บทที่ 7 ต้นไม้
(Trees)

7.1 **ต้นไม้เบื้องต้น** (Introduction to Trees)

ต้นไม้ครอบครัว หมายถึง กราฟ ซึ่ง จุด แทน สมาชิกของครอบครัว และ ด้าน แทน ความสัมพันธ์ ของ พ่อแม่ กับ ลูก

(A **family tree** is a graph where the vertices represent family members and edges represent parent-child relationships.)

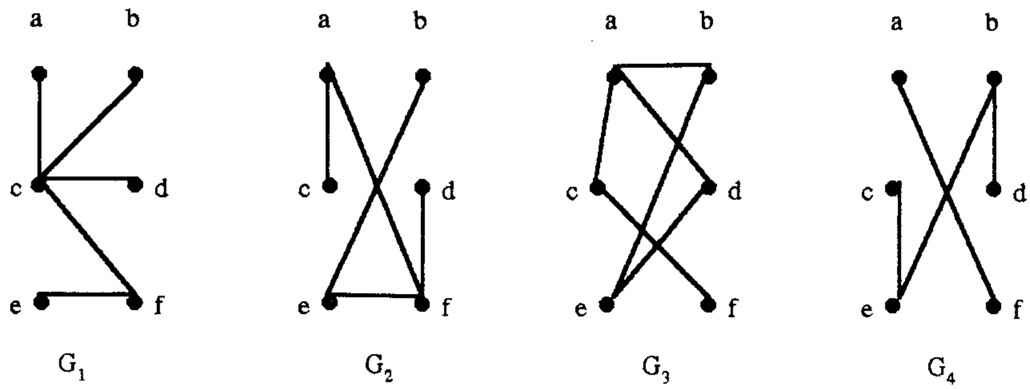
กราฟแบบไม่มีทิศทาง ซึ่งแทน ผังสายพันธุ์ (genealogical chart) เป็น ตัวอย่างหนึ่ง ของ กราฟชนิดพิเศษ ซึ่งเรียกว่า **ต้นไม้** (tree)

บทนิยาม 1 **ต้นไม้** หมายถึง กราฟไม่มีทิศทาง แบบไม่ขาดตอน และ ไม่มีวงจรเชิงเดียว

(A **tree** is a connected undirected graph with no simple circuit.) [1]

เนื่องจาก ต้นไม้ ไม่มีวงจรเชิงเดียว ต้นไม้ ไม่มีด้านหลายเส้น หรือ ไม่มีรูปบ่วง เพราะฉะนั้น ต้นไม้ใดๆ ต้องเป็นกราฟเชิงเดียว

ตัวอย่าง 1 กราฟที่แสดงในรูปที่ 1 ชุดใดบ้างเป็นต้นไม้



รูปที่ 1 G_1 และ G_2 เป็นต้นไม้
G_3 และ G_4 ไม่ใช่ต้นไม้

ผลเฉลย กราฟ G_1 และ G_2 เป็นต้นไม้ เพราะว่าทั้งคู่ เป็นกราฟไม่ขาดตอน และ ไม่มีวงจรเชิงเดียว

---

[1] Rosen, หน้า 505

กราฟ G_3 ไม่ใช่ต้นไม้ เพราะว่า e, b, a, d, e เป็นวงจรเชิงเดียว ในกราฟสุดท้าย กราฟ G_4 ไม่ใช่ต้นไม้ เพราะว่า มันไม่ใช่กราฟแบบ ไม่ขาดตอน

กราฟ ไม่ขาดตอน ใดๆ ซึ่ง ไม่มีวงจรเชิงเดียว คือ **ต้นไม้**

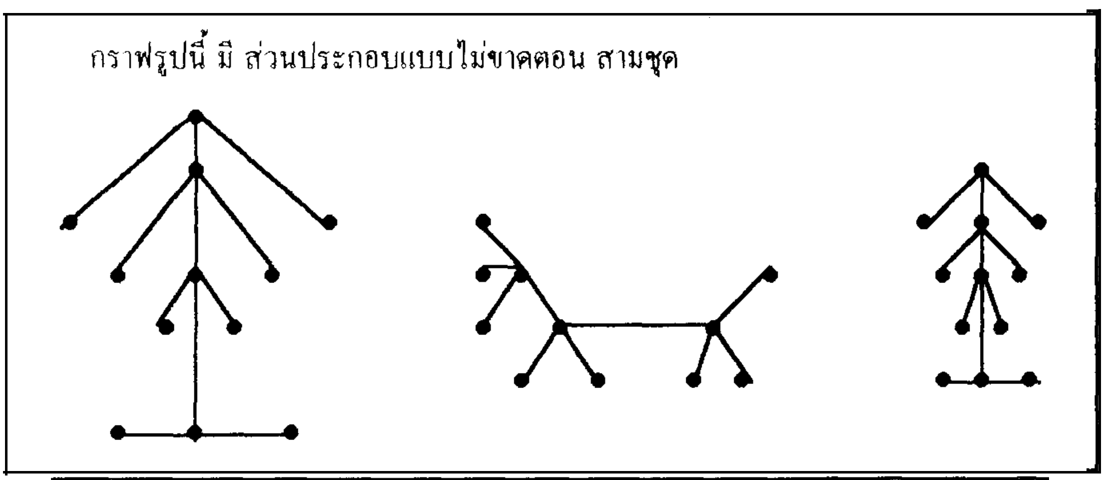
กราฟต่างๆ ซึ่ง ไม่มีวงจรเชิงเดียว และ ไม่จำเป็นต้องเป็น กราฟ ไม่ขาดตอน กราฟเหล่านี้ เรียกว่า **ป่า** (forest) และมีคุณสมบัติว่า ส่วนประกอบแบบไม่ขาดตอนแต่ละชุดนี้ คือ ต้นไม้ รูปที่ 3 แสดงให้เห็น ป่า บางชนิด

ต้นไม้เหล่านี้ บ่อยครั้ง นิยาม เป็น กราฟแบบ ไม่มีทิศทาง โดยมีคุณสมบัติว่า มี ทางเดินเชิงเดียวเพียงหนึ่งทางเท่านั้น ระหว่างแต่ละคู่ของจุด

ทฤษฎีบท 1 กราฟแบบไม่มีทิศทาง จะเป็นต้นไม้ ก็ต่อเมื่อ มีทางเดินเชิงเดียวเพียงหนึ่งทางเท่านั้น ระหว่างแต่ละสองจุดใดๆ ของมัน

(An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.)

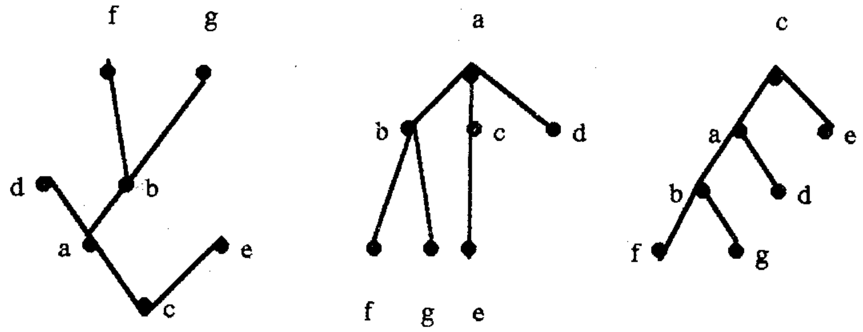
ในการประยุกต์ จำนวนมาก ของต้นไม้ จุดพิเศษหนึ่งจุด ของต้นไม้ ถูกกำหนด ให้เป็น **ราก** (root) เมื่อกำหนดรากแล้ว เราสามารถกำหนด ทิศทาง ให้กับแต่ละด้าน ที่ตามมา เนื่องจากมีทางเดินเพียงหนึ่งทางเท่านั้น จาก ราก ไปยังแต่ละ จุด ของ กราฟ เราชี้ทิศทางด้านแต่ละด้านออกจากราก ดังนั้น ต้นไม้รวมกับ รากของมัน จึงเกิดเป็น กราฟแบบมีทิศทาง เรียกว่า **ต้นไม้ราก** (rooted tree) เราสามารถ เปลี่ยนแปลง ต้นไม้แบบไม่มีราก (unrooted tree) ให้เป็นต้นไม้รากได้



รูปที่ 2 ตัวอย่างของ ป่า หนึ่งแห่ง

โดยการเลือก จุดใด จุดหนึ่ง เป็นราก โปรดสังเกตว่า การเลือกที่แตกต่างกัน ของราก จะทำให้ได้ ต้นไม้รากที่แตกต่างกัน ตัวอย่างเช่น ในรูปที่ 3 แสดงให้เห็น ต้นไม้ราก ซึ่งเกิดจากการกำหนดให้ a เป็นราก และ กำหนดให้ c เป็นราก ตามลำดับ ใน ต้นไม้ T

ปกติเราวาดรูปต้นไม้ราก โดยให้รากของมัน อยู่ตอนบนสุดของต้นไม้ ลูกศร แสดงทิศทาง ของด้าน ใน ต้นไม้ราก ซึ่งจะไม่ใส่ก็ได้ เพราะว่า การเลือกจุดราก จะบอกทิศทาง ของด้านต่างๆ



T a เป็นราก c เป็นราก

รูปที่ 3 ต้นไม้ และ ต้นไม้ราก ซึ่งกำหนดราก 2 ชุด

การใช้คำศัพท์ สำหรับต้นไม้ มีการเริ่มต้นเชิงสายพันธุ์ และ พฤกษศาสตร์

สมมติให้ T เป็นต้นไม้ราก ชุดหนึ่ง ถ้า v เป็นจุดหนึ่ง ใน T ซึ่งไม่ใช่จุดราก **parent** ของ v คือ จุด u เพียงหนึ่งจุดเท่านั้น โดยที่ มี ด้าน มีทิศทางจาก u ไป v

เมื่อ u เป็น **จุดแม่** (parent) ของ v, เราเรียก v ว่า **ลูก** (child) ของ u

จุดต่างๆ ซึ่งเกิดจาก parent เดียวกัน เรียกว่า **พี่น้องกัน** (siblings)

จุดบน (ancestors) ของ จุดหนึ่ง ซึ่งไม่ใช่ราก หมายถึง จุดต่างๆ จาก ราก ไปยัง จุดนี้ ไม่นับ จุดตัวมันเอง และ นับรวมรากด้วย

จุดล่าง (descendants) ของจุด v หมายถึง จุดต่างๆ ซึ่งมี v เป็นจุดบน

จุดใบ (leaf) หมายถึง จุด ใน ต้นไม้ ซึ่ง ตัวมันเองไม่มีลูก

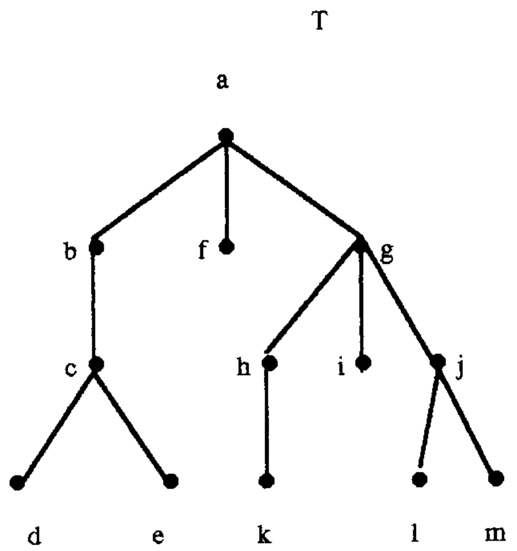
จุดซึ่งมีลูก เรียกว่า **จุดภายใน** (internal vertices)

ราก เป็น จุดภายใน ยกเว้น ถ้ามันเป็นเพียงจุดเดียวเท่านั้นในต้นไม้ กรณีนี้ ราก คือ ใบ

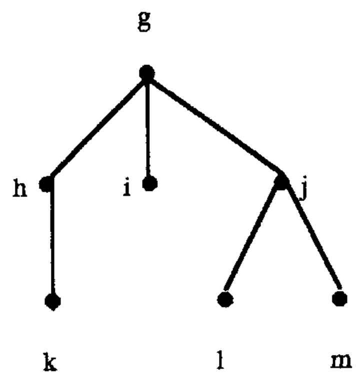
ถ้า a เป็นจุดหนึ่ง ในต้นไม้ **ต้นไม้ส่วนย่อย** (subtree) ที่มี a เป็น ราก หมายถึง กราฟ

ย่อย (subgraph) ของต้นไม้ ประกอบด้วย จุด a และ จุดล่างของ a และ ด้านทั้งหมด ซึ่ง ตกกระทบ กับ จุดล่างเหล่านี้

ตัวอย่าง 2 ในต้นไม้ราก T (มี a เป็นราก) ในรูปที่ 4 จงหา จุดแม่ ของ c, ลูก ของ g, พี่น้อง ของ h และ จุดบนทั้งหมด ของ e, จุดล่างทั้งหมด ของ b, จุดภายในทั้งหมด และ จุดใบทั้งหมด จากนั้น อะไรคือ ต้นไม้ส่วนย่อย ที่มีรากเป็น g



รูปที่ 4 ต้นไม้ราก T



รูปที่ 5 ต้นไม้ส่วนย่อย
มีจุด g เป็นราก

ผลเฉลย

parent ของ c คือ b

children ของ g คือ h, i, j

siblings ของ h คือ i และ j

ancestors ของ e คือ c, b, a

descendants ของ b คือ c, d, e

internal vertices คือ a, b, c, g, h, j

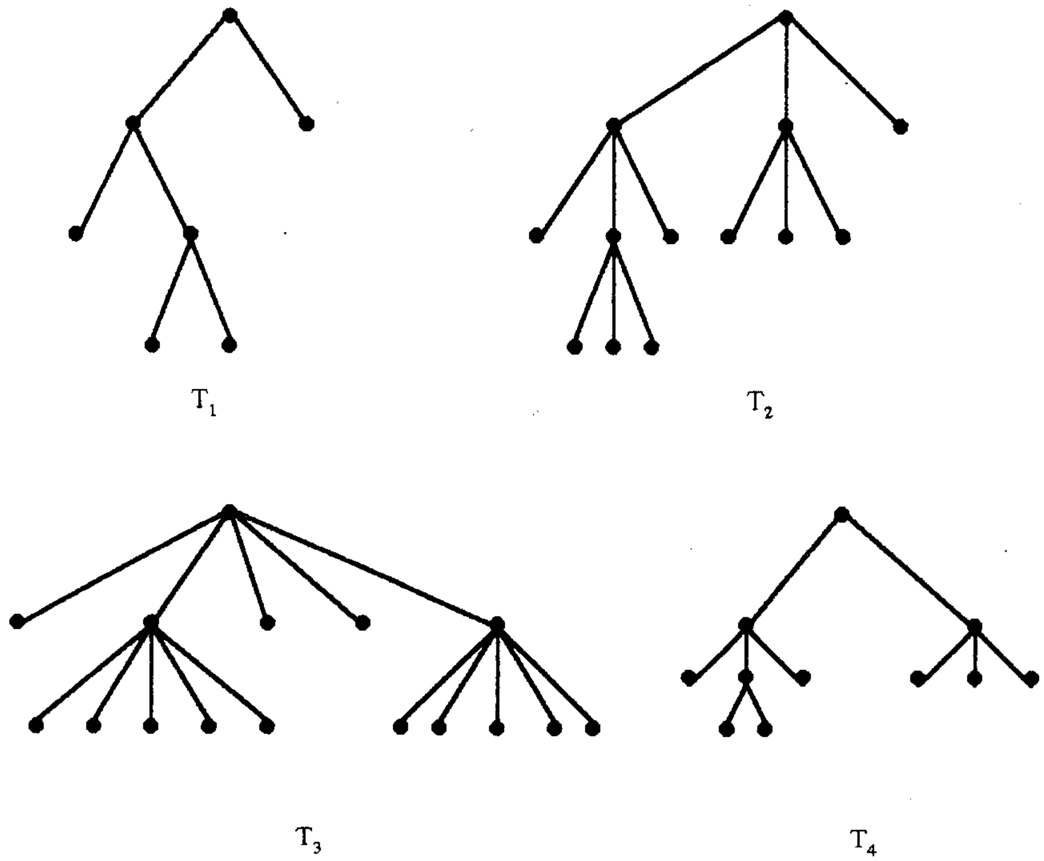
leave คือ d, e, f, i, k, l, m

ต้นไม้ส่วนย่อย มี g เป็นราก คือ รูปที่ 5

ต้นไม้ราก ซึ่งมีคุณสมบัติว่า จุดภายในทั้งหมด มี จำนวนลูกเท่ากัน ถูกนำมาใช้ ใน การประยุกต์ที่แตกต่างกัน มากมาย ในบทนี้ เราจะใช้ต้นไม้เช่นนี้ มาศึกษา ปัญหาซึ่งเกี่ยวกับ การค้น (searching), การเรียงลำดับ (sorting) และ การเข้ารหัส (coding)

บทนิยาม 2 ต้นไม้ราก จะเรียกว่า **ต้นไม้แบบ m-ary** ถ้าจุดภายใน ทุกจุด มีลูก ไม่มากกว่าจำนวน m ต้นไม้ จะเรียกว่า ต้นไม้ m-ary แบบเต็มต้น (full m-ary tree) ถ้าจุดภายในทุกจุด มีลูกเท่ากับจำนวน m ต้นไม้ m-ary ซึ่งมี m = 2 จะเรียกว่า **ต้นไม้แบบทวิภาค** (binary tree)

ตัวอย่าง 3 ต้นไม้ราก ในรูปที่ 6 เป็น ต้นไม้ m-ary แบบเต็มต้น สำหรับ จำนวนเต็ม บวก m บางตัว หรือไม่?



รูปที่ 6 ต้นไม้ราก สี่ ชุด

ผลเฉลย T_1 เป็น ต้นไม้ทวิภาคแบบเต็มต้น (full binary tree) เพราะว่า จุดภายในทุกจุด มีลูกเท่ากับสอง

T_2 เป็น full 3-ary tree เพราะว่า จุดภายในทุกจุด มีลูกเท่ากับ 3

T_3 เป็น full 5-ary tree เพราะว่า จุดภายในทุกจุด มีลูกเท่ากับ 5

T_4 ไม่ใช่ full m-ary tree สำหรับค่า m ใดๆ เพราะว่า จุดภายใน บางจุด มีลูกเท่ากับสอง จุดภายในบางจุด มีลูกเท่ากับ สาม

ต้นไม้รากแบบอันดับ (Ordered rooted tree)

หมายถึง ต้นไม้ราก ซึ่ง ลูกๆ ของจุดภายใน แต่ละจุด มีการเรียงอันดับ ต้นไม้รากแบบอันดับ ถูกเลือกขึ้นมา เพื่อให้ลูกๆ ของจุดภายใน แต่ละจุด แสดง ให้เห็น ในลำดับ จากซ้ายไปขวา โปรดสังเกตว่า การแทนที่ ของ ต้นไม้ราก ใน วิธีที่ดี จะบอก การเรียงอันดับเช่นนี้ สำหรับด้านต่างๆ ของมัน เราจะใช้ การเรียงอันดับเช่นนี้ ของด้าน ในการวาดรูป โดยไม่กล่าวถึงอย่างชัดแจ้ง ว่า เรากำลังพิจารณา ต้นไม้รากแบบอันดับ

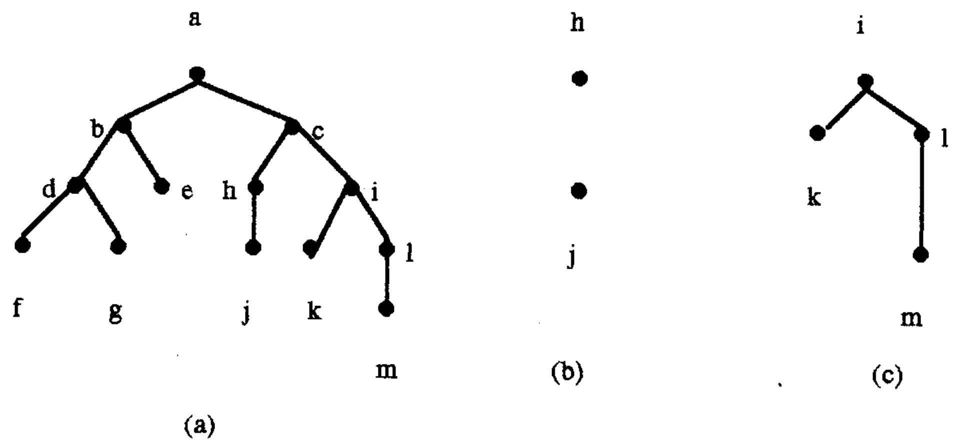
ในต้นไม้ทวิภาคแบบอันดับ (ordered binary tree) ถ้าจุดภายใน มีลูกสองคน ลูกคนแรก เรียกว่า ลูกทางซ้าย (left child) และ ลูกคนที่สอง เรียกว่า ลูกทางขวา (right child) ต้นไม้ ซึ่งมีรากเป็น ลูกทางซ้าย เรียกว่า **ต้นไม้ส่วนย่อยซ้าย** (left subtree) ของจุดนี้ และ ต้นไม้ซึ่งมีราก เป็นลูกทางขวา ของจุด เรียกว่า **ต้นไม้ส่วนย่อยขวา** (right subtree) ของจุด

ผู้อ่าน ควรมีข้อสังเกตว่า งานประยุกต์บางอย่าง ทุกจุดของ ต้นไม้แบบทวิภาค ไม่ใช่จุดราก ถูกกำหนดให้เป็น ลูกทางขวา หรือ ลูกทางซ้ายของ จุดแม่ ของมัน สิ่งนี้ กระทำขึ้นแม้กระทั่งเมื่อ บางจุด มีลูกเพียง หนึ่งจุด

ตัวอย่าง 4 ในต้นไม้แบบทวิภาค T ในรูปที่ 7 (a) อะไรคือ ลูกทางซ้าย และ ลูกทางขวา ของ d ? และ อะไรคือ ต้นไม้ส่วนย่อยซ้าย และ ต้นไม้ส่วนย่อยขวา ของจุด c ?

ผลเฉลย

ลูกทางซ้าย ของ d คือ f และลูกทางขวา ของ d คือ g ส่วนต้นไม้ส่วนย่อยซ้าย และต้นไม้ส่วนย่อยขวา ของ c แสดงให้เห็นในรูป 7 (b) และ รูป 7 (c) ตามลำดับ

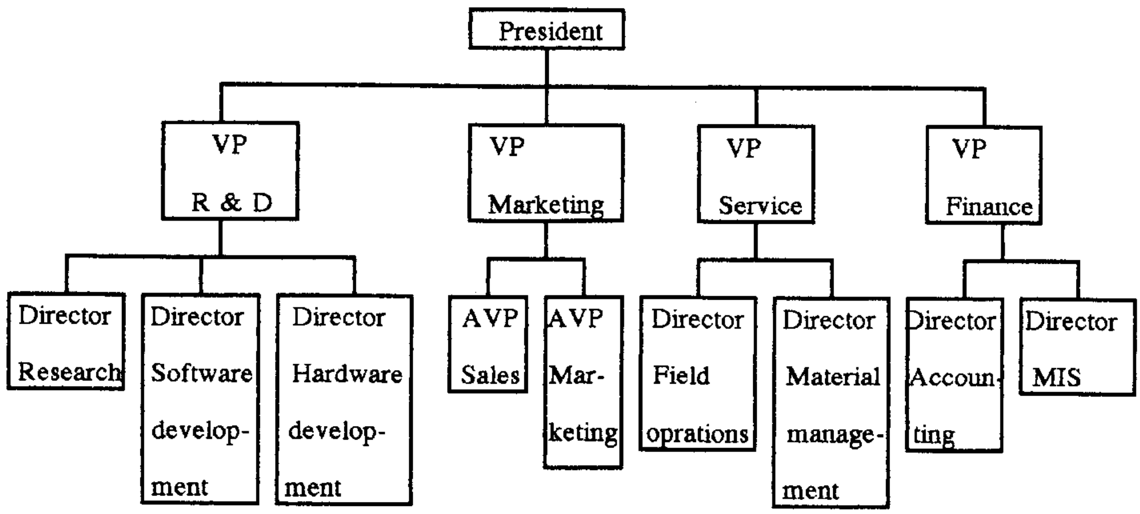


รูปที่ 7 ต้นไม้ แบบทวิภาค T และ ต้นไม้ส่วนย่อยซ้าย และ ต้นไม้ส่วนย่อยขวา ของจุด c

ต้นไม้ คือตัวแบบ (Trees as models)

ต้นไม้ ถูกนำมาใช้เป็นตัวแบบ ในสาขาวิชาต่างๆ เช่น วิทยาการคอมพิวเตอร์ เคมี ธรณีวิทยา พฤกษศาสตร์ และ จิตวิทยา

ตัวอย่าง 5 การแทนที่องค์กร (representing organizations) โครงสร้างของ องค์กรขนาดใหญ่ สามารถทำเป็นตัวแบบโดยใช้ ต้นไม้ราก แต่ละจุด ใน ต้นไม้นี้ แทน หนึ่งตำแหน่ง ใน องค์กร ด้าน จาก จุดหนึ่ง ไปยัง อีกจุดหนึ่ง แสดงว่า บุคคล ซึ่ง แทนโดย จุดแรก (initial vertex) เป็น หัวหน้า (boss) ของ บุคคล ซึ่งแทนด้วย จุดปลาย (terminal vertex) กราฟในรูปที่ 8 แสดงถึง ต้นไม้เช่นนี้



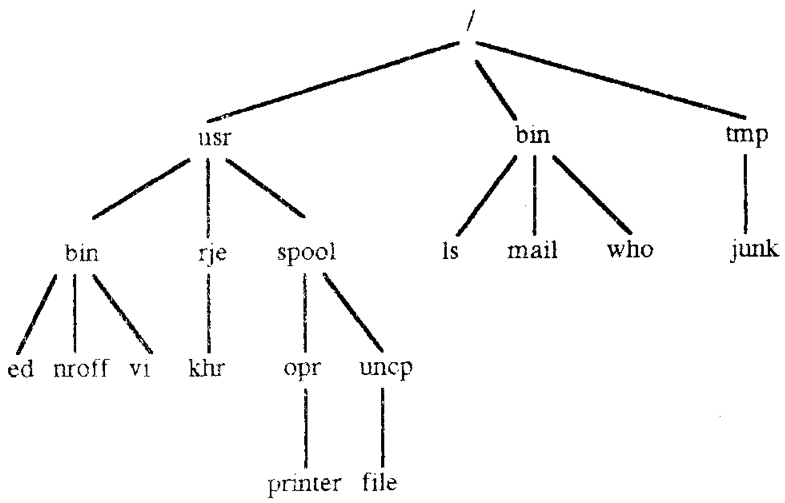
รูปที่ 8 ต้นไม้การจัดองค์กร สำหรับบริษัทคอมพิวเตอร์

ตัวอย่าง 6 ระบบแฟ้มของคอมพิวเตอร์ (Computer File Systems)

แฟ้มต่างๆ ใน หน่วยความจำของ คอมพิวเตอร์ ถูกจัดระเบียบ ให้เป็น ไดเรกทอรี (directories)

หนึ่งไดเรกทอรี อาจจะประกอบด้วย แฟ้ม (files) และ ไดเรกทอรีย่อย (subdirectories) ทั้งคู่

ไดเรกทอรีราก (root directory) ประกอบด้วย ระบบแฟ้มทั้งหมด ดังนั้น ระบบแฟ้ม จึงอาจแทนด้วย ต้นไม้ราก หนึ่งต้น เมื่อ จุดราก แทน ไดเรกทอรีราก จุดภายใน แทน ไดเรกทอรีย่อย และ ใบ แทน แฟ้มปกติ (ordinary file) หรือ ไดเรกทอรีว่าง (empty directories) ระบบแฟ้มเช่นนี้ แสดง ให้เห็น ในรูปที่ 9 ในระบบนี้ แฟ้ม khr อยู่ใน ไดเรกทอรี rje



รากคือไดเรกทอรีราก /
จุดภายในคือ ไดเรกทอรี
ใบคือ แฟ้ม

รูปที่ 9 ระบบแฟ้ม คอมพิวเตอร์

คุณสมบัติของต้นไม้ (Properties of Trees)

ทฤษฎีบท 2 ต้นไม้ ที่มี n จุด จะมี $n - 1$ ด้าน (A tree with n vertices has $n - 1$ edges.)

