Lecture 9

เอกสารประกอบการบรรยาย
Data Structure

เรื่อง Graph

Graph

เนื้อหา

- โครงสร้างข้อมูลแบบกราฟ
- นิยามของกราฟ
- การแทนที่กราฟในหน่วยความจำ
- การท่องไปในกราฟ

จุดประสงค์การเรียนรู้

1. เพื่อให้นักศึกษาทราบโครงสร้างข้อมูลแบบกราฟ และการทำงาน
2. เพื่อให้นักศึกษาทราบนิยามของกราฟ
3. เพื่อให้นักศึกษาทราบกระบวนวิธีการแทนที่กราฟในหน่วยความจำ
4. เพื่อให้นักศึกษาทราบวิธีการท่องไปในกราฟในแบบข้อมูลชนิดต่าง ๆ

Graph (Cont.)

กราฟ (Graph) เป็นโครงสร้างข้อมูลแบบไม่ใช่เชิงเส้น อีกชนิดหนึ่ง กราฟเป็นโครงสร้างข้อมูลที่มีการนำไปใช้ในงานที่เกี่ยวข้องกับการแก้ปัญหาที่ค่อนข้างซับซ้อน เช่น การวางข่าย งานคอมพิวเตอร์ การวิเคราะห์เส้นทางวิกฤติ และปัญหาเส้นทางที่สั้นที่สุด เป็นต้น

Graph (Cont.)

นิยามของกราฟ

กราฟ เป็นโครงสร้างข้อมูลแบบไม่ใช่เชิงเส้น ที่ประกอบ ด้วยกลุ่มของสิ่งสองสิ่งคือ

(1) โหนด (Nodes) หรือ เวอร์เทกซ์ (Vertexes)

(2) เส้นเชื่อมระหว่างโหนด เรียก เอ็จ (Edges)

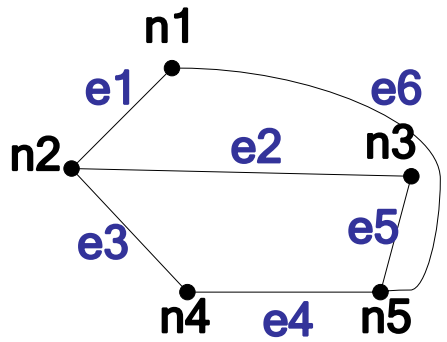
กราฟที่มีเอ็จเชื่อมระหว่างโหนดสองโหนด ถ้าเอ็จไม่มีลำดับ ความสัมพันธ์จะเรียกกราฟนั้นว่า กราฟแบบไม่มีทิศทาง (Undirected Graphs)

Graph (Cont.)

และถ้ากราฟนั้นมีเอ็จที่มีลำดับความสัมพันธ์หรือมีทิศทางกำกับด้วยเรียกกราฟนั้นว่า กราฟแบบมีทิศทาง (Directed Graphs) บางครั้งเรียกว่า ไดกราฟ (Digraph)

ถ้าต้องการอ้างถึงเอ็จแต่ละเส้นสามารถเขียนชื่อเอ็จกำกับไว้ก็ได้

Graph (Cont.)



กราฟแบบไม่มีทิศทางมีชื่อโหนดและเอ็จ

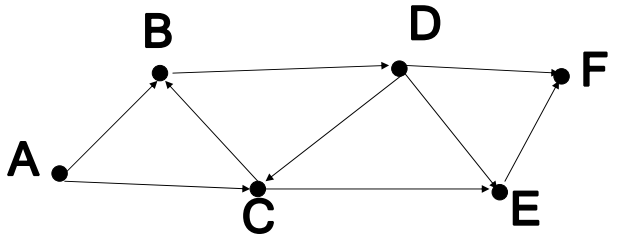
Graph (Cont.)

การเขียนกราฟแสดงโหนดและเส้นเชื่อมความสัมพันธ์ ระหว่างโหนดไม่มีรูปแบบที่ตายตัว การลากเส้นความสัมพันธ์เป็นเส้นลักษณะไหนก็ได้ที่สามารถแสดงความสัมพันธ์ระหว่างโหนดได้ถูกต้อง นอกจากนี้เอ็จจากโหนดใด ๆ สามารถวนเข้าหาตัวมันเองได้

Graph (Cont.)

