ขั้นตอนวิธีการหาวิถีที่สั้นที่สุดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้

Shortest path Algorithm with GUI

นายรัตนโชติ พันธุ์วิไล รหัส 47380357

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน **ขั้นตอนวิธีการหาวิถีที่สั้นที่สุดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้**

ผู้ดำเนินโครงงาน นายรัตนโชติ พันธุ์วิไล รหัส 47380357

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.ธนิต มาลากร

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2550

....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ธนิต มาลากร)

....กรรมการ
(ดร.พนมขวัญ ริยะมงคล)

....กรรมการ
(ดร.ไพศาล มุณีสว่าง)

หัวข้อโครงงาน	ขั้นตอนวิธีการหาวิถีที่สั้นที่สุดด้วยส่วนต่อประสานกราฟิกกับผู้ใช้
ผู้ดำเนินโครงงาน	นายรัตนโชติ พันธุ์วิไล รหัส 47380357
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.ธนิต มาลากร
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

บทคัดย่อ

ส่วนต่อประสานกราฟิกกับผู้ใช้ คือส่วนต่อประสานเชิงรูปภาพในโปรแกรมแมทแลปซึ่งมีสิ่งแวดล้อมที่ผู้ใช้คุ้นเคย อันได้แก่ ปุ่มกด ปุ่มเลือก กล่องข้อความ ตัวเลื่อน เป็นต้น โครงงานนี้มุ่งเน้นในการพัฒนาโปรแกรมส่วนต่อประสานกราฟิกกับผู้ใช้เพื่อช่วยในการคำนวณในปัญหาการหาวิถีที่สั้นที่สุด—ปัญหาในการหาวิถีระหว่างสองบัพในกราฟถ่วงน้ำหนักที่ซึ่งผลรวมของค่าน้ำหนักถ่วงบนเส้นเชื่อมทั้งหมดในวิถีนั้นมีค่าน้อยที่สุด มีหลายขั้นตอนวิธีที่ถูกนำเสนอเพื่อใช้แก้ปัญหาดังกล่าว ได้แก่ ขั้นตอนวิธีของ Dijkstra ขั้นตอนวิธีการค้นหา A* ขั้นตอนวิธีของ Bellman-Ford และขั้นตอนวิธีของ Floyd-Warshall เป็นต้น แต่มีเพียงสองขั้นตอนวิธีที่นำเสนอในโครงงานนี้ นั่นคือ ขั้นตอนวิธีของ Dijkstra และขั้นตอนวิธีของ Floyd-Warshall ผลการคำนวณเชิงตัวเลขแสดงให้เห็นว่าวิถีที่สั้นที่สุดที่ถูกคำนวณด้วยขั้นตอนวิธีทั้งคู่มีค่าเท่ากัน

Project Title	Shortest Path Algorithm with GUI
Name	Mr. Rattanachot Phwanwilai ID. 47380357
Project Advisor	Asst. Prof. Tanit Malakorn, PhD
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic Year	2007

....

ABSTRACT

A graphical user interface (GUI) is a pictorial interface to a MATLAB program. It provides the user with a familiar environment in which to work. This environment contains pushbuttons, toggle buttons, text boxes, sliders, and so forth, all of which are already familiar to the user. This project primarily concerns with the development of the MATLAB GUI-based program for the computational aids in the shortest path problem—the problem of finding a path between two vertices in a weighted graph so that the sum of the weights of its constituent edges is minimized. Several algorithms have been proposed for solving such a problem, e.g., Dijkstra's algorithm, A* search algorithm, Bellman-Ford algorithm, and Floyd-Warshall algorithm; only two of which are presented in this project, namely Dijkstra's and Floyd-Warshall algorithms. The numerical examples show that the shortest paths computed by two algorithms are identical.

กิตติกรรมประกาศ

โครงงานฉบับนี้สำเร็จได้ ด้วยความกรุณาของอาจารย์ที่ปรึกษาโครงงาน ผู้ช่วยศาสตราจารย์ ดร.ธนิต มาลากร ซึ่งนอกจากให้คำปรึกษาด้านทฤษฎีกราฟและขั้นตอนวิธีต่าง ๆ ที่เกี่ยวข้องกับโครงงาน ตลอดจนการพิสูจน์ทฤษฎีบทต่าง ๆ ท่านยังแนะแนวทางในการสืบค้นเอกสารอ้างอิง และถ่ายทอดความรู้ในการใช้งาน Matlab 7 ด้วยความจริงใจโดยไม่ปิดบังซ่อนเร้น โดยเริ่มแรกข้าพเจ้าไม่มีความรู้เกี่ยวกับโปรแกรม Matlab 7 แต่หลังจากที่เริ่มศึกษาและทำโครงงานนี้ ทำให้ข้าพเจ้าเริ่มเข้าใจในหลักการเขียนโปรแกรม Matlab 7 โดยเฉพาะอย่างยิ่ง หลักการเขียนส่วนประสานกราฟฟิก (Graphic User Interface: GUI) ในโปรแกรม Matlab 7 นอกจากนี้ ท่านยังมีเวลาให้ข้าพเจ้าในการเข้าพบเพื่อขอรับคำปรึกษาโดยเสมอมา ทำให้โครงงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

ผู้จัดทำใคร่ขอกราบขอบพระคุณ ดร.พนมขวัญ ริยะมงคล และดร.ไพศาล มุณีสว่าง ซึ่งได้สละเวลามาเป็นกรรมการในการสอบ รวมทั้งตรวจสอบข้อบกพร่องต่าง ๆ ของโครงงานฉบับนี้ นอกจากนี้ ผู้จัดทำขอขอบคุณและขอบใจ รุ่นพี่ และเพื่อน ๆ ทุกคนที่ให้กำลังใจ และช่วยตรวจหาข้อผิดพลาดของโปรแกรม ช่วยกระตุ้นให้ข้าพเจ้ามุมานะทำโครงงานนี้ ช่วยเป็นที่ปรึกษาในเรื่องที่ไม่สบายใจที่เกี่ยวกับโครงงาน ตลอดจนความช่วยเหลือด้านการทำรายงาน จนทำให้โครงงานนี้สำเร็จลุล่วงไปได้

สุดท้ายนี้ ผู้จัดทำใคร่ขอกราบขอบพระคุณพ่อ และคุณแม่ บุพการีผู้ให้ทุกสิ่งทุกอย่าง รวมทั้งให้กำลังใจมาโดยตลอด

ผู้จัดทำโครงงาน

รัตนโชติ พันธุ์วิไล

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ก

บทคัดย่อภาษาอังกฤษ ..... ข

กิตติกรรมประกาศ ..... ค

สารบัญ ..... ง

สารบัญตาราง ..... ฉ

สารบัญรูป ..... ช

บทที่ 1 1

1.1 หลักการและเหตุผล 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 ขอบข่ายของโครงงาน 2

1.4 ขั้นตอนการดำเนินงาน 2

1.5 ผลที่คาดว่าจะได้รับ 2

1.6 งบประมาณ 3

บทที่ 2 4

2.1 กราฟ หรือกราฟไม่ระบุทิศทาง (Graphs or Undirected Graphs) 5

2.2 ไดกราฟ หรือกราฟระบุทิศทาง (Digraphs or Directed Graphs) 7

2.3 แนวเดิน ทางเดิน และวิถี 8

2.3.1 แนวเดิน รอยเดิน และ วิถี ในกราฟ (Walks, trails, and paths in graph) 9

2.3.2 แนวเดิน รอยเดิน และ วิถี ในไดกราฟ (Walks, trails, and paths in digraph) 10

2.4 การแทนกราฟด้วยเมทริกซ์ 10

บทที่ 3 14

3.1 ขั้นตอนวิธีของ Dijkstra 15

ตัวอย่างที่ 1 17

ตัวอย่างที่ 2 20

3.2 ขั้นตอนวิธีของ Floyd-Warshall 23

ตัวอย่างที่ 3 25

ตัวอย่างที่ 4 28

สารบัญ (ต่อ)

หน้า

บทที่ 4 32

การใช้โปรแกรม 32

บทที่ 5 52

5.1 สรุปการทดสอบขั้นตอนวิธีของ Dijkstra และ ขั้นตอนวิธีของ Floyd-Warshall 52

5.2 ปัญหาและอุปสรรค 52

5.3 ข้อเสนอแนะและแนวทางการพัฒนา 53

เอกสารอ้างอิง 54

ประวัติผู้เขียนโครงงาน 55

สารบัญตาราง

หน้า

ตารางที่1 การคำนวณหาวิถีที่สั้นที่สุดโดยขั้นตอนวิธี Dijkstra 16
ตารางที่2 การคำนวณหาวิถีที่สั้นที่สุดโดยขั้นตอนวิธี Floyd-Warshall 24

สารบัญรูป

หน้า

รูปที่ 2.1 แสดงรูปวาดของสะพานเคอนิกส์เบิร์ก 4
รูปที่ 2.2 แสดงตัวอย่างของกราฟ 5
รูปที่ 2.3 แสดงกราฟหลายเชิง (Multiple graph) 5
รูปที่ 2.4 แสดงกราฟเชิงเดียว 6
รูปที่ 2.5 แสดงกราฟเชิงเดียวถ่วงน้ำหนัก 6
รูปที่ 2.6 แสดงตัวอย่างของไดกราฟ 7
รูปที่ 2.7 แสดงไดกราฟที่มีแหล่งต้นทางและแหล่งปลายทาง 8
รูปที่ 2.8 แสดงตัวอย่างของแนวเดินประเภทต่าง ๆ ในกราฟ 9
รูปที่ 2.9 แสดงตัวอย่างของแนวเดินประเภทต่าง ๆ ในไดกราฟ 10
รูปที่ 2.10 การแปลงเมทริกซ์ให้อยู่ในรูปของกราฟ 12
รูปที่ 3.1 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra 17
รูปที่ 3.2 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra ขั้นตอนที่ 2 17
รูปที่ 3.3 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra ขั้นตอนที่ 3 18
รูปที่ 3.4 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra 20
รูปที่ 3.5 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra ขั้นตอนที่ 2 20
รูปที่ 3.6 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธี Dijkstra ขั้นตอนที่ 3 21
รูปที่ 3.7 แสดงตัวอย่างการคำนวณในขั้นตอนวิธีของ Floyd-Warshall 23
รูปที่ 3.8 กราฟตัวอย่างสำหรับการคำนวณในวิธีของ Floyd-Warshall 25
รูปที่ 3.9 ไดกราฟตัวอย่างสำหรับการคำนวณในวิธีของ Floyd-Warshall 28

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

การหาวิถีที่สั้นที่สุด (Shortest paths) เป็นตัวอย่างหนึ่งของปัญหาค่าเหมาะที่สุด (Optimization problem) ซึ่งมีขั้นตอนวิธีในการคำนวณหลายวิธี เช่น ขั้นตอนวิธีของ Dijkstra (Dijkstra's Algorithm) ขั้นตอนวิธีของ Floyd (Floyd's Algorithm) ขั้นตอนวิธีของ Bellman และ Ford (Bellman-Ford's algorithm) และขั้นตอนวิธีของ Johnson (Johnson's algorithm) เป็นต้น ซึ่งแต่ละขั้นตอนวิธีนั้น เหมาะสมกับปัญหาที่แตกต่างกันออกไป แต่อย่างไรก็ตาม ไม่ว่าจะเลือกขั้นตอนวิธีใดในการแก้ปัญหานั้น จำเป็นต้องอาศัยการเขียนโปรแกรมคอมพิวเตอร์ในการคำนวณเพื่อทำให้เกิดผลลัพธ์ที่ต้องการ

ในยุคแรกของการใช้งานคอมพิวเตอร์ พบว่าโดยทั่วไป การออกแบบและพัฒนาโปรแกรมยังมีประสิทธิภาพไม่ดีเท่าที่ควร ผู้ที่จะใช้งานโปรแกรมคอมพิวเตอร์ได้ต้องมีองค์ความรู้พื้นฐานด้านคอมพิวเตอร์เป็นอย่างมาก ในการพัฒนาโปรแกรมแต่ละครั้ง ผู้ใช้งานจะต้องจดจำคำสั่งรวมทั้งโครงสร้างของโปรแกรมต่างๆ ซึ่งก่อให้เกิดความยุ่งยากในการใช้งาน ต่อมาได้มีการพัฒนาโปรแกรมที่เรียกว่า ส่วนต่อประสานกราฟิกกับผู้ใช้ หรือ Graphic User Interface (GUI) ทำให้ผู้ใช้งานทั่วไปสามารถนำโปรแกรมที่ได้ไปใช้งานได้อย่างสะดวกและรวดเร็ว โดยส่วนต่อประสานกราฟิกกับผู้ใช้นี้อาศัยสัญลักษณ์ (icon) แทนการเขียนคำสั่งต่างๆ ซึ่งผู้ใช้งานทั่วไปสามารถใช้เมาส์คลิกเลือกสัญลักษณ์ต่าง ๆ หรือพิมพ์ข้อความลงในกล่องข้อความ หรือกดปุ่มเลือก แทนการเขียนคำสั่งโดยตรงในโปรแกรม โปรแกรมที่มีการพัฒนารูปแบบของ GUI มาใช้งานได้แก่ โปรแกรม visual basic2005 โปรแกรม Autocad โปรแกรม Matlab เป็นต้น

โครงงานฉบับนี้มุ่งเน้นศึกษาการพัฒนาส่วนประสานกราฟิกในโปรแกรม Matlab เพื่ออำนวยความสะดวกกับผู้ใช้งานทั่วไปในการคำนวณหาวิถีที่สั้นที่สุด

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาขั้นตอนวิธีประเภทต่าง ๆ ในการคำนวณหาวิถีที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดปลายทาง
2. ออกแบบและพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ในโปรแกรม Matlab เพื่ออำนวยความสะดวกแก่ผู้ใช้งานทั่วไปในการคำนวณหาวิถีที่สั้นที่สุด

1.3 ขอบข่ายของโครงงาน

1. ศึกษาขั้นตอนวิธีประเภทต่างๆ ที่ใช้ในการคำนวณหาวิถีที่สั้นที่สุดพร้อมทั้งระบุเวลาในการใช้ในการประมวลผลของแต่ละขั้นตอนวิธีได้
2. ออกแบบและพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ในโปรแกรม Matlab version 7.0
3. ผู้ใช้งานสามารถโหลดแผนที่หรือรูปแผนที่ที่ต้องการจากไฟล์รูปภาพนามสกุล bmp, jpg, หรือ jpeg ได้
4. ผู้ใช้งานสามารถกำหนดจุดต่างๆ บนแผนที่ได้ไม่เกิน 20 จุด
5. โปรแกรมสามารถคำนวณหาวิถีที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดปลายทางได้

1.4 ขั้นตอนการดำเนินงาน

กิจกรรม	ระยะเวลาดำเนินงาน								
	พ.ย. - ธ.ค. 49	ม.ค. - ก.พ. 50	มี.ค. - เม.ย. 50	พ.ค. - มิ.ย. 50	ก.ค. - ส.ค. 50	ก.ย. - ต.ค. 50	พ.ย. - ธ.ค. 50	ม.ค. - ก.พ. 51	มี.ค. - เม.ย. 51
1.ศึกษาขั้นตอนวิธี	↔	↔							
2.นำขั้นตอนวิธีมาประยุกต์ใช้			↔	↔					
3.ศึกษาการใช้ Matlab GUI					↔				
4.ออกแบบโปรแกรม						↔	↔		
5.จัดทำรายงานฉบับสมบูรณ์								↔	↔

1.5 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ของขั้นตอนวิธีต่างๆที่ใช้ในการคำนวณหาวิถีที่สั้นที่สุด
2. ได้รับความรู้ทางการพัฒนาส่วนต่อประสานกราฟิกกับผู้ใช้ในโปรแกรม Matlab
3. สามารถประยุกต์และนำโปรแกรมมาแก้ปัญหาต่างๆ ได้อย่างเหมาะสม
4. นำโปรแกรมที่พัฒนาขึ้นไปใช้ในการแก้ปัญหาที่พบเห็นในชีวิตประจำวันได้

1.6 งบประมาณ

1. ค่าวัสดุคอมพิวเตอร์	500	บาท
2. ค่าวัสดุสำนักงาน	450	บาท
3. ค่าถ่ายเอกสาร	150	บาท
รวมเป็นเงิน (หนึ่งพันบาทถ้วน)	1,000	บาท

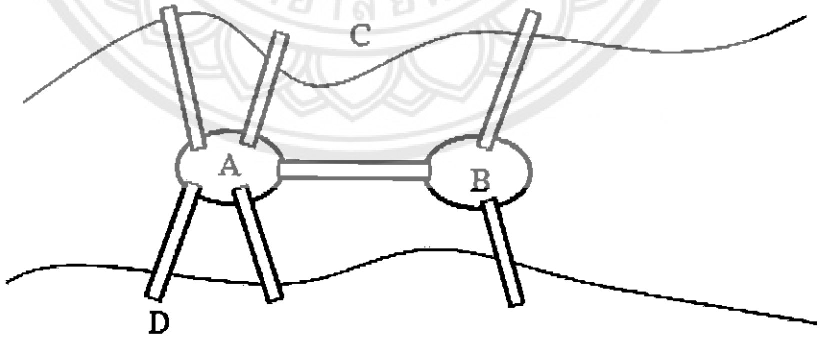
หมายเหตุ ถัวเฉลี่ยทุกรายการ

บทที่ 2

ทฤษฎีกราฟ

(Graph Theory)

ทฤษฎีกราฟมีประวัติความเป็นมาตั้งแต่ปี ค.ศ. 1736 จากผลงานของนักคณิตศาสตร์ชาวสวิส ชื่อ เลียวนาร์ด ออยเลอร์ (Leonhard Euler) ในการแก้ปัญหาหาวิธีการเดินข้ามสะพานที่เมืองเคอนิกส์เบิร์ก (The Konigsberg Bridges Problem) ที่เมืองแห่งนี้ มีแม่น้ำสายหนึ่งชื่อว่า Pregel ไหลผ่านและมีเกาะเล็ก ๆ สองเกาะอยู่กลางแม่น้ำ โดยมีสะพาน 7 แห่งเชื่อมต่อระหว่างเกาะ ดังแสดงในรูปที่ 2.1 จากสะพานที่เชื่อมต่อทั้ง 7 นี้ จึงเกิดคำถามขึ้นมาว่า มีความเป็นไปได้หรือไม่ที่เริ่มเดินจากจุดใดจุดหนึ่ง ข้ามสะพานทั้ง 7 แห่ง แห่งละ 1 เที่ยวแล้ววกกลับมาที่จุดตั้งต้นได้ คำตอบของปัญหาดังกล่าวนี้ถูกค้นพบ โดย ออยเลอร์ ซึ่งเขาได้ตีพิมพ์คำตอบของปัญหาดังกล่าวว่า เป็นไปไม่ได้ โดยออยเลอร์ เขียนบรรยายปัญหาดังกล่าวออกมาในรูปของกราฟ ซึ่งจัดว่าปัญหาการหาวิธีการเดินข้ามสะพานที่เมืองเคอนิกส์เบิร์กนี้เป็นต้นกำเนิดของการพัฒนาทฤษฎีกราฟ ต่อมาทฤษฎีกราฟนี้ได้รับความสนใจเป็นอย่างมาก มีการพัฒนาทฤษฎีต่าง ๆ ขึ้นอย่างมากมายและสามารถนำไปประยุกต์เพื่อแก้ปัญหาต่างๆ ในสาขาวิชาอื่นๆ ได้ เช่น ฟิสิกส์ เคมี ชีววิทยา วิศวกรรม เศรษฐศาสตร์ สังคมศาสตร์ เป็นต้น

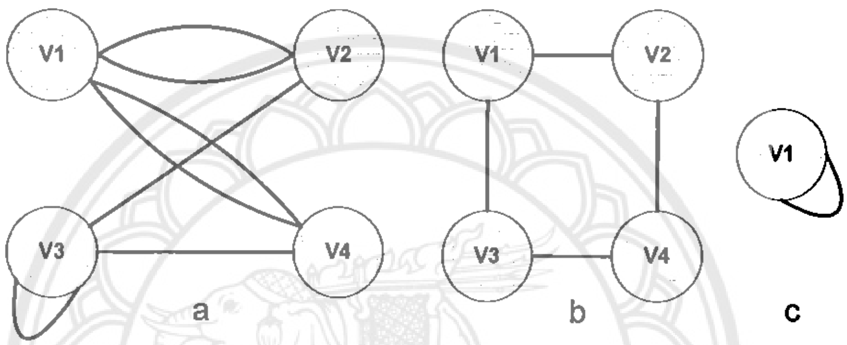


รูปที่ 2.1 แสดงรูปวาดของสะพานเคอนิกส์เบิร์ก

2.1 กราฟ หรือกราฟไม่ระบุทิศทาง (Graphs or Undirected Graphs)

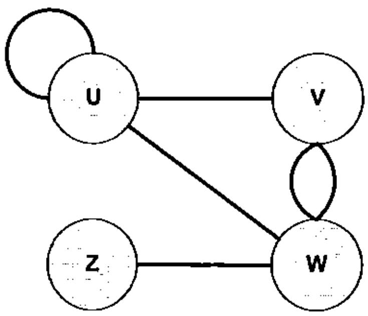
กราฟ (บางครั้งอาจเน้นว่า กราฟไม่ระบุทิศทาง หรือ Undirected graph) G := G(V,E) ประกอบด้วยเซต $V \neq \varnothing$ ซึ่งเป็นเซตของจุดที่เรียกว่า **บัพ (Nodes)** หรือ**จุดยอด (Vertices)** และเซต E ซึ่งเป็นเซตของเส้นเชื่อม **(Edges)**[1] $e_{ij} = \left(v_i, v_j\right)$ โดยที่ v_i, v_j เป็นสมาชิกในเซต V หากกราฟมีจำนวนเชิงการนับ (Cardinality) ของเซต V จำกัด กราฟดังกล่าวถูกเรียกว่า **กราฟจำกัด หรือ Finite graph** ซึ่งจากนิยามข้างต้น กราฟจำกัดสามารถสรุปได้ดังนี้

G:= G(V,E) โดยที่ $V = \{v_i\}_{i=1}^{n} \neq \varnothing$ และ $E: V \times V \to (V,V) = \left\{e_{ij} \middle| \text{for some } v_i, v_j \in V\right\}$



รูปที่ 2.2 แสดงตัวอย่างของกราฟ

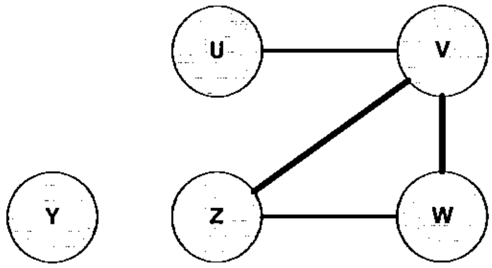
หาก $e_{ij} = \left(v_i, v_j\right) \in E(G)$ แล้ว v_i, v_j ถูกกล่าวว่าเป็น **จุดปลาย (Ends)** ของ e_{ij} และ**บัพ** v_i, v_j ถูกกล่าวว่าเป็น**บัพประชิดกัน (Adjacent nodes)** เส้นเชื่อมที่ซึ่งมีจุดปลายเป็นจุดเดียวกัน นั่นคือ $e_{ii} = \left(v_i, v_i\right)$ ถูกกล่าวว่าเป็น **วงวน (Loops)** หากมีเส้นเชื่อมมากกว่าหนึ่งเส้นที่เชื่อมต่อระหว่างบัพ v_i, v_j นั่นคือมี $e_{ij,k} = \left(v_i, v_j\right)$ โดยที่ $k > 1$ แล้วเส้นเชื่อมเหล่านั้นถูกกล่าวว่าเป็น **เส้นเชื่อมขนานกัน (Parallel edges)** กราฟถูกกล่าวว่าเป็น **กราฟเชิงเดียว (Simple graph)** หากไม่มีเส้นเชื่อมแบบวงวน หรือ เส้นเชื่อมขนานกันในกราฟนั้น กราฟที่ไม่เป็นกราฟเชิงเดียว เรียกว่า **กราฟหลายเชิง (Multiple graph)**