ทฤษฎีบท 3 ต้นไม้ m-ary แบบเต็มต้น ที่มี จุดภายใน n จุด จะมีจุดทั้งหมด $n = mi + 1$ จุด

(A full m-ary tree with i internal vertices contain $n = mi + 1$ vertices.)

ระดับ ของ จุด v ในต้นไม้ราก หมายถึง ความยาว ของทางเดินเพียงหนึ่งเดียว จาก ราก ไปยัง จุดนี้

(The **level** of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.)

ระดับ ของ ราก นิยามให้ เป็น ศูนย์ (The level of the root is defined to be zero.)

ความสูง ของต้นไม้ราก หมายถึง ระดับสูงที่สุด ของ จุดต่างๆ

(The **height** of a rooted tree is the maximum of the levels of vertices.)

พูดอีกอย่างหนึ่งคือ ความสูง ของต้นไม้ราก หมายถึง ความยาวของ ทางเดินยาวที่สุด จาก ราก ไปยัง จุดหนึ่ง

(The height of a rooted tree is the length of the longest path from the root to any vertex.)

ตัวอย่าง 7 จงหา ระดับของทุกจุด ใน ต้นไม้ราก ที่แสดงในรูปที่ 10 ต้นไม้นี้ มีความสูงเท่าใด?

ผลเฉลย

ราก a มีระดับเป็น 0

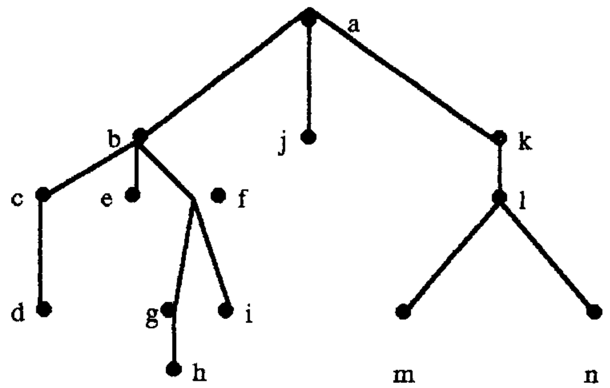
จุด b, j และ k อยู่ที่ระดับ 1

จุด c, e, f และ l อยู่ที่ระดับ 2

จุด d, g, i, m และ n อยู่ที่ระดับ 3

สุดท้าย จุด h อยู่ที่ระดับ 4

เนื่องจาก ระดับสูงที่สุด ของ จุดใดจุดหนึ่งคือ 4 เพราะฉะนั้น ต้นไม้นี้ ความสูงเท่ากับ 4



รูปที่ 10 ต้นไม้ราก

ต้นไม้ราก m-ary ที่มีความสูงเท่ากับ h จะเรียกว่า **ต้นไม้ได้ดุล** ถ้าจุดใบทั้งหมด อยู่ที่ระดับ h หรือ ระดับ h - 1

(A root m-ary tree of height h is called **balanced if all leaves are at levels** h or h-1.)

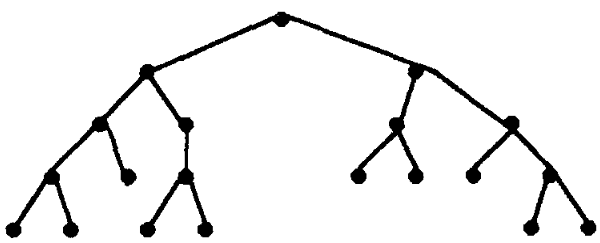
ตัวอย่าง 8 ต้นไม้ราก ที่แสดงในรูปที่ 11 ชุดใดบ้างเป็นต้นไม้ได้ดุล?

ผลเฉลย

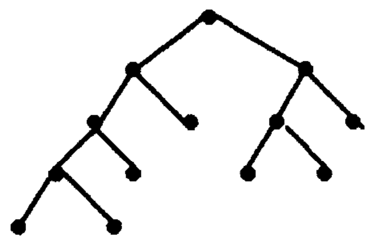
T_1 เป็นต้นไม้ได้ดุล เพราะว่า จุดใบทั้งหมดของมัน อยู่ที่ระดับ 3 และ 4

T_2 ไม่ใช้ต้นไม้ได้ดุล เพราะว่า จุดใบของมัน อยู่ที่ระดับ 2, 3 และ 4

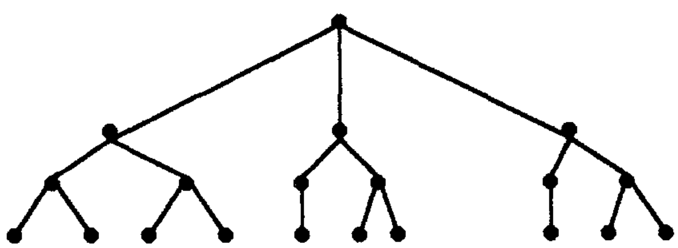
สุดท้าย T_3 เป็นต้นไม้ได้ดุล เพราะว่า จุดใบทั้งหมด อยู่ที่ระดับ 3



T_1



T_2



T_3

รูปที่ 11 ต้นไม้ราก สามชุด

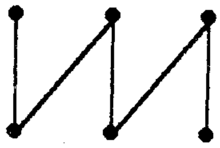
ทฤษฎีบท 4 ต้นไม้ m-ary ที่มีความสูงเท่ากับ h จะมีจุดใบ อย่างมากที่สุด m^h จุด

(There are at most m^h leaves in an m-ary tree of height h.)

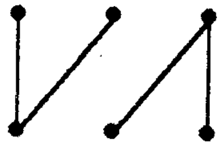
แบบฝึกหัด 7.1

1. กราฟข้างล่างนี้ ชุดใดบ้าง คือต้นไม้?

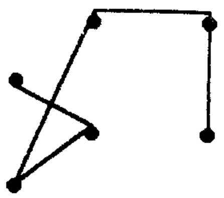
a)



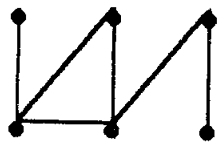
b)



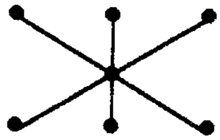
c)



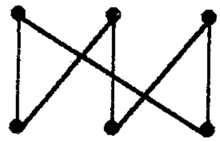
d)



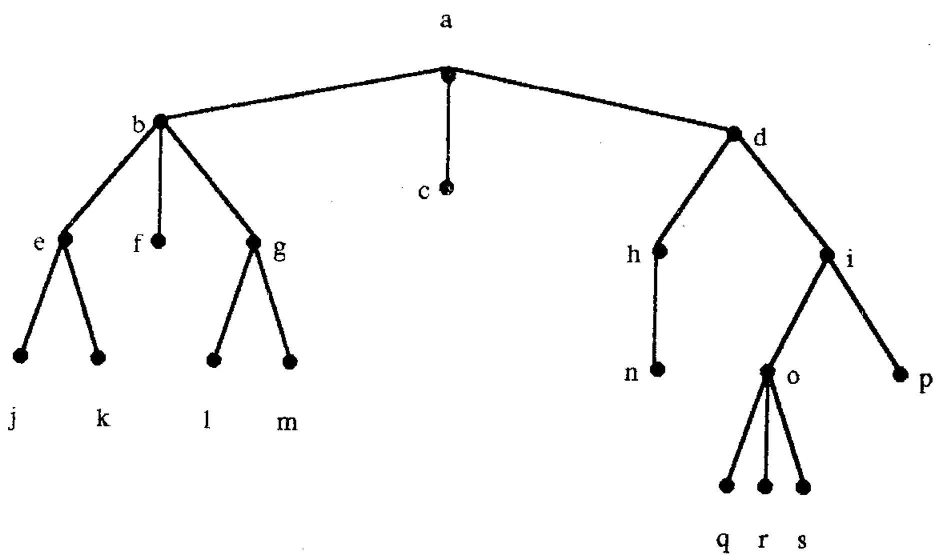
e)



f)



2. จงตอบคำถามเกี่ยวกับ ต้นไม้ราก ข้างล่างนี้



a) จุดใด เป็น ราก (root)

b) จุดใดบ้าง เป็น จุดภายใน (internal)

c) จุดใดบ้าง เป็น จุดใบ (leaves)

d) จุดใด คือ ลูก (children) ของ i

e) จุดใด คือ แม่ (parent) ของ h

f) จุดใดบ้าง เป็น จุดพี่น้อง (siblings) ของ o

g) จุดใดบ้าง เป็น จุดบน (ancestors) ของ m

h) จุดใดบ้าง เป็น จุดล่าง (descendants) ของ b

3. ต้นไม้รากในแบบฝึกหัดข้อ 2 เป็นต้นไม้ m-ary แบบเต็มต้น สำหรับ จำนวนเต็มบวก m บางค่าหรือไม่?

4. จงหา ระดับ ของทุกจุด ของต้นไม้ ในแบบฝึกหัดข้อ 2

5. จงวาดรูป ต้นไม้ส่วนย่อย ของต้นไม้ ในแบบฝึกหัดข้อ 2 ซึ่งมีรากอยู่ที่

   a) a b) c c) e

6. ต้นไม้ ซึ่งมี 10,000 จุดจะมี กี่ด้าน?

   (How many edges does a tree with 10,000 vertices have?)

7. ต้นไม้ 5-ary แบบเต็มต้น ที่มี จุดภายใน 100 จุด จะมีจำนวนจุดทั้งหมด กี่จุด?

   (How many vertices does a full 5-ary tree with 100 internal vertices have?)

8. ต้นไม้ทวิภาคแบบเต็มต้น ที่มี จุดภายใน 1,000 จุด จะมีด้านทั้งหมด กี่ด้าน?

   (How many edges does a full binary tree with 1,000 internal vertices have?)

ต้นไม้ m-ary แบบบริบูรณ์ หมายถึง ต้นไม้ m-ary แบบเต็มต้น ซึ่ง จุดใบ ทุกจุด อยู่ที่ระดับเดียวกัน

(A **complete m-ary tree** is a full m-ary bee where every leaf is at the same level.)

9. จงสร้าง ต้นไม้ทวิภาคแบบบริบูรณ์ มีความสูงเท่ากับ 4 และต้นไม้ 3-ary แบบบริบูรณ์ มีความสูง เท่ากับ 3

7.2 การประยุกต์ของต้นไม้ (Applications of Trees)

เราจะอภิปราย ปัญหา สามเรื่อง ซึ่งสามารถนำมาศึกษาได้โดยใช้ต้นไม้

ปัญหาแรก : ข้อมูลต่างๆ ในรายการ จะเรียงลำดับอย่างไร เพื่อให้สามารถ หาตำแหน่งของข้อมูลหนึ่งตัวได้ง่าย

ปัญหาที่สอง : ลำดับ (series) อะไร ของการตัดสินใจ ซึ่งควรจะกระทำในการหา สิ่งของสิ่งหนึ่ง ที่มีคุณสมบัติหนึ่งอย่างใน กลุ่มของสิ่งของ ที่มีชนิดคุณสมบัตินั้น

ปัญหาที่สาม : เซตของตัวอักขระจะลงรหัสอย่างมีประสิทธิภาพ ด้วยสายบิต ได้อย่างไร?

(How should a set of characters be efficiently coded by bit strings?)

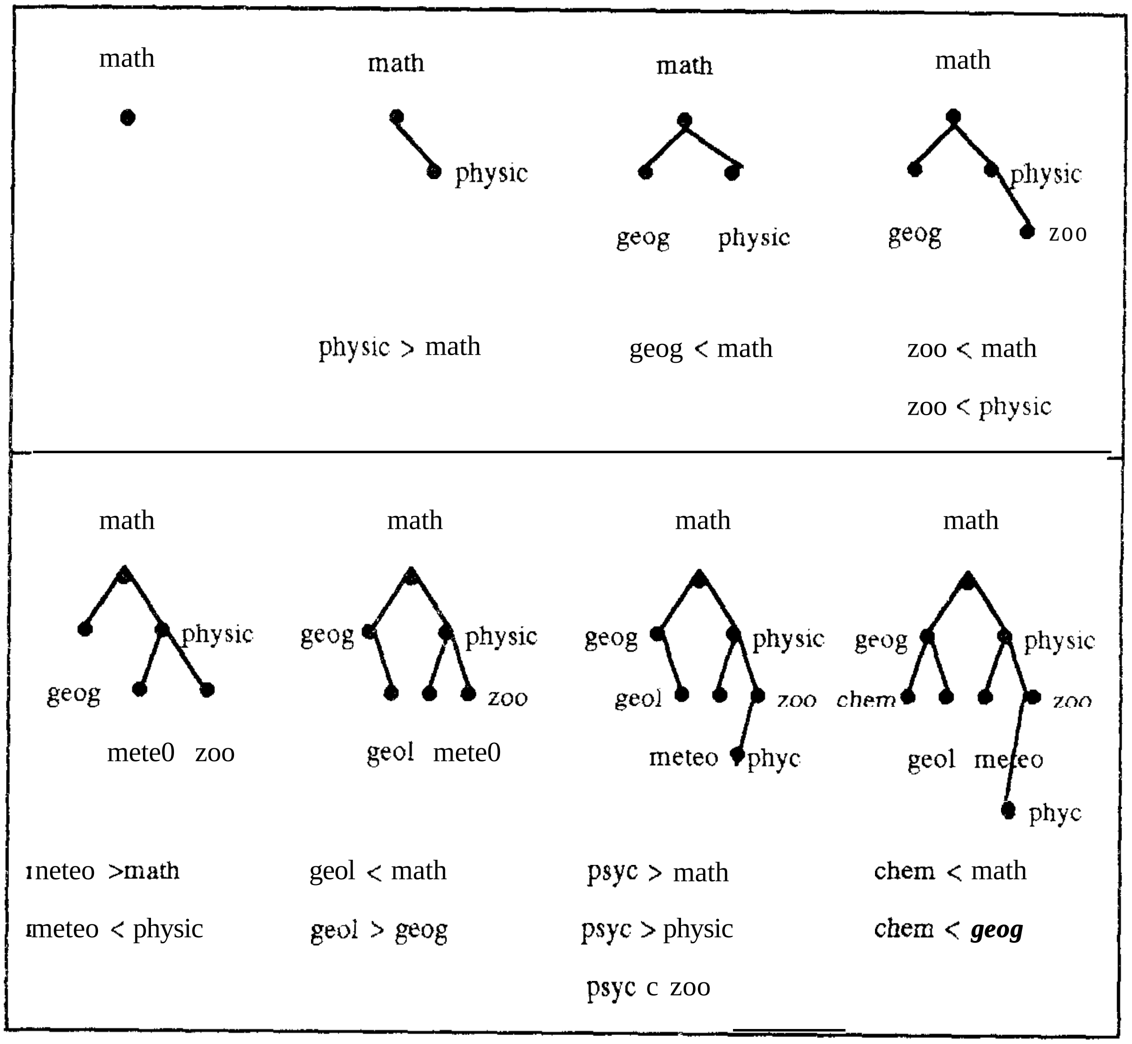
ต้นไม้ค้นแบบทวิภาค (Binary Search Trees)

การค้นหา ข้อมูล (items) ในรายการ เป็นงานที่สำคัญมากที่สุดอย่างหนึ่ง ในสาขาคอมพิวเตอร์ เป้าหมายแรกของเราคือ การทำให้ อัลกอริทึมการค้น (searching algorithm) ปฏิบัติงานหา items ได้อย่างมีประสิทธิภาพ เมื่อ item ทั้งหมด มีการเรียงอันดับ สิ่งนี้ทำให้สำเร็จได้ โดยใช้ต้นไม้ค้นแบบทวิภาค

ต้นไม้ค้นแบบทวิภาค หมายถึง ต้นไม้แบบทวิภาค ซึ่งลูกแต่ละคน ของหนึ่งจุด ถูกกำหนดให้เป็น ลูกทางซ้าย (left child) หรือ เป็นลูกทางขวา (right child) ไม่มีจุดใดเลย มี ลูกทางซ้าย หรือ ลูกทางขวา มากกว่า หนึ่ง แต่ละจุด ระบุโดย **คีย์** (key) ซึ่งเป็นข้อมูลตัวหนึ่ง นอกจากนั้นแล้ว จุดต่างๆ ถูกกำหนดค่าให้เป็น keys ดังนั้น คีย์ ของจุดหนึ่ง จะมีค่ามากกว่า คีย์ ของจุดทั้งหมด ใน ต้นไม้ส่วนย่อยซ้ายของมัน และ มีค่าน้อยกว่า คีย์ของจุดทั้งหมด ใน ต้นไม้ส่วนย่อยขวาของมัน

กระบวนการเรียกซ้ำต่อไปนี้ ถูกนำมาใช้เพื่อประกอบเป็นต้นไม้ค้นแบบทวิภาค สำหรับรายการของข้อมูล เริ่มต้นด้วยต้นไม้ ซึ่งมีเพียงหนึ่งจุดเท่านั้น ชื่อ ราก (root) ข้อมูลตัวแรกในรายการ ถูกกำหนด ให้เป็นคีย์ของราก การใส่ข้อมูลตัวใหม่ ขั้นแรก เปรียบเทียบ ข้อมูลตัวนี้ กับ คีย์ ของ จุดที่มีอยู่แล้ว ในต้นไม้ เริ่มต้นจาก root และย้ายไปทางซ้าย ถ้าข้อมูลนี้ มีค่าน้อยกว่า คีย์ของ จุดโดยลำดับ (respective vertex) ถ้าจุดนี้ มีลูกทางซ้าย หรือ ย้ายไปทางขวา ถ้าข้อมูลนี้ มีค่ามากกว่า คีย์ของจุดโดยลำดับ ถ้าจุดนี้มีลูกทางขวา เมื่อข้อมูล นี้มีค่าน้อยกว่า คีย์โดยลำดับ และ จุดนี้ไม่มีลูกทางซ้าย ดังนั้น จุดใหม่ ซึ่งมี ข้อมูลเป็นคีย์ของมัน จะใส่เป็น ลูกทางซ้ายตัวใหม่ ในทำนองเดียวกัน เมื่อข้อมูลมีค่ามากกว่าจุดโดยลำดับ และจุดนี้ ไม่มี ลูกทางขวา ดังนั้น จุดใหม่ ที่มี ข้อมูลเป็นคีย์ของมัน จะใส่ เป็น ลูกทางขวาตัวใหม่

ตัวอย่าง 1 จงสร้างต้นไม้ค้นแบบทวิภาค สำหรับคำต่อไปนี้ **math**, physic, geogr, zoo, meteo, geolo, psyc และ chem (ใช้การเรียงลำดับ ตามตัวอักษร)



รูปที่ 1 การสร้างต้นไม้ค้นแบบทวิภาค

อัลกอริทึม 1 Binary Search Tree Algorithm

procedure insertion (T : binary search tree, x : item)

v := root of T

{a vertex not present in T has the value null }

while v $\neq$ null and label(v) $\neq$ x

begin

if x< label(v) **then**

if left child of v $\neq$ **null** then v := left child of v

else add new vertex as a left child of v and set v := null

else

if right child of y $\neq$ null then v := right child of v

else add new vertex as a right child of v to T and set v := null

end

if root of T = null then add a vertex r to the tree and label it with x

else if label(v) $\neq$ x then label new vertex with x

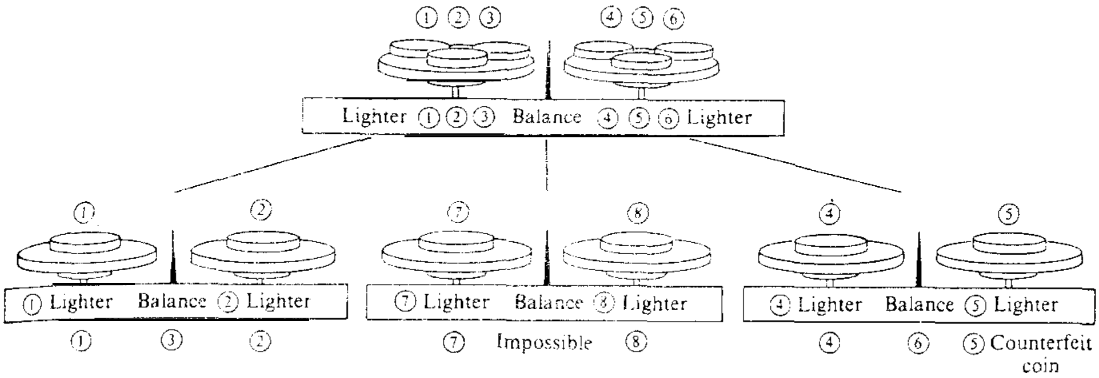
{v = location of **x**}

ต้นไม้การตัดสินใจ (Decision Trees)

ต้นไม้ราก สามารถนำมาใช้ เป็นตัวแบบ ปัญหา ซึ่ง ลำดับของการตัดสินใจ นำไปสู่ ผลเฉลย ตัวอย่างเช่น ต้นไม้ค้นแบบทวิภาค นำมาใช้ หา ตำแหน่ง ข้อมูล โดยขึ้นอยู่กับ ลำดับ ของการเปรียบเทียบ ซึ่ง การเปรียบเทียบแต่ละครั้ง บอกให้เราทราบว่า จะพบ ตำแหน่งของ ข้อมูล หรือไม่ หรือ เราควรจะไปทางต้นไม้ส่วนย่อยขวา หรือ ควรจะไปทางต้นไม้ส่วนย่อย ซ้าย ต้นไม้ราก ซึ่ง จุดภายใน แต่ละจุด สมนัย กับการตัดสินใจ กับ ต้นไม้ส่วนย่อย ที่จุดเหล่านี้ สำหรับ outcome ที่เป็นไปได้แต่ละชุดของการตัดสินใจ เรียกว่า **ต้นไม้การตัดสินใจ** (decision tree)

ผลเฉลยที่เป็นไปได้ ของ ปัญหา สมนัย กับ ทางเดิน ไปยังจุดใบ ของต้นไม้รากนี้

ตัวอย่าง 2 สมมติว่า มีเหรียญอยู่ เจ็ดเหรียญ ทุกเหรียญมีน้ำหนักเท่ากัน และมีเหรียญเก๊ อยู่ หนึ่งเหรียญ ซึ่ง มีน้ำหนักน้อยกว่าเหรียญอื่นๆ จะมีการชั่งน้ำหนักเท่าที่จำเป็น กี่ครั้ง โดยใช้ ตราชั่ง แบบลูกตุ้ม (balance scale) เพื่อหาว่า ในแปดเหรียญนี้ มีเหรียญอันไหนเป็นเหรียญเก๊



รูปที่ 2 ต้นไม้การตัดสินใจ เพื่อหาเหรียญเก๊

ผลเฉลย มีสิ่งที่เป็นไปได้ สามสิ่ง สำหรับการชั่งน้ำหนัก แต่ละครั้ง บนตราชั่ง คือ จาน สองใบ น้ำหนักเท่ากัน จานใบแรก หนักกว่า หรือ จานใบที่สอง หนักกว่า เพราะฉะนั้น ต้นไม้การตัดสินใจ สำหรับ ลำดับของการชั่งน้ำหนัก คือ ต้นไม้ 3-ary ซึ่งมีจุดใบอย่างน้อยที่สุด แปดใบ ใน ต้นไม้การตัดสินใจ เพราะว่า มี outcome ที่เป็นไปได้ 8 อย่าง (เพราะว่า ทั้งแปดเหรียญ จะมีเหรียญอยู่ หนึ่งเหรียญ ที่เป็น เหรียญเก๊) และ outcome ที่เป็นไปได้แต่ละอย่าง แทนด้วย จุดใบ อย่างน้อยหนึ่งจุด จำนวนมากที่สุดของการชั่งน้ำหนักที่จำเป็น ในการหาเหรียญเก๊ คือ ความสูงของต้นไม้การตัดสินใจ

