ (preorder)

$Trav_p(T)$

Trav -> Traverse

p -> preorder

T ในที่นี้คือ ต้นไม้ T



ท่องต้นไม้แบบก่อนลำดับ

v, $Trav_p(TL_v)$, $Trav_p(TR_v)$

v, s, $Trav_p(TL_s)$, $Trav_p(TR_s)$, m, $Trav_p(TL_m)$, $Trav_p(TR_m)$

v, s, u, $Trav_p(TL_u)$, $Trav_p(TR_u)$, m, n, $Trav_p(TL_n)$, $Trav_p(TR_n)$, o, $Trav_p(TL_o)$, $Trav_p(TR_o)$

v, s, u, m, n, o,

ผลลัพธ์: v, s, u, m, n, o

การท่องต้นไม้ตามลำดับ (inorder)

$Trav_i(T)$

Trav -> Traverse

i -> inorder

T ในที่นี้คือ ต้นไม้ T

อัลกอริทึมการท่องต้นไม้ตามลำดับจะแทนด้วยสัญลักษณ์ $Trav_i(T)$

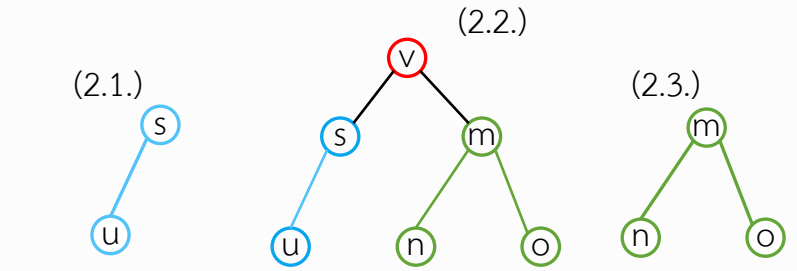
1. ถ้าต้นไม้ T เป็นต้นไม้ว่างจะไม่มีการท่อง

2. ถ้าต้นไม้ T ไม่เป็นต้นไม้ว่างจะทำกระบวนการต่อไปนี้

   2.1 $Trav_i(TL_v)$

   2.2 เยี่ยม v

   2.3 $Trav_i(TR_v)$



คำสั่ง $Trav_i(T)$ จะได้ $Trav_i(TL_v)$, v, $Trav_i(TR_v)$

$Trav_i(TL_s)$, s, $Trav_i(TR_s)$, v, $Trav_i(TL_m)$, m, $Trav_i(TR_m)$

$Trav_i(TL_u)$, u, $Trav_i(TR_u)$, s, v, $Trav_i(TL_n)$, n, $Trav_i(TR_n)$, m, $Trav_i(TL_o)$, o, $Trav_i(TR_o)$

u, s, v, n, m, o,

ผลลัพธ์: u, s, v, n, m, o

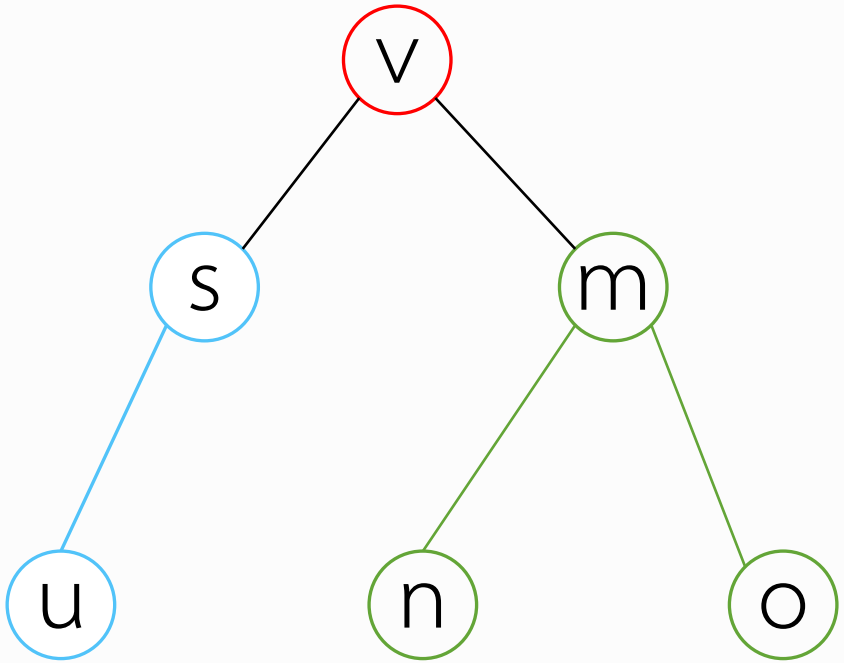
การท่องต้นไม้ตามลำดับ (inorder)