รูปที่ 2.3 แสดงกราฟหลายเชิง (Multiple graph)

[1] เซต E สามารถเป็นเซตว่างได้ หากบัพแต่ละบัพในกราฟไม่มีเส้นเชื่อมระหว่างกัน

กราฟในรูปที่ 2.3 เป็นกราฟหลายเชิง ซึ่งเห็นได้ว่ามีวงวนที่บัพ u เนื่องจากที่บัพดังกล่าวมีเส้นเชื่อม e = (u,u) นอกจากนี้ พบว่ามีเส้นเชื่อมสองเส้นระหว่างบัพ v และบัพ w ดังนั้นจึงเกิดเส้นเชื่อมขนานขึ้นในกราฟ ในขณะที่กราฟในรูปที่ 2.4 คือกราฟเชิงเดียว

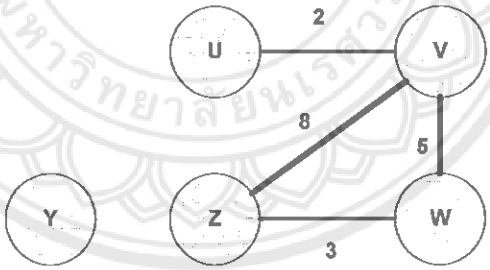


รูปที่ 2.4 แสดงกราฟเชิงเดียว

กราฟเชิงเดียวที่มีการระบุค่าจริง w_{ij} ให้กับแต่ละเส้นเชื่อม e_{ij} ถูกเรียกว่า **กราฟถ่วงน้ำหนัก (Weighted graph)** หรือเขียนแทนด้วย

$$G := G(V,E,w) \text{ โดยที่ } w : E \to \mathbb{R}$$

ที่ซึ่ง w_{ij} ถูกเรียกว่า น้ำหนักถ่วง (Weight) ของเส้นเชื่อม โดยทั่วไป น้ำหนักถ่วงมักเป็นจำนวนจริงบวก แต่ในการประยุกต์บางอย่าง อาจจะใช้น้ำหนักถ่วงเป็นจำนวนจริงได้ ตัวอย่างของกราฟถ่วงน้ำหนัก แสดงไว้ในรูปที่ 2.5



รูปที่ 2.5 แสดงกราฟเชิงเดียวถ่วงน้ำหนัก

ให้ v เป็นบัพใด ๆ ในเซต V(G) ระดับขั้นบัพ (Degree of a node) คือ จำนวนของเส้นเชื่อมที่ต่อออกจากบัพนั้น และเขียนแทนด้วยสัญลักษณ์ deg(v) บัพใดที่มีระดับขั้นเป็นศูนย์ บัพนั้นถูกกล่าวว่าเป็น บัพเอกเทศ (Isolated node) ดังนั้น บัพ Y ในรูปที่ 2.5 เป็นบัพเอกเทศเนื่องจากที่บัพ Y ไม่มีเส้นเชื่อมใด ๆ ซึ่งจะได้ว่า deg(Y) = 0 และที่บัพ Z มีเส้นเชื่อมที่ต่อออกมา 2 เส้น ดังนั้น deg(Z) = 2

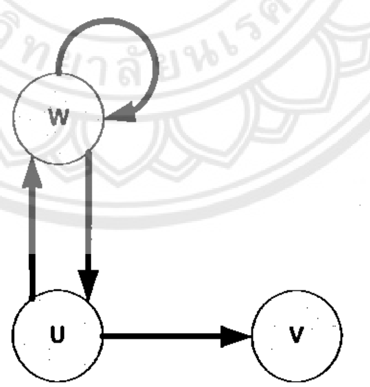
2.2 ไดกราฟ หรือกราฟระบุทิศทาง (Digraphs or Directed Graphs)

ในหัวข้อนี้ มุ่งพิจารณากราฟที่ซึ่งเส้นเชื่อมระหว่างบัพมีการระบุทิศทาง กล่าวคือ หากในกราฟมีเส้นเชื่อม (u,v) ไม่จำเป็นที่ในกราฟนั้นต้องมีเส้นเชื่อม (v,u) ซึ่งกราฟที่มีทิศทางระบุจากบัพหนึ่งไปยังอีกบัพหนึ่งเช่นนี้ เรียกว่า กราฟระบุทิศทาง หรือ ไดกราฟ ซึ่งมีนิยามดังต่อไปนี้

ไดกราฟ (บางครั้งเรียกว่า กราฟระบุทิศทาง หรือ Directed graph) $G := G(V,A)$ ประกอบด้วยเซต $V \neq \varnothing$ ซึ่งเป็นเซตของจุดที่เรียกว่า **บัพ (Nodes) หรือจุดยอด (Vertices)** และเซต A ซึ่งเป็นเซตของคู่ลำดับ (v_i, v_j) โดยที่ v_i, v_j เป็นสมาชิกในเซต V ซึ่งสมาชิกของเซต A เรียกว่า **ส่วนโค้ง (Arcs)** หรือ **เส้นเชื่อมระบุทิศทาง (Directed edges)** และใช้สัญลักษณ์แทนด้วย $a_{ij} = (v_i, v_j)$ ไดกราฟสามารถมีวงวนได้ แต่ไม่สามารถมีส่วนโค้งแบบขนาน **กล่าวคือ จะมีส่วนโค้งเชื่อม**ระหว่างบัพสองบัพ(ที่แตกต่างกัน)ได้ไม่เกิน 1 เส้นในไดกราฟ หากเซต **V มีจำนวนเชิงการนับ** (Cardinaltiy) จำกัด ไดกราฟดังกล่าวถูกเรียกว่า ไดกราฟจำกัด หรือ **Finite digraph** ซึ่งจาก**นิยามข้าง**ต้น ไดกราฟจำกัดสามารถสรุปในเชิงคณิตศาสตร์ได้ดังนี้

$G := G(V,A)$ โดยที่ $V = \{v_i\}_{i=1}^{n} \neq \varnothing$ และ $A : V \times V \to (V,V) = \{a_{ij} \mid \text{for some } v_i, v_j \in V\}$

หากมีส่วนโค้ง $a_{ij} = (v_i, v_j) \in A(G)$ แล้ว v_i, v_j ถูกกล่าวว่าเป็น **ส่วนหาง (Tail)** และ **ส่วนหัว** (Head) ของ a_{ij} ตามลำดับ และบัพ v_i ถูกกล่าวว่าเป็น **บัพประชิดกัน (Adjacent nodes)** กับบัพ v_j ของส่วนโค้ง a_{ij} แต่บัพ v_j ไม่ใช่บัพประชิดกันกับบัพ v_i นอกจากมีส่วนโค้ง $(v_j, v_i) \in A(G)$



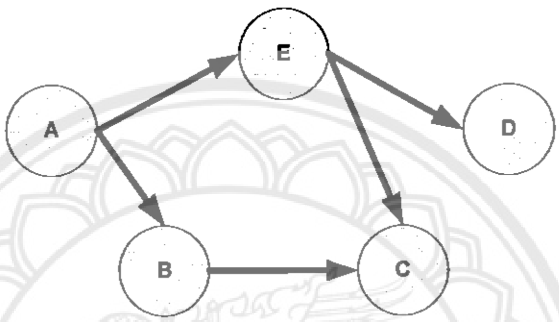
รูปที่ 2.6 แสดงตัวอย่างของไดกราฟ

กราฟที่แสดงไว้ในรูปที่ 2.6 คือ ไดกราฟที่ซึ่ง u เป็นส่วนหางและ v เป็นส่วนหัวของส่วนโค้ง (u,v) นอกจากนี้ยังพบอีกว่า บัพ u เป็นบัพประชิดกับบัพ v เนื่องจากมีส่วนโค้ง (u,v) ในไดกราฟ แต่อย่างไรก็ตาม บัพ v ไม่เป็นบัพประชิดกับบัพ u เพราะไม่มีส่วนโค้ง (v,u) ในไดกราฟ นอกจากนี้ บัพ u และบัพ w เป็นบัพประชิดซึ่งกันและกัน เนื่องจากมีส่วนโค้ง (w,u) และส่วนโค้ง (u,w) ในไดกราฟ และไดกราฟนี้มีวงวนเกิดขึ้นที่บัพ w

ให้ v เป็นบัพใด ๆ ของไดกราฟ G **ระดับขั้นเข้า (In-degree)** ของ v คือ จำนวนของส่วนโค้งที่มี v เป็นส่วนหัว และใช้สัญลักษณ์แทนด้วย indeg(v) ในขณะที่จำนวนของส่วนโค้งที่มี v เป็นส่วนหาง เรียกว่า **ระดับขั้นออก (Out-degree)** ซึ่งเขียนแทนด้วย outdeg(v) **ระดับขั้น (Degree)** ของบัพ v ใช้แทนด้วย deg(v) คือผลรวมของระดับขั้นเข้า และระดับขั้นออก นั่นคือ

$$deg(v) = indeg(v) + outdeg(v)$$

บัพใดที่มีระดับขั้นเข้าเป็นศูนย์ บัพนั้นถูกกล่าวว่าเป็น **แหล่งต้นทาง (Source)** ส่วนบัพใดที่มีระดับขั้นออกเป็นศูนย์ เรียกว่า **แหล่งปลายทาง (Sink)**



รูปที่ 2.7 แสดงไดกราฟที่มีแหล่งต้นทางและแหล่งปลายทาง

พิจารณาไดกราฟในรูปที่ 2.7 พบว่า ที่บัพ E มีระดับขั้น 3 เนื่องจาก indeg(E) = 1 และ outdeg(E) = 2 สำหรับบัพ A เป็นแหล่งต้นทาง เนื่องจาก indeg(A) = 0 ในขณะที่บัพ C และ D เป็นแหล่งปลายทาง เพราะว่า outdeg(C) = 0 = outdeg(D)

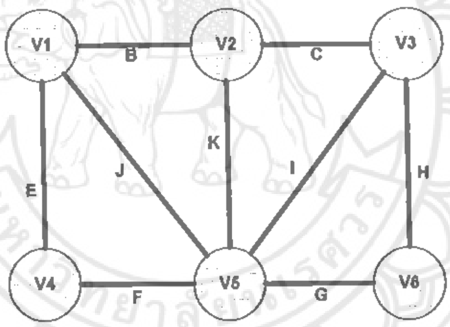
2.3 แนวเดิน ทางเดิน และวิถี

จากที่ ออยเลอร์ ได้ใช้กราฟเป็นแบบจำลองในการแก้ปัญหาการเดินทางข้ามสะพานที่เมืองเคอนิกส์เบิร์กที่กล่าวไว้ในตอนต้น ซึ่งปัญหาดังกล่าวนี้สามารถบรรยายในรูปแบบของปัญหาเชิงกราฟได้ดังนี้ "เป็นไปได้หรือไม่ที่สามารถหาลำดับที่สลับกันระหว่างบัพและเส้นเชื่อมของกราฟ โดยที่เส้นเชื่อม<u>ทุกเส้น</u>ในกราฟนั้นต้องถูกนำมาเรียงในลำดับดังกล่าวและต้องถูกนำมาเรียงเพียงครั้งเดียวเท่านั้น (แม้ว่าบัพอาจจะถูกนำมาเรียงเพียงครั้งเดียวหรือหลายครั้งได้) และท้ายสุด ลำดับจะต้องวนกลับมาสู่บัพเริ่มต้น" หากลำดับลักษณะดังกล่าวเกิดขึ้นจริงในกราฟเชิงเดียวใด ๆ ลำดับประเภทนี้ถูกเรียกว่า **วงจรแห่งออยเลอร์ (Eulerian circuit)** เพื่อให้เป็นเกียรติกับ ออยเลอร์ อย่างไรก็ตาม ออยเลอร์ได้พิสูจน์ว่า วงจรออยเลอร์ ไม่สามารถเกิดได้ในปัญหาการเดินทางข้ามสะพานดังกล่าว ดังนั้น จึงเป็นไปไม่ได้ที่จะเดินจุดใดจุดหนึ่ง ข้ามสะพานทั้ง 7 แห่ง แห่งละ 1 เที่ยวแล้ววกกลับมาที่จุดตั้งต้นได้

2.3.1 แนวเดิน รอยเดิน และ วิถี ในกราฟ (Walks, trails, and paths in graph)

ให้ G(V,E) เป็นกราฟ **แนวเดิน (Walk: W)** ในกราฟ G คือ ลำดับสลับกันระหว่างบัพและเส้นเชื่อมโดยขึ้นต้นและลงท้ายของลำดับด้วยบัพ และเส้นเชื่อมที่คั่นกลาง คือเส้นที่เชื่อมระหว่างบัพที่ประชิดกัน นั่นคือ $W = v_1e_{12}v_2e_{23}v_3 \ldots v_{n-1}e_{n-1,n}v_n$ เมื่อ $n > 0$ ถ้าจุดเริ่มต้นและลงท้ายของลำดับคือบัพเดียวกัน แนวเดิน W นั้นถูกกล่าวว่าเป็น **แนวเดินปิด (Closed walk)** ความยาวของแนวเดินคือจำนวนของเส้นเชื่อมในแนวเดินนั้น แนวเดินที่มีความยาวเป็นศูนย์เรียกว่า **แนวเดินชัด (Trivial walk)** นั่นคือ $W = v_k$

รอยเดิน (Trail: Tr) ในกราฟ G คือ แนวเดินที่ไม่มีเส้นเชื่อมที่ซ้ำกัน และ**วิถี (Path: P)** คือแนวเดินที่ไม่มีบัพที่ซ้ำกัน **วงจร (Circuit)** คือรอยเดินปิด **วงจรชัด (Trivial circuit)** คือวงจรที่มีเพียงบัพเดียวไม่มีเส้นเชื่อม รอยเดิน (หรือวงจร) ถูกกล่าวว่าเป็น **รอยเดินแห่งออยเลอร์ (Eulerian trail)** (หรือ**วงจรแห่งออยเลอร์ (Eulerian circuit)**) หากทุก ๆ เส้นเชื่อมในกราฟนั้นอยู่ในรอยเดิน (หรือวงจร) **วง (Cycle)** คือวงจรที่ไม่ชัด (Nontrivial circuit) ที่ซึ่งมีบัพที่ซ้ำกันเพียงบัพเดียว นั่นคือบัพแรกและบัพสุดท้าย



รูปที่ 2.8 แสดงตัวอย่างของแนวเดินประเภทต่าง ๆ ในกราฟ

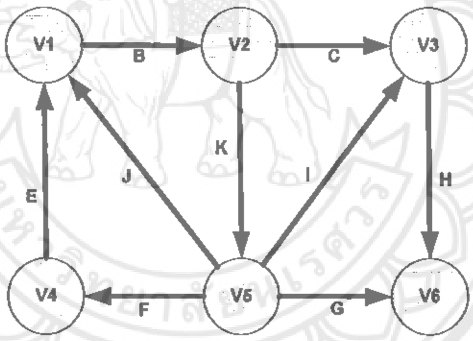
พิจารณากราฟในรูปที่ 2.8 และพิจารณาแนวเดินต่อไปนี้

1. $W^{(1)} = v_4\text{-}E\text{-}v_1\text{-}B\text{-}v_2\text{-}K\text{-}v_5\text{-}J\text{-}v_1\text{-}B\text{-}v_2\text{-}C\text{-}v_3\text{-}H\text{-}v_6$ เป็นแนวเดิน
2. $W^{(2)} = v_4\text{-}E\text{-}v_1\text{-}J\text{-}v_5\text{-}K\text{-}v_2\text{-}C\text{-}v_3\text{-}I\text{-}v_5\text{-}G\text{-}v_6$ เป็นรอยเดิน
3. $W^{(3)} = v_4\text{-}E\text{-}v_1\text{-}J\text{-}v_5\text{-}K\text{-}v_2\text{-}C\text{-}v_3\text{-}I\text{-}v_5\text{-}F\text{-}v_4$ เป็นรอยเดินปิด หรือ วงจร
4. $W^{(4)} = v_4\text{-}E\text{-}v_1\text{-}B\text{-}v_2\text{-}C\text{-}v_3\text{-}I\text{-}v_5\text{-}G\text{-}v_6$ เป็นวิถี
5. $W^{(5)} = v_4\text{-}F\text{-}v_5\text{-}I\text{-}v_3\text{-}C\text{-}v_2\text{-}B\text{-}v_1\text{-}E\text{-}v_4$ เป็นวิถีปิด หรือ วง

2.3.2 แนวเดิน รอยเดิน และ วิถี ในไดกราฟ (Walks, trails, and paths in digraph)

ให้ G(V,A) เป็นไดกราฟ **แนวเดิน (Walk: $\vec{W}$)** ในไดกราฟ G คือ ลำดับสลับกันระหว่างบัพและส่วนโค้งที่เชื่อมระหว่างบัพส่วนหางไปยังบัพส่วนหัวของบัพที่ประชิดกัน โดยขึ้นต้นและลงท้ายของลำดับด้วยบัพ นั่นคือ $\vec{W} = v_1a_{12}v_2a_{23}v_3 \ldots v_{n-1}a_{n-1,n}v_n$ เมื่อ $n > 0$ ถ้าจุดเริ่มต้นและลงท้ายคือบัพเดียวกัน แนวเดิน $\vec{W}$ ถูกกล่าวว่าเป็น **แนวเดินปิด (Closed walk)** ความยาวของแนวเดินคือจำนวนของส่วนโค้งในแนวเดินนั้น แนวเดินที่มีความยาวเป็นศูนย์เรียกว่า **แนวเดินชัด (Trivial walk)** นั่นคือ $\vec{W} = v_k$

รอยเดิน (Trail: Tr) ในไดกราฟ G คือ แนวเดินที่ไม่มีส่วนโค้งที่ซ้ำกัน และวิถี (Path: P) คือ แนวเดินที่ไม่มีบัพที่ซ้ำกัน **วงจร (Circuit)** คือรอยเดินปิด **วงจรชัด (Trivial circuit)** คือวงจรที่มีเพียงบัพเดียวไม่มีส่วนโค้ง รอยเดิน (หรือวงจร) ถูกกล่าวว่าเป็น **รอยเดินแห่งออยเลอร์ (Eulerian trail)** (หรือวงจรแห่งออยเลอร์ (Eulerian circuit)) หากทุก ๆ ส่วนโค้งในกราฟนั้นอยู่ในรอยเดิน (หรือวงจร) วง (Cycle) คือวงจรที่ไม่ชัด (Nontrivial circuit) ที่ซึ่งมีบัพที่ซ้ำกันเพียงบัพเดียว นั่นคือบัพแรกและบัพสุดท้าย



รูปที่ 2.9 แสดงตัวอย่างของแนวเดินประเภทต่าง ๆ ในไดกราฟ

พิจารณากราฟในรูปที่ 2.9 และพิจารณาแนวเดินต่อไปนี้

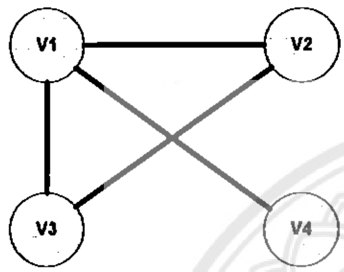
1. $\vec{W}^{(1)} = v_4\text{-E-}v_1\text{-B-}v_2\text{-K-}v_5\text{-J-}v_1\text{-B-}v_2\text{-C-}v_3\text{-H-}v_6$ เป็นแนวเดิน
2. $\vec{W}^{(2)} = v_4\text{-E-}v_1\text{-B-}v_2\text{-K-}v_5\text{-I-}v_3\text{-H-}v_6$ เป็นวิถี
3. $\vec{W}^{(3)} = v_4\text{-E-}v_1\text{-B-}v_2\text{-K-}v_5\text{-F-}v_4$ เป็นวิถีปิด หรือ วง

2.4 การแทนกราฟด้วยเมทริกซ์

โดยทั่วไป เมื่อปัญหาที่พบถูกนำมาเขียนบรรยายด้วยกราฟแล้ว วิธีที่นิยมใช้ในการแก้ปัญหาคือการแทนกราฟที่ได้ให้อยู่ในรูปของเมทริกซ์เพื่อสะดวกต่อการคำนวณ ในหัวข้อนี้กล่าวถึงวิธีการแทนกราฟด้วยเมทริกซ์ ซึ่งเมทริกซ์ที่ได้จากวิธีนี้เรียกว่า **Adjacency matrix**

ให้ G(V,E) เป็นกราฟจำกัดใด ๆ (ไม่จำเป็นต้องเป็นกราฟเชิงเดียว หรือกราฟไม่ระบุทิศทาง) โดยมีจำนวนบัพ n บัพ แล้วการแทนกราฟนี้ด้วยเมทริกซ์ทำได้โดยการสร้างเมทริกซ์ขนาด $n \times n$ โดยที่ดัชนี (index) ที่ระบุในแนวตั้งและแนวนอน คือบัพแต่ละบัพ และสมาชิกของเมทริกซ์ในตำแหน่งแนวนอนที่ i และแนวตั้งที่ j หรือ A_{ij} มีค่าเท่ากับจำนวนเส้นเชื่อม (หรือส่วนโค้ง) ที่เชื่อมจากบัพที่ i ไปยังบัพที่ j ดังแสดงในตัวอย่างต่อไปนี้

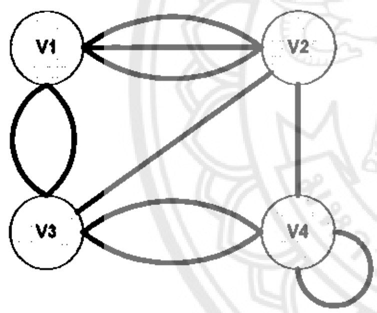
<u>ตัวอย่างที่ 1</u> [กราฟเชิงเดียว]



กราฟทางซ้ายมือ เขียนแทนด้วย Adjacency matrix ได้เป็น

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

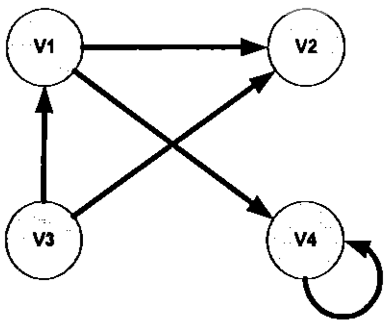
<u>ตัวอย่างที่ 2</u> [กราฟหลายเชิง]



กราฟทางซ้ายมือ เขียนแทนด้วย Adjacency matrix ได้เป็น

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 3 & 2 & 0 \\ 3 & 0 & 1 & 1 \\ 2 & 1 & 0 & 2 \\ 0 & 1 & 2 & 1 \end{bmatrix} \end{array}$$

<u>ตัวอย่างที่ 3</u> [ไดกราฟ]



กราฟทางซ้ายมือ เขียนแทนด้วย Adjacency matrix ได้เป็น

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

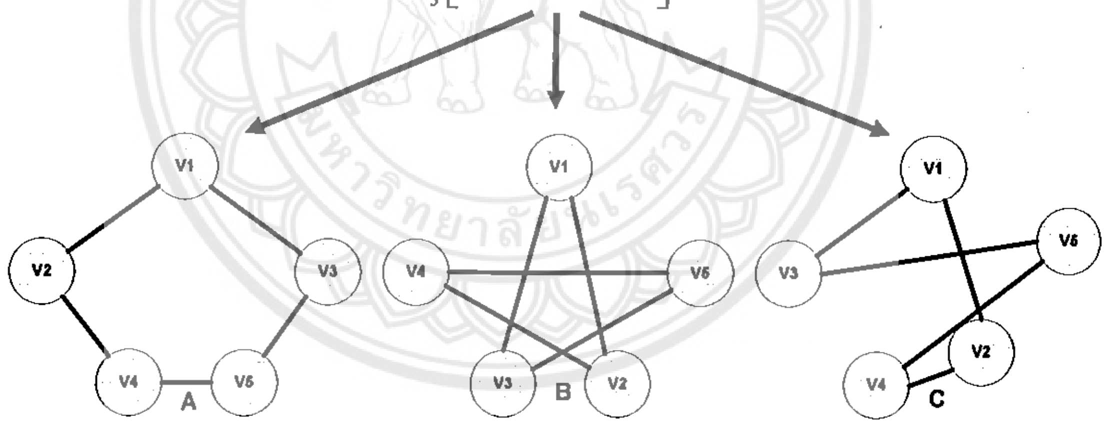
<u>ข้อสังเกต</u>

1. หากกราฟที่กำหนดมาเป็นกราฟไม่ระบุทิศทางแล้ว เมทริกซ์ที่ได้จะเป็นเมทริกซ์สมมาตรเสมอ เนื่องจาก $(v_i,v_j) = (v_j,v_i)$

