คำถามบทที่ 6

1) Non-Linear List คือ

2) Basic Tree Concepts คือ

3) Basic Tree Concepts มีศัพท์สำคัญคือ

4) ทรีประกอบด้วยอะไรบ้าง

5) ทรีไม่เป็นทรีว่าง มีลักษณะอย่างไร

6) โหนดไม่เป็นรูทโหนดมีลักษณะอย่างไร

7) โครงสร้างทรีเป็นลำดับชั้นมีลักษณะอย่างไร

8) Subtrees คือ

9) ความสูงของทรี คือ

10) ไบนารีทรี คือ

11) Null Tree คือ

12) คุณสมบัติของไบนารีทรี คือ

13) Maximum Height คือ

14) Minimum Height คือ

15) Minimum Nodes คือ

16) Maximum Nodes คือ

17) Balance คือ

18) ไบนารีทรีแบบสมบูรณ์และเกือบสมบูรณ์ แตกต่างกันอย่างไร

19) การแทนไบนารีทรีในหน่วยความจำ คือ

20) การแทนไบนารีทรีด้วยอาร์เรย์ คือ

21) การแทนไบนารีทรีด้วยลิงก์ลิสต์ คือ

22) การท่องเข้าไปในไบนารีทรีมีกี่แบบ

23) วิธีการท่องแบบแนวลึก มีทั้งหมดกี่มาตรฐาน

24) NLR คือ

25) LNR คือ

26) LRN คือ

27) วิธีการท่องแบบแนวกว้างคือ

28) การท่องเข้าไปในไบนารีทรีแบบแนวลึก คือ

29) เอ็กซ์เพรสชันทรี คือ

30) การท่องเข้าไปในเอ็กซ์เพรสชันทรีแบบ Infix คือ

31) การท่องเข้าไปในเอ็กซ์เพรสชันทรีแบบ Postfix คือ

NO.....

Date...../...../.....

32) General Trees คือ

33) การเรียกโหนดแบบ FIFO ในเจเนอริคทรี คือ

34) การเรียกโหนดแบบ LIFO ในเจเนอริคทรี คือ

35) การเรียกโหนดแบบ key-Sequenced คือ

36) การลบโหนดในเจเนอริคทรี คือ

37) การเปลี่ยนแปลงเจเนอริคมาเป็นไบนารีทรี คือ

Date...../...../.....

คำตอบบทที่ 6

1) ลิสต์แบบไม่ใช่เชิงเส้น (Non-Linear List) คือ ลิสต์ที่สมาชิกแต่ละตัวสามารถมีสมาชิกตัวถัดไปได้มากกว่าหนึ่งตัว นอกจากนี้ ลิสต์แบบไม่ใช่เชิงเส้นยังแบ่งออกเป็น 2 ประเภทใหญ่ๆ ได้แก่ ทรี และกราฟ โดยโครงสร้างข้อมูลแบบทรี สมาชิกแต่ละตัว ยกเว้นรูทโหนด จะมีสมาชิกตัวก่อนหน้าเพียงตัวเดียวหนึ่งตัวในขณะที่โครงสร้างข้อมูลแบบกราฟ สมาชิกแต่ละตัวสามารถมีตัวก่อนหน้าได้มากกว่าหนึ่งตัว

2) โครงสร้างข้อมูลแบบทรีเปรียบเสมือนกับต้นไม้ที่มีอยู่ในธรรมชาติ โดยมีราก (Root) และมีลำดับที่แน่นอนจากลำต้น สาขา แต่ละกิ่ง (Branch) จะมีการแตกใบ (Leaf) ไปด้วย ซึ่งเป็นไปตามความสัมพันธ์ในลักษณะแบบลำดับชั้น และสิ่งที่มีโครงสร้างแบบลำดับชั้นนี้เอง จึงมักนำไปใช้เพื่อนำเสนอเป็นแผนผังองค์กร (Organization chart) โดยเรียกไดเรกทอรี (Directory) หรือแฟ้มระหว่างสมาชิกรอบตัวของทรีเอง

3) ความสำคัญก็คือ โดยเรียงข้อมูลแบบทรีสามารถสื่อแนวคิดในรูปแบบเป็นวิดีโอรีซัฟฟาส์ ดังนั้น จึงมักนำการแบ่งการของทรีไปประยุกต์ใช้แต่ละโหนดลึกลงไปแต่ละความสัมพันธ์กันในลักษณะเหมือนครอบครัวเดียวกันโดยมีโหนดพ่อซึ่งอยู่ระดับเหนือกว่ามีเส้นเชื่อมโยงจากโหนดพ่อไปยังโหนดลูกคือฟรุยเดียวกัน ดังนั้น ในการศึกษาเกี่ยวกับทรีจึงจำเป็นต้องทำใจศึกษาพื้นฐานที่เกี่ยวข้องเสียก่อน

4) ทรีประกอบไปด้วยโหนดต่างๆ ที่ใช้งานอยู่ข้อมูล โดยมีกิ่งใช้เป็นเส้นเชื่อมโยงจากโหนดที่เรียกว่า บรรพชน (Branch) จำนวนของบรรพชนที่มีความสัมพันธ์กับโหนดจะเรียกว่า ดีกรี (Degree) สำหรับบรรพชนที่ขึ้นตรงต่อโหนดใดๆ จะเรียกว่า อินดีกรี (Indegree) สำหรับบรรพชนที่ออกจากโหนดจะเรียกว่า เอาต์ดีกรี (Outdegree) ซึ่งผลรวมของบรรพชนอินดีกรีกับเอาต์ดีกรีก็คือ ดีกรีของโหนดนั่นเอง

5) กรีนั้นไม่ใช่เป็นทรีว่าง โหนดแรกจะเรียกว่า รูต Root ซึ่งอินดีกรีของรูตโหนดจะมีค่าเป็นศูนย์เสมอเป็นที่แน่นอนคือ ทุกๆ โหนดในทรีจะมีอินดีกรีเป็นหนึ่ง ยกเว้นรูตโหนด นั่นหมายความว่า ทุกๆ โหนดยกเว้นรูตโหนด จะมีเพียงหนึ่ง Predecessor ในขณะที่ Successor อาจเป็น 0 หรือ 1 หรือมากกว่า 1 ก็ได้

6) สำหรับโหนดที่ไม่มีบรรพชนเชื่อมโยงไปยังโหนดลำดับถัดไปจะเรียกว่า โหนดใบ (Leaf) โดยโหนดใบจะเป็นโหนดที่ไม่มีลำดับต่อหลัง ในขณะที่ โหนดพ่อ (Parent) จะมีโหนดตามหลัง หรือกล่าวอีกนัยหนึ่งว่า โหนดที่มีลำดับก่อนหน้าก็คือโหนดลูก (child) โดยโหนดลูกตั้งแต่สองโหนดขึ้นไปถือมาจากพ่อเดียวกันจะเรียกว่า โหนดพี่น้อง (Siblings) สำหรับความสัมพันธ์อื่นๆ ภายในครอบครัวของทรี ไม่ว่าจะเป็น ลุง ป้า น้า อา หรือหลาน เหลน โหลนนั้นไม่ถือเป็นสิ่งที่น่าใส่ใจ ขอเพียงพิจารณาแต่ความสัมพันธ์หลักๆ ก็พอแล้ว เช่น บรรพบุรุษ (Ancestor) และลูกหลาน (Descendent)

Date...../...../.....