$Trav_i(T)$

Trav -> Traverse

i -> inorder

T ในที่นี้คือ ต้นไม้ T



ท่องต้นไม้แบบตามลำดับ

$Trav_i(TL_v)$, v, $Trav_i(TR_v)$

$Trav_i(TL_s)$, s, $Trav_i(TR_s)$, v, $Trav_i(TL_m)$, m, $Trav_i(TR_m)$

$Trav_i(TL_u)$, u, $Trav_i(TR_u)$, s, v, $Trav_i(TL_n)$, n, $Trav_i(TR_n)$, m, $Trav_i(TL_o)$, o, $Trav_i(TR_o)$

u, s, v, n, m, o,

ผลลัพธ์: u, s, v, n, m, o

การท่องต้นไม้หลังลำดับ (postorder)

$Trav_i(T)$

Trav -> Traverse

i -> inorder

T ในที่นี้คือ ต้นไม้ T

อัลกอริทึมการท่องต้นไม้หลังลำดับจะแทนด้วยสัญลักษณ์ $Trav_o(T)$

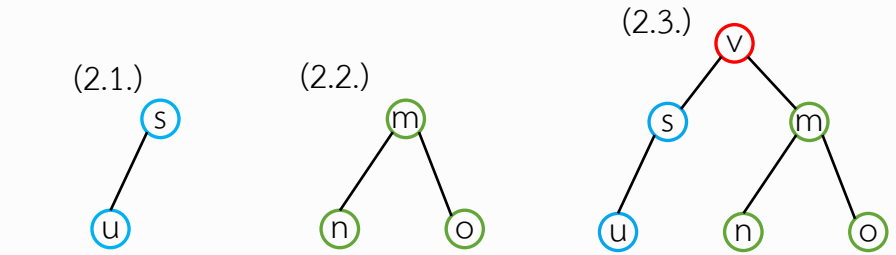
1. ถ้าต้นไม้ T เป็นต้นไม้ว่างจะไม่มีการท่อง

2. ถ้าต้นไม้ T ไม่เป็นต้นไม้ว่างจะทำกระบวนการต่อไปนี้

   2.1 $Trav_o(TL_v)$

   2.2 $Trav_o(TR_v)$

   2.3 เยี่ยม v



คำสั่ง $Trav_i(T)$ จะได้ $Trav_o(TL_v)$, $Trav_o(TR_v)$, v,

$Trav_o(TL_s)$, $Trav_o(TR_s)$, s, $Trav_o(TL_m)$, $Trav_o(TR_m)$, m, v,

$Trav_o(TL_u)$, $Trav_o(TR_u)$, u, s, $Trav_o(TL_n)$, $Trav_o(TR_n)$, n, $Trav_o(TL_o)$, $Trav_o(TR_o)$, o, m, v,