2. หากกราฟที่กำหนดมาไม่มีวงวนแล้ว สมาชิกของเมทริกซ์ในเส้นทแยงมุม จะมีค่าเป็นศูนย์เสมอ
3. หากกราฟที่กำหนดมาเป็นกราฟเชิงเดียวแล้ว เมทริกซ์ที่ได้จะเป็นเมทริกซ์สมมาตร ที่มีสมาชิกแนวเส้นทแยงมุมเป็นศูนย์ และสมาชิกที่เหลือจะมีค่าเพียง 0 หรือ 1 เท่านั้น

ผู้อ่านจะเห็นว่า การแปลงระหว่างกราฟกับเมทริกซ์มีลักษณะเป็นหนึ่งต่อหนึ่งแบบสมนัย (one-to-one correspondence) หรือนั่นคือ หากให้ G และ M แทนกราฟและเมทริกซ์ที่มีลักษณะดังอธิบายข้างต้นและให้ T เป็นการแปลงระหว่างกราฟและเมทริกซ์ดังกล่าว จะได้ว่า T^{-1} หาได้เสมอ กล่าวคือ เมื่อกำหนด Adjacency matrix มาให้ ย่อมสามารถเขียนกราฟที่สมนัยกับเมทริกซ์ที่กำหนดให้ได้เสมอ ยกตัวอย่างเช่น

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

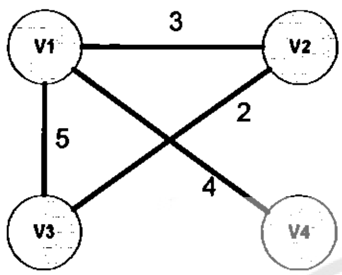


รูปที่ 2.10 การแปลงเมทริกซ์ให้อยู่ในรูปของกราฟ

จาก Adjacency matrix ข้างบนสามารถเขียนกราฟได้หลายแบบดังแสดงในรูปที่ 2.10 แม้ว่าจะเขียนกราฟได้หลายรูปแบบ แต่ทุกรูปแบบจะต้องเป็นกราฟสมสัณฐาน (Graph isomorphism) กล่าวคือ กราฟทุกแบบคือกราฟเดียวกัน หากแต่ต่างกันตรงตำแหน่งการวาง หรือ ทอพอโลยี

ในกรณีของกราฟ (หรือไดกราฟ) เชิงเดียวถ่วงน้ำหนัก สามารถเขียนแทนด้วยเมทริกซ์ที่ซึ่งสมาชิกที่ตำแหน่ง (i , j) คือค่าน้ำหนักถ่วงจากบัพที่ i ไปยังบัพที่ j และหากไม่มีเส้นเชื่อมจากบัพที่ i ไปยังบัพที่ j แล้วให้แทนสมาชิกที่ตำแหน่ง (i , j) ในเมทริกซ์ด้วย ∞ ยกตัวอย่างเช่น

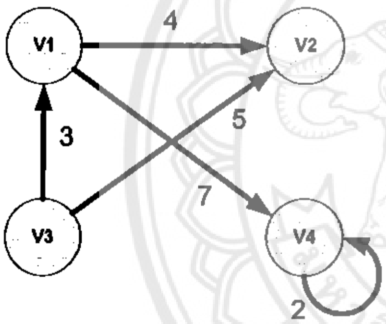
<u>ตัวอย่างที่ 4</u> [กราฟเชิงเดียวถ่วงน้ำหนัก]



กราฟทางซ้ายมือ เขียนแทนด้วยเมทริกซ์ได้ดังนี้

$$\begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{bmatrix} 0 & 3 & 5 & 4 \\ 3 & 0 & 2 & \infty \\ 5 & 2 & 0 & \infty \\ 4 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

<u>ตัวอย่างที่ 5</u> [ไดกราฟเชิงเดียวถ่วงน้ำหนัก]



กราฟทางซ้ายมือ เขียนแทนด้วยเมทริกซ์ได้ดังนี้

$$\begin{array}{c c} & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{bmatrix} 0 & 4 & \infty & 7 \\ \infty & 0 & \infty & \infty \\ 3 & 5 & 0 & \infty \\ \infty & \infty & \infty & 2 \end{bmatrix} \end{array}$$

บทที่3
การวิเคราะห์หาวิถีที่สั้นที่สุด
(Shortest Path Analysis)

จากบทที่ผ่านมาได้ศึกษาเกี่ยวกับนิยามต่าง ๆ ของกราฟ อันได้แก่ กราฟ ไดกราฟ กราฟหลายเชิง แนวเดิน รอยเดิน วิถี รวมทั้ง การแทนกราฟด้วยเมทริกซ์ เป็นต้น ในบทนี้มุ่งเน้นศึกษาการวิเคราะห์หาวิถีที่สั้นที่สุดในกราฟและไดกราฟ

การวิเคราะห์หาวิถีที่สั้นที่สุด (Shortest Path Analysis)

ขั้นตอนวิธีใช้ในการค้นหาเส้นทางบนกราฟมีผู้ที่คิดค้นขึ้นหลายคนด้วยกัน ได้แก่ Kruskal, Prim, Warshall, Floyd, Ford-Fulkerson's Labeling, Dijkstra เป็นต้น ในโครงงานนี้จะมุ่งศึกษาขั้นตอนวิธีที่ใช้ในการคำนวณหาวิถีที่สั้นที่สุด 2 วิธี กล่าวคือ ขั้นตอนวิธีของ Dijkstra และขั้นตอนวิธีของ Floyd ดังมีรายละเอียด ดังนี้

ขั้นตอนวิธีของ Dijkstra เป็นขั้นตอนวิธีที่ใช้ค้นหาวิถีที่สั้นที่สุดวิธีหนึ่งที่นิยมใช้อย่างแพร่หลาย ขั้นตอนวิธีนี้ถูกคิดค้นขึ้นในปี ค.ศ 1959 โดย นายเอ็คส์เกอร์ ไวบ์ ดิชค์สตรา (Edsger Wybe Dijkstra)ซึ่งเป็นนักวิทยาศาสตร์ชาวเนเธอร์แลนด์ ขั้นตอนวิธีของ Dijkstra เป็นการแก้ปัญหาแบบแหล่งต้นทางเดียว หรือ Single-source นั่นคือการคำนวณหาวิถีที่สั้นที่สุดจากบัพที่กำหนดให้ไปยังอีกบัพที่เหลือในไดกราฟถ่วงน้ำหนัก (Weighted digraph) หรือกราฟถ่วงน้ำหนักที่ไม่มีเส้นเชื่อมขนาน (Weighted graph without parallel edges) โดยที่ค่าน้ำหนักถ่วงเป็นค่าจริงไม่ติดลบ ซึ่งพบว่าประสิทธิภาพของขั้นตอนวิธีของ Dijksar อยู่ในระดับของ $O(|V|^2)$ โดยที่ $|V|$ คือจำนวนบัพทั้งหมดในกราฟ แต่ขั้นตอนวิธีของ Dijkstra ก็มีข้อด้อยอยู่คือ ไม่สามารถประยุกต์ใช้กับกราฟ หรือไดกราฟที่มีค่าน้ำหนักถ่วงเป็นค่าติดลบและถ้าต้องการหาวิถีที่สั้นที่สุดเมื่อมีการเปลี่ยนแหล่งต้นทางเป็นจุดใหม่ จะต้องเริ่มคำนวณใหม่ทั้งหมด ทำให้เสียเวลา

ต่อมา นายโรเบิร์ท ฟอล์ย (Robert Floyd) และนายสตีเวน วอร์แชลล์ (Stephen Warshall) ได้พัฒนาขั้นตอนวิธีใหม่เพื่อลดข้อด้อยของขั้นตอนวิธีของ Dijkstra กล่าวคือ ขั้นตอนวิธีของฟอล์ย-วอร์แชลล์ (Floyd-Warshall algorithm) เป็นวิธี ที่สามารถคำนวณหาระยะทางที่สั้นที่สุดจากแต่ละบัพในกราฟ (หรือไดกราฟ) ไปยังบัพที่เหลือทุกบัพภายในการคำนวณเพียงครั้งเดียว ไม่ต้องเริ่มคำนวณใหม่ดังเช่นในขั้นตอนวิธีของ Dijkstra นอกจากนี้ ขั้นตอนวิธีของ Folyd-Warshall ยังสามารถใช้ได้กับกราฟ (หรือไดกราฟ) ที่มีค่าถ่วงน้ำหนักที่เป็นค่าติดลบได้ แต่อย่างไรก็ตาม ในกราฟ (หรือไดกราฟ) นั้นต้องไม่มีส่วนวัฏจักรติดลบ (negative cycles) เนื่องจาก หากคู่ของบัพที่ได้

จากการคำนวณด้วยขั้นตอนวิธีของ Floyd-Warshall เป็นส่วนหนึ่งของวัฏจักรติดลบ ทำให้ขั้นตอนวิธีของ Floyd-Warshall ไม่สามารถทำงานได้ อันเนื่องจากค่าน้ำหนักถ่วงโดยรวมจะมีค่าลดลงเรื่อย ๆ ภายในวัฏจักรติดลบนั้นไปอย่างไม่มีที่สิ้นสุด หากมองในแง่ของประสิทธิภาพแล้ว พบว่าขั้นตอนวิธีของ Floyd-Warshall มีความซับซ้อนเชิงเวลา (Time-complexity) ในระดับของ $O\left(|V|^3\right)$

3.1ขั้นตอนวิธีของ Dijkstra

ให้ G เป็นกราฟถ่วงน้ำหนัก G = G(V,E,w) โดยที่ $V = \{v_i\}_{i=1}^{n}$ และ w เป็นฟังก์ชันค่าน้ำหนักถ่วงบนเส้นเชื่อม ซึ่งมีข้อกำหนดดังนี้

1. หากมีเส้นเชื่อมจากบัพ v_i ไปยังบัพ v_j ใด ๆ ที่ซึ่งมีค่าน้ำหนักถ่วงระหว่างสองบัพดังกล่าวเป็น k ดังนั้น $w(v_i, v_j) = k$
2. หากไม่มีเส้นเชื่อมจากบัพ v_i ไปยังบัพ v_j จะกำหนดให้ $w(v_i, v_j) = \infty$
3. หากไม่มีวง (loop) ที่บัพ v_i ให้ค่า $w(v_i, v_j) = 0$

ในขั้นตอนวิธีของ Dijkstra นั้น จะเริ่มจากบัพแรก v_1 ซึ่งเป็นแหล่งต้นทาง จากนั้นจะทำการคำนวณหาบัพถัดไปที่ซึ่งค่าน้ำหนักถ่วงรวมต่ำที่สุด ซึ่งหลักการนี้แสดงไว้ในตารางที่ 1 โดยที่

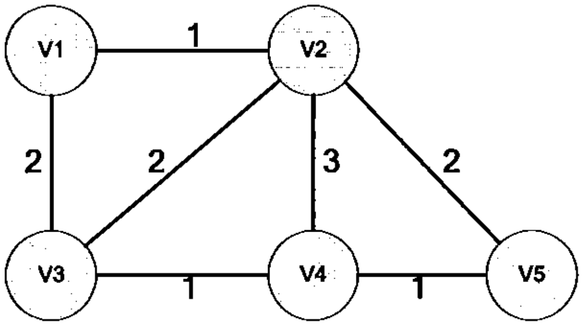
1. dist[v_i] เป็นฟังก์ชันระยะทางที่เก็บค่าน้ำหนักถ่วงรวมจากบัพเริ่มต้น v_1 ถึงบัพ v_i
2. new_dist[v_j] ใช้คำนวณหาค่าน้ำหนักถ่วงค่าใหม่อันเกิดจาก ค่าน้ำหนักถ่วงรวมจากบัพ v_1 จนถึงบัพ v_i (ซึ่งเป็นบัพประชิดกับบัพ v_j) รวมกับค่าน้ำหนักถ่วงจากบัพ v_i ไปยังบัพ v_j นั่นคือ
   new_dist[v_j] = dist[v_i] + $w(v_i, v_j)$
3. previous[v_i] ทำหน้าที่เก็บบัพที่ถูกเก็บก่อนหน้าบัพ v_i
4. pop(Q) ทำหน้าที่ดึงบัพออกจากเซต Q

ตารางที่1 การคำนวณหาวิถีที่สั้นที่สุดโดยขั้นตอนวิธี Dijkstra

บรรทัดที่	รหัสเทียม	คำอธิบาย	ประมวลผล	รวม
1	for i = 1,2,..., $\lvert V\rvert$ dist[v_i] := ∞ previous[v_i] := ∅	กำหนดค่าเริ่มต้น	วนลูป for ทั้งหมด n ครั้ง ในแต่ละครั้งมีการประมวลผล 2 คำสั่ง ดังนั้นจึงประมวลผลรวม 2n ครั้ง	2n
2	dist[v_1] := 0	กำหนดค่าน้ำหนักถ่วงรวมจากบัพเริ่มต้น v_1 ไปยังบัพ v_1 เป็น 0	มีการประมวลผล 1 คำสั่ง	2n+2
3	Q := copy(Graph)	เก็บบัพทั้งหมดของกราฟไว้	มีการประมวลผล 1 คำสั่ง	
4	while Q is not empty:	ตราบเท่าที่ Q ยังมีสมาชิกอยู่	วนลูป while ทั้งหมด n ครั้ง	2n+2+ [4(n-1)+1]n
	v_i := pop(Q) with min(dist[v_i])	เลือก v_i จาก Q ซึ่งมีค่าน้ำหนักถ่วงรวมจากบัพเริ่มต้น v_1 ไปยังบัพ v_i น้อยที่สุดสำหรับทุกบัพใน Q	มีการประมวลผล 1 คำสั่ง	
4.1	for each $v_j \in Q$ with $w(v_i, v_j) \neq \infty$	พิจารณาบัพใน Q ที่ประชิดกับบัพ v_i	วนลูป for ทั้งหมด n-1 ครั้ง	
	new_dist[v_j] = dist[v_i]+w(v_i , v_j)	คำนวณหาค่าน้ำหนักถ่วงค่าใหม่	มีการประมวลผล 1 คำสั่ง	
	if new_dist[v_j] < dist[v_j]	ถ้าค่าน้ำหนักถ่วงรวมค่าใหม่น้อยกว่าค่าน้ำหนักถ่วงรวมค่าเก่า	มีการประมวลผล 1 คำสั่ง	
	dist[v_j] := new_dist[v_j]	ปรับค่าน้ำหนักถ่วงรวมจากบัพเริ่มต้น v_1 ไปยังบัพ v_j เป็นค่าน้ำหนักถ่วงรวมค่าใหม่	มีการประมวลผล 1 คำสั่ง	
	previous[v_j] := v_i	เก็บบัพก่อนหน้าบัพ v_j ด้วยบัพ v_i	มีการประมวลผล 1 คำสั่ง	
	return previous[ ]	ส่งค่าใน previous กลับมา	มีการประมวลผล 1 คำสั่ง	

เนื่องจากมีการประมวลผลโดยประมาณ $\left[4(n\text{-}1)+1\right]$n+2n+2+1=4$n^2$-n+3 ดังนั้น ความซับซ้อนของขั้นตอนวิธีของ Dijkstra คือ O($|V|^2$)

ตัวอย่างที่ 1 พิจารณากราฟในรูปที่ 3.4 โดยกำหนดให้บัพเริ่มต้นคือ v_1

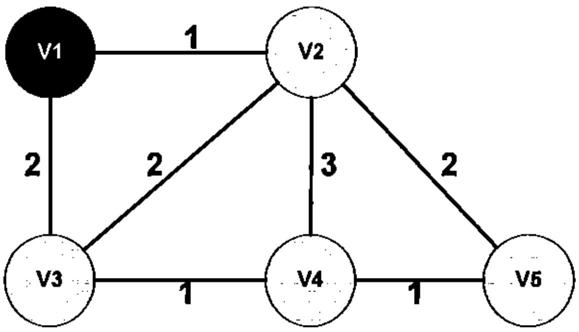


รูปที่ 3.1 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra

ขั้นตอนที่ 1 กำหนดค่าเริ่มต้น กล่าวคือให้ dist[v_i] = ∞ และ previous[v_i] = ∅ สำหรับทุกค่า i = 1,...,5 จากนั้นให้ค่าน้ำหนักถ่วงรวมเริ่มต้นเป็นศูนย์ นั่นคือ dist[v_1] = 0 ให้ Q เก็บบัพทุกบัพในกราฟ นั่นคือ $Q=\{v_1,v_2,v_3,v_4,v_5\}$ ซึ่งแสดงไว้ในตารางต่อไปนี้

บัพ	dist[v_i]	previous[v_i]
v_1	0	∅
v_2	∞	∅
v_3	∞	∅
v_4	∞	∅
v_5	∞	∅

ขั้นตอนที่ 2 เนื่องจากเซต Q ≠ ∅ ให้เลือกบัพ v_i จากเซต Q ที่ซึ่งมีค่า dist[v_i] น้อยที่สุด ซึ่งจากตารางข้างต้น พบว่า บัพ v_1 มีค่า dist[v_1] = 0 ซึ่งเป็นค่าต่ำสุด ดังนั้น จึงเลือก $v_i = v_1$ ดังนั้น เซต $Q=\{v_2,v_3,v_4,v_5\} \neq \varnothing$ ดังรูปที่ 3.2



รูปที่ 3.2 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra ในขั้นตอนที่ 2

พิจารณาบัพใน Q ที่ประชิดกับบัพ v_1 ซึ่งนั้นคือบัพ v_2 และ v_3

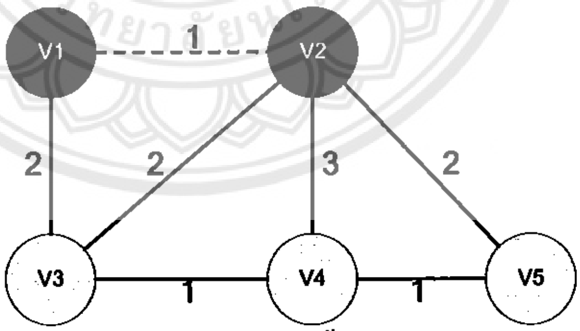
<u>ที่บัพ</u> v_2 : new_dist[v_2] = dist[v_1] + w(v_1 , v_2) = 0 + 1 = 1 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_2] < dist[v_2] = ∞ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_2] = 1 และ previous[v_2] = v_1

<u>ที่บัพ</u> v_3 : new_dist[v_3] = dist[v_1] + w(v_1 , v_3) = 0 + 2 = 2 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_3] < dist[v_3] = ∞ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_3] = 2 และ previous[v_3] = v_1

ค่าน้ำหนักถ่วงรวม รวมทั้งบัพที่ถูกเก็บใน previous[v_i] แสดงไว้ในตาราง

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	1	v_1
v_3	2	v_1
v_4	∞	$\varnothing$
v_5	∞	$\varnothing$

ขั้นตอนที่ 3 เนื่องจากเซต $Q=\{v_2,v_3,v_4,v_5\} \neq \varnothing$ ให้เลือกบัพ v_i จากเซต Q ที่ซึ่งมีค่า dist[v_i] น้อยที่สุด ซึ่งจากตารางข้างต้น พบว่า บัพ v_2 มีค่า dist[v_2] = 1 ซึ่งเป็นค่าต่ำสุด ดังนั้น จึงเลือก $v_i = v_2$ ดังนั้น เซต $Q=\{v_3,v_4,v_5\} \neq \varnothing$ ดังรูปที่ 3.3



รูปที่ 3.3 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra ในขั้นตอนที่ 3

พิจารณาบัพใน Q ที่ประชิดกับบัพ v_2 ซึ่งนั้นคือบัพ v_3 , v_4 และ v_5

<u>ที่บัพ</u> v_3 : new_dist[v_3] = dist[v_2] + w(v_2 , v_3) = 1 + 2 = 3 ซึ่งไม่เข้าเงื่อนไข if statement นั่นคือ new_dist[v_3] > dist[v_3] = 2 ดังนั้นจึงไม่ทำการปรับค่าใด ๆ

<u>ที่บัพ</u> v_4 : new_dist[v_4] = dist[v_2] + w(v_2 , v_4) = 1 + 3 = 4 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_4] < dist[v_4] = ∞ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_4] = 4 และ previous[v_4] = v_2

<u>ที่บัพ</u> v_5 : new_dist[v_5] = dist[v_2] + w(v_2 , v_5) = 1 + 2 = 3 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_5] < dist[v_5] = ∞ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_5] = 3 และ previous[v_5] = v_2

ค่าน้ำหนักถ่วงรวม รวมทั้งบัพที่ถูกเก็บใน previous[v_i] แสดงไว้ในตาราง

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	1	v_1
v_3	2	v_1
v_4	4	v_2
v_5	3	v_2

ขั้นตอนที่ 4 เนื่องจาก Q= $\{v_3, v_4, v_5\} \neq \varnothing$ จึงย้อนกลับไปทำเช่นเดียวกับในขั้นตอนที่ 2 หรือ 3 ข้างต้น และเมื่อเสร็จสิ้นกระบวนการทั้งหมดแล้วจะมีผลลัพธ์ดังแสดงในตาราง

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	1	v_1
v_3	2	v_1
v_4	3	v_3
v_5	3	v_2

จากตารางสุดท้ายนี้ พบว่า

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_2 คือ 1 และวิถีคือ v_1 v_2

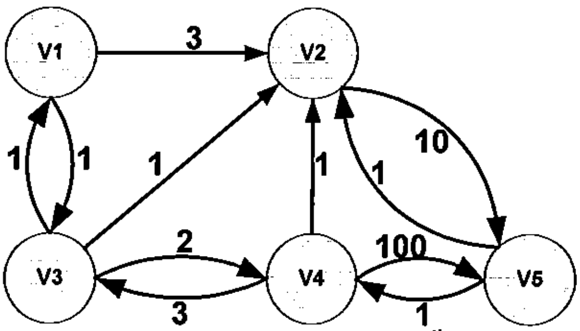
วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_3 คือ 2 และวิถีคือ v_1 v_3

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_4 คือ 3 และวิถีคือ v_1 v_3 v_4

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_5 คือ 3 และวิถีคือ v_1 v_2 v_5

แต่หากต้องการหาวิถีที่สั้นที่สุดจากบัพอื่น ๆ ที่มิใช่บัพ v_1 ต้องเริ่มต้นคำนวณใหม่