จากต้นไม้การตัดสินใจ ที่แสดงให้เห็นในรูปที่ 2 การหาเหรียญเก๊ จะต้องชั่งน้ำหนักสองครั้ง

รหัสเติมหน้า (Prefix Codes)

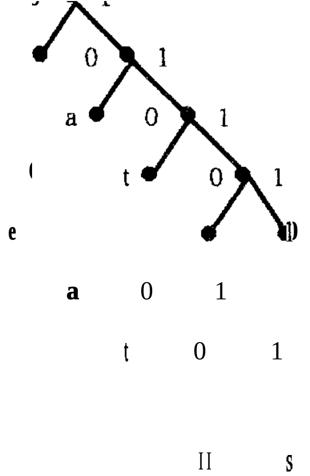
จงพิจารณา ปัญหา ของการใช้ สายบิต (bit strings) เพื่อเข้ารหัส (encode) ตัวอักษรของตัวอักษรภาษาอังกฤษ (เมื่อ ตัวอักษรตัวเล็ก และตัวใหญ่ไม่แตกต่างกัน) เราแทนอักษรแต่ละตัวด้วยสายบิตความยาวเท่ากับ 5 เพราะว่า มีอักษรเพียง 26 ตัวเท่านั้น และ สายบิตความยาวเท่ากับ 5 ที่แตกต่างกันมี 32 ชุด จำนวนบิตทั้งหมด ที่ใช้เพื่อเข้ารหัส ข้อมูล จะเป็นห้าเท่า ของ จำนวนตัวอักขระในข้อความ (text) เมื่อตัวอักระแต่ละตัวเข้ารหัสด้วยห้าบิต มันเป็นไปได้หรือไม่ ที่จะหา**โครงร่างรหัส** (coding scheme) ของ ตัวอักษรเหล่านี้ เมื่อมีการใช้รหัสข้อมูล โดยใช้ บิตที่มีจำนวนน้อยกว่า? ถ้าสามารถทำได้ จะเป็นการประหยัดหน่วยความจำและลดเวลาการส่ง

พิจารณาการใช้สายบิต ของ ความยาวแตกต่างกัน เพื่อเข้ารหัสตัวอักษรต่างๆ ตัวอักษรที่ใช้บ่อยกว่า ควรจะเข้ารหัส โดยใช้สายบิตสั้น และตัวอักษรที่ไม่ค่อยได้ใช้ ควรจะใช้สายบิตยาวกว่าในการเข้ารหัส เมื่อ ตัวอักษร ถูกเข้ารหัส โดยใช้ จำนวนบิตแปรเปลี่ยนได้ มีบางวิธีต้องถูกนำมาใช้ เพื่อหา ที่ใดของบิต สำหรับตัวอักขระที่เริ่มต้นและจบ ตัวอย่างเช่น ถ้า e เข้ารหัสด้วย 0, a ด้วย 1, และ t ด้วย 01 ดังนั้น สายบิต 0101 ควรจะสมนัยกับ eat, tea, eaea หรือ tt

เพื่อให้แน่ใจว่าไม่มีสายบิตใดๆ สมนัยกับ ลำดับ ของตัวอักษรมากกว่า หนึ่งชุด สายบิตสำหรับ ตัวอักษรหนึ่งตัว ต้อง ไม่เกิด เป็นส่วนแรก ของ สายบิต สำหรับ ตัวอักษรอีกหนึ่งตัว รหัสที่มีคุณสมบัตินี้ เรียกว่า **รหัสเติมหน้า** (prefix codes) ตัวอย่างเช่น การเข้ารหัส ของ e คือ 0, a คือ 10 และ t คือ 11 เป็น รหัสเติมหน้า คำหนึ่งคำ สามารถ เอากลับคืนจาก สายบิตที่เป็นได้อย่างเดียว ซึ่งเข้ารหัสตัวอักษรของมัน ตัวอย่างเช่น สายบิต 10110 คือการเข้ารหัสของ ate โปรดสังเกตว่า เลข 1 ตัวแรกไม่ได้แทนตัวอักขระ แต่ 10 แทน a (และไม่สามารถเป็นส่วนแรกสายบิตของตัวอักษรอีกหนึ่งตัว) ดังนั้น 1 ตัวถัดไป จึง ไม่ใช่แทนตัวอักขระ แต่ 11 แทนตัว t บิตสุดท้าย 0 แทน e

รหัสเติมหน้า สามารถถูกแทนที่ โดยใช้ ต้นไม้แบบทวิภาค เมื่อ ตัวอักขระคือ labels ของจุดใบ ในต้นไม้ ด้านของต้นไม้ ถูกระบุเพื่อให้ด้านที่นำไปสู่ลูกทางซ้าย ถูกกำหนดค่าเป็น 0 และด้านที่นำไปสู่ลูกทางขากถูกกำหนดค่า เป็น 1 สายบิตใช้เข้ารหัสตัวอักขระ หนึ่งตัว คือลำดับของ labels ของด้าน ในทางเดินเป็นได้อย่างเดียว (unique path) จาก ราก ไปยัง ใบ ซึ่งมีตัวอักขระนี้ เป็น label ของมัน ตัวอย่างเช่น ต้นไม้ในรูปที่ 3 แทนการเข้ารหัส ของ e ด้วย 0, a ด้วย 10, t ด้วย 110, n ด้วย 1110 และ s ด้วย 1111

ต้นไม้ ซึ่งแทนที่ รหัส สามารถนำมาใช้ เพื่อ**ถอดรหัส** (decode) สายบิต หนึ่งชุด ตัวอย่างเช่น คำซึ่งเข้ารหัสด้วย 11111011100 โดยใช้รหัส ในรูปที่ 3 สายบิตนี้ สามารถถอดรหัสโดยเริ่มต้นที่ราก ใช้ลำดับ ของบิตเพื่อประกอบเป็นทางเดิน ซึ่งหยุด เมื่อถึง ใบ ของจุดสุดท้ายในทางเดิน และบิต 1 แต่ละตัว สมนัยกับ ลูกทางขวาของจุดนี้ เพราะฉะนั้น เริ่มแรก 1111 สมนัยกับ ทางเดิน ตั้งต้นที่ราก ไปทางขวามือ สี่ครั้ง ไปจนถึง ใบในกราฟ ซึ่งมี s เป็น label ของมัน เพราะว่า สายบิต 1111 คือ รหัสของ s ต่อไปคือบิตที่ 5 เรามาถึง ใบ หลังจาก เริ่มจากรากไปทางขวาแล้วไปทางซ้าย จุดนั้น ชื่อ a ถูกเยี่ยม ซึ่งเข้ารหัสด้วย 10 เริ่มต้นใหม่ด้วยบิตที่เจ็ด เรามาถึงใบ หลังจากไปทางขวา สามครั้ง แล้วมาทางซ้าย จุดนั้นชื่อ n ถูกเยี่ยม ซึ่งเข้ารหัสด้วย 1110 สุดท้าย บิต 0 นำไปสู่ใบ ซึ่งมี label ด้วย e เพราะฉะนั้น คำต้นฉบับ คือ sane



รูปที่ 3 ต้นไม้แบบทวิภาค ด้วย รหัสเติมหน้า

เราสามารถสร้าง รหัสเติมหน้า จาก ต้นไม้แบบทวิภาค ชุดใด ชุดหนึ่ง เมื่อ ด้านซ้าย ที่จุดภายในแต่ละจุด ติดป้าย (labels) ด้วย 0 และด้านขวา ด้วย 1 และที่ใบ ถูก label ด้วยตัวอักขระ ทั้งนี้ ตัวอักขระต่างๆ เข้ารหัสด้วย สายบิต สร้าง โดยใช้ labels ของด้านต่างๆ ในทางเดินเป็นได้อย่างเดียว จาก ราก ไปจนถึง ใบ

แบบฝึกหัด 7.2

1. จงสร้าง ต้นไม้ค้นแบบทวิภาค สำหรับ คำต่อไปนี้
   banana, peach, apple, pear, coconut, mango, และ papaya โดยเรียงลำดับตามตัวอักษร
2. มีการเปรียบเทียบ กี่ครั้ง ในการหา หรือ ใส่คำแต่ละคำข้างล่างนี้ ในต้นไม้ค้นแบบทวิภาค ในแบบฝึกหัดข้อ 1 ทั้งนี้ให้เริ่มต้นใหม่ ทุกครั้ง

   a) pear b) banana

   c) kumquat d) orange
3. จะมีการชั่งน้ำหนักที่จำเป็นบนตราชั่งกี่ครั้ง ในการหาเหรียญเก๊ ที่มีน้ำหนักเบากว่าเหรียญอื่นๆ ในเหรียญทั้งหมด สี่เหรียญ จากนั้น ให้เขียนอัลกอริทึม หา เหรียญที่มีขนาดเบากว่า โดยใช้ จำนวนการชั่งน้ำหนักนี้
4. จะมีการชั่งน้ำหนักที่จำเป็นบนตราชั่งกี่ครั้ง ในการหาเหรียญเก๊ หนึ่งเหรียญ จากเหรียญทั้งหมด 12 เหรียญ ถ้าเหรียญเก๊ มีน้ำหนักเบากว่าเหรียญอื่นๆ จากนั้นให้เขียนอัลกอริทึม หา เหรียญที่เบากว่า โดยใช้จำนวนการชั่งนี้

5. รหัสต่อไปนี้ ชุดใดเป็นรหัสเติมหน้า?

a) a : 11, e : 00, t : 10, s : 01

b) a: 0, e : 1, t : 01, s : 001

c) a : 101, e : 11, t : 001, s : 011, n : 010

d) a : 010, e : 11, t : 011, s : 1011, n : 1001, i : 10101

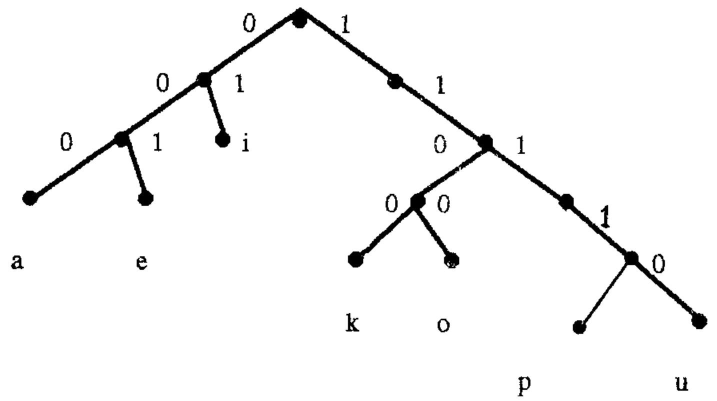
6. จงสร้างต้นไม้แบบทวิภาค ด้วย รหัสเติมหน้า แทนที่ โครงร่างรหัสต่อไปนี้

a) a : 11, e : 0, t : 101, s : 100

b) a : 1, e : 01, t : 001, s : 0001, n : 00001

c) a : 1010, e : 0, t : 11, s : 1011, n : 1001, i : 100001

7. จงหารหัส ของ ตัวอักษร a, e, i, k, o, p และ u ถ้า โครงร่างรหัส ถูกแทนที่ ด้วย ต้นไม้ข้างล่างนี้



8. กำหนดโครงร่างรหัส ให้ดังนี้

a : 001, b : 0001, e : 1, r : 0000, s : 0100, t : 011, x : 01010

จงหาคำซึ่งแทนด้วยรหัส ข้างล่างนี้

a) 01110100011

b) 0001110000

c) 0100101010

d) 01100101010

7.3 การแวะผ่านต้นไม้ (Tree Traversal)

ต้นไม้รากแบบอันดับ (ordered rooted tree) **บ่อยครั้ง นำมาใช้** เก็บสารสนเทศ เรา **จำเป็นต้องมี** กระบวนงาน (procedure) สำหรับ เยี่ยม ทุกจุด ของ ต้นไม้รากแบบอันดับ เพื่อ **เข้าถึงข้อมูล ในหัวข้อนี้** จะได้อธิบาย อัลกอริทึม ที่สำคัญ หลายชุด สำหรับเยี่ยม จุดทั้งหมด **ของ ต้นไม้รากแบบอันดับ**

ต้นไม้รากแบบอันดับ สามารถนำมาใช้แทน **นิพจน์** ได้หลายชนิด เช่น นิพจน์คำนวณ **เกี่ยวข้องกับ เลข** (numbers) ตัวแปร (variables) หรือ การปฏิบัติการ (operations) รายการที่ แตกต่างกัน **ของจุด ของ ต้นไม้รากแบบอันดับ ที่ใช้แทนที่ นิพจน์ จะเป็นประโยชน์** ในการ ประเมินผล นิพจน์เหล่านี้

ระบบการให้เลขที่อยู่แบบสากล (Universal Address System)

กระบวนงาน สำหรับ การแวะผ่าน ทุกจุด ของ ต้นไม้รากแบบอันดับ วางอยู่บน การ เรียงอันดับ **ของลูก** ในต้นไม้รากแบบอันดับ บรรดาลูกของจุดภายใน แสดงให้เห็น จากซ้าย ไปขวา ในการวาดรูป แทน กราฟแบบมีทิศทางเหล่านี้

เราจะอธิบาย วิธีหนึ่ง เพื่อเรียงอันดับ จุดทั้งหมด ของ ต้นไม้รากแบบอันดับ ในการ สร้าง การเรียงลำดับ นี้ สิ่งแรก เราต้อง ให้ชื่อ (label) ทุกจุด เราทำสิ่งนี้ แบบเรียกซ้ำ ดังนี้

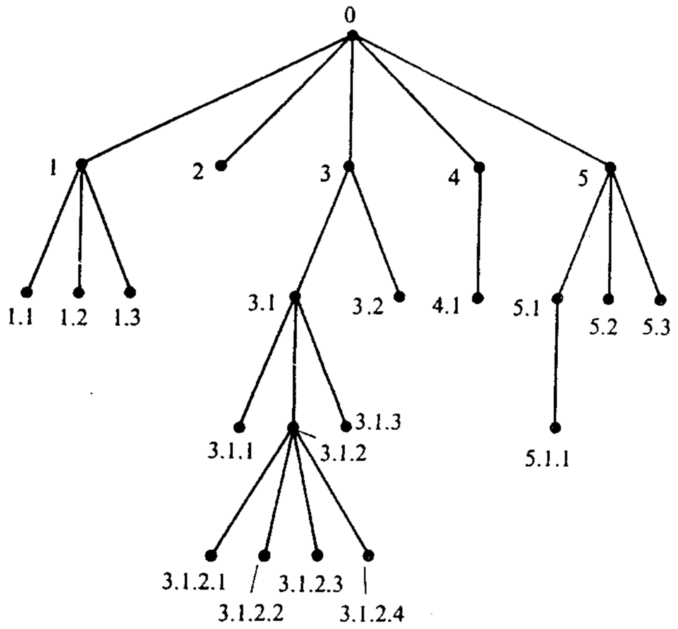
1) ให้ชื่อ ราก ด้วยจำนวนเต็ม 0 จากนั้น ชื่อลูก ของมัน k จุด (ที่ระดับ 1) จากซ้ายไป ขวา คือ 1, 2, 3, . . . , k

2) สำหรับจุด v แต่ละตัว ที่ระดับ n ด้วยชื่อ A ให้ชื่อลูก k_v ตัวของมัน วาดจากรูปซ้าย ไปขวา ด้วย A.1, A.2, . . . , A.k_v

การทำตาม กระบวนงานนี้ จุด v ที่ระดับ n เมื่อ $n \geq 1$ มีชื่อ $x_1, x_2, \ldots, x_n$ โดยที่ ทางเดินเป็นเพียงอย่างเดียว จากรากไปยัง v ไปจนถึง จุดที่ x_1 อยู่ที่ระดับ 1, จุดที่ x_2 อยู่ระดับ 2 เช่นนี้เรื่อยไป การให้ชื่อเช่นนี้ เรียกว่า **ระบบให้เลขที่อยู่แบบสากล** (universal address system) ของ ต้นไม้รากแบบอันดับ

ตัวอย่าง 1 ในรูปที่ 1 เราแสดงให้เห็น การให้ชื่อ ของระบบ ให้เลขที่อยู่แบบสากล กับ จุดต่างๆ ใน ต้นไม้รากแบบอันดับ การเรียงลำดับอักษร (lexicographic ordering) ของการให้ชื่อ เป็นดังนี้

0 < 1 < 1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 < 3.1.2.3 < 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.5.1 < 5.2 < 5.3



รูปที่ 1 ระบบการให้เลขที่อยู่แบบสากลของต้นไม้รากแบบอันดับ

อัลกอริทึมการแวะผ่าน (Traversal Algorithms)

กระบวนงาน สำหรับการเยี่ยม ทุกจุด อย่างมีระบบ ของ ต้นไม้รากแบบอันดับ เรียกว่า อัลกอริทึมการแวะผ่าน

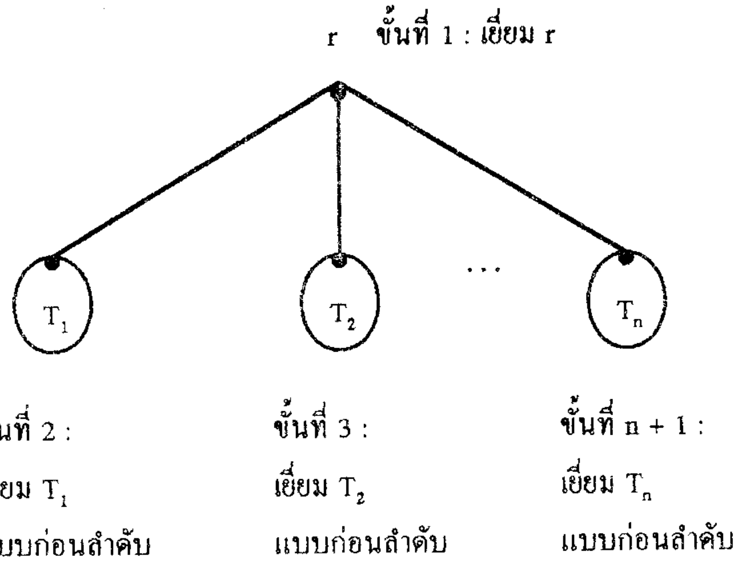
(Procedures for systematically visiting every vertex of an ordered rooted tree are called **traversal algorithms**.)

เราจะอธิบาย อัลกอริทึม เช่นนี้ ซึ่งใช้ร่วมกันมากที่สุด สามชนิด คือ

- การแวะผ่านแบบก่อนลำดับ (preorder traversal)
- การแวะผ่านแบบตามลำดับ (inorder traversal)
- การแวะผ่านแบบหลังลำดับ (postorder traversal)

บทนิยาม 1 ให้ T เป็นต้นไม้รากแบบอันดับ มีราก ชื่อ r ถ้า T มีรากเพียงจุดเดียวเท่านั้น r คือ **การแวะผ่านแบบก่อนลำดับ** (preorder traversal) ของ T กรณีอื่นๆ สมมติว่า $T_1, T_2, \ldots, T_n$ เป็นเซตย่อยของ r จาก ซ้ายไปขวา ใน T **การแวะผ่านแบบก่อนลำดับ** เริ่มต้นโดยการเยี่ยม

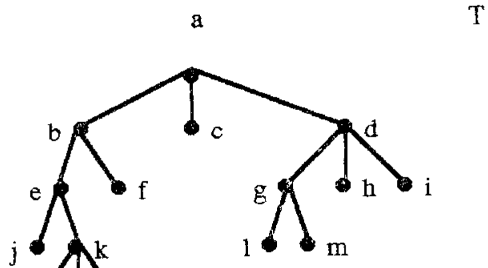
r ต่อไป แวะผ่าน T_1 แบบก่อนลำดับ จากนั้น แวะผ่าน T_2 แบบก่อนลำดับ เช่นนี้เรื่อยไป จนกระทั่ง T_n ถูกแวะผ่าน แบบก่อนลำดับ



รูปที่ 2 การแวะผ่านแบบก่อนลำดับ (Preorder Traversal)

ตัวอย่าง 2 จงหา อันดับของการ แวะผ่านแบบก่อนลำดับ เยี่ยมจุดต่างๆ ในต้นไม้รากแบบอันดับ ชื่อ T ในรูปที่ 3

ผลเฉลย ขั้นตอนต่างๆ ของ การแวะผ่านแบบก่อนลำดับ ของ T แสดงให้เห็นในรูปที่ 4 เราแวะผ่าน ต้นไม้ T แบบก่อนลำดับ โดยสิ่งแรก คือ ชื่อ ราก a ตามด้วย รายการแสดงแบบก่อนลำดับ ของต้นไม้ส่วนย่อยที่มีรากเป็น b รายการแสดงแบบก่อนลำดับ ของต้นไม้ส่วนย่อยด้วยราก c (ซึ่งมีแค่ c จุดเดียว) และรายการแบบก่อนลำดับของต้นไม้ส่วนย่อย ด้วยราก d



รูปที่ 3 ต้นไม้รากแบบอันดับ T

รายการแสดงแบบก่อนลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก b เริ่มต้นด้วย ชื่อ b จากนั้นจุดต่างๆ ของ ต้นไม้ส่วนย่อย ด้วยราก e แบบก่อนลำดับ แล้ว ต้นไม้ส่วนย่อย ด้วยราก f แบบก่อนลำดับ (มีเพียงจุด f)

รายการแสดงแบบก่อนอันดับ ของต้นไม้ส่วนย่อย ด้วย ราก d เริ่มต้นด้วย ชื่อ d ตามด้วยรายการแสดงแบบก่อนลำดับ ของต้นไม้ส่วนย่อย ด้วย ราก g ตามด้วยต้นไม้ส่วนย่อย ด้วย ราก h (มีเพียงจุด h) ตามด้วย ต้นไม้ส่วนย่อย ด้วย ราก i (มีเพียงจุด i)

รายการแสดงแบบก่อนอันดับ ของต้นไม้ส่วนย่อย ด้วย ราก e เริ่ มต้นด้วยชื่อ e ตามด้วยรายการแสดงแบบก่อนอันดับ ของ ต้นไม้ส่วนย่อย ด้วย ราก j (มีเพียงจุด j) ตามด้วย รายการแสดงแบบก่อนอันดับ ของต้นไม้ส่วนย่อย ด้วย ราก k

รายการแสดงแบบก่อนลำดับ ของต้นไม้ส่วนย่อย ด้วย ราก g คือชื่อ g ตามด้วย l ตามด้วย m รายการแสดงแบบก่อนลำดับ ของ ต้นไม้ส่วนย่อย ด้วย ราก k คือ k, n, o, p

เพราะฉะนั้น การแวะผ่านแบบก่อนลำดับ ของ T คือ

a, b, e, j, k, n, o, p, f, c, d, g, l, m, h, i

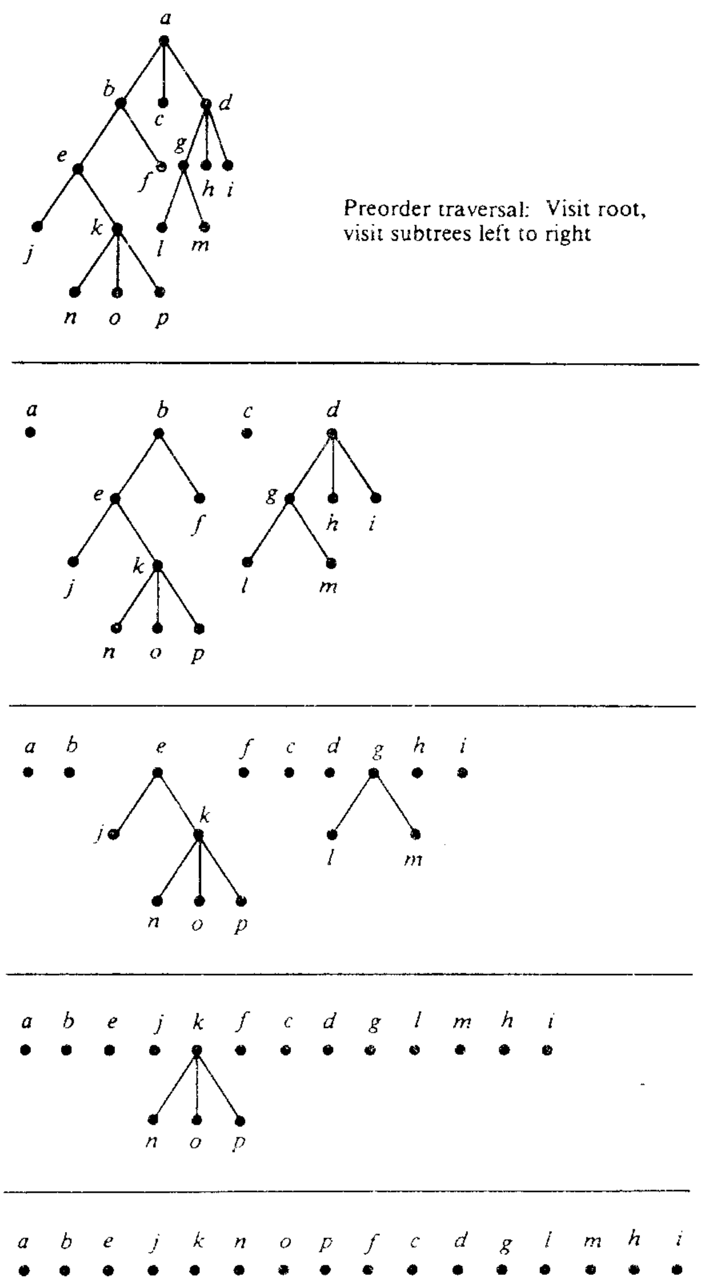
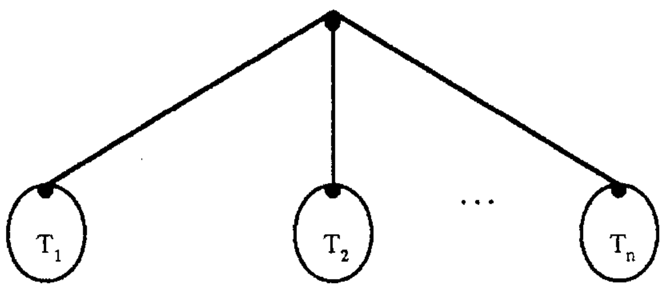


