กราฟปะติดเพื่อแก้ปัญหาโครงข่ายอี-โลจิสติกส์

GLUED GRAPHS FOR SOLVING E-LOGISTICS NETWORK PROBLEMS

จิรพงษ์ เมฆเวียน
JIRAPONG MEKWIAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2550

กราฟปะติดเพื่อแก้ปัญหาโครงข่ายอี-โลจิสติกส์

GLUED GRAPHS FOR SOLVING E-LOGISTICS NETWORK PROBLEMS

จิรพงษ์ เมฆเวียน
JIRAPONG MEKWIAN

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาคณิตศาสตร์ประยุกต์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2550

GLUED GRAPHS FOR SOLVING E-LOGISTICS NETWORK PROBLEMS

JIRAPONG MEKWIAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENT FOR THE DEGREE OF
MASTER OF SCIENCE IN APPLIED MATHEMATICS
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2007

หัวข้อวิทยานิพนธ์	กราฟปะติดเพื่อแก้ปัญหาโครงข่ายอี-โลจิสติกส์
นักศึกษา	นายจิรพงษ์ เมฆเวียน
รหัสประจำตัว	48067403
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	คณิตศาสตร์ประยุกต์
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.ฉัฐไชย์ ลีนาวงศ์

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ศึกษาการใช้กราฟปะติด (Glued Graph) เพื่อแก้ปัญหาโครงข่ายอี-โลจิสติกส์ เมื่อกำหนดให้ G_1 และ G_2 เป็นกราฟ 2 กราฟใด ๆ สมมุติให้ $H_1 \subseteq G_1$ และ $H_2 \subseteq G_2$ เป็นกราฟเชื่อมโยงที่มีอย่างน้อยหนึ่งเส้นเชื่อม (Non-Trivial Connected Graph) โดยที่ H_1 สมสัณฐาน (Isomorphism) กับ H_2 ด้วยฟังก์ชัน f หรือ เขียนแทนด้วย $H_1 \cong_f H_2$ กราฟปะติดของ G_1 และ G_2 ที่ H_1 และ H_2 เทียบกับ f เขียนแทนด้วย $G_1 \underset{H_1 \cong_f H_2}{\diamond} G_2$ หรือ $G_1 \diamond G_2$ ซึ่งจะเป็นกราฟที่ได้ผลลัพธ์มาจากการรวมกันของ G_1 กับ G_2 โดย H_1 และ H_2 เป็นส่วนที่ปะติดกัน และกำหนดชื่อส่วนปะติดกันนั้นใหม่ว่า H โดยจะเรียก H ว่า โคลน (Clone)

ปัญหาที่ต้องการศึกษาในงานวิจัยนี้เป็นปัญหาโครงข่ายอี-โลจิสติกส์ ซึ่งได้แก่การสร้างโครงสร้างเพื่อส่งข้อมูลซึ่งนับเป็นหัวใจสำคัญในการดำเนินกิจกรรมโลจิสติกส์ โดยโครงข่ายที่สร้างอาจได้แก่ โครงข่ายระบบโทรศัพท์ โครงข่ายอินทราเน็ต และอินเทอร์เน็ต หรือระบบสื่อสารอิเล็กทรอนิกส์ใหม่ๆ อื่นๆ โดยจะนำมาแปลงให้อยู่ในรูปปัญหากราฟและโครงข่าย โดยให้จุดยอด (Node) แทนตำแหน่งของสถานประกอบการในโครงข่าย เช่น โรงงาน คลังสินค้า ศูนย์กระจายสินค้า และให้เส้นเชื่อม (Edge) แทนเส้นทางที่เชื่อมต่อระหว่าง 2 จุดยอด โดยทราบระยะทางของเส้นเชื่อมเหล่านั้น โดยมีวัตถุประสงค์เพื่อหาเส้นทางการเชื่อมต่อระหว่างจุดยอดทั้งหมดในโครงข่ายให้สั้นที่สุด หรือคือปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimum Spanning Tree) จากนั้นจะนำโครงข่าย 2 โครงข่ายมาเชื่อมต่อกันโดยการปะติด (Glue) และจะศึกษาถึงการหาต้นไม้แผ่ทั่วที่น้อยที่สุดในโครงข่ายกราฟปะติดใหม่นี้ ว่ามีความสัมพันธ์ถึงโครงข่ายเดิมอย่างไรซึ่งจะช่วยทำให้การรวมโครงข่ายอี-โลจิสติกส์ เป็นไปอย่างมีประสิทธิภาพและช่วยลดต้นทุนโลจิสติกส์ได้ นอกจากนี้ได้สร้างโปรแกรมคอมพิวเตอร์ขึ้นมาเพื่อทำการปะติดกราฟ และหาผลลัพธ์ของปัญหาโครงข่ายอี-โลจิสติกส์ดังกล่าวข้างต้น

Thesis Title	Glued Graphs for Solving e-Logistics Network Problems
Student	Mr.Jirapong Mekwian
Student ID	48067403
Degree	Master of Science
Program	Applied Mathematics
Year	2007
Thesis Advisor	Asst.Prof.Dr.Chartchai Leenawong

ABSTRACT

This thesis studies how to use glued graphs in solving e-logistic network problems. Let G_1 and G_2 be any two graphs. Assume that $H_1 \subseteq G_1$ and $H_2 \subseteq G_2$ are non-trivial connected graphs and also that H_1 and H_2 are isomorphic with f function denoted by $H_1 \cong_f H_2$, the glued graph G_1 and G_2 at H_1 and H_2 with respect to f denoted by $G_1 \underset{H_1 \cong_f H_2}{\diamond} G_2$ or $G_1 \diamond G_2$, is the combined graph of G_1 with G_2. H_1 and H_2 are glue sections which will be called H and defined as clone.

The problem of interest here is e-logistic network problems. It involves constructing basic infrastructure for transmitting and communicating data and information which is essential to logistics activities. The infrastructure networks include telephone network, Intranet and Internet networks and other new electronic communication systems. The system will be converted to a graph and network problem by letting the nodes be the locations of facilities such as plants, warehouses, or distribution centers. Also, let the edges be the possibilities of connecting each pair of facility locations whose distance is known. The objective of this graph problem is to find the shortest overall distances of connecting every vertex. The problem is simply known as the minimum spanning tree (MST) problem. Afterwards, in this thesis, two such networks will be combined according to the glue operation. A minimum spanning tree in the glued graph will be sought out and studied how it is related to the previous MSTs in the two separated graphs. Hopefully, this study can help reduce the cost of combining e-logistics networks. Last but not least, a computer program will be developed for gluing graphs and finding the optimal solutions to the above e-logistic network problems.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ผศ.ดร.ฉัฐไชย์ ลีนาวงศ์ ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ รศ.ดร.ไมตรี โพธิ์สุข, ดร.จริยา อุ่ยยะเสถียร, ดร.พรรณทิพย์ ภัทรอินทากร และดร.ดุษฎี ศุขวัฒน์ กรรมการสอบหัวข้อและโครงร่างวิทยานิพนธ์ที่ได้ให้คำแนะนำตลอดจนข้อชี้แนะ จนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ทุกคน ที่คอยเป็นกำลังใจ และให้ความช่วยเหลือตลอดมา

สำหรับคุณงานความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดาซึ่งเป็นที่รักและเคารพอย่างยิ่ง และพี่ๆ ที่ให้การสนับสนุนเรื่องการเรียนตลอดมา ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ประสิทประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

จิรพงษ์ เมฆเวียน

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญรูป VI

สารบัญตาราง VIII

บทที่ 1 บทนำ 1

1.1 ความสำคัญและที่มาของงานวิจัย 1

1.2 จุดประสงค์ของงานวิจัย 3

1.3 ขอบเขตของงานวิจัย 3

1.4 ประโยชน์ที่คาดว่าจะได้รับ 3

1.5 ขั้นตอนดำเนินงานวิจัย 3

1.6 ตารางการดำเนินงานวิจัย 4

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 5

2.1 ความหมายของโลจิสติกส์ 5

2.2 บทนิยาม ทฤษฎีและสัญลักษณ์ที่เกี่ยวข้อง 6

2.3 ต้นไม้แผ่ทั่วที่น้อยที่สุด 9

2.4 งานวิจัยที่เกี่ยวข้อง 15

บทที่ 3 การปะติดกราฟ ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดและการประยุกต์ 16

3.1 ตัวอย่างกราฟปะติดแบบต่างๆ 16

3.2 การปะติดกราฟต้นไม้แผ่ทั่วที่น้อยที่สุด 18

3.3 ตัวอย่างปัญหาอี-โลจิสติกส์ 30

สารบัญ (ต่อ)

หน้า

บทที่ 4 การออกแบบโปรแกรมคอมพิวเตอร์สำหรับการปะติดกราฟและปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด.....36

4.1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก.....40

4.2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก.....44

4.3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด.....48

4.4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์.....53

บทที่ 5 บทสรุปและข้อเสนอแนะ.....60

5.1 สรุปผลงานวิจัย.....60

5.2 ข้อเสนอแนะ.....63

เอกสารอ้างอิง.....64

ภาคผนวก.....65

ประวัติผู้เขียน.....82

สารบัญรูป

รูป หน้า

2.1 ตัวอย่างการใช้กราฟในการแก้ปัญหาโลจิสติกส์ 9

2.2 ผลที่ได้จากการทำซ้ำรอบที่ 1 ด้วยวิธีของพริม 10

2.3 ผลที่ได้จากการทำซ้ำรอบที่ 2 ด้วยวิธีของพริม 10

2.4 ผลที่ได้จากการทำซ้ำรอบที่ 3 ด้วยวิธีของพริม 11

2.5 ผลที่ได้จากการทำซ้ำรอบที่ 4 ด้วยวิธีของพริม 11

2.6 ผลที่ได้จากการทำซ้ำรอบที่ 5 ด้วยวิธีของพริม 12

2.7 ผลที่ได้จากการทำซ้ำรอบที่ 1 ด้วยวิธีของครุสกอล 12

2.8 ผลที่ได้จากการทำซ้ำรอบที่ 2 ด้วยวิธีของครุสกอล 13

2.9 ผลที่ได้จากการทำซ้ำรอบที่ 3 ด้วยวิธีของครุสกอล 13

2.10 ผลที่ได้จากการทำซ้ำรอบที่ 4 ด้วยวิธีของครุสกอล 14

2.11 ผลที่ได้จากการทำซ้ำรอบที่ 5 ด้วยวิธีของครุสกอล 14

3.1 โครงข่าย M 18

3.2 โครงข่าย N 19

3.3 โครงข่าย O 19

3.4 โครงข่าย P 19

3.5 โครงข่าย $M \diamond N$ ที่ $w_H = 2$ 20

3.6 โครงข่าย $M \diamond N$ ที่ $w_H = 6$ 21

3.7 โครงข่าย $O \diamond P$ ที่ $w_H = 7$ 22

3.8 โครงข่าย $O \diamond P$ ที่ $w_H = 9$ 23

3.9 โครงข่าย $M \diamond N$ ที่ $w_H = 7$ 24

3.10 โครงข่าย $O \diamond P$ ที่ $w_H = 14$ 25

3.11 ตัวอย่างโคลนเป็นวงแบบที่ 1 26

3.12 ตัวอย่างโคลนเป็นวงแบบที่ 2 27

3.13 ตัวอย่างสำหรับกรณีที่ 1 ของวิธีอุปนัยขั้นที่ 1 29

3.14 ตัวอย่างสำหรับกรณีที่ 2 ของวิธีอุปนัยขั้นที่ 1 30

3.15 ตัวอย่างสำหรับขั้นที่ 2 ของวิธีอุปนัย 31

3.16 โครงข่าย ก 33

3.17 โครงข่าย ข 34

สารบัญรูป (ต่อ)

รูป หน้า

3.18 โครงข่าย ก และ ข ปะติดกันที่จุด AD,DC 35

4.1 ภาพแสดงการใส่ค่าในการเลือกโปรแกรม 39

4.2 โครงข่าย A 41

4.3 โครงข่าย B 41

4.4 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.1 41

4.5 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.1 43

4.6 โครงข่าย $A \diamond B$ 43

4.7 โครงข่าย A แบบมีน้ำหนัก 45

4.8 โครงข่าย B แบบมีน้ำหนัก 45

4.9 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.2 45

4.10 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.2 47

4.11 โครงข่าย $A \diamond B$ แบบมีน้ำหนัก 47

4.12 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.3 49

4.13 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.3 51

4.14 โครงข่าย C แบบมีน้ำหนัก 54

4.15 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4 ของโครงข่าย A 55

4.16 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4 ของโครงข่าย B และส่วนโคลน 57

4.17 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4 58

4.18 โครงข่าย $C \diamond B$ แบบมีน้ำหนัก 59

สารบัญตาราง

ตาราง หน้า

1.6 การดำเนินงานวิจัย....4
3.1 โครงข่าย ก มีน้ำหนักบนเส้นเชื่อม....33
3.2 โครงข่าย ข มีน้ำหนักบนเส้นเชื่อม....33

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของวิทยานิพนธ์

เนื่องจากปัจจุบันนี้ กราฟได้เข้ามามีส่วนในการแก้ปัญหาในหลายรูปแบบไม่ว่าจะเป็นการเดินทาง การจัดเวลา การสร้างเส้นทาง การวางโครงข่ายต่างๆ เป็นต้น

ลำดับต่อไปจะกล่าวถึงกราฟปะติด เมื่อกำหนดให้ G_1 และ G_2 เป็นกราฟ 2 กราฟใด ๆ สมมุติให้ $H_1 \subseteq G_1$ และ $H_2 \subseteq G_2$ เป็นกราฟเชื่อมโยงที่มีอย่างน้อยหนึ่งเส้นเชื่อม (Non-Trivial Connected Graph) โดยที่ H_1 สมสัณฐาน (Isomorphism) กับ H_2 ด้วยฟังก์ชัน f หรือ เขียนแทนด้วย $H_1 \cong_f H_2$ กราฟปะติดของ G_1 และ G_2 ที่ H_1 และ H_2 เขียนแทนด้วย $G_1 \underset{H_1 \cong_f H_2}{\diamond} G_2$ หรือ $G_1 \diamond G_2$ ซึ่งจะเป็นกราฟที่ได้ผลลัพธ์มาจากการรวมกันของ G_1 กับ G_2 โดย H_1 และ H_2 เป็นส่วนที่ปะติดกัน และกำหนดชื่อส่วนปะติดกันนั้นใหม่ว่า H โดยจะเรียก H ว่า โคลน (Clone) และ G_1 กับ G_2 คือกราฟตั้งต้น (Original Graph)

ความหมายของโลจิสติกส์ (Logistic) หรืออาจเรียกอีกแบบว่าโลจิสติกส์กายภาพ (Physical Logistic) ตามนิยามของโลจิสติกส์ที่บัญญัติโดย สภาการบริหารโลจิสติกส์ (The Council of Logistics Management -- CLM) ไว้ว่า "กระบวนการในการวางแผน ดำเนินการ และควบคุมประสิทธิภาพและประสิทธิผลใน การเคลื่อนย้ายและการจัดเก็บ สินค้าและบริการ (Production Services) และสารสนเทศ (Information) จากจุดเริ่มต้นไปยังจุดที่มีการใช้งาน โดยมีเป้าหมายที่สอดคล้องกับความต้องการของผู้บริโภค" ซึ่งจากนิยามจะพบว่ามีการวางแผนและจัดการเกี่ยวกับตัวสินค้าซึ่งเป็นตัวหลักของการจัดการโลจิสติกส์ ในขณะเดียวกันก็ยังมีการวางแผน จัดการ และเคลื่อนย้ายเกี่ยวกับสารสนเทศ ให้เกิดประโยชน์ต่อการดำเนินงานเช่นกัน ซึ่งสิ่งสำคัญที่จะช่วยให้ข้อมูลสารสนเทศต่างๆ ส่งผ่านกันได้กับหน่วยงานที่เกี่ยวข้องก็คือโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ ซึ่งต้องมีการวางแผนและดำเนินการสร้างให้แล้วเสร็จก่อนที่กิจกรรมโลจิสติกส์ (Physical Logistics Activities) อื่นๆ จะดำเนินต่อไปได้ โดยจะขอให้นิยามอี-โลจิสติกส์ (E-Logistic) ดังนี้ อี-โลจิสติกส์ คือการดำเนินกิจกรรมโลจิสติกส์โดยเฉพาะในส่วนที่เกี่ยวข้องกับการไหลของข้อมูลสารสนเทศ (Information Flow) เช่น ข้อมูลเกี่ยวกับวัตถุดิบที่ใช้ในการผลิตและดำเนินงาน หรือข้อมูลจำนวนสินค้าที่มีอยู่ในโรงงานหรือคลังสินค้า ข้อมูลการสั่งซื้อสินค้าจากลูกค้า และข้อมูลอื่นๆ เพื่อสนับสนุนกิจกรรมโลจิสติกส์เชิงกายภาพให้ดำเนินไปอย่างมีประสิทธิภาพ รวมถึงสาธารณูปโภคพื้นฐานที่ช่วยเสริมให้การดำเนินกิจกรรมทางด้านโลจิสติกส์เป็นไปอย่างมีประสิทธิภาพเช่นกัน

โครงข่ายอี-โลจิสติกส์นี้อาจรวมถึง โครงข่ายสายโทรศัพท์ สายเคเบิ้ลใยแก้ว เพื่อเชื่อมต่อระบบ (Local Area Networks -- LANs), (Wide Area Networks -- WANs) หรือแม้แต่ระบบอินทราเน็ต นั่นคือรวมทุกโครงข่ายที่เกี่ยวข้องกับการไหลของข้อมูลสารสนเทศ เพื่อให้การดำเนินงานด้านโลจิสติกส์เป็นไปอย่างมีประสิทธิภาพที่สุด โดยการเชื่อมต่อโครงข่ายนี้ให้ผลรวมของการเชื่อมต่อไปยังทุกจุดในโครงข่ายสั้นที่สุด เพื่อประหยัดต้นทุน และให้การส่งข้อมูลรวดเร็ว ตัวอย่างหนึ่งของการนำกราฟปะติดไปใช้ให้เกิดประโยชน์ คือการนำไปใช้แก้ปัญหาโครงข่ายอี-โลจิสติกส์ เนื่องจากปัจจุบันนี้ ราคาน้ำมันมีราคาสูง วัสดุต่างๆ มีราคาเพิ่มขึ้นทำให้ต้นทุนในการขนส่งและปฏิบัติงานเพิ่มขึ้นทำให้ต้องหาวิธีในการลดต้นทุนรวมในการผลิต ซึ่งต้นทุนในการขนส่งและต้นทุนในการซื้อวัสดุ ก็เป็นอีกปัจจัยหนึ่งที่จะช่วยลดต้นทุนรวมในการผลิตลงได้ การใช้กราฟปะติดเข้ามาแก้ปัญหา ก็เป็นอีกวิธีที่ช่วยลดต้นทุนการขนส่งรวมถึงช่วยลดต้นทุนในการจัดซื้อวัสดุให้น้อยลงได้ วิธีการใช้กราฟปะติดมาแก้ปัญหาก็คือ การนำโครงข่ายที่มีระยะทางที่สั้นที่สุดมาหรือที่เรียกกันในปัญหากราฟว่า ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด การนำกราฟปะติดมาแก้ปัญหาโครงข่ายอี-โลจิสติกส์นั้น กระทำได้โดย บริษัทผู้รับเหมาก่อสร้างการวางโครงข่ายต่างๆ นำโครงข่ายที่รับมาจากโรงงาน 2 โรงงานหรือมากกว่านั้น มาพิจารณาดูว่ามีเส้นทางการวางโครงข่ายใดบ้างที่ทำการปะติดกันได้ จากนั้นหาโครงข่ายขึ้นมาใหม่ โครงข่ายที่ได้ขึ้นมาจากการปะติดใหม่นั้น จะทำให้ระยะทางที่สั้นที่สุดมีระยะทางน้อยกว่าระยะทางของ 2 โครงข่ายก่อนหน้านี้ นอกจากนี้ได้ออกแบบโปรแกรมคอมพิวเตอร์สำหรับทำการปะติดกราฟ และหาผลลัพธ์ของปัญหาโครงข่ายที่ปะติดใหม่ โดยโปรแกรมที่สร้างขึ้นมาจะมีลักษณะของโปรแกรมแตกต่างกันไป โดยจะมีโปรแกรมอยู่ 4 แบบด้วยกันคือ 1) โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก 2) โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก 3) โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด 4) โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