โดยทั่ว ๆ ไปการเขียนกราฟเพื่อแสดงให้เห็นความสัมพันธ์ ของสิ่งที่เราสนใจแทนโหนดด้วย จุด (pointes) หรือวงกลม (circles) ที่มีชื่อหรือข้อมูลกำกับ เพื่อบอกความแตกต่างของแต่ละโหนดและเอ็จแทนด้วยเส้นหรือเส้นโค้งเชื่อมต่อระหว่างโหนดสองโหนด ถ้าเป็นกราฟแบบมีทิศทางเส้นหรือเส้นโค้งต้องมีหัวลูกศรกำกับทิศทางของความสัมพันธ์ด้วย

Graph (Cont.)

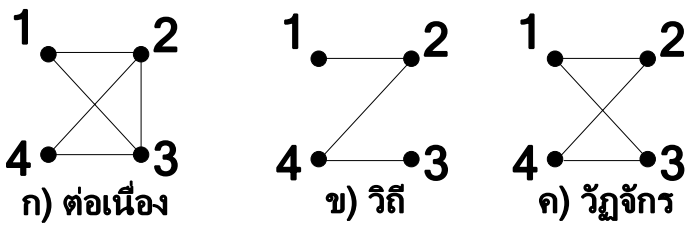


กราฟแบบมีทิศทาง

Graph (Cont.)

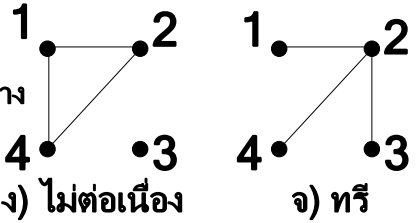
กราฟแบบไม่มีทิศทางเป็นเซตแบบจำกัดของโหนดและเอ็จ โดยเซตอาจจะว่างไม่มีโหนดหรือเอ็จเลยเป็นกราฟว่าง (Empty Graph) แต่ละเอ็จจะเชื่อมระหว่างโหนดสองโหนด หรือเชื่อมตัวเอง เอ็จไม่มีทิศทางกำกับ ลำดับของการเชื่อมต่อกันไม่สำคัญ นั่นคือไม่มีโหนดใดเป็นโหนดแรก (First Node) หรือไม่มีโหนดเริ่มต้น และไม่มีโหนดใดเป็นโหนดสิ้นสุด

Graph (Cont.)



ก) ต่อเนื่อง ข) วิถี ค) วัฏจักร

กราฟแบบไม่มีทิศทาง



ง) ไม่ต่อเนื่อง จ) ทรี

Graph (Cont.)

โหนดสองโหนดในกราฟแบบไม่มีทิศทางถือว่าเป็นโหนดที่ใกล้กัน (Adjacent) ถ้ามีเอ็จเชื่อมจากโหนดที่หนึ่งไปโหนดที่สอง

กราฟ (ก) แสดงกราฟที่มีลักษณะ ต่อเนื่อง (Connected) เป็นกราฟที่มีเส้นทางเชื่อมจากโหนดใด ๆ ไปยังโหนดอื่นเสมอ

กราฟ (ข) แสดงกราฟที่มีลักษณะเป็น วีถี (Path) มีเส้นทางเชื่อมไปยังโหนดต่าง ๆ อย่างเป็นลำดับ โดยแต่ละโหนดจะเป็นโหนดที่ใกล้กันกับโหนดที่อยู่ถัดไป

Graph (Cont.)

กราฟ (ค) แสดงกราฟที่เป็นวัฏจักร (Cycle) ซึ่งต้องมีอย่างน้อย 3 โหนด ที่โหนดสุดท้ายต้องเชื่อมกับโหนดแรก

กราฟ (ง) แสดงกราฟที่มีลักษณะ ไม่ต่อเนื่อง (Disconnected) เนื่องจากไม่มีเส้นทางเชื่อมจากโหนด 3 ไปยังโหนดอื่นเลย

กราฟ (จ) แสดงกราฟที่เป็นทรี โดยทรีเป็นกราฟที่ต่อเนื่อง ไม่มีทิศทาง และไม่เป็นวัฏจักร

Graph (Cont.)