FIGURE 4 **The Preorder Traversal of *T*.**

บทนิยาม 2 ให้ T เป็นต้นไม้รากแบบอันดับ มีรากเป็น r ถ้า T มี r เพียงจุดเดียว ดังนั้น r คือ **การแวะผ่านแบบตามลำดับ** (inorder travesal) ของ T กรณีอื่นๆ สมมติว่า $T_1, T_2, \ldots, T_n$ เป็นต้นไม้ส่วนย่อย ที่ r จาก ซ้ายไปขวา การแวะผ่านแบบตามลำดับ เริ่มต้นด้วย การแวะผ่าน T_1 แบบตามลำดับ จากนั้นเยี่ยมจุด r ต่อไป แวะผ่าน T_2 แบบตามลำดับ จากนั้น T_3 แบบตามลำดับ เช่นนี้เรื่อยไป และสุดท้าย แวะผ่าน T_n แบบตามลำดับ ในรูปที่ 5 แสดงให้เห็นว่า การแวะผ่านแบบตามลำดับ ทำจนสำเร็จได้อย่างไร

ตัวอย่าง 3 จงหาอันดับของการแวะผ่านแบบตามลำดับ เยี่ยมจุดต่างๆ ของต้นไม้รากแบบอันดับ T ในรูปที่ 3 (In which order does an inorder traversal visit the vertices of the ordered rooted tree T in Figure 3?)

r ขั้นที่ 2 : เยี่ยม r



ขั้นที่ 1 : เยี่ยม T_1 แบบตามลำดับ

ขั้นที่ 3 : เยี่ยม T_2 แบบตามลำดับ

ขั้นที่ n + 1 : เยี่ยม T_n แบบตามลำดับ

รูปที่ 5 การแวะผ่านแบบตามลำดับ (Inorder Traversal)

ผลเฉลย ขั้นตอนต่างๆ ของ การแวะผ่านแบบตามลำดับ ของ ต้นไม้รากแบบอันดับ T แสดงให้เห็นในรูปที่ 6

การแวะผ่านแบบตามลำดับ เริ่มต้น การแวะผ่านแบบตามลำดับ ของต้นไม้ส่วนย่อย ด้วยราก b จากนั้น ราก a ต่อด้วยรายการแสดงแบบตามลำดับ ของต้นไม้ส่วนย่อย ด้วยราก c ซึ่งมีเพียงจุด c และรายการแสดงแบบตามลำดับ ของต้นไม้ส่วนย่อย d ด้วยราก d

รายการแสดงแบบตามลำดับ ของต้นไม้ส่วนย่อย ด้วยราก b เริ่มต้น ด้วยรายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก e ราก b และ f

รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก d เริ่มต้นด้วย รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก g ตามด้วย ราก d และ ตามด้วย h, ตามด้วย i

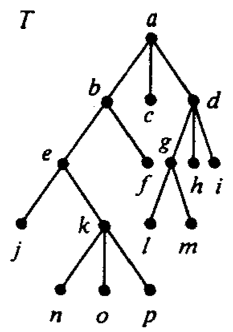
รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก e คือ j ตามด้วย ราก e ตามด้วย รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก k

รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก g คือ l, g, m

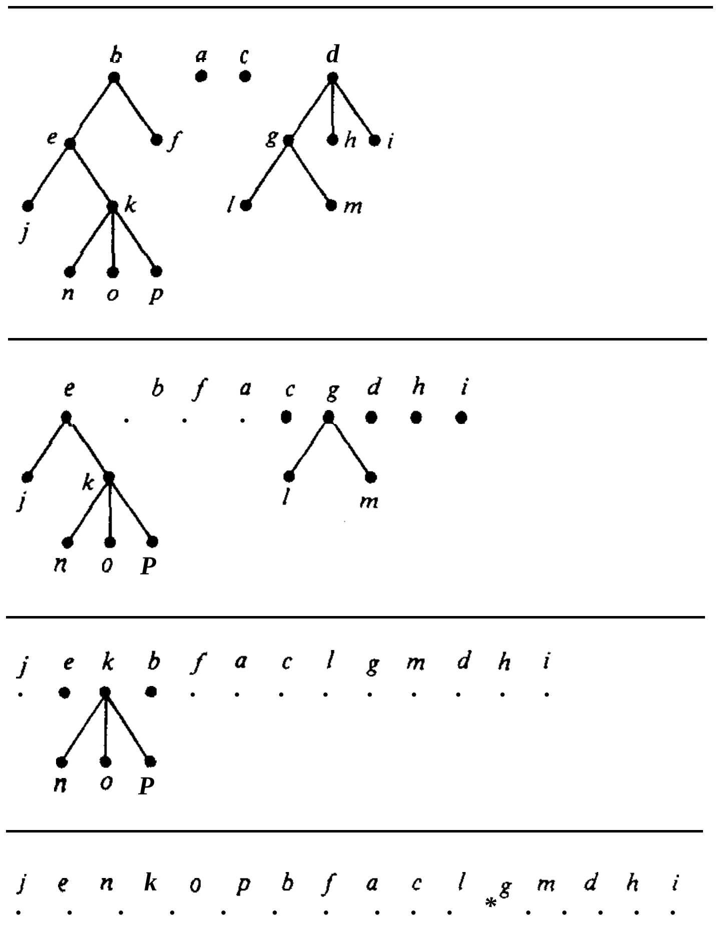
รายการแสดงแบบตามลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก k คือ n, k, o, p

เพราะฉะนั้น รายการแสดงแบบตามลำดับ ของ ต้นไม้รากแบบอันดับ คือ

j, e, n, k, o, p, b, f, a, c, l, g, m, d, h, i

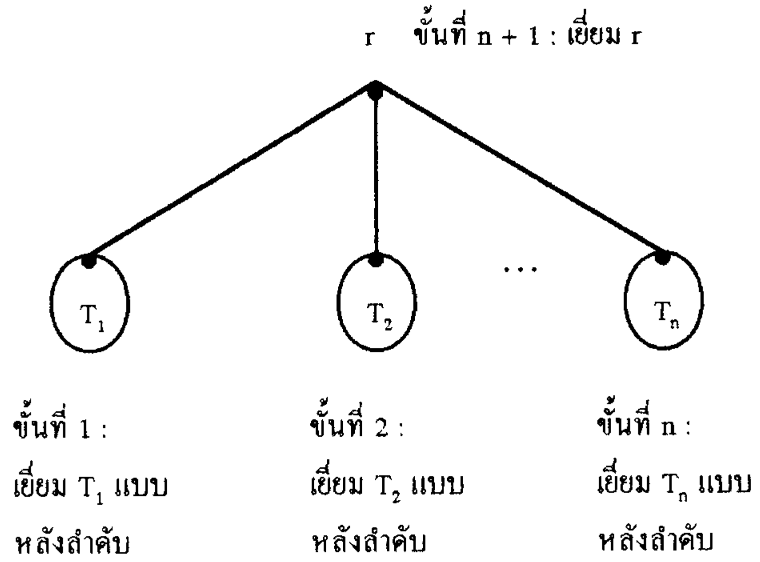


Inorder traversal: Visit leftmost **subtree,** visit root, visit other **subtrees** left to right



รูปที่ 6 การแวะผ่านแบบตามลำดับ ของ ต้นไม้ T

บทนิยาม 8 ให้ T เป็นต้นไม้รากแบบอันดับ มีรากเป็น r ถ้า T มีเพียง r เพียงจุดเดียว จะได้ว่า r คือ **การแวะผ่านแบบหลังลำดับ** (postorder traversal) ของ T กรณีอื่นๆ สมมติว่า $T_1, T_2, \ldots, T_n$ คือ ต้นไม้ส่วนย่อย ที่ r จาก ซ้ายไปขวา การแวะผ่านแบบหลังลำดับ เริ่มต้นด้วย การแวะผ่าน T_1 แบบหลังลำดับ จากนั้น T_2 แบบหลังลำดับ เช่นนี้เรื่อยไป จนถึง T_n แบบหลังลำดับ และสุดท้าย การเยี่ยม r ในรูปที่ 7 แสดงให้เห็นว่า การแวะผ่านแบบหลังลำดับ ทำให้สำเร็จได้อย่างไร



รูปที่ 7 การแวะผ่านแบบหลังลำดับ (Postorder Traversal)

ตัวอย่าง 4 จงหา อันดับ ของการแวะผ่านแบบหลังลำดับ เยี่ยมจุดต่างๆ ของ ต้นไม้รากแบบอันดับ T ซึ่งแสดงในรูปที่ 3

ผลเฉลย ขั้นตอนต่างๆ ของ การแวะผ่านแบบหลังลำดับ ของ ต้นไม้รากแบบอันดับ T แสดงให้เห็นในรูปที่ 8 การแวะผ่านแบบหลังลำดับ เริ่มต้นด้วย การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก b การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก c ซึ่งมีเพียงจุด c ตามด้วย การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก d และ สุดท้าย ตามด้วย ราก a

การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก b เริ่มต้นด้วย การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก e ตามด้วย f ตามด้วยราก b

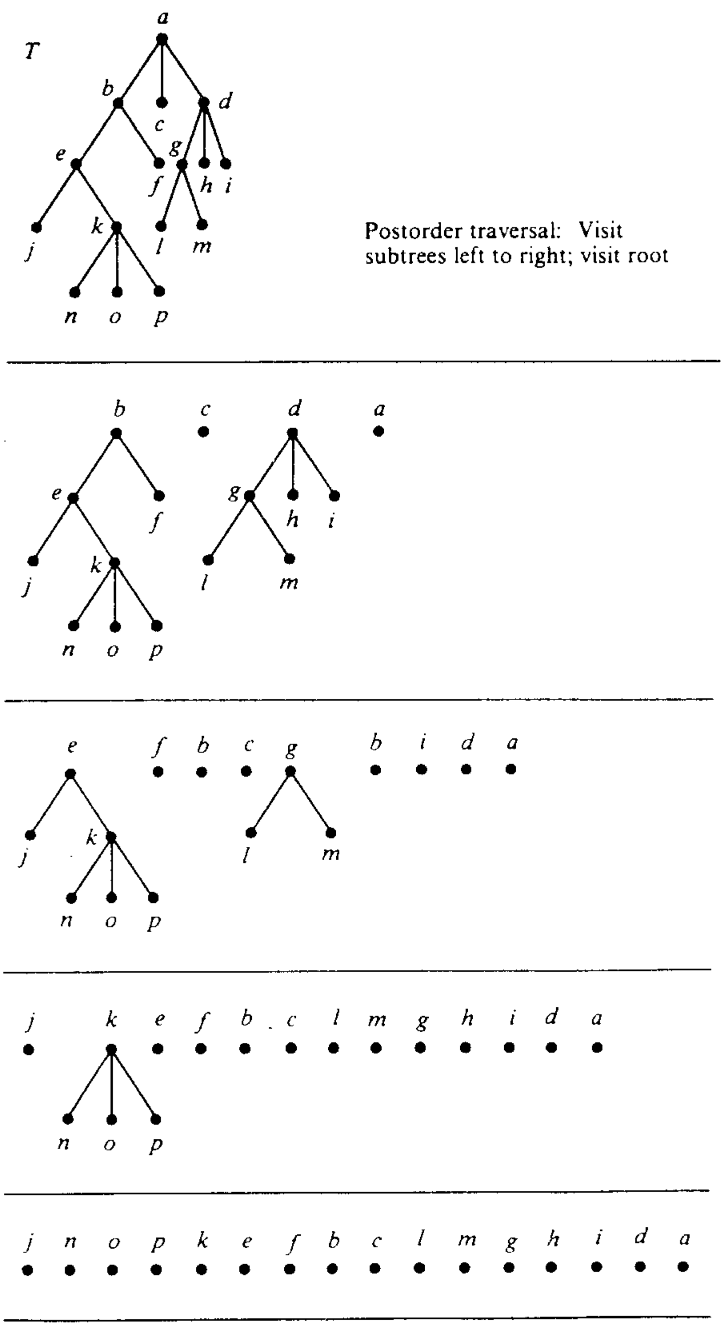
การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ราก ด้วยราก d เริ่มต้นด้วย การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก g ตามด้วย h ตามด้วย i และตามด้วยราก d

การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก e เริ่มต้นด้วย j ตามด้วย การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก k ตามด้วย ราก e

การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก g คือ l, m, g

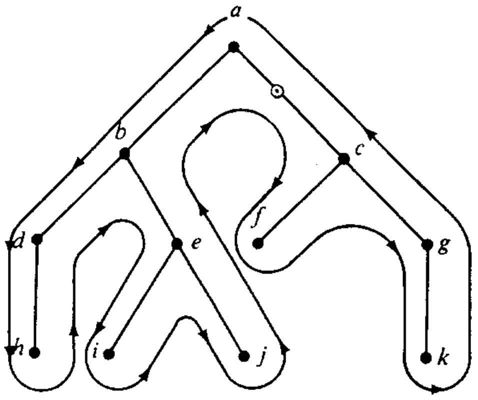
การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ส่วนย่อย ด้วยราก k คือ n, o, p, k

เพราะฉะนั้น การแวะผ่านแบบหลังลำดับ ของ ต้นไม้ T คือ j, n, o, p, k, e, f, b, c, l, m,g, h, i, d, a



รูปที่ 8 การแวะผ่านแบบหลังลำดับของ T

มีหลายวิธีง่ายๆ เพื่อแสดงรายการ จุดต่างๆ ของต้นไม้รากแบบอันดับ ในลักษณะ ก่อนลำดับ ตามลำดับ และหลังลำดับ ในการทำสิ่งนี้ ขั้นแรก วาดรูป เส้นโค้ง รอบ ต้นไม้รากแบบอันดับ เริ่มต้นที่ราก เคลื่อนย้ายไปตามด้านต่างๆ เช่นที่แสดงให้เห็นในรูปที่ 9 เราสามารถเขียนรายการจุดต่างๆ แบบก่อนลำดับ โดย แสดงรายการ แต่ละจุด ในครั้งแรกที่เส้นโค้งนี้ ผ่านจุดนั้น และ รายการแสดง แต่ละจุด ของจุดภายใน ในครั้งที่สอง ซึ่ง เส้นโค้งผ่านจุดนั้น เราสามารถเขียนรายการจุดต่างๆ แบบหลังลำดับ โดย รายการแสดง จุด ใน ครั้ง หลังสุด ที่ มันผ่าน บนทางย้อนกลับ ไป จุด ถึง จุดแม่ (parent) ของมัน เมื่อ สิ่งนี้กระทำเสร็จสิ้น ในรูปที่ 9 จะพบว่า



รูปที่ 9 วิธีลัด สำหรับการแวะผ่าน ต้นไม้รากแบบอันดับ แบบก่อนอันดับ ตามลำดับ และแบบหลังลำดับ

การแวะผ่านแบบก่อนลำดับ คือ a, b, d, h, e, i, j, c, f, g, k

การแวะผ่านแบบตามลำดับ คือ h, d, b, i, e, j, a, f, c, k, g

การแวะผ่านแบบหลังลำดับ คือ h, d, i, j, e, b, f, k, g, c, a

อัลกอริทึม สำหรับ การแวะผ่าน ต้นไม้รากแบบอันดับ ในลักษณะ ก่อนลำดับ ตามลำดับ หรือ หลังลำดับ แสดงให้เห็นง่ายที่สุด แบบเรียกซ้ำ

Algorithm 1 Preorder traversal

```
procedure preorder (T : ordered rooted tree)
r := root of T
list r
for each child c of r from left to right
begin
    T(c) := subtree with c as its root
    preorder (T(c))
end
```

Algorithm 2 Inorder traversal

```
procedure inorder (T : ordered rooted tree)
r := root of T
if r is a leaf then list r
eke
    begin
        1 := first child of r from left to right
        T(1) := subtree with 1 as its root
        inorder (T(l))
        list r
        for each child c of r except for 1 from left to right
            T(c) := subtree with c as its root
            inorder (T(c))
    end
```

Algorithm 3 Postorder traversal

procedure postorder (T : ordered rooted tree)

r := root of T

for each child c of r from left to right

begin

T(c) := subtree with c as its root

postorder (T(c))

end

list r

สัญกรณ์เติมกลาง เติมหน้า และเติมหลัง (Unfix, Prefix, and Postfix Notation)

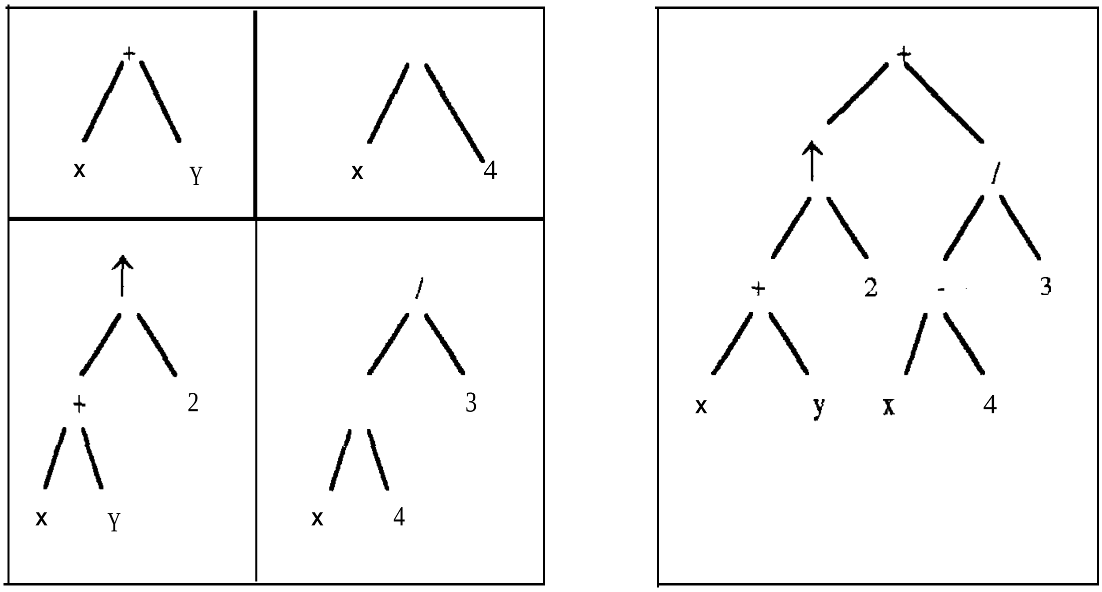
เราสามารถ แทน นิพจน์ซับซ้อน (complicated expressions) เช่น ประพจน์เชิงประกอบ (compound propositions) การจัดหมู่ของเซต (combinations of sets) และ นิพจน์คำนวณ (arithmetic expressions) โดยใช้ ต้นไม้รากแบบอันดับ ตัวอย่างเช่น จงพิจารณาการแทนที่ ของ นิพจน์คำนวณ เกี่ยวกับ ตัวปฏิบัติการ + (การบวก), - (การลบ), * (การคูณ), / (การหาร), $\uparrow$ (การยกกำลัง) เราจะใช้ เครื่องหมายวงเล็บ เพื่อแสดงถึง ลำดับของ การปฏิบัติการ ต้นไม้ราก แบบอันดับ สามารถนำมาใช้ เพื่อแทนนิพจน์เช่นนี้ได้ เมื่อ จุดภายใน แทน การปฏิบัติการ และ จุดใบ แทน ตัวแปร หรือ เลข การปฏิบัติการแต่ละชุด กระทำกับ ต้นไม้ส่วนย่อยซ้าย และ ต้นไม้ส่วนย่อยขวา ตามลำดับ

ตัวอย่าง 5 วงวาดรูปต้นไม้รากแบบอันดับ ซึ่งแทนนิพจน์

$$((x + y) \uparrow 2) + ((x - 4)/3)$$

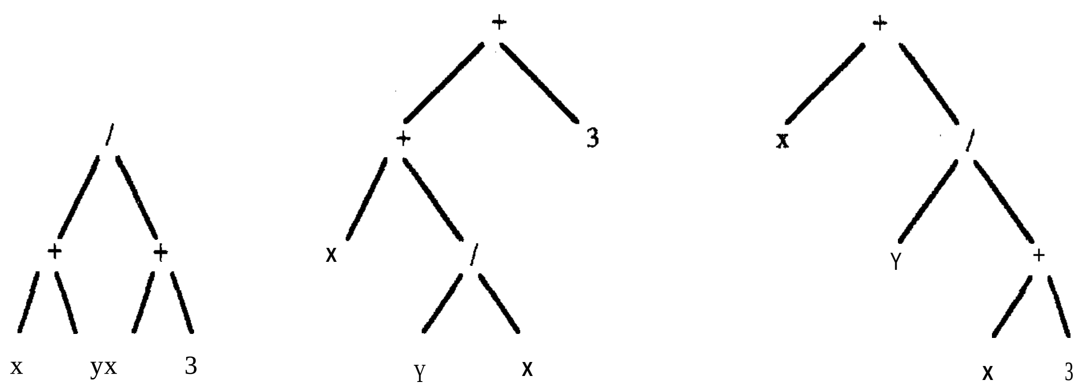
ผลเฉลย ต้นไม้แบบทวิภาค สำหรับนิพจน์นี้ สามารถสร้างจาก ล่างขึ้นบน (bottom up) ขั้นแรก สร้างต้นไม้ส่วนย่อย สำหรับ นิพจน์ x + y จากนั้น สิ่งนี้ รวมกัน เป็น ต้นไม้ส่วนย่อย ที่มีขนาดใหญ่ขึ้นแทน $(x + y) \uparrow 2$ เช่นเดียวกัน สร้างต้นไม้ส่วนย่อย สำหรับ x - 4 และรวมเข้า

เป็น ต้นไม้ส่วนย่อย แทน (x - 4)/3 สุดท้าย ต้นไม้ส่วนย่อย ซึ่งแทน (x + y) ↑ 2 และต้นไม้ส่วนย่อยซึ่งแทน (x - 4)/3 รวมเข้าด้วยกัน เป็น ต้นไม้รากแบบอันดับ เพื่อแทน ((x + y) ↑ 2 + ((x - 4)/3) ขั้นตอนเหล่านี้ แสดงให้เห็นในรูปที่ 10



รูปที่ 10 ต้นไม้แบบทวิภาค แทนนิพจน์ ((x + y) ↑ 2) = ((x - 4)/3)

การแวะผ่านแบบตามลำดับ ของต้นไม้แบบทวิภาค ซึ่งแทนนิพจน์ จะให้นิพจน์เดิม (original expression) ด้วยสมาชิกและการปฏิบัติการ ในลำดับ เหมือนเช่นเดียวกับ เมื่อมันเกิดตั้งแต่แรก ยกเว้น การดำเนินการ ชนิดเอกภาค (unary operation) ซึ่ง ตัวถูกดำเนินการ (operands) จะตามหลังตัวปฏิบัติการทันที ตัวอย่างเช่น การแวะผ่านแบบตามลำดับ ของ ต้นไม้แบบทวิภาคในรูปที่ 11 ซึ่งแทนนิพจน์ (x + y)/(x + 3), (x + (y/x)) + 3 และ x + (y/(x + 3)) ทั้งสามชุด นี้ นำไปสู่ นิพจน์เติมกลาง x + y/x + 3 เพื่อทำให้นิพจน์เช่นนี้ **ไม่กำกวม** (unambiguous) จึงมีความจำเป็นที่จะต้องใส่ เครื่องหมายวงเล็บ ในการแวะผ่านแบบตามลำดับ เมื่อใดก็ตามที่ พบการปฏิบัติการ นิพจน์ที่มีวงเล็บแบบเต็มรูป ซึ่งได้มาโดยวิธีนี้ เรียกว่า **รูปแบบเติมหน้า** (infix form)



รูปที่ 11 ต้นไม้ราก แทนนิพจน์ $(x + y)/(x + y)$, $(x + (y/x)) + 3$, และนิพจน์ $x + (y/(x + 3))$

เราได้ **รูปแบบเติมหน้า** (prefix form) ของนิพจน์ เมื่อเราแวะผ่าน ต้นไม้รากของมัน แบบก่อนลำดับ (preorder) นิพจน์ ซึ่งเขียนด้วย รูปแบบเติมหน้า เรียกว่า **สัญกรณ์โพลิช** (Polish notation) นิพจน์ ในรูปแบบเติมหน้า (ซึ่งการดำเนินการ แต่ละชุด จะมี ตัวถูกดำเนินการ จำนวนคงที่) จะไม่กำกวม ดังนั้น นิพจน์เช่นนี้ จึงไม่จำเป็นต้องมีเครื่องหมายวงเล็บกำกับ

ตัวอย่าง 6 จงหารูปแบบเติมหน้า (prefix form) ของนิพจน์

$$((x + y) \uparrow 2) + ((x - 4)/3)$$

ผลเฉลย เราได้รูปแบบเติมหน้า ของนิพจน์ นี้ โดยการแวะผ่านต้นไม้แบบทวิภาค ซึ่งแทนนิพจน์ แสดงให้เห็น ในรูปที่ 10 ผลลัพธ์คือ $+ \uparrow + x\ y\ 2\ /\ - x\ 4\ 3$