u, s, n, o, m, v,

ผลลัพธ์: u, s, n, o, m, v

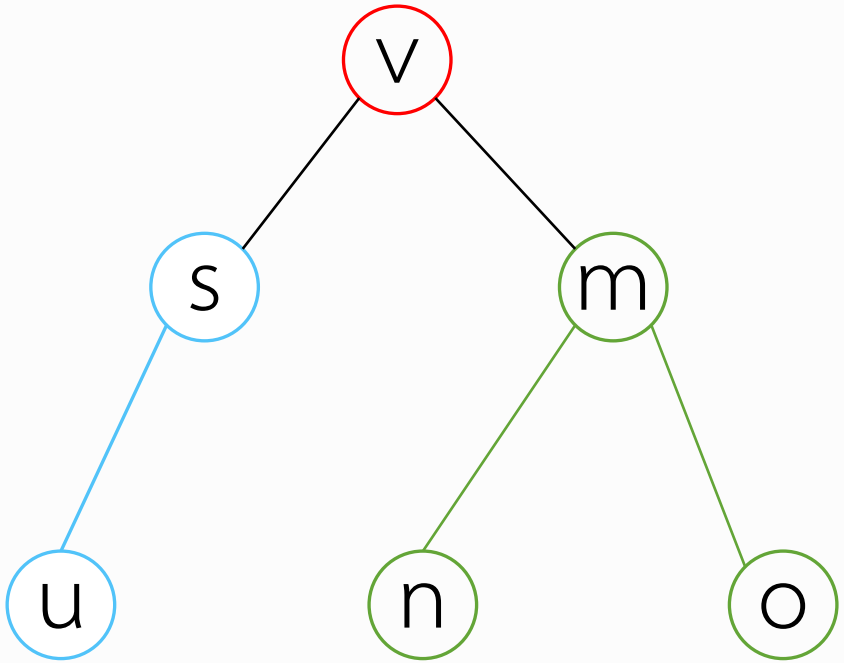
การท่องต้นไม้หลังลำดับ (postorder)

$Trav_i(T)$

Trav -> Traverse

i -> inorder

T ในที่นี้คือ ต้นไม้ T



ท่องต้นไม้แบบหลังลำดับ

$Trav_o(TL_v)$, $Trav_o(TR_v)$, v,

$Trav_o(TL_s)$, $Trav_o(TR_s)$, s, $Trav_o(TL_m)$, $Trav_o(TR_m)$, m, v,

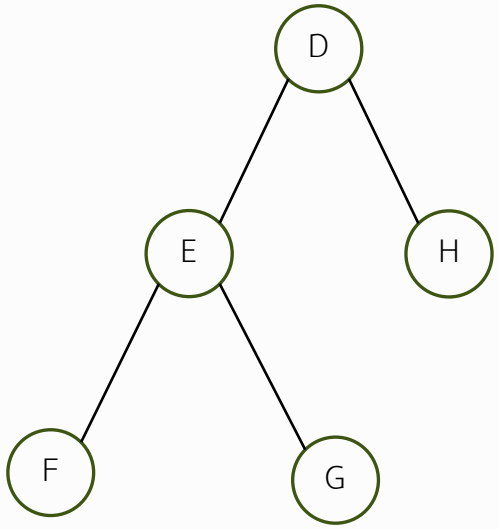
$Trav_o(TL_u)$, $Trav_o(TR_u)$, u, s, $Trav_o(TL_n)$, $Trav_o(TR_n)$, n, $Trav_o(TL_o)$, $Trav_o(TR_o)$, o, m, v,