ตัวอย่างที่ 2 พิจารณากราฟในรูปที่ 3.4 โดยกำหนดให้บัพเริ่มต้นคือ v_1

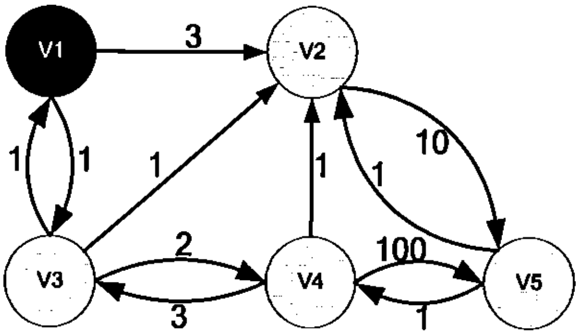


รูปที่ 3.4 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra

ขั้นตอนที่ 1 กำหนดค่าเริ่มต้น กล่าวคือให้ dist[v_i] = ∞ และ previous[v_i] = $\varnothing$ สำหรับทุกค่า i = 1,...,5 จากนั้นให้ค่าน้ำหนักถ่วงรวมเริ่มต้นเป็นศูนย์ นั่นคือ dist[v_1] = 0 ให้ Q เก็บบัพทุกบัพในกราฟ นั่นคือ $Q=\{v_1,v_2,v_3,v_4,v_5\}$ ซึ่งแสดงไว้ในตารางต่อไปนี้

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	∞	$\varnothing$
v_3	∞	$\varnothing$
v_4	∞	$\varnothing$
v_5	∞	$\varnothing$

ขั้นตอนที่ 2 เนื่องจากเซต $Q \neq \varnothing$ ให้เลือกบัพ v_i จากเซต Q ที่ซึ่งมีค่า dist[v_i] น้อยที่สุด ซึ่งจากตารางข้างต้น พบว่า บัพ v_1 มีค่า dist[v_1] = 0 ซึ่งเป็นค่าต่ำสุด ดังนั้น จึงเลือก $v_i = v_1$ ดังนั้น เซต $Q=\{v_2,v_3,v_4,v_5\} \neq \varnothing$ ดังรูปที่ 3.5



รูปที่ 3.5 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra ในขั้นตอนที่ 2

พิจารณาบัพใน Q ที่ประชิดกับบัพ v_1 ซึ่งนั้นคือบัพ v_2 และ v_3

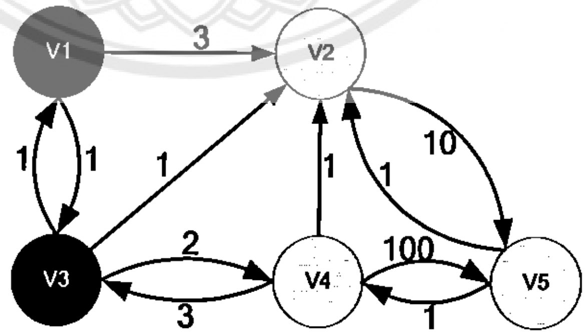
<u>ที่บัพ</u> v_2 : $new\_dist[v_2] = dist[v_1] + w(v_1, v_2) = 0 + 3 = 3$ ซึ่งเข้าเงื่อนไข if statement นั่นคือ $new\_dist[v_2] < dist[v_2] = \infty$ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม $dist[v_2] = 3$ และ $previous[v_2] = v_1$

<u>ที่บัพ</u> v_3 : $new\_dist[v_3] = dist[v_1] + w(v_1, v_3) = 0 + 1 = 1$ ซึ่งเข้าเงื่อนไข if statement นั่นคือ $new\_dist[v_3] < dist[v_3] = \infty$ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม $dist[v_3] = 1$ และ $previous[v_3] = v_1$

ค่าน้ำหนักถ่วงรวม รวมทั้งบัพที่ถูกเก็บใน $previous[v_i]$ แสดงไว้ในตาราง

บัพ	$dist[v_i]$	$previous[v_i]$
v_1	0	$\varnothing$
v_2	3	v_1
v_3	1	v_1
v_4	∞	$\varnothing$
v_5	∞	$\varnothing$

ขั้นตอนที่ 3 เนื่องจากเซต $Q=\{v_2,v_3,v_4,v_5\} \neq \varnothing$ ให้เลือกบัพ v_i จากเซต Q ที่ซึ่งมีค่า $dist[v_i]$ น้อยที่สุด ซึ่งจากตารางข้างต้น พบว่า บัพ v_3 มีค่า $dist[v_3] = 1$ ซึ่งเป็นค่าต่ำสุด ดังนั้น จึงเลือก $v_i = v_3$ ดังนั้น เซต $Q=\{v_2,v_4,v_5\} \neq \varnothing$ ดังรูปที่ 3.6



รูปที่ 3.6 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Dijkstra ในขั้นตอนที่ 3

พิจารณาบัพใน Q ที่ประชิดกับบัพ v_3 ซึ่งนั้นคือบัพ v_2 และ v_4

<u>ที่บัพ</u> v_2 : new_dist[v_2] = dist[v_3] + w(v_3 , v_2) = 1 + 1 = 2 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_2] < dist[v_2] = 3 ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_2] = 2 และ previous[v_2] = v_3

<u>ที่บัพ</u> v_4 : new_dist[v_4] = dist[v_3] + w(v_3 , v_4) = 1 + 2 = 3 ซึ่งเข้าเงื่อนไข if statement นั่นคือ new_dist[v_4] < dist[v_4] = ∞ ดังนั้นจึงทำการปรับค่าน้ำหนักถ่วงรวม dist[v_4] = 3 และ previous[v_4] = v_3

ค่าน้ำหนักถ่วงรวม รวมทั้งบัพที่ถูกเก็บใน previous[v_i] แสดงไว้ในตาราง

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	2	v_3
v_3	1	v_1
v_4	3	v_3
v_5	∞	$\varnothing$

ขั้นตอนที่4 เนื่องจากเซต $Q=\{v_2,v_4,v_5\} \neq \varnothing$ จึงย้อนกลับไปทำเช่นเดียวกับในขั้นตอนที่ 2 หรือ 3 ข้างต้น และเมื่อเสร็จสิ้นกระบวนการทั้งหมดแล้วจะมีผลลัพธ์ดังแสดงในตาราง

บัพ	dist[v_i]	previous[v_i]
v_1	0	$\varnothing$
v_2	2	v_3
v_3	1	v_1
v_4	3	v_3
v_5	12	v_2

จากตารางสุดท้ายนี้ พบว่า

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_2 คือ 2 และวิถีคือ v_1 v_3 v_2

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_3 คือ 1 และวิถีคือ v_1 v_3

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_4 คือ 3 และวิถีคือ v_1 v_3 v_4

วิถีที่สั้นที่สุดจากบัพ v_1 ไปยังบัพ v_5 คือ 12 และวิถีคือ v_1 v_3 v_2 v_5

แต่หากต้องการหาวิถีที่สั้นที่สุดจากบัพอื่น ๆ ที่มิใช่บัพ v_1 ต้องเริ่มต้นคำนวณใหม่

3.2 ขั้นตอนวิธีของ Floyd-Warshall

ขั้นตอนวิธีของ Floyd-Warshall เป็นขั้นตอนวิธีอีกชนิดหนึ่งที่ใช้ในการค้นหาเส้นทางที่สั้นที่สุด โดยมีลักษณะการคำนวณคล้ายคลึงกับขั้นตอนวิธีของ Dijkstra แต่ต่างกันตรงที่วิธีการปรับค่าน้ำหนักถ่วง ให้ G เป็นกราฟถ่วงน้ำหนัก G = G(V,E,w) โดยที่ $V = \{v_i\}_{i=1}^{n}$ และ w เป็นฟังก์ชันค่าน้ำหนักถ่วงบนเส้นเชื่อม ซึ่งมีข้อกำหนดดังเช่นในขั้นตอนวิธีของ Dijkstra

ขั้นตอนวิธีของ Floyd-Warshall อาศัย 2 เมทริกซ์ในการเก็บค่าพารามิเตอร์ที่สำคัญ ดังนี้ เมทริกซ์ A_k ใช้ในการเก็บค่าน้ำหนักถ่วงในแต่ละรอบ k และเมทริกซ์ P_k ทำหน้าที่เป็น ดัชนี (index) ในรอบนั้น ๆ โดยสมาชิกในตำแหน่ง (i,j) ของเมทริกซ์ทั้งคู่ มีการเปลี่ยนแปลงตามเงื่อนไขนี้

<u>ในรอบที่ k</u> โดยที่ k มีค่าอยู่ในช่วง $1 \le k \le n$

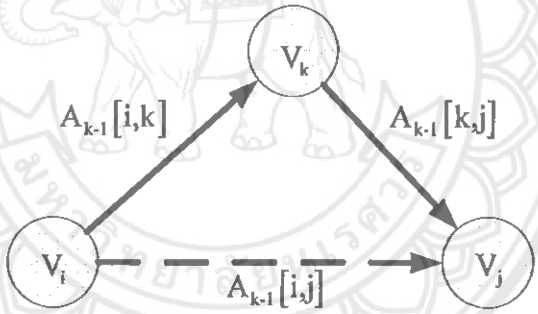
ถ้า $A_{k-1}[i,j] < A_{k-1}[i,k]+A_{k-1}[k,j]$

กำหนดให้ $A_k[i,j] = A_{k-1}[i,j]$ และ $P_k[i,j] = P_{k-1}[i,j]$

แต่ถ้า $A_{k-1}[i,j] > A_{k-1}[i,k]+A_{k-1}[k,j]$

กำหนดให้ $A_k[i,j]=A_{k-1}[i,k]+A_{k-1}[k,j]$ และ $P_k[i,j] = k$

โดยที่ ค่าเริ่มต้น $A_0[i,j] = w(i,j)$ และ $P_k[i,j] = 0$ สำหรับทุกค่า i , j



รูปที่ 3.7 แสดงตัวอย่างการคำนวณในขั้นตอนวิธีของ Floyd-Warshall

ดังนั้น ในแต่ละรอบ k หาก $P_k[i,j] = k \neq 0$ นั่นย่อมแสดงว่า ในรอบที่ k นั้น ค่าน้ำหนักถ่วงระหว่างบัพ v_i ไปบัพ v_j มีค่ามากกว่าผลรวมของค่าน้ำหนักถ่วงระหว่างบัพ v_i ไปบัพ v_k และบัพ v_k ไปบัพ v_j ด้วยเหตุนี้ เมื่อการคำนวณเสร็จสิ้น ตำแหน่ง (i,j) ของเมทริกซ์ P จะบ่งบอกบัพที่ประชิดกับบัพที่ i ที่ทำให้ค่าถ่วงน้ำหนักรวมมีค่าน้อยที่สุด ดังนั้น ในการคำนวณหาวิถีที่สั้นที่สุด ทำโดยการดูตำแหน่ง (i,j) ของเมทริกซ์ P ตามขั้นตอนดังนี้

หากตำแหน่ง (i,j) ของเมทริกซ์ P มีค่าเป็นศูนย์ นั่นย่อมแสดงว่า มีเส้นเชื่อมระหว่างบัพ v_i และบัพ v_j ที่ซึ่งทำให้ค่าน้ำหนักถ่วงรวมมีค่าน้อยที่สุด แต่หากตำแหน่ง (i,j) ของเมทริกซ์ P มีค่าเท่ากับ $k_1 \neq 0$ นั่นย่อมแสดงว่า ต้องมีเส้นเชื่อมระหว่างบัพ v_i และบัพ v_{k_1} ที่ซึ่งทำให้ค่าน้ำหนักถ่วงรวมมีค่าน้อยที่สุด จากนั้นให้พิจารณาต่อว่า ระหว่างบัพ v_{k_1} ไปยังบัพ v_j ให้ค่าน้ำหนักถ่วงซึ่งทำให้ค่าน้ำหนักถ่วงรวมมีค่าน้อยที่สุดหรือไม่โดยดูจากตำแหน่ง (k_1,j) ของเมทริกซ์ P หาก

ตำแหน่ง (k_1,j) ของเมทริกซ์ P เท่ากับ $k_2 \neq 0$ นั่นย่อมแสดงว่า ต้องมีบัพ k_2 เชื่อมต่อกับบัพ k_1 ทำเช่นนี้ไปเรื่อย ๆ จนกระทั่งตำแหน่ง (k_m,j) ของเมทริกซ์ P เท่ากับศูนย์ และวิถีที่สั้นที่สุดจากบัพ v_i ไปยังบัพ v_j คือ $v_i e_{i,k_1} v_{k_1} e_{k_1,k_2} v_{k_2} \cdots e_{k_{m-1},k_m} v_{k_m} e_{k_m,j} v_j$ โดยที่ค่าถ่วงน้ำหนักรวมที่น้อยที่สุด คือ ตำแหน่ง (i,j) ของเมทริกซ์ A นั่นเอง การคำนวณหาวิถีที่สั้นที่สุดโดยขั้นตอนวิธีของ Floyd-Warshall แสดงไว้ในตารางที่ 2

ตารางที่ 2 การคำนวณหาวิถีที่สั้นที่สุดโดยขั้นตอนวิธีของ Floyd-Warshall

คำสั่งที่	รหัสเทียม	คำอธิบาย	ประมวลผล	รวม
1	Initialization:	ให้ค่าเริ่มต้น		
	$A_0(i,j)=w(i,j)$; $P_0(i,j)=0$;	สร้างเมทริกซ์ A และ P เริ่มต้น	ประมวลผลทั้งหมด 2 ครั้ง	2
2	for k = 1 to n	วนลูป for เพื่อเพิ่มค่า k	ประมวลผลทั้งหมด n ครั้ง	$2+4n^3$
3	for i = 1 to n for j = 1 to n	ที่ k ค่าหนึ่ง ทำการวนลูป for เพื่อเปรียบเทียบค่าน้ำหนักถ่วง	ประมวลผลทั้งหมด n ครั้ง ในแต่ละลูป	
3.1	if $(A_{k-1}[i,j] > A_{k-1}[i,k] + A_{k-1}[k,j])$	หากค่าน้ำหนักถ่วงเข้าเงื่อนไข $w[i,j] > w[i,k] + w[k,j]$	ประมวลผลทั้งหมด 1 ครั้ง	
	$A_k[i,j] = A_{k-1}[i,k] + A_{k-1}[k,j]$	กำหนด $w[i,j]$ เป็นค่าใหม่	ประมวลผลทั้งหมด 1 ครั้ง	
	$P_k[i,j] = k$	ระบุว่าวิถีจาก v_i ไป v_j ต้องผ่าน v_k	ประมวลผลทั้งหมด 1 ครั้ง	
3.2	else $A_k[i,j] = A_{k-1}[i,j]$	หากค่าน้ำหนักถ่วงเข้าเงื่อนไข $w[i,j] \leq w[i,k] + w[k,j]$ ให้คงค่าเดิมซึ่งเป็นค่าต่ำสุด	ประมวลผลทั้งหมด 1 ครั้ง	
4	Let v_i=source and v_j=destination Path = $\{v_i\}$	กำหนดบัพเริ่มต้นและบัพสุดท้าย และให้ v_i เป็นบัพแรกในวิถี	ประมวลผลทั้งหมด 3 ครั้ง	$2+4n^3+3+3n$
4.1	while $P_k[i,j] = k \neq 0$	ตรวจสอบว่าต้องมีบัพอื่นเชื่อมระหว่าง v_i และ v_j	ประมวลผลทั้งหมด n ครั้ง	
4.2	Path = push($\{v_k\}$)	ใส่ v_k เพิ่มลงในวิถี	ประมวลผลทั้งหมด 1 ครั้ง	
4.3	Set i = k	เลื่อนบัพเริ่มต้นมาที่ v_k	ประมวลผลทั้งหมด 1 ครั้ง	
4.4	return path	ให้ส่งวิถีที่สั้นที่สุดที่เก็บใน path	ประมวลผลทั้งหมด 1 ครั้ง	

*จากตารางจะได้ประสิทธิภาพของอัลกอริทึมนี้จะเป็น $O(|V|^3)$

หมายเหตุ

จากหลักการคำนวณในขั้นตอนวิธีของ Floyd-Warshall พบว่าเงื่อนไขที่จำเป็นในการหาค่าน้ำหนักถ่วงที่ต่ำที่สุดนั้นคือ

$$A_k[i,j] = \min\{A_{k-1}[i,j], A_{k-1}[i,k] + A_{k-1}[k,j]\}$$

ในรอบที่ k ให้พิจารณากรณีที่ i = k พบว่า

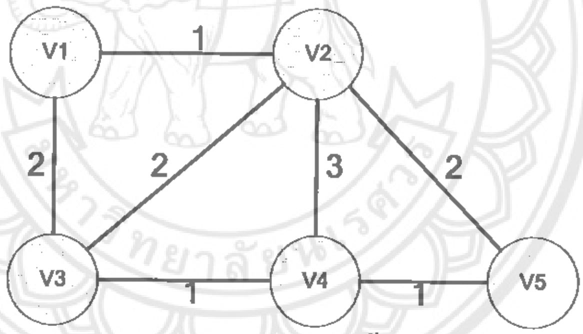
$$A_{k-1}[k,j] < A_{k-1}[k,k] + A_{k-1}[k,j]$$

ดังนั้น $A_k[k,j] = A_{k-1}[k,j]$ และ $P_k[k,j] = P_{k-1}[k,j]$ ซึ่งคือค่าเดิม และกรณีที่ j = k พบว่า

$$A_{k-1}[i,k] < A_{k-1}[i,k] + A_{k-1}[k,k]$$

จะได้ว่า $A_k[i,k] = A_{k-1}[i,k]$ และ $P_k[i,k] = P_{k-1}[i,k]$ ซึ่งคือค่าเดิมเช่นกัน ดังนั้น เพื่อลดเวลาในการคำนวณรอบที่ k ลง จึงพิจารณาเมทริกซ์ย่อยของ A_{k-1} โดยตัดแถวที่ k และหลักที่ k ออกจากเมทริกซ์ A_{k-1} เนื่องจากเป็นตำแหน่งที่ไม่มีการเปลี่ยนแปลง ดังนั้น จำนวนครั้งในการคำนวณจึงลดลงจาก n^2 เป็น $(n-1)^2$ แทน

ตัวอย่างที่ 3 พิจารณากราฟในรูปที่ 3.8



รูปที่ 3.8 กราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Floyd-Warshall

จากกราฟในรูปที่3.8 สามารถนำมาเขียนเป็นเมทริกซ์ A_0 และ P_0 ได้ดังนี้

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}
\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & \infty & \infty \\ 1 & 0 & 2 & 3 & 2 \\ 2 & 2 & 0 & 1 & \infty \\ \infty & 3 & 1 & 0 & 1 \\ \infty & 2 & \infty & 1 & 0 \end{bmatrix} \\ A_0 \end{array}
\quad \text{และ} \quad
\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}
\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ P_0 \end{array}$$

<u>ขั้นตอนที่1</u>

เมื่อ k = 1 ให้ตัดแถวที่ 1 หลักที่ 1 ของเมทริกซ์ A_0 ดังเส้นประ

$$
\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\
\begin{bmatrix} 0 & 1 & 2 & \infty & \infty \\ 1 & 0 & 2 & 3 & 2 \\ 2 & 2 & 0 & 1 & \infty \\ \infty & 3 & 1 & 0 & 1 \\ \infty & 2 & \infty & 1 & 0 \end{bmatrix}
\end{array}
$$

A_0

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้นด้วยเงื่อนไข ดังนี้

$A_1[2, 2] = \min(A_0[2, 2], A_0[2, 1] + A_0[1, 2]) = \min(0, 1+1) = 0$

$A_1[2, 3] = \min(A_0[2, 3], A_0[2, 1] + A_0[1, 3]) = \min(2, 1+2) = 2$

$A_1[2, 4] = \min(A_0[2, 4], A_0[2, 1] + A_0[1, 4]) = \min(3, 1+\infty) = 3$

$A_1[2, 5] = \min(A_0[2, 5], A_0[2, 1] + A_0[1, 5]) = \min(2, 1+\infty) = 2$

$A_1[3, 3] = \min(A_0[3, 3], A_0[3, 1] + A_0[1, 3]) = \min(0, 2+2) = 0$

$A_1[3, 4] = \min(A_0[3, 4], A_0[3, 1] + A_0[1, 4]) = \min(1, 2+\infty) = 1$

$A_1[3, 5] = \min(A_0[3, 5], A_0[3, 1] + A_0[1, 5]) = \min(\infty, 2+\infty) = \infty$

$A_1[4, 4] = \min(A_0[4, 4], A_0[4, 1] + A_0[1, 4]) = \min(0, \infty+\infty) = 0$

$A_1[4, 5] = \min(A_0[4, 5], A_0[4, 1] + A_0[1, 5]) = \min(1, \infty+\infty) = 1$

$A_1[5, 5] = \min(A_0[5, 5], A_0[5, 1] + A_0[1, 5]) = \min(0, \infty+\infty) = 0$

ในรอบนี้ เมทริกซ์ A ไม่มีการเปลี่ยนแปลง จึงเริ่มคำนวณในรอบถัดไป

<u>ขั้นตอนที่2</u>

เมื่อ k = 2 ให้ตัดแถวที่ 2 หลักที่ 2 ของเมทริกซ์ A_1 ดังเส้นประ

$$
\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\
\begin{bmatrix} 0 & 1 & 2 & \infty & \infty \\ 1 & 0 & 2 & 3 & 2 \\ 2 & 2 & 0 & 1 & \infty \\ \infty & 3 & 1 & 0 & 1 \\ \infty & 2 & \infty & 1 & 0 \end{bmatrix}
\end{array}
$$

A_1

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้นด้วยเงื่อนไข ดังนี้

$A_2[1,1] = \min(A_1[1,1], A_1[1,2] + A_1[2,1]) = \min(0, 1+1) = 0$

$A_2[1,3] = \min(A_1[1,3], A_1[1,2] + A_1[2,3]) = \min(2, 1+2) = 2$

$A_2[1,4] = \min(A_1[1,4], A_1[1,2] + A_1[2,4]) = \min(\infty, 1+3) = 4,\ P_2[1,4] = 2$

$A_2[1,5] = \min(A_1[1,5], A_1[1,2] + A_1[2,5]) = \min(\infty, 1+2) = 3,\ P_2[1,5] = 2$

$A_2[3,1] = \min(A_1[3,1], A_1[3,2] + A_1[2,1]) = \min(2, 2+1) = 2$

$A_2[3,3] = \min(A_1[3,3], A_1[3,2] + A_1[2,3]) = \min(0, 2+2) = 0$

$A_2[3,4] = \min(A_1[3,4], A_1[3,2] + A_1[2,4]) = \min(1, 2+2) = 1$

$A_2[3,5] = \min(A_1[3,5], A_1[3,2] + A_1[2,5]) = \min(\infty, 2+2) = 4,\ P_2[3,5] = 2$

$A_2[4,1] = \min(A_1[4,1], A_1[4,2] + A_1[2,1]) = \min(\infty, 3+1) = 4,\ P_2[4,1] = 2$

$A_2[4,3] = \min(A_1[4,3], A_1[4,2] + A_1[2,3]) = \min(1, 3+2) = 1$

$A_2[4,4] = \min(A_1[4,4], A_1[4,2] + A_1[2,4]) = \min(0, 3+3) = 0$

$A_2[4,5] = \min(A_1[4,5], A_1[4,2] + A_1[2,5]) = \min(1, 3+2) = 1$

$A_2[5,1] = \min(A_1[5,1], A_1[5,2] + A_1[2,1]) = \min(\infty, 2+1) = 3,\ P_2[5,1] = 2$

$A_2[5,3] = \min(A_1[5,3], A_1[5,2] + A_1[2,3]) = \min(\infty, 2+2) = 4,\ P_2[5,3] = 2$

$A_2[5,4] = \min(A_1[5,4], A_1[5,2] + A_1[2,4]) = \min(1, 2+3) = 1$

$A_2[5,5] = \min(A_1[5,5], A_1[5,2] + A_1[2,5]) = \min(0, 2+2) = 0$

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้น จะได้เมทริกซ์ A_2 และ P_2 ดังนี้

$$
\begin{array}{c|ccccc|}
 & V_1 & V_2 & V_3 & V_4 & V_5 \\
\hline
V_1 & 0 & 1 & 2 & 4 & 3 \\
V_2 & 1 & 0 & 2 & 3 & 2 \\
V_3 & 2 & 2 & 0 & 1 & 4 \\
V_4 & 4 & 3 & 1 & 0 & 1 \\
V_5 & 3 & 2 & 4 & 1 & 0 \\
\hline
\end{array}
\quad \text{และ} \quad
\begin{array}{c|ccccc|}
 & V_1 & V_2 & V_3 & V_4 & V_5 \\
\hline
V_1 & 0 & 0 & 0 & 2 & 2 \\
V_2 & 0 & 0 & 0 & 0 & 0 \\
V_3 & 0 & 0 & 0 & 0 & 2 \\
V_4 & 2 & 0 & 0 & 0 & 0 \\
V_5 & 2 & 0 & 2 & 0 & 0 \\
\hline
\end{array}
$$