ในรูปแบบเติมหน้า ของ นิพจน์ ตัวดำเนินการทวิภาค เช่น + จะอยู่หน้า ตัวถูกดำเนินการสองตัว ของมัน ดังนั้น เราสามารถประเมินผล นิพจน์ ในรูปแบบเติมหน้า โดย ทำงานจาก ขวาไปซ้าย เมื่อพบตัวดำเนินการ หนึ่งตัว เรากระทำการดำเนินการซึ่งสมนัยกันกับ ตัวถูกดำเนินการสองตัว ทันที ซึ่งอยู่ทางขวาของ ตัวดำเนินการนี้ เมื่อใดก็ตาม เมื่อการดำเนินการ หนึ่งอย่าง ถูกกระทำ ผลลัพธ์จะเป็นตัวถูกดำเนินการตัวใหม่ (new operand)

ตัวอย่าง 7 จงหาค่าของนิพจน์เติมหน้า $+ - * 2\ 3\ 5\ / \uparrow 2\ 3\ 4$

ผลเฉลย

ขั้นตอนต่างๆ ซึ่งใช้ ประเมินผลนิพจน์ นี้ ทำจาก ขวาไปซ้าย และ กระทำการดำเนินการ โดยใช้ ตัวถูกกระทำ ทางขวา ซึ่งแสดงให้เห็น ในรูปที่ 12 ค่าของนิพจน์นี้ คือ 3

$+ \; - \; * \; 2 \; 3 \; 5 \; / \uparrow \underbrace{2 \; 3}_{} \; 4$

$2 \uparrow 3 = 8$

$+ \; - \; * \; 2 \; 3 \; 5 \; / \; 8 \; 4$

$8/4 = 2$

$+ \; - \; * \; 2 \; 3 \; 5 \; 2$

$2*3=6$

$+ \; - \; 6 \; 5 \; 2$

$6-5=1$

$+ \; 1 \; 2$

$1+2=3$

ค่าของนิพจน์ คือ 3

รูปที่ 12 การประเมินผลนิพจน์เติมหน้า (Evaluating a prefix expression)

เราได้**รูปแบบเติมหลัง** (postfix form) ของนิพจน์ โดยการ แวะผ่านต้นไม้แบบทวิภาค ของมัน แบบหลังลำดับ นิพจน์ ซึ่ง เขียนด้วย รูปแบบเติมหลัง เรียกว่า **สัญกรณ์โพลิชผันกลับ** (Reverse Polish Notation)

นิพจน์ ใน สัญกรณ์โพลิชผันกลับ ไม่กำกวม ดังนั้น จึงไม่จำเป็นต้องใช้เครื่องหมายวงเล็บ

ตัวอย่าง 8 จงหารูปแบบเติมหลัง (postfix form) **ของนิพจน์** $((x + y) \uparrow 2) + (x - 4)/3)$

ผลเฉลย รูปแบบเติมหลัง ของนิพจน์ ได้มาโดยการแวะผ่านแบบหลังลำดับ ของต้นไม้แบบทวิภาค สำหรับนิพจน์นี้ แสดงให้เห็นในรูปที่ 10 ซึ่งให้นิพจน์เติมหลัง ดังนี้ x y + 2 ↑ x 4 - 3 / +

ในรูปแบบเติมหลัง ของ นิพจน์ ตัวดำเนินการทวิภาค จะตามหลัง (follow) ตัวถูกดำเนินการสองตัวของมัน ในการประเมินผล นิพจน์ จาก รูปแบบเติมหลัง ทำจากซ้ายไปขวา กระทำ การดำเนินการ เมื่อใดก็ตามที่มี ตัวดำเนินการหนึ่งตัวตามหลังตัวถูกดำเนินการสองตัว หลังจาก การดำเนินการ หนึ่งอย่าง ทำสำเร็จแล้ว ผลลัพธ์ของการดำเนินการนี้ จะเป็น ตัวถูกดำเนินการตัวใหม่ (new operand)

ตัวอย่าง 9 จงหาค่าของ นิพจน์เติมหลัง 7 2 3 * - 4 ↑ 9 3 / +

ผลเฉลย ขั้นตอนต่างๆ ซึ่ง ใช้ประเมินผลนิพจน์นี้ เริ่มต้น จากซ้าย และกระทำการดำเนินการให้ได้ผลลัพธ์ เมื่อมี ตัวถูกดำเนินการ สองตัว ตามด้วย ตัวดำเนินการหนึ่งตัว แสดงให้เห็น ในรูปที่ 13 ค่าของนิพจน์นี้ เท่ากับ 4

7 2 3 * - 4 ↑ 9 3 / +

2*3 = 6

7 6 - 4 ↑ 9 3 / +

7-6 = 1

1 4 ↑ 9 3 / +

1 ↑ 4 = 1

1 9 3 / +

9/3 = 3

1 3 +

1+3 = 4

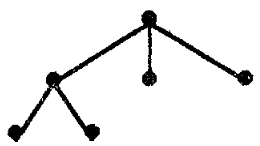
รูปที่ 13 การประเมินผลนิพจน์เติมหลัง (Evaluating a postfix expression)

เนื่องจาก นิพจน์เติมหน้า และ นิพจน์เติมหลัง ไม่กำกวม และเพราะว่า มันประเมินผลง่าย ไม่ต้องทำการ กราดตรวจ (scanning) กลับไป กลับมา นิพจน์เหล่านี้ จึงถูกนำมาใช้อย่างกว้างขวาง ใน วิชาคอมพิวเตอร์ นิพจน์เช่นนี้ มีประโยชน์โดยเฉพาะ ในการสร้างคอมไพเลอร์ (compilers)

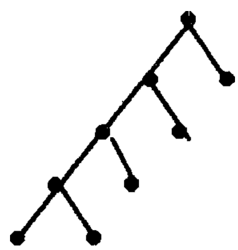
แบบฝึกหัด 7.3

ในแบบฝึกหัดข้อ 1 - 3 จงสร้างระบบการให้เลขที่อยู่แบบสากล สำหรับ ต้นไม้รากแบบอันดับที่กำหนดให้ จากนั้น ใช้สิ่งนี้ เพื่อเรียงอันดับจุดต่างๆ ของมัน โดยใช้ การเรียงอันดับ ตามตัวอักษร ของ labels ของมัน

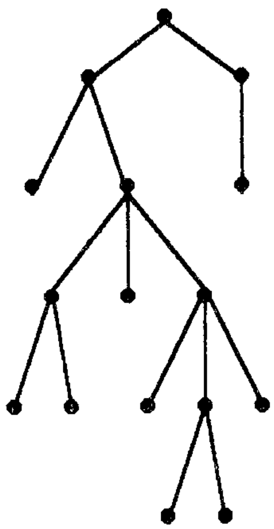
1



2.



3.



4. สมมติว่า เลขที่อยู่ ของ จุด v ใน ต้นไม้รากแบบอันดับ T คือ **3.4.5.2.4**

a) จุด v อยู่ที่ระดับ อะไร?

b) เลขที่อยู่ ของจุดแม่ ของ v คืออะไร?

c) เลขน้อยที่สุด ของ พี่น้อง ของ v ควรจะเป็นอะไร?

d) เลขที่เป็นไปได้ เล็กที่สุด ของ จุด ใน T ถ้า v มีเลขที่อยู่นี้ คืออะไร?

(What is the smallest possible number of vertices in T if v has this address?)

e) จงหาเลขที่อยู่อื่นๆ ที่ ต้องเกิดขึ้น

5. สมมติว่า จุด ที่มีเลขที่อยู่ใหญ่ที่สุด ใน ต้นไม้รากแบบอันดับ T มีเลขที่อยู่ **2.3.4.3.1** จะเป็นไปได้หรือไม่? ในการคำนวณหา จำนวนจุด ทั้งหมด ของ T

6. จุดใบต่างๆ ของ ต้นไม้รากแบบอันดับ จะมีรายการของเลขที่อยู่แบบสากล ข้างล่างนี้ ได้หรือไม่? ถ้าทำได้ จงสร้าง ต้นไม้รากแบบอันดับ เช่นนั้น

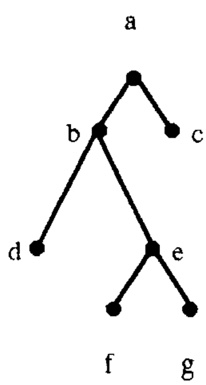
a) **1.1.1, 1.1.2, 1.2, 2.1.1.1, 2.1.2, 2.1.3, 2.2, 3.1.1, 3.1.2.1, 3.1.2.2, 3.2**

b) **1.1, 1.2.1, 1.2.2, 1.2.3, 2.1, 2.2.1, 2.3.1, 2.3.2, 2.4.2.1, 2.4.2.2, 3.1, 3.2.1, 3.2.2**

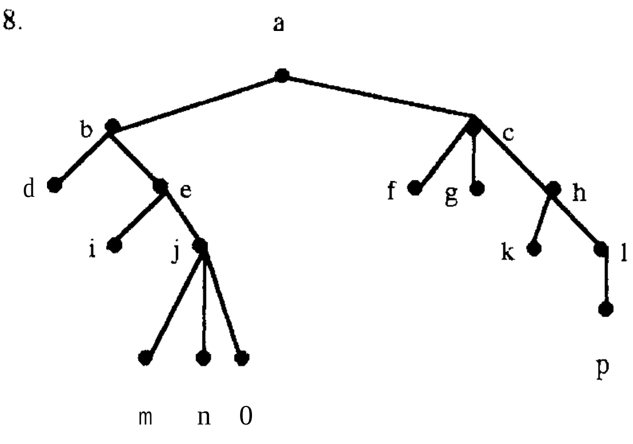
c) **1.1. 1.2.1, 1.2.2, 1.2.2.1, 1.3, 1.4, 2, 3.1, 3.2, 4.1.1.1**

ในแบบฝึกหัดข้อ 7 - 9 จงหา อันดับ ของ การแวะผ่าน แบบก่อนลำดับ เยี่ยมจุดต่างๆ ของต้นไม้รากแบบอันดับที่กำหนดให้

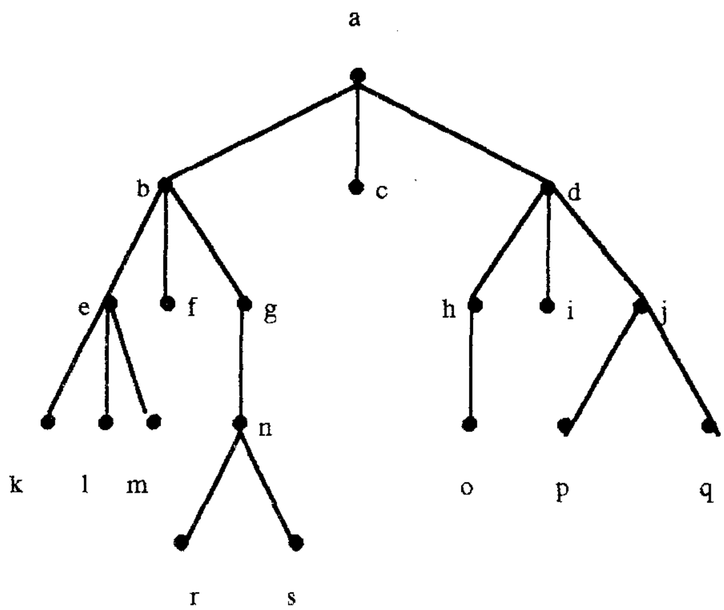
7.



8.



9.



10. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 7 ถูกเยี่ยมโดยการแวะผ่านแบบตามลำดับ

11. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 8 ถูกเยี่ยมโดยการแวะผ่านแบบตามลำดับ

12. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 9 ถูกเยี่ยมโดยการแวะผ่านแบบตามลำดับ

13. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 7 ถูกเยี่ยมโดยการแวะผ่านแบบหลังลำดับ

14. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 8 ถูกเยี่ยมโดยการแวะผ่านแบบหลังลำดับ

15. จงหา อันดับ ของ จุดต่างๆ ของ ต้นไม้รากแบบอันดับ ในแบบฝึกหัดข้อ 9 ถูกเยี่ยมโดยการแวะผ่านแบบหลังลำดับ

16. จงแทนนิพจน์ $((x + 2) \uparrow 3) * (y - (3 + x)) - 5$ โดยใช้ ต้นไม้แบบทวิภาค

17. จงเขียน นิพจน์ ในแบบฝึกหัด ข้อ 16 ด้วย

a) สัญกรณ์เติมหน้า

b) สัญกรณ์เติมหลัง

c) สัญกรณ์เติมกลาง

18. จงแทน นิพจน์ $(x + xy) + (x/y)$ และ นิพจน์ $x + ((xy) + x)/y)$ โดยใช้ ต้นไม้แบบทวิภาค

19. จงเขียน นิพจน์ ในแบบฝึกหัด ข้อ 18 ด้วย

a) สัญกรณ์เติมหน้า

b) สัญกรณ์เติมหลัง

c) สัญกรณ์เติมกลาง

20. จงแทน $(A \cap B) - (A \cup (B - A))$ โดยใช้ ต้นไม้รากแบบอันดับ

21. จงเขียน นิพจน์ ในข้อ 20 ด้วย

a) สัญกรณ์เติมหน้า

b) สัญกรณ์เติมหลัง

c) สัญกรณ์เติมกลาง

22. จะมีกี่วิธี ที่ สายอักขระ $A \cap B - A \cap B - A$ ใส่วงเล็บให้ครบ เพื่อให้เป็น นิพจน์เติมกลาง

23. จงวาดรูป ต้นไม้รากแบบอันดับ ซึ่งสมนัยกับ นิพจน์คำนวณ เขียนด้วย สัญกรณ์เติมหน้า แต่ละชุด ข้างล่างนี้ จากนั้น ให้เขียน นิพจน์ แต่ละชุด โดยใช้ สัญกรณ์เติมกลาง

a) + * + - 5 3 2 1 4

b) $\uparrow$ + 2 3 - 5 1

c) * / 9 3 + * 2 4 - 7 6

24. จงหาค่าของ นิพจน์เติมหน้า แต่ละชุด ข้างล่างนี้

a) - * 2 / 8 4 3

b) $\uparrow$ - * 3 3 * 4 2 5

c) + - $\uparrow$ 3 2 $\uparrow$ 2 3 / 6 - 4 2

d) * + 3 + 3 $\uparrow$ 3 + 3 3 3

25. จงหาค่าของ นิพจน์เติมหลัง แต่ละชุด ข้างล่างนี้

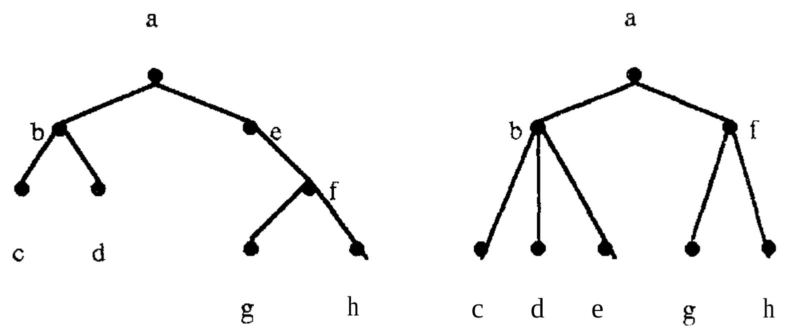
a) 5 2 1 - - 3 1 4 + + *

b) 9 3 / 5 + 7 2 - *

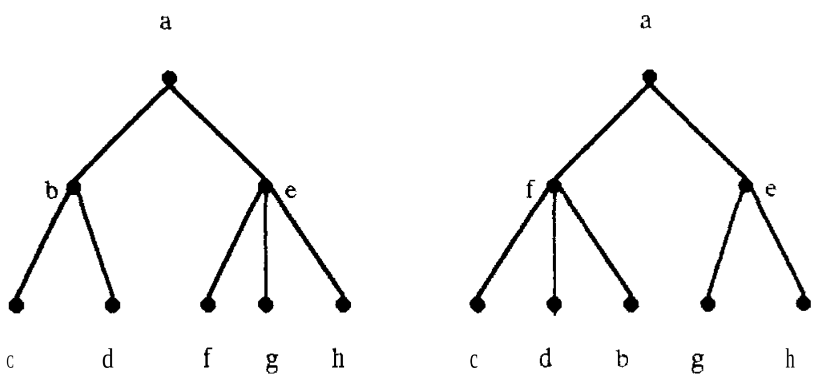
c) 3 2 * 2 $\uparrow$ 5 3 - 8 4 / * -

26. จงสร้างต้นไม้รากแบบอันดับ ซึ่งการแวะผ่านแบบก่อนลำดับของมันคือ a, b, f, c, g, h, i, d, e, j, k, l เมื่อ a มีลูกสี่คน, c มีลูกสามคน, j มีลูกสองคน, b และ e แต่ละจุด มีลูกหนึ่ง และจุดอื่น เป็น จุดใบทั้งหมด

27. จงแสดงให้เห็นว่า การแวะผ่านแบบก่อนลำดับ (preorder traversals) ของ ต้นไม้รากแบบอันดับ สองชุดที่แสดง ข้างล่างนี้ ให้ รายการของจุดเหมือนกัน



28. จงแสดงให้เห็นว่า การแวะผ่านแบบหลังลำดับ (postorder traversals) ของ ต้นไม้รากแบบอันดับ สองชุด ที่แสดงข้างล่างนี้ ให้ รายการของจุดเหมือนกัน



7.4 ต้นไม้ และ การเรียงลำดับ (Trees and Sortig)

ปัญหาของการเรียงอันดับ สมาชิก ใน เซต เกิดขึ้น ใน หลายๆ บริบท (contexts) ตัวอย่าง เช่น การพิมพ์ สารบบของโทรศัพท์ (telephone directory) จำเป็นจะต้องเรียงลำดับตามตัวอักษร ชื่อของสมาชิก

สมมติว่า มีการเรียงอันดับ ของ สมาชิกทั้งหมด เริ่มต้น สมาชิกในเซต จะอยู่ในอันดับ อย่างไรก็ได้ **การเรียงลำดับ** หมายถึง การเรียงอันดับใหม่ ของ สมาชิกเหล่านี้ ให้เป็น รายการ ซึ่ง สมาชิกอยู่ในอันดับ จาก น้อยไปหามาก

(A **sorting** is a reordering of these elements into a list in which the elements are in increasing order.)

ตัวอย่างเช่น การเรียงลำดับ รายการ 7, 2, 1, 4, 5, 9 จะให้ รายการ 1, 2, 4, 5, 7, 9

การเรียงลำดับรายการ d, h, c, a, f (ใช้การเรียงอันดับตามตัวอักษร) จะให้รายการ a, c, d, f, h

เปอร์เซนต์จำนวนมาก ของ การใช้คอมพิวเตอร์ สละให้กับการเรียงลำดับ หนึ่งสิ่ง หรือ อีกหนึ่งสิ่ง ดังนั้น จึงมีความพยายามอย่างมากได้อุทิศให้กับการพัฒนา อัลกอริทึมการ เรียงลำดับที่มีประสิทธิภาพ ในหัวข้อนี้ จะได้อภิปรายถึง อัลกอริทึมการเรียงลำดับ หลายชุด และความซับซ้อนของการคำนวณของมัน จะได้เห็นว่า ต้นไม้ถูกนำมาใช้อธิบาย อัลกอริทึม การเรียงลำดับ และการนำมาใช้ในการวิเคราะห์ความซับซ้อนของมัน

ความซับซ้อนของการเรียงลำดับ (The complexity of sorting)

อัลกอริทึมการเรียงลำดับต่างๆ จำนวนมาก ได้มีการพัฒนาขึ้น ในการตัดสินใจว่า อัล-กอริทึมการเรียงลำดับ ชุดหนึ่ง มีประสิทธิภาพหรือไม่ จะดูที่ความซับซ้อนของมัน การใช้ต้นไม้เป็นตัวแบบขอบเขตล่าง สำหรับ ความซับซ้อนกรณีแย่ที่สุด (worst-case complexity) ของ อัลกอริทึมการเรียงลำดับ สามารถหาได้

มี n! วิธีของการเรียงอันดับที่เป็นไปได้ ของสมาชิก n ตัว เพราะว่า การเรียงสับเปลี่ยน n! วิธีนั้น แต่ละวิธี สมาชิกเหล่านี้ เรียงอันดับถูกต้อง อัลกอริทึมการเรียงลำดับ ซึ่งเราจะได้ศึกษานี้ วางอยู่บน การเปรียบเทียบแบบทวิภาค นั่นคือ การเปรียบเทียบสมาชิกครั้งละสองตัว ผลลัพธ์ของการเปรียบเทียบเช่นนี้ แต่ละชุด ทำให้ เซตของการเรียงอันดับที่เป็นไปได้แคบลง ดังนั้น อัลกอริทึมการเรียงลำดับ จึงมีพื้นฐาน บนการเปรียบเทียบแบบทวิภาค สามารถถูกแทนได้ด้วย ต้นไม้การตัดสินใจแบบทวิภาค ซึ่ง จุดภายในแต่ละจุด แทนการเปรียบเทียบของสมาชิก

สองตัว จุดใบ แต่ละจุด แทน หนึ่งใน วิธีเรียงสับเปลี่ยน n! วิธีของสมาชิก n ตัว

ตัวอย่าง 1 ในรูปที่ 1 แสดงให้เห็น ต้นไม้การตัดสินใจ ซึ่งเรียงอันดับ สมาชิกของรายการ a, b, c

ความซับซ้อนของการเรียงอันดับ ยึดพื้นฐาน การเปรียบเทียบแบบทวิภาค วัด (measured) ในเทอมของ จำนวนการใช้การเปรียบเทียบเช่นนั้น

การเปรียบเทียบแบบทวิภาค มากที่สุด ซึ่งจำเป็น ในการเรียงลำดับรายการ ที่มีสมาชิก n ตัว ให้ความสามารถ กรณีแย่ที่สุด ของ อัลกอริทึม การเปรียบเทียบที่ใช้มากที่สุด เท่ากับ ความยาวของทางเดิน ยาวที่สุด ใน ต้นไม้การตัดสินใจซึ่งแทน กระบวนงานการเรียงลำดับ

(The most comparisons used equals to the longest path length in the decision tree representing the sorting procedure.)

พูดอีกอย่างหนึ่งคือ การเปรียบเทียบมากที่สุดที่จำเป็น เท่ากับ ความสูงของต้นไม้ตัดสินใจ เนื่องจาก ความสูงของต้นไม้แบบทวิภาค ที่มีจุดใบ n! จุด คือ การเปรียบเทียบ น้อยที่สุด $\lceil \log n! \rceil$ ครั้ง

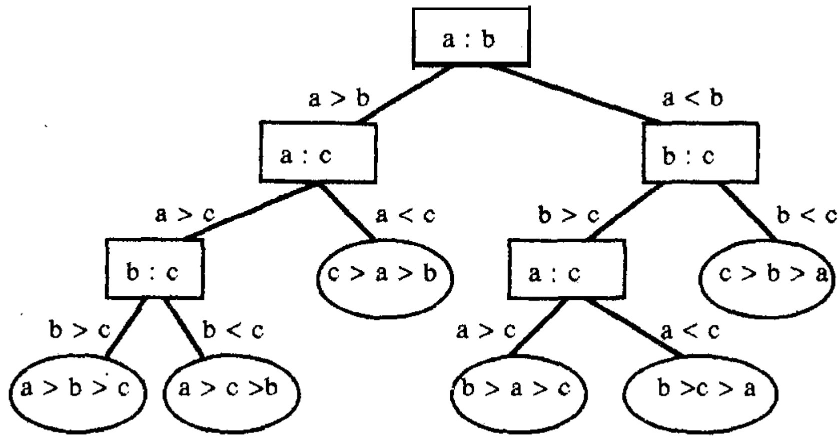
ทฤษฎีบท 1 อัลกอริทึมการเรียงลำดับ ขึ้นอยู่กับ การเปรียบเทียบแบบทวิภาค ที่จำเป็นที่ต้องใช้อย่างน้อยที่สุด $\lceil \log n! \rceil$ ครั้ง

(A sorting algorithm based on binary comparisons requires at least $\lceil \log n! \rceil$ comparison.) [1]

เนื่องจาก $\lceil \log n! \rceil$ เท่ากับ $O(n \log n)$ เพราะว่า $\log n!$ มีค่ามากกว่า $(n \log n)/4$ สำหรับ $n > 4$ จะได้ว่า ไม่มี อัลกอริทึมการเรียงลำดับใดๆ ซึ่ง ใช้การเปรียบเทียบ เป็นวิธีการของการเรียงลำดับ มี ความซับซ้อน กรณีแย่ที่สุด ซึ่งดีกว่า $O(n \log n)$ เพราะฉะนั้น อัลกอริทึมการเรียงลำดับ จะมีประสิทธิภาพ เท่าที่เป็นไปได้ ถ้ามันมีความซับซ้อนของเวลา เท่ากับ $O(n \log n)$

---

[1] Rosen, หน้า 544



รูปที่ 1 ต้นไม้การตัดสินใจ สำหรับการเรียงลำดับ สมาชิกที่แตกต่างกัน สามสิ่ง

การเรียงลำดับแบบฟอง (The bubble sort)

การเรียงลำดับแบบฟอง หมายถึง อัลกอริทึมการเรียงลำดับง่ายที่สุด วิธีหนึ่ง แต่ ไม่ใช่วิธีที่มีประสิทธิภาพมากที่สุด วิธีนี้ ใส่รายการ ให้ เรียงลำดับ จากน้อยไปหามาก โดย การเปรียบเทียบสมาชิกชิดกัน (adjacent elements) อย่างสืบเนื่อง ถ้าสมาชิกสองตัวนี้ เรียงอันดับผิด ให้สลับที่กัน การเรียงอันดับแบบฟอง ทำให้ประสบผลสำเร็จ เรากระทำ การดำเนินการพื้นฐาน นั่นคือ สลับที่กันระหว่าง สมาชิกตัวที่มีค่ามากกว่า กับ สมาชิกตัวที่ชิดกันและมีค่าน้อยกว่ามัน เริ่มต้นที่ จุดแรกของรายการ สำหรับการผ่านเต็มแบบ ทำซ้ำกระบวนงานนี้ จนกระทั่ง การเรียงลำดับเสร็จสิ้น เราสามารถนึกภาพ สมาชิกต่างๆ ในรายการวางในสดมภ์