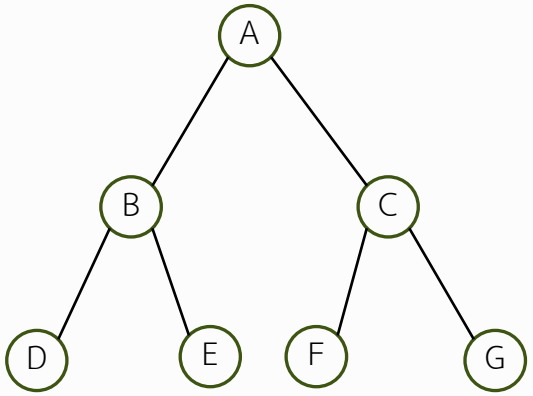
u, s, n, o, m, v,

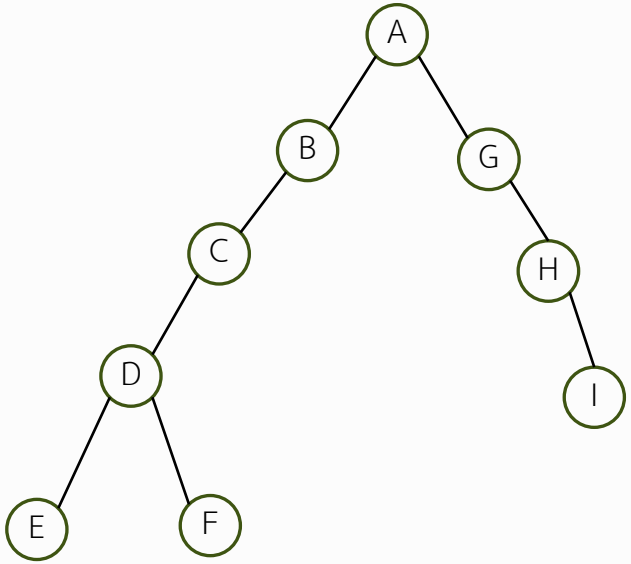
ผลลัพธ์: u, s, n, o, m, v

แบบฝึกหัด การท่องต้นไม้ Tree traversal

จงท่องต้นไม้ต่อไปนี้ทั้งสามวิธี 1.ก่อนลำดับ, 2. ตามลำดับ, และ 3. หลังลำดับ







การประยุกต์ใช้ของการท่องต้นไม้ – Tree traversal applications

Hierarchical File Systems

- การประยุกต์ใช้: ใช้ในการ จัดเก็บ, จัดระเบียบ, หรือ เดินทาง ผ่านโครงสร้างที่เป็นลำดับชั้นของ ไฟล์ และ โฟลเดอร์

Network Routing Algorithms

- การประยุกต์ใช้: ใช้ในการ วาด หรือ จัดระเบียบ โครงสร้างของโครงข่ายการเชื่อมต่อ (internet & telecommunication)

Graph Algorithms

- การประยุกต์ใช้: จัดระเบียบให้อยู่ในรูปแบบต้นไม้ จะสามารถใช้การท่องต้นไม้เป็นอัลกอริทึมในการค้นหา เช่น depth-first search (DFS) and breadth-first search (BFS).

ค้นหา – Searching

การค้นหาในกราฟเป็นการท่องเข้าไปในกราฟ โดยเป็นการเข้าเยี่ยมแต่ละจุดยอดอย่างมีระบบ โดยทั่วไปจะมีการค้นหาอยู่สองแบบ คือ

1. การค้นหาแนวกว้าง หรือ Breath-First Search (BFS)

2. การค้นหาแนวลึก หรือ Depth-First Search (DFS)

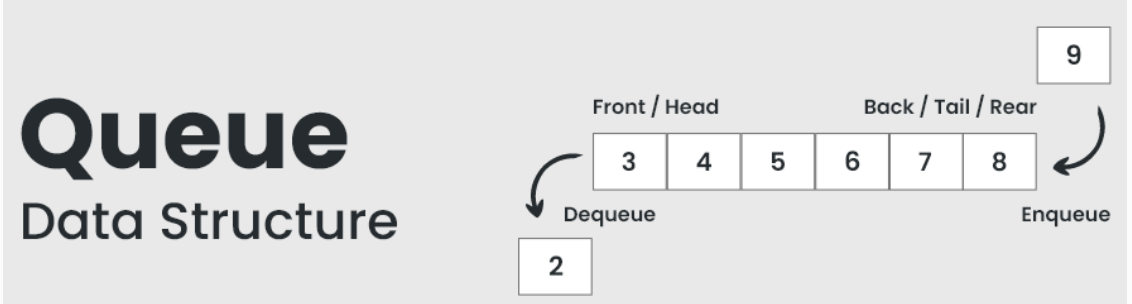
การประยุกต์ใช้หลักๆคือการค้นหาต้นไม้ที่ใหญ่มากๆเช่น

- Searching directories เพื่อหาไฟล์แค่ไฟล์เดียว
- ค้นหาข้อมูลบุคลากรชื่อ/รหัสนั้นๆในบริษัทยักษ์ใหญ่

การค้นหาแนวกว้าง - Breath-First Search (BFS)