กราฟแบบมีทิศทาง เป็นเซตแบบจำกัดของโหนดและเอ็จ โดยเซตอาจจะว่างไม่มีโหนดหรือเอ็จเลยเป็นกราฟว่าง (Empty Graph) แต่ละเอ็จจะเชื่อมระหว่างโหนดสองโหนด เอ็จมีทิศทางกำกับแสดงลำดับของการเชื่อมต่อกัน โดยมีโหนดเริ่มต้น (Source Node) และ โหนดสิ้นสุด (Target Node)

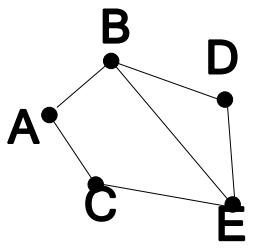
รูปแบบต่าง ๆ ของกราฟแบบมีทิศทางเหมือนกับรูปแบบ ของกราฟไม่มีทิศทาง ต่างกันตรงที่กราฟแบบนี้จะมีทิศทางกำกับด้วยเท่านั้น

Graph (Cont.)

การแทนกราฟในหน่วยความจำ

ในการปฏิบัติการกับโครงสร้างกราฟ สิ่งที่ต้องการจัดเก็บ จากกราฟโดยทั่วไปก็คือ เอ็จ ซึ่งเป็นเส้นเชื่อมระหว่างโหนดสองโหนด มีวิธีการจัดเก็บหลายวิธี วิธีที่ง่ายและตรงไปตรงมาที่สุดคือ การเก็บเอ็จในแถวลำดับ 2 มิติ

Graph (Cont.)



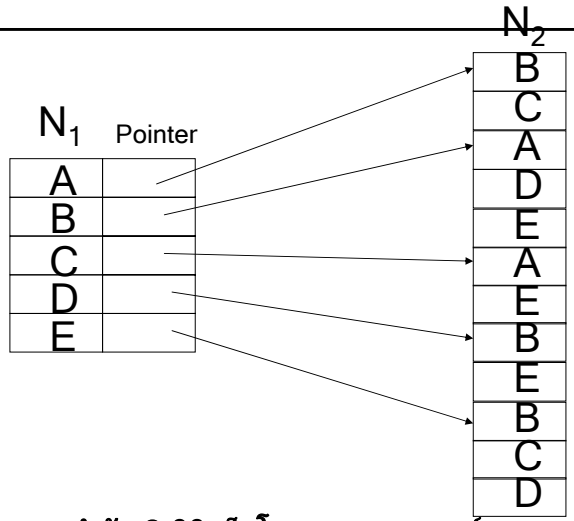
N_1	N_2
A	B
A	C
B	A
B	D
B	E
C	A
C	E
D	B
D	E
E	B
E	C
E	D

การแทนกราฟด้วยแถวลำดับ 2 มิติ กรณีกราฟไม่มีทิศทาง

Graph (Cont.)

การแทนกราฟในหน่วยความจำด้วยวิธีเก็บเอ็จทั้งหมดใน แถวลำดับ 2 มิติ จะค่อนข้างเปลืองเนื้อที่ เนื่องจากมีบางเอ็จที่เก็บซ้ำ อาจหลีกเลี่ยงปัญหานี้ได้โดยใช้แถวลำดับ 2 มิติเก็บโหนดและ พอยเตอร์ชี้ไปยังตำแหน่งของโหนดต่าง ๆ ที่สัมพันธ์ด้วย และใช้ แถวลำดับ 1 มิติเก็บโหนดต่าง ๆ ที่มีความสัมพันธ์กับโหนดในแถวลำดับ 2 มิติ

Graph (Cont.)



กราฟในแถวลำดับ 2 มิติ เก็บโหนดและพอยเตอร์ของกราฟ

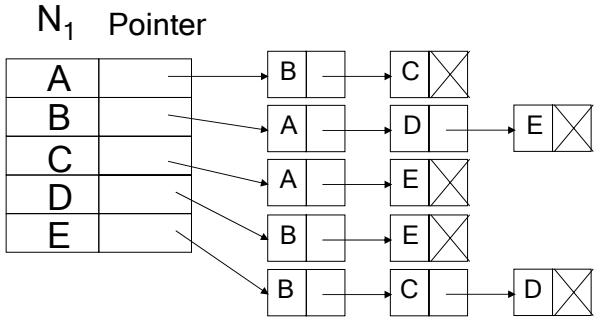
Graph (Cont.)

