บทที่8 ต้นไม้ทั่วไปและต้นไม้แบบทวิภาค

8.1 ต้นไม้ทั่วไป
- รูปแบบของการแทนที่

8.2 ต้นไม้แบบทวิภาค

8.3 การแทนที่ของต้นไม้แบบทวิภาค

8.4 การแทนที่ต้นไม้แบบทวิภาคของต้นไม้ทั่วไป

8.5 ต้นไม้ตัวอย่าง

8.6 ต้นไม้ค้นแบบทวิภาค

8.7 การค้นแบบลำดับ
- อัลกอริทึม
- การแวะผ่านแบบตามลำดับ
- การแวะผ่านตามลำดับแบบไม่ใช่เรียกซ้ำ
- การแวะผ่านแบบตามลำดับใน Pascal
- การแวะผ่านแบบหลังลำดับใน COBOL

8.8 ต้นไม้ทวิภาคแบบ threaded

8.9 การค้นโดยตรง

8.10 การใส่โหนด
- การใส่แบบ unthreaded
- การใส่แบบ threaded

8.11 การใส่โหนดในต้นไม้ค้นแบบทวิภาค

8.12 การลบโหนด
- การลบแบบ threaded

8.13 การลบโหนดจากต้นไม้ค้นแบบทวิภาค

8.14 ต้นไม้ค้นทวิภาคแบบได้ดุล

8.15 ต้นไม้ความสูงได้ดุล หรือต้นไม่ AVL

8.16 ต้นไม้ขอบเขตได้ดุล หรือต้นไม้ BB

บทสรุป

แบบฝึกหัด

IT 204 225

บทที่ 8
ต้นไม้ทั่วไปและต้นไม้แบบทวิภาค
(General and Binary Trees)

กราฟชนิดที่สำคัญมีโครงสร้างเป็นต้นไม้ ต้นไม้หมายถึงกราฟไม่มีวง, อย่างง่าย, และไม่ขาดตอน ต้นไม้จะไม่มีรูปบ่วง (loops) และไม่มีวัฏจักร ; ระหว่างแต่ละคู่ของโหนดจะมีเพียงหนึ่งด้านเท่านั้น กราฟในรูป 7-5 คือต้นไม้ ส่วนกราฟรูปก่อนหน้านี้ในบทที่ 7 ไม่ใช่ต้นไม้

8.1 ต้นไม้ทั่วไป (General Trees)

ความสนใจของเราจะจำกัดที่ต้นไม้ชนิดที่เรียกว่าต้นไม้ราก (rooted trees) ต้นไม้ราก หมายถึงต้นไม้ที่มีหนึ่งโหนด(เรียกว่า ราก) ซึ่งแตกต่างจากโหนดอื่น ๆ รากของต้นไม้ T ใช้สัญลักษณ์ root(T)

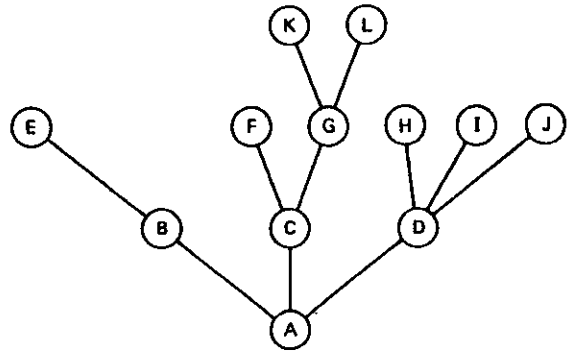
กล่าวเป็นทางการมากขึ้น ต้นไม้ T หมายถึงเซตจำกัดของโหนดมีจำนวนเท่ากับศูนย์หรือมากกว่า (v1,v2,. . .,vn) โดยที่

(1) มีโหนดพิเศษหนึ่งตัว(ชื่อ v1) เรียกว่า Root(T)

(2) โหนดอื่น ๆ , ที่เหลือ (v2,v3,. . .,vn) ถูกแบ่งเป็นส่วน ๆ ให้เป็นเซตไร้สมาชิกร่วม m ชุด ชื่อ T1,T2, . . . , Tm

(disjoint sets) $m \geq 0$ T1,T2, . . . , Tm โดยที่ T_i แต่ละตัวโดยตัวมันเองเป็นต้นไม้ เซต $T_1, \ldots, T_m$ เรียกว่าเซตย่อย (subtree) ของ Root(T) โปรดสังเกตธรรมชาติการเรียกซ้ำของบทนิยามนี้ เราให้นิยามต้นไม้ในเทอมของต้นไม้ สำหรับต้นไม้ที่ไม่มีโหนดใด ๆ เลย เรียกว่า **ต้นไม้ว่าง (null tree)**

รูป 8-1 แสดงให้เห็นต้นไม้ซึ่งเราระบุโหนดแต่ละตัวด้วยตัวอักษรหนึ่งตัวภายในวงกลม สิ่งนี้คือสัญกรณ์ปกติซึ่งเราจะใช้ในการวาดรูปต้นไม้



รูป 8-1 ต้นไม้ตัวอย่าง

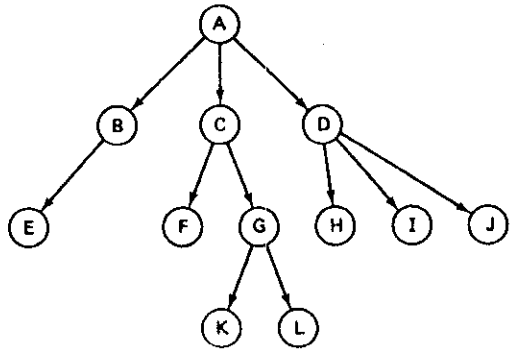
ในที่นี้ Root(T) = A ต้นไม้ส่วนย่อยของ root A จะมีรากอยู่ที่ B, C และ D ตามลำดับ B เป็น root ของต้นไม้ซึ่งมีต้นไม้ส่วนย่อยหนึ่งชุด ซึ่งมีรากอยู่ที่ E ต้นไม้นี้ไม่มีต้นไม้ส่วนย่อย ต้นไม้ซึ่งมี root อยู่ที่ C มีต้นไม้ส่วนย่อยสองชุด มีรากอยู่ที่ F และ G ตามลำดับ

บางครั้งเราจะประยุกต์คุณสมบัติเชิงทิศทางให้กับด้านของต้นไม้ หนึ่งด้านจากโหนดที่เป็นรากไปยังรากของต้นไม้ส่วนย่อย ดังแสดงให้เห็นในรูป 8-2 ในที่นี้เราวาดรูปต้นไม้ในรูปแบบปกติมากกว่าของตำรา การจัดการข้อมูลโดยให้รากของมันอยู่ตอนบนสุดและด้านของมัน(หรือกิ่ง)เติบโตไปด้านล่าง

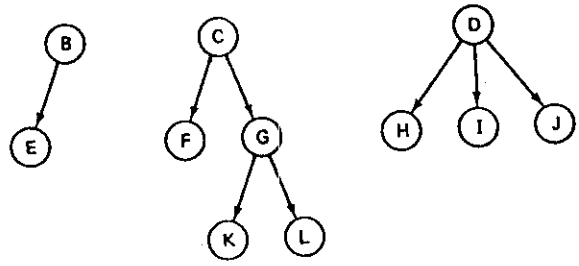
เราพูดว่าต้นไม้มีโหนดพิเศษหนึ่งตัวเรียกว่ารากของมัน รากไม่เพียงแต่เลือกอย่างสุ่มเท่านั้น ยังต้องเป็นโหนดที่มีคุณสมบัติดังนี้

In-degree(v) = 0 for v = root(T)

โหนดรากจะไม่มีกิ่งเข้ามา (incoming branches) เนื่องจากต้นไม้เป็นกราฟไม่ขาดตอน จึงไม่มีโหนดอื่นใดทั้งสิ้นที่มีคุณสมบัตินี้ จงพิจารณาต้นไม้ส่วนย่อยของ A จากต้นไม้ใน รูป 8-2 ซึ่งแสดงในรูปรูป 8-3 โปรดสังเกตว่าด้านต่อจาก A ไป B, C และ D จะไม่ปรากฏในต้นไม้ส่วนย่อยเหล่านี้ เพราะว่า A ไม่ใช่โหนดของมัน



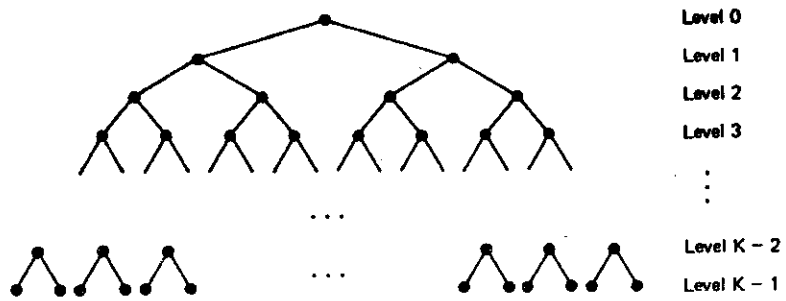
รูป 8-2 ต้นไม้กับด้านมีทิศทาง



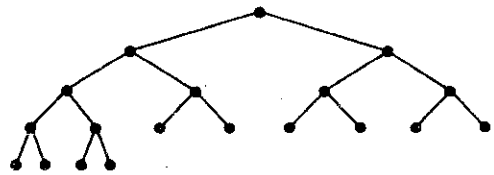
รูป 8-3 ต้นไม้ส่วนย่อยสามต้นจากรูป 8-2

ร่วมสองชุด ซึ่งเรียกว่าต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวา (A binary tree is a finite

คำศัพท์ที่ใช้ในการพูดเกี่ยวกับต้นไม้ทั่วไปคล้ายกับที่ใช้กับต้นไม้สายพันธุ์ (genealogical trees) และต้นไม้ธรรมชาติ ถ้ามีหนึ่งด้าน (A,B) เพราะฉะนั้นพูดว่า A เป็นพ่อ (parent) ของ B และ B เป็นลูก (child) ของ A ดังนั้นลูกจึงมี parent nodeเพียงหนึ่งโหนดเท่านั้น สำหรับโหนดสองตัวที่มี parent node เดียวกันจึงเรียกว่าพี่น้องกัน (brothers or sisters



รูป 8-8 ต้นไม้ทวิภาคแบบบริบูรณ์ที่มี K ระดับ



รูป 8-9 ต้นไม้แบบทวิภาคแบบเกือบจะบริบูรณ์ที่มี 5 ระดับ

ความยาวของทางเดินสูงสุดแบบสั้นที่สุด (The shortest maximum path length) ใช้สัญลักษณ์ PL_{max} สำหรับต้นไม้แบบทวิภาคจะบรรลุผลสำเร็จโดยการทำให้ต้นไม้เป็นแบบบริบูรณ์เท่าที่จะทำได้ ต้นไม้เกือบจะบริบูรณ์มีความสูงต่ำสุดสำหรับเซตโหนดของมันความสูงต่ำสุด (ใช้สัญลักษณ์ H_{min}) สำหรับต้นไม้แบบทวิภาคที่มี N โหนดคือ

$$H_{min} \lceil \log_2(N+1) \rceil$$

เมื่อ $\lceil x \rceil$ หมายถึงฟังก์ชัน ceiling คือจำนวนเต็มเล็กที่สุดซึ่ง $\geq X$

ตัวอย่างเช่น เมื่อ N = 1 เพราะฉะนั้น $H_{min} = 1$

แต่ละตัวถูกจัดสรรเนื้อที่สำหรับจำนวนคงที่ของพอยน์เตอร์ของต้นไม้ส่วนย่อย ไม่ว่าจะ

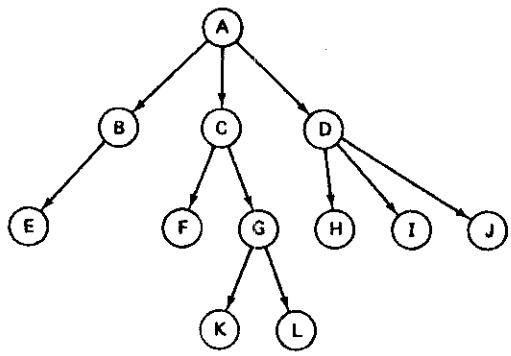
ในที่นี้ Root(T) = A ต้นไม้ส่วนย่อยของ root A จะมีรากอยู่ที่ B, C และ D ตามลำดับ B เป็น root ของต้นไม้ซึ่งมีต้นไม้ส่วนย่อยหนึ่งชุด ซึ่งมีรากอยู่ที่ E ต้นไม้นี้ไม่มีต้นไม้ส่วนย่อย ต้นไม้ซึ่งมี root อยู่ที่ C มีต้นไม้ส่วนย่อยสองชุด มีรากอยู่ที่ F และ G ตามลำดับ

บางครั้งเราจะประยุกต์คุณสมบัติเชิงทิศทางให้กับด้านของต้นไม้ หนึ่งด้านจากโหนดที่เป็นรากไปยังรากของต้นไม้ส่วนย่อย ดังแสดงให้เห็นในรูป 8-2 ในที่นี้เราวาดรูปต้นไม้ในรูปแบบปกติมากกว่าของตำรา การจัดการข้อมูลโดยให้รากของมันอยู่ตอนบนสุดและด้านของมัน(หรือกิ่ง)เติบโตไปด้านล่าง

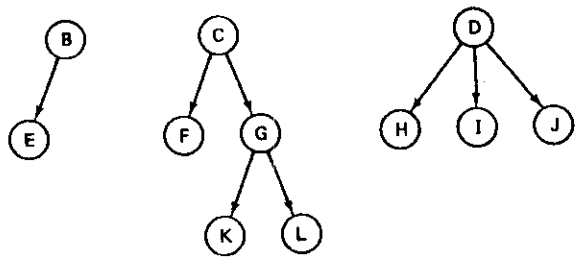
เราพูดว่าต้นไม้มีโหนดพิเศษหนึ่งตัวเรียกว่ารากของมัน รากไม่เพียงแต่เลือกอย่างสุ่มเท่านั้น ยังต้องเป็นโหนดที่มีคุณสมบัติดังนี้

In-degree(v) = 0 for v = root(T)

โหนดรากจะไม่มีกิ่งเข้ามา (incoming branches) เนื่องจากต้นไม้เป็นกราฟไม่ขาดตอน จึงไม่มีโหนดอื่นใดทั้งสิ้นที่มีคุณสมบัตินี้ จงพิจารณาต้นไม้ส่วนย่อยของ A จากต้นไม้ในรูป 8-2 ซึ่งแสดงในรูปรูป 8-3 โปรดสังเกตว่าด้านต่อจาก A ไป B, C และ D จะไม่ปรากฏในต้นไม้ส่วนย่อยเหล่านี้ เพราะว่า A ไม่ใช่โหนดของมัน



รูป 8-2 ต้นไม้กับด้านมีทิศทาง



รูป 8-3 ต้นไม้ส่วนย่อยสามต้นจากรูป 8-2

IT 204 227

รูป 8-3 ต้นไม้ส่วนย่อยสามต้นจากรูป 8-2

คำศัพท์ที่ใช้ในการพูดเกี่ยวกับต้นไม้ทั่วไปคล้ายกับที่ใช้กับต้นไม้สายพันธุ์ (genealogical trees) และต้นไม้ธรรมชาติ ถ้ามีหนึ่งด้าน (A,B) เพราะฉะนั้นพูดว่า A เป็นพ่อ (parent) ของ B และ B เป็นลูก (child) ของ A ดังนั้นลูกจึงมี parent nodeเพียงหนึ่งโหนดเท่านั้น สำหรับโหนดสองตัวที่มี parent node เดียวกันจึงเรียกว่าพี่น้องกัน (brothers or sisters หรือ sibling nodes) เช่นเดียวกับที่ใช้คำศัพท์ในต้นไม้ครอบครัว(family trees) โหนดซึ่งเอาต์ดีกรี(out-degree) เท่ากับศูนย์เรียกว่า **ใบ(leaves)** ของต้นไม้ โหนดซึ่งเป็นใบของต้นไม้รูป 8-2 คือ E, F, K, L, H, I และ J

โหนดของต้นไม้จะแบ่งเป็น **ระดับ(levels)** ซึ่งระดับของโหนดกำหนดโดยความยาวของทางเดิน (path) จากรากไปยังโหนดตัวนั้น ตัวอย่างเช่น ต้นไม้ในรูป 8-2

ระดับ 0	A
ระดับ 1	B, C,D
ระดับ 2	E, F, G, H, I, J
ระดับ 3	K, L

ความสูง ของต้นไม้หมายถึง ตัวเลขของระดับสูงสุดของโหนดบวกหนึ่ง (The height of a tree is one plus the number of the highest level on which there are nodes.)

น้ำหนัก ของต้นไม้หมายถึงจำนวนจุดแตกใบของต้นไม้ (The weight of a tree is its number of leaf nodes.)

จากรูป 8-2 ความสูงและน้ำหนักของต้นไม้คือ 4 และ 7 ตามลำดับ

กลุ่มของต้นไม้รากเรียกว่า **ป่า** (forest) รูป 8-3 แสดงให้เห็นว่าป่าแห่งหนึ่งมีต้นไม้สามต้น

รูปแบบของการแทนที่ (Forms of Representation)

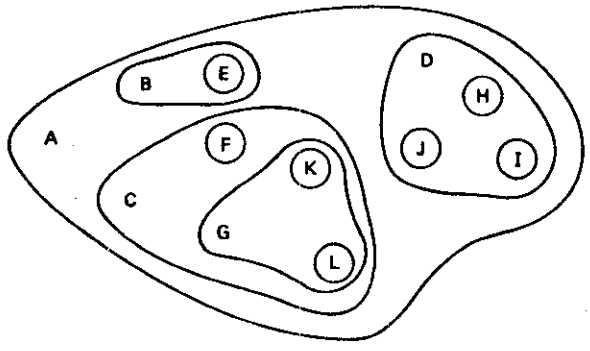
เราใช้สัญกรณ์เชิงภาพเพื่อแทนที่ต้นไม้ สัญกรณ์เชิงภาพอื่น ๆ สำหรับโครงสร้างต้นไม้ได้แก่

(1) เซตซ้อนใน (nested sets) รูป 8-4

(2) วงเล็บซ้อนใน (nested parentheses)

(A(B(E))(C(F)(G(K)(L))) (D(H)(I)(J)))

ซึ่งคู่วงเล็บล้อมรอบรากและต้นไม้ส่วนย่อยของต้นไม้แต่ละต้นและ(3) การย่อหน้า (indentatiopn) เช่นที่แสดงในรูป 8-5



รูป 8-1 การแทนที่แบบเซตซ้อนในของต้นไม้รูป 8-2

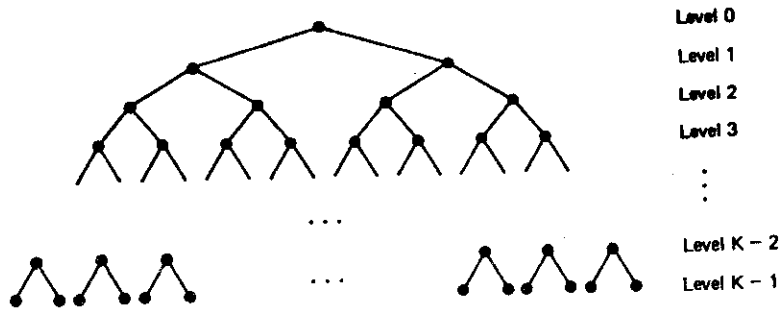
A. . . . . .
B . . . . . .
E. . . . . .
C . . . . . ,
F. . . . . .
G. . . . . .
K. . . .
L. . . .
D. . . . . .
H. . . . . .
I . . . . . .
J. . . . . .