7) เมื่อมองจากทรีเป็นโครงสร้างที่มีความสัมพันธ์กันเป็นลำดับชั้น ดังนั้น โหนดต่างๆ ภายในทรีจึงอยู่ในระดับที่แตกต่างกัน เริ่มจากรูตโหนดซึ่งเป็นระดับบนสุดคือระดับที่ 0 ส่วนลูกๆ ของรูตโหนดก็คือระดับที่ 1 และลูกๆ ของโหนดระดับที่ 1 ก็จะอยู่ระดับที่ 2 ซึ่งอยู่เพิ่มระดับไปเรื่อยๆ แต่มีลูกหลานเพิ่มขึ้น ดังนั้น การคำนวณระดับความลึกหรือความสูงของทรี จะคำนวณได้จากการนำค่าระดับสูงสุด Leaf Node ของทรีนั้นมาบวกด้วย 1

8) ทรีนั้นสามารถแบ่งออกเป็นทรีย่อย (Subtree) ซึ่งซับทรีแต่ละต้นเป็นโครงสร้างที่เชื่อมต่อเข้ากับรูตโหนดสำหรับโหนดแรกของซับทรีก็คือ รูตโหนดของซับทรีนั้นๆ และพ่อใช้เป็นชื่อเรียกของซับทรีนั้นด้วย นอกจากนี้ซับทรีนั้นสามารถแบ่งออกเป็นซับทรีย่อยๆ ได้อีก เช่น BCD คือซับทรี และรวมถึง E และ FGHI ก็คือซับทรี 1 ต้นเดียวกับโหนดรูต B สามารถแบ่งออกเป็นอีก 2 ซับทรี คือ C กับ D ส่วนซับทรี E มีสมาชิกเพียงโหนดเดียว ในขณะที่ซับทรี F ประกอบด้วยซับทรี G, H และ I

9) การแทนค่าทรี (Tree Representation) เราสามารถแทนค่าทรีได้ 3 รูปแบบด้วยกัน คือ

1) การแทนค่าด้วยโครงสร้างทรีแบบทั่วไป (General Tree) เป็นรูปแบบการแทนค่าเหมือนกับของอุตสาหกรรมทั่วไปที่มีการลดหรือเป็นลำดับชั้น ซึ่งก็เป็นรูปแบบของทรีที่เป็นที่ใช้กันอยู่ทั่วไป

2) การแทนค่าด้วยโครงสร้างทรีแบบพ่อนำ (Indented List) รูปแบบการจัดเก็บการเขียนโปรแกรมเชิงโครงสร้างด้วยการใช้พ่อนำเป็นตัวจัดรูปแบบโครงสร้างที่มีการย่อหน้าระดับลึกลงไป

3) การแทนค่าด้วยโครงสร้างทรีแบบวงเล็บ (Parenthetical List) รูปแบบนี้คล้ายกับการแทนค่าข้อมูลด้วยนิพจน์คณิตศาสตร์ โดยใช้เครื่องหมายวงเล็บสำคัญ

10) ไบนารีทรี (Binary Trees) ถือเป็นทรีชนิดหนึ่งที่มีความสำคัญมาก มีคุณสมบัติสำคัญคือ เป็นทรีที่สามารถมีลูกได้ไม่เกินสองโหนด กล่าวคือ ทุกๆ โหนดในไบนารีทรีอาจมีลูกเพียงด้านซ้ายหรือด้านขวา หรืออาจมีลูกทั้งซ้ายและขวา หรืออาจไม่มีลูกเลยก็ได้ หรือกล่าวอีกนัยหนึ่งก็คือ เป็นทรีที่ทุกโหนดจะมีซับทรี ≤ 2 นั่นเอง

11) และเพื่อให้เกิดความเข้าใจในโครงสร้างของไบนารีทรีที่ลึกซึ้งขึ้น ในการศึกษาจากรูปที่ 6.6 ต่อไปนี้ ที่ทรงกลวงกลุ่มของไบนารีทรีในรูปแบบต่างๆ โดยรูปที่ 6.6 (a) เป็นทรีว่างเปล่า (Null Tree) หรือเป็นทรีที่ไม่มีโหนดใดๆ เลยซึ่งในที่นี้คือว่าเป็นแทนของทรีที่เราไม่สนใจ รูปที่ 6.6 ไบนารีทรีในรูปแบบต่างๆ

12) คุณสมบัติของไบนารีทรี (Binary Trees Properties) สำหรับคุณสมบัติของไบนารีทรีจะทำให้ไบนารีทรีมีคุณสมบัติเพิ่มเติมจากทรีทั่วไป และพื้นฐานการนำไปคำนวณเพื่อหาผลลัพธ์ต่างๆ ได้ดังนี้ ความสูงของไบนารีทรี (Height of Binary Trees) และความสมดุลของไบนารีทรี (Balance)

NO.....

Date...../...../.....

13) ความสูงมากที่สุดของทรี (Maximum Height) หมายถึง การจัดเก็บโหนดจำนวน N โหนดในไบนารีทรี ความสูงมากที่สุดของทรีสังเกตว่าสามารถคำนวณได้จากสูตร

$$H_{max} = N$$

ตัวอย่างเช่น มีโหนดจำนวน 3 โหนด ต้องการจัดเก็บไว้ในไบนารีทรี ดังนั้น ความสูงมากที่สุดของทรีนี้ก็คือ H_{max} เท่ากับ 3 นั่นเอง

14) ความสูงน้อยที่สุดของทรี (Minimum Height) หมายถึง การจัดเก็บโหนดจำนวน N โหนดในไบนารีทรี ความสูงน้อยที่สุดของทรีสังเกตว่าสามารถคำนวณได้จากสูตร

$$H_{min} = [\log_2 N] + 1$$

ตัวอย่างเช่น มีโหนดจำนวน 3 โหนด ที่จัดเก็บอยู่ในไบนารีทรี ดังนั้น จากสูตรการคำนวณข้างต้น H_{min} ก็จะมีค่าเท่ากับ 2

15) จำนวนโหนดน้อยที่สุด (Minimum Nodes) เราสามารถคำนวณจำนวนโหนดน้อยที่สุดที่สามารถมีได้ในไบนารีทรี ว่ามีจำนวนเท่าไรได้จากสูตร

$$N_{min} = H$$

ตัวอย่างเช่น ทรีต้นหนึ่งมีความสูงเท่ากับ 3 จงหาจำนวนโหนดน้อยที่สุดที่สามารถมีได้ในไบนารีทรี จากสูตรการคำนวณข้างต้นคือ N_{min} ก็จะมีค่าเท่ากับ 3 นั่นเอง

16) จำนวนโหนดมากที่สุด (Maximum Nodes) สูตรการคำนวณหาจำนวนโหนดมากที่สุดที่มีได้ในไบนารีทรี จะพิจารณาจากความเป็นองค์ประกอบสมบัติของไบนารีทรีก็คือ ทุกๆโหนดสามารถมีลูกได้สูงสุดไม่เกิน 2 โหนด ซึ่งสามารถคำนวณได้จากสูตร

$$N_{max} = 2^H - 1$$

ตัวอย่างเช่น ทรีต้นหนึ่งมีความสูงเท่ากับ 3 จงหาจำนวนโหนดสูงสุดที่สามารถมีได้ในไบนารีทรี จากสูตรการคำนวณข้างต้น N_{max} ก็จะมีค่าเท่ากับ 7

17) ความสมดุล (Balance) ความสมดุลของไบนารีทรีสามารถหาได้จากค่า Balance Factor (เท่ากับ 0) ซึ่งคำนวณค่าได้จากการนำความสูงของซับทรีด้านซ้าย (H_L) มาพิจารณากับความสูงของซับทรีด้านขวา (H_R) ซึ่งเป็นไปตามสูตรนี้ $B = H_L - H_R$ จากสูตรสังเกตว่าการคำนวณความสมดุลของไบนารีทรี

18) ไบนารีทรีที่สมบูรณ์ จะมีจำนวนโหนดสูงสุดที่สามารถมีได้ตามความสูงของทรี ซึ่งเป็นไปตามสูตร N_{max} โดยจะมีโหนดเต็มทั้งซับทรีด้านซ้ายและด้านขวา ส่วน ไบนารีทรีที่เกือบสมบูรณ์ จะเป็นไปตามสูตร H_{min} คือ จำนวนน้อยที่สุดของทรีที่จะเกิดขึ้นได้ ที่มีโหนดเต็มทุกโหนดยกเว้นระดับสุดท้ายที่มีเพียงบางโหนดเท่านั้น

Johnson

19) การแทนไบนารีทรีในหน่วยความจำ (Binary Tree Representations) การแทนโครงสร้างไบนารีทรีลงในหน่วยความจำ สามารถดำเนินการได้จากการนำอาร์เรย์หรือลิงก์ลิสต์มาประยุกต์ใช้ ซึ่งเป็นไปตามรายละเอียดต่อไปนี้ การแทนไบนารีทรีด้วยอาร์เรย์ และการแทนไบนารีทรีด้วยลิงก์ลิสต์