1.2 วัตถุประสงค์การวิจัย

1. เพื่อศึกษาคุณสมบัติและลักษณะของกราฟปะติดแบบต่าง ๆ
2. เพื่อประยุกต์ใช้กราฟปะติดในปัญหาโครงข่ายอี-โลจิสติกส์
3. เพื่อศึกษาถึงการหาคำตอบที่ดีที่สุดในปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดของกราฟปะติด
4. เพื่อพัฒนาโปรแกรมคอมพิวเตอร์ที่ช่วยสร้างกราฟปะติดและหาคำตอบที่ดีที่สุดของปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด

1.3 ขอบเขตของงานวิจัย

1. กราฟที่นำมาปะติดเป็นกราฟเชื่อมโยงที่มีอย่างน้อยหนึ่งเส้นเชื่อม
2. ทำการปะติดคราวละ 2 กราฟ
3. ศึกษาคุณลักษณะของกราฟปะติด โดยเน้นไปที่โคลนที่ไม่มีวงเป็นหลัก
4. ศึกษาเฉพาะการปะติดกราฟที่สัมพันธ์กับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทราบคุณสมบัติต่างๆ ของกราฟปะติดและการปะติดที่เกี่ยวข้องกับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด
2. การนำกราฟปะติดไปประยุกต์ใช้ในธุรกิจอุตสาหกรรมจริง
3. โปรแกรมคอมพิวเตอร์ที่สามารถปะติดกราฟ และหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติดได้

1.5 ขั้นตอนการดำเนินงานวิจัย

ขั้นตอนการดำเนินการวิจัยสำหรับวิทยานิพนธ์ฉบับนี้ มีดังต่อไปนี้

1. ค้นคว้าและศึกษาความรู้พื้นฐานที่เกี่ยวกับทฤษฎีกราฟ
2. ศึกษางานวิจัยที่เกี่ยวข้องกับกราฟปะติดและปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด
3. ศึกษาปัญหาโครงข่ายอี-โลจิสติกส์
4. นำการปะติดกราฟไปประยุกต์ใช้กับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด
5. สร้างสถานการณ์จำลองเกี่ยวกับโครงข่ายอี-โลจิสติกส์
6. สร้างโปรแกรมคอมพิวเตอร์เพื่อให้กราฟทำการปะติดกัน
7. สรุปผลและเขียนวิทยานิพนธ์

1.6 ตารางการดำเนินงานวิจัย

ตารางดำเนินงานวิจัยตามขั้นตอนต่างๆ มีดังต่อไปนี้

ขั้นตอนการดำเนินงานวิจัย	2549							2550				
	6	7	8	9	10	11	12	1	2	3	4	5
ขั้นตอนที่ 1	→	→										
ขั้นตอนที่ 2		→	→	→								
ขั้นตอนที่ 3				→	→							
ขั้นตอนที่ 4					→	→						
ขั้นตอนที่ 5						→	→	→				
ขั้นตอนที่ 6							→	→	→	→	→	
ขั้นตอนที่ 7										→	→	→

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ความหมายของโลจิสติกส์

โลจิสติกส์เป็นหน้าที่งานที่มีขอบข่ายที่กว้างขวางและบรรลุวัตถุประสงค์ได้ยาก และยังมีผลกระทบอย่างมากต่อมาตรฐานความเป็นอยู่ของผู้คนในสังคม ในสังคมสมัยใหม่นี้เรามักจะมีความคาดหมายต่อการบริการด้านโลจิสติกส์ที่เยี่ยมยอดและมักจะให้ความสนใจกับโลจิสติกส์ก็ต่อเมื่อมีปัญหาเกิดขึ้น เพื่อให้เกิดความเข้าใจเกี่ยวกับโลจิสติกส์มากยิ่งขึ้นขอให้พิจารณาตัวอย่างต่อไปนี้

- ความไม่สะดวกเมื่อเราไปซื้ออาหาร เสื้อผ้า และสินค้าอื่นๆ ถ้าระบบโลจิสติกส์ไม่เอื้ออำนวยให้สิ่งเหล่านี้มาอยู่รวมกันในสถานที่เดียวกัน เช่น ในร้านค้า หรือในห้างสรรพสินค้า
- ความท้าทายในการกำหนดขนาดหรือรูปลักษณ์ที่เหมาะสม ถ้าหากระบบโลจิสติกส์ไม่สามารถทำให้เราเลือกสรรสินค้า สีสัน ขนาด หรือรูปแบบที่หลากหลายได้ ซึ่งสิ่งต่างๆ เหล่านี้เคยเป็นปัญหาที่เกิดขึ้นอย่างยาวนานในอดีตสหภาพโซเวียตมาก่อน
- ความผิดหวังเมื่อเราไปซื้อสินค้าที่ร้านตามที่มีโฆษณา แล้วพบว่าไม่มีสินค้าเนื่องจากการจัดส่งสินค้าล่าช้า

ตัวอย่างที่แสดงมานี้เป็นตัวอย่างเพียงเล็กน้อยเท่านั้น ที่แสดงให้เห็นว่าโลจิสติกส์ได้เข้ามามีส่วนในชีวิตประจำวันของเราอย่างไร

ความหมายของการจัดการด้านโลจิสติกส์

โลจิสติกส์อาจถูกเรียกได้หลายๆ ชื่อ ดังนี้

- โลจิสติกส์ทางธุรกิจ (Business logistics)
- การจัดการช่องทาง (การจำหน่าย) (Channel management)
- การกระจาย (สินค้า) (Distribution)
- โลจิสติกส์ทางอุตสาหกรรม (Industrial logistics)
- การจัดการโลจิสติกส์ (Logistics management)
- การจัดการวัสดุ (Materials management)
- การกระจายวัตถุ (Physical distribution)
- ระบบการตอบสนองที่รวดเร็ว (Quick-response systems)
- การจัดการโซ่อุปทาน (Supply chain management)

ความหมายของคำต่างๆ เหล่านี้ในภาพโดยรวมแล้ว ก็คือ การบริหารกระบวนการไหล (flow) ของสินค้าหรือวัตถุดิบจากจุดเริ่มต้นไปยังจุดที่มีการใช้สินค้าหรือวัตถุดิบนั้น และในบางกรณีก็ไปยังจุดทำลายสินค้านั้น ซึ่ง สภาการบริหารโลจิสติกส์ ซึ่งเป็นองค์กรทางวิชาชีพทางด้านโลจิสติกส์ของประเทศสหรัฐอเมริกา ได้ให้คำจำกัดความของการจัดการด้านโลจิสติกส์เอาไว้คือ

"กระบวนการในการวางแผน ดำเนินการ และควบคุมประสิทธิภาพและประสิทธิผลในการเคลื่อนย้ายและการจัดเก็บ สินค้าและบริการ และสารสนเทศจากจุดเริ่มต้นไปยังจุดที่มีการใช้งานโดยมีเป้าหมายที่สอดคล้องกับความต้องการของผู้บริโภค"

ตัวอย่างกิจกรรมที่มีโลจิสติกส์เข้าไปเกี่ยวข้อง

การจัดการวัตถุดิบ

การจัดการวัตถุดิบเป็นเรื่องใหญ่ เกี่ยวข้องกับการดำเนินงานตั้งแต่การจัดหาวัตถุดิบ การปฏิบัติงานระหว่างดำเนินการ หรือสินค้าสำเร็จรูปภายในโรงงานหรือคลังสินค้า เนื่องจากองค์กรพบว่าต้นทุนเพิ่มสูงขึ้นโดยไม่ได้มีการเพิ่มมูลค่าให้กับตัวสินค้าทุกครั้งที่สินค้าถูกเคลื่อนย้ายหรือจัดการ จุดมุ่งหมายประการแรกของการจัดการวัตถุดิบหรือการลดงานให้ได้มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งก็ได้แก่ การลดระยะทาง การลดปัญหาคอขวดของระดับสินค้าคงคลัง การลดการสูญเสียและการสูญหาย การจัดการวัตถุดิบนี้จะสามารถช่วยประหยัดค่าใช้จ่ายลงได้อย่างชัดเจน

2.2 บทนิยาม ทฤษฎีและสัญลักษณ์ที่เกี่ยวข้อง

ในหัวข้อจะกล่าวถึงทฤษฎีเบื้องต้นที่ใช้ในงานวิจัย โดยที่เนื้อหาหลักจะมุ่งประเด็นไปที่กราฟปะติด รวมถึงการหาปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด เพื่อนำมาปรับใช้กับสถานการณ์จำลองจริงที่ผู้วิจัยกำลังศึกษาอยู่ในขณะนี้ ทั้งนี้ได้ทำการรวบรวมนิยามและทฤษฎีบทที่เกี่ยวข้องมาจาก [6] และ [7] เป็นหลัก ดังต่อไปนี้

บทนิยาม 2.2.1 กราฟ $G = (V,E)$ เป็นโครงสร้างคณิตศาสตร์ที่ประกอบด้วยเซต 2 เซต V และ E สมาชิกของ V เรียกว่า **จุด** (vertex) และสมาชิกของ E เรียกว่า **เส้น** หรือ **เส้นเชื่อม** (edge)

บทนิยาม 2.2.2 **กราฟไม่ชัด** (trivial graph) เป็นกราฟที่ประกอบด้วยจุด 1 จุดและไม่มีเส้น

บทนิยาม 2.2.3 **จุดประชิด** (adjacent vertices) คือ 2 จุดที่มีการเชื่อมต่อด้วยเส้น

บทนิยาม 2.2.4 **เส้นประชิด** (adjacent edges) คือ 2 เส้นที่มีจุดปลายเป็นจุดเดียวกัน

บทนิยาม 2.2.5 ถ้าจุด v เป็นจุดปลายของเส้น e แล้ว v พูดว่า**ตกกระทบ** (incident) บน e และ e ตกกระทบบน v

บทนิยาม 2.2.6 **แนวเดิน** (walk) W ในกราฟ G คือ ลำดับจำกัดของจุดและเส้นสลับกันของกราฟ G ดังนี้

$$W : v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n$$

ซึ่งเส้น e_i มีจุดปลายคือ v_{i-1} และ v_i สำหรับ $1 \le i \le n$

บทนิยาม 2.2.7 **กราฟเชื่อมโยง** (connected) ถ้าทุก ๆ คู่ของจุด u และ v มีแนวเดินจาก u ไป v

บทนิยาม 2.2.8 **กราฟมีน้ำหนัก** (weighted graph) เป็นกราฟที่แต่ละเส้นระบุจำนวนเรียกว่า น้ำหนักของเส้น ใช้สัญลักษณ์ $w(e)$

บทนิยาม 2.2.9 **สับกราฟ** (subgraph) ของกราฟ G คือกราฟ H ซึ่งจุดและเส้นทั้งหมดอยู่ใน G

บทนิยาม 2.2.10 **สับกราฟแผ่ทั่ว** (spanning subgraph) ของกราฟ G คือสับกราฟ H ของ G ที่ $V_H = V_G$

บทนิยาม 2.2.11 **ต้นไม้** (tree) กราฟเชื่อมโยงที่ไม่มีวง

บทนิยาม 2.2.12 **ต้นไม้แผ่ทั่ว** (spanning tree) ของกราฟ คือสับกราฟแผ่ทั่วถึงที่เป็นต้นไม้

บทนิยาม 2.2.13 **กราฟสมสัณฐาน** (graph isomorphism) ให้ H_1 และ H_2 เป็นกราฟ 2 กราฟใดๆ H_1 สมสัณฐานกับ H_2 ถ้ามีฟังก์ชัน $f : V(H_1) \to V(H_2)$ เป็นฟังก์ชันแบบหนึ่งต่อหนึ่งและทั่วถึง โดยที่ $vu \in (H_1) \Leftrightarrow f(v)f(u) \in E(H_2)$

บทนิยาม 2.2.14 เมื่อกำหนดให้ G_1 และ G_2 เป็นกราฟ 2 กราฟใด ๆ สมมุติให้ $H_1 \subseteq G_1$ และ $H_2 \subseteq G_2$ เป็นกราฟเชื่อมโยงที่มีอย่างน้อยหนึ่งเส้นเชื่อม (Non-Trivial Connected Graph) โดยที่ H_1 สมสัณฐาน (Isomorphism) กับ H_2 ด้วยฟังก์ชัน f หรือเขียนแทนด้วย $H_1 \cong_f H_2$ การปะติดของ G_1 และ G_2 ที่ H_1 และ H_2 เทียบกับ f จากข้างต้นจะเรียกว่า **กราฟปะติด** (Glued Graph) เขียนแทนด้วย $G_1 \underset{H_1 \cong_f H_2}{\diamond} G_2$ หรือ $G_1 \diamond G_2$

บทนิยาม 2.2.15 กราฟที่ได้ผลลัพธ์มาจากการรวมกันของ G_1 กับ G_2 โดย H_1 และ H_2 เป็นส่วนที่ปะติดกัน และกำหนดชื่อส่วนปะติดกันนั้นใหม่ว่า H โดยจะเรียก H ว่า **โคลน** (Clone)

ทฤษฎีบท 2.2.1 ให้ T เป็นกราฟอย่างง่าย ที่มีจุดยอด n จุด สิ่งต่อไปนี้จะสมมูลกัน

1. T เป็นต้นไม้
2. T เป็นกราฟเชื่อมโยงและไม่มีวง
3. T เป็นกราฟเชื่อมโยงและมี n-1 เส้น
4. T ไม่มีวงและมี n-1 เส้น

ทฤษฎีบท 2.2.2 กราฟ G เป็นกราฟเชื่อมโยงก็ต่อเมื่อกราฟนั้นมีต้นไม้แผ่ทั่ว

ทฤษฎีบท 2.2.3 ขั้นตอนวิธีของครุสกอลหาน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของกราฟเชื่อมโยงที่ไม่มีทิศทางได้

ทฤษฎีบท 2.2.4 ให้ G_1 และ G_2 เป็นกราฟที่ปะติดกันที่โคลน H แล้ว

$|V(G_1 \diamond G_2)| = |V(G_1)| + |V(G_2)| - |V(H)|$ และ

$|E(G_1 \diamond G_2)| = |E(G_1)| + |E(G_2)| - |E(H)|$

ทฤษฎีบท 2.2.5 ให้ T_1 และ T_2 เป็นกราฟ $T_1 \diamond T_2$ เป็นกราฟต้นไม้ (tree) ก็ต่อเมื่อ T_1 และ T_2 เป็น กราฟต้นไม้ (tree)

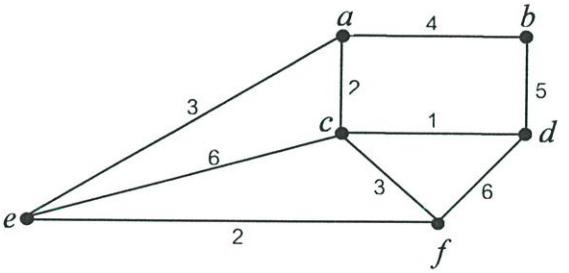
ทฤษฎีบท 2.2.6 ให้ G_1 และ G_2 เป็นกราฟถ้า $G_1 \diamond G_2$ ประกอบด้วยวงที่เกิดขึ้นมาใหม่ (created cycle) แล้ว G_1 และ G_2 ทั้งสองประกอบด้วยวง

ในที่นี้จะขอรวบรวมสัญลักษณ์ที่จะใช้ในการวิจัยเล่มนี้ทั้งหมด ดังนี้

n	คือจำนวนเส้นในกราฟ G
T	คือต้นไม้
T^*	คือต้นไม้แผ่ทั่วที่น้อยที่สุด
$\|V(G)\|$	คือจำนวนของจุดในกราฟ G
$\|E(G)\|$	คือจำนวนของเส้นในกราฟ G
w_G	คือน้ำหนักทั้งหมดของกราฟ G
$w(e_G)$	คือน้ำหนักของเส้น e ในกราฟ G
$A \diamond B$	คือกราฟ A ปะติดกับกราฟ B
$H_1 \cong_f H_2$	คือ H_1 สมสัณฐานกับ H_2 ที่ฟังก์ชัน f

2.3 ต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimum Spanning Tree)

ในกราฟที่มีน้ำหนักรูปที่ 2.1 แสดงถึงเมือง 6 แห่ง และค่าใช้จ่ายในการก่อสร้างสายไฟฟ้าระหว่างเมือง 2 เมือง โดยจะให้จุด แทนเมืองแต่ละเมือง และให้เส้น แทนเส้นทางในการก่อสร้างสายไฟฟ้า ซึ่งจะให้น้ำหนักที่อยู่บนเส้น แทนค่าใช้จ่ายในการก่อสร้าง เราต้องการสร้างระบบสายไฟฟ้าที่มีค่าใช้จ่ายน้อยที่สุดที่จะเชื่อมเมืองทั้งหมดนี้



รูปที่ 2.1 ตัวอย่างการใช้กราฟในการแก้ปัญหาโลจิสติกส์

คำตอบจะถูกแทนได้ด้วยสับกราฟรูปหนึ่งของกราฟเริ่มต้น ซึ่งสับกราฟนี้จะต้องเป็นต้นไม้แผ่ทั่วกราฟ เพราะทุกเมืองต้องอยู่ในระบบสายไฟฟ้า สับกราฟนี้ต้องติดต่อกัน เพราะเมืองใดๆ ต้องมีสายไฟฟ้าไปเชื่อมต่อกับเมืองอื่นๆ ได้ และสับกราฟนี้ต้องมีทางเดินเพียงทางชุดเดียวระหว่างแต่ละคู่ของจุดยอด เพราะกราฟที่มีทางเดินขนานระหว่างจุดยอดคู่หนึ่งจะไม่ทำให้ระบบการก่อสร้างสายไฟฟ้ามีค่าใช้จ่ายน้อยที่สุด สรุปสิ่งที่ต้องการคือ ต้นไม่แผ่ทั่วที่น้อยที่สุดโดยมีผลบวกของน้ำหนักบนเส้นน้อยที่สุด ต้นไม้เช่นนี้เรียกว่าต้นไม้แผ่ทั่วที่น้อยที่สุด

แบบการคำนวณที่ใช้หาต้นไม้แผ่ทั่วที่น้อยที่สุด ที่จะกล่าวถึงมีชื่อว่า ขั้นตอนวิธีของพริม (Prim's Algorithm) และ ขั้นตอนวิธีของครุสกอล (Kruskal's Algorithm) มีขั้นตอนวิธีดังนี้

ขั้นตอนวิธีของพริม (Prim's Algorithm) [1957]

ขั้นที่ 1 ให้ T^* เป็นต้นไม้ที่ประกอบด้วยจุดยอดหนึ่งจุด $T^* = \{1\}, B = \{2,...,n\}$

ขั้นที่ 2 ถ้า $B = \varnothing$, แล้วให้หยุด

ขั้นที่ 3 หาจุด $u^* \in T^*$ และ $v^* \in B$ ซึ่ง $c(u^*v^*) = \min\{c(uv) : u \in T^* \text{ and } v \in B\}$