รูป 8-5 การแทนที่แบบย่อหน้าของต้นไม้รูป 8-2

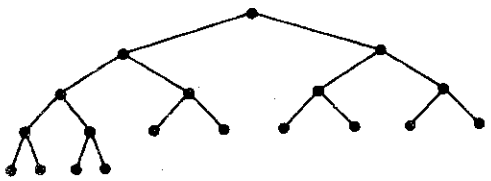
8.2 ต้นไม้แบบทวิภาค (Binary Trees)

ต้นไม้ทั่วไปชนิดที่มีความสำคัญมากที่สุดคือต้นไม้แบบทวิภาค ต้นไม้แบบทวิภาคหมายถึงเซตจำกัดของโหนดซึ่งอาจจะเป็นเซตว่างหรือ ประกอบด้วยต้นไม้ทวิภาคไร้สมาชิกร่วมสองชุด ซึ่งเรียกว่าต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวา (A binary tree is a finite set of nodes which either is empty or contains two disjoint binary trees that are called its left and right subtrees.)

โปรดสังเกตว่า นอกเหนือจากคุณสมบัติที่ว่า จำนวนเอาต์ดีกรีสูงสุดของโหนดทุกตัวของต้นไม้แบบทวิภาคคือ 2 แล้ว มีข้อจำกัดเพิ่มคือการเป็นซ้ายหรือเป็นขวาของคุณสมบัติต้นไม้ส่วนย่อยด้วย ต้นไม้ทวิภาคของรูป 8-6 (a) และ (b) ไม่เหมือนกันทั้งสองรูปเป็นต้นไม้แตกต่างกัน ต้นหนึ่งมีต้นไม้ส่วนย่อยซ้าย และอีกต้นหนึ่งมีต้นไม้ส่วนย่อยขวา ต้นไม้รูป 8-6(c) ไม่ใช่ต้นไม้แบบทวิภาคเพราะว่าต้นไม้ส่วนย่อยของมันไม่มีความเป็นซ้ายหรือเป็นขวามีเพียงส่วนล่าง(downness) เท่านั้น



รูป 8-8 ต้นไม้ทวิภาคแบบบริบูรณ์ที่มี K ระดับ



รูป 8-9 ต้นไม้แบบทวิภาคแบบเกือบจะบริบูรณ์ที่มี 5 ระดับ

ความยาวของทางเดินสูงสุดแบบสั้นที่สุด (The shortest maximum path length) ใช้สัญลักษณ์ PL_{max} สำหรับต้นไม้แบบทวิภาคจะบรรลุผลสำเร็จโดยการทำให้ต้นไม้เป็นแบบบริบูรณ์เท่าที่จะทำได้ ต้นไม้เกือบจะบริบูรณ์มีความสูงต่ำสุดสำหรับเซตโหนดของมันความสูงต่ำสุด (ใช้สัญลักษณ์ H_{min}) สำหรับต้นไม้แบบทวิภาคที่มี N โหนดคือ

$$H_{min} \lceil \log_2(N+1) \rceil$$

เมื่อ $\lceil x \rceil$ หมายถึงฟังก์ชัน ceiling คือจำนวนเต็มเล็กที่สุดซึ่ง $\geq X$

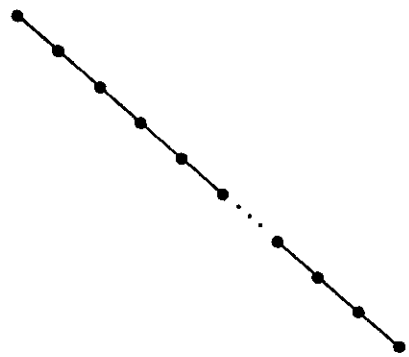
ตัวอย่างเช่น เมื่อ N = 1 เพราะฉะนั้น $H_{min} = 1$

เมื่อ N = 3, ระดับ 1จะถูกใส่ และ$H_{min} = 2$

เมื่อ N = 7, ระดับ 2 จะถูกใส่ และ $H_{min} = 3$

(ข้อควรจำ ถ้า $\log_2 x = y$, ดังนั้น $2^y = x$)

ความสูงกรณีแย่ที่สุดที่จะบรรลุผลเมื่อต้นไม้แบบทวิภาคยาวและ leggy เช่นที่แสดงในรูป 8-10



รูป 8-10 ต้นไม้ long leggy ที่มีความสูงมากที่สุด

ขณะที่ความสูงมากที่สุด(ใช้สัญลักษณ์ H_{max}) สำหรับต้นไม้แบบทวิภาคที่มี N โหนดคือ

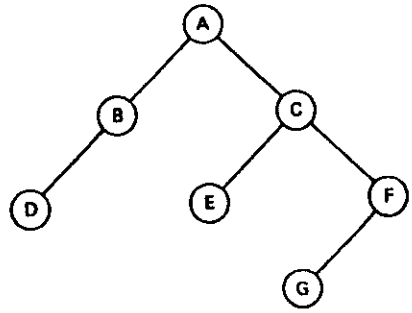
$$H_{max} = N$$

จะเห็นว่าการใช้ที่สำคัญของต้นไม้คือเพื่อเป็นโครงสร้างกลุ่มของข้อมูลในลักษณะซึ่งอำนวยความสะดวกการค้นสมาชิกตัวใดตัวหนึ่ง โดยทั่วไปเป็นความต้องการที่จะจัดการข้อมูลในต้นไม้ในวิธีซึ่งความยาวทางเดินการค้นต่ำที่สุด ปกติต้นไม้เกือบจะบริบูรณ์ถูกนำมาใช้สำหรับเหตุผลนี้

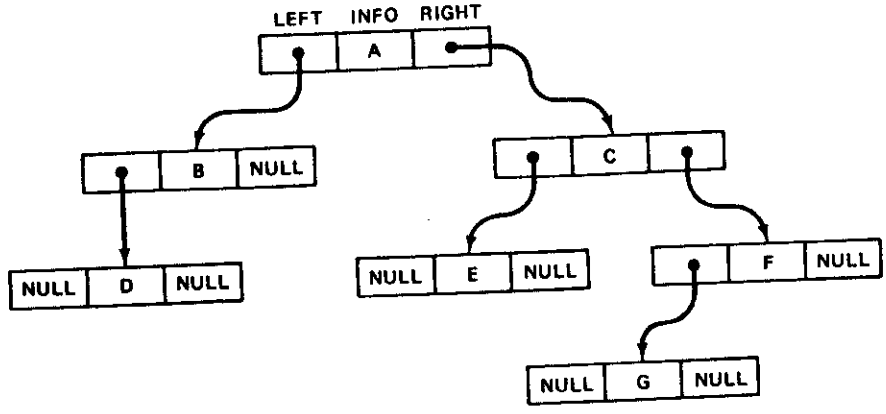
8.3 การแทนที่ของต้นไม้แบบทวิภาค (Representation of Binary Trees)

ต้นไม้แบบทวิภาคส่วนใหญ่ถูกแทนที่ด้วยรายการโยงซึ่งโหนดแต่ละตัวจะแบ่งเป็นสามเขตข้อมูลพื้นฐาน : เนื้อที่สารสนเทศ, พอยน์เตอร์ไปยังต้นไม้ส่วนย่อยซ้ายและพอยน์เตอร์ไปยังต้นไม้ส่วนย่อยขวา ตัวอย่างเช่น ต้นไม้แบบทวิภาคในรูป 8-11 ถูกแทนที่ด้วยรายการโยงรูป 8-12

IT 204 233



รูป 8-11 ต้นไม้ทวิภาคตัวอย่าง



รูป 8-12 การแทนที่แบบรายการโยงของต้นไม้ในรูป 8-11

การใส่กล่องของรายการโยงบนกระดาษไม่มีความหมายแต่อย่างใด หน่วยเก็บสำหรับโหนดอาจกระจัดกระจายตลอดหน่วยความจำ

การนิยามต้นไม้ใน Pascal

ใน Pascal, โครงสร้างข้อมูลชนิดต้นไม้แบบทวิภาคอาจนิยามโดยใช้พอยน์เตอร์เช่นที่ได้ทำแล้วในบทเรื่องรายการโยง สมมติว่าเซตข้อมูล information มีค่าเป็นจำนวนเต็ม

```
type    nodeptr = ↑ nodetype;
        nodetype = record
                        left : nodeptr;
                        info : integer;
                        right : nodeptr;
                   end;
```

ทุกครั้งที่มีการสร้างหนึ่งโหนด มันจะมีโครงสร้างนี้โปรแกรมต้องกำหนดค่าของเขตข้อมูลพอยน์เตอร์ทางซ้ายและพอยน์เตอร์ทางขวาอย่างถูกต้อง เพื่อสร้างการต่อระหว่างกันของโหนดซึ่งประกอบขึ้นเป็นต้นไม้ เนื้อที่สามารถถูกจัดสรรอย่างพลวัตให้กับโหนด ขณะที่ใส่เพิ่มไปในต้นไม้ และเนื้อที่สามารถคืนให้อย่างพลวัตขณะที่ลบโหนดออกจากต้นไม้

การนิยามต้นไม้ใน COBOL

ใน COBOL, ต้นไม้แบบทวิภาคที่มีโหนดมากที่สุด 500 ตัวประกาศดังนี้

```
01    BINARY-TREE.
      02    NODE OCCURS  500 TIMES.
            03    LEFTN       PICTURE    9(3).
            03    INFO        PICTURE    9(5).
            03    RIGHTN      PICTURE    9(3).
```

เนื่องจากภาษา COBOL ไม่มีสิ่งสนับสนุนตัวแปรชนิดพอยน์เตอร์ในตัว ต้นไม้แบบทวิภาคจึงเก็บในแถวลำดับ เราใช้ LEFTN และ RIGHTN เป็นชื่อของพอยน์เตอร์ทางซ้ายและ พอยน์เตอร์ทางขวาเพราะว่า LEFT และ RIGHT เป็นคำสงวนในภาษา COBOL การจัดการเนื้อที่ เช่น แถวลำดับต้องใช้รายการ AVAIL ขวาโหนดว่าง เช่นที่เราได้อภิปรายมาแล้วในบทเรื่องรายการโยง เนื้อที่ต้องถูกจัดสรรเพื่อเก็บขนาดสูงสุดของต้นไม้ เพราะว่าเนื้อที่แถวลำดับของ COBOL ไม่สามารถจัดสรรได้อย่างพลวัต

8.4 การแทนที่ต้นไม้แบบทวิภาคของต้นไม้ทั่วไป(Binary tree Representation of General Trees)

การพิจารณาเพื่อแทนที่ต้นไม้แบบทวิภาคในโปรแกรมง่ายกว่าแทนที่ด้วยต้นไม้ทั่วไป สำหรับต้นไม้ทั่วไปจะไม่สามารถคาดคะเนได้ว่ามีจำนวนด้านเท่าใดที่จะออกมา (emanate)จากหนึ่งโหนดที่เวลาใดเวลาหนึ่งที่กำหนดให้ ดังนั้นเนื้อที่โหนดจึงต้องถูกจัดการในวิธีซึ่งแต่ละโหนดยอมให้ตัวแปรจำนวนของพอยน์เตอร์ของต้นไม้ส่วนย่อยหรือโหนด

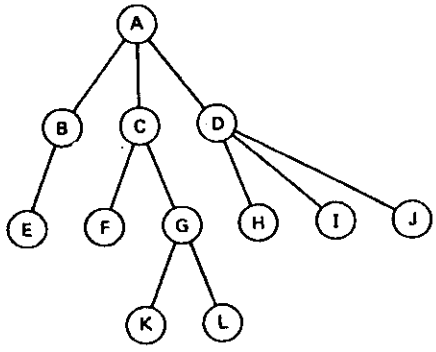
แต่ละตัวถูกจัดสรรเนื้อที่สำหรับจำนวนคงที่ของพอยน์เตอร์ของต้นไม้ส่วนย่อย ไม่ว่าจะต้องการโหนดทั้งหมดหรือไม่ก็ตาม

ต้นไม้แบบทวิภาคน่าสนใจตรงที่ว่าโหนดแต่ละตัวมีจำนวนสูงสุดของพอยน์เตอร์ของต้นไม้ส่วนย่อยที่คาดคะเนได้คือ 2

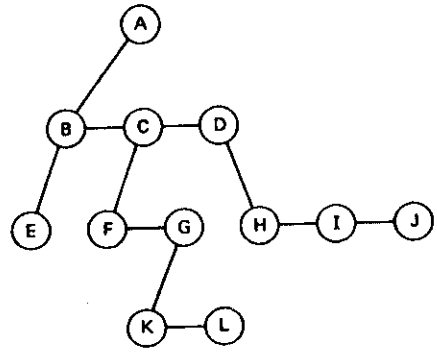
โชคดีมีเทคนิคตรงไปตรงมาสำหรับการแปลงผันต้นไม้ทั่วไปให้เป็นรูปแบบต้นไม้ทวิภาค อัลกอริทึมมีสองขั้นตอนง่าย ๆ ดังนี้

1. ใส่ด้านเพื่อต่อระหว่าง sibling และลบด้านทั้งหมดของ parent ไปยังลูก ๆ ยกเว้นด้านไปยังลูกคนซ้ายมือสุด (Insert edges connecting siblings and delete all of a parent"s edges to its children except to its leftmost offspring.)
2. หมุนแผนภาพผลลัพธ์ 45 องศา เพื่อแยกความแตกต่างระหว่างต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวา (Rotate the resultant diagram 45° to distinguish between left and right subtrees.)

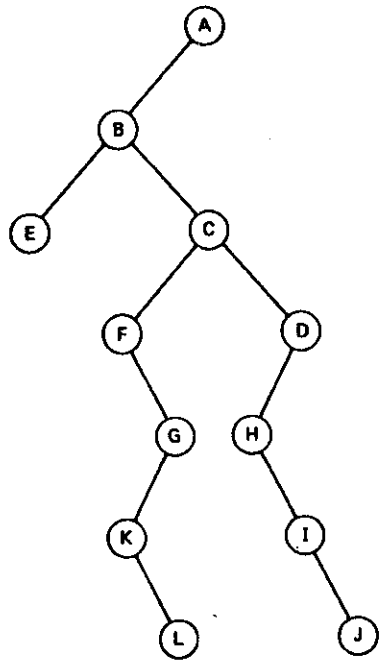
ตัวอย่างเช่น จงพิจารณาต้นไม้ทั่วไปในรูป 8-2 ซึ่งเขียนใหม่ในที่นี้คือรูป 8-13(a) ต้นไม้นี้ไม่ใช่ต้นไม้แบบทวิภาค โหนดบางตัวมีต้นไม้ส่วนย่อยมากกว่าสองชุด ต้นไม้ส่วนย่อยไม่มีความเป็นซ้าย หรือความเป็นขวา ขั้นที่1 ของการแปลงรูป ผลลัพธ์คือโครงสร้างแสดงในรูป 8-13 (b) โปรดสังเกตว่า ไม่มีการใส่ด้านเพื่อต่อ E และ F ทั้งสองโหนดนี้ไม่ใช่ sibling ในต้นไม้เดิม เพราะว่ามันมี parents ต่างกัน (B และ C) ขั้นที่ 2 ผลลัพธ์เป็นรูปต้นไม้แบบทวิภาคแสดงให้เห็นในรูป 8-13 (c)



รูป 8-13(a) ต้นไม้ทั่วไปตัวอย่าง



รูป 8-13(b) การแปลงผันขั้นแรกของต้นไม้ทั่วไปรูป 8-13



รูป 8-13(C) การแทนที่ต้นไม้แบบทวิภาคของต้นไม้ทั่วไปของรูป 8-13(a)

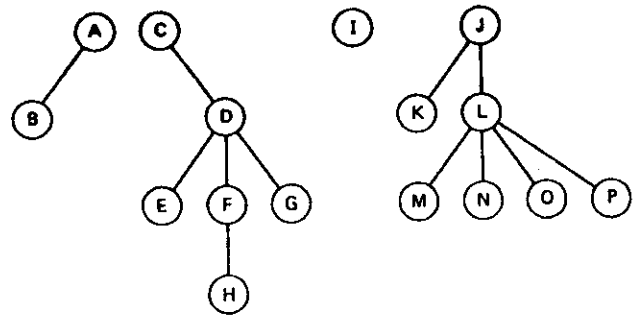
ในต้นไม้แบบทวิภาคผลลัพธ์ พอยน์เตอร์ทางซ้ายจะมาจาก parent node ไปยังลูกคนแรก (ซ้ายมือสุด) ในต้นไม้ทั่วไปเดิมเสมอ ส่วนพอยน์เตอร์ทางขวาจะมาจากโหนดตัวหนึ่งไปยังโหนดของ siblings ของมันในต้นไม้เดิมเสมอ

ต้นไม้ทั่วไปเดิมจะสร้างใหม่จากต้นไม้แบบทวิภาคผลลัพธ์ได้หรือไม่ ? จะเกิดอะไรขึ้นเมื่อการแปลงรูปนี้ประยุกต์กับต้นไม้ทั่วไปซึ่งเป็นต้นไม้แบบทวิภาคแล้ว ?