20) การแทนไบนารีทรีด้วยอาร์เรย์ ในที่นี้ขอกล่าวเฉพาะการแทนไบนารีทรีลงในหน่วยความจำอาร์เรย์แบบหนึ่งมิติ โดยกรณีเป็นไบนารีทรีแบบสมบูรณ์ จะมีการลำดับตำแหน่งข้อมูลในทรีแต่ละระดับ การเรียงจากซ้ายไปขวา ไบนารีทรีแบบสมบูรณ์กับการแทนที่ลงในหน่วยความจำอาร์เรย์หนึ่งมิติ

21) การแทนไบนารีทรีด้วยลิงก์ลิสต์ การแทนโครงสร้างไบนารีทรีด้วยหน่วยความจำแบบลิงก์ลิสต์จะช่วยแก้ปัญหาการสูญเสียพื้นที่ว่างเปล่าที่เกิดขึ้นในหน่วยความจำแบบอาร์เรย์ได้ และด้วยลิงก์ลิสต์เป็นหน่วยความจำแบบไดนามิกนั่นเอง จึงนำมาใช้หน่วยความจำได้อย่างคุ้มค่า โดยแต่ละโหนดจะประกอบด้วยข้อมูลอยู่ 3 ฟิลด์ด้วยกันคือ ข้อมูลที่บรรจุอยู่ภายในโหนด และพอยน์เตอร์ที่ชี้ไปยังซับทรีด้านซ้ายกับพอยน์เตอร์ที่ชี้ไปยังซับทรีด้านขวา

22) การท่องเข้าไปในไบนารีทรี (Binary Tree Traversals) การท่องเข้าไปในไบนารีทรีคือการเดินทางเข้าไปในทรี โดยขณะที่ท่องเข้าไป แต่ละโหนดภายในทรีจะถูกประมวลผลได้เพียงครั้งเดียวตามลำดับที่กำหนดไว้ และโดยปกติวิธีการท่องเข้าไปในทรีจะมีอยู่ 2 วิธีด้วยกันคือวิธีการท่องแบบเน้นลึกกับวิธีการท่องแบบเน้นกว้าง

23) วิธีการท่องแบบเน้นลึก (Depth-First Traversals) เป็นวิธีการท่องเข้าไปในทรีด้วยการเดินทางผ่านจากรูทไปยังโหนดลูก ซึ่งมีทั้งด้านซ้ายและขวา โดยมีการใช้คำย่อดังนี้ ■ N แทนรูทโหนด ■ L แทนซับทรีด้านซ้าย ■ R แทนซับทรีด้านขวา สำหรับวิธีการท่องแบบเน้นลึก ในที่นี้จะขอกล่าว 3 วิธีมาตรฐานที่นิยมใช้กันอย่างแพร่หลายด้วยกันได้แก่วิธี Preorder, Inorder และ Postorder

24) แบบพรีออร์เดอร์ (Preorder Traversal : NLR) การท่องเข้าไปในทรีแบบ Preorder จะเป็นไปในรูปแบบ NLR กล่าวคือ จะเริ่มต้นกระทำที่รูทโหนดก่อนเป็นอันดับแรก จากนั้นก็ท่องผ่านซับทรีด้านซ้าย และซับทรีด้านขวา พิจารณารูปที่ 6.13 (a) เป็นการท่องเข้าไปในทรีแบบ Preorder และผลลัพธ์ที่ได้ ในขณะที่ซูโดโค้ดการท่องแบบ Preorder แสดงไว้ในอัลกอริทึม

25) แบบอินออร์เดอร์ (Inorder Traversal : LNR) การท่องเข้าไปในทรีแบบ Inorder จะเป็นไปในรูปแบบ LNR กล่าวคือ จะเริ่มท่องไปยังซับทรีด้านซ้ายก่อน หลังจากนั้นผ่านไปยังรูทโหนด และท้ายสุดคือซับทรีด้านขวา พิจารณารูปที่ 6.13 (b) เป็นการท่องเข้าไปในทรีแบบ Inorder และผลลัพธ์ที่ได้ ในขณะที่ซูโดโค้ดของการท่องแบบ Inorder แสดงไว้ในอัลกอริทึม.

Date...../...../.....

26) ทรีแบบโพสต์ออร์เดอร์ (Postorder Traversal : LRN) วิธีสุดท้ายของการท่องเข้าไปในทรีที่จะกล่าวถึงก็คือ วิธี Postorder เป็นการท่องในรูปแบบ LRN คือจะกระทำที่รูทโหนดเป็นสิ่งสุดท้าย โดยจะเริ่มต้นจากซับทรีด้านซ้ายสุดก่อนเสมอ จากนั้นจึงจะย้อนลงที่รูทโหนด ดังภาพรูปที่ 6.13 (c) เป็นการท่องเข้าไปในทรีแบบ Postorder ซึ่งผลลัพธ์ที่ได้ในขณะที่ดูโครงสร้างการท่องแบบ Postorder ตรงไปในอัลกอริทึม

27) วิธีการท่องแบบทนากว้าง (Breadth - First Traversals) การท่องเข้าไปในทรีแบบทนากว้างเป็นวิธีการท่องเข้าไปในทรีโดยการเริ่มจากโหนดที่อยู่ระดับจากบนลงล่างซึ่งจะเรียงลำดับจากรูทโหนดก่อนแล้วจึงท่องเข้าถึงแต่ละโหนดในแต่ละระดับตามแนวกว้างจากซ้ายไปขวาจนกระทั่งครบทุกระดับ

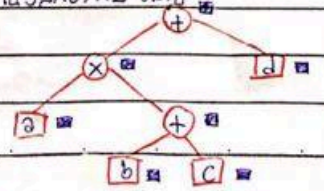
28) การท่องเข้าไปในไบนารีทรีแบบแนวลึกจะนำโครงสร้างข้อมูลแบบสแต็กมาใช้ ทบทวนคำว่าสแต็กช่วยในเรื่องเทคนิคเรียกซ้ำได้เป็นอย่างดี ในขณะเดียวกันการท่องเข้าไปในไบนารีทรีแบบทนากว้างจะใช้โครงสร้างข้อมูลแบบคิว โดยดูโครงสร้างการท่องเข้าไปในไบนารีทรีแบบทนากว้างตรงไปในอัลกอริทึม

29) เอ็กซ์เพรสชั่นทรี (Expression Trees) หนึ่งในการประยุกต์ใช้งานของไบนารีทรีที่น่าสนใจคือ การนำโครงสร้างทรีเพื่อเก็บนิพจน์ทางคณิตศาสตร์อีกเป็นที่มาของคำว่า เอ็กซ์เพรสชั่นทรี ซึ่งเอ็กซ์เพรสชั่นทรีก็คือไบนารีทรีสำหรับการทนนิพจน์ในเอ็กซ์เพรสชั่นทรีจะต้องมีคุณสมบัติดังนี้

1) โหนดใบ (Leaf Node) คือโอเปอแรนด์
2) รูทโหนดและโหนดภายใน คือโอเปอเรเตอร์
3) ซับทรีก็คือนิพจน์ย่อย ซึ่งรูทโหนดของซับทรีต้องเป็นโอเปอเรเตอร์

30) การท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Infix ในการสร้างอัลกอริทึมเพื่อท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Infix เพื่อพิมพ์นิพจน์ออกมานั้นเราจะต้องเพิ่มเครื่องหมายวงเล็บเปิดที่จุดเริ่มต้นของทุกนิพจน์ และปิดท้ายเครื่องหมายวงเล็บปิดที่จุดสิ้นสุดของทุกนิพจน์ ที่เป็นเช่นนี้ก็เพราะว่า นิพจน์แบบ Infix เป็นนิพจน์ที่ใช้เครื่องหมายวงเล็บเพื่อระบุตำแหน่งว่าควรคำนวณที่นิพจน์ใดก่อนนั่นเอง

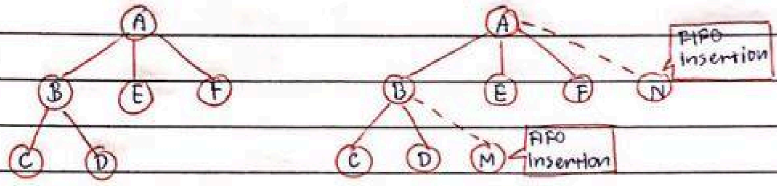
31) การท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Postfix วิธีการท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Postfix ใช้หลักการเดียวกันกับ Postfix ที่ท่องเข้าไปในไบนารีทรี โดยไม่ต้องใช้เครื่องหมายวงเล็บเพื่อช่วยใด ผลลัพธ์จากการท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Postfix เป็นไปตาม a b c + x d + ส่วนดูโครงสร้างการท่องเข้าไปในเอ็กซ์เพรสชั่นทรีแบบ Postfix ตรงไปในอัลกอริทึม ตัวอย่าง