จัด $T^* = T^* + u^*v^*$ และ $B = B - \{v^*\}$ แล้วย้อนกลับไปทำขั้นที่ 2

ตัวอย่างที่ 2.1 การใช้แบบคำนวณการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของพริม ในรูป 2.1 โดยมีจุดยอดเรียงกันดังนี้ a,b,c,d,e,f

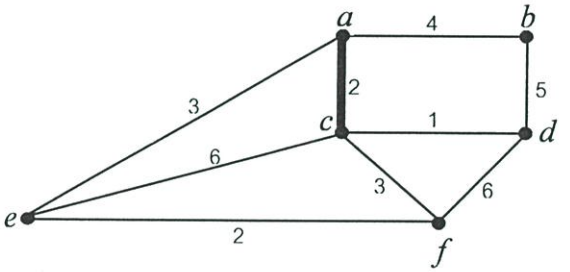
วิธีทำ

ขั้นที่ 1 $T^* = \{a\}, \qquad B = \{b,c,d,e,f\}$

ขั้นที่ 2 $B \neq \varnothing$

ขั้นที่ 3 $c(u^*v^*) = c(ac) = 2$

จัด $T^* = T^* + ac$ และ $B = B - \{c\}$



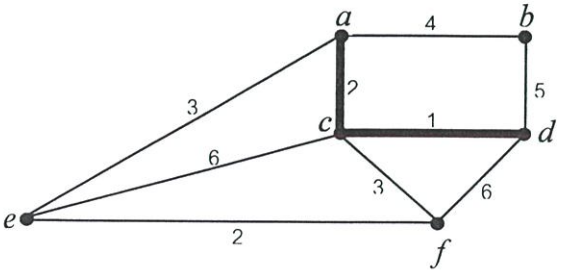
รูปที่ 2.2 ผลที่ได้จากการทำซ้ำรอบที่ 1 ด้วยวิธีของพริม

ขั้นที่ 1 $T^* = \{a,c\}, \qquad B = \{b,d,e,f\}$

ขั้นที่ 2 $B \neq \varnothing$

ขั้นที่ 3 $c(u^*v^*) = c(cd) = 1$

จัด $T^* = T^* + cd$ และ $B = B - \{d\}$



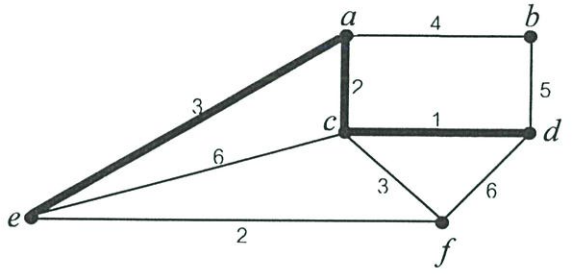
รูปที่ 2.3 ผลที่ได้จากการทำซ้ำรอบที่ 2 ด้วยวิธีของพริม

ขั้นที่ 1 $T^* = \{a, c, d\}, \qquad B = \{b, e, f\}$

ขั้นที่ 2 $B \neq \varnothing$

ขั้นที่ 3 $c(u^*v^*) = c(ae) = 3$

จัด $T^* = T^* + ae$ และ $B = B - \{e\}$



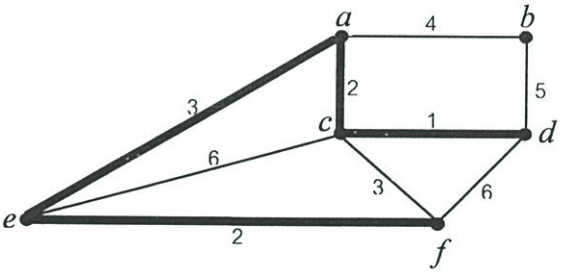
รูปที่ 2.4 ผลที่ได้จากการทำซ้ำรอบที่ 3 ด้วยวิธีของพริม

ขั้นที่ 1 $T^* = \{a, c, d, e\}, \qquad B = \{b, f\}$

ขั้นที่ 2 $B \neq \varnothing$

ขั้นที่ 3 $c(u^*v^*) = c(ef) = 2$

จัด $T^* = T^* + ef$ และ $B = B - \{f\}$



รูปที่ 2.5 ผลที่ได้จากการทำซ้ำรอบที่ 4 ด้วยวิธีของพริม

ขั้นที่ 1 $T^* = \{a, c, d, e, f\}, \qquad B = \{b\}$

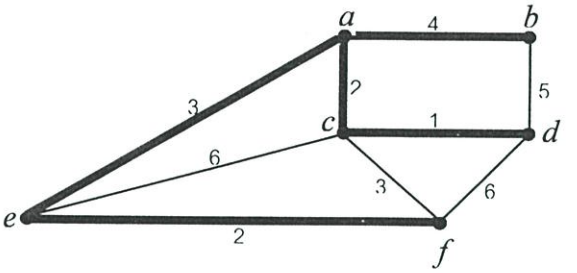
ขั้นที่ 2 $B \neq \varnothing$

ขั้นที่ 3 $c(u^*v^*) = c(ab) = 4$

จัด $T^* = T^* + ab$ และ $B = B - \{b\}$

ขั้นที่ 2 $B = \varnothing$ แล้วจึงหยุด

จบการทำงานตามขั้นตอนวิธีของพริม คำอบที่ได้ คือ $T^* = \{e(cd), e(ef), e(ac), e(ae), e(ab)\}$



รูปที่ 2.6 ผลที่ได้จากการทำซ้ำรอบที่ 5 ด้วยวิธีของพริม

ขั้นตอนวิธีของครุสกอล (Kruskal's Algorithm) [1956]

ขั้นที่ 1 กำหนดส่วนของผลเฉลย $T^* = \varnothing$, $k = 1$ และเรียงน้ำหนักของ k โดย

$$c(e_1) \le c(e_2) \le \ldots \le c(e_q)$$

ขั้นที่ 2 ถ้า T^* เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว หยุด

ขั้นที่ 3 ถ้า $T^* + e_k$ ไม่มีวง แล้ว $T^* = T^* + e_k$ กำหนด $k = k+1$ และย้อนกลับไปขั้นที่ 2

ตัวอย่างที่ 2.2 การใช้แบบคำนวณการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของครุสกอล ในรูป 2.1 โดยมีจุดยอดเรียงกันดังนี้ a,b,c,d,e,f

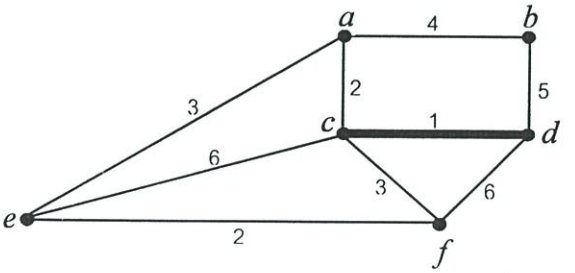
วิธีทำ

ขั้นที่ 1 $T^* = \varnothing$ จัดลำดับของเส้นโดยเรียงน้ำหนักจากน้อยไปมาก

1, 2, 2, 3, 3, 4, 5, 6, 6

ขั้นที่ 2 T^* ยังไม่เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว

ขั้นที่ 3 $T^* + e(cd)$ ไม่มีวง จะได้ว่า $T^* = T^* + e(cd)$



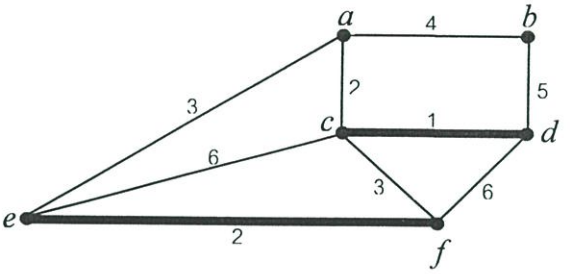
รูปที่ 2.7 ผลที่ได้จากการทำซ้ำรอบที่ 1 ด้วยวิธีของครุสกอล

ขั้นที่ 1 $T^* = \{e(cd)\}$ จัดลำดับของเส้นโดยเรียงน้ำหนักจากน้อยไปมาก

2, 2, 3, 3, 4, 5, 6, 6

ขั้นที่ 2 T^* ยังไม่เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว

ขั้นที่ 3 $T^* + e(ef)$ ไม่มีวง จะได้ว่า $T^* = T^* + e(ef)$



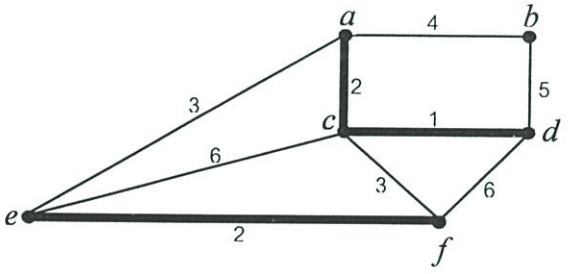
รูปที่ 2.8 ผลที่ได้จากการทำซ้ำรอบที่ 2 ด้วยวิธีของครุสกอล

ขั้นที่ 1 $T^* = \{e(cd), e(ef)\}$ จัดลำดับของเส้นโดยเรียงน้ำหนักจากน้อยไปมาก

2, 3, 3, 4, 5, 6, 6

ขั้นที่ 2 T^* ยังไม่เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว

ขั้นที่ 3 $T^* + e(ac)$ ไม่มีวง จะได้ว่า $T^* = T^* + e(ac)$



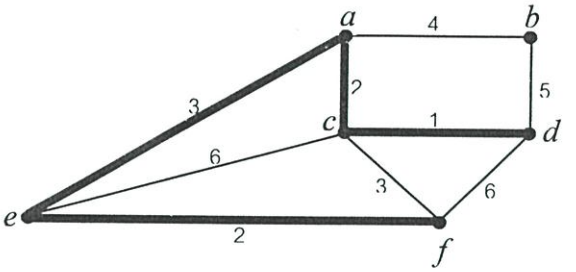
รูปที่ 2.9 ผลที่ได้จากการทำซ้ำรอบที่ 3 ด้วยวิธีของครุสกอล

ขั้นที่ 1 $T^* = \{e(cd), e(ef), e(ac)\}$ จัดลำดับของเส้นโดยเรียงน้ำหนักจากน้อยไปมาก

3, 3, 4, 5, 6, 6

ขั้นที่ 2 T^* ยังไม่เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว

ขั้นที่ 3 $T^* + e(ae)$ ไม่มีวง จะได้ว่า $T^* = T^* + e(ae)$



รูปที่ 2.10 ผลที่ได้จากการทำซ้ำรอบที่ 4 ด้วยวิธีของครุสกอล

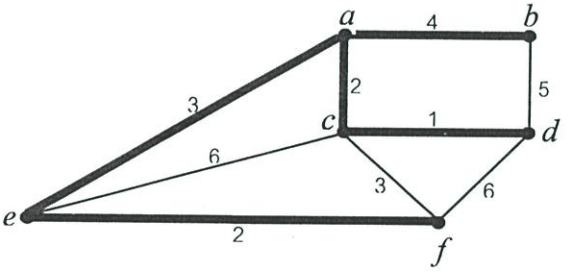
ขั้นที่ 1 $T^* = \{e(cd), e(ef), e(ac), e(ae)\}$

จัดลำดับของเส้นโดยเรียงน้ำหนักจากน้อยไปมาก

3, 4, 5, 6, 6

ขั้นที่ 2 T^* ยังไม่เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว

ขั้นที่ 3 $T^* + e(cf)$ มีวง เพราะฉะนั้น $T^* + e(ab)$ จะได้ว่า $T^* = T^* + e(ab)$



รูปที่ 2.11 ผลที่ได้จากการทำซ้ำรอบที่ 5 ด้วยวิธีของครุสกอล

ขั้นที่ 1 $T^* = \{e(cd), e(ef), e(ac), e(ae), e(ab)\}$

ขั้นที่ 2 T^* เป็นต้นไม้แผ่ทั่วที่น้อยที่สุดแล้ว หยุด

2.4 งานวิจัยที่เกี่ยวข้อง

เนื่องจากงานวิจัยที่เกี่ยวข้องกับกราฟปะติด (Glued Graph) มีเพียง 1 งานวิจัยเท่านั้น กราฟปะติดเป็นเรื่องที่สร้างขึ้นมาใหม่ยังไม่แผ่หลายมากนัก โดยงานวิจัยนี้คิดค้นขึ้นโดย Promsakon และ Uiyyasathian [6] เมื่อกำหนดให้ G_1 และ G_2 เป็นกราฟ 2 กราฟใด ๆ สมมุติให้ $H_1 \subseteq G_1$ และ $H_2 \subseteq G_2$ เป็นกราฟเชื่อมโยงที่มีอย่างน้อยหนึ่งเส้นเชื่อม (Non-Trivial Connected Graph) โดยที่ H_1 สมสัณฐาน (Isomorphism) กับ H_2 ด้วยฟังก์ชัน f หรือ เขียนแทนด้วย $H_1 \cong_f H_2$ กราฟปะติดของ G_1 และ G_2 ที่ H_1 และ H_2 เขียนแทนด้วย $G_1 \underset{H_1 \cong_f H_2}{\diamond} G_2$ หรือ $G_1 \diamond G_2$ จะเลือกปะติดกราฟกันระหว่าง กราฟต้นไม้, กราฟป่าไม้, และกราฟสองส่วน และเมื่อกำหนดให้ ขอบเขตบนของจำนวนของสีที่เกิดจากการปะติดกราฟในเทอมของกราฟเริ่มต้น ผู้วิจัยได้พิสูจน์ไว้ว่ากราฟปะติดมีขอบเขตบนจริง ทั้งนี้ได้ยกนิยามและทฤษฎีบทที่เกี่ยวข้องไว้ในหัวข้อ 2.2 ด้วยแล้ว

แต่มีงานวิจัยจำนวนมากที่ได้นำเสนองานเกี่ยวกับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimum Spanning Tree) ยกตัวอย่างเช่น งานวิจัยของ M. Chip [8] โดยงานวิจัยนี้จะศึกษาการดำเนินการของขั้นตอนวิธีต้นไม้แผ่ทั่วที่น้อยที่สุดของพริม เป็นวิธีที่ใช้กันอยู่จำนวนมาก จะแสดงถึงวิธีการกระทำในเชิงเส้น หรือเกือบจะเชิงเส้นโดยหวังว่าเวลาจะอยู่บนขอบเขตของกราฟ และจะช่วยอธิบายว่าทำไมขั้นตอนวิธีของพริมเป็นขั้นตอนที่ดี โดยเฉพาะ จะแสดงว่าถ้าเริ่มต้นที่ n จุด m เส้นของกราฟ และโดนการสุ่มเรียงลำดับน้ำหนักของเส้น แล้วขั้นตอนวิธีของพริมกระทำใน $O(m + n \log n \log (2m / n)) = O(m)$ เมื่อ $m = \Omega(n \log n \log \log n)$ จะประเมินค่าผลลัพธ์ที่จะแสดงว่าเวลากระทำที่จะประยุกต์เมื่อสามารถเลือกน้ำหนักของ $m / \log n$ เส้น และเป็นไปได้ที่น้ำหนักของเส้นยังคงอยู่ และงานวิจัยของ M. ZHU [9] ก็ได้กล่าวถึงลำดับผลคูณของเส้นตรง (Multiple Sequence Alignment (MSA)) เป็นวิธีหนึ่งที่ทันสมัยและสำคัญมากในทางชีววิทยา ปัญหา MSA เป็น NP – hard เพราะฉะนั้น การแก้ปัญหาเป็นความต้องการที่จะปรับข้อมูลให้อยู่ภายในเวลาที่เหมาะสม ในงานวิจัยนี้จะเสนอขั้นตอนวิธี MSA ใหม่ จะใช้ขั้นตอนวิธีการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของครุสกอลเป็นโครงสร้างต้นไม้ในการปรับลำดับ เวลาที่ซับซ้อนของ MSA เข้าใกล้ $O(n^3L^2)$ (เมื่อ n คือจำนวนของลำดับ และ L คือค่ามากสุดของทุกๆ ลำดับ) ขั้นตอนวิธีมีการยืนยันโดยผลลัพธ์ที่ได้มาจากการทดลอง

ซึ่งขั้นตอนวิธีการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของพริมและของครุสกอลเป็นวิธีการหาที่ต่างกันโดยได้อธิบายไว้ในหัวข้อก่อนหน้านี้แล้ว

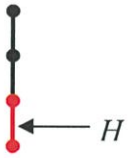
บทที่ 3
การปะติดกราฟ ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด และการประยุกต์

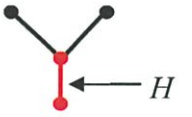
ในบทนี้ได้นำเสนอกราฟปะติดในรูปแบบต่างๆ รวมถึงการนำกราฟปะติดไปประยุกต์ใช้กับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimum Spanning Tree) และยังจะใช้กราฟปะติดไปประยุกต์ใช้กับปัญหาโครงข่ายอี-โลจิสติกส์

3.1 ตัวอย่างกราฟปะติดแบบต่าง ๆ

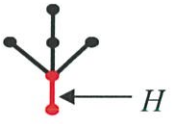
ผลลัพธ์ที่ได้จากการปะติดกราฟ 2 กราฟอาจได้มากกว่า 1 แบบ ซึ่งในส่วนนี้จะนำเสนอตัวอย่างการปะติดกราฟแบบต่างๆ และผลลัพธ์ทุกแบบที่สามารถเกิดขึ้นได้

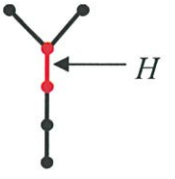
ตัวอย่างที่ 3.1.1

ให้กราฟ A คือ 

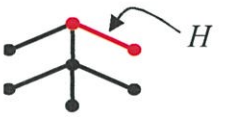
ให้กราฟ B คือ 

โดยที่ส่วนโคลน H เป็นส่วนของกราฟดังรูป จะได้ผลลัพธ์การปะติด คือ

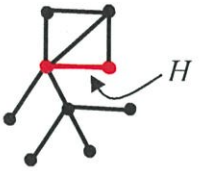
กราฟ $A \diamond B$ แบบที่ 1 หรือ $(A \diamond B)_1$ คือ 

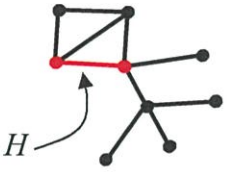
และกราฟ $A \diamond B$ แบบที่ 2 หรือ $(A \diamond B)_2$ คือ 