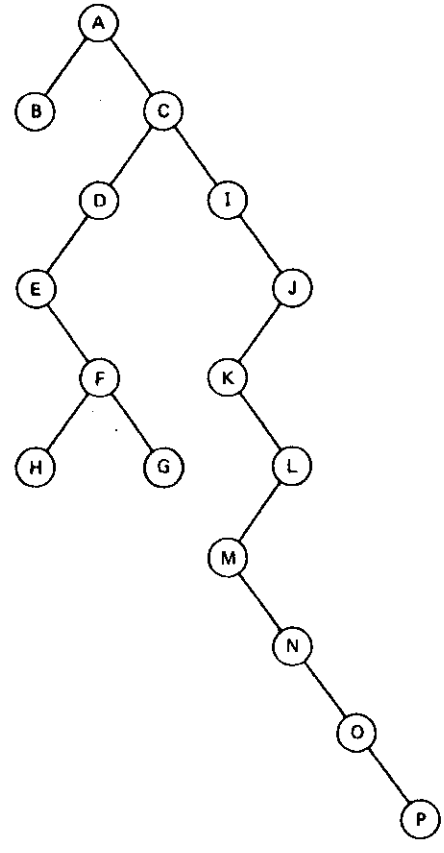
อัลกอริทึมการแปลงรูปนี้สามารถนำไปประยุกต์กับการแปลงผัน(convert)ป่าแห่งหนึ่งของต้นไม้ทั่วไปให้เป็นต้นไม้แบบทวิภาคหนึ่งต้นโดยพิจารณาว่า รากของต้นไม้ทั้งหมดเป็นพี่น้อง (siblings)

ตัวอย่าง เช่น ป่ารูป 8-14(a) ถูกแทนด้วยต้นไม้แบบทวิภาครูป 8-14 (b)

ต้นไม้แบบทวิภาคซึ่งเกิดขึ้นโดยการแปลงผันชนิดนี้จะมีความสูงมากสำหรับจำนวนโหนดซึ่งประกอบเป็นต้นไม้ทางเดินการค้นของมันจึงค่อนข้างยาวแต่การจัดการหน่วยความจำไม่ซับซ้อน



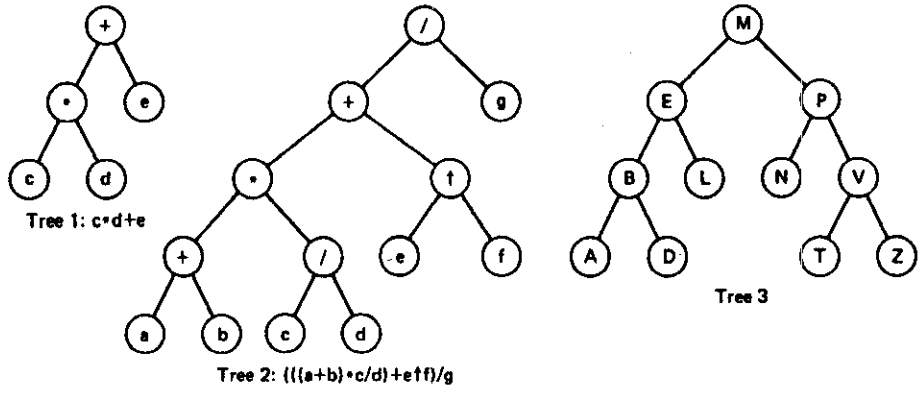
รูป 8-14 (a) ป่าตัวอย่าง



รูป 8-14 (b) การแทนที่ต้นไม้ทวิภาคของป่ารูป 8-14 (a)

8.5 ต้นไม้ตัวอย่าง (Example Trees)

จะเห็นว่าโดยปกติต้นไม้ถูกนำมาใช้เป็นโครงสร้างข้อมูลเพื่อให้สะดวกกับการค้นสำหรับโหนดเฉพาะ ต้นไม้ยังเป็นประโยชน์สำหรับการแทนที่ กลุ่มของข้อมูลซึ่งมีกิ่งเป็นโครงสร้างเชิงตรรกะตัวอย่างเช่น ต้นไม้แบบทวิภาค1 และ 2 ของรูป 8-15 แทนข้อความสั่งคำนวณ (arithmetic statements) โหนดแต่ละตัวที่ไม่ใช่จุดแตกใบจะเป็นตัวดำเนินการ (operator) ต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาของมันคือตัวถูกดำเนินการของมัน โดยปกติคอมไพเลอร์ (compilers) สร้างต้นไม้แบบทวิภาคในกระบวนการของการกราดตรวจ (scanning), การวิเคราะห์กระจาย (parsing) และการก่อกำเนิด (generating) รหัสสำหรับการประเมินผลของนิพจน์คำนวณ ต้นไม้ 3 ในรูปแทนกลุ่มของสมาชิกข้อมูลซึ่งเป็นตำแหน่งในต้นไม้โดยที่ถ้า K คือป้ายชื่อ(label) ของโหนด ป้ายชื่อทั้งหมดของโหนดในต้นไม้ส่วนย่อยซ้ายของมันจะมีค่าน้อยกว่า(หรือเท่ากับ) K และป้ายชื่อทั้งหมดของโหนดในต้นไม้ส่วนย่อยขวาของมัน จะมีค่ามากกว่า K ต้นไม้ชนิดนี้ปกติใช้เป็นโครงสร้างกลุ่มของชื่อ, คีย์, หรือ ป้ายชื่อ การค้นหาโหนดหนึ่งตัวในต้นไม้จะทำได้เร็วกว่าการค้นหากับกลุ่มเดียวกันที่เป็นแบบลำดับ เราจะจงพิจารณาต้นไม้ชนิดนี้ในรายละเอียดที่มากขึ้น



รูป 8-15 ต้นไม้ตัวอย่าง

8.6 ต้นไม้ค้นแบบทวิภาค (Binary Search Trees)

ต้นไม้ค้นแบบทวิภาคของกลุ่มระเบียน N ตัวที่มีคีย์เป็น $K_1, K_2, \ldots, K_N$ หมายถึงต้นไม้แบบทวิภาค ซึ่งโหนด R_i แต่ละตัวมีค่าคีย์เป็น K_i สำหรับ $i = 1, 2, \ldots, n$ คีย์(keys) หมายถึง ไอเดนติไฟเออร์ของโหนด การค้นหาโหนดใด ๆ จะกระทำโดยมองหาค่าคีย์ของมัน โหนดของต้นไม้แบบทวิภาคถูกจัดการเพื่อให้โหนด R_i, แต่ละตัวมีคุณสมบัติดังนี้

1. คีย์ทุกตัวของโหนดในต้นไม้ส่วนย่อยซ้ายของ R_i, จะอยู่ก่อนคีย์ที่มีป้ายชื่อเป็น R_i

   ถ้า $R_j \in \text{left}(R_i)$

   then $K_j < K_i$,

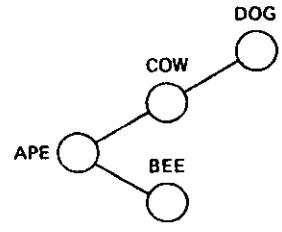
3. คีย์ที่มีป้ายชื่อเป็น R_i ก่อนหน้าคีย์ของโหนดทั้งหมดในต้นไม้ส่วนย่อยขวาของ R_i

   ถ้า $R_j \in \text{right}(R_i)$

   then $K_i < K_j$,

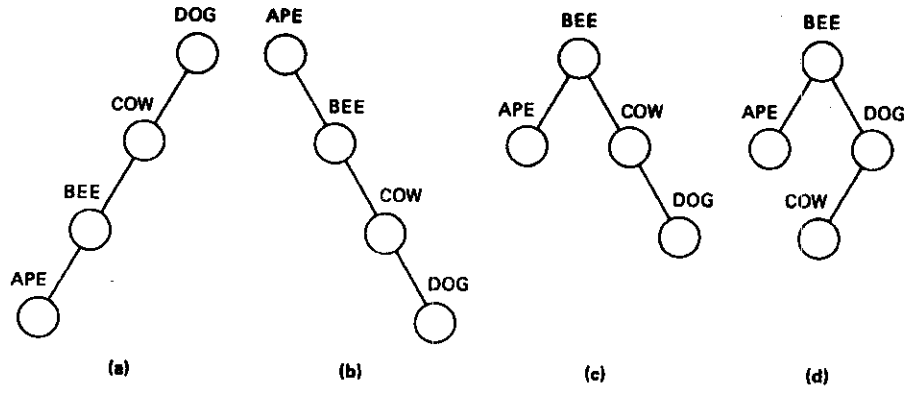
โดยทั่วไป เราต้องการให้คีย์ในกลุ่มแตกต่างกันถึงแม้ว่าขีดจำกัดนี้บางครั้งหย่อนไป การทำก่อน (precedence) ของคีย์กำหนดโดยการเรียงอันดับเชิงเส้นตามลำดับการเรียง (collating sequence)

ตัวอย่างเช่น ต้นไม้แบบทวิภาคในรูป 8-16 เป็นไปตามคุณสมบัติข้างต้นและ qualifies เป็นต้นไม้ค้นแบบทวิภาคผ่านคีย์ APE, BEE, COW และ DOG



รูป 8-16 ต้นไม้ค้นแบบทวิภาคตัวอย่าง

สิ่งสำคัญที่เป็นข้อสังเกตไว้คือมีหลายวิธีที่จะจัดการคีย์ของกลุ่มระเบียนเพื่อให้ผลลัพธ์เป็นต้นไม้ค้นแบบทวิภาคที่ถูกต้อง รูป 8-17 แสดงให้เห็นต้นไม้ค้นแบบทวิภาคเพิ่มเติมที่มีคีย์ APE, BEE, COW และ DOG



รูป 8-17 ต้นไม้ค้นแบบทวิภาค

ขณะที่เราสร้างต้นไม้ค้นแบบทวิภาค เราควรสังเกตว่าเมื่อเริ่มต้นสร้างต้นไม้ และตัดสินใจว่าคีย์ตัวใดควรจะใส่ต่อไปไม่มีทางเลือกใด ๆ เกี่ยวกับตำแหน่งซึ่งจะใส่คีย์นั้น คุณสมบัติซึ่งนิยามต้นไม้ค้นแบบทวิภาคบังคับ (dictate) ความสัมพันธ์ระหว่างโหนดและกำหนดว่า โหนดถัดไปควรจะอยู่ที่ใด โครงสร้างของต้นไม้ค้นแบบทวิภาคใด ๆ ถูกกำหนดโดยอันดับซึ่งโหนดถูกใส่ในต้นไม้ ถ้าเราไม่ยอมให้เกิดเป็นโครงสร้างใหม่ระหว่างกระบวนการของการใส่ โหนดตัวใหม่จะต้องใส่เป็นจุดแตกใบเสมอ

โครงสร้างโหนดสำหรับต้นไม้ค้นแบบทวิภาค ประกาศใน Pascal ดังนี้

```
type    nodeptr = ↑ nodetype;
        nodekey = pack array [1..10] of char;
        nodetype = record
                     left : nodeptr:
                     key : nodekey;
                     info : integer;
                     right : nodeptr
                   end;
```

สำหรับคีย์ที่มีค่าเป็นอักขระ 10 ตัวและเซต info มีค่าเป็นจำนวนเต็ม

8.7 การค้นแบบลำดับ (Sequential Searches)

กระบวนการผ่านตลอดต้นไม้โดยที่โหนดแต่ละตัวถูกเยี่ยมเพียงครั้งเดียวเท่านั้น เรียกว่าการแวะผ่านต้นไม้ (tree traversal) เมื่อมีการแวะผ่านต้นไม้กลุ่มทั้งหมดของโหนดจะถูกค้น การแวะผ่านต้นไม้แบบทวิภาคมีหลายวิธีซึ่งเป็นที่รู้จักกันอย่างแพร่หลายแต่ละวิธีเป็นแบบลำดับ การเรียงอันดับเชิงเส้นขึ้นอยู่กับโหนดของต้นไม้ โหนดจะเรียกว่าถูกเยี่ยม (visited) เมื่อมันถูกพบในการแวะผ่าน การประมวลผลจะเป็นอะไรก็ได้ตามต้องการบน content ของมันซึ่งถูกกระทำ ณ.เวลานั้น

อัลกอริทึม (The Algorithms)

มีกิจกรรมพื้นฐานสามชนิด ในอัลกอริทึมการแวะผ่านต้นไม้แบบทวิภาคแต่ละวิธีได้แก่

เยี่ยมราก (visit the root)

แวะผ่านต้นไม้ส่วนย่อยซ้าย (Traverse the left subtree)

แวะผ่านต้นไม้ส่วนย่อยขวา (traverse the right subtree)

ทั้งสามวิธีแตกต่างกันในอันดับซึ่งชนิดของกิจกรรมเหล่านี้ถูกกระทำ อัลกอริทึมซึ่งนำเสนอในที่นี้ทั้งหมดเป็นแบบเรียกซ้ำ(recursive) แต่ละชุดนิยามในเทอมของตัวมันเอง

การแวะผ่านแบบก่อนลำดับ (Preorder Traversal)

1. Visit the root
2. Traverse the left subtree in preorder
3. Travirse the right subtree in preorder

การแวะผ่านแบบตามลำดับ (Inorder Traversal)

1. Traverse the left subtree in inorder
2. Visit the root
3. Traverse the right subtree in inorder

การแวะผ่านแบบหลังลำดับ (postorder Traversal)

1. Traverse the left subtree in postorder
2. Traverse the right subtree in postorder
3. Visit the root

โหนดของต้นไม้ในรูป 8-15 ถูกเยี่ยมในลำดับข้างล่างนี้เมื่อต้นไม้ถูกแวะผ่านเป็นไปตามวิธีที่เขียนไว้ข้างต้น

ต้นไม้ 1 :	Preorder :	+ * c d e
	Inorder:	c * d + e
	Postorder:	c d * e +
ต้นไม้ 2:	Preorder:	/ + * + a b / c d ↑ e f g
	Inorder :	a + b * c / d + e ↑ f / g
	Postorder:	a b + c d / * e f ↑ + g /
ต้นไม้ 3:	Preorder:	MEBADLPNVTZ
	Inoder:	ABDELMNPTVZ
	Postorder:	ADBLENTZVPM

ผลลัพธ์ของการแวะผ่านเหล่านี้ของต้นไม้ 1 และต้นไม้ 2 อาจเตือนใจเราให้นึกถึงตัวอย่างงานประยุกต์ของกองซ้อนในบทที่ 4 เรื่องสัญกรณ์ของนิพจน์คำนวณ ผลลัพธ์ของการแวะผ่านแบบหลังลำดับในการแทนที่สายอักขระในรูปแบบ fostfix ; ตัวดำเนินการอยู่หลังตัวถูกดำเนินการสองตัวของมัน

ผลลัพธ์การแวะผ่านแบบก่อนลำดับในการแทนที่สายอักขระอยู่ในรูปแบบ prefix ; ตัวดำเนินการอยู่หน้าตัวถูกดำเนินการของมัน

ผลลัพธ์การแวะผ่านแบบตามลำดับในการแทนที่สายอักขระอยู่ในรูปแบบ infix; ตัวดำเนินการอยู่ระหว่างตัวถูกดำเนินการของมัน

โปรดสังเกตว่า การแทนที่ prefix และการแทนที่ infix อาจสูญเสียการทำก่อนของตัวดำเนินการที่ถูกต้อง ถ้านิพจน์คำนวณเดิมต้องใช้วงเล็บกำกับเพื่อให้อยู่เหนือ (override) กฎการทำก่อนแบบธรรมชาติ ในสัญกรณ์เติมหลัง การทำก่อนของตัวดำเนินการจะถูกรักษาไว้และแสดงถึงโดยอันดับตัวดำเนินการอย่างเดียว สำหรับตัวอย่างชนิดนี้ การแวะผ่านแบบหลังลำดับจึงเป็นประโยชน์มากที่สุด

ในทางตรงกันข้ามการแวะผ่านแบบตามลำดับของต้นไม้ที่ 3 ให้ผลลัพธ์ที่เป็นประโยชน์มากที่สุด ; โหนดถูกแวะผ่าน ในลำดับเรียงตามตัวอักษร ต้นไม้ที่ 3 เป็นต้นไม้ค้นแบบทวิภาค การแวะผ่านแบบตามลำดับใช้เพื่อค้นต้นไม้ค้นแบบทวิภาคอย่างแบบลำดับ การแวะผ่านแบบก่อนลำดับ มีประโยชน์มากที่สุดสำหรับงานประยุกต์อื่น ๆ ตัวอย่างหนึ่งของสิ่งที่สำคัญมากที่สุดของงานประยุกต์เหล่านี้คือการแวะผ่านของต้นไม้ซึ่งจัดการโดยระบบจัดการฐานข้อมูลเชิงลำดับชั้น เช่น ระบบจัดการสารสนเทศ (IMS = Information

Management System) ของ IBM การแวะผ่านแบบก่อนลำดับจะเหมือนกับ อันดับการเรียงเชิงลำดับขั้นของ IMS

ด้วยเหตุนี้เทคนิคการแวะผ่านทั้งสามวิธีนี้จึงมีความสำคัญ และมีงานประยุกต์ในทางปฏิบัติ เทคนิคซึ่งเหมาะสมมากที่สุด สำหรับสถานะการณ์เฉพาะเรื่องถูกกำหนดโดยวิธีที่สารสนเทศมีโครงสร้างในต้นไม้

การแวะผ่านแบบตามลำดับ (Inorder Traversal)

โปรซีเดอร์ Pascal เพื่อแวะผ่านต้นไม้ค้นแบบทวิภาคแบบตามลำดับ ข้างล่างนี้ โปรดสังเกตว่าโปรซีเดอร์นี้คล้ายกับบทนิยามสำหรับการแวะผ่านคือเป็นการเรียกซ้ำ

```
procedure inorder (var rootptr : nodeprt);
begin
        if rootptr <> nil
        then    begin   inorder (rootptr↑.left);
                        writeln (rootprt↑.info);
                        inorder (rootptr↑.right)
                end;
end;
```

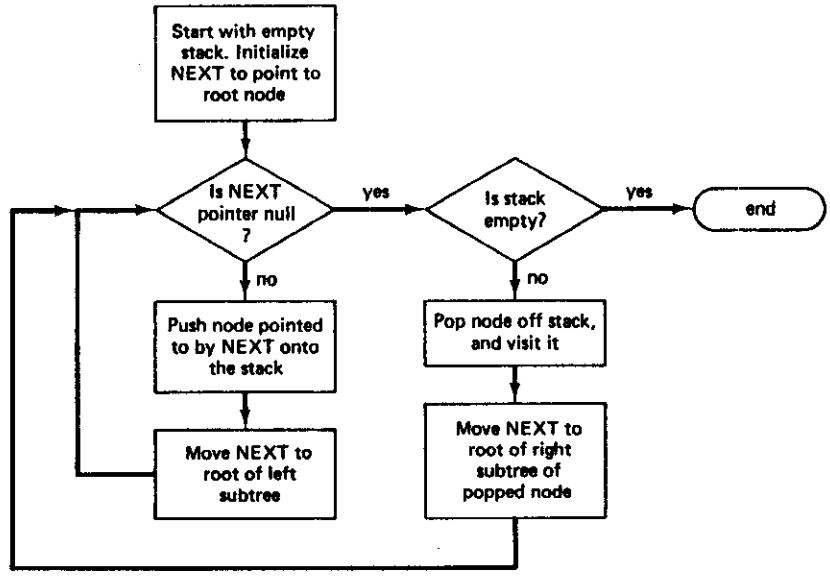
ชนิดของพอยน์เตอร์และโครงสร้างโหนดได้ให้นิยามมาแล้วก่อนหน้านี้ จะเห็นว่าการเขียนโปรซีเดอร์เรียกซ้ำที่สมนัยเพื่อทำให้เกิดผลในทางปฏิบัติของการแวะผ่านแบบก่อนลำดับและหลังลำดับค่อนข้างง่าย