A_2 $\qquad\qquad\qquad$ P_2

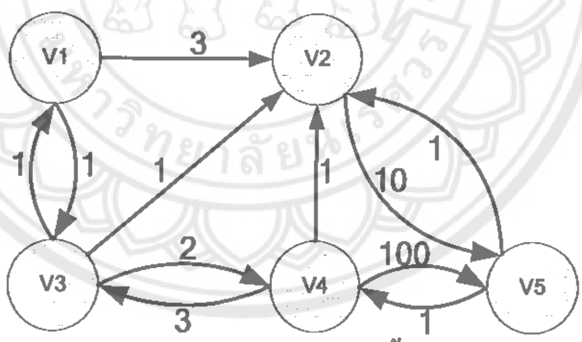
หลังจากทำตามขั้นตอนวิธี ดังแสดงในตารางที่ 2 จนกระทั่ง k = 5 จะได้

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 & 3 \\ 1 & 0 & 2 & 3 & 2 \\ 2 & 2 & 0 & 1 & 2 \\ 3 & 3 & 1 & 0 & 1 \\ 3 & 2 & 2 & 1 & 0 \end{bmatrix} \\ A_5 \end{array} \quad \text{และ} \quad \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 0 \end{bmatrix} \\ P_5 \end{array}$$

หากต้องการหาวิถีที่ต่ำที่สุดจากบัพ V_1 ไปยังบัพ V_5 ให้พิจารณาตามขั้นตอนดังนี้ ให้บัพเริ่มต้นถูกเก็บไว้ในตัวแปร path นั่นคือ path = $\{V_1\}$ แล้วพิจารณาสมาชิกในตำแหน่ง (1,5) ของเมทริกซ์ P_5 ซึ่งพบว่า $P_5(1,5) = 2$ แสดงว่าวิถีต้องเริ่มจากบัพ V_1 ไปยังบัพ V_2 ก่อนไปหาบัพ V_5 ดังนั้น path = $\{V_1, V_2\}$ ต่อมาให้พิจารณาตำแหน่ง (2,5) ของเมทริกซ์ P_5 ซึ่งพบว่า $P_5(2,5) = 0$ แสดงว่า V_2 และ V_5 เป็นบัพประชิดกันที่ให้ค่าถ่วงน้ำหนักรวมต่ำที่สุด ดังนั้น path = $\{V_1, V_2, V_5\}$ โดยค่าน้ำหนักถ่วงรวมเท่ากับค่าสมาชิกในตำแหน่ง (1,5) ของเมทริกซ์ A นั่นคือ 3

ผู้อ่านสามารถสังเกตได้ว่า ผลรวมของค่าน้ำหนักถ่วงที่เกิดขึ้นในแต่ละคู่บัพในวิถีมีค่าเท่ากับค่าถ่วงน้ำหนักรวมที่ได้ในตำแหน่ง (1, 5) นั่นคือ $A(1,2) + A(2,5) = 1 + 2 = 3$

ตัวอย่างที่ 4 พิจารณาไดกราฟในรูปที่ 3.9



รูปที่ 3.9 ไดกราฟตัวอย่างสำหรับการคำนวณในขั้นตอนวิธีของ Floyd-Warshall

จากกราฟที่ 3.9 นำมาเขียนเป็นเมทริกซ์ A_0 และ P_0 ดังนี้

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ \infty & 0 & \infty & \infty & 10 \\ 1 & 1 & 0 & 2 & \infty \\ \infty & 1 & 3 & 0 & 100 \\ \infty & 1 & \infty & 1 & 0 \end{bmatrix} \\ A_0 \end{array} \quad \text{และ} \quad \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ P_0 \end{array}$$

<u>ขั้นตอนที่1</u>

เมื่อ k = 1 ให้ตัดแถวที่ 1 หลักที่ 1 ของเมทริกซ์ A_0 ดังเส้นประ

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ \infty & 0 & \infty & \infty & 10 \\ 1 & 1 & 0 & 2 & \infty \\ \infty & 1 & 3 & 0 & 100 \\ \infty & 1 & \infty & 1 & 0 \end{bmatrix} \end{array}$$

$$A_0$$

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้นด้วยเงื่อนไขข้อที่ 3.3 ดังนี้

$A_1[2, 2] = \min(A_0[2, 2],\ A_0[2, 1] + A_0[1, 2]) = \min(0,\ \infty+3) = 0$

$A_1[2, 3] = \min(A_0[2, 3],\ A_0[2, 1] + A_0[1, 3]) = \min(\infty,\ \infty+1) = \infty$

$A_1[2, 4] = \min(A_0[2, 4],\ A_0[2, 1] + A_0[1, 4]) = \min(\infty,\ \infty+\infty) = \infty$

$A_1[2, 5] = \min(A_0[2, 5],\ A_0[2, 1] + A_0[1, 5]) = \min(10,\ \infty+\infty) = 10$

$A_1[3, 2] = \min(A_0[3, 2],\ A_0[3, 1] + A_0[1, 2]) = \min(1,\ 1+3) = 1$

$A_1[3, 3] = \min(A_0[3, 3],\ A_0[3, 1] + A_0[1, 3]) = \min(0,\ 1+1) = 0$

$A_1[3, 4] = \min(A_0[3, 4],\ A_0[3, 1] + A_0[1, 4]) = \min(2,\ 1+\infty) = 2$

$A_1[3, 5] = \min(A_0[3, 5],\ A_0[3, 1] + A_0[1, 5]) = \min(\infty,\ 1+\infty) = \infty$

$A_1[4, 2] = \min(A_0[4, 2],\ A_0[4, 1] + A_0[1, 2]) = \min(1,\ \infty+3) = 1$

$A_1[4, 3] = \min(A_0[4, 3],\ A_0[4, 1] + A_0[1, 3]) = \min(3,\ \infty+1) = 3$

$A_1[4, 4] = \min(A_0[4, 4],\ A_0[4, 1] + A_0[1, 4]) = \min(0,\ \infty+\infty) = 0$

$A_1[4, 5] = \min(A_0[4, 5],\ A_0[4, 1] + A_0[1, 5]) = \min(100,\ \infty+\infty) = 100$

$A_1[5, 2] = \min(A_0[5, 2],\ A_0[5, 1] + A_0[1, 2]) = \min(1,\ \infty+3) = 1$

$A_1[5, 3] = \min(A_0[5, 3],\ A_0[5, 1] + A_0[1, 3]) = \min(\infty,\ \infty+1) = \infty$

$A_1[5, 4] = \min(A_0[5, 4],\ A_0[5, 1] + A_0[1, 4]) = \min(1, \infty+\infty) = 1$

$A_1[5, 5] = \min(A_0[5, 5],\ A_0[5, 1] + A_0[1, 5]) = \min(0, \infty+\infty) = 0$

ในรอบนี้ เมทริกซ์ A ไม่มีการเปลี่ยนแปลง จึงเริ่มคำนวณในรอบถัดไป

<u>**ขั้นตอนที่2**</u>

เมื่อ k = 2 ให้ตัดแถวที่ 2 หลักที่ 2 ของเมทริกซ์ A_1 ดังเส้นประ

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & \infty & \infty \\ \infty & 0 & \infty & \infty & 10 \\ 1 & 1 & 0 & 2 & \infty \\ \infty & 1 & 3 & 0 & 100 \\ \infty & 1 & \infty & 1 & 0 \end{bmatrix} \end{array}$$

A_1

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้นด้วยเงื่อนไข ดังนี้

$A_1\,[1, 1] = \min(A_0\,[1, 1],\ A_0\,[1, 2] + A_0\,[2, 1]) = \min(0, 3+\infty) = 0$

$A_1\,[1, 3] = \min(A_0\,[1, 3],\ A_0\,[1, 2] + A_0\,[2, 3]) = \min(1, 3+\infty) = 1$

$A_1\,[1, 4] = \min(A_0\,[1, 4],\ A_0\,[1, 2] + A_0\,[2, 4]) = \min(\infty, 3+\infty) = \infty$

$A_1\,[1, 5] = \min(A_0\,[1, 5],\ A_0\,[1, 2] + A_0\,[2, 5]) = \min(\infty, 3+10) = 13,\ P_2\,[1, 5] = 2$

$A_1\,[3, 1] = \min(A_0\,[3, 1],\ A_0\,[3, 2] + A_0\,[2, 1]) = \min(1, 1+\infty) = 1$

$A_1\,[3, 3] = \min(A_0\,[3, 3],\ A_0\,[3, 2] + A_0\,[2, 3]) = \min(0, 1+\infty) = 0$

$A_1\,[3, 4] = \min(A_0\,[3, 4],\ A_0\,[3, 2] + A_0\,[2, 4]) = \min(2, 1+\infty) = 2$

$A_1\,[3, 5] = \min(A_0\,[3, 5],\ A_0\,[3, 2] + A_0\,[2, 5]) = \min(\infty, 1+10) = 11,\ P_2\,[3, 5] = 2$

$A_1\,[4, 1] = \min(A_0\,[4, 1],\ A_0\,[4, 2] + A_0\,[2, 1]) = \min(\infty, 1+\infty) = \infty$

$A_1\,[4, 3] = \min(A_0\,[4, 3],\ A_0\,[4, 2] + A_0\,[2, 3]) = \min(3, 1+\infty) = 3$

$A_1\,[4, 4] = \min(A_0\,[4, 4],\ A_0\,[4, 2] + A_0\,[2, 4]) = \min(0, 1+\infty) = 0$

$A_1\,[4, 5] = \min(A_0\,[4, 5],\ A_0\,[4, 2] + A_0\,[2, 5]) = \min(100, 1+10) = 11,\ P_2\,[4, 5] = 2$

$A_1\,[5, 1] = \min(A_0\,[5, 1],\ A_0\,[5, 2] + A_0\,[2, 1]) = \min(\infty, 1+\infty) = \infty$

$A_1\,[5, 3] = \min(A_0\,[5, 3],\ A_0\,[5, 2] + A_0\,[2, 3]) = \min(\infty, 1+\infty) = \infty$

$A_1\,[5, 4] = \min(A_0\,[5, 4],\ A_0\,[5, 2] + A_0\,[2, 4]) = \min(1, 1+\infty) = 1$

$A_1\,[5, 5] = \min(A_0\,[5, 5],\ A_0\,[5, 2] + A_0\,[2, 5]) = \min(0, 1+10) = 0$

แล้วทำการปรับค่าสมาชิกในเมทริกซ์ย่อยที่ได้นั้น จะได้เมทริกซ์ A_2 และ P_2 ดังนี้

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & \infty & 13 \\ \infty & 0 & \infty & \infty & 10 \\ 1 & 1 & 0 & 2 & 11 \\ \infty & 1 & 3 & 0 & 11 \\ \infty & 1 & \infty & 1 & 0 \end{bmatrix} \\ A_2 \end{array} \quad \text{และ} \quad \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ P_2 \end{array}$$

หลังจากทำตามขั้นตอนวิธี ดังแสดงในตารางที่ 2 จนกระทั่ง $k = 5$ จะได้

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 3 & 12 \\ 15 & 0 & 14 & 11 & 10 \\ 1 & 1 & 0 & 2 & 11 \\ 4 & 1 & 3 & 0 & 11 \\ 5 & 1 & 4 & 1 & 0 \end{bmatrix} \\ A_5 \end{array} \quad \text{และ} \quad \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array}\begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{bmatrix} 0 & 3 & 0 & 3 & 3 \\ 5 & 0 & 5 & 5 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 3 & 0 & 0 & 0 & 2 \\ 4 & 0 & 4 & 0 & 0 \end{bmatrix} \\ P_5 \end{array}$$

หากต้องการหาวิถีที่ต่ำที่สุดจากบัพ V_1 ไปยังบัพ V_5 ให้พิจารณาตามขั้นตอนดังนี้ ให้บัพเริ่มต้นถูกเก็บไว้ในตัวแปร path นั่นคือ path = $\{V_1\}$ แล้วพิจารณาสมาชิกในตำแหน่ง (1,5) ของเมทริกซ์ P_5 ซึ่งพบว่า $P_5(1,5) = 3$ แสดงว่าวิถีต้องเริ่มจากบัพ V_1 ไปยังบัพ V_3 ก่อนไปยังบัพ V_5 ดังนั้น path = $\{V_1, V_3\}$ ต่อมาให้พิจารณาตำแหน่ง (3,5) ของเมทริกซ์ P_5 ซึ่งพบว่า $P_5(3,5) = 2$ แสดงว่าวิถีจากบัพ V_3 ไปยับบัพ V_5 ต้องผ่านบัพ V_2 ดังนั้น path = $\{V_1, V_3, V_2\}$ จากนั้นพิจารณาตำแหน่ง (2,5) ของเมทริกซ์ P_5 ซึ่งพบว่า $P_5(2,5) = 0$ แสดงว่า V_2 และ V_5 เป็นบัพประชิดกันที่ให้ค่าถ่วงน้ำหนักรวมต่ำที่สุด ดังนั้น path = $\{V_1, V_3, V_2, V_5\}$ โดยค่าน้ำหนักถ่วงรวมเท่ากับค่าสมาชิกในตำแหน่ง (1,5) ของเมทริกซ์ A นั่นคือ 12

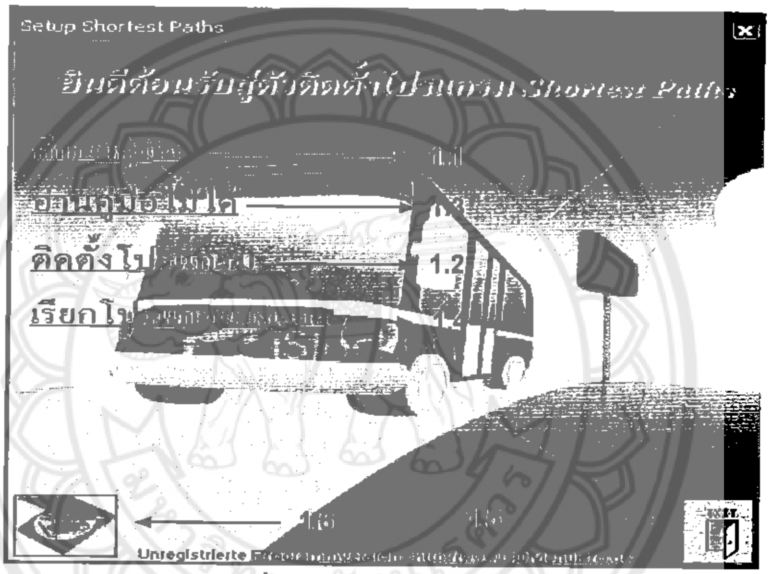
ผู้อ่านสามารถสังเกตได้ว่า ผลรวมของค่าน้ำหนักถ่วงที่เกิดขึ้นในแต่ละคู่บัพในวิถีมีค่าเท่ากับค่าถ่วงน้ำหนักรวมที่ได้ในตำแหน่ง (1,5) นั่นคือ $A(1,3) + A(3,2) + A(2,5) = 1 + 1 + 10 = 12$

บทที่4
การใช้งานโปรแกรม

การใช้งานโปรแกรมจะแบ่งออกเป็น 3 ส่วนดังนี้

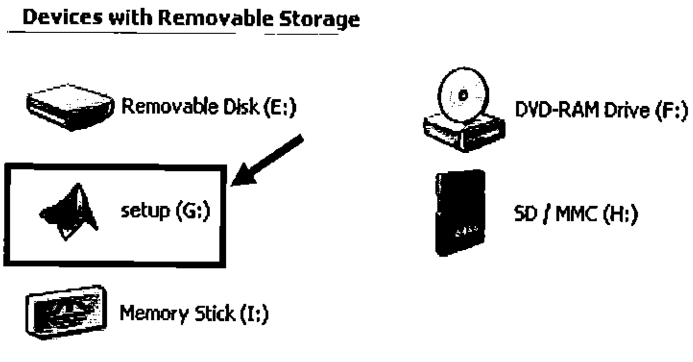
<u>ส่วนที่ 1 การติดตั้งโปรแกรม</u>

ให้ผู้ใช้นำแผ่น Setup ใส่ลงในเครื่องแล้วโปรแกรมจะทำงานแบบ Auto run เมื่อโปรแกรมทำงานจะเห็นว่ามีหน้าต่างของส่วนประต่อสานกราฟิกและมีฟังก์ชั่นให้เลือกใช้ 5 ฟังก์ชั่น



รูปที่ 4.1 การทำงาน Auto run

กรณีที่แผ่นโปรแกรมไม่ทำการ Auto run ให้ผู้ใช้เข้าไปที่ My Computer แล้ว double click ตรง cd - rom ที่มีรูป 

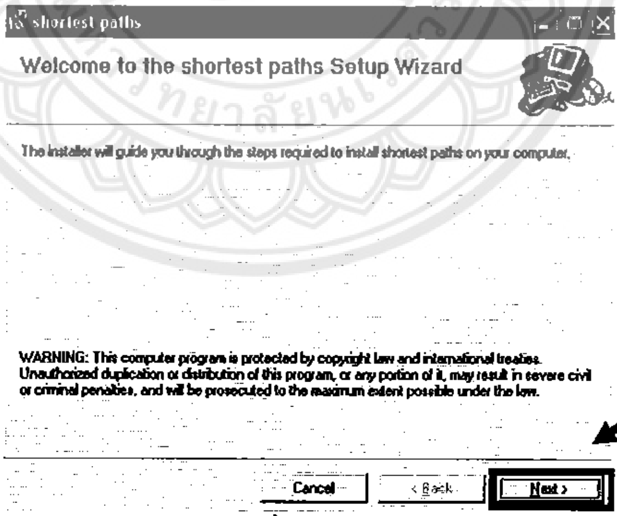


รูปที่ 4.2 เลือก click จาก icon โดยตรง

เมื่อโปรแกรมทำงานแล้วขั้นตอนแรกให้ผู้ใช้ Click ติดตั้งโปรแกรมเพื่อเป็นการเริ่มต้นใช้งานโปรแกรมทั้งหมด

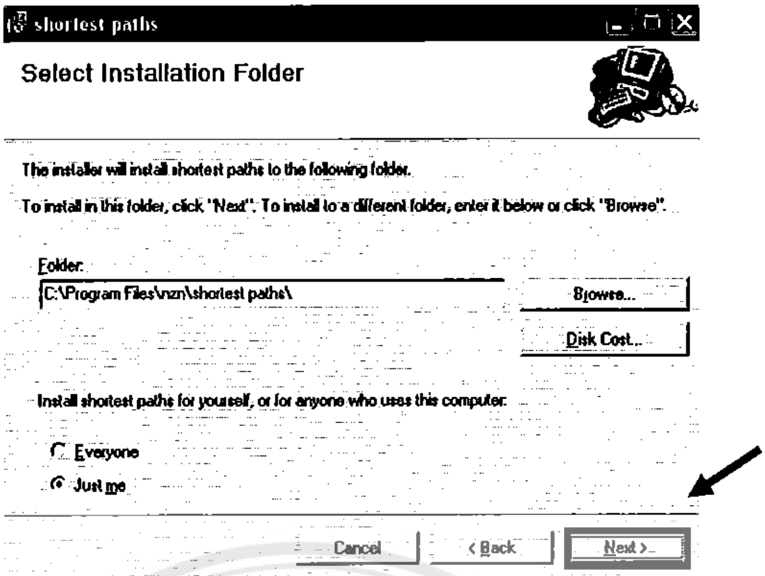
รูปที่ 4.3 ติดตั้งโปรแกรม

ให้ผู้ใช้ Click next เพื่อทำการติดตั้งโปรแกรม



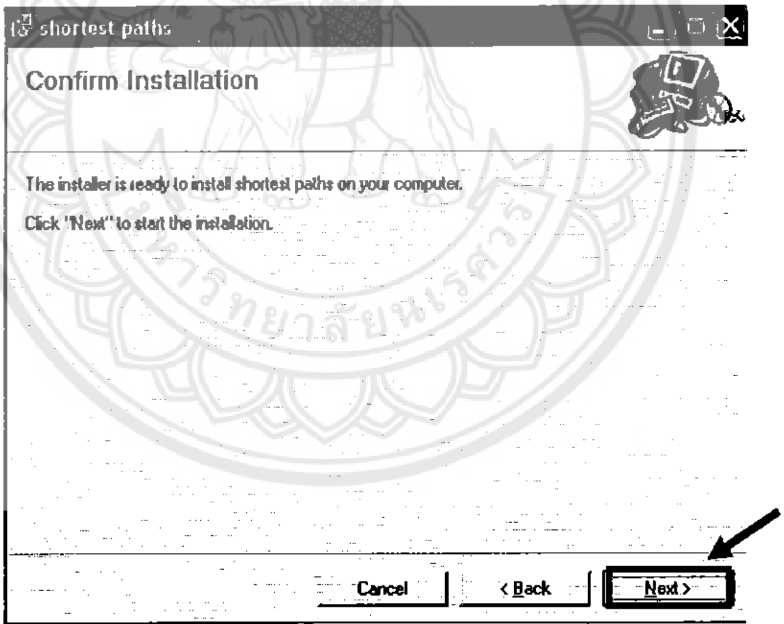
รูปที่ 4.4 setup

ให้ผู้ใช้ Click next โดยไม่ต้องปรับค่าใดๆเพราะตัวติดตั้งจะสร้าง directory ในเครื่องคอมพิวเตอร์ให้กับผู้ใช้งานเอง



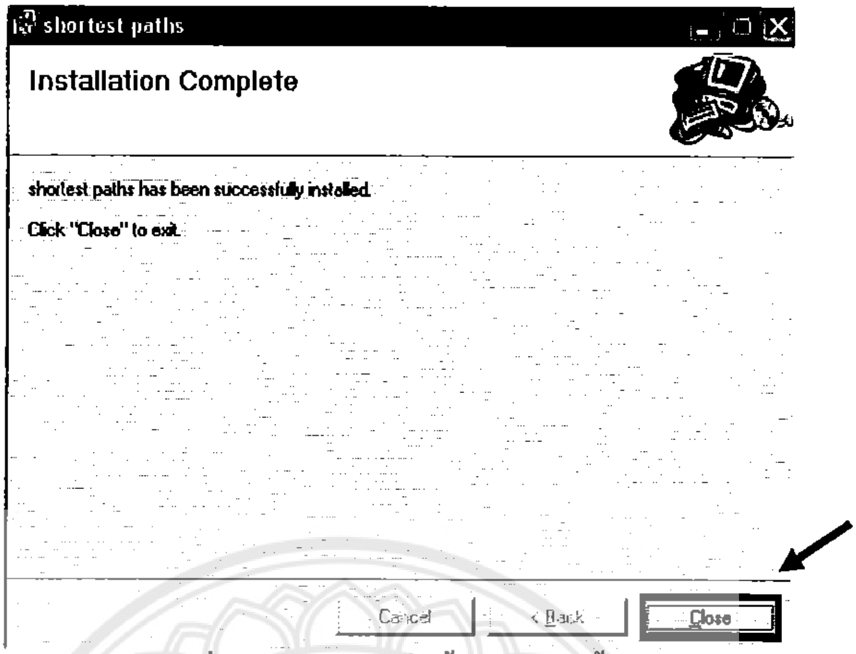
รูปที่ 4.5 click next

ให้ผู้ใช้ Click next เพื่อยืนยันครั้งสุดท้าย



รูปที่ 4.6 click next เพื่อยืนยันว่าจะติดตั้ง

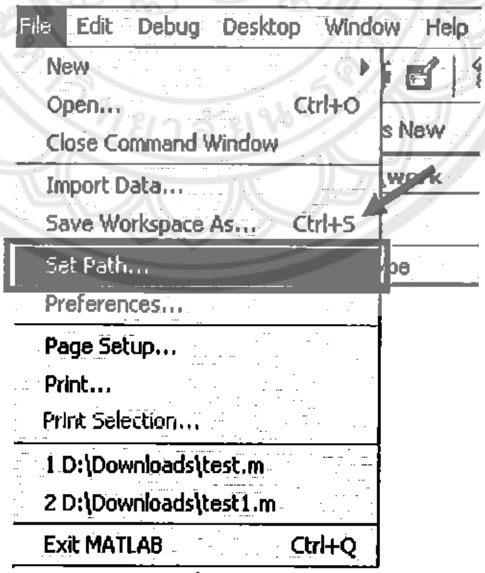
ให้ผู้ใช้ Click close เพื่อสิ้นสุดการติดตั้งโปรแกรม



รูปที่ 4.7 click close เพื่อสิ้นสุดการติดตั้งโปรแกรม

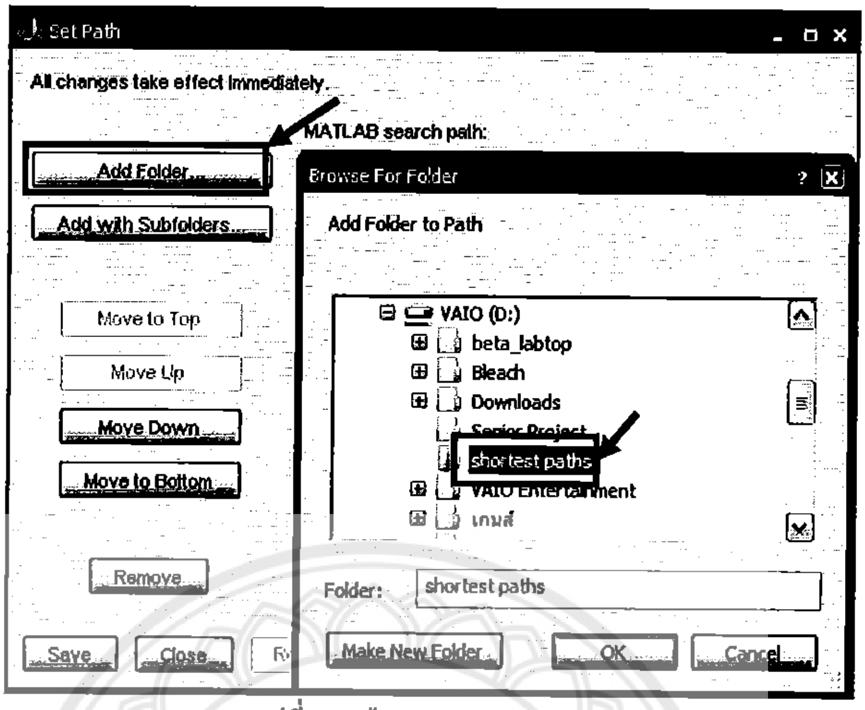
เมื่อติดตั้งโปรแกรมเสร็จแล้วให้ผู้ใช้งานเข้าไปตรวจสอบใน Directory ของ D:\ shortest Path\ ว่ามีโปรแกรมที่ได้ติดตั้งหรือไม่

เข้าโปรแกรม MATLAB ไป set path...