Johnson

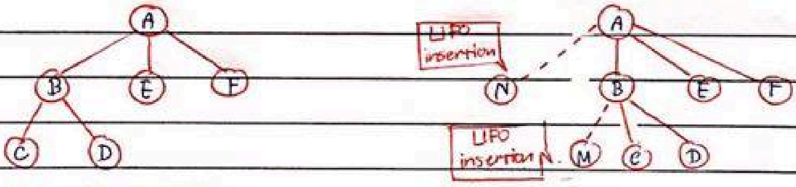
32) เจเนอรัลทรี (General Trees) หรือเจเนอรัลทรีแบบทั่วไป เป็นทรีที่แต่ละโหนดสามารถมีโหนดพี่น้องได้ไม่จำกัด กล่าวคือ แต่ละโหนดสามารถมีลูกได้หลายตัว โหนดหรือมากกว่านั้น ในที่นี้จะกล่าวถึงเฉพาะการเพิ่มกับการแทรกโหนดในเจเนอรัลทรี การลบโหนดในเจเนอรัลทรี และการแปลงเจเนอรัลทรีมาเป็นไบนารีทรี

33) การแทรกโหนดแบบ FIFO ในเจเนอรัลทรี จะแทรกโหนดที่ตำแหน่งตอนท้ายของลิสต์พี่น้อง (Sibling) ซึ่งหมายความว่าเป็นการแทรกโหนดใหม่ให้อยู่ตำแหน่งท้ายลิสต์นั่นเอง เมื่อลิสต์ถูกประมวลผล กลุ่มโหนดพี่น้องก็จะถูกโปรเซสตามลำดับคิวแบบมาก่อนออกก่อน ดังรูป



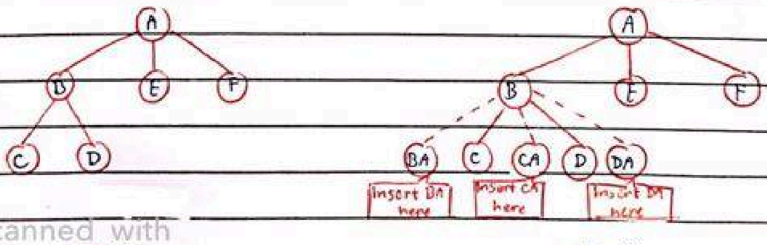
(a) Before insertion (b) After insertion

34) การแทรกโหนดแบบ LIFO การแทรกโหนดแบบ LIFO ในเจเนอรัลทรี จะตรงกันข้ามกับวิธี FIFO กล่าวคือ จะแทรกโหนดที่ตำแหน่งตอนต้นของลิสต์พี่น้อง ซึ่งเป็นไปตามโครงสร้างแบบสแต็กนั่นเอง (มาทีหลังออกก่อน) ดังรูป



(a) Before insertion (b) After insertion

35) การแทรกโหนดแบบ (Key-Sequenced) ในบางครั้งกฎการแทรกโหนดที่ใช้กันมากที่สุดกลับเป็นการแทรกโหนดตามลำดับคีย์ โดยจะแทรกที่ตำแหน่งโหนดใหม่ตามลำดับคีย์ซึ่งอยู่ระหว่างโหนดพี่น้อง สำหรับการกระทำการแทรกในรูปแบบนี้จะคล้ายกันกับการแทรกโหนดลงในลิสต์ ดังรูป



(a) Before (b) After

NO.....

Date...../...../.....

36) การลบโหนดในเจเนอริคทรี แม้ทุกอย่างไปสามารถพัฒนาตามกฎเกณฑ์มาตรฐานที่แปลกเพื่อใช้กับการแทรกโหนดในเจเนอริคทรีแต่สามารถพัฒนาตามกฎเกณฑ์มาตรฐานสำหรับการลบโหนดได้ กฎข้อแรกคือ โหนดที่ถูกลบจะต้องเป็นโหนดใบเท่านั้น ดังนั้น โหนดที่มีลูกอยู่ไม่สามารถลบได้ และหากมีความพยายามลบโหนดที่มีลูก โปรแกรมจะแสดงข้อความเตือนว่า "ไม่สามารถลบได้ หากต้องการลบโหนดลูกออกไปเสียก่อน" เป็นต้น

37) การแปลงเจเนอริคทรีมาเป็นไบนารีทรี โดยปกติการแทนค่าไบนารีทรีลงในโปรแกรมจะทำได้ง่ายและสะดวกกว่าเจเนอริคทรี ดังนั้น หากต้องการแทนค่าเจเนอริคทรีในโปรแกรม ก็สามารถแปลงจากเจเนอริคทรีมาเป็นไบนารีทรีได้ ซึ่งวิธีการแปลงจะมีอยู่ 3 ขั้นตอนหลักๆ ดังนี้คือ

1) กำหนดลอยน์โหนดให้กับโหนดน้องเพื่อชี้ไปยังโหนดลูกซ้ายสุด

2) เชื่อมโยงโหนดพี่น้อง (Sibling) เข้าด้วยกัน

3) ลบ Branch ที่ไม่เกี่ยวข้องออกไปให้หมด

ภายหลังจากการทำเป็นงานจนครบ 3 ขั้นตอนแล้ว ให้ปรับทรีเสียใหม่ก็จะได้ผลลัพธ์ออกมาเป็นไบนารีทรี

NO..............................

Date........../........../........

สรุปท้ายบทที่ 6

- ทรี ประกอบด้วยโหนดต่างๆ สำหรับบรรจุข้อมูล โดยเส้นที่เชื่อมโยงโหนดเข้าด้วยกันจะเรียกว่า บรานช์ Branch
- จำนวนของบรานช์ที่มีความสัมพันธ์กับโหนดก็คือ ดีกรีของโหนด
- ถ้าทรีไม่ใช่เป็นทรีว่าง โหนดแรกจะเรียกว่า รูทโหนด (Root)
- โหนดที่มีตั้งแต่สองโหนดขึ้นไปและมีพ่อเดียวกันจะเรียกว่า โหนดพี่น้อง (Siblings)
- การแทนที่ทรี สามารถแทนด้วยแผนภาพโครงสร้างทรีทั่วไป แบบย่อหน้า และแบบวงเล็บ
- ไบนารีทรี คือทรีที่สามารถมีลูกได้ไม่เกินสองโหนด โดยอาจมีลูกเพียงด้านซ้าย หรือด้านขวา หรืออาจมีลูกทั้งซ้ายและขวา หรืออาจไม่มีลูกเลยก็ได้
- ความลึกสูงสุดของทรี คำนวณได้จากสูตร $H_{max} = N$
- ความลึกต่ำสุดของทรี คำนวณได้จากสูตร $H_{min} = [\log_2 N] + 1$
- จำนวนโหนดน้อยที่สุด ที่ทรีสามารถมีได้ คำนวณได้จากสูตร $N_{min} = H$
- จำนวนโหนดสูงสุด ที่ทรีสามารถมีได้ คำนวณได้จากสูตร $N_{max} = 2^H - 1$
- ความสมดุล ของไบนารีทรีจะมีความสมดุลต่อเมื่อ ค่าของซับทรีมี Balance Factor เท่ากับ 0 ซึ่งคำนวณได้จากสูตร $B = H_L - H_R$
- ไบนารีทรีแบบสมบูรณ์ จะมีจำนวนโหนดสูงสุดที่สามารถมีได้ ซึ่งเป็นไปตามสูตร N_{max} โดยโหนดของซับทรีด้านซ้ายและซับทรีด้านขวาจะมีจำนวนเท่ากัน
- วิธีการท่องเข้าไปในทรี จะมีอยู่ 2 วิธีด้วยกันคือ วิธีการท่องแบบแนวลึก Depth-First และวิธีการท่องแบบแนวกว้าง Breadth-First โดยวิธีการท่องแบบแนวลึกนั้นสามารถแบ่งออกเป็น 3 วิธีพื้นฐานด้วยกันคือ แบบ Preorder, Inorder และ Postorder
- เอ็กซ์เพรสชั่นทรี คือไบนารีทรีที่บรรจุข้อมูลเป็นนิพจน์คณิตศาสตร์ ซึ่งการบรรจุต้องเป็นไปตามคุณสมบัติของเอ็กซ์เพรสชั่นทรีด้วย
- วิธีการท่องเข้าไปในเอ็กซ์เพรสชั่นทรี มี 3 วิธีด้วยกันคือ แบบ Infix, Postfix และ Prefix
- เจเนอรัลทรี หรือทรีแบบทั่วไป เป็นทรีที่สามารถมีโหนดได้ไม่จำกัดจำนวน นอกจากนี้ยังสามารถแปลงเจเนอรัลทรีมาเป็นไบนารีทรีได้ ซึ่งการทำเช่นนี้ก็ทำให้การแทนไบนารีทรีลงในหน่วยความจำนั้นทำได้ง่ายกว่า