การแวะผ่านแบบก่อนลำดับและหลังลำดับไม่น่าจะนำมาใช้กับต้นไม้ค้นแบบทวิภาคแต่อาจจะนำไปใช้กับต้นไม้ทวิภาคชนิดอื่น ๆ

การแวะผ่านตามลำดับแบบไม่ใช่การเรียกซ้ำ (Nonrecursive Inorder Traversal)

ในการทำให้เกิดผลในทางปฏิบัติบางอย่าง รูทีนแบบไม่ใช่การเรียกซ้ำซึ่งกระทำอย่างชัดแจ้งด้วยการทำกองซ้อนของตนเอง และออกจากกองซ้อนมีประสิทธิภาพมากกว่ารูทีนแบบเรียกซ้ำ อัลกอริทึมไม่ใช่แบบเรียกซ้ำเพื่อแวะผ่านต้นไม้ทวิภาคแบบตามลำดับแสดงให้เห็นในรูป 8-18 โหนดแต่ละตัวของต้นไม้ถูกใส่บนกองซ้อนเพื่อให้ถูกนำออกแบบตามลำดับ เริ่มแรกอัลกอริทึมหยิบโหนดตัวซ้ายมือล่างสุดจากนั้นเป็นรากของมัน หลังจากนั้นเป็นโหนดตัวขวามือ (ซึ่งอาจจะมีต้นไม้ส่วนย่อยของมันเอง) หลังจากนั้นย้ายขึ้นหนึ่ง

ระดับ(โดยลงไปหนึ่งระดับในกองซ้อน)เพื่อเยี่ยมรากนั้นและเก็บโหนดทางขวาของมัน เช่น นี้เรื่อยไปจนกระทั่งโหนดทั้งหมดได้รับการเยี่ยม



รูป 8-18 อัลกอริทึมการแวะผ่านตามลำดับแบบไม่ใช่เรียกซ้ำ

การแวะผ่านแบบตามลำดับใน Pascal (Pascal Inorder Traversal)

โปรซีเดอร์ Pascal เพื่อทำให้เกิดผลของอัลกอริทึมนี้เป็นดังนี้ โปรซีเดอร์ใช้ตัวแปรพอยน์เตอร์ชนิด built-in เพื่อทำให้เกิดผลกับต้นไม้แบบทวิภาค และแถวลำดับเพื่อทำให้เกิดผลกับกองซ้อน ซึ่งจำเป็นต้องเก็บร่องรอย (track) ของตำแหน่งโหนดในต้นไม้ระหว่างการแวะผ่าน สำหรับพอยน์เตอร์ไปยังรากของต้นไม้คือ rootptr

```
procedure inorder (rootptr: nodeptr);
type    stackstruct =  record  stack : array [1..500] of nodeptr;
                               topptr : 0..500
                       end;
var     next :  nodeptr;
        goflag : boolean;
             s :  stackstruct;
        begin
              s.topptr := 0;
              next := rootptr; {start at root }
              goflag := true;
        while (goflag)
        do     begin  { build stack of pointers to nodes }
                      { stop when encounter null left pointer }
                      while (next <> nil )
                      do begin s.topptr := s.topptr + 1;
                               s.tack[s.topptr] : = next;
                               next := next ↑.left
                          end;
                      { visit node on top of stack }
                      if (s.topptr > 0 )
                      then begin
                              next := s.stack[s.topptr]
                              s.topptr := s.topptr - 1;
                              writeln(next ↑.info)
```

```
                    { traverse right subtree }
                    next := next↑.right
                    end;
                else goflas := false ; { stack is empty }
            end;
        end;
```

การแวะผ่านแบบหลังลำดับใน COBOL (COBOL Postorder traversal)

โปรซีเดอร์ไม่ใช่การเรียกซ้ำใน COBOL เพื่อแวะผ่านต้นไม้ทวิภาคแบบหลังลำดับซึ่งจะศึกษาต่อไปนี้ โปรซีเดอร์นี้เก็บต้นไม้ทวิภาคและกองซ้อนในแถวลำดับ สมมติว่าต้นไม้ทวิภาคมีโหนดมากที่สุด 500 ตัว และค่า0ใช้เพื่อจำลอง (simulate) พอยน์เตอร์ว่าง

ต้นไม้และกองซ้อนประกาศใน DATA DIVISION ดังนี้

```
01    BINARY-TREE.
      02    NODE        OCCURS    500   TIMES.
            03    LEFTN       PICTURE 9(3).
            03    INFO        PICTURE 9(5).
            03    RIGHTH      PICTURE 9(3).
01    AUX-PTRS.
      02    ROOT        PICTURE 9(3).
      02    NEXTN       PICTURE 9(3).
01    STACK-STRUCT.
      02    STACK       OCCURS 500 TIMES PICTURE 9(3).
      02    TOP-PTR     PICTURE 9(3) VALUE 0.
```

ใน PROCEDURE DIVISION เขียนดังนี้

```
POSTORDER.
      MOVE  ROOT TO NEXTN.
      PERFORM     LOOP.
      PERFORM     LOOP  UNTIL  TDP-PTR  = 0.
```

เมื่อ

```
LOOP.
        PERFORM     SAVENODE UNTIL NEXTN = 0.
        MOVE        STACK(TOP-PTR) TO NEXTN.
        COMPUTE     TOP-PTR = TOP-PTR -1.
        DISPLAY     INFO(NEXTN).
        MOVE        RIGHTN(NEXTN) TO NEXTN.
SAVENODE.
        COMPUTE     TOP-PTR = TOP-PTR+1.
        MOVE        NEXTN TO STACK(TOP-PTR).
        MOVE        LEFTN(NEXTN) TO NEXTN.
```

8.8 ต้นไม้ทวิภาคแบบ Threaded (Threaded Binary Trees)

โปรซีเดอร์แบบไม่ใช่การเรียกซ้ำสำหรับการแวะผ่านต้นไม้แบบทวิภาคซึ่งกำหนดให้ในหัวข้อที่แล้วไม่ใช่เรื่องเล็กน้อย (trivial) ที่จริงการใช้กองซ้อนเพื่อเก็บร่องรอย (track) ของโหนดที่ถูกเยี่ยมได้และโหนดที่ไม่ถูกเยี่ยม (โดยเฉพาะการแวะผ่านแบบหลังลำดับ)ยุ่งยาก(messy) มาก เพื่ออำนวยความสะดวกการแวะผ่านต้นไม้แบบทวิภาค บางครั้งเรา thread ต้นไม้ด้วยพอยน์เตอร์ซึ่งแสดงการเรียงอันดับการแวะผ่านอย่างชัดแจ้ง Intuitively, threads link โหนดของต้นไม้ในลำดับของวิธีการแวะผ่าน threads มีสองชนิดคือ

(1) right thread เชื่อมโหนดนั้นไปยังโหนดหลังของมัน (its successor node) ในการเรียวอันดับการแวะผ่าน

(2) left thread เชื่อมโหนดนั้นไปยังโหนดก่อนของมัน (its predecessor node) ในการเรียงอันดับการแวะผ่าน

จากบทนิยามเหล่านี้จะเห็นชัดเจนว่า ต้นไม้หนึ่งต้นสามารถถูก thread เป็นไปตามวิธีการแวะผ่านได้เพียงหนึ่งวิธีการแวะผ่านได้เพียงหนึ่งวิธีเท่านั้น เป็น thread แบบ preorder หรือ inorder หรือ postorder (แน่นอน threads มีหลายชุดสามารถนำมาใช้กับต้นไม้ชนิด multi-thread) ต้นไม้ค้นแบบทวิภาคปกติ threaded เป็นแบบ preorder

เราแทน threads ในรูปภาพด้วยพอยน์เตอร์เส้นปะ (dashed pointers)

รูป 8-19 (a) แสดงต้นไม้สายต้นของรูป 8-15 threadded แบบ preorder

รูป 8-19(b) แสดงต้นไม้สามต้นเดิม threaded แบบ inorder รูป 8-19 (c) แสดงต้นไม้สามต้นเดิน threaded แบบ postorder ทั้งนี้แสดงให้เห็นเฉพาะ right threads

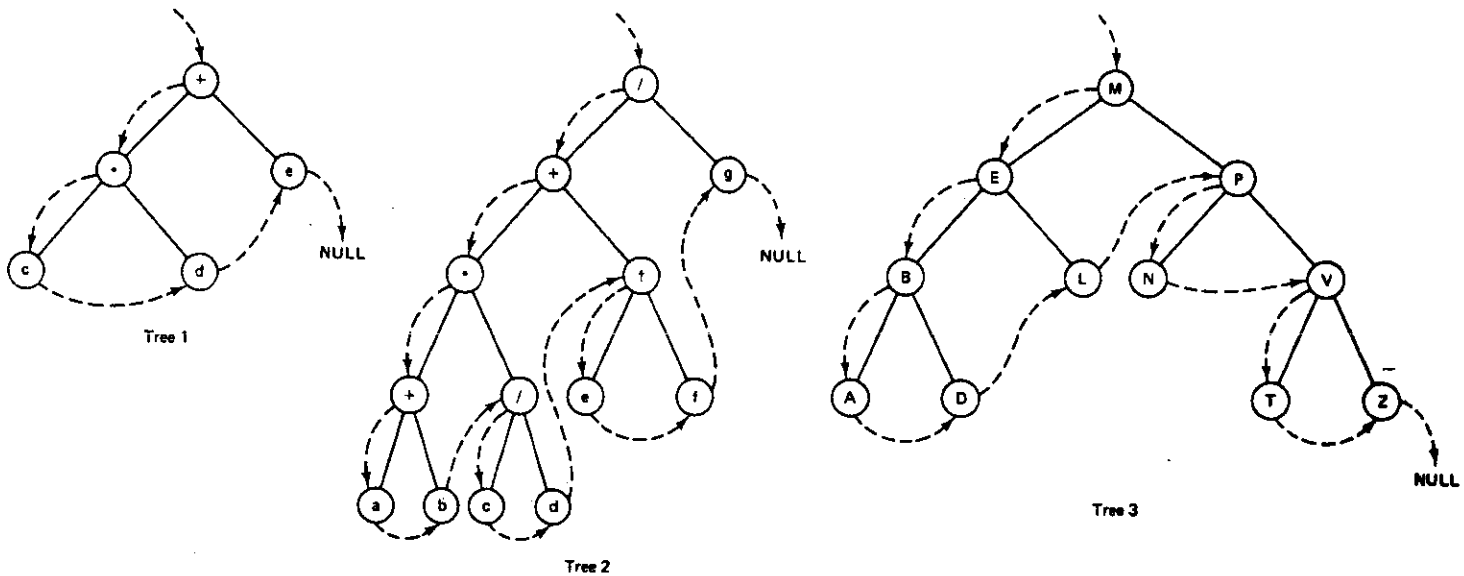


Figure 8–19(a) Trees of Fig. 8–15 threaded in pre-order.

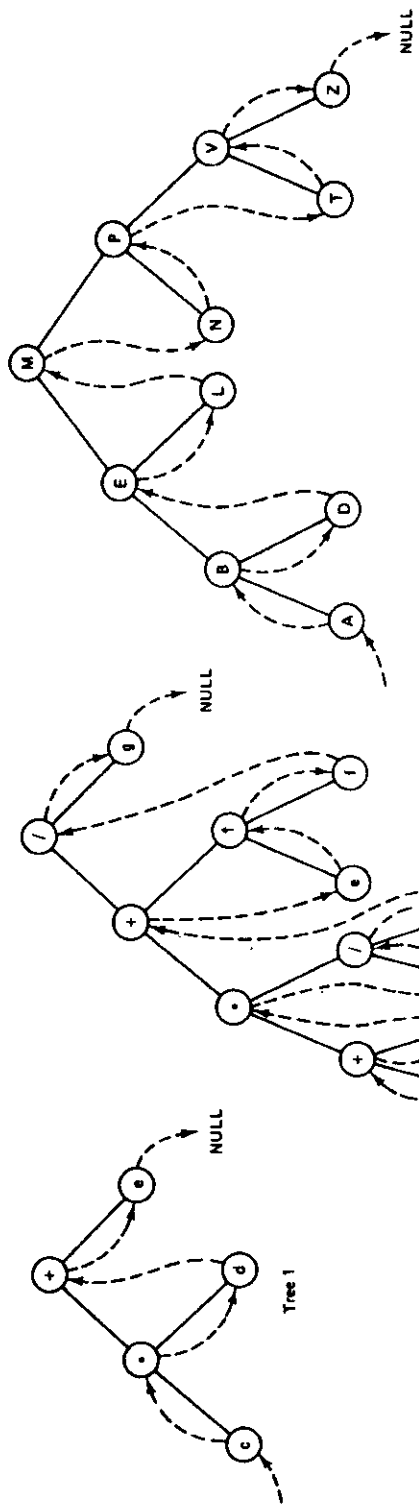


Figure 8–19(b) Trees of Fig. 8–15 threaded in in-order.

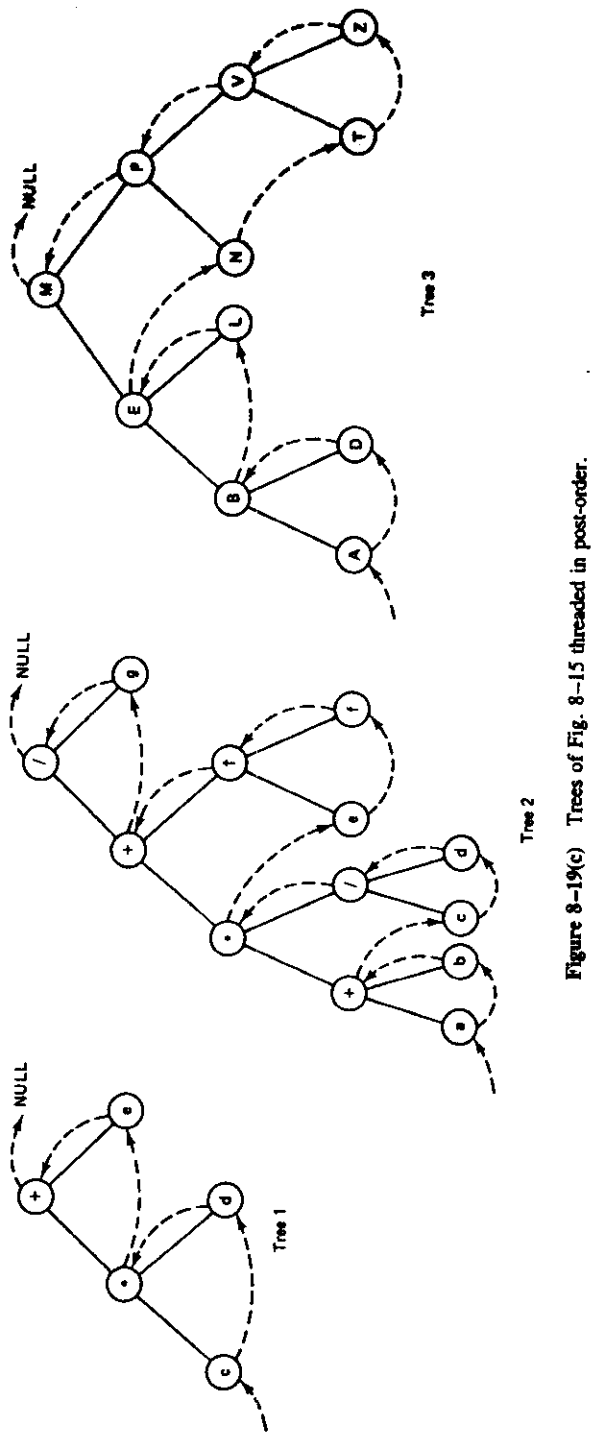


Figure 8–19(c) Trees of Fig. 8–15 threaded in post-order.

8.8.1 การแทนที่โหนด (Node Representation)

ทางเลือกหนึ่งสำหรับการแทนที่ต้นไม้ทวิภาคแบบ threaded อย่างง่าย ๆ คือให้โหนดแต่ละตัวมีโครงสร้างเป็นดังนี้

```
type    nodeptr = ↑ nodetype;
        nodetype = record
                        left : nodeptr;
                        info : integer;
                        right : nodeptr;
                        rightthread : nodeptr;
                        leftthread : nodeptr;
                   end;
```

โหนดแต่ละตัวมีพอยน์เตอร์ทางซ้ายและพอยน์เตอร์ทางขวาเหมือนปกติ เซต information ของมันและพอยน์เตอร์ thread ทางซ้ายและทางขวาของมัน พอยน์เตอร์ทั้งหมดที่ชี้ไปยังโหนดอื่น ๆ ซึ่งมีโครงสร้างเหมือนกันนี้ถ้าเป็นต้นไม้ค้นแบบทวิภาค โหนดแต่ละตัวจะมีเซตข้อมูลของคีย์ด้วย

8.8.2 การแวะผ่านแบบตามลำดับ (Inorder tranersal)

โปรซีเดอร์แบบไม่ใช่การเรียกซ้ำเพื่อแวะผ่านต้นไม้ทวิภาคแบบตามลำดับซึ่ง threaded แบบ inorder เขียนดังนี้ พอยน์เตอร์ firstin ชี้ที่โหนดตัวแรกของต้นไม้แบบ inorder ส่วน right threads ติดตามจากแต่ละโหนด ไปยังโหนดถัดไป

```
procedure inorder (firstin : nodeptr);
var p : nodeptr;
begin
        p := firstin;
        while p <> nil
        do begin
                writeln (p ↑.info) ;
                p := p↑.rightthread
           end;
end;
```

นี่คืออัลกอริทึมการแวะผ่านที่ง่าย

8.8.3 การแทนที่แบบทางเลือก (An Alternative Representation)

ทางเลือกที่สองสำหรับการแทนที่ต้นไม้ทวิภาคแบบ threaded ต้องการหน่วยเก็บที่น้อยกว่าสำหรับพอยน์เตอร์ แต่ทำให้การจัดการต้นไม้และอัลกอริทึมการแวะผ่านซับซ้อนมากกว่า สิ่งแรกจะมีพอยน์เตอร์ว่าง (null pointers) จำนวนเท่าใดในต้นไม้ทวิภาคแบบ unthreaded ที่มี n โหนดต้นไม้เช่นนี้มีพอยน์เตอร์ทั้งหมดเท่ากับ 2N (null และ non-null) : พอยน์เตอร์จำนวน N ชี้ไปยังต้นไม้ส่วนย่อยซ้ายและพอยน์เตอร์อีกจำนวน N ชี้ไปยังต้นไม้ส่วนย่อยขวา แน่นอนมีพอยน์เตอร์ชนิด non-null เพียง N-1 เท่านั้นเนื่องจากโหนดแต่ละตัวยกเว้นเฉพาะรากมีเพียงหนึ่งพอยน์เตอร์เท่านั้นที่ชี้ไปยังโหนดตัวนั้นเพราะฉะนั้นจึงมี พอยน์เตอร์ว่างเท่ากับ N+1 เสมอ

ถ้าเราตรวจสอบต้นไม้ซึ่ง threaded แบบ preorder จะพบว่า threads ของโหนดที่ไม่ใช่จุดแตกใบ (nonleaf nodes) จะขนานกับพอยน์เตอร์จริงทางซ้าย

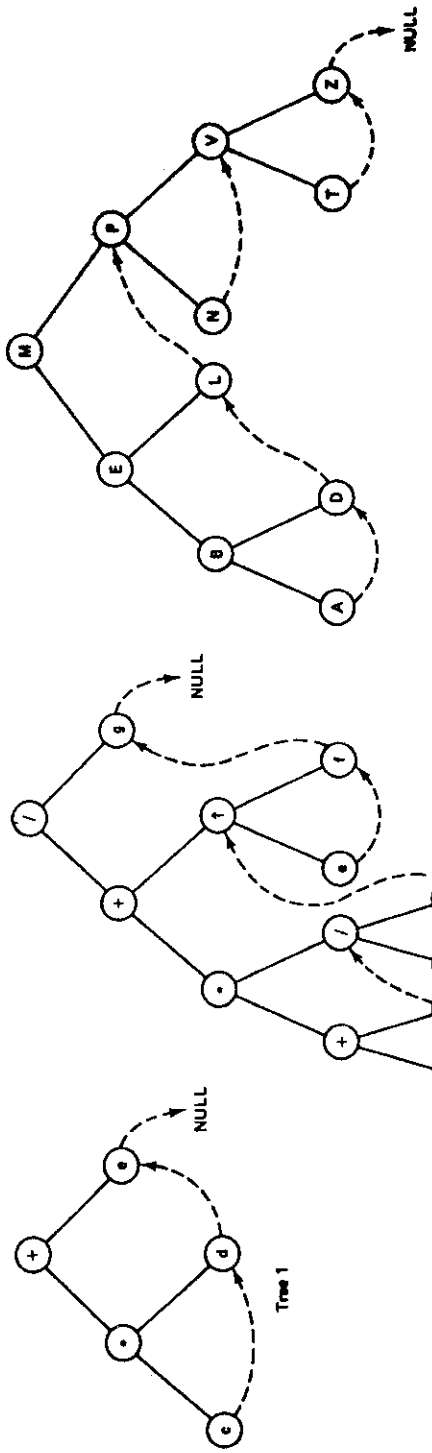


Figure 8–20 Alternative representation for threaded binary tree.