ตัวอย่างที่ 3.1.2

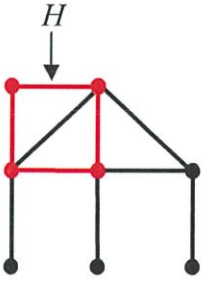
ให้กราฟ A คือ 

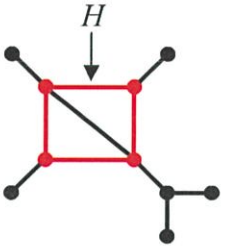
ให้กราฟ B คือ 

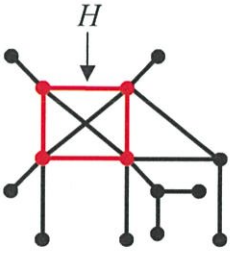
$(A \diamond B)_1$ คือ 

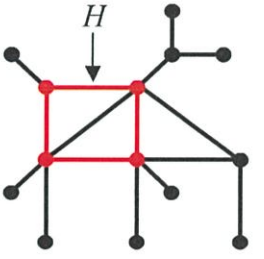
$(A \diamond B)_2$ คือ 

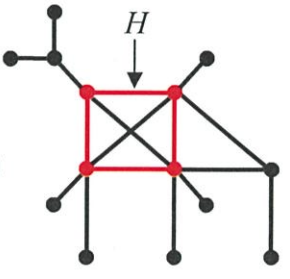
ตัวอย่างที่ 3.1.3

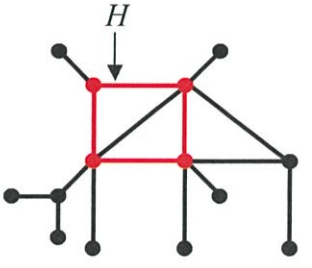
ให้กราฟ A คือ 

ให้กราฟ B คือ 

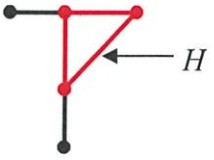
$(A \diamond B)_1$ คือ 

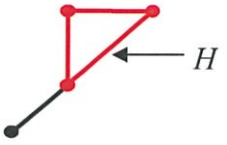
$(A \diamond B)_2$ คือ 

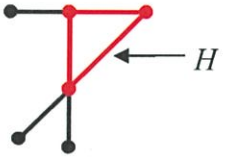
$(A \diamond B)_3$ คือ 

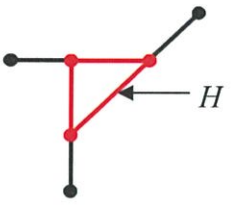
$(A \diamond B)_4$ คือ 

ตัวอย่างที่ 3.1.4

ให้กราฟ A คือ 

ให้กราฟ B คือ 

$(A \diamond B)_1$ คือ 

$(A \diamond B)_2$ คือ 

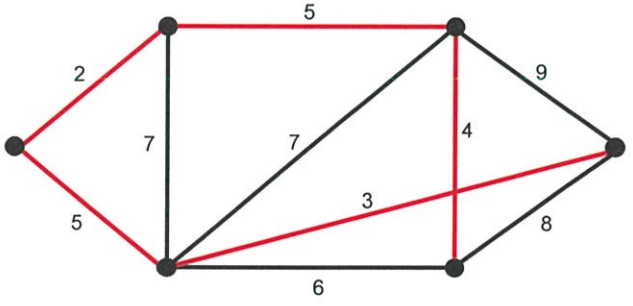
3.2 การปะติดกราฟต้นไม้แผ่ทั่วที่น้อยที่สุด

ในกรณีที่มีโครงข่าย 2 โครงข่าย และแต่ละโครงข่ายสามารถหาต้นไม้แผ่ทั่วที่น้อยที่สุดได้ การปะติดกันของโครงข่ายทั้ง 2 โครงข่ายนี้จะเกิดกรณีต่างๆ ขึ้นได้ดังต่อไปนี้

กรณีที่ 1 ทุกคู่เส้นของโคลนตรงส่วนที่ปะติดกัน จะมีอย่างน้อย 1 เส้นมาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

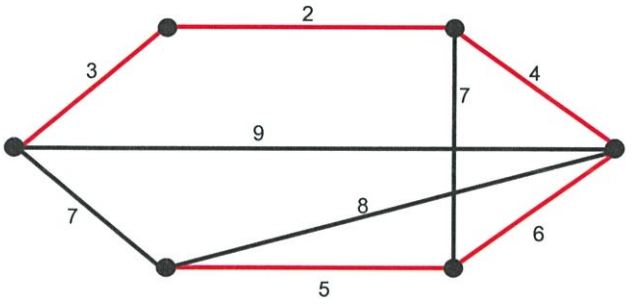
กรณีที่ 2 มีบางคู่เส้นของโคลนตรงส่วนที่ปะติดกัน ทั้ง 2 เส้นไม่ได้มาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

โครงข่าย M มีทั้งหมด 6 จุด 10 เส้น โดยที่มีน้ำหนักของแต่ละเส้นดังรูปที่ 3.1 ซึ่งจะได้คำตอบที่ดีที่สุดเท่ากับ 19 คือเส้นสีแดงดังในรูป



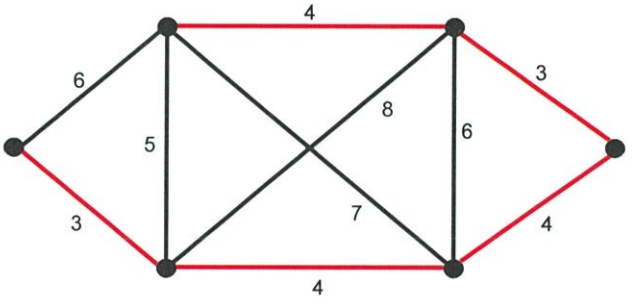
รูปที่ 3.1 โครงข่าย M

โครงข่าย N มีทั้งหมด 6 จุด 9 เส้น โดยที่มีน้ำหนักของแต่ละเส้นดังรูปที่ 3.2 ซึ่งจะได้คำตอบที่ดีที่สุดเท่ากับ 20 คือเส้นสีแดงดังในรูป



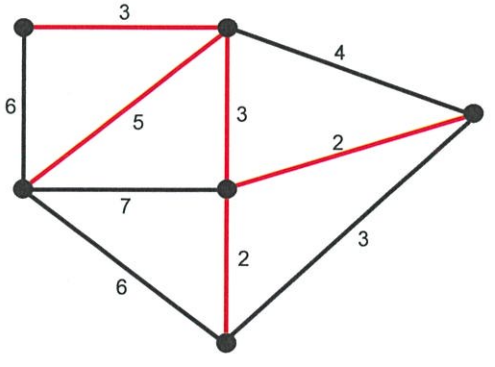
รูปที่ 3.2 โครงข่าย N

โครงข่าย O มีทั้งหมด 6 จุด 10 เส้น โดยที่มีน้ำหนักของแต่ละเส้นดังรูปที่ 3.3 ซึ่งจะได้คำตอบที่ดีที่สุดเท่ากับ 18 คือเส้นสีแดงดังในรูป



รูปที่ 3.3 โครงข่าย O

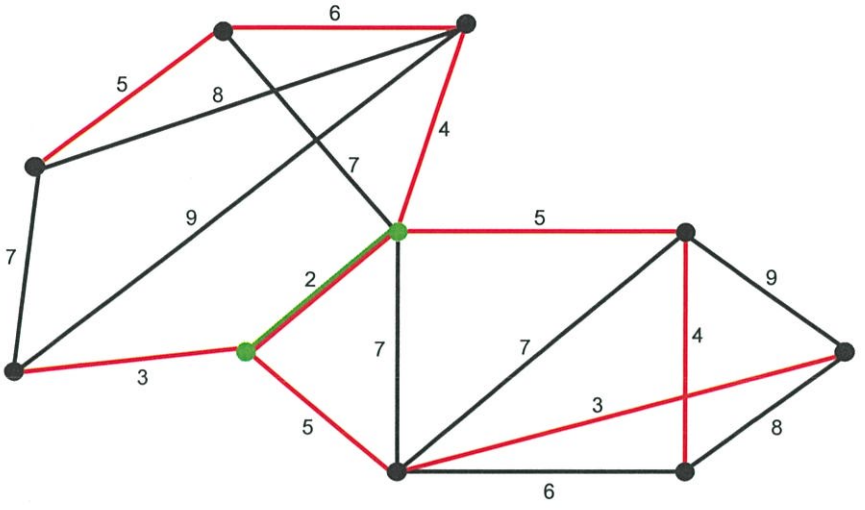
โครงข่าย P มีทั้งหมด 6 จุด 10 เส้น โดยที่มีน้ำหนักของแต่ละเส้นดังรูปที่ 3.4 ซึ่งจะได้คำตอบที่ดีที่สุดเท่ากับ 15 คือเส้นสีแดงดังในรูป



รูปที่ 3.4 โครงข่าย P

กรณีที่ 1 ทุกคู่เส้นของโคลนตรงส่วนที่ปะติดกันจะมีอย่างน้อย 1 เส้นมาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

ตัวอย่างที่ 3.2.1 การปะติดระหว่างโครงข่าย M และโครงข่าย N โดยเมื่อให้*โคลนมีเพียง 1 เส้น* และเป็นส่วนหนึ่งของคำตอบที่ดีที่สุดทั้ง 2 โครงข่าย คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $M \diamond N$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.5 โครงข่าย $M \diamond N$ ที่ $w_H = 2$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$ = 37

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย M

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย N

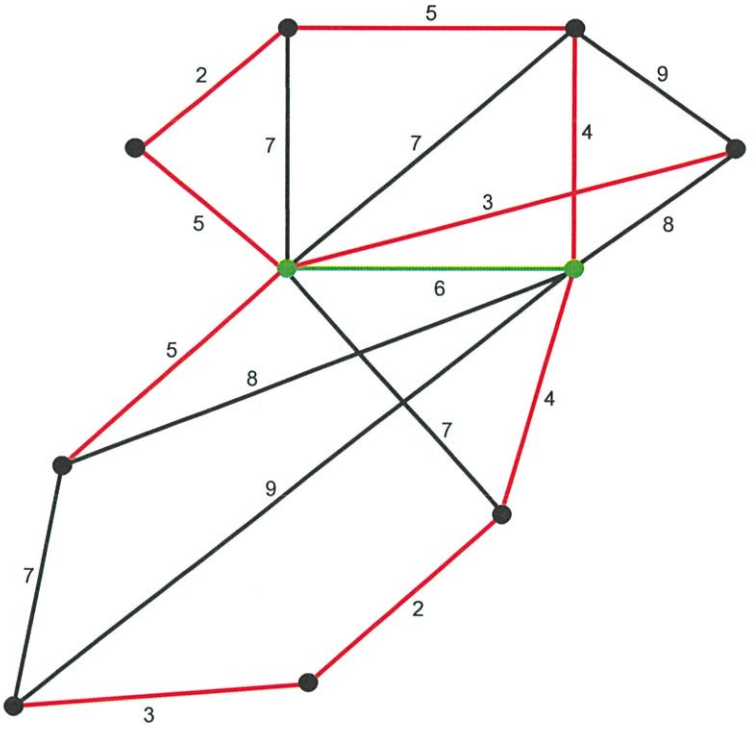
– น้ำหนักตรงส่วนที่โคลน

= 19 + 20 – 2

= 37

ซึ่งจะเห็นว่ามีค่าเท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$

ตัวอย่างที่ 3.2.2 การปะติดระหว่างโครงข่าย M และโครงข่าย N โดยเมื่อให้*โคลนมีเพียง 1 เส้น* และเป็นส่วนหนึ่งของคำตอบที่ดีที่สุดเพียง 1 โครงข่าย คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $M \diamond N$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.6 โครงข่าย $M \diamond N$ ที่ $w_H = 6$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$ = 33

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย M

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย N

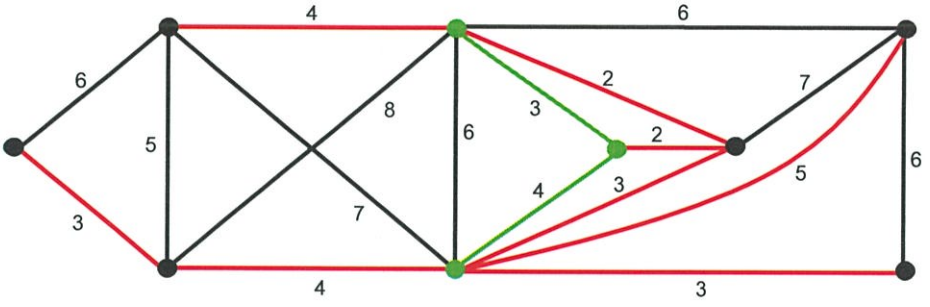
– น้ำหนักตรงส่วนที่โคลน

= 19 + 20 – 6

= 33

ซึ่งจะเห็นว่ามีค่าเท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$

ตัวอย่างที่ 3.2.3 การปะติดระหว่างโครงข่าย O และโครงข่าย P โดยเมื่อให้*โคลนมี 2 เส้น*โดยเส้นที่ปะติดกันเป็นส่วนหนึ่งของคำตอบที่ดีที่สุดของโครงข่าย O ส่วนเส้นที่ปะติดของโครงข่าย P เป็นเส้นที่ไม่ได้มาจากคำตอบที่ดีที่สุด คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $O \diamond P$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.7 โครงข่าย $O \diamond P$ ที่ $w_H = 7$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$ = 26

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย O

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย P

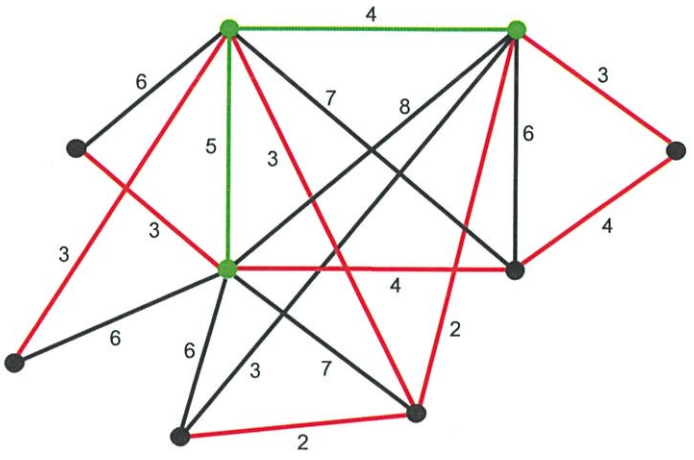
– น้ำหนักตรงส่วนที่โคลน

= 18 + 15 – 7

= 26

ซึ่งจะเห็นว่ามีค่าเท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$

ตัวอย่างที่ 3.2.4 การปะติดระหว่างโครงข่าย O และโครงข่าย P โดยเมื่อให้*โคลนมี 2 เส้น*โดยแต่ละโครงข่ายมีเส้นที่มาปะติดกันนั้น มาจากคำตอบที่ดีที่สุดและไม่ได้มาจากคำตอบที่ดีที่สุดอย่างละ 1 เส้น ซึ่งจะทำการปะติดสลับกัน คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $O \diamond P$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.8 โครงข่าย $O \diamond P$ ที่ $w_H = 9$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$ = 24

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย O

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย P

– น้ำหนักตรงส่วนที่โคลน

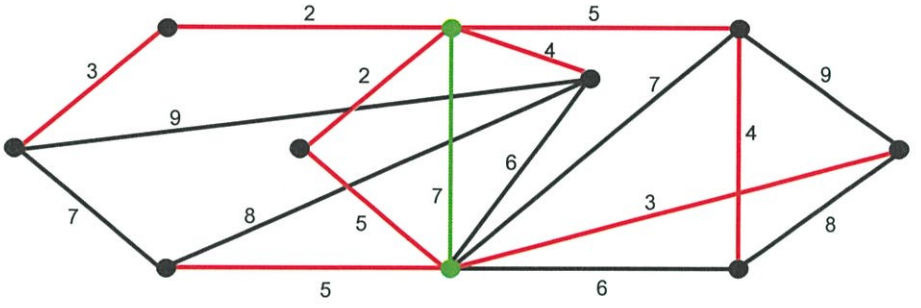
= 18 + 15 – 9

= 24

ซึ่งจะเห็นว่ามีค่าเท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$

กรณีที่ 2 มีบางคู่เส้นของโคลนตรงส่วนที่ปะติดกันทั้ง 2 เส้นไม่ได้มาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

ตัวอย่างที่ 3.2.5 การปะติดระหว่างโครงข่าย M และโครงข่าย N โดยเมื่อให้*โคลนมีเพียง 1 เส้น* และเป็นส่วนหนึ่งที่ไม่ได้มาจากคำตอบที่ดีที่สุดทั้ง 2 โครงข่าย คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $M \diamond N$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.9 โครงข่าย $M \diamond N$ ที่ $w_H = 7$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$ = 33

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย M

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย N

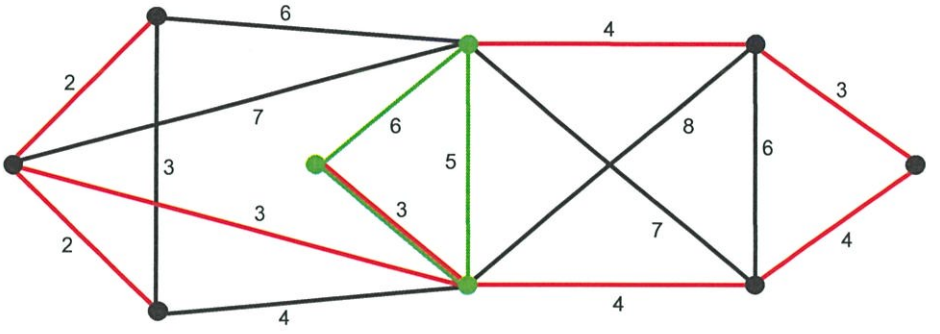
– น้ำหนักตรงส่วนที่โคลน

= 19 + 20 – 7

= 32

ซึ่งจะเห็นว่าในกรณีนี้น้ำหนักที่ได้มีค่าไม่เท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $M \diamond N$

ตัวอย่างที่ 3.2.6 การปะติดระหว่างโครงข่าย O และโครงข่าย P โดยเมื่อให้*โคลนมี 3 เส้น*โดยส่วนที่ปะติดกันนั้นมีอยู่ 1 คู่เส้นที่ไม่ได้มาจากส่วนของคำตอบที่ดีที่สุดของทั้ง 2 โครงข่าย คือส่วนสีเขียวดังในภาพ จะได้คำตอบที่ดีที่สุดของโครงข่ายใหม่ $O \diamond P$ เป็นดังเส้นสีแดงในรูป



รูปที่ 3.10 โครงข่าย $O \diamond P$ ที่ $w_H = 14$

โดยที่ผลรวมของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$ = 25

ขณะที่ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย O

+ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย P

– น้ำหนักตรงส่วนที่โคลน

= 18 + 15 – 14

= 19

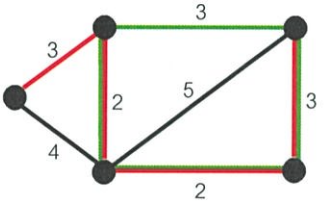
ซึ่งจะเห็นว่าในกรณีนี้น้ำหนักที่ได้มีค่าไม่เท่ากับน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $O \diamond P$

จากกรณีที่ 1 สามารถสรุปออกมาได้ 1 ทฤษฎี ซึ่งทุกคู่เส้นของโคลนตรงส่วนที่ปะติดกัน จะมีอย่างน้อย 1 เส้นมาจากคำตอบที่ดีที่สุดจากกราฟ G_1 หรือ G_2 แต่มีข้อยกเว้นอยู่ว่าโคลนต้องไม่เป็นวง เพราะว่าถ้าโคลนเป็นวงจะทำให้ $w_{T^*} \neq w_{T_1} + w_{T_2} - w_H$ ดังตัวอย่างต่อไปนี้

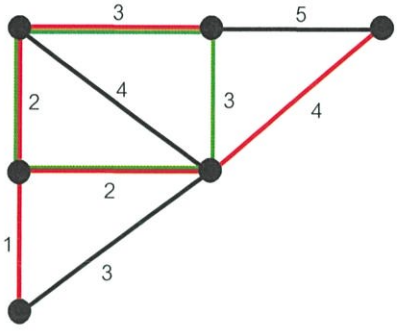
โดยให้ เส้นสีแดงคือต้นไม้แผ่ทั่วที่น้อยที่สุดในแต่ละกราฟ