คำถามบทที่ 7

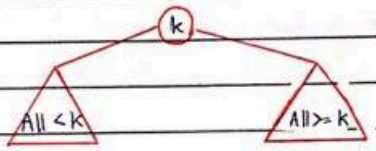
1) BST คือ
2) BST Operations คือ
3) Traversals คือ
4) การท่องเข้าไปในไบนารีเสิร์ชทรีด้วยวิธี Inorder มีประโยชน์อย่างไร
5) Searches คือ
6) การค้นหาค่าที่สูงสุดจาก BST มีอัลกอริทึมอย่างไร
7) การค้นหาค่าสูงสุดใน BST คือ
8) การค้นหาโหนดใน BST คือ
9) Insertion คือ
10) Deletion คือ
11) AVL Search Trees คือ
12) ความสูงของต้นไม้ทรีมีผลต่อความไม่สมดุลของต้นไม้อย่างไร
13) ความสมดุลของทรี คือ
14) สาเหตุที่ทำให้ไม่สมดุลสามารถเกิดขึ้นได้กี่กรณี
15) Left of Left มีอัลกอริทึมอย่างไร
16) Left of Left รูปแบบง่ายๆคือ
17) Left of Left รูปแบบซับซ้อนคือ
18) Right of Right รูปแบบง่ายๆ คือ
19) Right of Right รูปแบบซับซ้อน คือ
20) Right of Left รูปแบบง่ายๆคือ
21) Right of Left รูปแบบซับซ้อน คือ
22) Left of Right รูปแบบง่ายๆ คือ
23) ฮีปทรี คือ
24) การสร้างฮีปมีกี่แบบ อะไรบ้าง
25) Heap Up คือ
26) Heap Down คือ
27) การรีบฮีป คือ
28) Heapify คือ
29) การหาโหนดลูกในฮีป คือ
30) การลบโหนดออกจากฮีป คือ

คำตอบบทที่ 7

1) ไบนารีเสิร์ชทรี (Binary Search Trees : BST)

ไบนารีเสิร์ชทรีเป็นการนำไบนารีทรีมาประยุกต์ใช้เพื่อค้นหาข้อมูล โดยมีคุณสมบัติดังนี้

1. ทุก ๆ โหนดในซับทรีด้านซ้ายของมีค่าน้อยกว่ารูทโหนด
2. ทุก ๆ โหนดในซับทรีด้านขวาของมีค่ามากกว่าหรือเท่ากับรูทโหนด
3. แต่ละซับทรีของมีคุณสมบัติในตัวมันเองคือเป็นไบนารีเสิร์ชทรี



2) การดำเนินงานของไบนารีเสิร์ชทรี (BST Operations) เนื้อหาในบทนี้จะกล่าวถึงการดำเนินงานต่าง ๆ บนไบนารีเสิร์ชทรี โดยเริ่มต้นจากอัลกอริทึมการแทรกข้อมูลเข้าไปในทรีที่ได้สร้างขึ้นมาแล้ว ต่อมาก็จะเรียนรู้อัลกอริทึมการค้นหาข้อมูลตามด้วยการปฏิบัติการเกี่ยวกับการแทรกและลบโหนด

3) การท่องเข้าไปในทรี (Traversals) จากความรู้บทที่ 6 ที่ผ่านมาได้เรียนรู้ถึงวิธีการท่องเข้าไปในทรีชนิดต่าง ๆ มาแล้ว แต่การท่องเข้าไปในทรีในครั้งนี้ จะได้มุ่งในเรื่องการปฏิบัติงานหรือท่องฝากใด หรือจะนำผลลัพธ์ที่ได้จากการท่องเข้าไปในไบนารีเสิร์ชทรีมาใช้ ดังนั้นจึงได้นำวิธี Preorder, Postorder และ Inorder เพื่อดูว่าผลลัพธ์ที่ได้จะนำไปใช้ประโยชน์อย่างไรได้บ้าง

4) ปรากฏว่าการท่องเข้าไปในไบนารีเสิร์ชทรีด้วยวิธี Inorder จะได้เรียงประโยชน์ในทางปฏิบัติมากที่สุด กล่าวคือ ลูกค้าของที่ผ่านการท่องด้วยวิธีนี้จะได้เรียงการลำดับข้อมูลจากน้อยไปมาก ในขณะเดียวกันก็ใช้วิธีเดียวกันนี้ท่องจากโหนดฝั่งขวาไปซ้าย สำหรับลูกค้าเองที่ได้รับจะถูกจัดเรียงจากมากไปน้อย

5) การค้นหา (Searches) เนื้อหาต่อไปนี้จะสึกษาอัลกอริทึมเพื่อการค้นหาข้อมูล 3 โปรแกรมด้วยกัน อันได้แก่ การค้นหาโหนดที่มีค่าต่ำสุดคือ 12 ซึ่งอยู่โหนดใบด้านซ้ายของทรี การค้นหาโหนดที่มีค่าสูงสุดคือ BST เป็นวิธีการค้นข้ามกับหมวดแรก และการค้นหาโหนดตามต้องการ จะพบว่าค่าที่ต้องการค้นหานั้นอยู่ก่อนรูทโหนดจึงเปรียบเทียบค่าเป้าหมายกับทรีฝั่งซ้ายต่อไป

6) การค้นหาโหนดที่มีค่าต่ำสุด จาก BST จะพบว่าโหนดที่มีค่าต่ำสุดคือ 12 ซึ่งอยู่โหนดใบด้านซ้ายของทรี ดังนั้น การค้นหาโหนดที่มีค่าต่ำสุดใน BST จึงทำได้ง่ายมากด้วยการเดินท่องไปจนกระทั่งโหนดปลายด้านซ้ายที่มีค่าต่ำที่สุด สำหรับรหัสโค้ดการค้นหาโหนดที่มีค่าต่ำสุดจะกล่าวไว้ในอัลกอริทึม

NO.....

Date...../...../.....

7) การค้นหาโหนดที่มีค่าสูงสุดใน BST เป็นวิธีแบบค้นซ้ำแบบแรก ดังนั้น การค้นหาจะเริ่มค้นจากรูตโหนดแล้วเดินท่องไปตามบทนิยามฝั่งขวาจนกระทั่งพบโหนดใบที่มีค่าสูงสุด สำหรับขั้นตอนการค้นหาโหนดที่มีค่าสูงสุดแสดงไว้ในอัลกอริทึมการค้นหาโหนดที่มีค่าสูงสุดใน BST

8) การค้นหาโหนดใน BST ณ ขณะนี้เราได้ปรับปรุงขั้นตอนวิธีเสร็จเรียบร้อยแล้ว คราวนี้ลองมาดูว่าการค้นหาข้อมูลเฉพาะโหนดที่ต้องการใน BST นั้นทำอย่างไร พิจารณาที่ต้นไม้ BST ในตัวอย่างโดยสมมติว่าค่าเป้าหมายที่ต้องการค้นหาคือ 20 เมื่อได้ค่าค่านี้ไปเปรียบเทียบกับรูตโหนดก่อนว่าเล็กกว่าหรือมากกว่าในตัวอย่าง จะพบว่าค่าที่ต้องการค้นหานั้นน้อยกว่ารูตโหนด จึงเปรียบเทียบค่าเป้าหมายกับซ้ายเพื่อเข้าไปค้นต่อในซับทรี 15 ซึ่งผลการเปรียบเทียบค่าเป้าหมายมีค่ามากกว่า จึงต้องสอบถามโหนดนี้ทางขวาไปจนพบในที่สุด ก็จะได้ขั้นตอนการค้นหาโหนดที่ต้องการดังแสดงไว้ในอัลกอริทึม