การค้นหาแนวกว้างมีลักษณะคล้ายกับลำดับในองค์กรณ์ นั่นคือทำกาสรค้นหาที่ละชั้น ไปเรื่อยๆจนหมด ซึ่งมีขั้นตอนการทำงานดังนี้

1. เยี่ยมปมเริ่มต้น
2. ใส่ปมเริ่มต้นในโครงสร้างข้อมูลแบบคิว (queue)
3. ถ้าในคิวยังมีข้อมูลอยู่ให้ทำ ข้อ 3.1 และ ข้อ 3.2
    3.1 นำปมที่อยู่ในโครงสร้างข้อมูลแบบคิวออกมาใส่ในตัวแปรชื่อ x
    3.2 สำหรับทุกปม y ที่มีเส้นเชื่อมไปยังปม x ทำ
        3.2.1 ถ้า y ยังไม่ได้ถูกเยี่ยม ใส่ปม y ลงไปในโครงสร้างข้อมูลแบบคิว

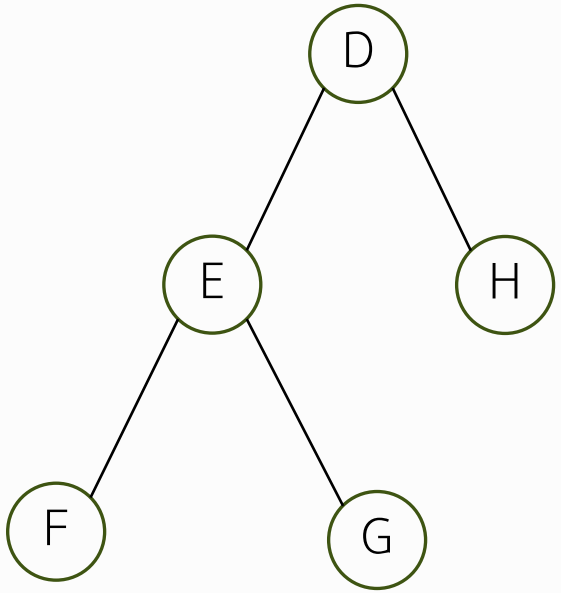


Src: Queue Data Structure - GeeksforGeeks

เว็บไซต์แนะนำ: Depth First Search or DFS for a Graph - GeeksforGeeks

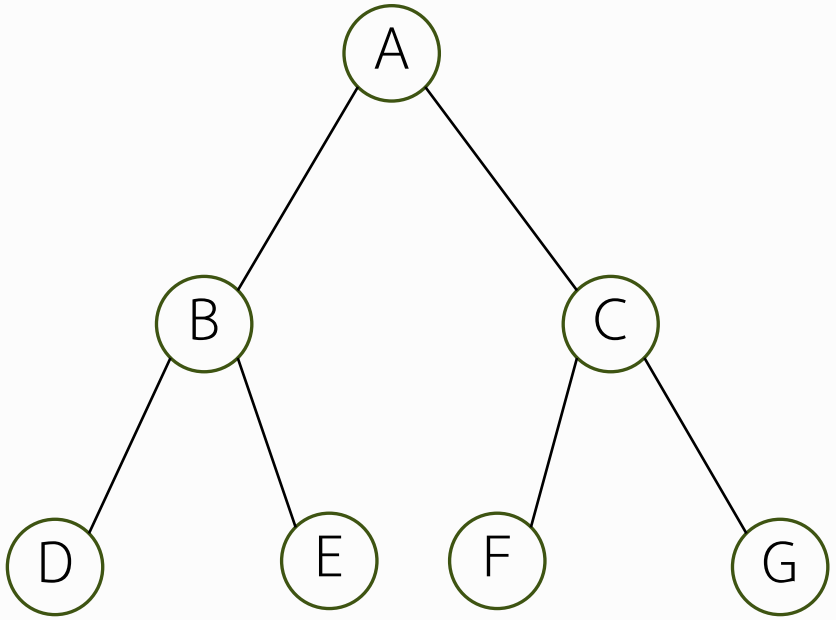
การค้นหาแนวกว้าง - Breath-First Search (BFS)