ในการเรียงลำดับแบบฟอง สมาชิกตัวที่มีค่าน้อยกว่า ลอยขึ้น (bubble) ตอนบน ขณะที่สลับที่กับสมาชิกตัวที่มีค่ามากกว่า สมาชิกตัวที่มีค่ามากกว่า จมลง (sink) ไปตอนล่าง สิ่งนี้ แสดงให้เห็น ในตัวอย่างต่อไปนี้

ตัวอย่าง 2 จงใช้การเรียงลำดับแบบฟอง ใส่เลข 3, 2, 4, 1, 5 เรียงอันดับ จากน้อยไปหามาก

ผลเฉลย

เริ่มต้น โดยการเปรียบเทียบ สมาชิก สองตัวแรก 3 และ 2 เนื่องจาก $3 > 2$ สลับที่กันระหว่าง 3 และ 2 ให้รายการเป็น 2, 3, 4, 1, 5 เนื่องจาก $3 < 4$ ทำต่อไป โดยการเปรียบเทียบ 4 กับ 1 เนื่องจาก $4 > 1$ สลับที่กันระหว่าง 1 และ 4 ให้รายการเป็น 2, 3, 1, 4, 5 เนื่องจาก $4 < 5$ การผ่านครั้งที่ 1 จึงเสร็จ การผ่านครั้งที่หนึ่ง รับประกันว่า สมาชิกตัวที่มีค่ามากที่สุด คือ 5 อยู่ในตำแหน่ง ถูกต้อง

การผ่านครั้งที่สอง เริ่มต้นโดยการเปรียบเทียบ 2 และ 3 เนื่องจาก เลขสองตัวนี้ เรียงอันดับถูกต้องแล้ว เปรียบเทียบ 3 และ 1 เพราะว่า 3 > 1 ให้สลับที่ เลขสองตัวนี้ ผลลัพธ์คือ 2, 1, 3, 4, 5 เพราะว่า 3 < 4 เลขสองตัวนี้ จึงเรียงอันดับถูกต้องแล้ว ไม่จำเป็นต้องทำการเปรียบเทียบอีกต่อไป เพราะว่า 5 อยู่ในตำแหน่งถูกต้องไปแล้ว การผ่านครั้งที่สอง รับประกัน สมาชิกสองตัวที่ใหญ่ที่สุด 4 และ 5 อยู่ในตำแหน่งถูกต้อง ของมัน

การผ่านครั้งที่สาม เริ่มต้น โดยการเปรียบเทียบ 2 และ 1 เนื่องจาก 2 > 1 ให้สลับที่เลขสองตัวนี้ ผลลัพธ์คือ 1, 2, 3, 4, 5 เพราะว่า 2 < 3, เลขสองตัวนี้ อยู่ในอันดับถูกต้อง ไม่จำเป็นต้อง *เปรียบเทียบต่อไป สำหรับการผ่านครั้งนี้ สำหรับการผ่านครั้งนี้ เพราะว่า 4 และ 5* อยู่ในตำแหน่งถูกต้องไปแล้ว การผ่านครั้งที่สาม รับประกันว่า เลขใหญ่ที่สุด สามตัว 3, 4 และ 5 อยู่ในตำแหน่ง ถูกต้องของมัน

การผ่านครั้งที่สี่ ประกอบด้วยการเปรียบเทียบครั้งเดียว คือ เปรียบเทียบ 1 และ 2 เนื่องจาก 1 < 2 สมาชิกสองตัวนี้ อยู่ในอันดับถูกต้อง การเรียงลำดับแบบฟอง จึงเสร็จสมบูรณ์ ขั้นตอนของอัลกอริทึมนี้ แสดงให้เห็นในรูปที่ 2

First pass

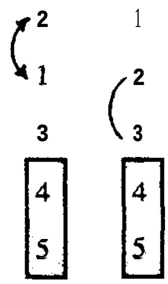
3	2	2	2
2	3	3	3
4	4	4	1
1	1	1	4
5	5	5	5

Second pass

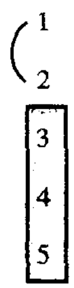
2	2	2
3	3	1
1	1	3
4	4	4
$\boxed{5}$	$\boxed{5}$	$\boxed{5}$

รูปที่ 2 ขั้นตอนของการเรียงลำดับแบบฟอง

Third pass



Fourth pass



: an interchange

: pair in correct order

: numbers in shade guaranteed in
to be in correct order

รูปที่ 2 (ต่อ) ขั้นตอนของการเรียงลำดับแบบฟอง

การอธิบาย การเรียงลำดับแบบฟอง ด้วย รหัสเทียม กำหนดให้แล้ว ในอัลกอริทึม 1 การเรียงลำดับแบบฟองมีประสิทธิภาพอย่างไร? เนื่องจาก มีการใช้การเปรียบเทียบ n - 1 ครั้ง ระหว่าง ครั้งที่ i จำนวนการเปรียบเทียบ ทั้งหมด ที่ใช้ในการเรียงลำดับแบบฟอง ของรายการที่มีสมาชิก n ตัว คือ

$$(n - 1) + (n - 2) + . . . + 2 + 1$$

สิ่งนี้ คือ ผลบวกสะสม ของ จำนวนเต็ม เล็กที่สุด n - 1 ตัว ซึ่งมีค่าเท่ากับ (n - 1)n / 2 เพราะฉะนั้น การเรียงลำดับแบบฟอง ใช้การเปรียบเทียบ n(n-1) / 2 ครั้ง ในการเรียงอันดับรายการของสมาชิก n ตัว

Algorithm 1 The bubble sort

procedure bubblesort ($a_1, \ldots, a_n$)

for i := **to** n - 1

begin

  for j := 1 **to** n - i

    if $a_j > a_{j+1}$ **then** interchange a_j and a_{j+1}

end

{$a_1, \ldots, a_n$ is in increasing order}

โปรดสังเกตว่า การเรียงลำดับแบบฟอง จะใช้การเปรียบเทียบ มากเท่ากับจำนวนนี้เสมอ เพราะว่า มันทำสืบเนื่อง แม้ว่า รายการ จะเรียงลำดับเสร็จแล้ว ที่ ขั้นตอนกลางบางแห่ง ดังนั้น อัลกอริทึมการเรียงลำดับแบบฟอง มีความซับซ้อนแย่ที่สุด เท่ากับ $O(n^2)$

เนื่องจาก สำหรับ จำนวนจริงบวกทุกตัว c, $n(n-1)/2 > cn \log n$ สำหรับ จำนวนเต็มบวก n ขนาดใหญ่ พอเพียง บางตัว จะได้ว่า การเรียงลำดับแบบฟอง จะไม่มีความซับซ้อนของเวลา กรณีแย่ที่สุด $O(n \log n)$ เราจำเป็น ต้องหาอัลกอริทึม อีกวิธีหนึ่ง เพื่อทำให้ การประมาณค่าที่เหมาะสม ของความซับซ้อน กรณีแย่ที่สุด ประสบผลสำเร็จ

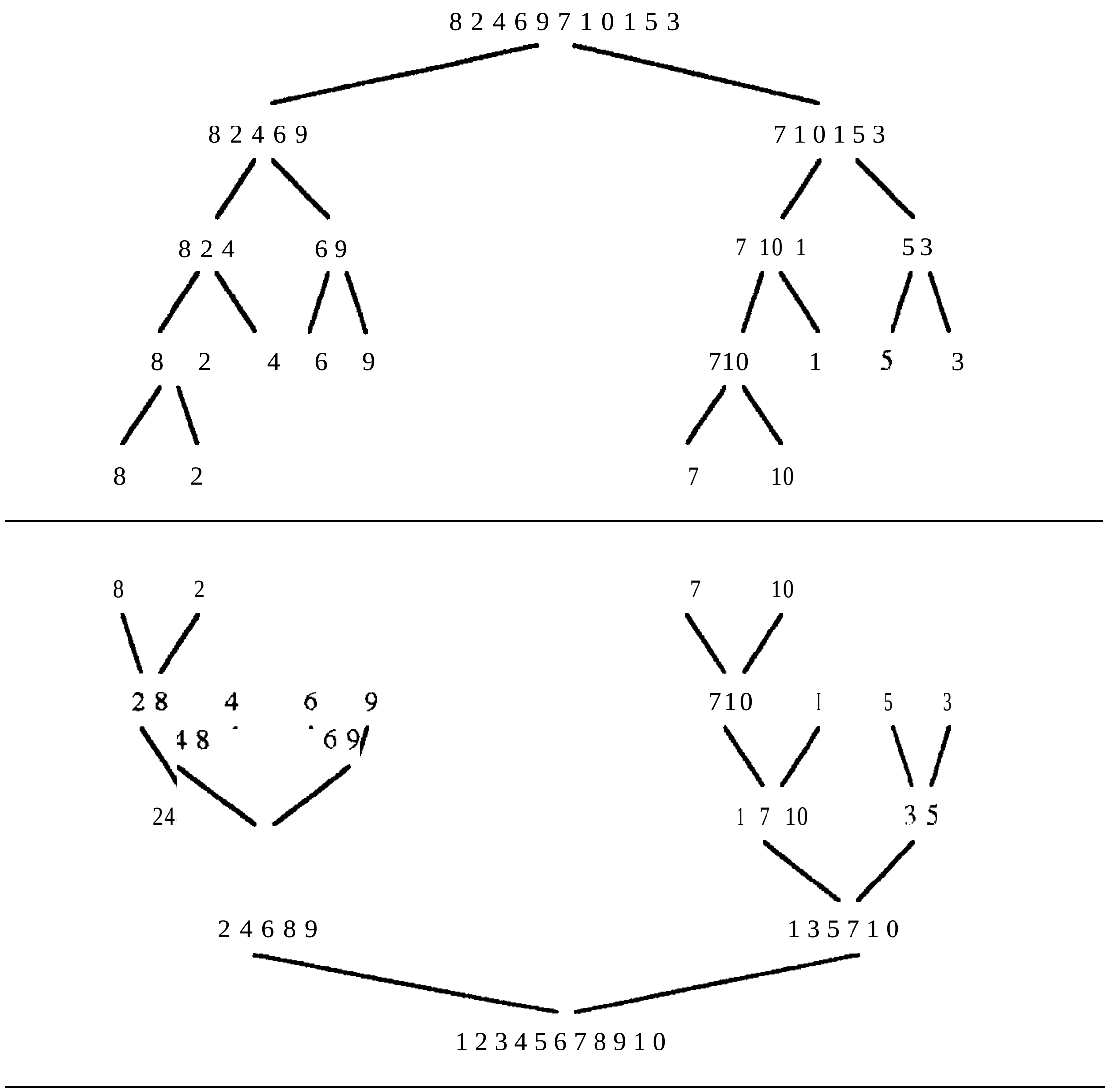
การเรียงลำดับแบบผสาน (The merge sort)

อัลกอริทึมการเรียงลำดับที่แตกต่างกัน จำนวนมาก ประสบผลสำเร็จ ความซับซ้อนกรณีแย่ที่สุด ที่เป็นไปได้ ดีที่สุด สำหรับ อัลกอริทึมการเรียงลำดับ คือ การเปรียบเทียบ $O(n \log n)$ กับการเรียงลำดับสมาชิก n ตัว

เราจะอธิบายอัลกอริทึม เช่นนี้ หนึ่งชุด เรียกว่า อัลกอริทึม**เรียงลำดับแบบผสาน** โดยแสดงให้เห็นว่า อัลกอริทึมนี้ ทำงานอย่างไร ด้วยตัวอย่าง จากนั้น จึงอธิบาย กรณีทั่วไป

ตัวอย่าง 3 ต้องการเรียงลำดับรายการ 8, 2, 4, 6, 9, 7, 10, 1, 5, 3 โดยใช้การเรียงลำดับแบบผสาน การเรียงลำดับแบบผสาน เริ่มต้นโดยการแบ่งรายการข้อมูล ให้เป็น สองชุด อย่างสืบเนื่อง ความก้าวหน้าของรายการย่อย สำหรับตัวอย่างนี้ ถูกแทนด้วย ต้นไม้ทวิภาคแบบได้ดุล (balanced binary tree) ของความสูง เท่ากับ 4 ในครึ่งบน ของรูปที่ 3

การเรียงลำดับ กระทำ โดย การผสานอย่างสืบเนื่อง คู่ ของรายการ ที่ขั้นตอนที่หนึ่ง คู่ของสมาชิกแต่ละตัว ผสานเข้าด้วยกัน ให้เป็นรายการความยาวเท่ากับสอง เรียงอันดับ จาก น้อยไปหามาก จากนั้น ผสานอย่างสืบเนื่อง คู่ ของ รายการ จนกระทั่ง รายการรวมทั้งหมด เรียงอันดับ จากน้อยไปหามาก รายการผสาน ซึ่งประสบผลสำเร็จ เรียงอันดับจากน้อย ไปหามาก ถูกแทนด้วย ต้นไม้ทวิภาคแบบได้ดุล ความสูง 4 แสดง ในครึ่งล่าง ของรูปที่ 3 (โปรดสังเกตว่า ต้นไม้ต้นนี้ แสดงให้เห็นแบบกลับด้าน)



รูปที่ 3 การเรียงลำดับแบบผสาน ของ 8, 2, 4, 6, 9, 7, 10, 1, 5, 3

กรณีทั่วไป การเรียงลำดับแบบผสาน กระทำโดยแบ่งรายการข้อมูล ซ้ำๆ กัน ให้เป็นรายการย่อย สองชุด ที่มีความยาวเท่ากัน (หรือ มีรายการย่อยหนึ่งชุด มี สมาชิกมากกว่า รายการย่อยอีกหนึ่งชุด เพียง หนึ่งตัว) จนกระทั่ง รายการย่อย แต่ละชุด มีสมาชิก หนึ่งตัว ความสำเร็จ ของรายการย่อย แทนด้วย ต้นไม้ทวิภาคแบบได้ดุล กระบวนงาน ทำต่อไป โดย ผสานอย่างสืบเนื่องคู่ ของรายการ เมื่อ รายการทั้งคู่ เรียงอันดับ จากน้อยไปหามาก ให้เป็นรายการขนาดใหญ่ขึ้นสมาชิกเรียงอันดับจากน้อยไปหามาก จนกระทั่ง รายการเดิม ถูกเรียงอันดับจากน้อยไปหามาก ความสำเร็จ ของรายการผสาน ถูกแทนด้วย ต้นไม้ทวิภาคแบบได้ดุล

เราสามารถอธิบาย การเรียงลำดับแบบผสาน อย่างเรียกซ้ำ ในการทำสิ่งนี้ ให้แบ่ง รายการข้อมูล ออกเป็นรายการย่อย สองชุด ขนาดเท่ากัน หรือ เกือบจะเท่ากัน เรียงลำดับ รายการย่อยแต่ละชุด โดยใช้ อัลกอริทึม การเรียงลำดับแบบผสาน จากนั้นผสานรายการสองชุดเข้าด้วยกัน

อัลกอริทึมที่มีประสิทธิภาพ สำหรับการผสาน รายการแบบอันดับ สองชุด ให้เป็น รายการแบบอันดับ หนึ่งชุด มีขนาดใหญ่ขึ้น คือ สิ่งจำเป็น ในการทำให้ การเรียงลำดับแบบผสาน เกิดผลในทางปฏิบัติ

ตัวอย่าง 4 เราจะอธิบาย ว่า การผสาน รายการสองชุด คือ 2, 3, 5, 6 และ 1, 4 ทำได้อย่างไร ตารางที่ 1 แสดงให้เห็น ขั้นตอนต่างๆ ที่เราใช้

ตารางที่ 1 การผสาน รายการ 2, 3, 5, 6 และ 1, 4

รายการที่ 1	รายการที่ 2	รายการผสาน	การเปรียบเทียบ
2 3 5 6	1 4		1 < 2
2 3 5 6	4	1	2 < 4
3 5 6	4	1 2	3 < 4
5 6	4	1 2 3	4 < 5
5 6		1 2 3 4	
		1 2 3 4 5 6	

ขั้นแรก เปรียบเทียบ สมาชิก ตัวเล็กที่สุด ใน รายการทั้งสองชุด 2 และ 1 ตามลำดับ เพราะว่า 1 เป็นตัวเล็กกว่า ใส่ 1 ที่ตอนต้นของรายการผสาน และ ลบมัน ออกจากรายการที่สอง

ณ ขั้นตอนนี้ รายการแรกคือ 2, 3, 5, 6 ส่วน รายการที่สองคือ 4 และรายการรวม (combined list) คือ 1 ต่อไป เปรียบเทียบ 2 และ 4, สมาชิกตัวเล็กที่สุด ของ สองรายการ เพราะว่า 2 เป็นตัวเล็กกว่า ใส่ 2 ในรายการรวม และ ลบมันออกจากรายการที่หนึ่ง ณ ขั้นตอนนี้ รายการที่หนึ่ง คือ 3, 5, 6 รายการที่สอง คือ 4 และ รายการรวม คือ 1, 2

ต่อไป เปรียบเทียบ 3 และ 4, สมาชิกตัวเล็กที่สุด ของรายการ สองชุด เนื่องจาก 3 เป็นตัวเล็กกว่า ใน สมาชิก สองตัวนี้ ใส่ 3 ในรายการรวม และลบมันออกจาก รายการที่หนึ่ง ณ ขั้นตอนนี้ รายการที่หนึ่งคือ 5, 6 ส่วนรายการที่สองคือ 4, รายการรวมคือ 1, 2, 3

จากนั้น เปรียบเทียบ 5 และ 4, สมาชิกตัวเล็กที่สุดใน สองรายการ เนื่องจาก 4 คือตัวเล็กกว่า ของ สมาชิก สองตัวนี้ ใส่ มัน ใน รายการรวม และลบมันออกจากรายการที่สอง ณ ขั้นตอนนี้ รายการที่หนึ่งคือ 5, 6 รายการที่สอง ว่าง (empty) ส่วนรายการรวมคือ 1, 2, 3, 4

ในที่สุด เนื่องจากรายการที่สอง ว่าง สมาชิกทั้งหมดของรายการที่หนึ่ง นำไปใส่เพิ่มตอนท้าย ของ รายการรวม ในอันดับเช่นเดียวกับที่มันอยู่ในรายการแรก สิ่งนี้ ให้ รายการแบบอันดับ

1, 2, 3, 4, 5, 6

ต่อไปเราจะพิจารณา ปัญหาทั่วไปของการผสาน รายการแบบอันดับ สองชุด L_1 และ L_2 ให้เป็นรายการแบบอันดับ L_3 เราสามารถใช้กระบวนงานต่อไปนี้ เริ่มต้นด้วย รายการว่าง L เปรียบเทียบสมาชิก ตัวเล็กที่สุด ของ สองรายการ ใส่ สมาชิกตัวที่เล็กกว่า ของสมาชิกสองตัวนี้ที่ ตอนท้ายของ L ลบ มันออกจากรายการที่มันเคยอยู่ ต่อไป ถ้ารายการหนึ่งของ L_1 และ L_2 ว่าง เอารายการที่ไม่ว่าง ไปใส่เพิ่ม ตอนท้ายของ L การผสานจะเสร็จสมบูรณ์ แต่ถ้ารายการ L_1 และ L_2 ไม่มีตัวใด เป็น รายการว่าง ทำกระบวนการนี้ ซ้ำ อัลกอริทึม 2 ให้ รหัสเทียม อธิบายกระบวนงานนี้

อัลกอริทึม 2 Merging two lists

procedure merge (L_1, L_2 lists)

L := empty ıst

while L, and L_2 are both nonempty

begin

remove smaller of first element of L, and L_2 from the list it is in and put it at the end of IL

```
    if removal of this element makes one list empty then remove all elements from
      the other list and append them to L
end {L is the merged list with elements in increasing order}
```

เราจำเป็น จะต้องประมาณค่า จำนวนของการเปรียบเทียบ ที่ใช้เพื่อผสาน รายการแบบอันดับสองชุด ในการวิเคราะห์ของ การเรียงลำดับแบบผสาน จะเห็นได้ง่ายว่าในอัลกอริทึม 2 ทุกครั้ง ที่มีการเปรียบเทียบ สมาชิก หนึ่งตัว จาก L_1 และ สมาชิกหนึ่งตัวจาก L_2 มี สมาชิกเพิ่มอีกหนึ่งตัว ใส่ไปในรายการผสาน L เมื่อใดก็ตามที่ L_1 หรือ L_2 ว่าง ไม่จำเป็นต้องเปรียบเทียบอีกต่อไป ดังนั้น อัลกอริทึม 2 จะมีประสิทธิภาพ อย่างน้อยที่สุด เมื่อเปรียบเทียบ $m + n - 2$ ครั้ง โดยที่เหลือสมาชิกหนึ่งตัว ใน รายการ L_1 และ อีกหนึ่งตัวใน L_2 การเปรียบเทียบครั้งต่อไปจะเป็นตัวสุดท้าย เพราะว่า มันจะทำให้ รายการชุดหนึ่ง ว่าง เพราะฉะนั้น อัลกอริทึม 2 จะใช้การเปรียบเทียบ ไม่เกิน $m + n - 2$ ครั้ง

ทฤษฎีประกอบ รายการแบบเรียงลำดับ สองชุด ที่มีสมาชิก m ตัว และสมาชิก n ตัว สามารถผสานกัน ให้เป็น รายการเรียงลำดับ โดยใช้การเปรียบเทียบ ไม่เกิน $m + n - 1$ ครั้ง

(Two sorted lists with an m elements and n elements can be merged into a sorted list using no more than $m + n - 1$ com[arisons.) [2]

[2] Rosen, หน้า 550

แบบฝึกหัด 7.4

1. จงใช้การเรียงลำดับแบบฟอง เพื่อเรียงลำดับเลข 3, 1, 5, 7, 4 พร้อมทั้งแสดงให้เห็น รายการต่างๆ ที่เป็นผลลัพธ์ในแต่ละขั้นตอน
2. จงใช้การเรียงลำดับแบบฟอง เพื่อเรียงตัวอักษร d, f, k, m, a, b พร้อมทั้งแสดงให้เห็นรายการต่างๆ ที่เป็นผลลัพธ์ในแต่ละขั้นตอน
3. จงดัดแปลง (adapt) อัลกอริทึมการเรียงลำดับแบบฟอง เพื่อให้มันหยุด เมื่อไม่จำเป็นต้องสลับที่ใดๆ อีก แสดงให้เห็นว่า เวอร์ชัน ของ อัลกอริทึม ที่เขียนด้วยรหัสเทียมชุดนี้ มีประสิทธิภาพมากกว่า ชุดเดิม
4. จงใช้การเรียงลำดับแบบผสาน เพื่อเรียงลำดับเลข 4, 3, 2, 5, 1, 8, 7, 6 พร้อมทั้ง แสดงให้เห็นทุกขั้นตอน โดยใช้อัลกอริทึม
5. จงใช้การเรียงลำดับแบบผสาน เพื่อเรียงลำดับ ตัวอักษร b, d, a, f, g, h, z, p, o, k พร้อมทั้งแสดงให้เห็นทุกขั้นตอน โดยใช้อัลกอริทึม
6. จะมีการเปรียบเทียบ กี่ครั้ง ในการผสาน คู่ของรายการข้างล่างนี้ โดยใช้อัลกอริทึม 2

   a) 1, 3, 5, 7, 9 ; 2, 4, 6, 8, 10

   b) 1, 2, 3, 4, 5 ; 6, 7, 8, 9, 10

   c) 1, 5, 6, 7, 8 ; 2, 3, 4, 9, 10
7. จงคำนวณหา จำนวนการเปรียบเทียบ น้อยที่สุด ที่จำเป็นต้องใช้ ในการผสาน รายการสองชุด ซึ่ง เรียงอันดับ จากน้อยไปหามาก ให้เป็นรายการ หนึ่งชุด เรียงอันดับจากน้อยไปหามาก เมื่อจำนวนสมาชิก ใน รายการทั้งสองชุด คือ

   a) 1, 4 b) 2, 4 c) 3, 4 d) 4, 4

การเรียงลำดับแบบเลือก (Selection sort)

เริ่มต้นโดยการหา สมาชิกตัวที่ มีค่าน้อยที่สุด ในรายการ ให้ย้ายสมาชิกตัวนี้ ไปไว้ตอนหน้า จากนั้น จึงหา สมาชิกตัวที่มีค่าน้อยที่สุด ระหว่างสมาชิกส่วนที่เหลือ เมื่อพบแล้ว ใส่ไว้ในตำแหน่งที่สอง กระบวนงานนี้ ทำซ้ำๆ กัน จนกระทั่ง รายการทั้งหมด เรียงลำดับ

8. จงเรียงลำดับ รายการข้างล่างนี้ โดยใช้การเรียงลำดับแบบเลือก

   a) 3, 5, 4, 1, 2

   b) 5, 4, 3, 2, 1

   c) 1, 2, 3, 4, 5

9. จงเขียนอัลกอริทึม การเรียงลำดับแบบเลือก ด้วยรหัสเทียม

10. จะมีการเปรียบเทียบ กี่ครั้ง ที่กระทำ ในการเรียงลำดับแบบเลือก ของ ข้อมูล n ตัว

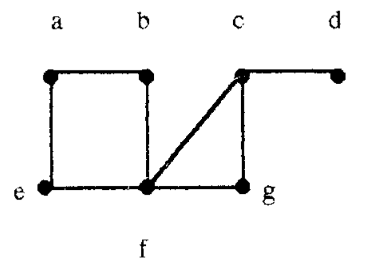
7.5 ต้นไม้แบบทอดข้าม (Spanning trees)

บทนิยาม 1 ให้ G เป็นกราฟเชิงเดียว ต้นไม้แบบทอดข้ามของ G หมายถึง กราฟย่อย ของ G ซึ่งเป็นต้นไม้ ประกอบด้วยทุกจุด ของ G

(Let G be a simple graph. A **spanning tree** of G is a subgraph of G that is a tree containing every vertex of G.)