เส้นสีเขียวคือส่วนที่โคลน

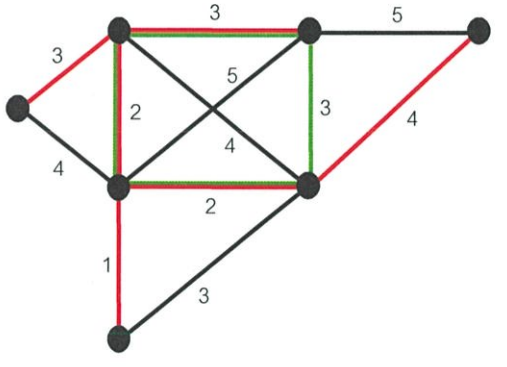
ตัวอย่างที่ 1 แสดงถึงโคลนที่เป็นวง



G_1 มี $w_{T_1} = 10$



G_2 มี $w_{T_2} = 12$

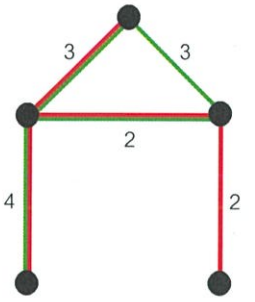


$G_1 \diamond G_2$ มี $w_{T^*} = 15$

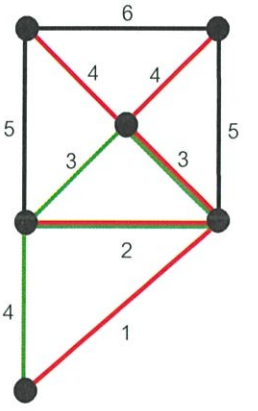
รูปที่ 3.11 ตัวอย่างโคลนเป็นวงแบบที่ 1

ซึ่งผลลัพธ์ที่ได้ของ w_{T^*} จะไม่เท่ากับ $w_{T_1} + w_{T_2} - w_H$ = 10 + 12 – 10 = 12

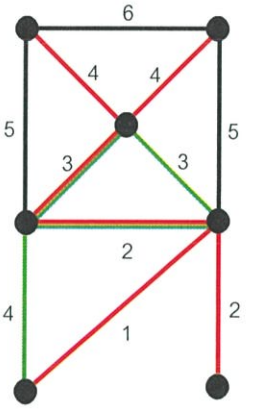
ตัวอย่างที่ 2 แสดงถึงโคลนที่มีสวนประกอบที่เป็นวง



G_1 มี $w_{T_1} = 11$



G_2 มี $w_{T_2} = 14$



$G_1 \diamond G_2$ มี $w_{T^*} = 16$

รูปที่ 3.12 ตัวอย่างโคลนเป็นวงแบบที่ 2

ซึ่งผลลัพธ์ที่ได้ของ w_{T^*} จะไม่เท่ากับ $w_{T_1} + w_{T_2} - w_H$ = 11 + 14 – 12 = 13

จากกรณีที่ 1 สามารถสรุปออกมาเป็นทฤษฎีได้ดังนี้

ทฤษฎีบทที่ 1 กำหนดให้ G_1 และ G_2 เป็นกราฟเชื่อมโยงที่มีน้ำหนัก และ H เป็นโคลนที่ไม่มีวงของ $G_1 \underset{H}{\diamond} G_2$ โดย T_1 และ T_2 เป็นต้นไม้แผ่ทั่วที่น้อยที่สุด (MST) ของ G_1 และ G_2 ตามลำดับ

ถ้า (ทุกๆ $e \in H, e \in T_1$ หรือ $e \in T_2$) แล้ว $w_{T^*} = w_{T_1} + w_{T_2} - w_H$

โดย T^* เป็น MST ของ $G_1 \underset{H}{\diamond} G_2$

พิสูจน์ (โดยวิธีอุปนัยเชิงคณิตศาสตร์)

สมมติให้ G_1 และ G_2 เป็นกราฟเชื่อมโยงที่มีน้ำหนัก และ H เป็นโคลนที่ไม่มีวงของ $G_1 \underset{H}{\diamond} G_2$ โดย T_1 และ T_2 เป็นต้นไม้แผ่ทั่วที่น้อยที่สุด (MST) ของ G_1 และ G_2 ตามลำดับ และ T^* เป็น MST ของ $G_1 \underset{H}{\diamond} G_2$

ให้ $G_1 = (V_1, E_1)$ $G_2 = (V_2, E_2)$

$|V(G_1)| = n_1$ $|V(G_2)| = n_2$

$|E(T_1)| = n_1 - 1 = p$ $|E(T_2)| = n_2 - 1 = q$

จะใช้วิธีอุปนัยบนค่า h เมื่อ h แทนจำนวนของเส้นในโคลน $h = |E(H)|$ นั่นคือ

กำหนดให้ $P(h)$: ถ้า (ทุกๆ $e \in H, e \in T_1$ หรือ $e \in T_2$) แล้ว $w_{T^*} = w_{T_1} + w_{T_2} - w_H$

ขั้นที่ 1 ของวิธีอุปนัย ต้องพิสูจน์ว่า $P(1)$ เป็นจริง

โดยไม่สูญเสียความเป็นทั่วไป (Without loss of generality) สมมติให้ e ที่เป็นเพียงเส้นเดียวในโคลน H นั้น มาจาก T_1 ต้องแสดงว่า $w_{T^*} = w_{T_1} + w_{T_2} - w_H$

นั่นคือ $w_{T^*} = w_{T_1} + w_{T_2} - w_e$

โดยขั้นตอนวิธีของครุสกอล เราทราบว่า

$e_1 \le e_2 \le \ldots \le e_p$ เมื่อ $\{e_1, \ldots, e_p\}$ เป็นเซตของเส้นใน T_1

$e'_1 \le e'_2 \le \ldots \le e'_q$ เมื่อ $\{e'_1, \ldots, e'_q\}$ เป็นเซตของเส้นใน T_2

เนื่องจาก $e \in T_1$ ดังนั้น $e \in \{e_1, \ldots, e_p\}$

สำหรับ $G_1 \diamond G_2$ จะเพิ่ม $n_2 - 2$ จุดไปยัง n_1 จุด เนื่องจาก H มีเพียง 1 เส้น (2 จุด)

จำนวนของจุดใน $G_1 \diamond G_2 = n_1 + n_2 - 2$

จำนวนของเส้นใน $T^* = |V(G_1 \diamond G_2)| - 1 = n_1 + n_2 - 2 - 1 = (n_1 - 1) + (n_2 - 1) - 1 = p + q - 1$

จะหาต้นไม้แผ่ทั่วที่น้อยที่สุดใหม่ใน $G_1 \diamond G_2$ โดยใช้ขั้นตอนวิธีของครุสกอล ซึ่งจะทำการเรียงเส้นทุกเส้นใหม่ทั้งหมดที่มีน้ำหนักจากน้อยไปมาก และจะเลือกเส้นที่ไม่ประกอบกันขึ้นเป็นวง โดยเส้นทั้ง ($p + q - 1$) เส้น จะถูกเลือกมาจากบรรดาเส้นในเซต $A = \{e_1, \ldots, e_p, e'_1, \ldots, e'_q\}$ ก่อน เนื่องจากเป็นเส้นที่มีน้ำหนักน้อยกว่าเส้นอื่นๆ ที่เหลือ โดยจะสามารถแบ่งออกได้เป็น 2 กรณีคือ

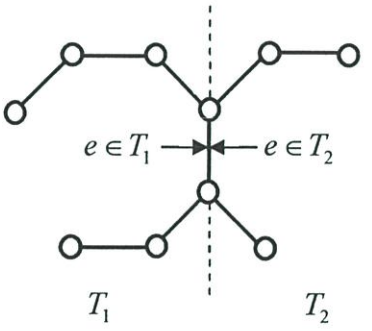
กรณีที่ 1 เมื่อ $e \in T_2$

ดังนั้น $A = \{e_1,...,e_p,e'_1,...,e'_q\}$ จะมีเส้นที่ไม่ซ้ำกันเพียง $p+q-1$ เนื่องจาก $e \in T_1$ และ T_2 ดังตัวอย่างในรูปที่ 3.11 พิจารณา $G_1 \underset{H}{\diamond} G_2$ จะเห็นว่า n_1 จุดใน G_1 จะถูกแผ่ทั่วโดย $\{e_1,...,e_p\}$ และ n_2-2 จุดใน G_2 จะถูกแผ่ทั่วโดย $\{e'_1,...,e'_q\}$ และจุดใน G_2 ก็จะถูกเชื่อมโยงโดยเส้น $e \in T_1$ เนื่องจาก T_1 และ T_2 เป็นต้นไม้ โดยที่ $e \in T_1$ และ T_2 จึงทำให้เซต A เป็นเหมือนกับการปะติดต้นไม้ทั้งสอง โดยทฤษฎีบทที่ 2.2.5 จึงได้ว่าเซต A เป็นต้นไม้ ซึ่งแผ่ทั่วทุกจุดใน $G_1 \diamond G_2$ นอกจากนี้น้ำหนักของต้นไม้แผ่ทั่วถึง A นี้ มีค่าน้อยที่สุดเนื่องจากทุกเส้นใน A มาจากเส้นใน T_1 และ T_2 ซึ่งผ่านการเลือกโดยวิธีครุสกอลแล้ว

ดังนั้น $A = \{e_1,...,e_p,e'_1,...,e'_q\}$ ที่มี $p+q-1$ เส้นนี้จึงเป็นต้นไม้แผ่ทั่วที่น้อยที่สุดของ $G_1 \underset{H}{\diamond} G_2$

นั่นคือ $w_{T^*} = w_{T_1} + w_{T_2} - w_e$

$= w_{T_1} + w_{T_2} - w_H$



รูปที่ 3.13 ตัวอย่างสำหรับกรณีที่ 1 ของวิธีอุปนัยขั้นที่

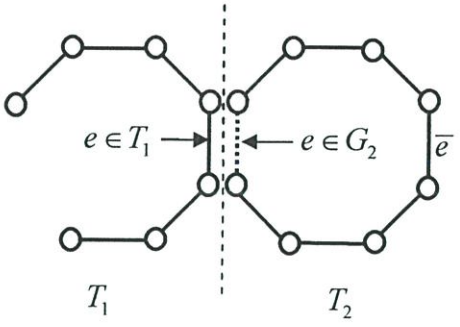
กรณีที่ 2 เมื่อ $e \notin T_2$

ดังนั้น $A = \{e_1,...,e_p,e'_1,...,e'_q\}$ จะมีจำนวนเส้นที่ไม่ซ้ำกันเท่ากับ $p+q$ เนื่องจาก $e \in T_1$ แต่ $e \notin T_2$ ดังตัวอย่างในรูปที่ 3.12 โดยขั้นตอนวิธีของครุสกอลในการสร้าง T_2 จะได้ว่า จะต้องมี $\bar{e} \in \{e'_1,...,e'_q\}$ โดยที่ $\bar{e} \le e$ ซึ่งไม่ทำให้เกิดวงสำหรับบรรดาจุดต่างๆ ของ G_2 และจะทำให้ $A' = (\{e_1,...,e_p\} \cup \{e'_1,...,e'_q\}) - e$ ครอบคลุมทุกจุดใน $G_1 \diamond G_2$ โดยไม่ทำให้เกิดวง นอกจากนี้น้ำหนักของต้นไม้แผ่ทั่วถึง A' นี้ มีค่าน้อยที่สุดเนื่องจากทุกเส้นใน A' มาจากเส้นใน T_1 และ T_2 ซึ่งผ่านการเลือกโดยวิธีครุสกอลแล้ว

สำหรับกรณีที่ $e < \bar{e}$ แต่เนื่องจาก $e \notin T_2$ แสดงว่าโดยวิธีของครุสกอล e นี้จะทำให้เกิดวงกับบรรดาเส้นอื่นๆ ที่เหลือของ T_2 ซึ่งเมื่อทำการสร้าง T^* ของ $G_1 \underset{H}{\diamond} G_2$ เส้น e ที่มีค่าน้อยกว่า $\bar{e}$ นี้ ก็จะไม่ถูกเลือกให้อยู่ใน T^* อีก เนื่องจากเหตุผลเดียวกัน นั่นก็คือ การลบ $e \in T_1$ ออกจากเซต A จึงทำให้เซต A มี $(p+q-1)$ เส้น

ดังนั้น น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุด A' คือ $w_{T^*} = w_{T_1} + w_{T_2} - w_e$

$= w_{T_1} + w_{T_2} - w_H$



รูปที่ 3.14 ตัวอย่างสำหรับกรณีที่ 2 ของวิธีอุปนัยขั้นที่ 1

จากทั้งสองกรณีข้างต้น จึงได้ว่า $P(1)$ เป็นจริง

ขั้นที่ 2 ของวิธีอุปนัย คือต้องพิสูจน์ว่า $P(h-1) \rightarrow P(h)$ เป็นจริง

กำหนดให้ T^x คือ MST ของ $G_1 \underset{H}{\Leftrightarrow} G_2$ เมื่อ $|E(H)| = x$

H_x คือโคลนเมื่อ $|E(H)| = x$

สมมติให้ $P(h-1)$ เป็นจริง

นั่นคือ $P(h-1)$: ถ้า (ทุกๆ $e_i \in H, e_i \in T_1$ หรือ $e_i \in T_2$ โดยที่ $i = 1,...,h-1$)

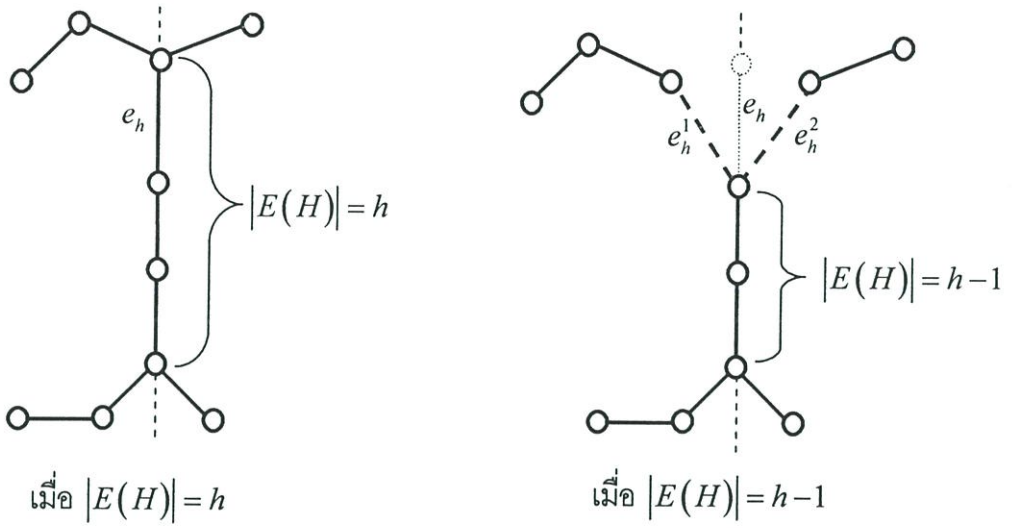
แล้ว $w_{T^{h-1}} = w_{T_1} + w_{T_2} - w_{H_{h-1}}$

$$= w_{T_1} + w_{T_2} - \sum_{i=1}^{h-1} w_{e_i}$$

จะพิสูจน์ว่า $P(h)$ เป็นจริง

สมมติให้ ทุกๆ e_i, $e_i \in T_1$ หรือ $e_i \in T_2$ โดยที่ $i = 1,...,h$ ดังตัวอย่างในรูปที่ 3.13 จะต้องแสดงว่า $w_{T^h} = w_{T_1} + w_{T_2} - w_{H_h}$

เนื่องจาก H ไม่มีวงจึงสามารถเขียนตัวอย่างเพื่อประกอบการพิสูจน์ในขั้นที่ 2 ได้ดังรูป 3.15 โดยไม่สูญเสียความเป็นทั่วไป สมมติให้เส้นหนึ่งเส้นในโคลน H ซึ่งจะให้ชื่อว่า e_h ไม่ทำการปะติด ดังตัวอย่างในรูปที่ 3.15 จะทำการแยกเส้น e_h นี้ออกเป็น 2 เส้น ในกราฟ G_1 และ G_2 คือ e_h^1 และ e_h^2 ตามลำดับ



รูปที่ 3.15 ตัวอย่างสำหรับขั้นที่ 2 ของวิธีอุปนัย

ซึ่งจะทำให้ $|E(H)| = h-1$ เส้น

และจากที่สมมติให้ $P(h-1)$ เป็นจริง จึงทำให้ได้ข้อสรุปของข้อความ $P(h-1)$

นั่นคือ $w_{T^{h-1}} = w_{T_1} + w_{T_2} - \sum_{i=1}^{h-1} w_{e_i}$ เป็นจริง

แต่เมื่อทำการพิสูจน์ $P(h)$ เราจะต้องทำการปะติดเส้น e_h กลับเข้าไป

เนื่องจากตามข้อสมมติของข้อความ $P(h)$ ที่กำหนดว่า ทุกเส้นในโคลนจะต้องมาจาก T_1 หรือ T_2

นั่นคือ e_h^1 หรือ e_h^2 จะต้องเป็นเส้นที่สามารถเขียนให้อยู่ในรูปของ MST ของ $G_1 \underset{H_{h-1}}{\Phi} G_2$ ได้

จะได้ว่าจำนวนของจุดใน $G_1 \underset{H_h}{\Phi} G_2$ จะน้อยกว่าจำนวนของจุดใน $G_1 \underset{H_{h-1}}{\Phi} G_2$ อยู่ 1 จุด