9) การแทรก (Insertion) มีขั้นตอนการแทรกโหนดที่มีค่าสูงสุดใน BST เป็นวิธีการค้นซ้ำแบบแรก ดังนั้น การค้นหาจะเริ่มค้นจากรูตโหนดแล้วเดินท่องไปตามบทนิยามฝั่งขวาจนกระทั่งพบโหนดใบที่มีค่าสูงสุด สำหรับขั้นตอนการค้นหาโหนดที่มีค่าสูงสุดแสดงไว้ในอัลกอริทึม

10) การลบ (Deletion) การลบโหนดออกจากไบนารีเสิร์ชทรี จะต้องค้นหาตำแหน่งโหนดที่ต้องการลบให้ได้ก่อน ซึ่งสถานการณ์เป็นไปได้อยู่ 4 ประการด้วยกัน ดังนี้

1) กรณีโหนดที่ลบไม่มีลูก ให้ลบโหนดนั้นออกไปได้เลยทันที

2) กรณีโหนดที่ลบมีเฉพาะซับทรีด้านขวา ให้ลบโหนดดังกล่าวทิ้งไปแล้วลิงค์ซับทรีด้านขวาขึ้นมาแทน

3. กรณีโหนดที่ลบมีเฉพาะซับทรีด้านซ้าย ให้ลบโหนดดังกล่าวทิ้งไปแล้วลิงค์ซับทรีด้านซ้ายขึ้นมาแทน

4) กรณีโหนดที่ลบมีสองซับทรี ซึ่งสถานการณ์เป็นไปได้ที่จะลบโหนดดังกล่าวออกไป เพราะว่าอาจจะทำให้เกิดความไม่สมดุลของทรีเกิดขึ้น

11) เอวีแอลไบนารีเสิร์ชทรี (AVL Search Trees) ซึ่งทรีแบบ AVL ก็คือ BST ที่มีความสมดุล โดยพิจารณาความสูงของทรีย่อยทางซ้ายและทางขวาของทรีนั้นมีค่าไม่มากไปกว่า 1 จึงกล่าวได้ว่าเป็นทรีที่มีความสมดุล จึงกล่าวสรุปได้ว่า เฮ้าเดอทรีก็คือไบนารีเสิร์ชทรีที่ซับทรีด้านซ้ายและซับทรีด้านขวาจะมีค่า Balance Factor เท่ากับ +1, 0 หรือ -1 เท่านั้น

12) สองเราได้ว่า ความสูงของทรีจะส่งผลต่อความเร็วในการค้นหาข้อมูล ดังนั้น เมื่อทรีมีความสมดุลดีโครงสร้างทรีย่อยไม่เอนเอียงไปด้านใดด้านหนึ่ง พิจารณาเปรียบเทียบข้อมูลในทรีจะมีจำนวนครั้งน้อยลงเมื่อสามารถค้นเข้าไปข้างใน แต่ถ้าเป็นทรีที่ไม่มีความสมดุล เช่น มีการเอนเอียงไปด้านใดด้านหนึ่งมากเป็นพิเศษ ก็จะทำให้จำนวนครั้งในการเปรียบเทียบข้อมูลมากขึ้นในระหว่างการค้นหาเข้าไปในทรี

Date...../...../.....

๒) ต้นไม้สมดุลของทรี (Balancing Trees) ทรีสามารถเติบโตหรือมีขนาดลดลงได้จากการเพิ่มหรือลบโหนดออกจากทรี ดังนั้น ผลลัพธ์จากการเพิ่มหรือลบโหนดฟังก์ชันอาจทำให้ทรีเกิดต้นไม้ไม่สมดุลขึ้นได้ หลายกรณีการแก้ต้นไม้ที่เกิดต้นไม้ไม่สมดุลขึ้นจะต้องทำการปรับให้กลับมาสมดุลด้วยวิธีการหมุนโหนด Rotating Nodes

๓) ทรีที่ไม่สมดุลสามารถเกิดขึ้นได้จาก 4 กรณี ดังนี้

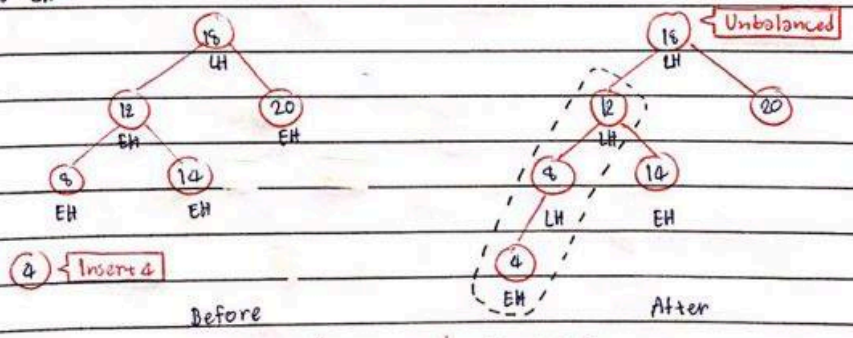
1) Left of Left หมายถึง ซับทรีของทรีต้นนั้นมีอาการซ้ายสูง จึงทำให้ทรีต้นซ้ายสูงไปด้านซ้าย (ซับทรี LH ของทรี LH)

2) Right of Right หมายถึง ซับทรีของทรีต้นนั้นมีอาการขวาสูง จึงทำให้ทรีต้นขวาสูงไปด้านขวา (ซับทรี RH ของทรี RH)

3) Right of Left หมายถึง ซับทรีของทรีต้นนั้นมีอาการซ้ายสูงจนกลายเป็นขวาสูง (รูต LH แต่ซับทรีซ้าย RH)

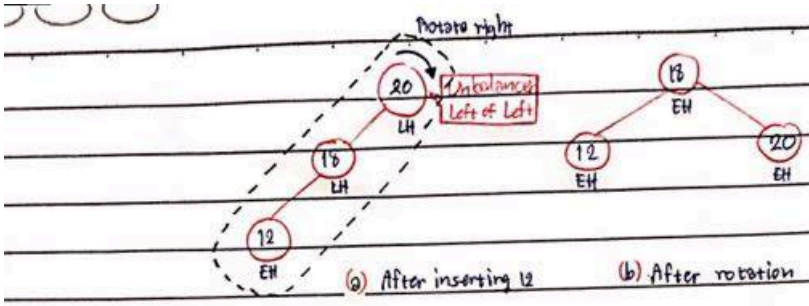
4) Left of Right หมายถึง ซับทรีของต้นนั้นมีอาการสูงจนกลายเป็นซ้ายสูง (รูต RH แต่ซับทรีขวา LH)

๔) กรณีที่ 1 : Left of Left เป็นกรณีไม่สมดุลตามกรณีที่ 1 ดังรูปที่ ๗.๑๑ (๑) โดยเมื่อ insert โหนด 4 เข้าไปทำให้ทรีต้นซ้ายของโหนดลูกฝั่งซ้าย ทำให้ทรีไม่สมดุลตามเงื่อนไข "ซับทรี LH ของทรี LH"



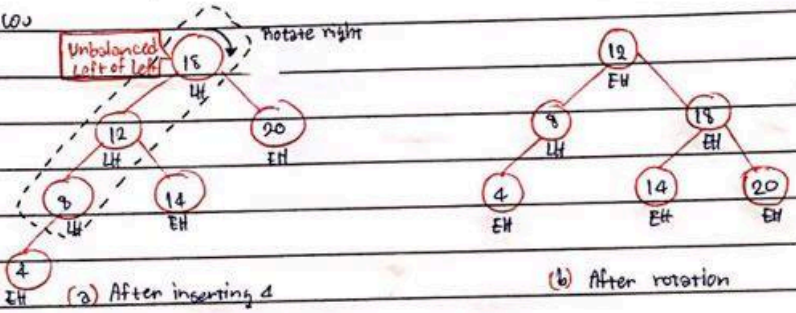
รูปที่ ๗.๑๑ (๑) กรณีที่ 1 : Left of Left