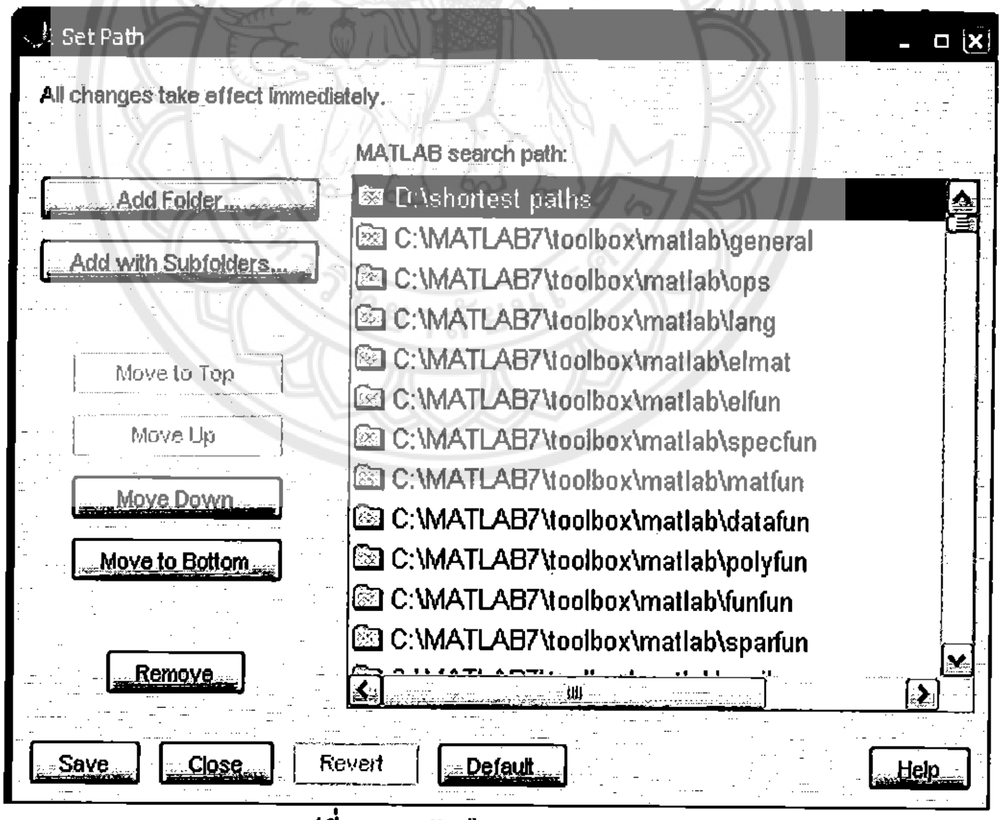


รูปที่ 4.8 set path

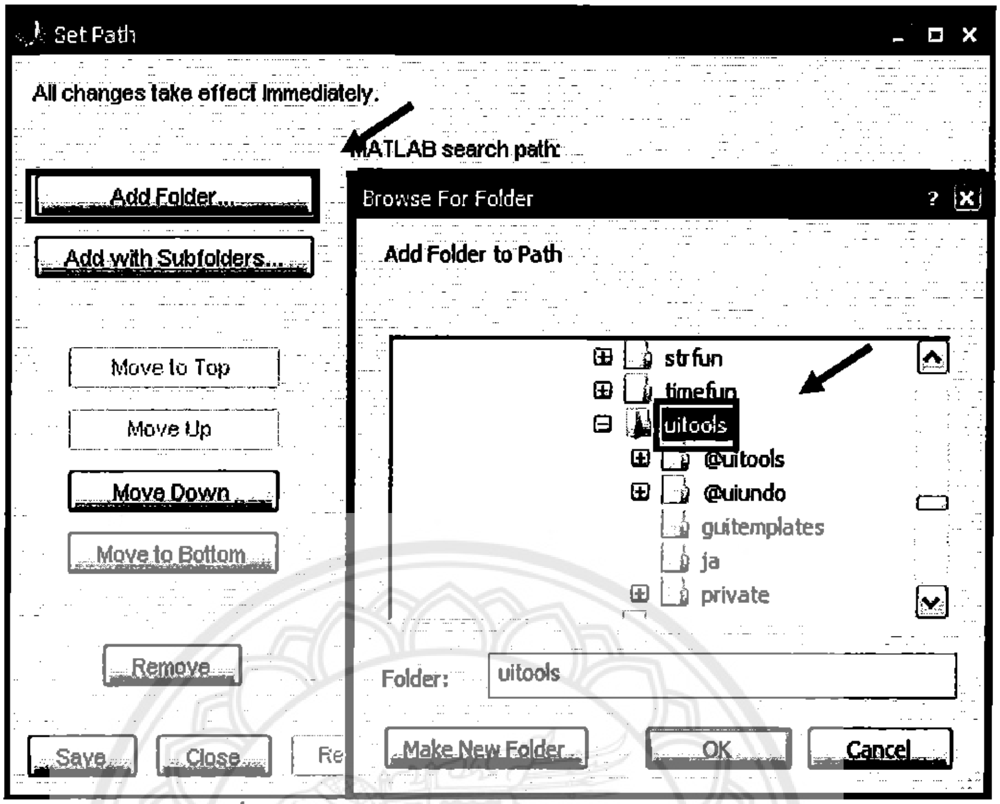
เข้าไปที่ Add Folder โดยเลือกนำ Folder เข้ามาอยู่ 2 Folder คือ D:\shortest paths และ C:\MATLAB7\toolbox\matlab\uitools



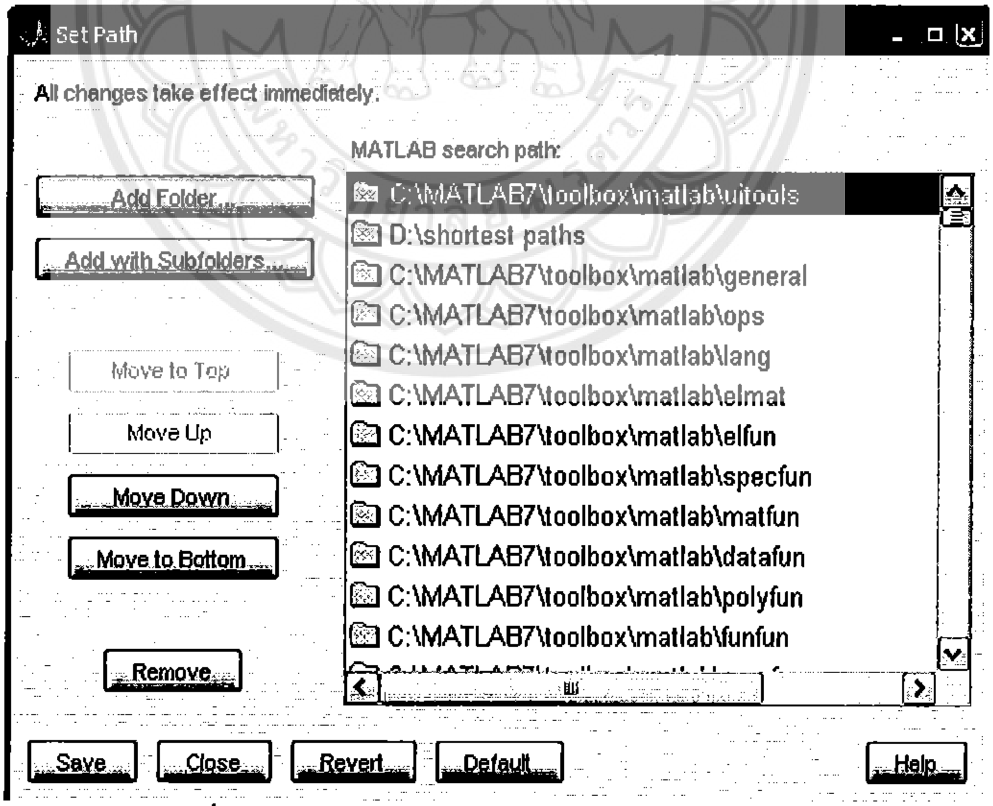
รูปที่ 4.9 เลือก D:\shortest paths



รูปที่ 4.10 หลังเลือก D:\shortest paths



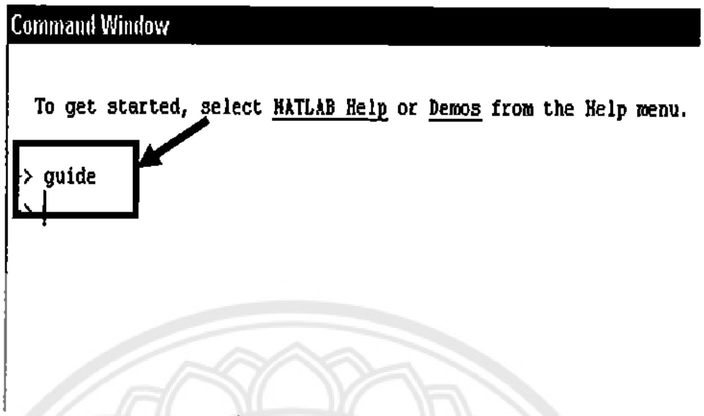
รูปที่ 4.11 เลือก C:\MATLAB7\toolbox\matlab\uitools



รูปที่ 4.12 หลังเลือก C:\MATLAB7\toolbox\matlab\uitools

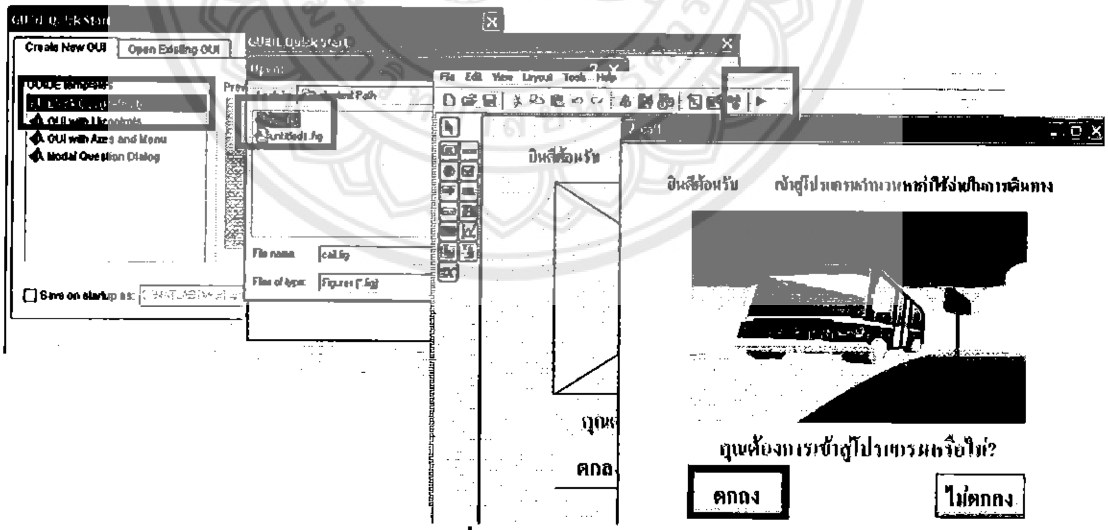
ส่วนที่ 2 การใช้งานโปรแกรม

เปิดโปรแกรม MATLAB ขึ้นมาไปที่หน้าต่าง command แล้วพิมพ์คำสั่ง guide เพื่อเรียกหน้าต่าง Guide Quick Start



รูปที่ 4.13 การใช้คำสั่ง guide

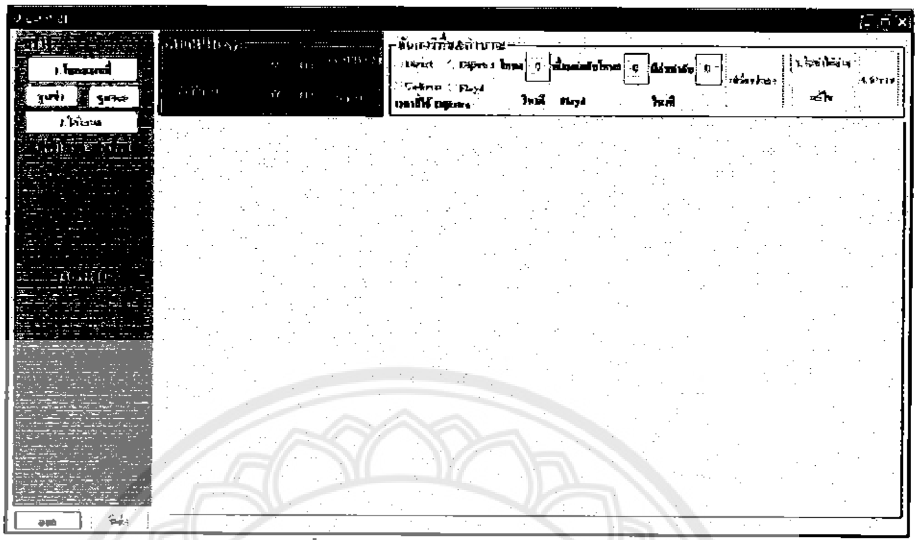
ให้ผู้ใช้งานพิมพ์คำสั่ง guide ที่ได้ใส่ไว้ใน command แล้วเลือกไฟล์ call.fig โดย เลือกที่ open existing gui แล้วกดฟังก์ชั่น Browse แล้วเลือกที่ D:\ shortest Path\ กดตกลงเพื่อทำการประมวลผลโปรแกรม โดยกดฟังก์ชั่นสามเหลี่ยมสีเขียว และกดตกลงเพื่อทำการโหลดโปรแกรมขึ้นมาใช้



รูปที่ 4.14 หน้าต่าง Call

รูปที่ 4.15 โหลดโปรแกรม

เมื่อโหลดโปรแกรมเสร็จสิ้นจะปรากฏส่วนต่อประสานกราฟิกของโปรแกรมที่พร้อมจะรับคำสั่งจากผู้ใช้งาน



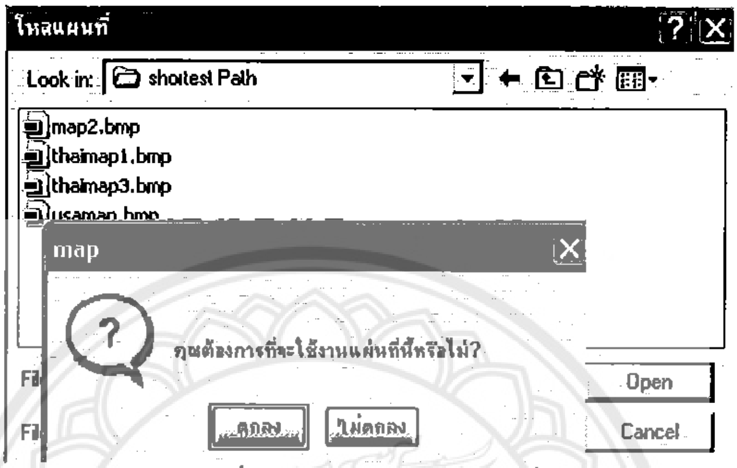
รูปที่ 4.16 ส่วนต่อประสานกราฟิก

มีฟังก์ชั่นให้เลือกหลายฟังก์ชั่น

- ฟังก์ชั่นโหลดแผนที่ ใช้ในการนำแผนที่เข้ามายังตัวโปรแกรม
- ฟังก์ชั่นซูมเข้า และ ซูมออก ใช้ในการขยายและดูแผนที่
- ฟังก์ชั่นใส่บัพ ใช้ในการสร้างบัพ คือให้คลิกใส่บัพในแผนที่ และจะสิ้นสุดก็ต่อเมื่อคลิกขวา (การคลิกขวาหมายถึงสร้างบัพสุดท้ายแล้วหยุดการสร้างบัพ)
- ฟังก์ชั่นเลือกขั้นตอนวิธี Dijkstra และ Floyd-Warshall สามารถเลือกขั้นตอนวิธีได้ทีละขั้นตอนวิธีได้
- ฟังก์ชั่นเปลี่ยนขั้นตอนวิธี ใช้ในการเปลี่ยนขั้นตอนวิธี Dijkstra เป็น Floyd - Warshall หรือ Floyd - Warshall เป็น Dijkstra จะใช้งานฟังก์ชั่นนี้ได้ต้องประมวลผลเสร็จสิ้นก่อน
- ฟังก์ชั่นใส่ค่าน้ำหนักถ่วง ใช้ในการใส่ค่าน้ำหนักถ่วง ระหว่างบัพ
- ฟังก์ชั่นแก้ไข ใช้ในการแก้ไขค่าน้ำหนักถ่วง เมื่อใส่ผิด
- ฟังก์ชั่นคำนวณ ใช้ในการคำนวณเพื่อหาระยะทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดสุดท้าย
- ฟังก์ชั่นออก ใช้ในการออกจากโปรแกรมเมื่อเลิกใช้โปรแกรมแล้ว
- ฟังก์ชั่นรีเซ็ต ใช้ในการ clear ค่าของตัวแปรต่างๆในโปรแกรม

ส่วนที่ 2.1 การสร้างบัพ

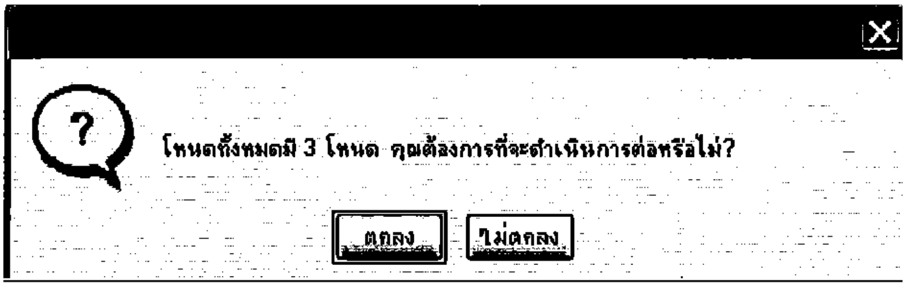
กดฟังก์ชั่นที่ 1 เพื่อโหลดแผนที่ และโปรแกรมจะถามว่า คุณต้องการที่จะใช้งานแผนที่นี้หรือไม่ ถ้าผู้ใช้งานต้องการใช้งานกด ตกลง ถ้าผู้ใช้ไม่ต้องการใช้งานกด ไม่ตกลง เมื่อกดไม่ตกลง โปรแกรมจะวนลูปให้โหลดแผนที่ใหม่จนกว่าผู้ใช้จะกด ตกลง เพื่อออกลูป



รูปที่ 4.17 หน้าต่างโหลดแผนที่

กดฟังก์ชั่น 2 เพื่อสร้างบัพ สร้างได้ไม่เกิน 20 บัพในฟังก์ชั่นนี้จะมีฟังก์ชั่น ลด และ เพิ่มบัพด้วย การสร้างบัพ คือให้ คลิกใส่บัพ ในสิ่งแวดล้อมที่อยู่บนแผนที่ และจะสิ้นสุดก็ต่อเมื่อคลิกขวา (การคลิกขวาหมายถึงสร้างบัพสุดท้ายแล้วหยุดการสร้างบัพ) เมื่อคลิกขวาโปรแกรมจะถามว่า คุณมี 3 บัพคุณต้องการดำเนินงานต่อหรือไม่ กดตกลงเพื่อ ไปใส่ค่าน้ำหนักถ่วง กดไม่ตกลงเพื่อทำการ <u>แก้ไข</u>

หมายเหตุ โดยข้อกำหนดของโปรแกรม บัพจะมีสีเขียว แต่ถ้าผู้ใช้ประสงค์จะเปลี่ยนสีของบัพ ให้เลือกสีจากเมนูด้านซ้ายมือของหน้าจอ ก่อนกดเม้าส์เพื่อสร้างบัพใน Axes และหลังจากการประมวลผลเพื่อหาวิถีที่สั้นที่สุดแล้ว หากผู้ใช้เลือกใช้ขั้นตอนวิธีของ Dijkstra วิถีที่สั้นที่สุดจะแสดงเป็นเส้นสีดำ และหากผู้ใช้เลือกใช้ขั้นตอนวิธีของ Floyd-Warshall วิถีที่สั้นที่สุดจะแสดงเป็นเส้นสีแดง หากผู้ใช้ประสงค์จะเปลี่ยนสี สามารถทำได้โดยการเลือกสีจากเมนูด้านซ้ายมือของหน้าจอก่อนกดปุ่ม 4.คำนวณ