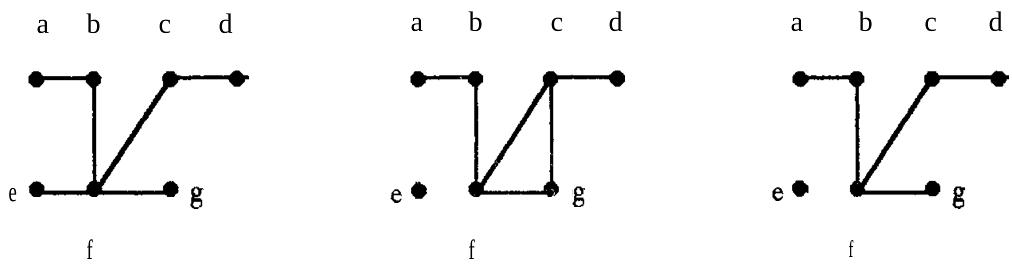
กราฟเชิงเดียว ที่มีต้นไม้แบบทอดข้าม ต้องเป็นกราฟไม่ขาดตอน เพราะว่า มี ทางเดิน ใน ต้นไม้แบบทอดข้าม ระหว่างสองจุดใดๆ บทกลับ เป็นจริงเช่นกัน นั่นคือ กราฟเชิงเดียวไม่ขาดตอนทุกชุด มี ต้นไม้แบบทอดข้าม (that is, every connected simple graph has a spanning tree.)

ตัวอย่าง 1 จงหาต้นไม้แบบทอดข้าม ของ กราฟเชิงเดียว G ในรูปที่ 1



รูปที่ 1 กราฟเชิงเดียว G

ผลเฉลย กราฟ G เป็นกราฟไม่ขาดตอน แต่ ไม่ใช่ต้นไม้ เพราะว่า มันมี วงจรเชิงเดียว ตัดด้าน {a, e} สิ่งนี้ ขจัดวงจรเชิงเดียว หนึ่งชุด และกราฟย่อยผลลัพธ์ ยังคงเป็นกราฟไม่ขาดตอน และยังคงประกอบด้วย ทุกจุด ของ G ต่อไป ตัดด้าน {e, f} เพื่อขจัด วงจรชิงเดียวชุดที่สอง สุดท้าย ตัดด้าน {c, g} เพื่อให้กราฟเชิงเดียว ไม่มีวงจรเชิงเดียวใดๆ กราฟย่อยนี้ คือ ต้นไม้แบบทอดข้าม เพราะว่า มันเป็นต้นไม้ ซึ่งประกอบด้วยทุกจุด ของ G ลำดับของด้าน ซึ่งตัดออก ทำให้เกิด ต้นไม้แบบทอดข้าม ซึ่งแสดงในรูปที่ 2

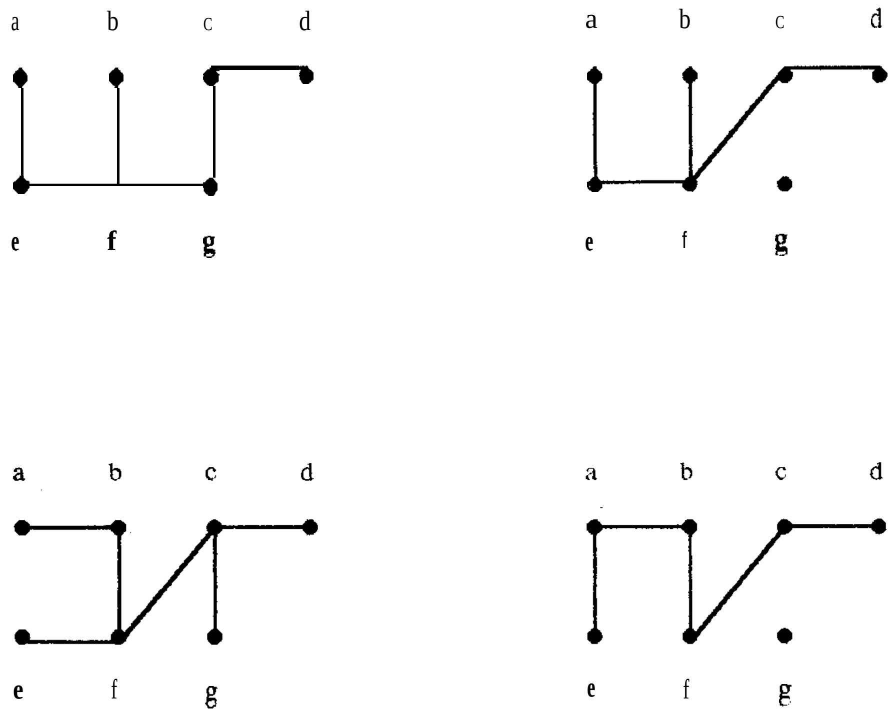


ตัดด้าน {a, e} ตัดด้าน {e, f} ตัดด้าน (c, g}

(a) (b) (c)

รูปที่ 2 การสร้างต้นไม้แบบทอดข้ามของ G โดยการตัดด้านต่างๆ
ซึ่งประกอบเป็นวงจรเชิงเดียว

ต้นไม้ที่แสดงในรูปที่ 2 ไม่ใช่ ต้นไม้แบบทอดข้าม เพียงต้นเดียวเท่านั้น ของ G ตัวอย่างเช่น ต้นไม้แต่ละต้น ในรูปที่ 3 คือ ต้นไม้แบบทอดข้าม ของ G ด้วย



รูปที่ 3 ต้นไม้แบบทอดข้าม ของ G

ทฤษฎีบท 1 กราฟเชิงเดียว จะเป็นกราฟไม่ขาดตอน ก็ต่อเมื่อ มันมีต้นไม้แบบทอดข้าม

(A simple graph is connected if and only if it has a spanning tree.)

อัลกอริทึมสำหรับการสร้างต้นไม้แบบทอดข้าม

(Algorithms for constructing spanning trees)

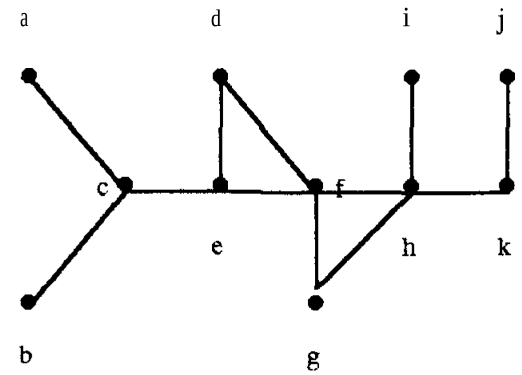
อัลกอริทึม สร้างต้นไม้แบบทอดข้าม โดยการตัดด้านต่างๆ จากวงจรเชิงเดียว ออกไป อัลกอริทึมนี้ จะไม่มีประสิทธิภาพ เพราะว่า มัน ต้องทราบวงจรเชิงเดียว ดังนั้น แทนที่จะสร้างต้นไม้แบบทอดข้าม โดย การตัดด้านต่างๆ ต้นไม้แบบทอดข้ามสามารถสร้างได้ โดย ใส่ด้านต่างๆ อย่างสืบเนื่อง อัลกอริทึม สองชุด ซึ่ง ขึ้นกับหลักเกณฑ์นี้ จะได้นำเสนอ ดังนี้

เราสามารถ สร้างต้นไม้แบบทอดข้าม สำหรับกราฟเชิงเดียว ไม่ขาดตอน โดยใช้ **การค้นในแนวลึก** (depth-first-search) เราจะก่อรูป (form) ต้นไม้ราก และ ต้นไม้แบบทอดข้ามจะเป็นกราฟแบบไม่มีทิศทาง ของต้นไม้ราก นี้ เลือกจุดใดจุดหนึ่งของกราฟให้เป็นราก ทางเดินเริ่มต้นจากจุดนี้ โดยการใส่ด้านต่างๆ อย่างสืบเนื่อง เมื่อด้านใหม่แต่ละด้านตกกระทบกับจุดสุดท้าย ในทางเดิน และ จุดนั้น ต้อง ไม่อยู่ในทางเดินแล้ว ใส่ด้านต่างๆ ต่อไป กับทางเดินนี้ ยาวเท่าที่เป็นไปได้ ถ้าทางเดินยาวไป ตลอดจุดทั้งหมดของกราฟ ต้นไม้ ซึ่งประกอบด้วยทางเดินนี้ คือต้นไม้แบบทอดข้าม แต่ ถ้าทางเดิน ไปไม่ตลอดทุกจุด ต้องใส่ด้าน มากขึ้นอีก ให้ย้อนกลับ ไปยัง จุดสุดท้ายถัดไป ในทางเดิน และถ้าเป็นไปได้ ก่อรูปทางเดินใหม่ เริ่มต้นจากจุดนี้ ผ่าน จุดต่างๆ ซึ่งยังไม่มีการเยี่ยม ถ้าไม่สามารถทำได้ ย้ายกลับไปยังอีกหนึ่งจุด ในทางเดิน นั่นคือ ย้อนกลับไปสองจุดในทางเดิน และพยายามอีกครั้งหนึ่ง ทำซ้ำกระบวนงานนี้ เริ่มต้นที่ จุดสุดท้าย ซึ่งเยี่ยมมาแล้ว ย้ายกลับขึ้นไปทางเดิน ครั้งละ หนึ่งจุด ก่อรูปทางเดินใหม่ ซึ่งยาวเท่าที่เป็นไปได้ จนกว่าจะไม่มีด้านให้ใส่อีก เนื่องจาก กราฟ มี จำนวนด้าน จำกัด และ ไม่ใช่กราฟขาดตอน กระบวน-การนี้ จบด้วยการให้ ต้นไม้แบบทอดข้าม แต่ละจุด ซึ่ง จบทางเดิน ที่ ขั้นตอนของอัลกอริทึมจะเป็นจุดใบ ใน ต้นไม้ราก และแต่ละจุด ซึ่ง ทางเดิน ถูกสร้างขึ้น จะเป็นจุดภายใน ผู้อ่านควรสังเกต ธรรมชาติการเรียกซ้ำของกระบวนงานนี้ และโปรดสังเกตว่า ถ้าจุดต่างๆ ในกราฟ เรียงลำดับ การเลือกด้าน ที่แต่ละขั้นของกระบวนงาน จะหาได้ทั้งหมด อย่างไรก็ตาม เราจะไม่เรียงลำดับ จุดต่างๆ อย่างชัดแจ้ง ของกราฟ เสมอไป

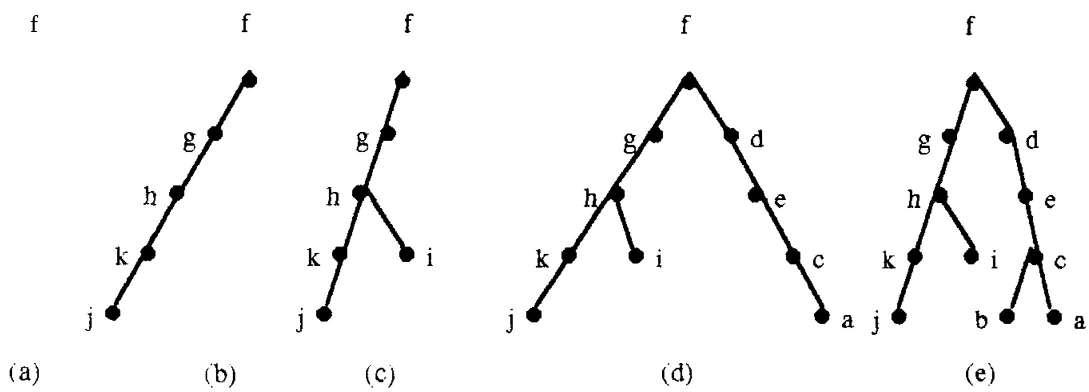
การค้นในแนวลึก เรียกอีกชื่อหนึ่งว่า **การย้อนรอย** (backtracking) เพราะว่าอัลกอริทึมย้อนกลับไปยังจุดก่อนหน้าที่เยี่ยม เพื่อใส่ทางเดิน

ตัวอย่าง 2 จงใช้การค้นในแนวลึก เพื่อหา ต้นไม้แบบทอดข้าม ของกราฟ G ในรูปที่ 4

ผลเฉลย ขั้นตอนต่างๆ ซึ่งใช้ การค้นในแนวลึก เพื่อสร้างต้นไม้แบบทอดข้าม ของ G แสดงให้เห็นในรูปที่ 5 เราเริ่มต้นด้วยจุด f ทางเดินถูกสร้างอย่างสืบเนื่อง ใส่ด้านต่างๆ ซึ่งตกกระทบกับ จุด ซึ่งยังไม่อยู่ในทางเดิน ยาวเท่าที่เป็นไปได้



รูปที่ 4 กราฟ G



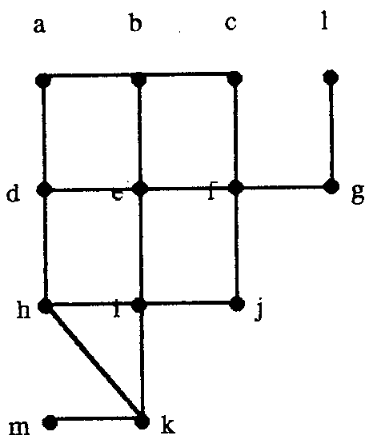
รูปที่ 5 การค้น ในแนวลึก ของ G

สิ่งนี้ เกิดทางเดิน f, g, h, k, j (โปรดสังเกตว่า อาจเป็นทางเดินอื่นได้ด้วย) ต่อไป ย้อนรอยมายัง k ไม่มีทางเดินใดๆ เริ่มต้นที่ k และยังไม่ได้เยี่ยม ดังนั้น เราย้อนรอยไปที่จุด h ก่อรูปทางเดิน h, i จากนั้น ย้อนรอย ไปที่ h และไปที่ f จากจุด f สร้างทางเดิน f, d, e, c, a จากนั้น ย้อนรอยไป c และก่อรูปทางเดิน c, b สิ่งนี้ ให้ต้นไม้แบบทอดข้าม

อีกวิธีหนึ่ง ในการสร้างต้นไม้แบบทอดข้าม ของกราฟเชิงเดียว คือโดยการใช้ **การค้นในแนวกว้าง** (breadth-first search) อีกครั้งหนึ่ง ต้นไม้ราก จะถูกสร้างขึ้นมา และ กราฟแบบไม่มีทิศทาง ของ ต้นไม้รากนี้ ก่อรูป ต้นไม้แบบทอดข้าม เลือกราก หนึ่งจุด จากจุดต่างๆ ของกราฟ จากนั้น ใส่ ด้านทั้งหมดซึ่งตกกระทบกับจุดนี้ จุดใหม่ต่างๆ ซึ่ง ใส่ ณ ขั้นตอนนี้ จะเป็นจุดที่ระดับที่ 1 ในต้นไม้แบบทอดข้าม เรียงอันดับอย่างไรก็ได้ ต่อไป สำหรับแต่ละจุดที่ระดับที่ 1 เยี่ยมตามลำดับ ใส่แต่ละด้าน ซึ่งตกกระทบกับจุดนี้ ให้เป็นต้นไม้ยาวเท่าที่ จะไม่เกิดวงจรเชิงเดียว ลูกๆ ของแต่จุดที่ระดับที่ 1 เรียงลำดับอย่างไรก็ได้ สิ่งนี้ทำให้เกิดจุดที่ระดับที่ 2 ในต้นไม้ ทำตามกระบวนงานนี้ จนกระทั่ง ใส่จุดทั้งหมดในกราฟ กระบวนงานนี้ จบลง เพราะว่า จำนวนด้านในกราฟ มี จำกัด ต้นไม้แบบทอดข้าม ถูกสร้างขึ้น เนื่องจาก เราสร้างต้นไม้ ซึ่งประกอบด้วยทุกจุด ของกราฟ

ตัวอย่าง 3 จงใช้การค้นในแนวกว้าง หา ต้นไม้แบบทอดข้าม ของกราฟ ในรูปที่ 6

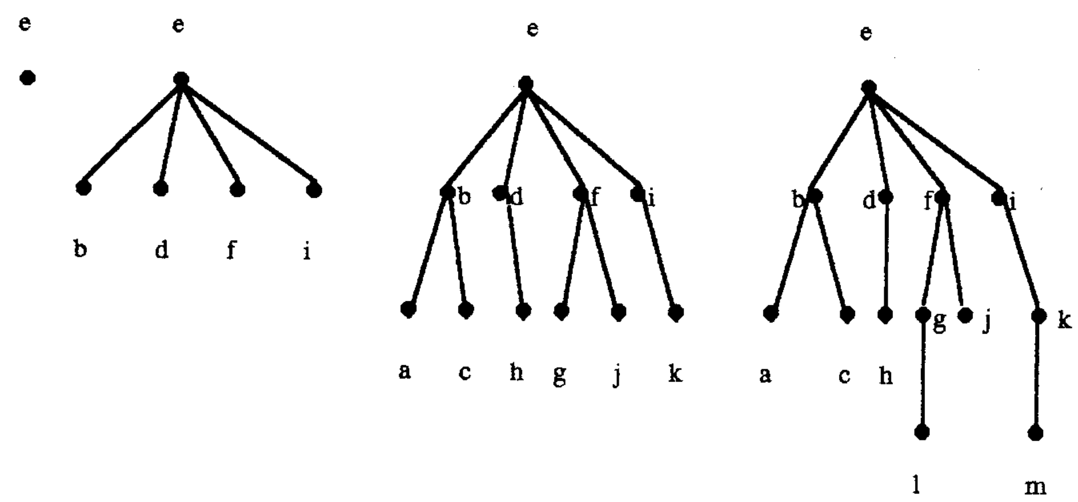
ผลเฉลย ขั้นตอนของกระบวนงานการค้นในแนวกว้าง แสดงให้เห็นในรูปที่ 7 เราเลือกจุด e ให้เป็นราก จากนั้น ใส่ด้าน ซึ่งตกกระทบกับ ทุกจุด ซึ่ง ประชิดกับ e ดังนั้น ด้านจาก e ไป d, b, f, i จะใส่เข้ามา ทั้งสี่จุดนี้ อยู่ที่ระดับ 1 ในต้นไม้ ถัดไป ใส่ ด้านจากจุดเหล่านี้ ที่ ระดับ 1 ไปยังจุดประชิด ซึ่งยังไม่มีอยู่ในต้นไม้ เพราะฉะนั้น ใส่ด้านจาก b ไป a และ b ไป c ด้านจาก d ไป h, จาก f ไป j และ g และจาก i ไป k จุดใหม่ทั้งหมดคือ a, c, h, j, g และ k จะอยู่ที่ระดับที่ 2 ต่อไป ใส่ด้านจากจุดเหล่านี้ ไปยัง จุดประชิดซึ่งยัง ไม่มีอยู่ในกราฟ ด้านนี้คือ จาก g ไป l และจาก k ไป m



รูปที่ 6 กราฟ G

การย้อนรอย (Backtracking)

มีปัญหาต่างๆ ซึ่ง สามารถแก้ปัญหาได้ เฉพาะโดยการ กระทำการค้น ของ ผลเฉลยที่เป็นไปได้ทั้งหมด วิธีหนึ่งของการค้นอย่างเป็นระบบ สำหรับผลเฉลยคือ ใช้ ต้นไม้การตัดสินใจ ซึ่ง จุดภายในแต่ละจุด แทน การตัดสินใจ และจุดใบแต่ละจุด แทน ผลเฉลยที่เป็นไปได้หนึ่งอย่าง



รูปที่ 7 การค้นในแนวกว้าง ของ G

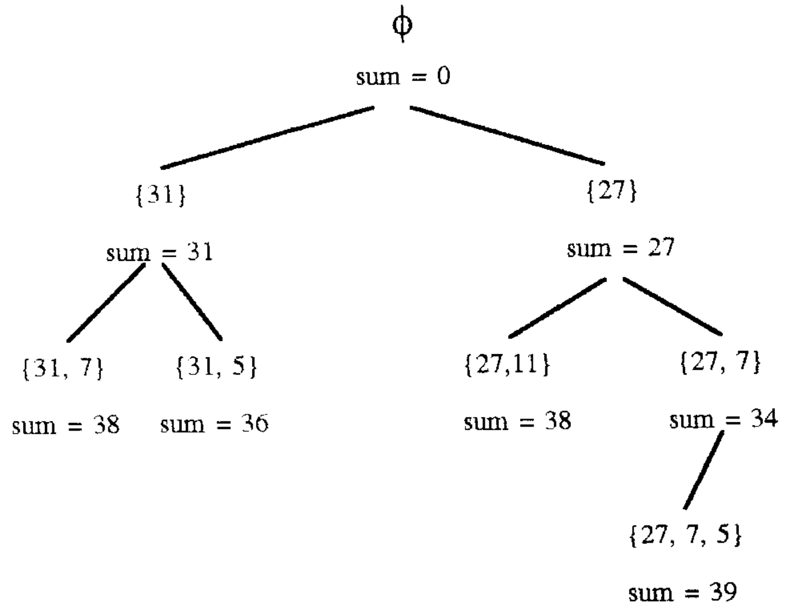
ในการหาผลเฉลย ผ่าน การย้อนรอย ขั้นแรก ทำ ลำดับของการตัดสินใจ ในความพยายาม ที่ ให้ถึง ผลเฉลย ยาวเท่าที่เป็นไปได้ ลำดับของการตัดสินใจ สามารถแทน ทางเดินหนึ่งในต้นไม้การตัดสินใจ เมื่อรู้แล้วว่า ไม่มีผลเฉลยใด เป็นผลลัพธ์ จาก ลำดับต่อไปของการตัดสินใจ ให้ย้อนรอย ไปยัง จุดแม่ ของ จุดปัจจุบัน และทำงานต่อไป หาผลเฉลย ด้วย การตัดสินใจอีกชุดหนึ่ง ถ้าสิ่งนี้เป็นไปได้ กระบวนงานทำต่อเนื่อง จนกระทั่งพบผลเฉลย สิ่งนี้แสดงให้เห็นประโยชน์ของการย้อนรอย

ตัวอย่าง 4 ผลรวมของเซตย่อย (Sums of subsets)

จงพิจารณาปัญหาต่อไปนี้ กำหนดเซต ของจำนวนเต็มบวก $x_1, x_2, \ldots, x_n$ จงหาเซตย่อย ของ เซตของจำนวนเต็มชุดนี้ ซึ่งมี M เป็นผลบวกของสมาชิกเซตย่อยมัน จะใช้การย้อนรอยเพื่อ แก้ปัญหาชุดนี้ได้อย่างไร?

ผลเฉลย เราเริ่มต้น ด้วย ผลบวก (sum) ไม่มีเทอมใดๆ เลย จากนั้น สร้าง sum โดย ใส่

เทอมต่างๆ อย่างสืบเนื่อง เลขจำนวนเต็ม ในลำดับ จะใส่ไว้ ถ้า ผลบวก ยังคงมีค่าน้อยกว่า M เมื่อจำนวนเต็มนี้ บวกกับ sum ถ้า sum ที่ได้ โดยการบวกของเทอมใดๆ มีค่ามากกว่า M ให้ย้อนรอย โดย ตัด เทอมสุดท้าย ออกจาก sum



รูปที่ 8 การหา sum ที่มีค่าเท่ากับ 39 (M = 39) โดยใช้ การย้อนรอย

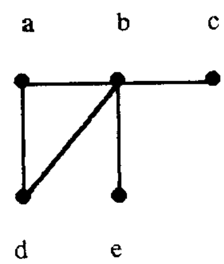
ในรูปที่ 8 แสดงให้เห็น ผลเฉลยของการย้อนรอย ของ ปัญหา การหา เซตย่อย ของ {31, 27, 15, 11, 7, 5} ที่มีผลบวกของสมาชิกเท่ากับ 39

แบบฝึกหัด 7.5

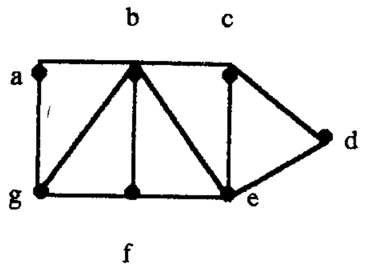
1. กราฟไม่ขาดตอน ที่มี n จุด และ m ด้าน ในการสร้างต้นไม้แบบทอดข้าม จะต้อง ตัดด้านออก กี่ด้าน?

ในแบบฝึกหัดข้อ 2 - 4 จงหาต้นไม้แบบทอดข้าม ของกราฟ ที่แสดงไว้ โดย การตัดด้านต่างๆ ในวงจรเชิงเดียว

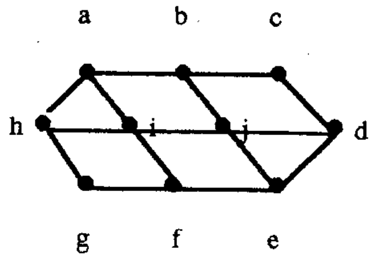
2.



3.



4.

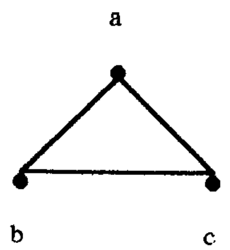


5. จงหาต้นไม้แบบทอดข้าม ของ กราฟแต่ละชุดข้างล่างนี้

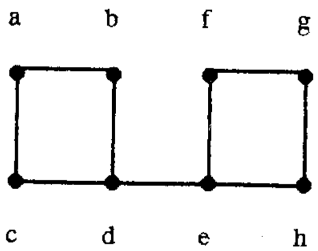
a) K_5 b) $K_{4,4}$ c) $K_{1,6}$ d) Q_3 e) C_5 f) W_5

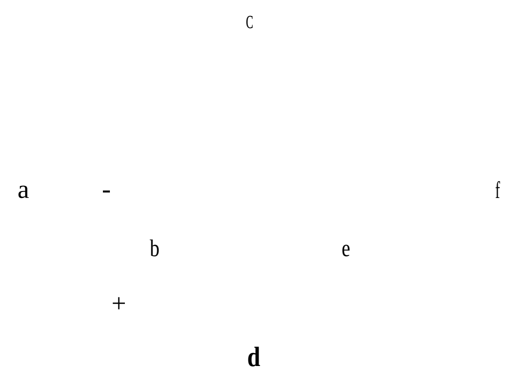
ในแบบฝึกหัดข้อ 6- 8 จงวาดรูปต้นไม้แบบทอดข้ามทั้งหมด ของ กราฟเชิงเดียวที่กำหนดให้

6.