จริง ๆ แล้วไม่จำเป็นต้องทำซ้ำพอยน์เตอร์เหล่านี้ โหนดซึ่งเป็นจุดแตกใบจำเป็นต้อง threads มีพอยน์เตอร์ของต้นไม้ส่วนย่อยเป็น null ดังนั้นเราสามารถแทนต้นไม้ทวิภาค threaded แบบก่อนลำดับโดยให้โหนดแต่ละตัวมีโครงสร้างเป็นดังนี้

```
type nodeptr = ↑ nodetype;
     notdtype = record
                    left : nodeptr;
                    threadflag : 0..1;
                    info : integer ;
                    right : nodeptr
                end;
```

เพื่อป้องกันการสับสนเรื่อง threads กับพอยน์เตอร์ของต้นไม้ส่วนย่อย threadflag สามารถกำหนดให้ค่าเป็น 0 เมื่อ left ซึ่งแทนพอยน์เตอร์ไปยังต้นไม้ส่วนย่อยซ้าย กำหนดให้ค่าเป็น 1 เมื่อ left แทน right thread อีกทางเลือกหนึ่งเมื่อต้นไม้ทวิภาคซึ่งเก็บในแถวลำดับ พอยน์เตอร์ทางซ้ายสามารถแทนด้วยค่าดรรชนีล่างซึ่งเป็นบวก และ right threads สามารถแทนด้วยค่าดรรชนีล่างที่เป็นลบหรือเป็นจริงในทางกลับกัน ต้นไม้ในรูป 8-15 threaded ด้วยพอยน์เตอร์หนึ่งตัวเช่นที่แสดงในรูป 8-20

โปรซีเดอร์การแวะผ่านแบบก่อนลำดับเขียนดังนี้พอยน์เตอร์ทางซ้ายจากโหนดแต่ละตัวไปยังโหนดถัดไป บางครั้งพอยน์เตอร์เหล่านี้คือ right threads และบางครั้งมันเป็นพอยน์เตอร์จริงทางซ้าย

```
procedure preordthr (root : nodeptr);
var p : nodeprt;
    begin
        p:= root;
        while p<> nil
        do begin
            writeln( p↑.left);
            p:= p↑.left
          end;
    end;
```

ให้สังเกตความคล้ายกันของโปรซีเดอร์นี้กับโปรซีเดอร์การแวะผ่านแบบตามลำดับของต้นไม้ threaded

8.9 การค้นโดยตรง (Direct searches)

คุณสมบัติของต้นไม้ค้นแบบทวิภาคแสดงว่ามีโปรซีเดอร์สำหรับหาโหนดซึ่งกำหนดค่าคีย์ให้ว่ามีอยู่ในต้นไม้หรือไม่และสำหรับการหาโหนดนั้นเมื่อมันอยู่ในต้นไม้การหาโหนดที่มีค่าคีย์เท่ากับ k ในต้นไม้ค้นแบบทวิภาคซึ่งมีรากอยู่ที่ R_i ทำตามขั้นตอนดังนี้

1. ถ้าเป็นต้นไม้ว่างการค้นหาจบแบบไม่ประสบผลสำเร็จ (unsuccessfully)
2. ถ้า $k = K_i$, การค้นหาจบแบบประสบผลสำเร็จ (successfully); โหนดตัวที่ค้นหาคือ R_i
3. ถ้า $k < K_i$, ค้นหาต้นไม้ส่วนย่อยซ้ายของ R_i นั่นคือ

   $R_i := left(R_i)$
4. 4. ถ้า $k > K_i$, ค้นหาต้นไม้ส่วนย่อยขวาของ R_i นั่นคือ

   $R_i := right(R_i)$

อัลกอริทึมนี้เขียนโปรแกรมเป็นโปรซีเดอร์แบบเรียกซ้ำใน Pascal เมื่อเริ่มต้น r ชี้ที่รากของต้นไม้

```
procedure search (k: nodekey,var r, foundnode: nodeptr);
begin
        if (r= nil)
        then foundnode := nil
        else if (k= r ↑.key)
              then  foundnode := r
              else if (k <  r ↑.key)
                    then  search (k, r ↑.left, foundnode)
                    else { k > r ↑.key }
                          search (k , r ↑.right, node)
end;
```

การค้นจบด้วย foundnode ซึ่งชี้ไปยังโหนดที่ต้องการค้นหาถ้ามีอยู่จริงหรือ จบด้วย null value ถ้าคีย์ k ไม่อยู่ในต้นไม้

การค้นหาโหนดเฉพาะในต้นไม้ซึ่งไม่ใช่ต้นไม้ค้นแบบทวิภาค ปกติกระทำด้วยการค้นแบบลำดับไม่ใช่การค้นโดยตรง ยกเว้นต้นไม้ที่มีโครงสร้างพิเศษ และโหนดมีเขตข้อ

มูลของคีย์ การค้นแบบลำดับของต้นไม้ที่มี N โหนดจะต้องมีการเยี่ยมโดยเฉลี่ย N/2 โหนดเพื่อหาโหนดเฉพาะ ในกรณีแย่ที่สุดโหนดทั้งหมด N ตัวจะต้องถูกแวะผ่านเมื่อโหนดที่ต้องการค้นนั้นไม่ได้อยู่ในต้นไม้

ความพยายามที่ต้องใช้ในการหาระเบียนเฉพาะ (a particular record) ในต้นไม้ค้นแบบทวิภาคขึ้นอยู่กับตำแหน่งของระเบียนในต้นไม้ เราได้เห็นแล้วว่ามีต้นไม้ค้นแบบทวิภาคที่เป็นไปได้จำนวนมากที่จัดระเบียบกลุ่มของระเบียน ระเบียนที่ต้องการค้นหายิ่งอยู่ไกลจากรากของต้นไม้เท่าไหร่ความพยายามที่ต้องใช้เพื่อหาระเบียนนั้นยิ่งมากขึ้นความพยายามการค้นถูกวัดโดยจำนวนของทำการเปรียบเทียบก่อนจบการค้นไม่ว่าจะประสบผลสำเร็จหรือไม่ประสบผลสำเร็จตาม (The search effort is measured by the number of comparisons made before the search terminates either successfully or unsuccessfully.)

ตัวอย่างเช่น การหาระเบียนชื่อ DOG ในต้นไม้ค้นแบบทวิภาคหลากหลายในรูป 8-17 ต้องการจำนวนครั้งการเปรียบเทียบดังนี้

ต้นไม้ (a) 1 comparison

(b) 4 comparison

(c) 3 comparison

(d) 2 comparison

ถ้ามีความจำเป็นบ่อยครั้งเพื่อหา DOG เพราะฉะนั้นต้นไม้ (a) น่าจะเป็นต้นไม้ค้นแบบทวิภาคที่ดีกว่าต้นไม้ (b) โดยทั่วไป เราจะทราบได้อย่างไรว่าต้นไม้ค้นแบบทวิภาคต้นหนึ่งดีกว่าต้นไม้ค้นแบบทวิภาคอีกต้นหนึ่ง , ต้นไม้ค้นแบบทวิภาคที่ดีหมายถึงอะไร, ต้นไม้ค้นแบบทวิภาคที่ดีจะสร้างอย่างไร

ต้นไม้ค้นแบบทวิภาคไม่สามารถถูกประเมินบนหลักการของทางเดินการค้นซึ่งจัดหาเพียงหนึ่งระเบียน แต่จะเป็นทางเดินการค้นตลอดต้นไม้ทั้งต้นควรจะใส่เข้าไปในการประเมินผล ดังนั้นความยาวที่คาดคะเนเช่น ค่าเฉลี่ยน้ำหนัก) ของการเดินการค้นในต้นไม้จะเป็นประโยชน์

ให้ S_i เป็นความน่าจะเป็นซึ่งค้นพบคีย์ K_i

เมื่อ K_i ($i = 1, 2, \ldots, n$) คือตัวใดตัวหนึ่งของคีย์ในต้นไม้ กำหนดให้การค้นประสบผลสำเร็จนั่นคือ

$$\sum_{i=1}^{n} S_i = 1$$

ขั้นแรก พิจารณาการค้นที่ประสบผลสำเร็จ ความยาวการค้นที่คาดคะเนสำหรับต้นไม้ค้นแบบทวิภาคคือ

$$\sum_{i=1}^{n} S_i C_i$$

เมื่อ C_i หมายถึงจำนวนของการเปรียบเทียบที่ต้องใช้เพื่อเข้าถึงคีย์ K_i , และ $C_i - 1$ หมายถึงจำนวนของระดับซึ่งโหนด R_i อยู่ ตัวอย่างเช่น ถ้าความน่าจะเป็นของการเข้าถึงสำหรับคีย์ซึ่งมีโครงสร้างในต้นไม้ค้นแบบทวิภาค รูป 8-17 คือ

APE	.2
BEE	.4
COW	.3
DOG	.1

ความยาวการค้นที่คาดคะเน(the expected search lengths) สำหรับต้นไม้เหล่านี้คือ

	APE		BEE		COW		DOG		
(a)	.2 * 4	+	.4 * 3	+	.3 * 2	+	.1 * 1	=	2.7
(b)	.2 * 1	+	.4 * 2	+	.3 * 3	+	.1 * 4	=	2.3
(c)	.2 * 2	+	.4 * 1	+	.3 * 2	+	.1 * 3	=	1.7
(d)	.2 * 2	+	.4 * 1	+	.3 * 3	+	.1 * 2	=	1.9

โปรดสังเกตว่าในที่นี้เพื่อลดความยาวการค้นคาดคะเน, ต้นไม้ต้องมีโครงสร้างเพื่อให้ชื่อที่ถูกเข้าถึงบ่อย มากที่สุดอยู่ในตำแหน่งใกล้รากมากที่สุดเท่าที่จะเป็นไปได้ เพื่อให้สิ่งนี้เป็นผลสำเร็จ คีย์ต้องถูกใส่ในต้นไม้ในอันดับถูกต้อง จะเห็นชัดเจนว่ามีปัญหากับการเข้าถึงนี้ถ้าไม่ทราบค่า (unknown) ความน่าจะเป็นของการเข้าถึง ในสถานะการณ์เหล่านี้บางครั้งนักออกแบบสามารถเดาความถี่สัมพัทธ์ของการเข้าถึง จากนั้นทำโครงสร้างใหม่ให้กับต้นไม้หลังจากวัดการใช้จรงได้ถูกรวบรวมแล้ว

โปรดสังเกตด้วยว่าความยาวทางเดินคาดคะเนของต้นไม้ค้นแบบทวิภาคจะเปลี่ยนขณะใส่โหนดเข้าไปในต้นไม้ อะไรคือโครงสร้างต้นไม้ที่ดีในวันนี้บางทีพรุ่งนี้อาจเป็นต้นไม้ที่มีโครงสร้างค่อยข้างแย่เพราะการใส่โหนด และ/หรือ การเปลี่ยนแปลงในรูปแบบของการเข้าถึงมูลค่าของต้นไม้

ก่อนที่จะอภิปรายว่าการเปลี่ยนแปลงเหล่านี้จะทำให้เหมาะสมอย่างไร เราควรจะพิจารณาถึงผลกระทบของการค้นหาที่ไม่ประสบผลสำเร็จด้วย ให้ u_i เป็นความถี่ของการค้นไม่ประสบผลสำเร็จ จบที่โหนด R_i โปรดสังเกตว่า $u_i = 0$ สำหรับโหนด R_i ซึ่งมีต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาเป็น non-null ทั้งคู่เพราะว่าการค้นไม่เคยจบแบบไม่ประสบผลสำเร็จกับโหนดเช่นนั้น ดังนั้นความยาวการค้นคาดคะเนคือ

$$p_s \sum_{i=1}^{n} s_i c_i + p_u \sum_{i=1}^{n} u_i c_i$$

เมื่อ p_s หมายถึงความน่าจะเป็นของการค้นที่ประสบผลสำเร็จ

เมื่อ p_u หมายถึงความน่าจะเป็นของการค้นที่ไม่ประสบผลสำเร็จ

และ $p_s + p_u = 1$

ในงานประยุกต์บางอย่าง p_u อาจจะค่ามากกว่า p_s มากตัวอย่างเช่น จงพิจารณาต้นไม้ค้นแบบทวิภาค ซึ่งเก็บจำนวนบัตรเครดิตที่ถูกชโมยในระบบการจัดการเครดิตแบบใต้ตอบ ก่อนทำการขายพนักงานร้านจะคีย์หมายเลขบัตรเครดิตของลูกค้า บนเทอร์มินอลและระบบจะค้นหาหมายเลขนั้นในต้นไม้ เกือบจะทุกกรณี การค้นจะไม่ประสบผลสำเร็จความยาวการค้นคาดคะเนสามารถทำให้ต่ำสุด โดยทำให้การค้นซึ่งไม่ประสบผลสำเร็จเหล่านี้จบด้วยจำนวนครั้งการเปรียบเทียบน้อยเท่าที่เป็นไปได้ สิ่งนี้แสดงว่าต้นไม้ควรจะเป็นพุ่ม (bushy) เท่าที่เป็นไปได้ ในที่นี้ความยาวการค้นคาดคะเนจะต่ำสุดเมื่อความยาวทางเดินมากที่สุดในต้นไม้มีค่าต่ำสุด โปรดสังเกตว่าในที่นี้มีปัญหาของการจัดการการใส่ซึ่งคาดการณ์ไม่ได้เข้าไปในต้นไม้ในวิธีซึ่งความสามารถของการค้นหาไม่แย่ลง (deteriorate)

8.10 การใส่โหนด (Inserting Nodes)

การดำเนินการของการใส่โหนดในต้นไม้แบบทวิภาคและการลบโหนดออกจากต้นไม้แบบทวิภาคเป็นเรื่องตรงไปตรงมา ตัวอย่างเช่น จงพิจารณา การใส่โหนดที่มีชื่อ GNU ซึ่งจะกลายเป็นรากของต้นไม้ส่วนย่อยขวาของโหนด x ในต้นไม้แบบทวิภาค มีสองกรณี : GNU อาจจะเป็นจุดแตกใบหรือ GNU อาจจะไม่ใช่จุดแตกใบ ขึ้นอยู่กับว่า ต้นไม้ส่วนย่อยขวาของ x เริ่มต้นว่างเปล่าหรือไม่ ผลลัพธ์ที่เป็นไปได้ของการใส่ GNU แสดงไไเห็นในรูป 8-12(a)

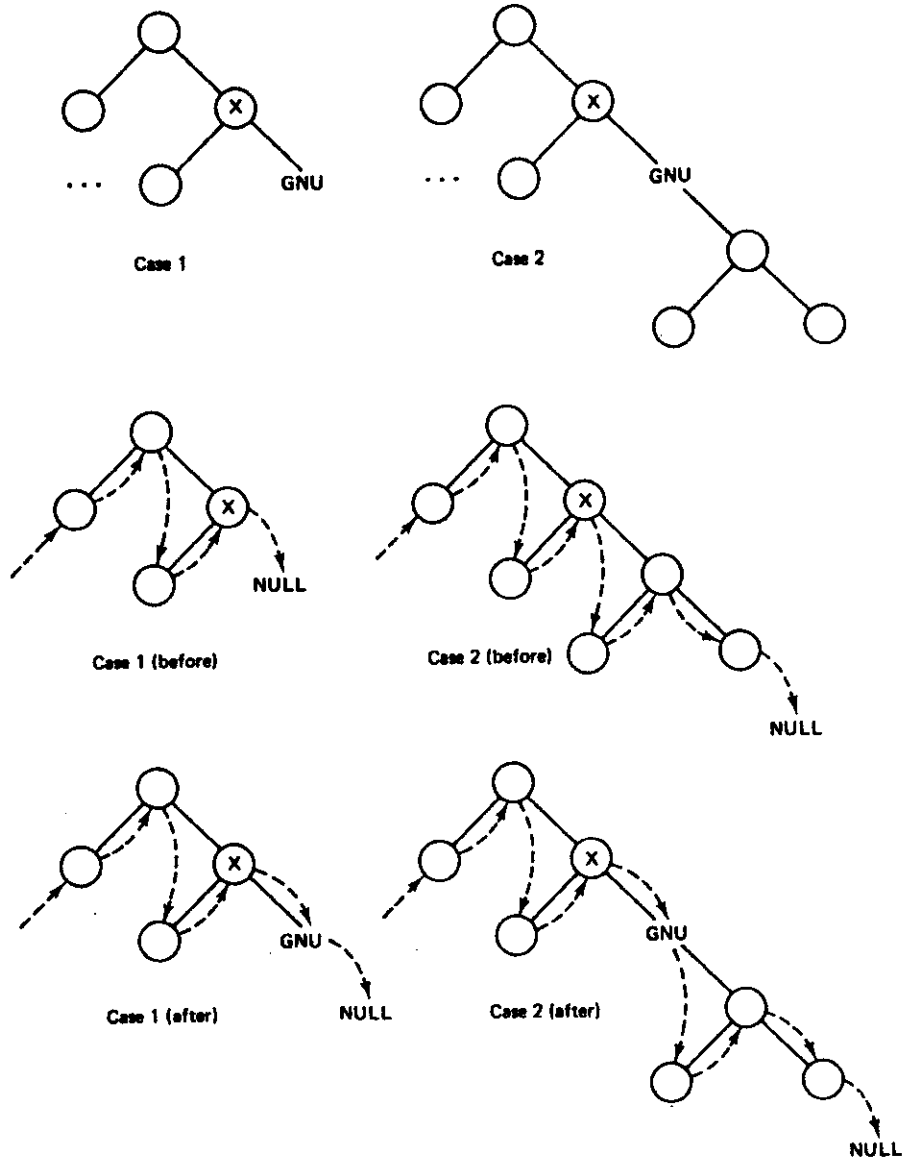


Figure 8–21 Inserting a node.