รูปที่ 4.18 บอกจำนวนบัพและถามผู้ใช้ว่าจะดำเนินการต่อหรือไม่

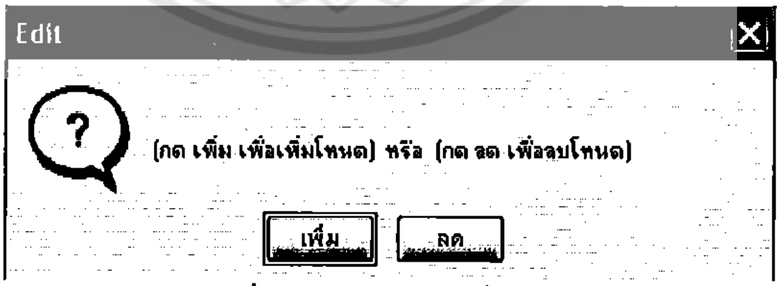
ถ้าต้องการเพิ่มบัพหรือลดบัพ ให้กด ฟังก์ชั่นไม่ตกลง

รูปที่ 4.19 เลือกไม่ตกลงเพื่อทำการเพิ่มหรือลดบัพ

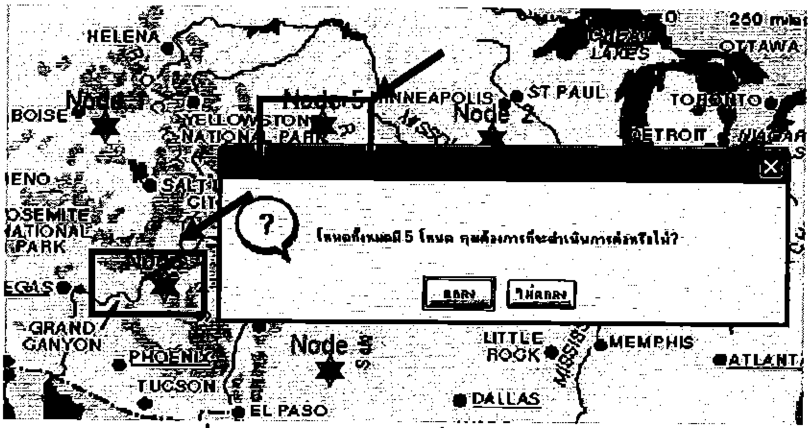
<u>**เพิ่ม – ลด** บัพ</u>

การเพิ่มบัพ คือการคลิกซ้ายธรรมดา

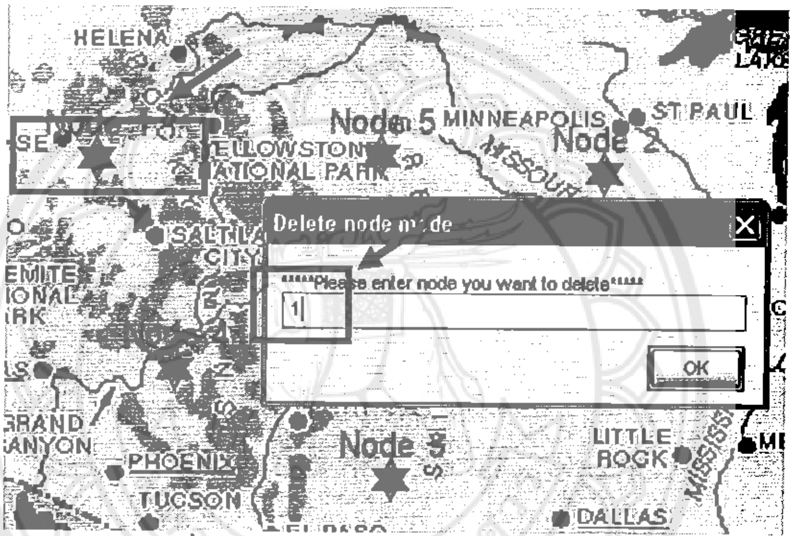
การลดบัพ คือให้ใส่ชื่อของบัพที่ต้องการจะลบเช่นลบบัพ 3 ใส่หมายเลข 3 ในช่องที่โปรแกรมจัดให้ และโปรแกรมจะวนให้ผู้ใช้แก้ไขได้ตามต้องการ จนกว่าผู้ใช้จะกดตกลง



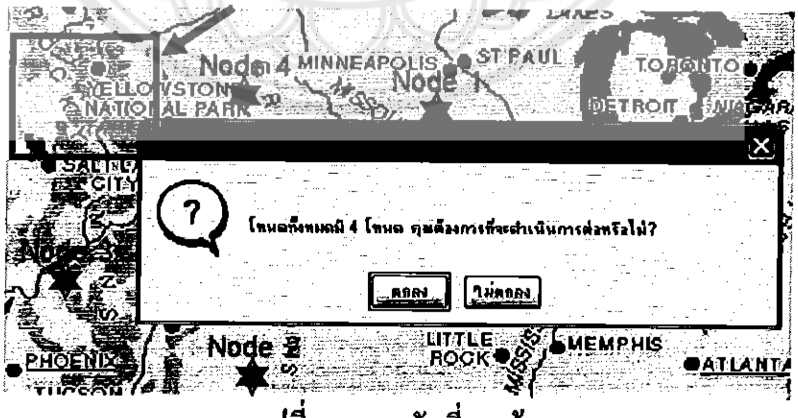
รูปที่ 4.20 เมนูการเลือกเพิ่ม – ลด



รูปที่ 4.21 เมนูการเลือกเพิ่มบัพ 4 และ บัพ 5

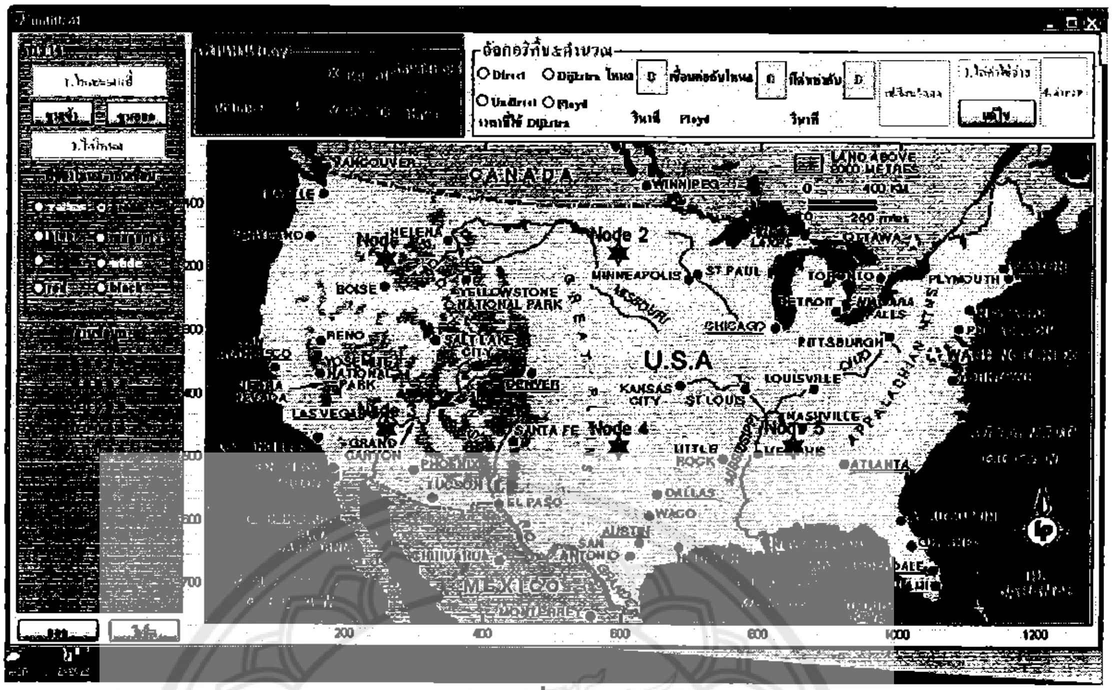


รูปที่ 4.22 เมนูการเลือกลดและใส่บัพที่จะลบ



รูปที่ 4.23 ลบบัพที่ 1 แล้ว

หมายเหตุ เมื่อลบบัพแล้วลำดับของบัพจะเปลี่ยนไปจากเดิม โดยทำการเลื่อนลำดับถัดไปขึ้นมาแทนที่ เช่นมี 4 บัพ คือ บัพ1 บัพ2 บัพ3 และ บัพ4 หากมีการลบบัพ1 ไปแล้วจะมีการเปลี่ยนลำดับของบัพใหม่คือ บัพ2 จะเป็น บัพ 1, บัพ3 จะเป็น บัพ 2, บัพ4 จะเป็น บัพ 3 โดยทันที



รูปที่ 4.24 สร้างบัพ

ส่วนที่ 2.2 ปรับเปลี่ยนขั้นตอนวิธี

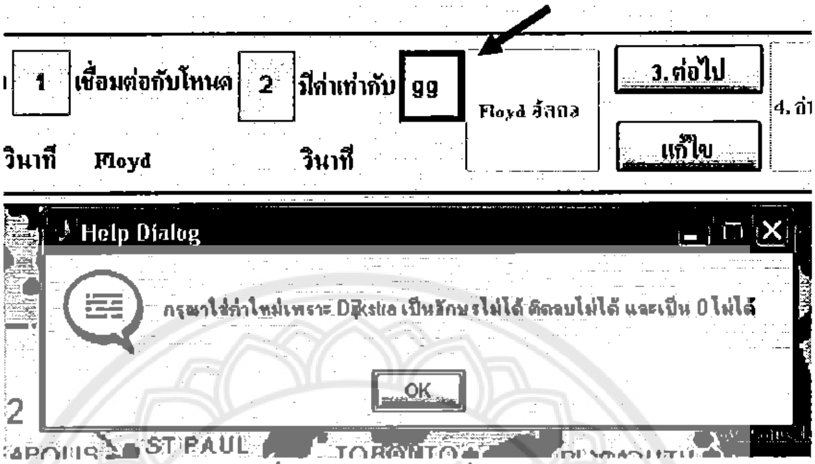
ถัดมา ให้เลือกชนิดของกราฟซึ่งมีอยู่ 2 ชนิดคือ ไดกราฟ (Direct) และ **กราฟ (Undirect)** **แล้วตาม**ด้วยการเลือกขั้นตอนวิธีซึ่งมีอยู่ 2 วิธีคือ Dijkstra และ Floyd จากนั้นทำการใส่ค่าน้ำหนักถ่วง **ใน**บริเวณที่ใส่ค่าน้ำหนักถ่วงนี้ จะไม่สามารถใส่ค่าได้หากผู้ใช้ยังไม่ได้เลือก**ขั้นตอนวิธี**

รูปที่ 4.25 ส่วนต่อประสานกราฟที่เลือกขั้นตอนวิธีและใส่ค่าน้ำหนักถ่วง

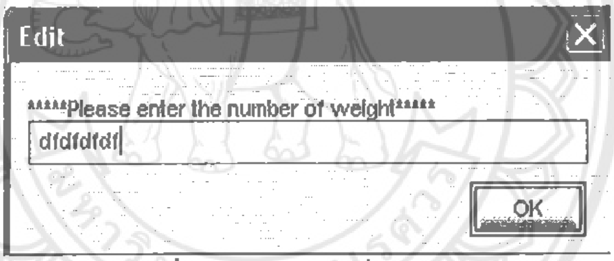
จากรูปที่ 4.25 ผู้ใช้งานจะเห็นสี่เหลี่ยมสามสีที่ต่างกัน สีเขียวไว้เลือก ชนิดของกราฟ และขั้นตอนวิธีที่ต้องการใช้ สี่เหลี่ยมสีแดงบอกความความสัมพันธ์ของบัพที่เชื่อมต่อกัน สี่เหลี่ยมสีฟ้าบอกการใส่ค่าน้ำหนักถ่วง

ส่วนที่ 2.3 การใส่ค่าน้ำหนักถ่วง และ การแก้ไขค่าน้ำหนักถ่วง

ให้ผู้ใช้งานใส่ค่าน้ำหนักถ่วงในช่อง สี่เหลี่ยมสีฟ้าบอกการใส่ค่าน้ำหนักถ่วง รูปที่ 4.25 เมื่อใส่ค่าน้ำหนักถ่วง ผิด โปรแกรมจะวนลูปให้ใส่ค่าใหม่จนผู้ใช้ใส่ค่าถูกต้องคือ ตัวเลข และมีตัวอักษรที่ยกเว้นก็คือ inf หมายความว่าไม่มีการเชื่อมต่อระหว่างบัพ



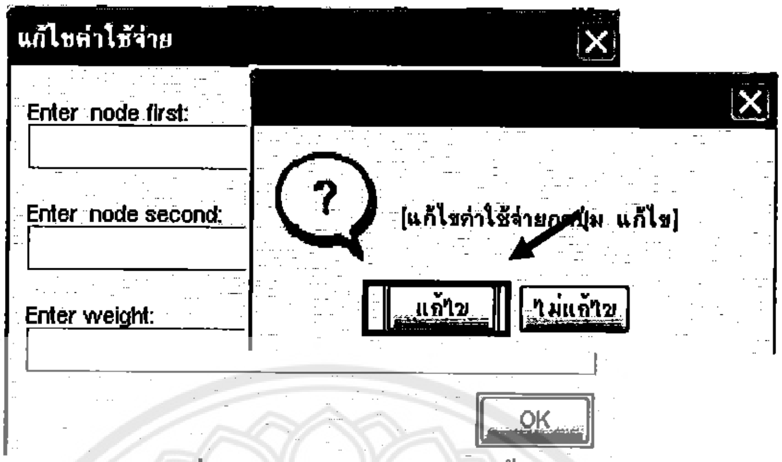
รูปที่ 4.26 แจ้งเตือนเมื่อใส่ค่าผิด



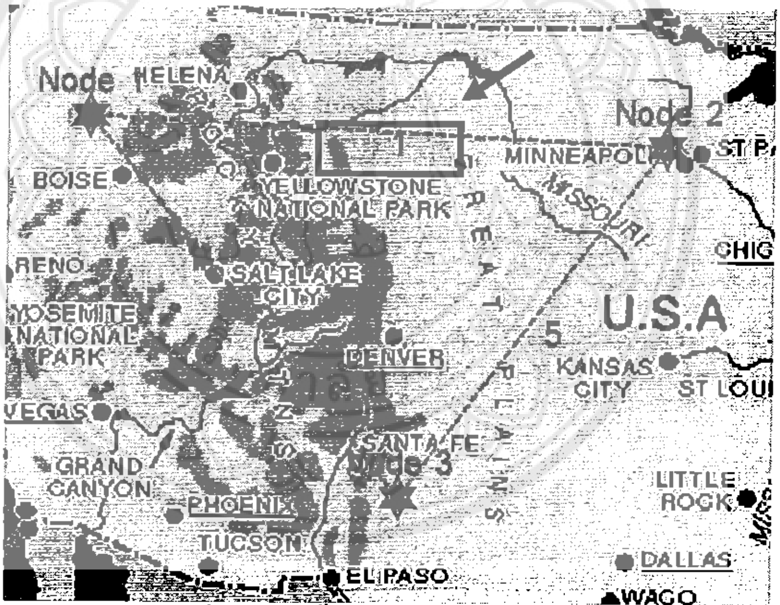
รูปที่ 4.27 แจ้งเตือนเมื่อใส่ค่าผิด

รูปที่ 4.28 ใส่ค่าน้ำหนักถ่วง ถูกต้อง

เมื่อมีการใส่ค่าใช้จ่ายไปแล้วผู้ใช้สามารถแก้ไขและปรับปรุงค่าน้ำหนักถ่วงต่างๆ ได้โดยตัวผู้ใช้เอง โดยการกดที่ฟังก์ชั่น แก้ไข แล้วโปรแกรมจะให้ผู้ใช้ใส่บัพที่ต้องการแก้ไข

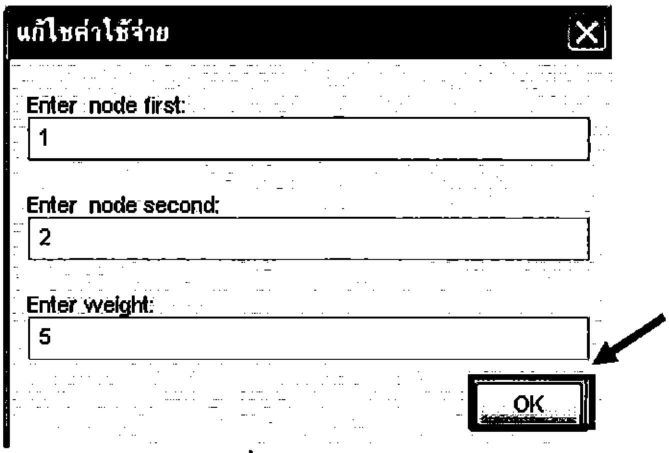


รูปที่ 4.29 ตัวอย่างการแก้ไขค่าน้ำหนักถ่วง

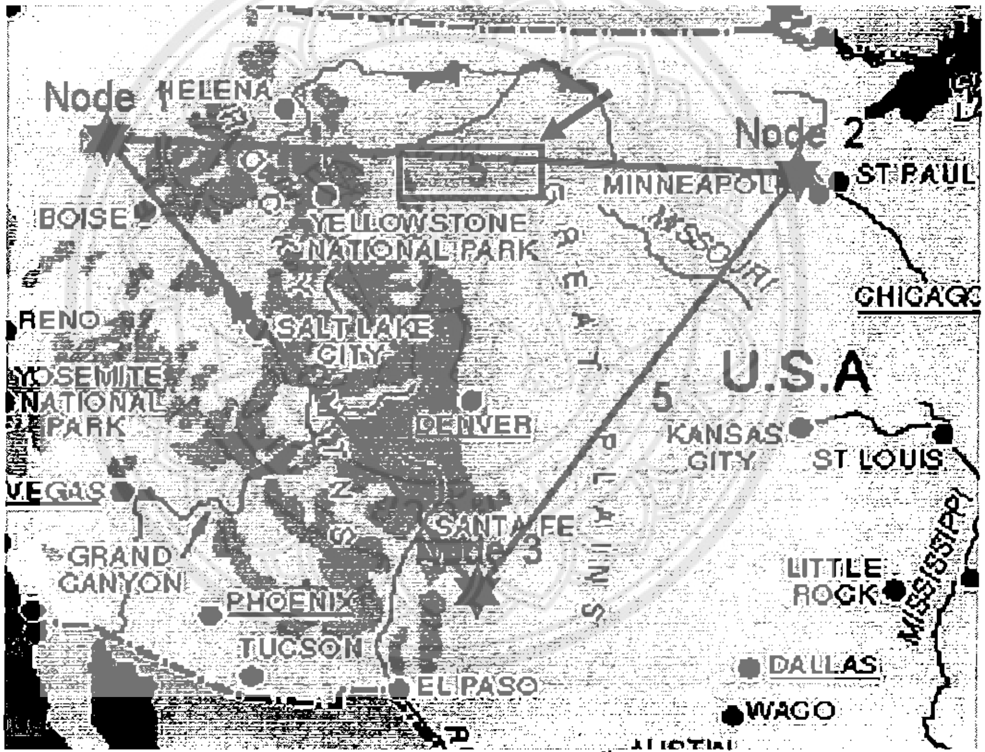


รูปที่ 4.30 ค่าน้ำหนักถ่วงที่ต้องการแก้ไข

จากรูปที่ 4.31 ช่องที่ชื่อว่า **Enter node first และ Enter node second** จะให้ใส่ชื่อบัพ ส่วนช่องที่ชื่อ **Enter weight** จะให้ผู้ใช้ใส่ค่าน้ำหนักถ่วง ที่จะแก้ไขต่อไป ตัวอย่างเช่น ผู้ใช้ต้องการแก้ไขค่าน้ำหนักถ่วง ระหว่างบัพ 1 กับบัพ 2 มีค่าน้ำหนักถ่วง เป็น 5 ผู้ใช้ต้องกรอกดังนี้

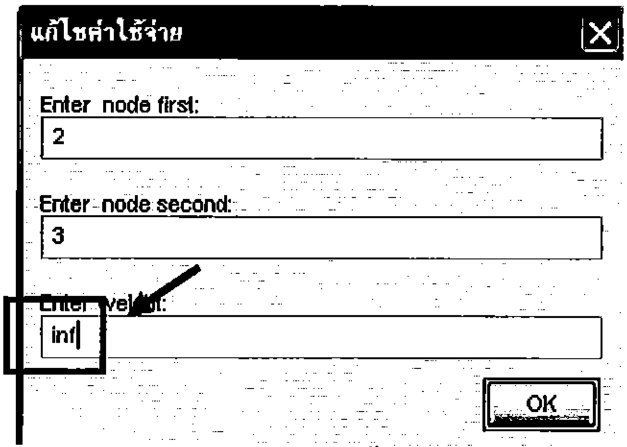


รูปที่ 4.31 แก้ไขค่า



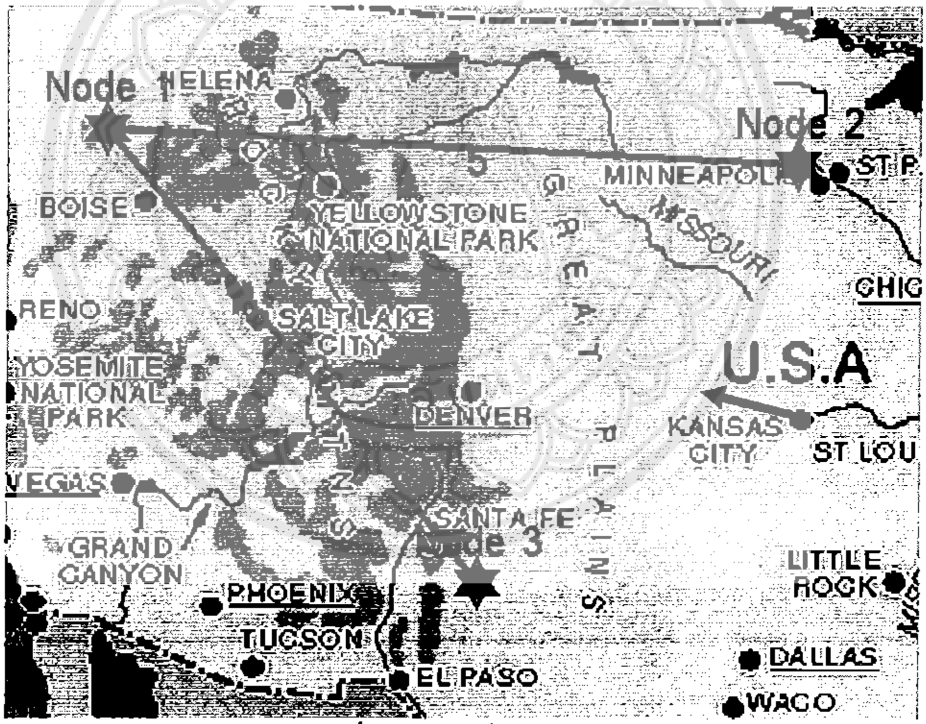
รูปที่ 4.32 ผลจากการแก้ไขค่าน้ำหนักถ่วง

จากรูปที่ 4.32 เมื่อผู้ใช้กรอกค่าแล้วโปรแกรมจะทำการเปลี่ยนค่าน้ำหนักถ่วง ถ้าผู้ใช้ไม่ต้องการให้บัพ 2 และบัพ 3 ไม่ให้เชื่อมต่อกันผู้ใช้สามารถแก้ไขค่าได้โดยการกดฟังก์ชั่นแก้ไขแล้วใส่ค่าตามนี้