เพราะฉะนั้น จำนวนของเส้นในต้นไม้แผ่ทั่วใดๆ ใน $G_1 \underset{H_h}{\Phi} G_2$ จะน้อยกว่าจำนวนของเส้นในต้นไม้แผ่ทั่วใดๆ ใน $G_1 \underset{H_{h-1}}{\Phi} G_2$ อยู่ 1 เส้น ซึ่งเมื่อทำการปะติดเส้น e_h เพิ่มจากขั้นของ $P(h-1)$ เราสามารถสร้าง T^h จาก T^{h-1} ได้โดยการเอาเส้นออกหนึ่งเส้นจะสามารถแยกออกได้เป็น 2 กรณี คล้ายกับในขั้นตอนที่ 1 ของวิธีอุปนัยที่ได้แสดงไปแล้ว โดยอาจเทียบเส้น e_h นี้ได้กับเส้น e ในขั้นตอนที่ 1 จึงสรุปได้ว่า น้ำหนักต้นไม้แผ่ทั่วที่น้อยที่สุดของ $G_1 \underset{H_h}{\Phi} G_2$ คือ w_{T^h} มีค่าดังนี้

$$w_{T^h} = w_{T^{h-1}} - w_{e_h}$$

$$w_{T^h} = \left(w_{T_1} + w_{T_2} - \sum_{i=1}^{h-1} w_{e_i} \right) - w_{e_h}$$

$$w_{T^h} = w_{T_1} + w_{T_2} - \sum_{i=1}^{h} w_{e_i}$$

$$w_{T^h} = w_{T_1} + w_{T_2} - w_{H_h}$$ □

เสร็จสินการพิสูจน์ขั้นที่ 2 ของวิธีอุปนัย และเป็นอันเสร็จสิ้นการพิสูจน์ทฤษฎีด้วย

3.3 ตัวอย่างปัญหาอี-โลจิสติกส์

สมมติให้ นิคมอุตสาหกรรม 2 นิคมที่อยู่ภายใต้การดำเนินงานของเอกชน 2 แห่ง แต่อยู่ในพื้นที่อุตสาหกรรมเดียวกัน แต่ละนิคมมีการบริหารจัดการเป็นเอกเทศ ต้องการวางแผนในเรื่องสาธารณูปโภคพื้นฐานต่างๆ เช่น โครงข่ายสายโทรศัพท์ ไฟฟ้า น้ำประปา หรือระบบอินทราเน็ต โดยบางโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ เหล่านี้จะช่วยส่งเสริมกิจกรรมทางด้านอี-โลจิสติกส์ที่เกี่ยวข้องกับการไหลของข้อมูล (Information flow) เพื่อรองรับโรงงานรูปแบบต่างๆ ที่จะเข้ามาอยู่ในนิคมอุตสาหกรรม ยกตัวอย่างเช่น มีนิคมอุตสาหกรรมหนึ่งที่ได้เปิดให้บริการแล้ว และมีโครงข่ายสาธารณูปโภคพื้นฐานอยู่แล้ว (ซึ่งเปรียบได้กับเป็นโครงข่ายที่มีต้นไม้แผ่ทั่วที่น้อยที่สุดอยู่แล้ว) ในขณะที่มีบริษัทอีกรายหนึ่งที่ต้องการเปิดนิคมอุตสาหกรรมแห่งใหม่ติดกับพื้นที่เดิมในนิคมอุตสาหกรรมเก่าที่มีอยู่ และกำลังทำการศึกษาความเป็นไปได้ของโครงการ (Feasibility Study) โดยเฉพาะในส่วนที่เกี่ยวข้องกับการก่อสร้างโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ โดยได้ศึกษาตำแหน่งที่ตั้งของสถานประกอบการภายในนิคมอุตสาหกรรมแห่งใหม่และเส้นทางการเชื่อมต่อโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ ดังกล่าวไว้แล้ว (เปรียบเหมือนเป็นอีกโครงข่ายหนึ่งที่ทราบต้นไม้แผ่ทั่วที่น้อยที่สุดแล้วเช่นกัน แต่ยังสามารถปรับเปลี่ยนเส้นทางได้ เนื่องจากยังไม่ได้ดำเนินการสร้างจริง) ต่อมาบริษัทอาจพิจารณาการเข้าไปซื้อนิคมอุตสาหกรรมเดิมที่มีอยู่แล้วและผนวกเข้ากับนิคมแห่งใหม่ที่ได้ทำการศึกษาความเป็นไปได้ไว้แล้ว ซึ่งจะเปรียบเสมือนเป็นการปะติด 2 โครงข่าย โดยจะเกิดกรณีของการปะติดได้หลายแบบ โดยทฤษฎีที่เกี่ยวข้องกับกราฟปะติดที่ได้นำเสนอไว้ก่อนหน้า จะได้เข้ามามีบทบาทในการเชื่อมโครงข่ายสาธารณูปโภครวมของทั้ง 2 นิคมอุตสาหกรรมได้ ยกตัวอย่างปัญหาเช่น นิคม ก ต้องการวางโครงข่ายสายเคเบิ้ลใยแก้วไปยังสถานประกอบการต่างๆ ในนิคม เช่น โรงงาน คลังสินค้า ดังนี้ A,B,C,D,E และ F โดยมีโครงข่ายดังโครงข่าย ก ในตารางที่ 3.1 แต่มีนิคมอุตสาหกรรม ข ได้วางโครงข่ายสายเคเบิ้ลใยแก้วไปยังสถานประกอบการต่างๆ ในนิคมไว้แล้ว ดังนี้ A,J,C,D,H และ G โดยมีโครงข่ายดังโครงข่าย ข ในตารางที่ 3.2 ซึ่งทางนิคมอุตสาหกรรม ก ยังไม่ได้ทำการวางเคเบิ้ลใยแก้วของโครงข่าย ก ดังนั้นทางนิคมอุตสาหกรรมจึงนำโครงข่ายทั้ง 2 มาพิจารณาดูว่าสามารถนำโครงข่าย ก ไปเชื่อมต่อ (ปะติด) กับโครงข่าย ข ได้หรือไม่เพื่อให้เกิดประโยชน์ในการลดต้นทุนในการวางโครงข่ายสายเคเบิ้ลใยแก้วโดยโครงข่าย ก และโครงข่าย ข มีระยะทางที่สั้นที่สุดอยู่แล้ว ระยะทางที่สั้นที่สุดของโครงข่าย ก คือ 17 หน่วย และโครงข่าย ข คือ 24 หน่วย หรือที่เรียกว่าปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดนั่นเอง

จากปัญหาดังกล่าวนี้สามารถนำมาแปลงให้เป็นปัญหากราฟได้ดังนี้ กำหนดให้

จุด แทนสถานประกอบการต่างๆ

เส้น แทนเส้นทางระหว่างสถานประกอบการต่างๆ

ตารางแสดงน้ำหนักของกราฟของโครงข่าย ก และโครงข่าย ข

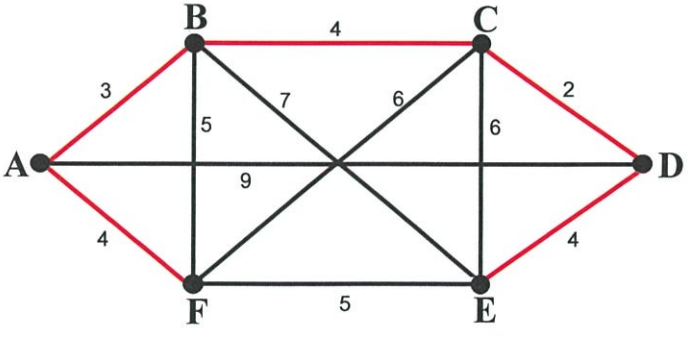
	A	B	C	D	E	F
A	-	3	-	9	-	4
B	3	-	4	-	7	5
C	-	4	-	2	6	6
D	9	-	2	-	4	-
E	-	7	6	4	-	5
F	4	5	6	-	5	-

ตาราง 3.1 โครงข่าย ก มีน้ำหนักบนเส้นเชื่อมดังตารางนี้

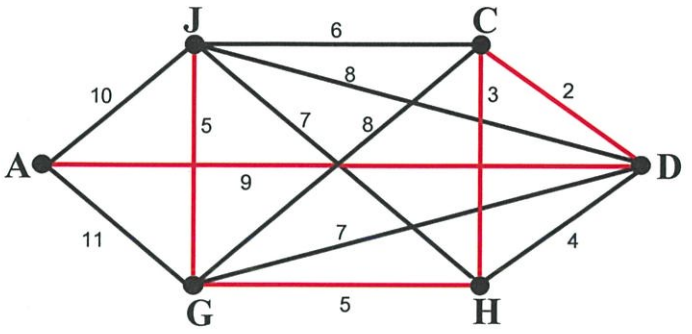
	A	J	C	D	H	G
A	-	10	-	9	-	11
J	10	-	6	8	7	5
C	-	6	-	2	3	8
D	9	8	2	-	4	7
H	-	7	3	4	-	5
G	11	5	8	7	5	-

ตาราง 3.2 โครงข่าย ข มีน้ำหนักบนเส้นเชื่อมดังตารางนี้

วาดเป็นกราฟได้ดังนี้



รูปที่ 3.16 โครงข่าย ก

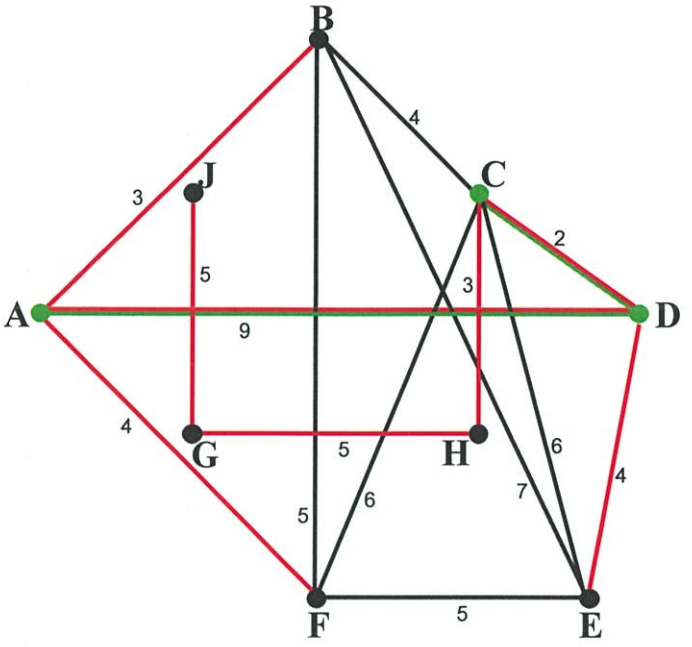


รูปที่ 3.17 โครงข่าย ข

การพิจารณาการปะติดจะทำตามขั้นตอนดังนี้

1.พิจารณาดูว่าโครงข่าย ก และ โครงข่าย ข มีเส้นทางไหนบ้างที่มีเส้นทางเดียวกัน โดยการพิจารณาครั้งแรกนั้นจะเลือกดูที่เส้นทางของโครงข่าย ข ที่สร้างโครงข่ายไว้แล้ว ว่ามีเส้นทางใดบ้างที่มีเส้นทางเดียวกับโครงข่าย ก ซึ่งจะสังเกตเห็นว่ามีเส้นทางจากสถานประกอบการ A ไปยังสถานประกอบการ D และเส้นทางจากสถานประกอบการ C ไปยังสถานประกอบการ D ที่มีเส้นทางเดียวกัน

2.ใช้นิยามของกราฟปะติด นั่นคือ เส้นทางจากสถานประกอบการ A ไปยังสถานประกอบการ D ของโครงข่าย ก ปะติดกับเส้นทางจากสถานประกอบการ A ไปยังสถานประกอบการ D ของโครงข่าย ข และเส้นทางจากสถานประกอบการ C ไปยังสถานประกอบการ D ของโครงข่าย ก ปะติดกับเส้นทางจากสถานประกอบการ C ไปยังสถานประกอบการ D ของโครงข่าย ข จะได้ดังรูป



รูปที่ 3.18 โครงข่าย ก และ ข ที่ปะติดกันที่จุด AD,DC

จะเห็นว่าจากเดิมโครงข่าย ก และ ข มีระยะทางที่สั้นที่สุดรวมกันแล้วเท่ากับ 41 หน่วย แต่เมื่อนำโครงข่ายทั้ง 2 มาปะติดกันแล้วจะทำให้ระยะทางรวมที่สั้นที่สุดทั้งหมดเท่ากับ 35 หน่วย ซึ่งจะเห็นว่าระยะทางลดน้อยลง

จากเหตุการณ์ที่สมมติขึ้นมานี้จะเห็นว่าการใช้กราฟปะติดมาประยุกต์ใช้กับโครงข่ายอี-โลจิสติกส์ ดังกล่าวจะช่วยในการประหยัดต้นทุนในการสร้างโครงข่ายสาธารณูปโภคพื้นฐานได้จริง

บทที่ 4
การออกแบบโปรแกรมคอมพิวเตอร์สำหรับการปะติดกราฟและปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ในบทนี้จะกล่าวถึงขั้นตอนการออกแบบโปรแกรมคอมพิวเตอร์ เพื่อทำการปะติดกราฟ โดยการปะติดจะทำการปะติดคราวละ 2 โครงข่าย การใส่ค่าและแสดงผลของแต่ละโครงข่ายในลักษณะของ Edge List โดยทำการเก็บจำนวนจุดในกราฟ และรายการแสดงชื่อเส้นทั้งหมด ซึ่งโปรแกรมนี้ได้ทำการเขียนบน Microsoft visual studio 2005 ด้วยภาษา C++ บนระบบปฏิบัติการ Windows XP การใส่ค่าของจุดในโครงข่ายจะมีลักษณะดังนี้

---'--- -

โปรแกรมจะรับค่าในแต่ละเส้นทั้งหมด 7 ตำแหน่ง โดย 1 จุดจะรับค่า 3 ตำแหน่ง ซึ่ง 6 ตำแหน่งแรกจะเป็นการรับจุดปลาย (end node) ของแต่ละเส้น ตำแหน่งแรกของแต่ละจุดคือชื่อโครงข่าย (โครงข่าย *A* ใช้หมายเลข 1 โครงข่าย *B* ใช้หมายเลข 2) ตำแหน่งที่ 2 และตำแหน่งที่ 3 ของแต่ละจุดคือหมายเลขจุด (ใส่ค่าได้ 100 จุด) ตำแหน่งที่ 7 คือน้ำหนักบนเส้นเชื่อม (กรณีที่เลือกโปรแกรมแบบไม่มีน้ำหนักโปรแกรมจะไม่รับค่าตรงตำแหน่งที่ 7)

โปรแกรมคอมพิวเตอร์ที่ทำการสร้างขึ้น จะมีหน้าที่หลักๆ ในการทำงาน เช่น ทำการปะติดกราฟ หาต้นไม้แผ่ทั่วที่น้อยที่สุด โดยจะแบ่งออกเป็น 4 แบบ ได้แก่

4.1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก

4.2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก

4.3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด

4.4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

ขั้นตอนวิธีการทำงานของโปรแกรม

เลือกเมนูโปรแกรมในการทำงาน 4 โปรแกรมดังนี้

โปรแกรมที่ 1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก ซึ่งมีขั้นตอนในการดำเนินงานดังนี้

ขั้นที่ 1 รับค่าโครงข่าย A

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น

ขั้นที่ 2 รับค่าโครงข่าย B

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น

ขั้นที่ 3 รับค่าส่วนที่โคลน

- รับจำนวนเส้นในส่วนโคลน
- รับเส้นที่มาปะติดกันทีละคู่เส้น (เส้นในส่วนโคลนที่ใส่ค่าเข้าไป ถ้าไม่เชื่อมโยง โปรแกรมจะไม่ทำงาน, ใส่จุดผิดโปรแกรมก็จะไม่ทำงานเช่นกัน)

ขั้นที่ 4 โปรแกรมทำการปะติดโดยใช้โครงข่าย A เป็นหลัก

โปรแกรมที่ 2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก ซึ่งมีขั้นตอนในการดำเนินงานดังนี้

ขั้นที่ 1 รับค่าโครงข่าย A

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 2 รับค่าโครงข่าย B

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 3 รับค่าส่วนที่โคลน

- รับจำนวนเส้นในส่วนโคลน
- รับเส้นที่มาปะติดกันทีละคู่เส้น (เส้นในส่วนโคลนที่ใส่ค่าเข้าไป ถ้าไม่เชื่อมโยง โปรแกรมจะไม่ทำงาน, ใส่จุดผิดโปรแกรมก็จะไม่ทำงานเช่นกัน)

ขั้นที่ 4 โปรแกรมทำการปะติดโดยใช้โครงข่าย A เป็นหลัก

โปรแกรมที่ 3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด ซึ่งมีขั้นตอนในการดำเนินงานดังนี้

ขั้นที่ 1 รับค่าโครงข่าย A

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 2 โปรแกรมคำนวณหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 3 โปรแกรมรวมค่าบนต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 4 รับค่าโครงข่าย B

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 5 โปรแกรมคำนวณหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 6 โปรแกรมรวมค่าบนต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 7 รับค่าส่วนที่โคลน

- รับจำนวนเส้นในส่วนโคลน
- รับเส้นที่มาปะติดกันทีละคู่เส้น (เส้นในส่วนโคลนที่ใส่ค่าเข้าไป ถ้าไม่เชื่อมโยง โปรแกรมจะไม่ทำงาน, ใส่จุดผิดโปรแกรมก็จะไม่ทำงานเช่นกัน)

ขั้นที่ 8 โปรแกรมทำการปะติดโดยใช้โครงข่าย A เป็นหลัก

ขั้นที่ 9 โปรแกรมคำนวณหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่10 โปรแกรมรวมค่าบนต้นไม้แผ่ทั่วที่น้อยที่สุด

โปรแกรมที่ 4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์ ซึ่งมีขั้นตอนในการดำเนินงานดังนี้

ขั้นที่ 1 รับค่าโครงข่าย A

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 2 รับค่าของต้นไม้แผ่ทั่วถึงของโครงข่าย A

- รับจุดปลายของแต่ละเส้น (เส้นที่ใส่เข้าไปถ้ายังไม่เป็นต้นไม้แผ่ทั่วถึง โปรแกรมจะให้ใส่ค่าใหม่)

ขั้นที่ 3 รับค่าโครงข่าย B

- รับจำนวนเส้นในโครงข่าย
- รับจุดปลายของแต่ละเส้น
- รับน้ำหนักของแต่ละเส้น

ขั้นที่ 4 โปรแกรมคำนวณหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 5 โปรแกรมรวมค่าบนต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 6 รับค่าส่วนที่โคลน

- รับจำนวนเส้นในส่วนโคลน
- รับเส้นที่มาปะติดกันทีละคู่เส้น (เส้นในส่วนโคลนที่ใส่ค่าเข้าไป ถ้าไม่เชื่อมโยง โปรแกรมจะไม่ทำงาน, ใส่จุดผิดโปรแกรมก็จะไม่ทำงานเช่นกัน)

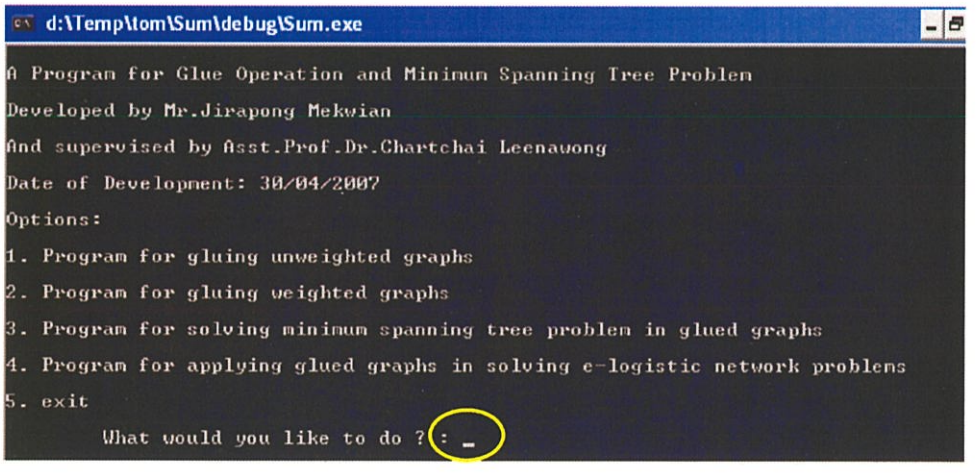
ขั้นที่ 7 โปรแกรมทำการปะติดโดยใช้โครงข่ายต้นไม้แผ่ทั่วถึงของโครงข่าย A เป็นหลัก

ขั้นที่ 8 โปรแกรมคำนวณหาต้นไม้แผ่ทั่วที่น้อยที่สุด

ขั้นที่ 9 โปรแกรมรวมค่าบนต้นไม้แผ่ทั่วที่น้อยที่สุด

ซึ่งนำเสนอตัวอย่างการใช้งานของโปรแกรมดังนี้

เริ่มต้นโปรแกรมจะประกาศว่าโปรแกรมที่ 1, 2, 3 และ 4 คือโปรแกรมแบบใด จากนั้นให้ผู้ใช้งานเลือกโปรแกรมตามความต้องการ โดยจะให้ผู้ใช้ใส่หมายเลขของโปรแกรมที่ต้องการลงไปดังในภาพที่วงไว้



รูปที่ 4.1 ภาพแสดงการใส่ค่าในการเลือกโปรแกรม

หลังจากผู้ใช้งานเลือกโปรแกรมเรียบร้อยแล้ว โปรแกรมก็จะขึ้นมาตามที่ผู้ใช้งานเลือกโดยมีลักษณะของแต่ละโปรแกรมดังนี้

4.1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน โดยมีลักษณะดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย B อีกครั้งเพื่อเป็นการตรวจสอบ

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ดังนี้

จำนวนเส้นเชื่อมของโคลน = ...