8.10.1 การใส่แบบ Unthreaded

โปรซีเดอร์ Pascal เพื่อทำการใส่เหล่านี้ในต้นไม้แบบ unthreaded เขียนดังนี้

```
procedure insert (x : nodeptr ; var gnu : nodeptr);
begin
        new(gnu);
        gnu↑.left := nil;
        gnu↑.left := x↑.right;
        x↑.right := gnu
end;
```

สิ่งที่ยากมากกว่าของการใส่โหนดในต้นไม้คือการหาว่ามันควรจะไปที่ใด

8.10.2 การใส่แบบ Threaded

ถ้าเป็นต้นไม้ Threaded, โปรซีเดอร์การใส่จะซับซ้อนเล็กน้อย รูป 8-12(b) แสดงต้นไม้ของรูป 8-21(a) ก่อนและหลังการใส่ GNU เมื่อต้นไม้ถูก threaded แบบตามลำดับ โปรซีเดอร์การใส่จะเขียนดังนี้

```
procedure insert (x : nodeptr; var gnu : nodeptr);
begin
        new(gnu);
        gnu↑.left := nil;
        gnu↑.right := x↑. right ;
        gnu↑.leftthread := x;
        gnu↑.rightthread:= x↑.rightthread;
        x↑.right := gnu;
        x↑.rightthread := gnu
end;
```

ให้นักศึกษาพัฒนาอัลกอริทึมการใส่สำหรับต้นไม้ทวิภาคแบบ threaded แบบก่อนลำดับและแบบหลังลำดับสำหรับกรณีก่อนลำดับจงบอกความแตกต่าง (ถ้ามี) ซึ่งจำเป็นเมื่อ threadflag ที่ใช้เพื่อให้สังเกตว่า left เป็นพอยน์เตอร์ของต้นไม้ส่วนย่อยซ้ายหรือเป็น rightthread

8.11 การใส่โหนดในต้นไม้ค้นแบบทวิภาค (Inserting Nodes in a Binary Search Tree)

โหนดตัวใหม่ในต้นไม้ค้นแบบทวิภาคปกติจะใส่ในตำแหน่งของจุดแตกใบเสมอ โครงสร้างของต้นไม้ถูกบังคับอย่างบริบูรณ์โดยคุณสมบัติของต้นไม้ค้นแบบทวิภาค และโดยอันดับการใส่โหนด อัลกอริทึมการใส่จะคล้ายกับอัลกอริทึมการค้นโดยตรงมาก การใส่โหนดตัวใหม่ที่มีคีย์เป็น k ในต้นไม้ค้นแบบทวิภาคที่รากอยู่ที่ R_i ทำตามขั้นตอนข้างล่างนี้

1. ถ้าเป็นต้นไม้ว่าง โหนดใหม่ที่มีคีย์เป็น k จะกลายเป็น root
2. ถ้า k= K_i การใส่จบแบบ unsuccessfully; คีย์มีอยู่แล้วในต้นไม้
3. ถ้า k < K_i, ต้นไม้ส่วนย่อยซ้ายของ R_i จะถูกค้นหาจนกระทั่งพบตำแหน่งที่ถูกต้องสำหรับโหนดตัวใหม่
4. ถ้า k > K_i, ต้นไม้ส่วนย่อยขวาของ R_i จะถูกค้นหาจนกระทั่งพบตำแหน่งที่ถูกต้องสำหรับโหนดตัวใหม่

อัลกอริทึมนี้เขียนเป็นโปรแกรมง่ายมากโดยใช้โครงสร้างเชิงตรรกะเหมือนกับที่ทำมาแล้วในการเขียนโปรซีเดอร์เพื่อค้นหาต้นไม้ค้นแบบทวิภาค ให้ newptr เป็นพอยน์เตอร์ของโหนดซึ่งจะใส่ในต้นไม้ค้นแบบทวิภาคที่ r

ใน Pascal เขียนดังนี้

```
procedure insert (newptr, var r : nodeptr);
begin
        if (r = nil)
        then r := newptr
        else if (newptr↑.key = r↑.key)
             then DUPLICATE-ENTRY
             else if (newptr↑.key < r↑.key)
                  then insert (newptr, r↑.left)
                  else { newptr↑.key > r↑. key }
                       insert (newptr, r↑. right)
end;
```

เงื่อนไขที่ทำให้โปรซีเดอร์ search จบแบบ unsucessfully คือเงื่อนไขที่ทำให้โปรซีเดอร์ insertion จบแบบ successfully และเงื่อนไขที่ทำให้การค้นหาจบแบบ successfully คือเงื่อนไขที่ทำให้การใส่จบแบบ unsuccessfully

โปรดสังเกตว่าการใส่ keys ในต้นไม้ค้นแบบทวิภาคทำให้ผลลัพธ์เป็นต้นไม้สูง (a long tree) ซึ่งไม่มีการแตกกิ่ง (ดูต้นไม้ (b) รูป 8-17)

8.12 การลบโหนด (Deleting Nodes)

การลบแบบ threaded (Threaded Removal)

การลบโหนดจากต้นไม้แบบทวิภาคคล้ายกับการใส่ซึ่งเกี่ยวกับ "fixing up" พอยน์เตอร์ของต้นไม้ส่วนย่อยและ threads จากประสบการณ์ในเรื่องรายการโยงทำให้เห็นชัดเจนว่าเป็นอะไรที่ง่ายกว่าเพื่อลบโหนดเมื่อการโยงต้นไม้ส่วนย่อยเป็นสองทิศทาง การทำให้เกิดผลในทางปฏิบัติของการโยงต้นไม้ส่วนย่อยเชิงสองทิศทาง ต้องการเพียงเพิ่มเซตข้อมูลหนึ่งพอยน์เตอร์ให้กับแต่ละโหนด : ให้กับ parent ของโหนดเท่านั้น

ใน Pascal, เขียนดังนี้

```
type nodeptr = ↑ nodetype;
     nodetype = record
                    left : nodeptr;
                    info : integer;
                    right : nodeptr;
                    leftthread : nodeptr;
                    rightthread : nodeptr;
                    parent : nodeptr;
                end;
```

การลบโหนด(ชื่อ x) ออกจากต้นไม้แบบทวิภาคจะซับซ้อนมากกว่าถ้าโหนดนั้นไม่ใช่จุดแตกใบ การตัดสินใจต้องกระทำเกี่ยวกับการจัดวาง (disposition) ของลูกของโหนดที่ลบ (the removed node's children) อาจจะเป็นกรณีที่โหนด offspring เหล่านี้ควรจะถูกลบออกจากต้นไม้ด้วยความยากในการทำให้เกิดผลนโยบายนี้คือ การ fixing up ส่วนที่เป็น threads.

ทางเลือกอีกวิธีหนึ่งคือย้าย (move) รากของต้นไม้ส่วนย่อยซ้าย หรือ ต้นไม้ส่วนย่อยขวาของ x ไปที่ตำแหน่ง x สิ่งนี้ง่าย(ยกเว้นสำหรับ threads) เมื่อมีเพียงหนึ่งของโหนดเหล่านี้เป็น non-null ถ้าเป็น non-null ทั้งคู่ทางเลือกระหว่างสิ่งนี้ต้องถูกกระทำ ถ้าตัวเลือกมีทั้งต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาของมันเป็น non-null ความยากเกิดขึ้นอีก ทางเลือกที่มีเหตุผลในสถานการณ์บางอย่างคือหลีกเลี่ยงปัญหาเหล่านี้โดยไม่อนุญาติให้เป็นโหนดที่จะลบออกยกเว้นเฉพาะมันเป็นจุดแตกใบ

8.13 การลบโหนดออกจากต้นไม้ค้นแบบทวิภาค (Deleting Nodes From a Binary Search Tree)

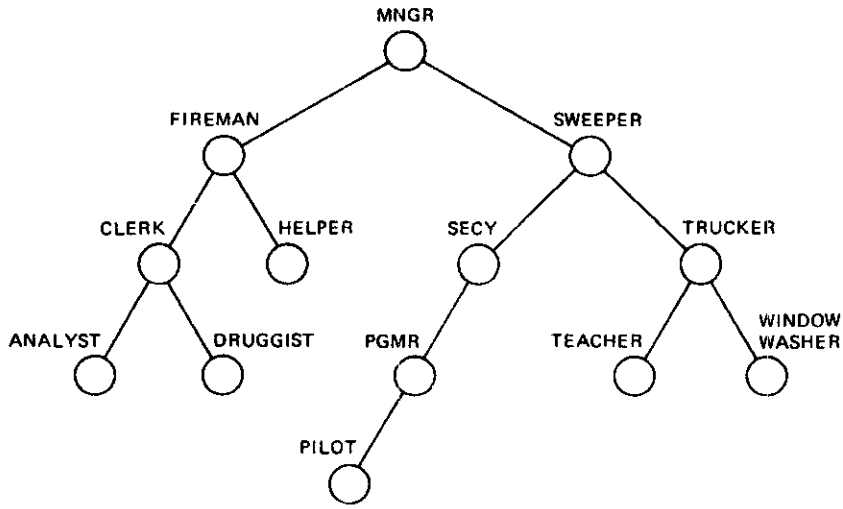
การปฏิบัติการของการลบโหนดออกจากต้นไม้ค้นแบบทวิภาคเป็นเรื่องซับซ้อน ถ้าโหนดที่จุดถูกลบคือจุดแตกใบ ดังนั้นกระบวนการจะง่าย; จุดแตกใบถูกตัดออกจากต้นไม้ ถ้าโหนดที่จะลบออกไม่ใช่จุดแตกใบ ดังนั้นกระบวนการต้องทำบางสิ่งเพื่อรักษาต้นไม้ส่วนย่อยของโหนดนั้น ขั้นแรกจงพิจารณาการลบโหนด PGMR ออกจากต้นไม้ค้นแบบทวิภาครูป 8-22 การลบ PGMR ไม่ได้หมายความว่าให้ลบ PILOT ด้วยแต่ PILOT จะต้องย้ายไปอยู่ตำแหน่งที่ว่างของ PGMR และกลายเป็นต้นไม้ส่วนย่อยซ้ายของ SECY ในทำนองเดียวกัน ถ้าเป็นการลบ SECY ดังนั้น PGMR จะต้องย้ายไปอยู่ตำแหน่งที่ว่าง จากนั้นต้นไม้ที่มีรากเป็น PGMR จะกลายเป็นต้นไม้ส่วนย่อยซ้ายของ SWEEPER จะเห็นว่าการลบโหนดที่มีหนึ่ง null subtree โดยนัยคือการย้ายรากของ non-null subtree ของโหนดนั้นไปยังตำแหน่งว่างของโหนด

ขณะนี้ให้พิจารณาการลบโหนดที่มีคีย์เป็น CLERK รากของต้นไม้ส่วนย่อยสองชุดนั้นชุดใดควรจะไปแทนที่ตำแหน่งว่างของมัน ไม่ว่าจะเป็นการเคลื่อนย้ายใดเข้ามาจะต้องไม่ฝ่าฝืนกฎของต้นไม้ค้นแบบทวิภาค

การลบโหนด MNGR จะยากมากกว่าถ้า FIREMAN เข้าไปแทนที่ตำแหน่งของ MNGR, ต้นไม้ส่วยย่อยของมันซึ่งมีรากอยู่ที่ CLERK และ HELPER จะจัดการอย่างไรให้นักศึกษาพัฒนาอัลกอริทึมซึ่งจะจัดกระทำการลบหนึ่งโหนดจากตำแหน่งใด ๆ ก็ได้ในต้นไม้ค้นแบบทวิภาคสิ่งแรกให้มั่นใจว่านักศึกษามีความเข้าใจความซับซ้อนของปัญหา

การเข้าถึงอีกวิธีหนึ่งของการลบคือหลีกเลี่ยง(อย่างน้อยที่สุดเป็นการชั่วคราว) ความซับซ้อนของปัญหาง่าย ๆ โดยการทำเครื่องหมาย โหนดที่ต้องการให้อยู่นิ่ง(inactive) การค้นหรือการใส่ภายหลังจำเป็นต้องใช้คีย์ของโหนดที่ถูกทำเครื่องหมายเพื่อหาทางเดินของมันผ่านต้นไม้ ถ้าคีย์ที่มองหาโดยโปรซีเดอร์การค้นหรือการใส่จับคู่กัน (matches) กับคีย์ของโหนดถูกทำเครื่องหมายหลังจากนั้นโปรซีเดอร์ต้องทำ action ที่แตกต่างจากที่เคยทำปกติ นักศึกษาควรจะดัดแปร (modify) รูทีนการค้นและการใส่เพื่อให้ครอบคลุมถึง (accommodate) โหนดซึ่งทำเครื่องหมายด้วย

การรวบรวมของ keyed data ซึ่งจะเข้าถึง(โดยตรง)อย่งรวดเร็วกับระเบียนเฉพาะและการเข้าถึงแบบลำดับผ่านกลุ่มของระเบียนปกติต้องใช้โครงสร้างเป็นต้นไม้ค้นแบบทวิภาค



รูป 8-22 ต้นไม้ค้นแบบทวิภาค

8.14 ต้นไม้ค้นทวิภาคแบบได้ดุล (Balancing Binary Search Trees)

เราได้เห็นความสามารถ (performance) ของต้นไม้ค้นแบบทวิภาคซึ่งคำนวณโดยโครงสร้างของต้นไม้และความน่าจะเป็นของการเข้าถึงโหนดต่างๆ โครงสร้างต้นไม้ถูกกระทบโดยลำดับของการดัดแปรต้นไม้ทั้งการใส่และการลบทั้งคู่ซึ่งเปลี่ยนแปลงรูปร่างของต้นไม้

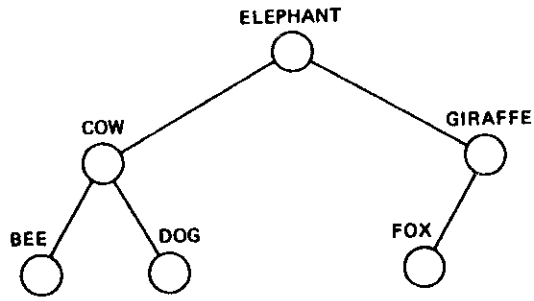
ที่จริงการประสบผลสำเร็จของโครงสร้างต้นไม้ค้นแบบทวิภาคซึ่งเหมาะที่สุดอาจต้องทำงานมากกว่าคุณค่าของมัน ความจริงมันอาจจะเป็นไปไม่ได้เพราะว่าขาดแคลนความรู้ที่พอเพียงความน่าจะเป็นของการเข้าถึง การสอบสวนได้แสดงให้เห็นว่าอย่างไรก็ตามความสามารถของต้นไม้ค้นแบบทวิภาคซึ่งสร้างโดย อัลกอริทึม heuristic มี่มีเหตุมีผล ถูกแข่งขันกับต้นไม้เหมาะที่สุด (optimal tree) ขณะที่ทำให้เกิด(incurring) ค่าใช้จ่ายในการออกแบบต่ำกว่ามาก heuristics ที่เห็นชัดเจนมากที่สุดสองข้อได้แก่สิ่งที่จะแนะนำดังนี้

1. ใส่โหนดที่มีการเข้าถึงคีย์บ่อย ๆ ใกล้กับรากของต้นไม้ (Put the nodes with frequently accessed keys near the root of the tree.)
2. รักษาต้นไม้ให้ได้ดุล (Keep the tree balanced)นั่นคือสำหรับโหนดแต่ละตัวต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาควรจะมีจำนวนโหนดเกือบเท่ากันเท่าที่จะเป็นไปได้ ด้วยเหตุผลนี้ความยาวทางเดินสูงสุดจึงทำให้ต่ำสุด

ถ้าทราบความน่าจะเป็นของการเข้าถึงและไม่มีการใส่ใด ๆ ซึ่งจะกระทำในต้นไม้ , ต้นไม้ค้นแบบทวิภาคซึ่งให้ความสามารถเหมาะที่สุดสามารถสร้างได้(ด้วยงานบางอย่าง)และอาจจะดีกว่าต้นไม้ซึ่งเป็นผลลัพธ์จากหนึ่งใน heuristic เหล่านั้น อย่างไรก็ตาม heuristics ดีมากและต้องการความพยายามน้อยกว่าเพื่อประยุกต์ใช้ซึ่งแสดงให้เห็นว่าโดยทั่วไปต้นไม้ค้นแบบทวิภาคสร้างจาก heuristic ครั้งที่สอง- การได้ดุล-ให้ความยาวการค้นที่คาดคะเนเหมาะที่สุด; โดยทั่วไปต้นไม้ได้ดุลกระทำได้ดีกว่าต้นไม้ซึ่งสร้างจาก heuristic ครั้งที่หนึ่ง ผลลัพธ์นี้คือบางสิ่งสะดวกในความน่าจะเป็นของการเข้าถึงซึ่งต้องการใช้โดย heuristic ที่หนึ่งซึ่งปกติไม่มีให้ใช้

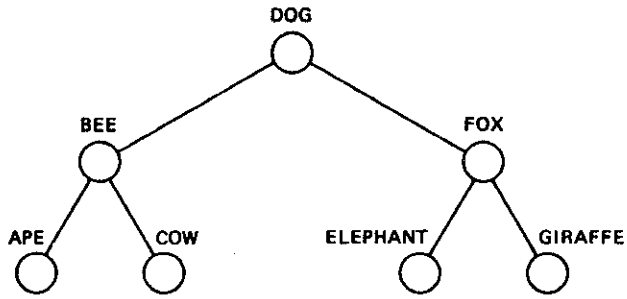
สำหรับรายละเอียดของการวิเคราะห์อัลกอริทึม heuristic ที่หลากหลายสำหรับการสร้างต้นไม้ค้นแบทวิภาค