7.



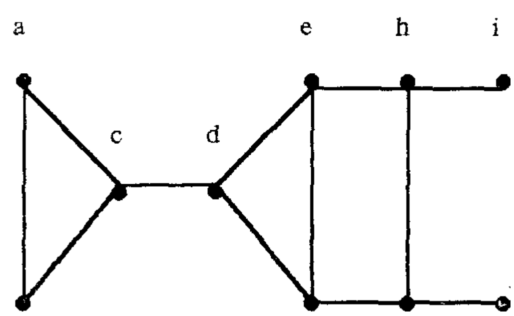


9. กราฟเชิงเดียวต่อไปนี้ แต่ละชุดมี ต้นไม้แบบทอดข้ามที่แตกต่างกัน กี่ชุด

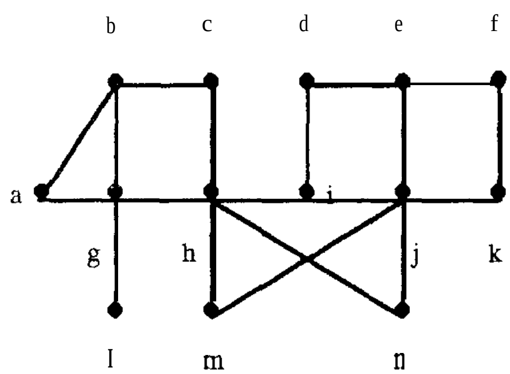
a) K_3 b) K_4 c) $K_{2,2}$ d) C_5

ในแบบฝึกหัดข้อ 10 - 12 จงใช้การค้นในแนวลึก (depth-first search) เพื่อสร้าง ต้นไม้แบบทอดข้าม สำหรับกราฟเชิงเดียวที่กำหนดให้ เลือก a เป็นราก ของ ต้นไม้แบบทอดข้ามนี้ และสมมติว่า จุดต่างๆ เรียงลำดับตามตัวอักษร

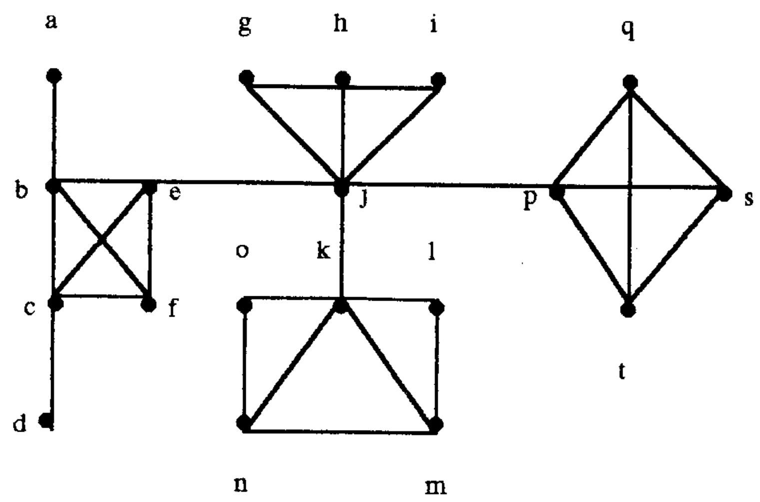
10.



11.



12.



13. จงใช้การค้นในแนวกว้าง (breadth-first search) สร้าง ต้นไม้แบบทอดข้าม สำหรับกราฟเชิงเดียวแต่ละชุด ในแบบฝึกหัดข้อ 10 - 12 เลือก a เป็น ราก ของ ต้นไม้แบบทอดข้ามแต่ละชุด

14. กราฟเชิงเดียวไม่ขาดตอน ชุดใดบ้าง ซึ่งมี ต้นไม้แบบทอดข้าม เพียงหนึ่งต้นเท่านั้น? (Which connected simple graphs have exactly one spanning tree?)

15. จงอธิบายว่า การค้นในแนวกว้าง หรือ การค้นในแนวลึก สามารถนำมาใช้ ในการเรียงอันดับจุดต่างๆ ของกราฟไม่ขาดตอน ได้อย่างไร

16. จงเขียน โปรซีเดอร์ การค้นในแนวลึก ด้วยรหัสเทียม

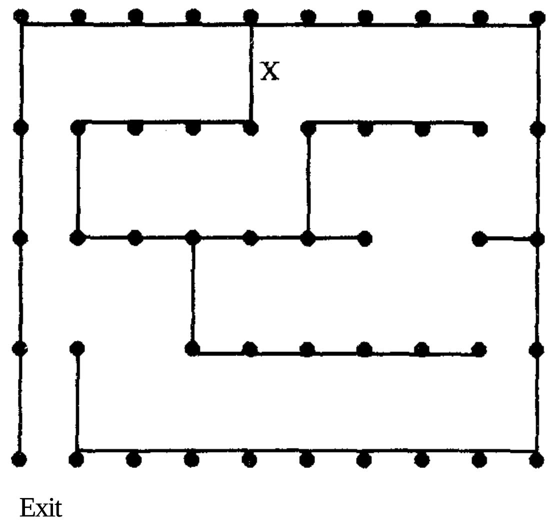
17. จงเขียน โปรซีเดอร์ การค้นในแนวกว้าง ด้วยรหัสเทียม

18. จงใช้การย้อนรอย หา เซตย่อย ถ้ามันมีอยู่จริง ของเซต {27, 24, 19, 14, 11, 8} ที่มี sum เท่ากับ

a) 20 b) 41 c) 60

19. จงอธิบายว่า การย้อนรอย สามารถนำมาใช้ หา ทางเดินแฮมมิลตัน หรือ วงจร ในกราฟ ได้อย่างไร

20. จงอธิบายว่า การย้อนรอย สามารถนำมาใช้ ในการหา ทางออก ของ ตารางปริศนา (maze) ได้อย่างไร เมื่อกำหนดตำแหน่งเริ่มต้น และ ตำแหน่งออก ให้ จงพิจารณา maze ซึ่งแบ่งออกเป็นตำแหน่งต่างๆ เมื่อ แต่ละตำแหน่ง เซตของการย้ายที่เป็นไปได้ มี 4 วิธี (ขึ้น, ลง, ขวา, ซ้าย)



ป่าแบบทอดข้าม ของ กราฟ G หมายถึงป่า ซึ่งประกอบด้วยทุกจุด ของ G โดยที่ สอง จุด ซึ่งอยู่ในต้นไม้ต้นเดียวกัน ของป่า จะมีทางเดิน ใน G ระหว่างสองจุดนี้

(A spanning forest of graph G is a forest that contains every vertex of G such that two vertices are in the same tree of the forest when there is a path in G between these two vertices.)

7.6 ต้นไม้แบบทอดข้ามต่ำสุด (Minimum Spanning Trees)

บริษัทคอมพิวเตอร์แห่งหนึ่ง วางแผน ที่จะสร้าง ข่ายงานการสื่อสาร ต่อ ระหว่างศูนย์คอมพิวเตอร์ ห้าแห่งของบริษัท แต่ละคู่ของศูนย์เหล่านี้ เชื่อม กันด้วยสายโทรศัพท์ ให้เช่า (leased telephone line) การเชื่อมจุดใดบ้าง ซึ่งทำให้เชื่อมั่นว่า มีทางเดินหนึ่งทาง ระหว่างศูนย์คอมพิวเตอร์ สองแห่งใดๆ เพื่อให้ ค่าใช้จ่ายทั้งหมดของข่ายงานนี้ ต่ำสุด? เราสามารถสร้างตัวแบบ ของปัญหานี้ โดยใช้ กราฟถ่วงน้ำหนัก (weighted graph) ซึ่งแสดงในรูปที่ 1 เมื่อ จุดแทน ศูนย์คอมพิวเตอร์ ด้าน แทน สายโทรศัพท์ ให้เช่าที่เป็นไปได้ และน้ำหนักบนด้าน คืออัตราค่าเช่าต่อเดือน ของ สายโทรศัพท์ ซึ่งแทนด้วยด้าน เราสามารถแก้ปัญหานี้ โดย การหาต้นไม้ทอดข้าม ซึ่งผลรวมของน้ำหนัก ด้านต่างๆ ของต้นไม้ มีค่าต่ำที่สุด ต้นไม้ทอดข้ามเช่นนี้ เรียกว่า ***ต้นไม้ทอดข้ามแบบต่ำสุด***

บทนิยาม 1 ต้นไม้แบบทอดข้ามต่ำสุด ใน กราฟถ่วงน้ำหนักไม่ขาดตอน หมายถึง ต้นไม้แบบทอดข้าม ซึ่ง ผลรวมที่เป็นไปได้ของน้ำหนักด้านของมัน มีค่าน้อยที่สุด

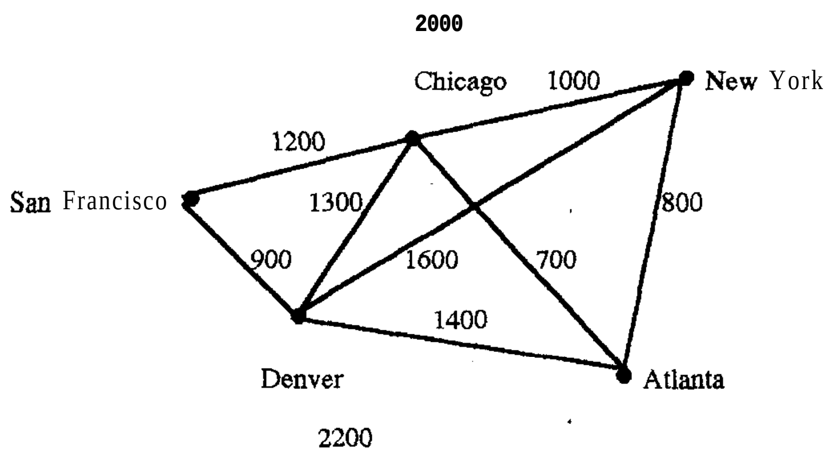
(A **minimal spanning tree** in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.)

เราจะนำเสนออัลกอริทึม สองชุด สำหรับสร้าง ต้นไม้แบบทอดข้ามต่ำสุด ทั้งสองชุดนี้ กระทำการใส่ด้าน ซึ่งมีน้ำหนักน้อยที่สุด อย่างสืบเนื่อง จากด้านต่างๆ ที่มีคุณสมบัติว่า ยังไม่ได้นำมาใช้ อัลกอริทึมเหล่านี้ คือตัวอย่าง**ของอัลกอริทึม จะกละ** (greedy algorithms)

อัลกอริทึมจะกละ หมายถึง กระบวนงาน ซึ่ง ทำให้การเลือก เหมาะที่สุด ในแต่ละครั้ง ของขั้นตอนของมัน

(A greedy algorithm is a procedure that makes an optional choice at each of its steps)

การทำให้เหมาะที่สุด ที่แต่ละขั้นตอน ของอัลกอริทึม ไม่ได้รับประกันว่า จะให้ผลเฉลย โดยรวม เหมาะที่สุด อย่างไรก็ตาม อัลกอริทึม สองชุดที่นำเสนอ ในหัวข้อนี้ สำหรับสร้าง ต้นไม้แบบทอดข้ามต่ำสุด คือ อัลกอริทึมจะกละ ซึ่งจะให้ผลเฉลยเหมาะที่สุด



รูปที่ 1 กราฟถ่วงน้ำหนัก แสดง ค่าใช้จ่ายสำหรับเช่าสายโทรศัพท์ต่อเดือน ในข่ายงานคอมพิวเตอร์

อัลกอริทึม ของ พริม (Prim's algorithm) ผู้พัฒนาคือ Robert Prim ในปี ค.ศ. 1957 เริ่มต้น โดยการเลือก ด้านใดด้านหนึ่ง ที่มี น้ำหนักน้อยที่สุด ใส่ไปใน ต้นไม้แบบทอดข้าม จากนั้น ใส่อย่างต่อเนื่อง ด้านที่มีน้ำหนักน้อยที่สุด ซึ่ง ตกกระทบ กับ จุด ที่มีอยู่แล้วในต้นไม้ และไม่ก่อรูป เป็นวงจรเชิงเดียว กับ ด้านต่างๆ ที่มีอยู่แล้วในต้นไม้ หยุด เมื่อมีการใส่ n - 1 ด้าน

Algorithm 1 Prim's algorithm

procedure Prim (G : weighted connected undirected graph with n vertices.)

T := a minimum - weight edge

for I := 1 to n - 1

e := and edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T

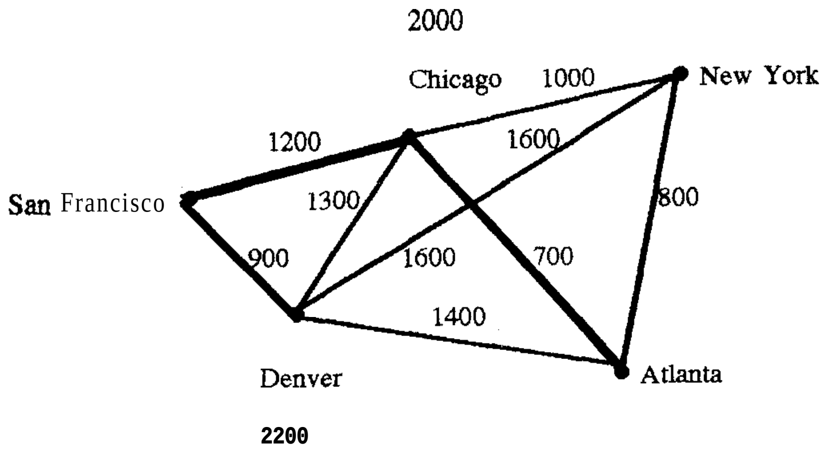
T := T with e added

end {T is a minimal spanning tree of G }

โปรดสังเกตว่า การเลือก หนึ่งด้าน เพื่อใส่ ณ ขั้นตอนหนึ่ง ของ อัลกอริทึม ไม่ได้กำหนดแน่นอน เมื่อ มีด้าน มากกว่าหนึ่งด้าน ที่มี น้ำหนักเท่ากัน และมีคุณสมบัติ ตามหลักเกณฑ์ เราจำเป็นต้องเรียงอันดับ เพื่อทำให้ การเลือก ถูกกำหนดแน่นอน และมีข้อสังเกตว่า สำหรับกราฟเชิงเดียว ถ่วงน้ำหนักไม่ขาดตอนหนึ่งชุด อาจจะมี ต้นไม้แบบทอดข้ามต่ำสุด มากกว่าหนึ่งต้นได้

ตัวอย่าง 1 จงใช้อัลกอริทึม ของ พริม ออกแบบ ข่ายงานการสื่อสาร ซึ่งมีค่าใช้จ่ายต่ำที่สุด ต่อกับ คอมพิวเตอร์ทุกเครื่อง ซึ่งแทนด้วย กราฟในรูปที่ 1

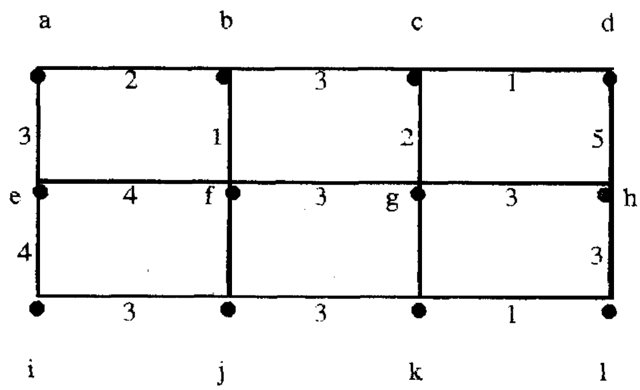
ผลเฉลย เราแก้ปัญหานี้ โดยการหา ต้นไม้แบบทอดข้ามต่ำสุด ใน กราฟรูปที่ 1 อัลกอริทึม ของพริม กระทำให้ประสบผลสำเร็จ โดยเลือก ด้านที่หนึ่ง ซึ่งมีน้ำหนักต่ำที่สุด และทำต่อเนื่องโดยการใส่ ด้าน ที่มี น้ำหนักต่ำสุด และ ตกกระทบ กับ จุดในต้นไม้ และ ไม่ก่อรูปเป็นวงจรเชิงเดียว ด้านเส้นทึบในรูปที่ 2 คือต้นไม้แบบทอดข้ามต่ำสุด ที่เกิดจากอัลกอริทึม ของ พริม ที่มี การเลือก ที่แต่ละขั้นตอนแสดงไว้



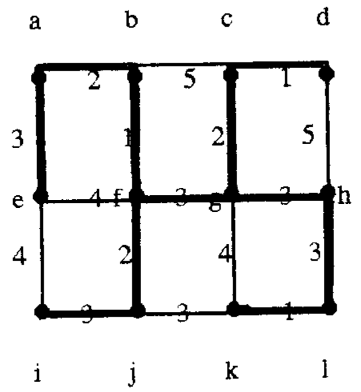
Choice	Edge	cost
1	{ Chicaco, Attlanta)	$700
2	{Atlanta, New York}	$ 800
3	[Chicago, San Francisco}	$1200
4	{San Francisco, Denver]	$ 900
	Total	$3600

รูปที่ 2 ต้นไม้แบทอดข้ามต่ำสุด สำหรับกราฟถ่วงน้ำหนักในรูปที่ 1

ตัวอย่าง 2 จงใช้อัลกอริทึม ของ พริม หาต้นไม้แบบทอดข้ามต่ำสุด ในกราฟรูปที่ 3
ผลเฉลย ต้นไม้แบบทอดข้ามต่ำสุด ซึ่งสร้าง โดยใช้ อัลกอริทึม ของ พริม แสดงในรูปที่ 4 ด้านอย่างต่อเนื่อง ถูกเลือก และแสดงให้เห็น



รูปที่ 3 กราฟถ่วงน้ำหนัก



(a)

Choice	Edge	Weight
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1
	Total	24

(b)

รูปที่ 4 ต้นไม้แบบทอดข้ามต่ำสุด สร้างขึ้นโดยใช้อัลกอริทึมของพริม

อัลกอริทึมชุดที่สอง ซึ่ง จะอภิปรายต่อไป ค้นพบโดย Joseph Kruskal ในปี ค.ศ. 1956 การทำให้อัลกอริทึม Kruskal ประสบผลสำเร็จ เลือกหนึ่งด้านในกราฟที่มีน้ำหนักน้อยที่สุด ใส่ด้านอย่างต่อเนื่อง ที่มีน้ำหนักน้อยที่สุด ซึ่ง ไม่ได้ก่อรูป วงจรเชิงเดียว กับ ด้านที่มีการเลือกไว้แล้ว หยุด หลังจาก ด้าน n - 1 ด้าน ถูกเลือกแล้ว

รหัสเทียม สำหรับ อัลกอริทึม ของ Kruskal กำหนดให้แล้ว ใน อัลกอริทึม 2

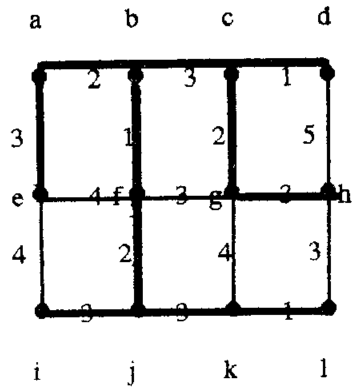
```
Algorithm 2  Kruskal's algorithm
procedure Kruskal (G : weighted connected undirected graph with n vertices)
T := empty graph
for i := 1 to n - 1
begin
    e := any edge in G with smallest weight that does not form a simple circuit
         when added to T
    T := T with e added
end {T is a minimum spanning tree of G}
```

ผู้อ่าน ควรมีข้อสังเกตความแตกต่างระหว่างอัลกอริทึมของพริม และ อัลกอริทึมของ Kruskal

ในอัลกอริทึมของพริม เลือกด้านที่มีน้ำหนักน้อยที่สุด ซึ่งตกกระทบ กับ จุดที่มีอยู่แล้ว ในต้นไม้ และ ไม่ก่อ ให้เกิดวงจรเชิงเดียว ในขณะที่ อัลกอริทึม ของ Kruskal ด้านที่มีน้ำหนัก น้อยที่สุด ไม่จำเป็นต้องตกกระทบ กับ จุดที่มีอยู่แล้ว ในต้นไม้ และ ไม่ได้ก่อรูปวงจร ถูกเลือก ขึ้นมา โปรดสังเกตว่า อัลกอริทึมของพริม ถ้าด้านต่างๆ ไม่ได้เรียงอันดับ อาจจะมี ทางเลือก มากกว่าหนึ่งทาง สำหรับ ด้าน ที่จะใส่ ณ ขั้นตอนหนึ่ง ของกระบวนงานนี้ เพราะฉะนั้น ด้าน ต่างๆ จำเป็นต้องเรียงอันดับ สำหรับกระบวนงาน เพื่อให้กำหนดแน่นอน

ตัวอย่าง 3 จงใช้อัลกอริทึม ของ Kruskal หาต้นไม้แบบทอดข้ามต่ำสุด ใน กราฟถ่วงน้ำหนัก รูปที่ 3

ผลเฉลย ต้นไม้แบบทอดข้ามต่ำสุด และ การเลือกด้านต่างๆ ที่แต่ละขั้นตอน ของ อัลกอริทึม ของ Kruskal แสดงในรูปที่ 5



(a)

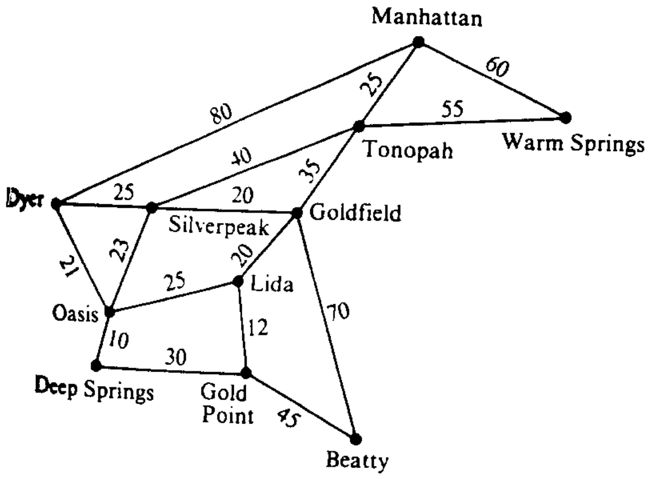
Choice	Edge	Weight
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3
	Total	24

(b)

รูปที่ 5 ต้นไม้แบบทอดข้ามต่ำสุด เกิดขึ้นโดย อัลกอริทึมของ Kruskal

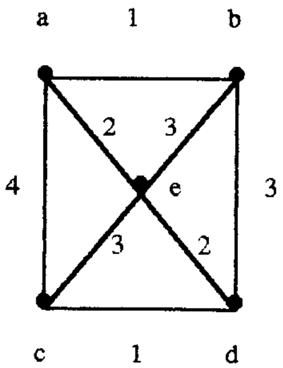
แบบฝึกหัด 7.6

1. ถนน ซึ่ง แทนด้วย กราฟข้างล่างนี้ ทั้งหมด ยังไม่ได้ ลาดยาง (unpaved) ความยาวของถนน ระหว่าง คู่ ของเมือง แสดงด้วยน้ำหนักด้าน ถนนสายใดบ้าง ซึ่ง ควรจะลาดยาง เพื่อให้มี ทางเดินของถนนลาดยาง ระหว่าง ทุกคู่ของเมือง และเป็นความยาวต่ำสุด ของถนนลาดยาง

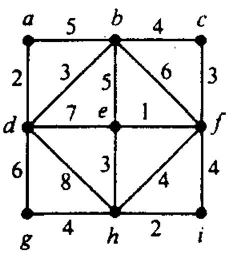


ในแบบฝึกหัดข้อ 2-4 จงใช้อัลกอริทึมของพริม หา ต้นไม้แบบทอดข้ามต่ำสุด สำหรับกราฟถ่วงน้ำหนัก ซึ่ง กำหนดให้

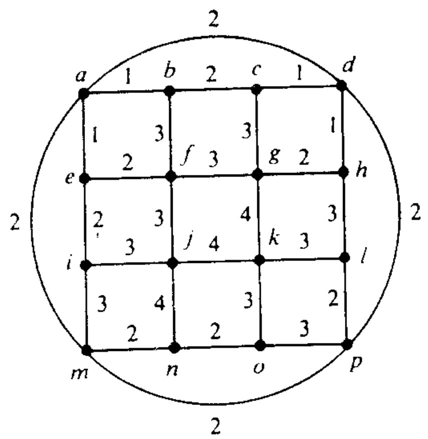
2.



3.



4.



5. จงใช้อัลกอริทึม ของ Kruskal ออกแบบ ข่ายงานการสื่อสาร ตามที่ได้อธิบาย ไว้ตั้งแต่ตอนต้น ของหัวข้อนี้

6. จงใช้อัลกอริทึม ของ Kruskal หาต้นไม้แบบทอดข้ามต่ำสุด สำหรับ กราฟถ่วงน้ำหนัก ในแบบฝึกหัดข้อ 2

7. จงใช้อัลกอริทึม ของ Kruskal หาต้นไม้แบบทอดข้ามต่ำสุด สำหรับ กราฟถ่วงน้ำหนัก ในแบบฝึกหัดข้อ 3

8. จงใช้อัลกอริทึม ของ Kruskal หาต้นไม้แบบทอดข้ามต่ำสุด สำหรับ กราฟถ่วงน้ำหนัก ในแบบฝึกหัดข้อ 4