เส้นที่ 1 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

เส้นที่ 2 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

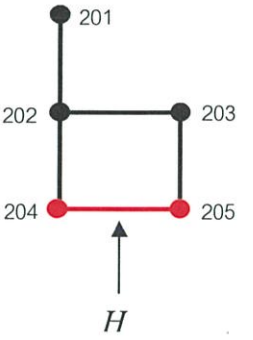
เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น

ตัวอย่างการใช้โปรแกรม

ให้โครงข่าย A คือ

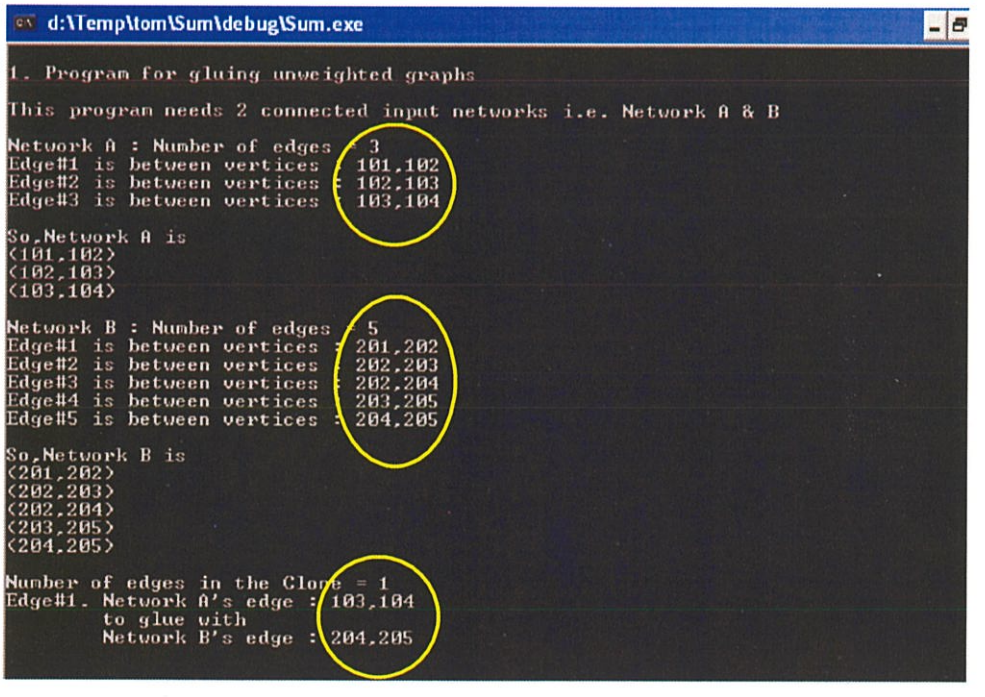
รูปที่ 4.2 โครงข่าย A

ให้โครงข่าย B คือ



รูปที่ 4.3 โครงข่าย B

จะต้องป้อนค่าต่างๆ ลงในโปรแกรม ดังที่วงไว้ในภาพต่อไปนี้



รูปที่ 4.4 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.1

ความหมายของรูปที่ 4.4

โปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน โดยมีรายละเอียด ดังนี้

โครงข่าย *A*

จำนวนเส้นเชื่อมทั้งหมด = 3 (จำนวนเส้นเชื่อมในโครงข่าย *A* มี 3 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 103

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 103 , 104

ดังนั้น โครงข่าย *A* คือ

(101,102)

(102,103)

(103,104)

โครงข่าย *B*

จำนวนเส้นเชื่อมทั้งหมด = 5 (จำนวนเส้นเชื่อมในโครงข่าย *B* มี 5 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 201 , 202

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 203

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 204

เส้นที่ 4 คู่จุดที่มีเส้นเชื่อมกันคือ 203 , 205

เส้นที่ 5 คู่จุดที่มีเส้นเชื่อมกันคือ 204 , 205

ดังนั้น โครงข่าย *B* คือ

(201,202)

(202,203)

(202,204)

(203,205)

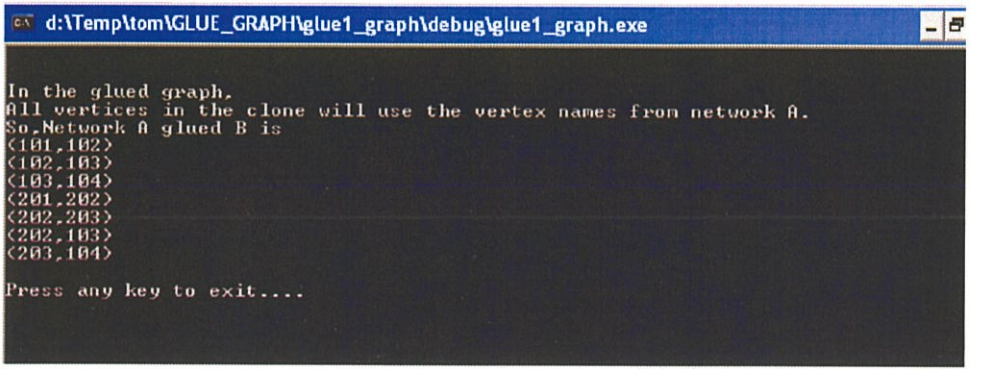
(204,205)

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ดังนี้

จำนวนเส้นเชื่อมในส่วนของโคลน = 1

เส้นที่ 1 โครงข่าย *A* เส้น 103 , 104 ปะติดกับโครงข่าย *B* เส้น 204 , 205

ต่อไปโปรแกรมจะทำการประมวลผลหลังจากใส่โคลนเรียบร้อยแล้ว ดังภาพ



รูปที่ 4.5 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.1

ความหมายของภาพที่ 4.5 และสามารถนำมาวาดเป็นภาพได้ดังนี้

โปรแกรมจะประกาศว่าในกราฟปะติด ทุกๆ จุดในโคลนจะใช้ชื่อจุดของโครงข่าย A เป็นหลัก โครงข่าย A ปะติดกับโครงข่าย B คือ

(101,102)

(102,103)

(103,104)

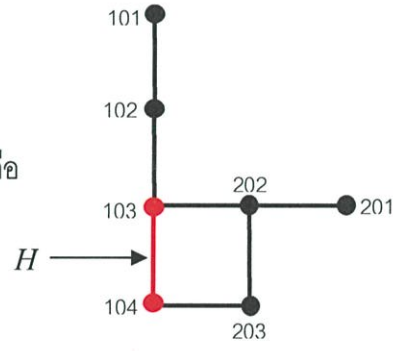
(201,202)

(202,203)

(202,103)

(203,104)

โครงข่าย $A \diamond B$ คือ



รูปที่ 4.6 โครงข่าย $A \diamond B$

4.2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก

โปรแกรมนี้จะมีลักษณะแตกต่างจากโปรแกรมที่ 1 ซึ่งโปรแกรมนี้จะรับน้ำหนักของแต่ละเส้นบนโครงข่ายเพิ่มเข้ามา โดยจะใส่น้ำหนักของเส้นไว้ด้านหลังคู่จุดแต่ละคู่ และส่วนโคลนนั้นเส้นที่จะทำการปะติดกันจะต้องมีน้ำหนักเท่ากัน

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน โดยมีลักษณะดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย B อีกครั้งเพื่อเป็นการตรวจสอบ

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วย ดังนี้

จำนวนเส้นเชื่อมของโคลน = ...

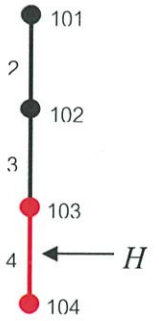
เส้นที่ 1 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

เส้นที่ 2 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น

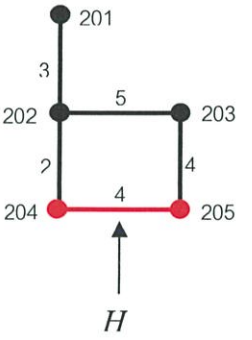
ตัวอย่างการใช้โปรแกรม

ให้โครงข่าย A คือ



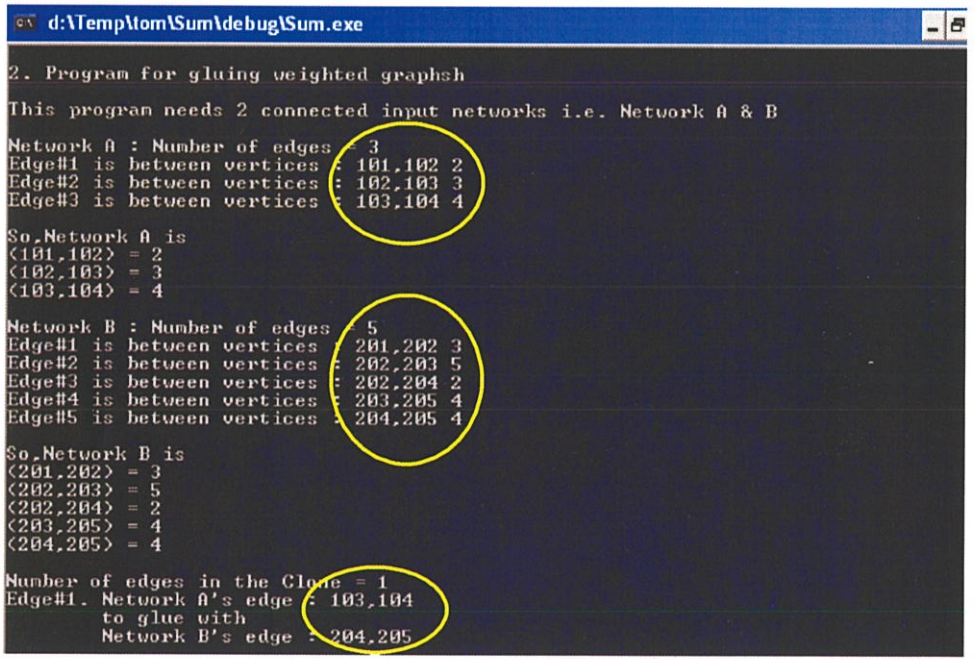
รูปที่ 4.7 โครงข่าย A แบบมีน้ำหนัก

ให้โครงข่าย B คือ



รูปที่ 4.8 โครงข่าย B แบบมีน้ำหนัก

จะต้องป้อนค่าต่างๆ ลงในโปรแกรม ดังที่วงไว้ในภาพต่อไปนี้



รูปที่ 4.9 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.2

ความหมายของรูปที่ 4.9

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน โดยมีรายละเอียด ดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = 3 (จำนวนเส้นเชื่อมในโครงข่าย A มี 3 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102 2

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 103 3

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 103 , 104 4

ดังนั้น โครงข่าย A คือ

(101,102) 2

(102,103) 3

(103,104) 4

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = 5 (จำนวนเส้นเชื่อมในโครงข่าย B มี 5 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 201 , 202 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 203 5

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 204 2

เส้นที่ 4 คู่จุดที่มีเส้นเชื่อมกันคือ 203 , 205 4

เส้นที่ 5 คู่จุดที่มีเส้นเชื่อมกันคือ 204 , 205 4

ดังนั้น โครงข่าย B คือ

(201,202) 3

(202,203) 5

(202,204) 2

(203,205) 4

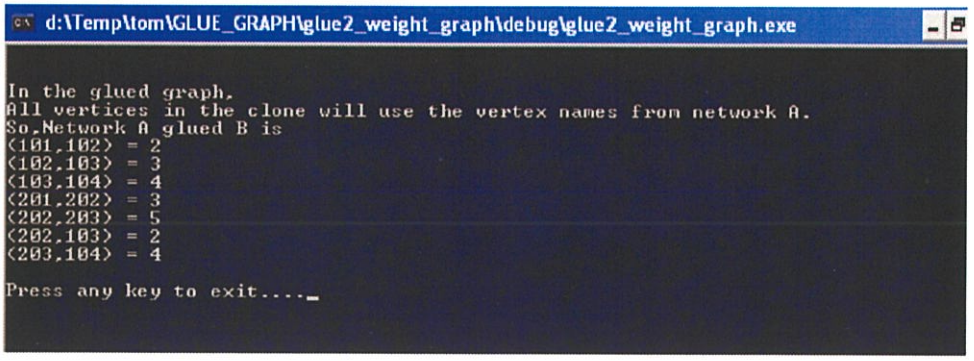
(204,205) 4

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วย ดังนี้

จำนวนเส้นเชื่อมในส่วนของโคลน = 1

เส้นที่ 1 โครงข่าย A เส้น 103 , 104 ปะติดกับโครงข่าย B เส้น 204 , 205

ต่อไปโปรแกรมจะทำการประมวลผลหลังจากใส่โคลนเรียบร้อยแล้ว ดังภาพ



รูปที่ 4.10 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.2

ความหมายของภาพที่ 4.10 และสามารถนำมาวาดเป็นภาพได้ดังนี้

โปรแกรมจะประกาศว่า ในกราฟปะติด ทุกๆ จุดในโคลนจะใช้ชื่อจุดของโครงข่าย A เป็นหลัก

โครงข่าย A ปะติดกับโครงข่าย B คือ

(101,102) 2

(102,103) 3

(103,104) 4

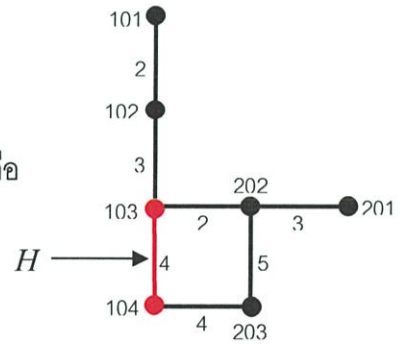
(201,202) 3

(202,203) 5

(202,103) 2

(203,104) 4

โครงข่าย $A \diamond B$ คือ



รูปที่ 4.11 โครงข่าย $A \diamond B$ แบบมีน้ำหนัก

4.3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด

โปรแกรมนี้จะต่างจากโปรแกรมที่ 2 โดยสามารถหาคำตอบที่ดีที่สุดของแต่ละโครงข่าย รวมถึงโครงข่ายที่ปะติดกันแล้วด้วย ซึ่งโปรแกรมนี้ออกแบบมาให้สอดคล้องกับทฤษฎีบทที่ 1 โปรแกรมที่เขียนขึ้นมีลักษณะดังนี้

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ จากนั้นโปรแกรมจะทำการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย A พร้อมทั้งผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน ดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ

โปรแกรมจะหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย A

พร้อมทั้งหาผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย B อีกครั้งเพื่อเป็นการตรวจสอบ

โปรแกรมจะหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย B

พร้อมทั้งหาผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วย ดังนี้

จำนวนเส้นเชื่อมของโคลน = ...

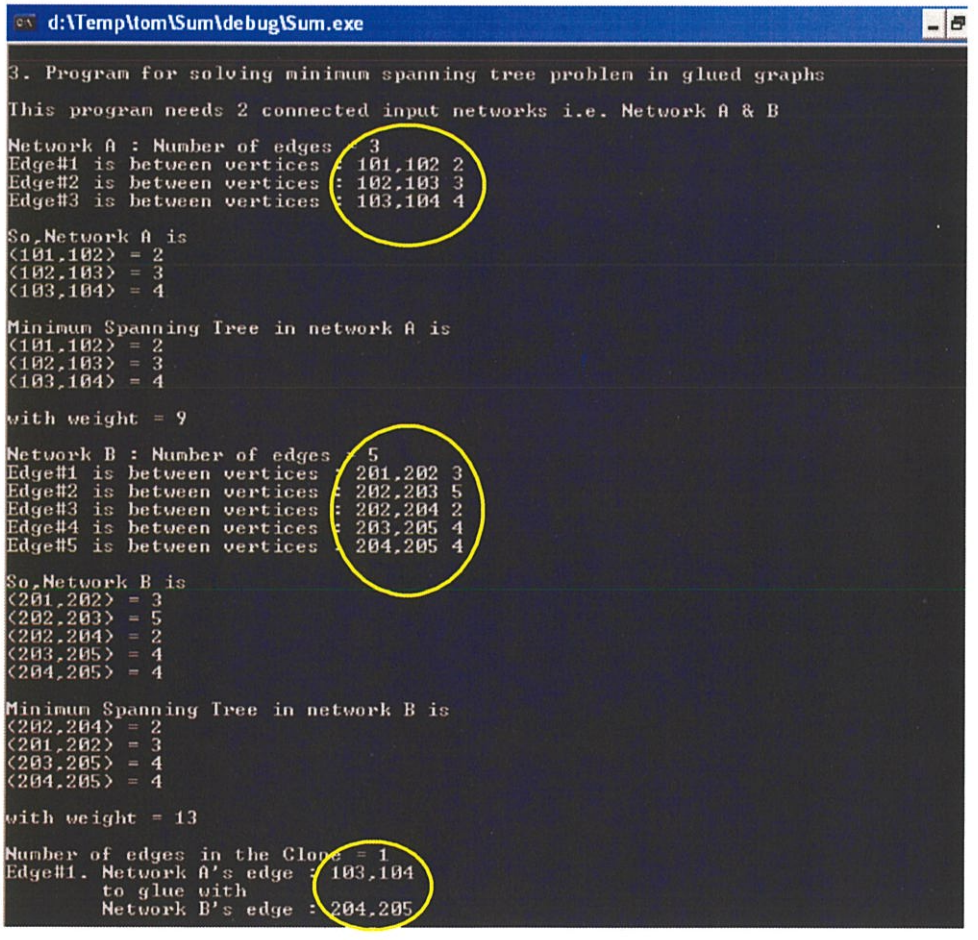
เส้นที่ 1 โครงข่าย *A* เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย *B* เส้น _ _ _ , _ _ _

เส้นที่ 2 โครงข่าย *A* เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย *B* เส้น _ _ _ , _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น

ตัวอย่างการใช้โปรแกรม (ใช้รูปที่ 4.7 และ 4.8)

จะต้องป้อนค่าต่างๆ ลงในโปรแกรม ดังที่วงไว้ในภาพต่อไปนี้



รูปที่ 4.12 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.3

ความหมายของรูปที่ 4.12

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ จากนั้นโปรแกรมจะทำการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย A พร้อมทั้งผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย เมื่อเสร็จแล้วจะรับค่าของโครงข่าย B ต่อไปในลักษณะเดียวกัน โดยมีรายละเอียด ดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = 3 (จำนวนเส้นเชื่อมในโครงข่าย A มี 3 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102 2

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 103 3

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 103 , 104 4

ดังนั้น โครงข่าย A คือ

(101,102) 2

(102,103) 3

(103,104) 4

ต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย A

(101,102) 2

(102,103) 3

(103,104) 4

โดยน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดคือ 9

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = 5 (จำนวนเส้นเชื่อมในโครงข่าย B มี 5 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 201 , 202 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 203 5

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 204 2

เส้นที่ 4 คู่จุดที่มีเส้นเชื่อมกันคือ 203 , 205 4

เส้นที่ 5 คู่จุดที่มีเส้นเชื่อมกันคือ 204 , 205 4

ดังนั้น โครงข่าย B คือ

(201,202) 3

(202,203) 5

(202,204) 2

(203,205) 4

(204,205) 4

ต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย B

(202,204) 2

(201,202) 3

(203,205) 4

(204,205) 4

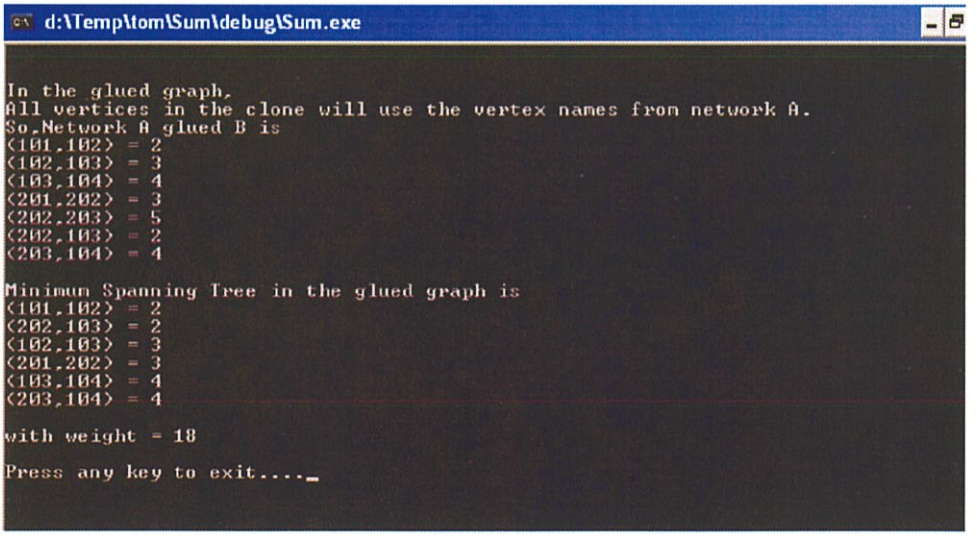
โดยน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดคือ 13