การรักษาต้นไม้ค้นแบบทวิภาคให้ได้ดุล เมื่อมีการใส่โหนดและลบโหนดเป็นงานยาก (hard work) ความยากเนื่องจากต้องรักษา(preserve) ความสัมพันธ์ที่ถูกต้องระหว่างคีย์ในโหนดและคีย์ในต้นไม้ส่วนย่อยของมัน ตัวอย่างเช่น จงพิจารณาต้นไม้ค้นแบบทวิภาคของรูป 8-23



รูป 8-23 ต้นไม้ค้นแบบทวิภาค

การใส่ SEBRA ง่าย มันจะเป็นต้นไม้ส่วนย่อยขวาของ GIRAFFE และต้นไม้ยังคงได้ดุล อย่างไรก็ตามการใส่ APE ตำแหน่ง ของ APE จะเป็นต้นไม้ส่วนย่อยซ้ายของ BEE ซึ่งจะทำให้ต้นไม้ไม่ได้ดุล เมื่อต้นไม้ได้ดุลจะแสดงในรูป 8-24 โหนดทั้งหมดของต้นไม้เดิม (original tree) ต้องมีการเคลื่อนที่ปัญหาที่คล้ายกันอาจเกิดขึ้นกับการลบโหนดออกจากต้นไม้



รูป 8-24 ต้นไม้ค้นทวิภาคแบบได้ดุล

เพื่อให้ลดปริมาณงานที่ต้องใช้เพื่อรักษาต้นไม้ค้นแบบทวิภาคขณะที่จัดหาความสามารถการค้นที่ดี ปกติเราลดข้อจำกัดและยอมให้ต้นไม้หันเห (deviate) จากการได้ดุล อย่างไรก็ตาม การหันเหที่ยอมให้จะน้อยพอเพียงที่จะคาดคะเนความยาวการค้นเฉลี่ย คือ เฉพาะความยาวกว่าต้นไม้ได้ดุลบริบูรณ์กว่านั้น ในสองหัวข้อถัดไปจะแนะนำต้นไม้ค้นแบบทวิภาคสองชนิดซึ่งยอมให้ดัดแปรค่อนข้างง่าย

8-15 ต้นไม้ความสูงได้ดุลหรือต้นไม้ AVL (Heiht-Balanced (AVL) Trees)

ต้นไม้ไดุดุลซึ่งเกือบจะบริบูรณ์ชนิดหนึ่งเรียกว่าต้นไม้ความสูงได้ดุลหรือ ต้นไม้ AVL หลังจาการค้นพบโดยชาวรัสเซียสองคนชื่อ G.M. Adelson-Velski และ E.M. Landis (1962)

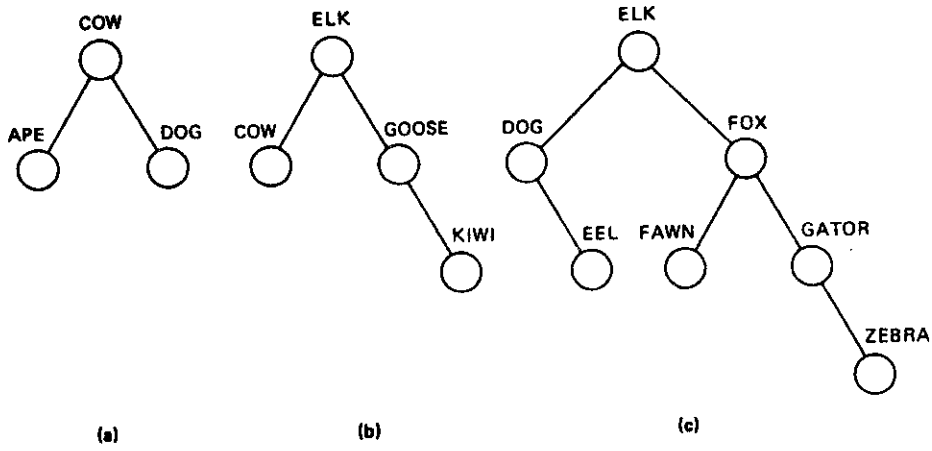
ต้นไม้จะเรียกว่าต้นไม้ความสูงได้ดุล ถ้าโหนดของมันถูกจัดการโดยที่ โหนดแต่ละต้ว R_i, มีคุณสมบัติดังนี้

ความสูงของต้นไม้ส่วนย่อยซ้ายของ R_i และความสูงของต้นไม้ส่วนย่อยขวา R_i แตกต่างมากที่สุดเท่ากับ 1 (The height of the leftsubtree of R_i and the height of the rightsubtree of R_i differ by at most 1.)

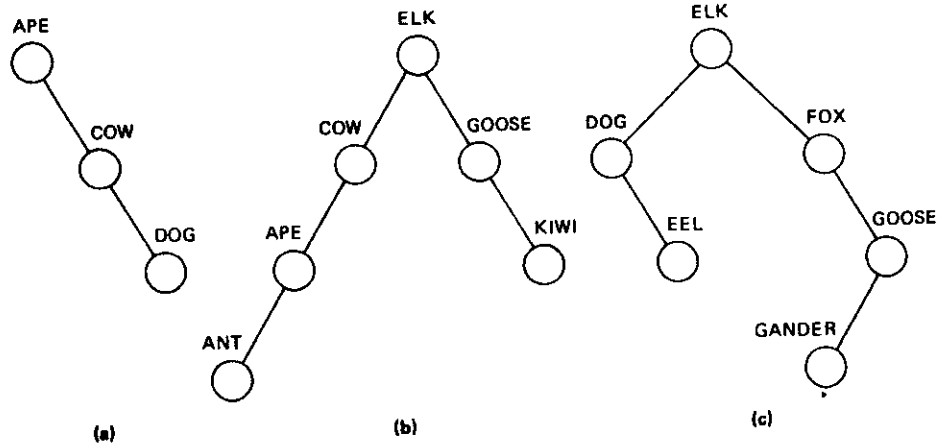
ข้อควรจำ ความสูงของต้นไม้หมายถึง หนึ่งบวกกับเลขระดับสูงสุดของโหนด

(Recall that the height of a tree is one plus the number of the highest level on which it has nodes.)

ต้นไม้ค้นแบบทวิภาคที่แสดงในรูป 8-25 ทั้งหมดเป็นต้นไม้ความสูงได้ดุล โปรดสังเกตว่า ถึงแม้ว่าจะเป็นต้นไม้ความสูงได้ดุลต้นไม้ (c) ไม่ใช่ต้นไม้ได้ดุลแบบบริบูรณ์ ต้นไม้ค้นแบบทวิภาครูป 8-26 ไม่ใช่ต้นไม้ความสูงได้ดุลข้อจำกัดความสูงผ่าผืนที่โหนด APE ในต้นไม้ (a) และที่โหน COW ในต้นไม้ (b) , และที่โหนด FOX ในต้นไม้ (c)



รูป 8-25 ต้นไม้ความสูงได้ดุล



รูป 8-26 ต้นไม้ทวิภาคซึ่งไม่ใช่ต้นไม้ความสูงได้ดุล

ต้นไม้ AVL ให้ความสามารถการค้นโดยตรงที่ดี ถึงแม้ว่ามันจะมีแนวโน้มดูบาง (sparse) มากกว่าที่ทำกับต้นไม้ได้ดุลแบบบริบูรณ์ , การค้นต้องการเฉพาะเวลาที่นานกว่าพอเสมควรความยาวการค้นที่เป็นไปได้กรณีแย่ที่สุดสำหรับต้นไม้ได้ดุลแบบบริบูรณ์ของดหนด n ตัวเท่ากัน $\log_2(n+1)$ ขณะที่ความยาวการค้นที่เป็นไปได้แย่ที่สุดสำหรับต้นไม้ AVL ของโหนด n ตัวเท่ากับ $1.44 \log_2(n+1)$

ความยาวการค้นโดยเฉลี่ยคาดคะเน สำหรับต้นไม้ได้ดุลแบบบริบูรณ์ของ n โหนด เท่ากับ $\log_2(n+1)-2$ และความยาวการค้นโดยเฉลี่ยคาดคะเนสำหรับต้นไม้ AVL ของ n โหนดเท่ากับ $\log_2(n+1)+\alpha$ เมื่อ α เป็นค่าคงที่ ซึ่งขึ้นอยู่กับคุณสมบัติของการแจกแจงชื่อ (consult Knuth [1973, pp.453-454] สำหรับรายละเอียดของการวิเคราะห์นี้) ด้วยเหตุนี้ต้นไม้ AVL จึงเกือบดีเท่ากับต้นไม้ได้ดุลแบบบริบูรณ์ ข้อจำกัดความสูงป้องกันต้นไม้ AVL จากการกลายเป็นญาติที่ไกลมากของต้นไม้ได้ดุลแบบบริบูรณ์

ถ้าการใส่หรือการลบทำให้ต้นไม้ AVL กลายเป็นไม่ได้ดุล ดังนั้นต้นไม้จึงต้องถูกทำโครงสร้างใหม่เพื่อกลับคืนสถานะได้ดุล กระบวนการทำโครงสร้างใหม่นี้ปกติเรียกว่า **การหมุน** (rotation) ซึ่งไม่ได้นำมาอภิปรายในที่นี้ ผู้อ่านที่สนใจให้ดูในเทคนิคการทำต้นไม้ให้ได้ดุลในหนังสือของ Knuth (1973, หน้า 453-463)

8.16 ต้นไม้ขอบเขตได้ดุลหรือต้นไม้ BB (Bounded-Balanced (BB) Tree)

ต้นไม้ได้ดุลซึ่งเกือบจะเป็นต้นไม้แบบบริบูรณ์อีกชนิดหนึ่งได้แก่ต้นไม้ขอบเขตได้ดุล ต้นไม้เหล่านี้จะเรียกว่าต้นไม้ BB หรือ **ต้นไม้น้ำหนักได้ดุล** (weight - balanced trees) ด้วยเช่นกันความคิดพื้นฐานของต้นไม้ขอบเขตได้ดุลคือการใส่ข้อจำกัดว่า ต้นไม้สามารถหันเหจากการเป็นต้นไม้ได้ดุลแบบบริบูรณ์ได้ไกลแค่ไหน ขีดจำกัดขอบเขตนี้ คือความแตกต่างระหว่างจำนวนโหนดในต้นไม้ส่วนย่อยซ้ายกับจำนวนโหนดในต้นไม้ส่วนย่อยขวา ถึงแม้ว่าพื้นฐานบนข้อจำกัดขนาดของต้นไม้ส่วนย่อย ไม่ใช่ความสูงของต้นไม้ส่วนย่อย ต้นไม้ขอบเขตได้ดุลคล้ายกับคุณสมบัติของต้นไม้ความสูงได้ดุล จุดประสงค์ทั้งคู่คือให้เป็นการประนีประนอมระหว่างความยาวการค้นสั้น และความต้องการทำให้ได้ดุล

ต้นไม้ขอบเขตได้ดุลจะมีพารามิเตอร์ β ซึ่งควบคุมการประนีประนอมนี้ ในต้นไม้ขอบเขตได้ดุล ของดุล β , เมื่อ $0 < \beta \frac{1}{2}$ สำหรับแต่ละโหนดของต้นไม้

เศษส่วน (fraction) ของโหนดในต้นไม้ส่วนย่อยซ้ายของโหนดจะอยู่ระหว่าง β และ $1-\beta$ กล่าวอย่างเป็นทางการมากขึ้นคือ ถ้า SIZE (N) คือจำนวนโหนดในต้นไม้มีรากที่โหนด N ดังนั้นข้อจำกัดคือ

$$\beta \leq \frac{\text{SIZE(LEFT(N))} + 1}{\text{SIZE(N)}+1} \leq 1 - \beta$$

ถ้า $\beta = \frac{1}{2}$ เพราะฉะนั้นต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาจะมีจำนวนโหนดเท่ากัน

นั่นคือต้นไม้จะเป็นต้นไม้ได้ดุลแบบบริบูรณ์ ถ้า $\beta = \frac{1}{4}$ ดังนั้นต้นไม้ส่วนย่อยชุดหนึ่งจะมี

จำนวนโหนดมากเป็นสามเท่าของจำนวนโหนดในต้นไม้ส่วนย่อยอีกชุดหนึ่งขณะที่ค่า β $\frac{1}{2}$

ขีดจำกัดของขอบเขตจะแน่นขึ้น นักออกแบบต้นไม้เลือกค่า β ที่เหมาะสมสำหรับงานประยุกต์ ความยาวการค้นโดยเฉลี่ยคาดคะเนสำหรับต้นไม้ขอบเขตแบบได้ดุลคือ $O(\log_2 n)$, เมื่อ n คือจำนวนโหนดในต้นไม้ ถ้าการใส่หรือการลบเป็นเหตุให้ต้นไม้ชนิดนี้กลายเป็นต้นไม้ไม่ได้ดุลสัมพันธ์กับขอบเขต β ของมัน ดังนั้น การหมุนชนิดเดียวกันซึ่งประยุกต์ใช้กับต้นไม้ชนิดความสูงได้ดุล จะนำมาประยุกต์ใช้เพื่อทำให้ต้นไม้กลับคืนสถานะได้ดุล

งานที่ต้องการเพื่อรักษาต้นไม้ AVL หรือต้นไม้ BB ตรงกัน(conformance) กับความสูงของมันหรือสัดส่วนขนาด (size-ratio) คือพิจารณาน้อยกว่างานที่ต้องการเพื่อรักษาต้นไม้ให้ได้ดุลแบบบริบูรณ์ ถ้าการเปลี่ยนรูปต้องทำเพื่อรักษาความได้ดุลแบบบริบูรณ์ของต้นไม้หลังจากการใส่หรือการลบซึ่งการเปลี่ยนรูปนั้นต้องเคลื่อนย้าย O(n) โหนดเมื่อ n คือจำนวนโหนดในต้นไม้ ในทางตรงกันข้าม ถ้าการเปลี่ยนรูปจำเป็นเพื่อบำรุงรักษาความสูงหรืออัตราส่วนขนาดให้ตรงกันในต้นไม้ AVL หรือต้นไม้ BB หลังจาการใส่หรือการลบต้องมีการเคลื่อนย้าย $O(\log_2 n)$ โหนดการลดการทำงาน หมายถึงการประสบผลสำเร็จใดยการเคลื่อนย้ายเฉพาะที่ไปยังโหนดตามหนึ่งทางเดินจากรากไปยังจุดแตกใบในต้นไม้ AVL หรือต้นไม้ BB ขณะที่การเคลื่อนย้ายในต้นไม้ได้ดุลแบบบริบูรณ์มีผลกระทบกับต้นไม้ทั้งหมด

บทสรุป (Summary)

บทนี้แนะนำโครงสร้างข้อมูลกราฟ (graph)ชนิดพิเศษ ซึ่งเรียกว่าต้นไม้ทั่วไป (general tree) , ไม่มีวง (an cyclic), ไม่ขาดตอน (connected), กราฟอย่างง่าย(simple graph), แนะนำคำศัพท์ที่เกี่ยวกับต้นไม้ จากนั้นกรณีพิเศษของต้นไม้ทั่วไปซึ่งเรียกว่าต้นไม้แบบทวิภาคกลายเป็นจุดเน้นความสนใจ ในต้นไม้แบบทวิภาคโหนดแต่ละตัวมีต้นไม้ส่วนย่อยได้มากที่สุดสองชุด ซึ่งแยกเป็นต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวา เพราะว่าการจัดการต้นไม้แบบทวิภาคง่ายกว่าการจัดการต้นไม้ทั่วไป จึงเป็นประโยชน์ที่จะแทนต้นไม้ทั่วไป (หรือป่าของต้นไม้ทั่วไป) ด้วยรูปแบบต้นไม้ทวิภาค มีการนำเสนออัลกอริทึมการแปลงผันนี้

ต้นไม้แบบทวิภาคชนิดที่สำคัญได้แก่ ต้นไม้เกือบจะบริบูรณ์ (almost complete) การทำโครงสร้างกลุ่มของโหนดให้เป็นต้นไม้ทวิภาคเกือบจะบริบูรณ์ ทำให้ความยาวทาง

เดินสูงสุดในต้นไม้มีค่าต่ำสุด ด้วยเหตุนี้ทางเดินการค้นยาวที่สุดผ่านต้นไม้ไปยังโหนดเฉพาะจึงถูกทำให้ต่ำสุด

บางครั้งแทนที่จะค้นหาโหนดเฉพาะมันเป็นความจำเป็นเพื่อแวะผ่านต้นไม้แบบทวิภาคนั้นคือ เยี่ยมโหนดของต้นไม้ทุกตัวอัลกอริทึมจำนวนมากถูกนำเสนอสำหรับการแวะผ่านต้นไม้ทวิภาคได้แก่ preorder, inorder, และ postorder อัลกอริทึมเหล่านี้แตกต่างกันในอับดับซึ่งราก , ต้นไม้ส่วนย่อยซ้ายและต้นไม้ส่วนย่อยขวาถูกแวะผ่าน การเรียงอันดับแต่ละชนิดเหมาะสมสำหรับงานประยุกต์ชนิดที่แตกต่างกัน เพื่ออำนวยความสะดวกการแวะผ่านต้นไม้อาจเป็น threaded เป็นไปตามชนิดของอัลกอริทึมการแวะผ่าน จากนั้นโหนดแต่ละตัวจะชี้ไปยังโหนดหลังของมัน (its successor) ในทางเดินการแวะผ่าน, Threads ปกติมีสองทิศทางซึ่งทำให้การใสโหนดและการลบโหนดทำได้ง่ายขึ้น การแวะผ่านของต้นไม้ทั่วไปทำให้เป็นผลสำเร็จได้โดยขั้นแรกแปลงผันต้นไม้นั้นให้เป็นรูปแบบทวิภาคของมัน

ต้นไม้ค้นแบบประยุกต์ใช้กับการจัดการกลุ่มระเบียนข้อมูลขนาดใหญ่ที่มีคีย์ซึ่งระบุไว้ เมื่อเข้าถึงโดยตรงกับระเบียนเฉพาะและเข้าถึงแบบลำดับกับเขตของระเบียนทั้งหมดและต้องการทั้งคู่ อัลกอริทึมถูกนำเสนอเพื่อให้กระทำปฏิบัติการพื้นฐาน บนต้นไม้ค้นแบบทวิภาคได้แก่การค้นโดยตรง, การค้นแบบลำดับ, การใส่โหนดและการลบโหนด

ความเข้าใจธรรมชาติของต้นไม้ค้นแบบทวิภาคทำให้เห็นชัดเจนว่าโครงสร้างบางอย่างให้ความสามารถการค้นที่ดีกว่าทำกับโครงสร้างอื่น ๆ มี heuristic ที่มีเหตุผลสองอย่างสำหรับการสร้างต้นไม้ค้นแบบทวิภาคถูกนำมาอภิปราย
ข้อแรกคือ –รักษาต้นไม้ให้ได้ดุล– ให้ความยาวการค้นเหมาะที่สุดใกล้มากที่สุด โดยไม่มีงานของการสร้างจริงบนต้นไม้เหมาะที่สุด