การจัดเก็บกราฟด้วยวิธีเก็บโหนดและพอยน์เตอร์นี้ยุ่งยาก ในการจัดการเพิ่มขึ้นเนื่องจากต้องเก็บโหนดและพอยน์เตอร์ในแถวลำดับ 2 มิติ และต้องจัดเก็บโหนดที่สัมพันธ์ด้วยในแถวลำดับ 1 มิติ อย่างไรก็ตามทั้งสองวิธีไม่เหมาะกับกราฟที่มีการเปลี่ยนแปลง ตลอดเวลา ควรใช้ในกราฟที่ไม่มีการเปลี่ยนแปลงตลอดการใช้งาน เพราะถ้ามีการเปลี่ยนแปลงส่วนใดส่วนหนึ่งของกราฟจะกระทบกับส่วนอื่น ๆ ที่อยู่ในระดับที่ต่ำกว่าด้วยเสมอ

Graph (Cont.)

กราฟที่มีการเปลี่ยนแปลงตลอดเวลา อาจจะใช้วิธีแอดจาเซนซีลิสต์ (Adjacency List) ซึ่งเป็นวิธีที่คล้ายวิธีจัดเก็บกราฟด้วยการเก็บโหนดและพอยน์เตอร์ แต่ต่างกันตรงที่ จะใช้ ลิงค์ลิสต์แทน เพื่อความสะดวกในการเปลี่ยนแปลงแก้ไข

Graph (Cont.)



การแทนกราฟด้วยวิธีแอดจาเซนซีลิสต์

Graph (Cont.)

วิธีแทนกราฟในความจำหลักอีกวิธีหนึ่งซึ่งเป็นที่นิยมใช้ กันมากที่สุดคือ การแทนด้วยแอดจาเซนซีเมทริกซ์ (Adjacency Matrix) โดยที่ถ้ากราฟ G มีทั้งหมด n โหนด แอดจาเซนซีเมทริกซ์เป็นเมทริกซ์จัตุรัสขนาด n x n สมมติว่าคือ
เมทริกซ์ M แต่ละ M(i,j) เมื่อ i, j = 1, 2, 3, . . ., n จะมีค่าเป็น 1 ถ้ามีเอ็จเชื่อมความสัมพันธ์ระหว่างโหนด i ไปยังโหนด j และ
มีค่าเป็น 0 ถ้าไม่มีเอ็จเชื่อมความสัมพันธ์จากโหนด i ไปยังโหนด j หรืออาจจะกำหนดด้วย ค่าตรรกะ (Boolean Value) ก็ได้

Graph (Cont.)

ลักษณะที่น่าสนใจอย่างหนึ่งของแอดจาเซนซีเมทริกซ์ก็คือ สามารถหาได้ว่ามีจำนวนเส้นทางกี่เส้นทางจากโหนด X_i ไปยังโหนด X_j ใด ๆ ได้ โดยดูจากค่าในแอดจาเซนซีเมทริกซ์ เช่น
ถ้า M เป็นแอดจาเซนซีเมทริกซ์ M^k เป็นการคูณเมทริกซ์ M ด้วยตัวมันเอง k ครั้ง และ M^k_{ij} เป็นสมาชิกในแถวที่ i คอลัมน์ที่ j ของเมทริกซ์ M^k

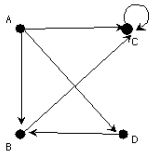
Graph (Cont.)

อาจกล่าวได้ว่า เมทริกซ์ M บอกให้ทราบว่ามีเส้นทางจากโหนดในแถวที่ i ไปยังโหนดในคอลัมน์ที่ j ขนาดหนึ่งจำนวนเส้นทาง เมทริกซ์ M^2 บอกให้เราทราบว่ามีเส้นทางจากโหนดในแถวที่ i ไปยังโหนดในคอลัมน์ที่ j ขนาดสองจำนวนเส้นทาง เมทริกซ์ M^3 บอกให้เราทราบว่ามีเส้นทางจากโหนดในแถวที่ i ไปยังโหนดในคอลัมน์ที่ j ขนาดสามจำนวนเส้นทาง และสมาชิกใด ๆ มีค่าดังนี้

Graph (Cont.)