๕) เมื่อต้นไม้สมดุลของทรีเกิดขึ้นเช่นกรณีดังกล่าว การแก้ไขให้ทรีกลับมาสมดุลสามารถทำได้ด้วยการหมุนโหนดที่ไม่สมดุลไปทางขวา รูปที่ ๗.๑๒ เป็นตัวอย่างการหมุนโหนดอย่างง่ายเพื่อให้เกิดภาพเข้าใจในเบื้องต้นก่อน หมุนย้ายโหนด 20 ไปสมดุลแทนซับทรี 18 สังเกตว่าซ้ายสูง จึงต้องหมุนโหนด 20 ไปทางขวาผลที่ได้คือ จะได้รูตโหนด 18 ที่มีโหนด 20 เป็นซับทรีอยู่ด้านขวาซึ่งเป็นทรีที่สมดุลในที่สุด



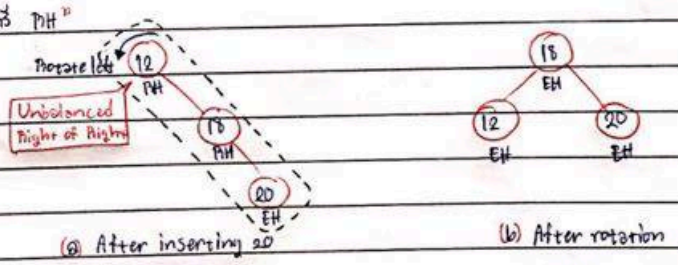
(a) After inserting 12 (b) After rotation

ที่ตรงนี้ลองมาดูรูปแบบที่ซับซ้อนขึ้นบ้าง กล่าวคือ เมื่อโหนดเดิมได้สูญเสียซับทรีด้านขวาไป 12 ไปใน ขณะที่หมุนจึงทำให้ซับทรีด้านซ้ายของโหนด 18 ว่างเปล่า เพราะถึงโหนด 14 มาทดแทนตำแหน่งโหนดที่ว่าง การกระทำเช่นนี้ก็เพื่อคงคุณสมบัติของ BST ที่ทุกๆ โหนดบนพื้นราบของมุมตรงต้องมีค่ามากกว่าซ้ายสำหรับมุม โหนดนั่นเอง



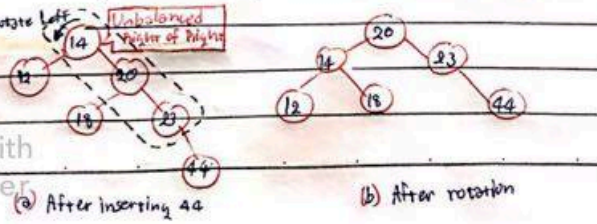
(a) After inserting 4 (b) After rotation

18) กรณีที่ 2 : Right of Right เป็นกรณีที่ไม่สมดุลตามกรณีที่ 2 ดังรูป b ซึ่งตรงข้ามกับแบบแรกโดยมีการแทรกโหนด 44 เข้าไปตรงตำแหน่งซับทรีด้านขวาของโหนดสูงด้านขวา ที่ไม่สมดุลตามข้อที่ไว้ "ซับทรี RH ของที่ RH"



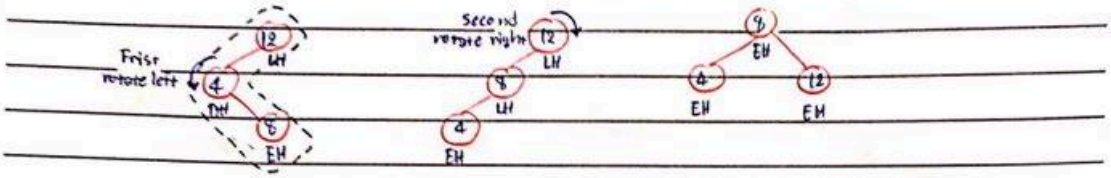
(a) After inserting 20 (b) After rotation

19) เมื่อการไม่สมดุลของทรีเกิดขึ้นกับกรณีดังกล่าว การปรับทรีให้กลับมาสมดุลจะต้องทำการ หมุนโหนดที่ไม่สมดุลไปทางซ้าย ในส่วนของเกิพาก็นสับรูป ซึ่งเป็นการหมุนโหนด Right of Right ในรูปแบบของฝั่งซ้าย ในขณะที่รูปเป็น การที่ทำซับซ้อนขึ้นขึ้น

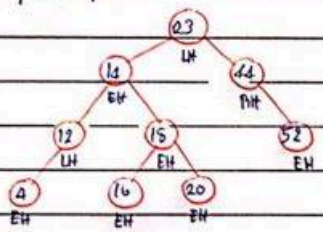


(a) After inserting 44 (b) After rotation

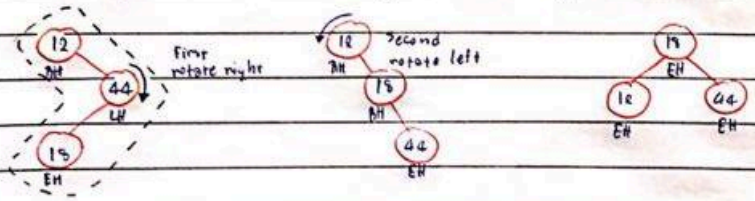
22) กรณีที่ 3 : Right of Left เป็นกรณีไม่สมดุลจากกรณีที่ 3 ที่ได้มาจากโหนด 12 เข้าไปตรงตำแหน่งซ้ายสันขวา ภาพใส่รูตทรีเบื้องซ้าย จึงเกิดในทรีไม่สมดุลตามเงื่อนไข "รูต LH และซับทรีซ้าย RH" การไม่สมดุลของทรีในกรณี Right of Left จะต้องแบ่งโหนดลงสองรอบซ้ายขวาสลับสี่ แบ่งโหนดแรกไปทางซ้ายพอ แบ่งอีกโหนดแรกไปทางขวา จึงสามารถทำให้ทรีโนกรับมาสมดุล



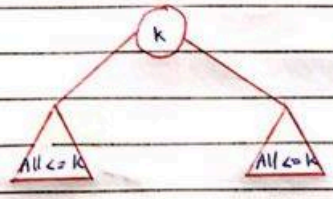
23) กรณีสุดท้ายรูปแบบที่มีความซับซ้อนเพิ่มขึ้น เพราะทรีที่ได้สิ้นสุดได้ซับทรีด้านซ้ายใหม่คือโหนด 12 ซึ่งทำให้ไปเชื่อมรูปจนกลายเป็นทรีที่สมดุลในที่สุด



24) กรณีที่ 4 : Left of Right เป็นกรณีทรีไม่สมดุลตรงตามกรณีที่ 4 ต้องรูปที่ได้มาจากโหนด 19 เข้าไปตรงตำแหน่งซับทรีด้านซ้าย ภาพใส่รูตทรีเบื้องขวา จึงทำให้ทรีไม่สมดุลตรงตามเงื่อนไข "รูต RH แลซับทรีขวา LH"



25) อีกกรณี คือ ไม่มีทรีที่ซับทรีด้านซ้ายและซับทรีด้านขวามีความสูงน้อยกว่าหรือเท่ากับโหนดพ่อในขณะที่รูตโหนดของซีเควนซ์ประสบปัญหาต้องมีสามารถที่สุดในทรี



NO...............................

Date............/............./.........