ความยากของการรักษาต้นไม้ค้นแบบทวิภาคให้ได้ดุลอย่างบริบูรณ์ เมื่อมีการใส่โหนดและการลบโหนดได้ถูกอภิปราย และแนะนำต้นไม้ค้นแบบทวิภาคเกือบจะได้ดุลสองชนิดซึ่งให้ความสามารถการค้นที่ดีแต่ต้องการงานบำรุงรักษาน้อยกว่าที่ทำกับต้นไม้ได้ดุลอย่างบริบูรณ์ ต้นไม้ AVL (เรียกว่าต้นไม้ความสูงได้ดุล)มีข้อจำกัดว่า ความแตกต่างของความสูงระหว่างต้นไม้ส่วนย่อยซ้ายกับต้นไม้ส่วนย่อยขวาของแต่ละโหนดมากที่สุดคือหนึ่ง

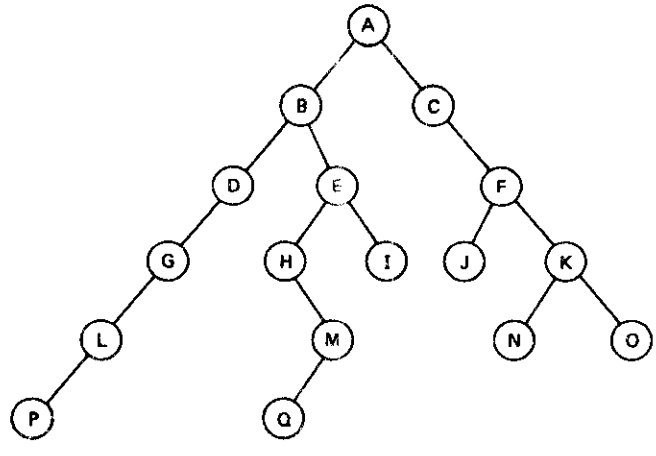
ต้นไม้ขอบเขตได้ดุล (เรียกว่า ต้นไม้ BB หรือ ต้นไม้น้ำหนักได้ดุล มีข้อจำกัดความแตกต่างระหว่างขนาด (นั่นคือ จำนวนโหนด) ของต้นไม้ส่วนย่อยซ้ายกับต้นไม้ส่วนขวาของโหนด นักออกแบบต้นไม้ ในที่นี้เป็นผู้กำหนด พารามิเตอร์ที่ใช้ในการสร้างข้อจำกัด

การใส่และการลบโหนดในต้นไม้ค้นทวิภาคแบบได้ดุลอาจทำให้ต้นไม้นั้นไม่ได้ดุล ; ความสูงหรือ ข้อจำกัดขนาดอาจถูกฝ่าฝืน เพื่อให้ต้นไม้กลับคืนสู่เงื่อนไขได้ดุลการหมุนอาจต้องประยุกต์กับต้นไม้

ความยาวการค้นโดยเฉลี่ยคาดคะเนสำหรับต้นไม้ค้นทวิภาคแบบได้ดุลคือ O $(\log_2 n)$

แบบฝึกหัด

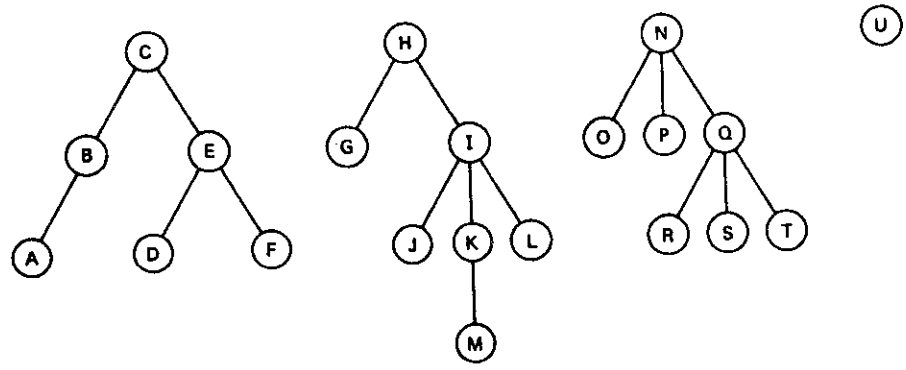
1. จงวาดรูปต้นไม้แบบทวิภาคสองชุดที่คล้ายกัน (Draw two binary trees that are similar.)
2. จงวาดรูปต้นไม้แบบทวิภาคสองชุดที่เหมือนกัน (Draw two binary trees that are equivalent.)
3. ต้นไม้แบบทวิภาคสองชุดซึ่งเหมือนกัน แต่ไม่คล้ายกันได้หรือไม่ ถ้าได้ให้ยกตัวอย่าง ถ้าไม่ได้ ให้บอกเหตุผลว่าทำไมจึงไม่ได้
4. วัฏจักร (cycle)มีอยู่ในต้นไม้ได้หรือไม่ ถ้าได้ให้ยกตัวอย่าง ถ้าไม่ได้ให้บอกเหตุผล
5. ต้นไม้ทวิภาคแบบบริบูรณ์ซึ่งมีความสูงเท่ากับ K ที่ระดับ I จะมีจำนวนโหนดกี่ตัว เมื่อ I < K
6. ต้นไม้ทวิภาคแบบเกือบจะบริบูรณ์ซึ่งมีความสูงเท่ากับ K ที่ระดับ I จะมีจำนวนโหนดกี่ตัวเมื่อ I < K
7. a) จงวาดรูปต้นไม้ทวิภาคแบบบริบูรณ์ที่มีโหนด 10ตัว
   b) จงวาดรูปต้นไม้ทวิภาคเกือบจะบริบูรณ์ที่มีโหนด 10 ตัว
   c) จงคำนวณหาความยาวทางเดินสูงสุดที่ต่ำสุด (the minimum maximum path length) สำหรับต้นไม้ทวิภาคที่มีโหนด 10 ตัว
   d) จงคำนวณหาความยาวทางเดินสูงสุดที่มากที่สุด (the maximum maximum path length) ของต้นไม้ที่มีโหนด 10 ตัว
8. ทำไมการแทนที่ต้นไม้แบบทวิภาคในโปรแกรมจึงง่ายกว่า การแทนที่ต้นไม้ทั่วไป
9. พิจารณาต้นไม้แบบทวิภาคข้างล่างนี้



จงบอกรายชื่อโหนดของต้นไม้แบบ (a) preorder,(b) postorder, (c) inorder จงวาดรูปต้นไม้ threaded แบบ (d) preorder, (e) postoreder, (f) inorder, ให้แทน threads ด้วยเส้นประ

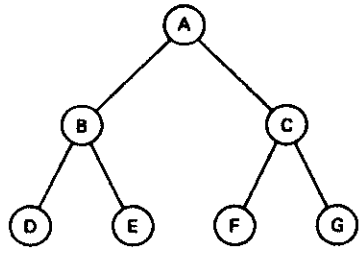
10. ข้อความต่อไปนี้เป็นจริงหรือเป็นเท็จ "จุดแตกใบของต้นไม้แบบทวิภาคเกิดขึ้นในตำแหน่งสัมพัทธ์ที่เหมือนกันทั้งใน preorder, postorder และ inorder" (The leaves of a binary tree occur in same relative position in preorder, postorder, and inorder.)

11. จงวาดรูปต้นไม้แบบทวิภาคซึ่งสมนัยกับป่า ในรูป P8-11



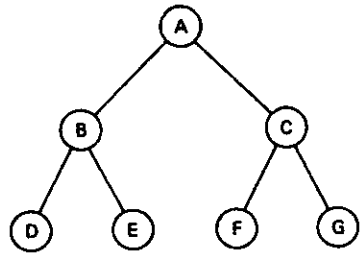
รูป P8-11

12. จงแปลงผันต้นไม้ทั่วไปข้างล่างนี้ให้เป็นต้นไม้แบบทวิภาค



รูป P8-12

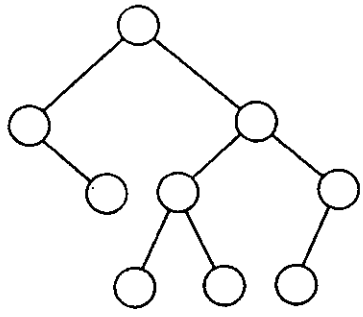
13. ต้นไม้แบบทวิภาคข้างล่างนี้เป็นผลลัพธ์จากการแปลงผันของป่าแห่งหนึ่งไปเป็นรูปแบบทวิภาค จงวาดรูปป่า



รูป P8-13

14. จงแทนนิพจน์ ((A+B)*C)/(D-E) ให้เป็นต้นไม้แบบทวิภาคจากนั้นให้แวะผ่านต้นไม้แบบก่อนลำดับ, หลังลำดับ, และตามลำดับ ทำไมการแวะผ่านแบบหลังลำดับจึงถูกเลือกสำหรับการประเมินผลนิพจน์ นั่นคือทำไมจึงสามารถเห็นได้การแวะผ่านแบบหลังลำดับจึงทำให้การคำนวณง่ายกว่าการเรียงอันดับอื่น ๆ

15. จงแสดงการแทนที่ต้นไม้แบบทวิภาคสำหรับสายอักขระ ICDFEHABC เกิดจาก (a) การใช้ preorder เมื่อต้นไม้มีโครงสร้างดังนี้



รูป P8-15

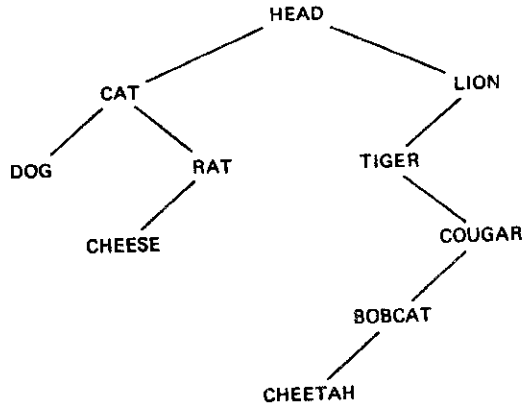
(b) การใช้ inorder

(c) การใช้ postorder

16. กำหนดกลุ่มของคีย์ $K_1, K_2, \ldots, K_n$ จงบอกลำดับซึ่งคีย์เหล่านี้จะใส่ให้เป็นต้นไม้ค้นทวิภาคแบบได้ดุล (Given a collection of keys $K_1, K_2, \ldots, K_n$, in what sequence should those keys be inserted to form a balanced binary search tree ?)

17. คอมไพเลอร์ทั้งหมดที่มีให้ใช้ที่ในระบบคอมพิวเตอร์นั้น ลำดับการเรียง (collating sequence) เหมือนกันหรือไม่ ถ้าไม่ใช่แตกต่างกันอย่างไร

18. จงแสดง inorder threads ในต้นไม้ข้างล่างนี้



รูป P8-18

19. ต้นไม้ค้นแบบทวิภาคควรจะมี threads ชนิดใดเพื่อให้สะดวกในการเยี่ยมของโหนดในลำดับโดยชื่อของมัน

20. จงสร้างต้นไม้ค้นแบบทวิภาคเพิ่มอีกสองรูปสำหรับกลุ่มคีย์ซึ่งมีโครงสร้างในรูป 8-17 โดยใช้ความน่าจะเป็นของการเข้าถึงดังนี้

APE	.2
BEE	.4
COW	.3
DOG	.1

จงคำนวณความยาวการค้นคาดคะเนของต้นไม้นี้

21. จงยกตัวอย่างสถานที่ซึ่งการแวะผ่านแบบหลังลำดับเป็นวิธีที่เหมาะสมที่สุดเพื่อเยี่ยมต้นไม้และให้ทำเหมือนกันสำหรับการแวะผ่านแบบตามลำดับ และการแวะผ่านแบบก่อนลำดับ

22. จงกำหนดโครงสร้างโหนดของต้นไม้แบบทวิภาคซึ่งมี threads ทั้ง inorder และ postorder ทั้งคู่

23. ต้นไม้แบบทวิภาคจริงอาจมี threads สำหรับการแวะผ่านมากกว่าหนึ่งชนิดได้หรือไม่ ทำไมมีได้ หรือทำไมมีไม่ได้

24. จงพิจารณาต้นไม้แบบทวิภาค T ซึ่งมีโหนด n ตัวเมื่อ $n \geq 0$ถ้า T ไม่ใช่ต้นไม้ว่าง ระดับสูงสุดซึ่งโหนดใด ๆ ของ T สามารถมาถึงคือ n ถ้า T ไม่ใช่ต้นไม้ว่างดังนั้นจะมีอย่างน้อยที่สุดหนึ่งโหนดซึ่งประสบผลสำเร็จที่ระดับสูงสุดสำหรับต้นไม้นั้น จงคำนวณหาค่าต่ำสุดซึ่งระดับสูงสุดนี้สามารถเป็นได้ ตัวอย่างเช่น สำหรับ n = 1 คำตอบคือ 1 ในขณะ

ที่ n = 2, คำตอบคือ 2 และสำหรับ n = 3 คำตอบคือ 2 จงแสดงคำตอบเป็นสูตรในเทอมของ n, $\log_2$ (ล็อกการิทึมฐาน 2) และฟังก์ชัน ceiling $\lceil\ \rceil$ และให้พิสูจน์ว่าผลลัพธ์เป็นจริง

25. การเรียงอันดับพื้นฐานสามชนิดสำหรับการแวะผ่านต้นไม้แบบทวิภาคได้แสดงให้เห็นแล้ว ยังมีอีกหนึ่งทางเลือกซึ่งเป็นดังนี้

(a) เยี่ยม root

(b) แวะผ่านต้นไม้ส่วนย่อยขวา

(c) แวะผ่านต้นไม้ส่วนย่อยซ้าย

การใช้กฎการเรียกซ้ำเหมือนกันบนต้นไม้ส่วนย่อยไม่ว่างทั้งหมด การเรียงอันดับใหม่นี้มีความสัมพันธ์อย่างง่ายอย่างใดกับการเรียงอันดับสามชนิด ที่ได้อภิปรายไปแล้วหรือไม่ threads ชนิดใดซึ่งควรจะถูกนิยามเพื่อทำให้การแวะผ่านในการเรียงอันดับนี้ง่ายขึ้น

26. จงอธิบายว่าเกิดอะไรขึ้น เมื่อการแปลงผันต้นไม้ทั่วไปให้เป็นต้นไม้แบบทวิภาค นำไปประยุกต์กับต้นไม้แบบทวิภาค

27. จงเขียนโปรซีเดอร์เพื่อสร้างต้นไม้ทั่วไปเดิมขึ้นใหม่จากต้นไม้แบบทวิภาคซึ่งเป็นผลลัพธ์จากอัลกอริทึมการแปลงผันที่นำเสนอในบทนี้

28. จงเขียนโปรแกรมรับต้นไม้ทั่วไปเป็นอินพุตและให้ต้นไม้ทวิภาคที่สมนัยกันเป็นเอาต์พุต

29. จงเขียนโปรซีเดอร์วนซ้ำ (iterative procedures) เพื่อกระทำการแวะผ่านแบบตามลำดับบนต้นไม้ทวิภาคซึ่ง threaded แบบตามลำดับ กระทำการแวะผ่านแบบก่อนลำดับบนต้นไม้ทวิภาคซึ่ง threaded แบบก่อนลำดับ และกระทำการแวะผ่านแบบหลังลำดับบนต้นไม้ทวิภาคซึ่ง threaded แบบหลังลำดับ จากนั้นเขียนโปรแกรมเพื่อทำให้เกิดผลในทางปฏิบัติของโปรซีเดอร์ทั้งสามชนิด

30. จงเขียนโปรซีเดอร์เรียกซ้ำ (recursive procedure) เพื่อทำให้เกิดผลในทางปฏิบัติของการแวะผ่านแบบตามลำดับ ก่อนลำดับและหลังลำดับจากนั้นจงเขียนโปรแกรมเพื่อทำให้เกิดผลในทางปฏิบัติของโปรซีเดอร์

31. จงเขียนโปรซีเดอร์แบบไม่ใช่การเรียกซ้ำ (ได้แก่ iterative) นั่นคือ การวนซ้ำเพื่อทำให้เกิดผลในทางปฏิบัติของการแวะผ่านแบบตามลำดับ , ก่อนลำดับ , และแบบหลังลำดับสำหรับต้นไม้แบบ unthreaded จากนั้นจงเขียนโปรแกรมเพื่อทำให้เกิดผลในทางปฏิบัติของโปรซีเดอร์

32. อะไรคือ trade-offs ซึ่งจะต้องพิจารณาในทางการตัดสินใจว่าการแวะผ่านต้นไม้ควรจะทำให้เกิดผลในทางปฏิบัติโดยรูทีนการเรียกซ้ำหรือรูทีนการวนซ้ำ

33. จงเขียนโปรซีเดอร์ใส่โหนดตัวใหม่ในต้นไม้แบบทวิภาคซึ่ง threaded แบบตามลำดับ, ก่อนลำดับ, หลังลำดับ
34. จงเขียนโปรซีเดอร์เพื่อลบโหนดหนึ่งตัวออกจากต้นไม้แบบทวิภาค จากต้นไม้ทวิภาคแบบ threaded
35. จงบอกความแตกต่างระหว่างต้นไม้ได้ดุลและต้นไม้ความสูงได้ดุล
36. จงเขียนโปรซีเดอร์แบบไม่ใช่การเรียกซ้ำ(นั่นคือการวนซ้ำ) เพื่อค้นหาต้นไม้ค้นแบบทวิภาคและใส่โหนดตัวใหม่ในต้นไม้ค้นแบบทวิภาค
37. จงดัดแปร(modify) โปรซีเดอร์ซึ่งค้นหาต้นไม้ค้นแบบทวิภาคและใส่โหนดตัวใหม่ในต้นไม้ค้นแบบทวิภาคเพื่อให้มันรู้จักโหนดซึ่งมีการทำเครื่องหมายว่าถูกลบแล้วแต่ยังไม่ได้ลบออกจากต้นไม้
38. จงเขียนโปรซีเดอร์เพื่อลบโหนดที่กำหนดออกจากต้นไม้ค้นแบบทวิภาคทั้งนี้ ให้แน่ใจว่าครอบคลุมสถานะการณ์ซึ่งโหนดที่จะถูกลบ ไม่มีต้นไม้ส่วนย่อยแบบ non-null, มี 1 ชุดหรือมี 2 ชุด
39. จงอ่านเทคนิคเกี่ยวกับการรักษาต้นไม้ให้ได้ดุล จากนั้นจงเขียนโปรแกรมเพื่อทำให้เกิดผลในทางปฏิบัติกับเทคนิคนั้นหนึ่งวิธี (Read about techniques for keeping a tree in balance. Write a program to implement one of the techniques.)