- ถ้า $M^k_{ij} = 0$ จะได้ว่าไม่มีเส้นทางขนาด k จากโหนดในแถวที่ i ไปยังโหนดในคอลัมน์ที่ j
- ถ้า $M^k_{ij} = p$ เมื่อ p เป็นจำนวนเต็มบวก ได้ว่ามีเส้นทางขนาด k จำนวน p เส้นทาง จากโหนดในแถวที่ i ไปยังโหนดในคอลัมน์ที่ j

Graph (Cont.)



(ก) ตัวอย่างกราฟ

	A	B	C	D
A	0	1	1	1
B	0	0	1	0
C	0	0	1	0
D	0	1	0	0

(ข) เมทริกซ์ M

	A	B	C	D
A	0	1	2	0
B	0	0	1	0
C	0	0	1	0
D	0	1	0	0

(ค) เมทริกซ์ M^2

	A	B	C	D
A	0	0	3	0
B	0	0	1	0
C	0	0	1	0
D	0	0	1	0

(ง) เมทริกซ์ M^3

รูปที่ 7.44 แสดงการแทนกราฟด้วยแอดจาเซนซีเมทริกซ์

Graph (Cont.)

การท่องไปในกราฟ

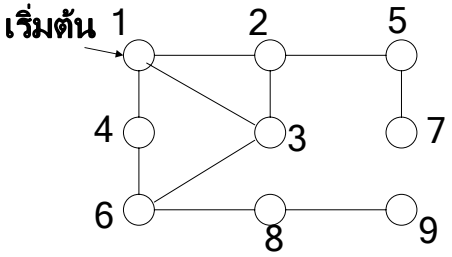
การท่องไปในกราฟ (graph traversal) คือ กระบวนการเข้าไปเยือนโหนดในกราฟ โดยมีหลักในการทำงานคือ แต่ละโหนดจะถูกเยือนเพียงครั้งเดียว สำหรับการท่องไปในทรีเพื่อเยือนแต่ละโหนดนั้นจะมีเส้นทางเดียว แต่ในกราฟระหว่างโหนดอาจจะมีหลายเส้นทาง ดังนั้นเพื่อป้องกันการท่องไปในเส้นทางที่ซ้ำเดิมจึงจำเป็นต้องทำเครื่องหมายบริเวณที่ได้เยือนเสร็จเรียบร้อยแล้ว

เพื่อไม่ให้เข้าไปเยือนอีก สำหรับเทคนิคการท่องไปในกราฟมี 2 แบบดังนี้

Graph (Cont.)

1. การท่องแบบกว้าง (Breadth First Traversal) วิธีนี้ทำโดยเลือกโหนดที่เป็นจุดเริ่มต้น ต่อมาให้เยือนโหนดอื่นที่ใกล้กันกับโหนดเริ่มต้นทีละระดับจนกระทั่งเยือนหมดทุกโหนดในกราฟ

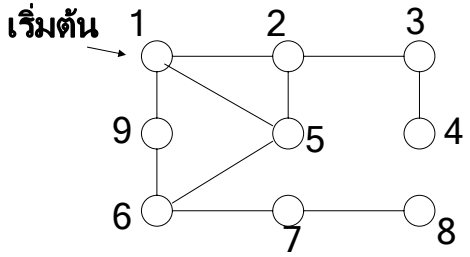
ผลลัพธ์จากการท่อง 1 4 6 2 3 8 5 7 9



Graph (Cont.)

2. การท่องแบบลึก (Depth First Traversal) การทำงานคล้ายกับการท่องทีละระดับของทรี โดยกำหนดเริ่มต้นที่โหนดแรกและเยือนโหนดถัดไปตามแนววิถีนั้นจนกระทั่งนำไปสู่ปลายวิถีนั้น จากนั้นย้อนกลับ (backtrack) ตามแนววิถีเดิมนั้น จนกระทั่งสามารถดำเนินการต่อเนื่องเข้าสู่แนววิถีอื่น ๆ เพื่อเยือนโหนดอื่น ๆ ต่อไปจนครบทุกโหนด

Graph (Cont.)



ผลลัพธ์จาการท่องไปในกราฟ 1 9 6 7 8 5 2 3 4

แบบฝึกหัด

1. อธิบายโครงสร้างรูปแบบ กระบวนการทำงานของ กราฟ
2. กราฟ แบบ Directed แตกต่างจาก Undirected หรือไม่อย่างไร
3. ให้อธิบายประโยชน์จากการนำกราฟไปประยุกต์ใช้ในชีวิตประจำวัน หรืองานที่เกิดขึ้น