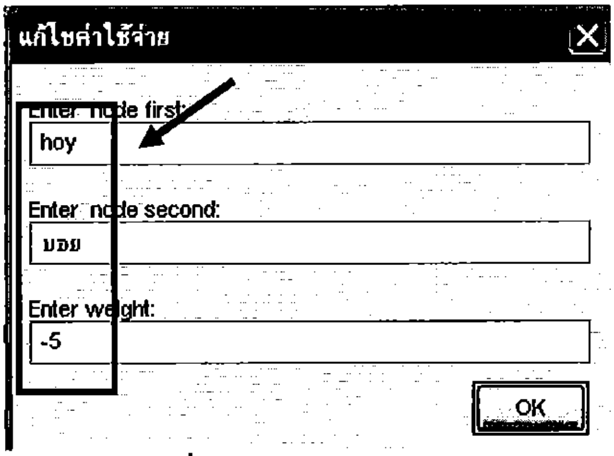
รูปที่ 4.33 แก้ไขค่าน้ำหนักถ่วง

จากรูปที่ 4.32 ค่าน้ำหนักถ่วง ที่ใส่ไปใหม่คือ inf ซึ่งหมายความว่าบัพ 2 และบัพ 3 ไม่สามารถเชื่อมกันได้

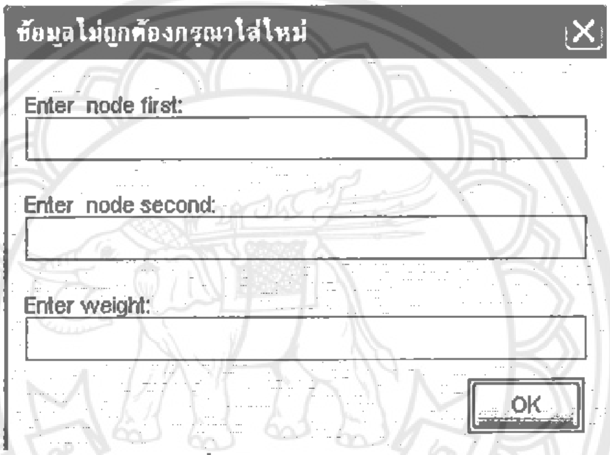


รูปที่ 4.34 เส้นเชื่อมหายไป

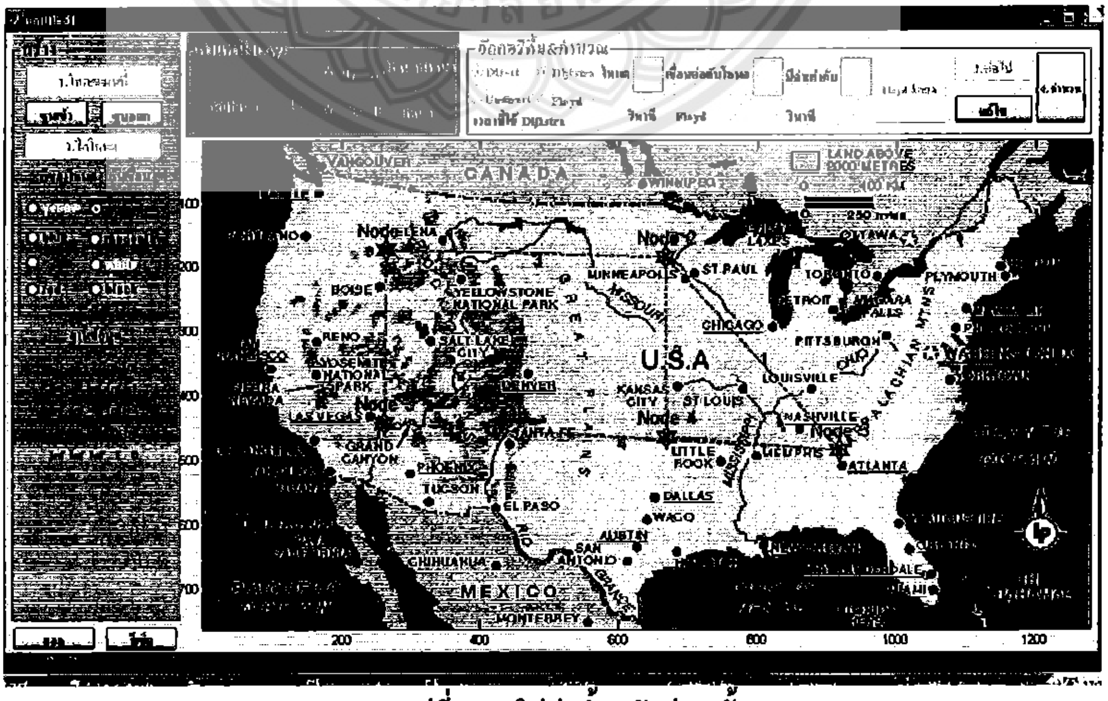
ถ้าผู้ใช้ใส่ข้อมูลผิดโปรแกรมจะทำการวนลูปจนกว่าผู้ใช้ใส่ข้อมูลถูกต้องหมายความว่าโปรแกรมจะตรวจสอบว่าค่าที่เข้ามาเป็นตัวเลขที่ติดลบ ตัวอักษรทั้งภาษาไทยและภาษาอังกฤษ



รูปที่ 4.35 ใส่ข้อมูลไม่ถูกต้อง



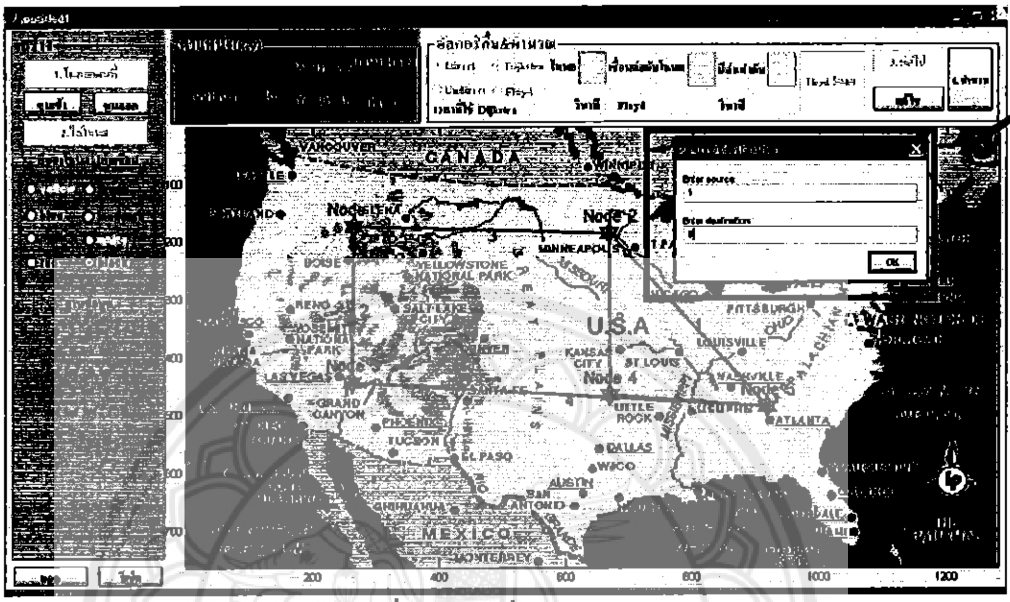
รูปที่ 4.36 วนลูปใส่ข้อมูล



รูปที่ 4.37 ใส่ค่าน้ำหนักถ่วง ทั้งหมด

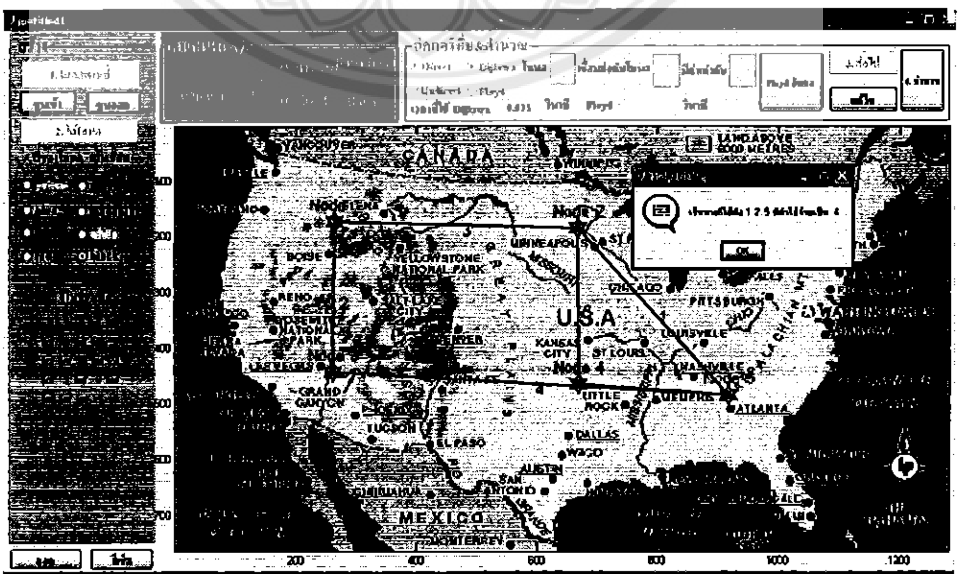
ส่วนที่ 2.4 การแสดงผล

เมื่อใส่ค่าน้ำหนักถ่วง แล้วให้กดที่ฟังก์ชั่น คำนวณ เพื่อคำนวณหาระยะทางที่สั้นที่สุด เมื่อกดฟังก์ชั่นแล้วโปรแกรมจะทดสอบระยะทางเริ่มต้นก่อน และเมื่อเสร็จสิ้นการทบสอบแล้ว เส้นเชื่อมที่เป็นเส้นประจะเปลี่ยนเป็นเส้นทึบ พร้อมกับให้ใส่จุดเริ่มต้น (1) กับ จุดปลายทาง (5)



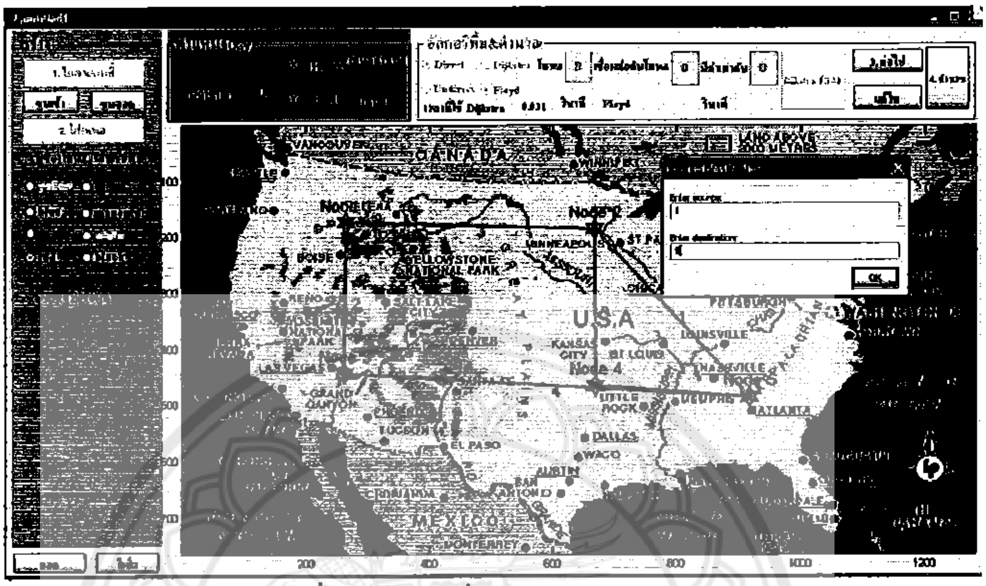
รูปที่ 4.38 จุดเริ่มต้นและจุดปลายทาง

เมื่อกดฟังก์ชั่น OK แล้วโปรแกรมจะแสดงระยะทางที่ใช้ค่าน้ำหนักถ่วงที่น้อยที่สุดและแสดงออกมาเป็นกราฟเส้นสีดำ (ถ้าเป็นขั้นตอนของ Dijkstra แต่เป็นเส้นสีแดง ถ้าเป็นขั้นตอนวิธีของ Floyd-Warshall ผู้ใช้สามารถเปลี่ยนเป็นสีอื่นได้ โดยการเลือกสีที่เมนูด้านซ้ายมือ ก่อนกดปุ่ม 4. คำนวณ)



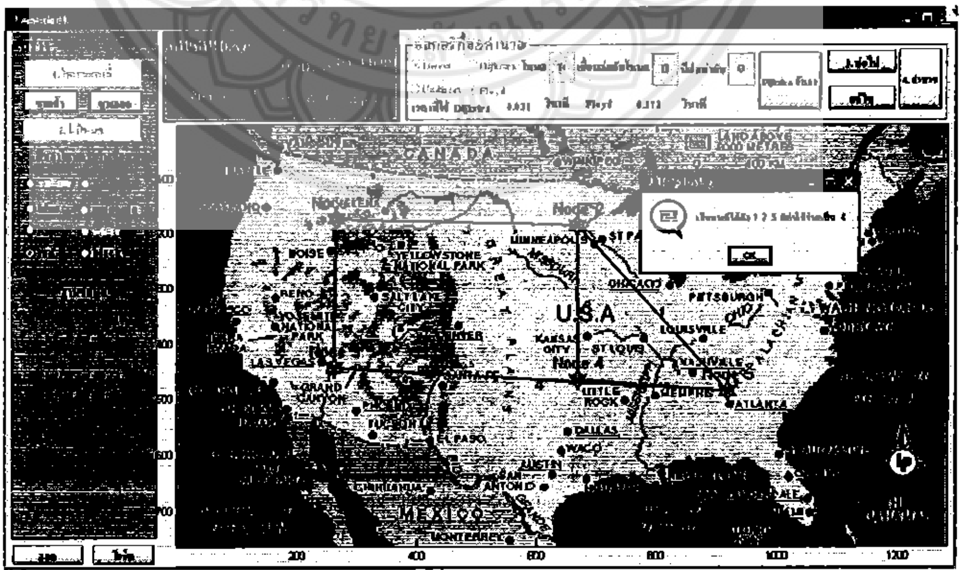
รูปที่ 4.39 แสดงผลลัพธ์ของ Dijkstra ขั้นตอนวิธี แบบ ไดกราฟ

ถ้าต้องการเปลี่ยนขั้นตอนวิธีจาก Dijkstra เป็น Floyd - Warshall ให้กดฟังก์ชั่น Floyd- Warshall แล้วตามด้วยปุ่ม 4. คำนวณ โปรแกรมจะทำการลบเส้นเชื่อมที่เป็นผลลัพธ์ที่เกิดจากขั้นตอนวิธีของ Dijkstra แล้วเริ่มคำนวณใหม่ โดยผู้ใช้ต้องกำหนดจุดเริ่มต้นและจุดปลายทางอีกครั้งหนึ่ง



รูปที่ 4.40 ใส่จุดเริ่มต้นและจุดปลายทาง

เมื่อกดฟังก์ชั่น OK แล้วโปรแกรมจะแสดงระยะทางที่ใช้ค่าน้ำหนักถ่วง ที่น้อยที่สุดและแสดงออกมาเป็นกราฟเส้นสีแดง



รูปที่ 4.41 แสดงผลลัพธ์ขั้นตอนวิธี ของ Floyd - Warshall

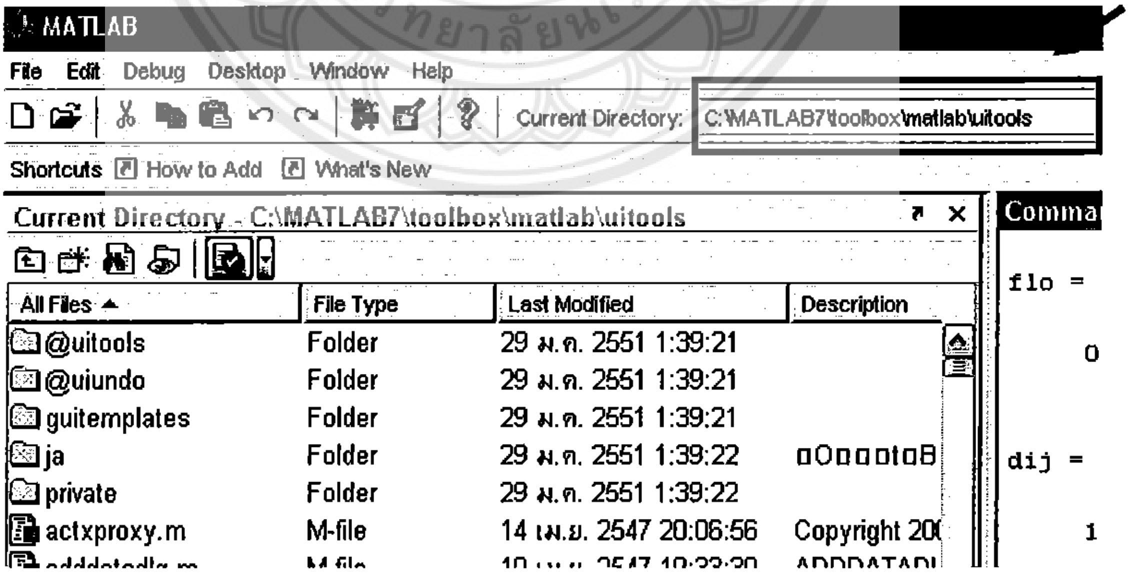
ส่วนที่ 3 แก้ไขข้อผิดพลาด

เมื่อทำการประมวลผลแล้วเกิดโปรแกรมฟ้องว่า

```
??? Undefined command/function 'inputdlgm'.

Error in ==> untitled1>pushbutton14_Callback at 3314
               a = inputdlgm(prompt(1),title1,1);

Error in ==> gui_mainfcn at 75
      feval(varargin{:});

Error in ==> untitled1 at 20
   gui_mainfcn(gui_State, varargin{:});

??? Error while evaluating uicontrol Callback.
```

รูปที่ 4.42 โปรแกรมแจ้ง error

ให้ทำการตั้งค่า Current directory ใหม่โดยตั้งค่าให้เป็น C:\MATLAB7\toolbox\matlab\uitools ดังรูปและสามารถเปลี่ยนการตั้งค่าใหม่ตามที่โปรแกรม MATLAB7 ได้ถูก setup ลงใน drive ต่างๆ เช่น D:\MATLAB7\toolbox\matlab\uitools หรือ C:\Program\MATLAB7\toolbox\matlab\uitools ดังรูป



รูปที่ 4.43 การตั้งค่า Current directory ใหม่

บทที่ 5
บทสรุป

โครงการนี้ได้ทำการประยุกต์ใช้ขั้นตอนวิธีที่ใช้สำหรับหาวิถีสั้นที่สุด(Shortest paths) ในการเดินทาง โดยมีการออกแบบโปรแกรมเป็นกราฟิก GUI เพื่อจำลองเส้นทางการเดินทาง วางแผนการเดินทาง ตลอดจนผลลัพธ์ที่เกิดจากการใช้โปรแกรมนี้จะแสดงผลออกมาเป็นแบบกราฟิก GUI ทั้งหมด

โครงการนี้เลือกใช้ขั้นตอนวิธีของ Dijkstra และของ Floyd-Warshall ในการคำนวณ เนื่องจากเป็นขั้นตอนวิธีที่มีความนิยมใช้งานอย่างแพร่หลาย พบว่าหลายสถาบันการศึกษามีการเรียนการสอนถึงขั้นตอนวิธีทั้งสองในบางรายวิชา อาทิ เช่น วิชาวิยุตคณิต วิชาการวิเคราะห์ขั้นตอนวิธีขั้นต้น วิชาโครงสร้างข้อมูล เป็นต้น

5.1 สรุปการทดสอบขั้นตอนวิธีของ Dijkstra และ ขั้นตอนวิธีของ Floyd-Warshall

โปรแกรมที่พัฒนาขึ้นมุ่งเน้นการแสดงผลออกมาในรูปแบบของส่วนต่อประสานกราฟิกกับผู้ใช้ (GUI) ซึ่งเอื้ออำนวยความสะดวกต่อผู้ใช้งานทั่วไป จากการทดสอบ พบว่าขั้นตอนวิธีที่เลือกใช้ให้ผลการคำนวณมีความถูกต้อง มีความสอดคล้องกับการคำนวณเชิงวิเคราะห์ทุกประการ

5.2 ปัญหาและอุปสรรค

การออกแบบและพัฒนาโปรแกรมนี้มีข้อจำกัดหลายด้าน ดังนี้

1. เนื่องจากโปรแกรมนี้ถูกพัฒนาโดยใช้โปรแกรม Matlab ซึ่งผู้จัดทำโครงงานไม่มีคุ้นเคยกับการใช้งาน โดยเฉพาะด้านการออกแบบ ส่วนต่อประสานกราฟิกกับผู้ใช้ (GUI) ทำให้การออกแบบและพัฒนาเป็นไปด้วยความล่าช้า ต่างจากการพัฒนาในโปรแกรม Visual basic หรือ Visual Studio 2005 ซึ่งผู้จัดทำโครงงานมีความคุ้นเคย จึงสามารถออกแบบ ส่วนต่อประสานกราฟิกกับผู้ใช้ (GUI) ได้สะดวกกว่า
2. การออกแบบและพัฒนาโปรแกรมสำหรับโครงงานนี้ มีรูปลักษณ์ที่เหมาะสมกับจอภาพที่เป็นแบบ Wide - Screen ดังนั้น หากมีการใช้งานโปรแกรมนี้ในหน้าจอประเภทอื่น อาจจะแสดงผลได้ไม่ดีเท่าที่ควร
3. คอมพิวเตอร์ที่ใช้ในการประมวลผลโปรแกรมที่ถูกพัฒนาขึ้นนี้ควรมีความเร็วในการประมวลผลค่อนข้างสูง เนื่องจาก ต้อมีการประมวลผลและแสดงผลเชิงกราฟิก

5.3 ข้อเสนอแนะและแนวทางการพัฒนา

1. ควรใช้ระบบฐานข้อมูลเพื่อใช้เก็บข้อมูลการเดินทาง เพราะหากประสงค์จะใช้ข้อมูลเดิม จะได้สามารถเรียกใช้ได้ทันที โดยที่ไม่ต้องคำนวณใหม่
2. ปรับปรุง GUI ให้ผู้ใช้สามารถเลือกให้มีการแสดงผลทั้งแบบบนจอ Wide - Screen และจอ Full-Screen ได้
3. ควรมีฟังก์ชั่นในการพิมพ์ผลที่ได้จากการคำนวณ เมื่อผู้ใช้งานประสงค์ที่จะนำผลที่ได้ติดตัวไปในระหว่างเดินทาง

เอกสารอ้างอิง

[1] ดร. วนิดา เหมะกุล, "**Discrete mathematics**", กรุงเทพมหานคร : เอช – เอนการพิมพ์. 2531.

[2] สำนักวิทยบริการและเทคโนโลยี มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา "วิชาคณิตศาสตร์ดิสครีตต์สำหรับวิศวกรรม." [Online]. Available: http://courseware.rmutl.ac.th

[3] G. Agnarsson, and R.G. Law, "**Graph Theory modeling Application and Algorithm**", Pearson Education, US, 2007.

[4] S. J. Chapman, "**Matlab programming for Engineer**", Thomson, US, 2004.

[5] Rashid bin muhamma "Graph Theory." [Online]. Available : http://www.cs.kent.edu/~rmuhamma/

[6] D.S. Malik. and M.K. SEN, "**Discrete Mathematical Structures**", Thomson, US, 2004.

[7] James A. mchugh, "**Algorithmic graph theory**", Prentice-Hall, New Jersey, 1990.

[8] Wikipedia, "**Shortest Path**." [Online]. Available: http://en.wikipedia.org

ประวัติผู้เขียนโครงงาน

ชื่อ นายรัตนโชติ พันธุ์วิไล

ภูมิลำเนา 115 หมู่ 2 ต.เวียง อ.เทิง จ.เชียงราย 57160

ประวัติการศึกษา

- สำเร็จการศึกษาระดับมัธยมศึกษาจาก
  โรงเรียนเทิงวิทยาคม
- ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail Tonhoy@hotmail.com