จงแสดงวิธีการค้นหาต้นไม้ โดยใช้วิธีการค้นหาแนวกว้าง (Breath-First Search)



การค้นหาแนวกว้าง - Breath-First Search (BFS)

จงแสดงวิธีการค้นหาต้นไม้ โดยใช้วิธีการค้นหาแนวกว้าง (Breath-First Search)



การค้นหาแนวลึก หรือ Depth-First Search (DFS)

การค้นหาแบบแนวกว้างจะมีลักษณะของการค้นหาแบบฟังก์ชันเรียกตัวเอง (recursive function) ซึ่งสามารถอธิบายด้วยอัลกิริทึมที่แทนด้วย ds(x) โดยที่ x คือปม

1. เยี่ยมปม x

2. สำหรับทุกปม y ที่มีเส้นเชื่อมไปยังปม x ทำ

    2.1 ถ้า y ยังไม่ได้ถูกเยี่ยมให้ทำ ds(y)

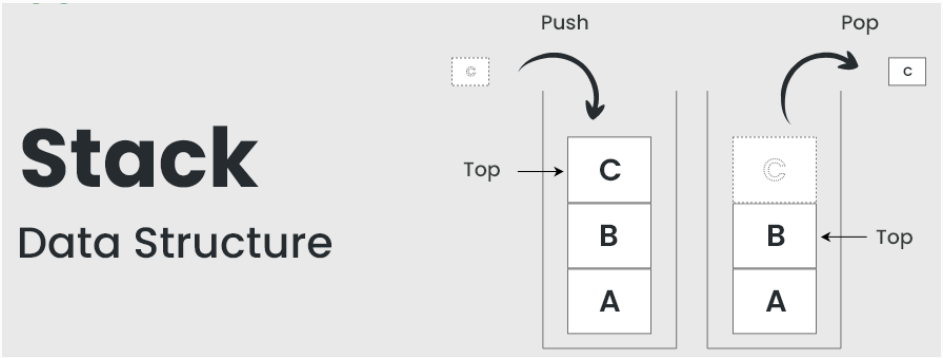
สำหรับการเรียกฟังก์ชัน ds(x) เมื่อไม่มี y ที่เป็นปมอื่นๆที่มีเส้นเชื่อมกับ x ที่ต้องเยี่ยมแล้วฟังก์ชัน ds(x) ทำงานเสร็จสิ้น ก็จะหยุดแล้วย้อนกลับมายังปม (v) ที่เรียกมันที่ปม v จะพิจารณาปมอื่นๆที่มีเส้นเชื่อมกับปม v หากมีปมที่ยังไม่เยี่ยมก็เรียกฟังก์ชัน ds มาทำงานต่อ หากไม่มีปมที่ต้องไปเยี่ยมแล้ว ก็จะจบการทำงานของ ds(v) แล้วย้อนกลับมายังปมที่เรียกมันเหมือนกันซึ่งการจบการทำงานของ ds ที่ถูกเรียกจะย้อนกลับส่วนที่เรียกมัน จนกระทั่งกลับไปยังตัวแรกที่เรียก จึงจะเป็นการจบการทำงานของการค้นหาแนวลึกที่เป็นลักษณะของฟังก์ชันเรียกตัวเอง ซึ่งการจบฟังก์ชันแล้วย้อนกลับไปฟังก์ชันที่เรียกมันนั้น ในกรณีนี้มีชื่อเรียกว่าการย้อนรอย (backtracking)

การค้นหาแนวลึก หรือ Depth-First Search (DFS) – เชิง Computing

1. เริ่มโดยการใส่จุดยอดเริ่มต้นเข้าไปในโครงสร้างข้อมูลแบบ สแต็ก (stack)