24) อัลกอริทึมการสร้างฮีป การสร้างฮีปได้ 2 แนวทางคือ 1) เริ่มจากอาร์เรย์ว่างเปล่า และทำการแทรกสมาชิกเข้าไปในอาร์เรย์ และ 2) นำข้อมูลในอาร์เรย์ไปตรวจสอบคุณสมบัติของฮีปให้ปรับตำแหน่งสมาชิกในอาร์เรย์ใหม่เพื่อให้ตรงตามคุณสมบัติ โดยจะมีอัลกอริทึมพื้นฐานอยู่ 2 โปรแกรมด้วยกันคือ การรีฮีปอัป (Reheap Up) และการรีฮีปดาวน์ (Reheap Down) เพื่อจัดการกับการดังกล่าว

25) Reheap Up เป็นอัลกอริทึมปรับโครงสร้างฮีปที่ใช้เมื่อมีการแทรกโหนดใหม่เข้ามา โดย Reheap Up จะใช้รีเคอร์ซีฟในการเลื่อนย้ายโหนดใหม่ขึ้นไปยังทรี หากคีย์ของโหนดใหม่ที่แทรกเข้ามามีค่ามากกว่าคีย์โหนดพ่อ ให้ทำการเปลี่ยนโหนดและเรียกตัวเอง (รีเคอร์ซีฟ) ซึ่งเป็นไปตามอัลกอริทึม

26) Reheap Down เป็นอัลกอริทึมปรับโครงสร้างฮีปกรณีที่มีการลบโหนดออกจากทรี หรือการคือ จะดันโหนดลงไปในฮีปในกรณีที่รูทโหนดคือโหนดพ่อนั้นมีค่าน้อยกว่าโหนดลูกด้วย ก็จะเปลี่ยนตำแหน่งกัน ซึ่งจะทำงานแบบรีเคอร์ซีฟไปจนกระทั่งทรีนั้นเป็นไปตามคุณสมบัติของฮีป โดยอัลกอริทึมเป็นซูโดโค้ดการ Reheap Down

27) การสร้างฮีป เมื่อลำดับโหนดสมาชิกของอาร์เรย์มีการจัดเรียงแบบสุ่ม การสร้างฮีปจึงมีการปรับโครงสร้างใหม่เพื่อให้แต่ละโหนดในฮีปมีค่ามากกว่าโหนดลูก เพราะเริ่มจากการแบ่งส่วนอาร์เรย์ออกเป็น 2 ส่วนด้วยกันคือฮีปฝั่งซ้ายกับฮีปฝั่งขวา โดยจุดเริ่มต้นของรูท (โหนดแรก) จะเป็นโหนดในฮีป ส่วนที่เหลือในอาร์เรย์ก็คือข้อมูลที่จะถูกแทรกเข้าไปโดยโครงสร้างดังกล่าว

28) การแทรกโหนดลงในฮีป เพราะจะทำตามเงื่อนไขของโหนดพ่อให้ขยับขึ้นไปในฮีป มีการสลับโหนด หากไม่ได้เรียงลำดับ แต่ถ้าโหนดเรียงลำดับอยู่แล้วการแทรกก็จะจบลง และโหนดถัดไปก็จะถูกเลือกให้ไปลงในฮีป ซึ่งกระบวนการดังกล่าวบางครั้งเรียกว่า Heapify อัลกอริทึมการสร้างฮีปดูเรียบง่ายมาก

29) การแทรกโหนดลงในฮีป เมื่อฮีปถูกสร้างขึ้นมาแล้ว จึงสามารถแทรกโหนดลงไปในอาร์เรย์ได้ โดยต้องหาตำแหน่งที่ว่างแรกในอาร์เรย์ก่อน ซึ่งจะพบได้ทันทีตรงท้ายโหนด ในการแทรกโหนดเพราะย้ายข้อมูลใหม่ไปยังพื้นที่ว่าง โหนดใบ และทำการ Reheap Up ซึ่งเป็นไปตามหลักการ Reheap Up

30) การลบโหนดออกจากฮีป เมื่อต้องการลบโหนดออกจากฮีป กรณีที่มักจะพบบ่อยที่สุดก็คือ การลบที่รูทโหนด และในความเป็นจริงเหตุผลของฮีปก็คือการทำแนก และแพทเทิร์นประกอบที่มีค่าใหญ่ที่สุดซึ่งก็คือรูทนั่นเอง โดยภายหลังจากการลบแล้วทำการ Reheap Down ซึ่งเป็นไปตามหลักการ Reheap Down

สรุปท้ายบทที่ 9

- ไบนารีเสิร์ชทรี มีคุณสมบัติดังนี้

1) ทุกๆ โหนดในซับทรีด้านซ้ายต้องมีค่าน้อยกว่ารูทโหนด

2) ทุกๆ โหนดในซับทรีด้านขวาต้องมีค่ามากกว่าหรือเท่ากับรูทโหนด

3) แต่ละซับทรีต้องมีคุณสมบัติในตัวมันเองคือเป็นไบนารีเสิร์ชทรี

- การท่องเข้าไปในไบนารีเสิร์ชทรี ด้วยวิธี Inorder ส่งผลให้ข้อมูลที่ได้เรียงลำดับ
- การค้นหาค่าต่ำสุดในไบนารีเสิร์ชทรี สามารถค้นหาได้จากโหนดใบที่อยู่ตำแหน่งซ้ายสุดของทรี
- การค้นหาค่าสูงสุดในไบนารีเสิร์ชทรี สามารถค้นหาได้จากโหนดใบที่อยู่ตำแหน่งขวาสุดของทรี
- การค้นหาค่าที่ต้องการในไบนารีเสิร์ชทรี ขั้นตอนจะต้องเปรียบเทียบค่าเป้าหมายกับรูทโหนด ถ้าค่าเป้าหมายมีค่าน้อยกว่ารูทโหนดก็จะค้นหาซ้ำๆ เฉพาะที่ซับทรีด้านซ้ายจนพบ หรือถ้าค่าเป้าหมายมีค่ามากกว่ารูทโหนดก็จะค้นหาซ้ำๆ เฉพาะที่ซับทรีด้านขวาจนพบ หรือถ้าค่าเป้าหมายตรงกับรูทโหนดก็จะค้นพบได้ทันที
- การแทรกโหนดในไบนารีเสิร์ชทรี จะทำการแทรกได้ที่ตำแหน่งโหนดใบ หรือโหนดที่คล้ายกับโหนดใบ (โหนดที่มีซับทรีเป็นทรีว่าง)
- การลบโหนดในไบนารีเสิร์ชทรี ปฏิบัติการได้ใน 4 กรณีคือ

1) โหนดที่ลบเป็นโหนดใบที่ไม่มีลูก

2) โหนดที่ลบมีเพียงซับทรีด้านขวา

3) โหนดที่ลบมีเพียงซับทรีด้านซ้าย

4) โหนดที่ลบมีทั้งสองซับทรี

- เอวีแอลทรี คือ ไบนารีเสิร์ชทรีที่ความสูงของซับทรีเมื่อพิจารณาแล้วจะมีค่าไม่เกินกว่า 1 ซึ่งหมายความว่าทรีนั้นจะเป็นทรีที่มีความสมดุล โดยค่า Balanced Factor ของโหนด ถ้าหากมีค่าเป็น +1 หมายถึงซ้ายสูง (Left Hight: LH), มีค่าเป็น 0 จะหมายถึงสูงเท่ากัน (Even Hight: EH) และมีค่าเป็น -1 หมายถึงขวาสูง (Right Hight: RH)
- ความไม่สมดุลของทรี เกิดขึ้นได้ภายหลังจากการแทรกหรือลบโหนดออกจากทรี โดยจะมี 4 กรณีด้วยกันที่ทำให้เกิดความไม่สมดุลของทรี ซึ่งประกอบด้วย Left of Left, Right of Right, Right of Left และ Left of Right
- ฮีปทรี คือ ไบนารีทรีที่ซับทรีด้านซ้ายและซับทรีด้านขวาจะมีค่าน้อยกว่าหรือเท่ากับโหนดพ่อ ในขณะที่รูทโหนดของฮีปจะรับประกันว่าต้องมีค่ามากที่สุดในทรี
- Reheap Up เป็นการปรับเปลี่ยนตำแหน่งการนำสมาชิกตัวสุดท้ายที่เพิ่มขึ้นไปบนทรี (โหนดลูกมีค่ามากกว่าโหนดพ่อ) และจะจัดตำแหน่งสับเปลี่ยนกระทั่งอยู่ในตำแหน่งที่เหมาะสมตรงตามคุณสมบัติของฮีปทรี

Reheap Down เป็นการปรับเปลี่ยนตำแหน่งการนำโหนดที่เป็นรูทลงไปในทรี (โหนดพ่อมีค่าน้อยกว่าโหนดลูก) และจะจัดตำแหน่งสับเปลี่ยนกระทั่งอยู่ในตำแหน่งที่เหมาะสมตรงตามคุณสมบัติของฮีปทรี