2. นำจุดยอดที่อยู่บนสุดของ stack ใส่เข้าไปใน (array)

visited list

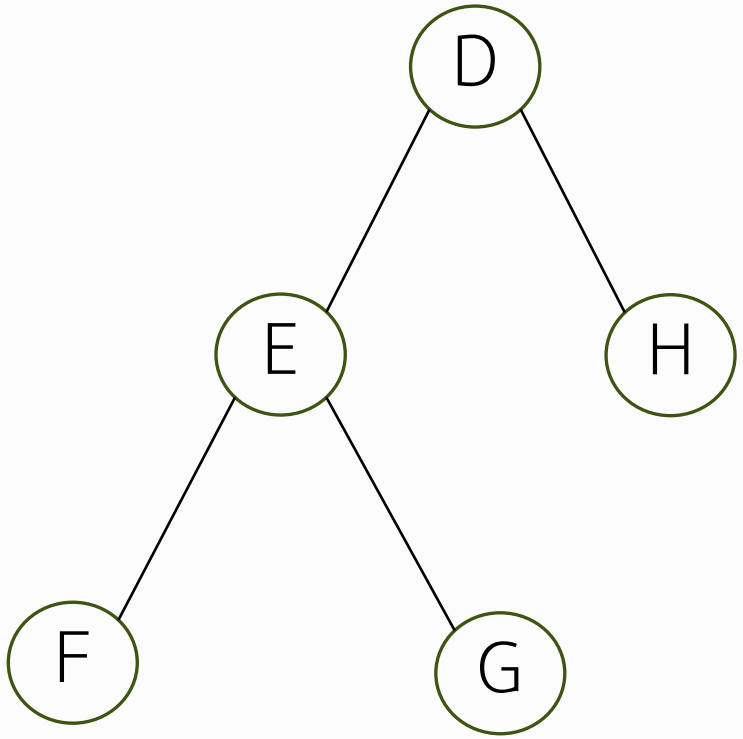


3. สร้างลิสของจุดยอดที่เชื่อมกับจุดยอดในข้อ 2. และเพิ่มจุดยอดที่ไม่อยู่ใน visited list เข้าไปด้านบนของ stack

4. ทำขั้นตอนที่ 2 และ 3 ไปเรื่อยไปจนกว่า stack นั้นจะว่าง

แนะนำให้ไปดูเว็บ: Depth First Search or DFS for a Graph - GeeksforGeeks

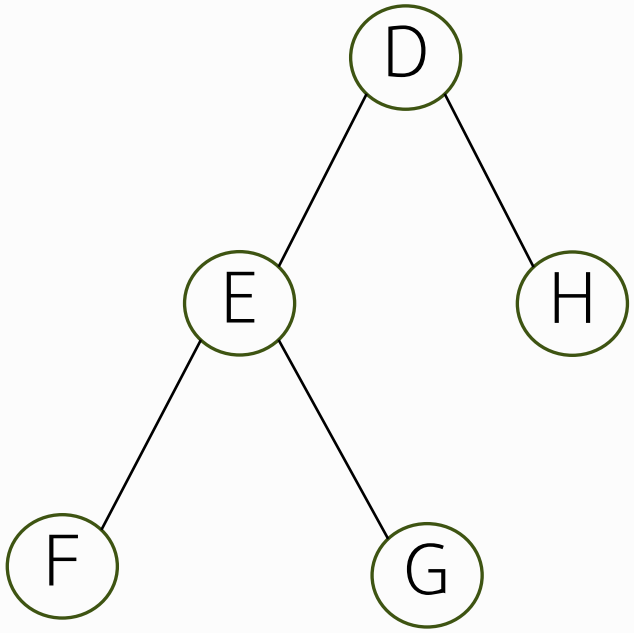
การค้นหาแนวลึก หรือ Depth-First Search (DFS)




```
DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)
init() {
    For each u ∈ G
        u.visited = false
     For each u ∈ G
       DFS(G, u)
}
```

การค้นหาแนวลึก หรือ Depth-First Search (DFS)

จงแสดงวิธีการค้นหาต้นไม้ โดยใช้วิธีการค้นหาแนวลึก (Depth-First Search)



การค้นหาแนวลึก หรือ Depth-First Search (DFS)

จงแสดงวิธีการค้นหาต้นไม้ โดยใช้วิธีการค้นหาแนวลึก (Depth-First Search)