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วย ดังนี้

จำนวนเส้นเชื่อมในส่วนของโคลน = 1

เส้นที่ 1 โครงข่าย A เส้น 103 , 104 ปะติดกับโครงข่าย B เส้น 204 , 205

ต่อไปโปรแกรมจะทำการประมวลผลหลังจากใส่โคลนเรียบร้อยแล้ว ดังภาพ



รูปที่ 4.13 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.3

ความหมายของภาพที่ 4.13 และสามารถนำมาวาดเป็นภาพได้ดังรูปที่ 4.11

โปรแกรมจะประกาศว่า ในกราฟปะติด ทุกๆ จุดในโคลนจะใช้ชื่อจุดของโครงข่าย A เป็นหลัก

โครงข่าย A ปะติดกับโครงข่าย B คือ

(101,102) 2

(102,103) 3

(103,104) 4

(201,202) 3

(202,203) 5

(202,103) 2

(203,104) 4

ต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $A \diamond B$

(101,102) 2

(202,103) 2

(102,103) 3

(201,202) 3

(103,104) 4

(203,104) 4

โดยน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดคือ 18

4.4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

ลักษณะของโปรแกรมออกแบบขึ้นมาเพื่อให้ใช้ประโยชน์ในระบบโครงข่ายที่มีอยู่จริง โดย

◆ โครงข่าย *A* นั้น จะให้ใส่ค่าโครงข่ายที่มีการสร้างไว้แล้วโดยไม่จำเป็นต้องเป็นโครงข่ายที่ดีที่สุดก็ได้

◆ โครงข่าย *B* จะใส่ค่า โครงข่ายที่ได้ทำการศึกษาไว้แล้วแต่ยังไม่ได้ทำการสร้างจริง โดยโปรแกรมจะทำการหาคำตอบที่ดีที่สุดให้

◆ โคลน ส่วนปะติดจะให้ผู้ใช้โปรแกรมพิจารณาดูเองว่าจะนำเส้นทางใดบ้างมาทำการปะติดกัน โดยผู้ใช้โปรแกรมสามารถลองปะติดดูหลายๆ เส้นทางที่เป็นไปได้ว่าการปะติดแบบใดจะเกิดประโยชน์มากที่สุด โปรแกรมนี้จะต่างจากโปรแกรมที่ 4.3 ตรงที่ว่าจะมีโครงข่าย *A* ที่ผู้ใช้โปรแกรมสามารถเลือกต้นไม้แผ่ทั่วใดก็ได้ ซึ่งเปรียบได้กับการมีโครงข่ายที่ได้ถูกสร้างขึ้นไว้แล้ว โปรแกรมนี้ถูกสร้างขึ้นมาเพื่อให้เหมาะกับผู้ที่จะนำกราฟปะติดไปใช้ในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

โปรแกรมมีลักษณะดังนี้

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย *A* และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย *A* โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย *A* อีกครั้งเพื่อเป็นการตรวจสอบ ลำดับต่อไปโปรแกรมจะรับค่าต้นไม้แผ่ทั่วของโครงข่าย *A* แต่ถ้าผู้ใช้โปรแกรมใส่ค่าของโครงข่ายแล้วไม่เป็นต้นไม้แผ่ทั่ว โปรแกรมจะให้ผู้ใช้โปรแกรมใส่ค่าอีกครั้ง และจะแสดงค่าของโครงข่าย *A* ที่เป็นต้นไม้แผ่ทั่วอีกครั้งเพื่อเป็นการตรวจสอบ เมื่อเสร็จแล้วจะรับค่าของโครงข่าย *B* โดยการรับค่าของโครงข่าย *B* นั้นจะเหมือนกับการรับค่าของโปรแกรมที่ 4.3 ดังนี้

โครงข่าย *A*

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย *A* อีกครั้งเพื่อเป็นการตรวจสอบ

โปรแกรมจะให้ใส่ค่าต้นไม้แผ่ทั่วของโครงข่าย *A*

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 1 _ _ , 1 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเป็นต้นไม้แผ่ทั่ว และจะแสดงค่าของโครงข่าย A ที่เป็นต้นไม้แผ่ทั่วอีกครั้งเพื่อเป็นการตรวจสอบ (ถ้าผู้ใช้โปรแกรมใส่ค่าแล้วไม่เป็นต้นไม้แผ่ทั่วของโครงข่าย A โปรแกรมจะรับค่าใหม่)

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = ...

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ และหลังคู่เส้นจะรับน้ำหนักของเส้นด้วย

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 2 _ _ , 2 _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น และจะแสดงค่าของโครงข่าย B อีกครั้งเพื่อเป็นการตรวจสอบ

โปรแกรมจะหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย B พร้อมทั้งหาผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วย ดังนี้

จำนวนเส้นเชื่อมของโคลน = ...

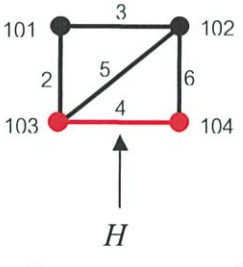
เส้นที่ 1 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

เส้นที่ 2 โครงข่าย A เส้น _ _ _ , _ _ _ ปะติดกับโครงข่าย B เส้น _ _ _ , _ _ _

เป็นเช่นนี้เรื่อยไปจนกว่าจะเท่ากับจำนวนเส้น

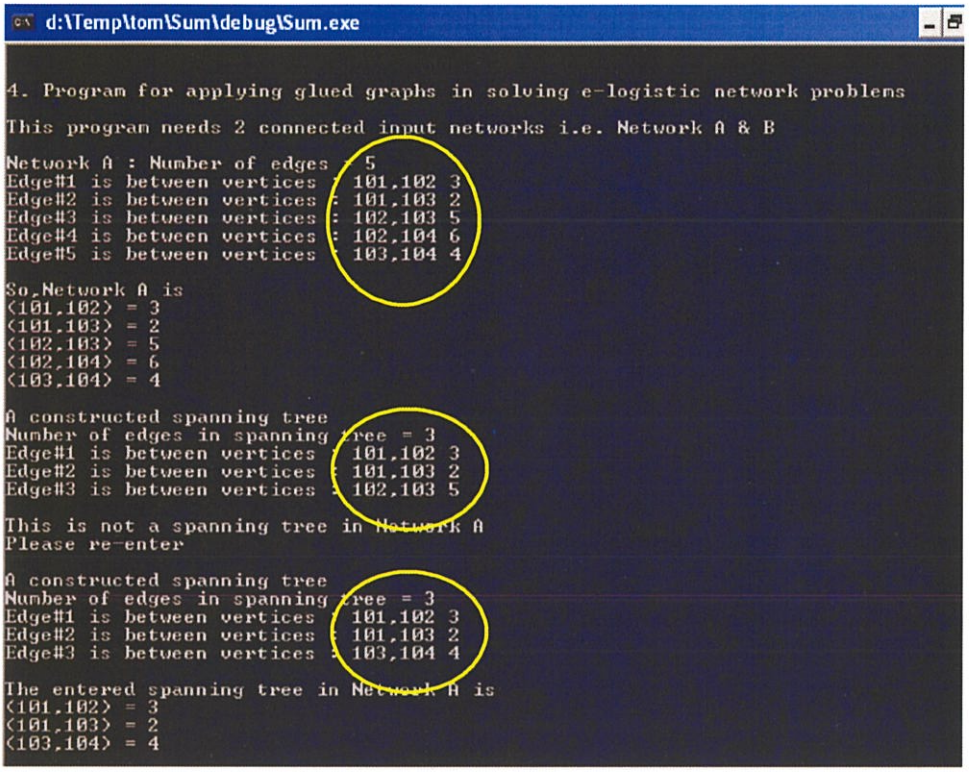
ตัวอย่างการใช้โปรแกรม (ใช้รูปที่ 4.14 และ 4.8 สำหรับใส่ค่าโครงข่าย A และ B ตามลำดับ)

ให้โครงข่าย C คือ



รูปที่ 4.14 โครงข่าย C แบบมีน้ำหนัก

จะต้องป้อนค่าต่างๆ ลงในโปรแกรมในส่วนของโครงข่าย A ดังที่วงไว้ในภาพต่อไปนี้



รูปที่ 4.15 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4 ของโครงข่าย A

ความหมายของรูปที่ 4.15

เริ่มต้นโปรแกรมจะประกาศว่าโครงข่ายที่ใช้ทั้ง 2 โครงข่ายจะต้องเป็นโครงข่ายที่เชื่อมโยง จากนั้นโปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย A และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย A โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย A อีกครั้งเพื่อเป็นการตรวจสอบ ลำดับต่อไปโปรแกรมจะรับค่าต้นไม้แผ่ทั่วของโครงข่าย A แต่ถ้าผู้ใช้โปรแกรมใส่ค่าของโครงข่ายแล้วไม่เป็นต้นไม้แผ่ทั่ว โปรแกรมจะให้ผู้ใช้โปรแกรมใส่ค่าอีกครั้ง และจะแสดงค่าของโครงข่าย A ที่เป็นต้นไม้แผ่ทั่วอีกครั้งเพื่อเป็นการตรวจสอบ โดยมีรายละเอียดดังนี้

โครงข่าย A

จำนวนเส้นเชื่อมทั้งหมด = 5 (จำนวนเส้นเชื่อมในโครงข่าย A มี 5 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 103 2

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 103 5

เส้นที่ 4 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 104 6

เส้นที่ 5 คู่จุดที่มีเส้นเชื่อมกันคือ 103 , 104 4

ดังนั้น โครงข่าย A คือ

(101,102) 3

(101,103) 2

(102,103) 5

(102,104) 6

(103,104) 4

โครงสร้างต้นไม้แผ่ทั่วของโครงข่าย A

จำนวนเส้นเชื่อมทั้งหมดในต้นไม้แผ่ทั่ว = 3

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 103 2

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 102 , 103 5

โปรแกรมประกาศว่าเส้นที่ใส่มายังไม่ใช่ต้นไม้แผ่ทั่วในโครงข่าย A

โปรดใส่ค่าใหม่อีกครั้ง

โครงสร้างต้นไม้แผ่ทั่วของโครงข่าย A

จำนวนเส้นเชื่อมทั้งหมดในต้นไม้แผ่ทั่ว = 3

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 102 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 101 , 103 2

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 103 , 104 4

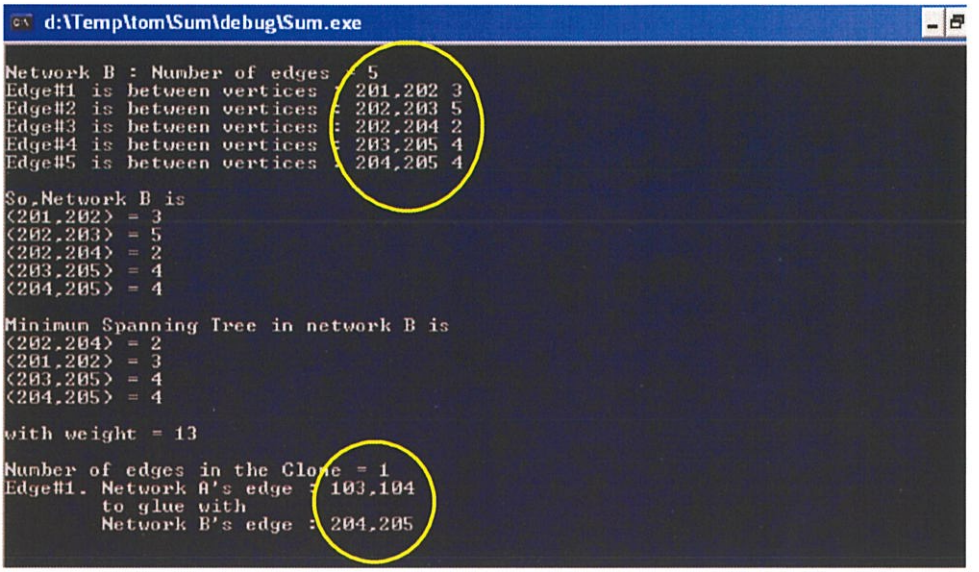
ดังนั้น ต้นไม้แผ่ทั่วในโครงข่าย A คือ

(101,102) 3

(102,103) 2

(103,104) 4

จะต้องป้อนค่าต่างๆ ลงในโปรแกรมในส่วนของโครงข่าย B และส่วนโคลนดังที่วงไว้ในภาพต่อไปนี้



รูปที่ 4.16 ภาพแสดงการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4 ของโครงข่าย B และส่วนโคลน

ความหมายของรูปที่ 4.16

โปรแกรมจะรับค่าจำนวนเส้นเชื่อมของโครงข่าย B และลำดับต่อไปจะรับค่าจุดปลายของแต่ละเส้นเชื่อมของโครงข่าย B โดยหลังจากที่รับค่าจุดปลายแล้วจะให้ใส่ค่าน้ำหนักของเส้นนั้นๆ ด้วย และจะแสดงค่าของโครงข่าย B อีกครั้งเพื่อเป็นการตรวจสอบ จากนั้นโปรแกรมจะทำการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย B พร้อมทั้งผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วย

โครงข่าย B

จำนวนเส้นเชื่อมทั้งหมด = 5 (จำนวนเส้นเชื่อมในโครงข่าย B มี 5 เส้น)

เส้นที่ 1 คู่จุดที่มีเส้นเชื่อมกันคือ 201 , 202 3

เส้นที่ 2 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 203 5

เส้นที่ 3 คู่จุดที่มีเส้นเชื่อมกันคือ 202 , 204 2

เส้นที่ 4 คู่จุดที่มีเส้นเชื่อมกันคือ 203 , 205 4

เส้นที่ 5 คู่จุดที่มีเส้นเชื่อมกันคือ 204 , 205 4

ดังนั้น โครงข่าย B คือ

(201,202) 3

(202,203) 5

(202,204) 2

(203,205) 4

(204,205) 4

ต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย *B*

(202,204) 2

(201,202) 3

(203,205) 4

(204,205) 4

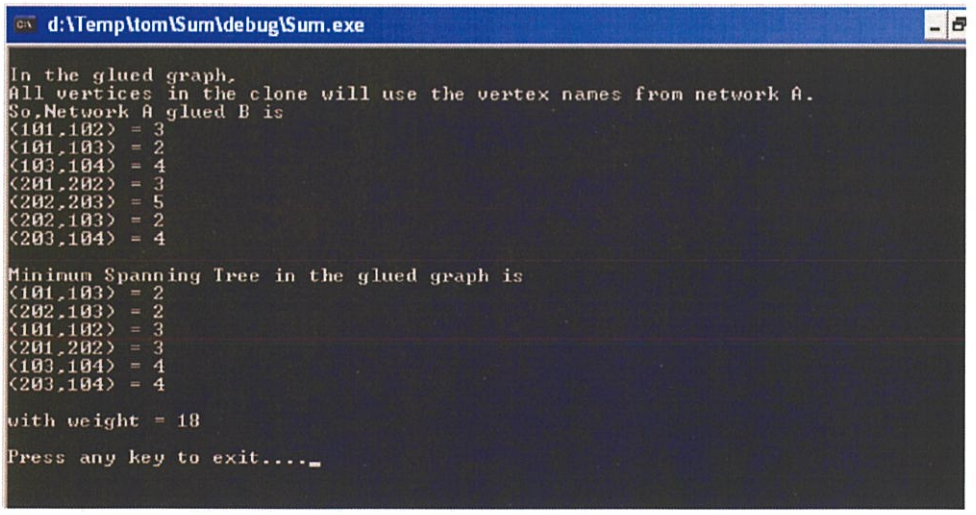
โดยน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดคือ 13

ลำดับต่อไปโปรแกรมจะรับส่วนของโครงข่ายที่จะนำมาปะติดกัน (โคลน) โดยจะรับค่าจำนวนของเส้นที่จะนำมาปะติดกันก่อนว่ามีกี่เส้น จากนั้นโปรแกรมจะรับค่าว่าเส้นของโครงข่าย A จะปะติดกับโครงข่าย B เส้นใด โดยเส้นที่จะนำมาปะติดของโครงข่าย A นั้นให้นำเส้นมาจากต้นไม้แผ่ทั่วที่ใส่ค่าเอาไว้ก่อนหน้านี้ ซึ่งเส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักเท่ากันด้วยดังนี้

จำนวนเส้นเชื่อมในส่วนของโคลน = 1

เส้นที่ 1 โครงข่าย *A* เส้น 103 , 104 ปะติดกับโครงข่าย *B* เส้น 204 , 205

ต่อไปโปรแกรมจะทำการประมวลผลหลังจากใส่โคลนเรียบร้อยแล้ว โปรแกรมจะประกาศว่า ในกราฟปะติด ทุกๆ จุดในโคลนจะใช้ชื่อจุดของโครงข่าย *A* เป็นหลัก จากนั้นโปรแกรมก็จะแสดงผลของโครงข่ายออกมาตามที่ผู้ใช้โปรแกรมทำการปะติดโครงข่าย พร้อมทั้งหาคำตอบที่ดีที่สุดของโครงข่ายและผลรวมของโครงข่ายที่ทำการปะติดกันแล้วให้ใหม่ ดังภาพ



รูปที่ 4.17 ภาพแสดงผลที่ได้จากการใส่ค่าข้อมูลเข้าของโปรแกรมที่ 4.4

ความหมายของภาพที่ 4.17 และสามารถนำมาวาดเป็นภาพได้ดังรูปที่ 4.18

โครงข่าย A ปะติดกับโครงข่าย B คือ

(101,102) 3

(101,103) 2

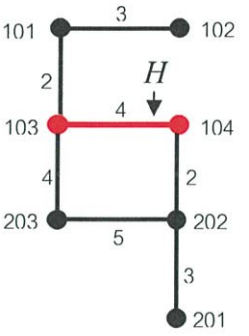
(103,104) 4

(201,202) 3

(202,203) 5

(202,103) 2

(203,104) 4



รูปที่ 4.18 โครงข่าย $C \diamond B$ แบบมีน้ำหนัก

ต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่าย $A \diamond B$

(101,102) 2

(202,103) 2

(101,102) 3

(201,202) 3

(103,104) 4

(203,104) 4

โดยน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดคือ 18

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลงานวิจัย

ในบทนี้นำเสนอการสรุปผลการวิจัย ที่ได้นำกราฟปะติดไปประยุกต์ใช้กับปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุด (Minimum Spanning Tree) และยังจะใช้กราฟปะติดไปประยุกต์ใช้กับปัญหาอี-โลจิสติกส์ รวมถึงการออกแบบโปรแกรมคอมพิวเตอร์เพื่อปะติดโครงข่าย 2 โครงข่าย โดยจะแบ่งการสรุปออกเป็น 4 หัวข้อ ดังนี้

1. **การปะติดกราฟแบบต่างๆ**

การปะติดกราฟ 2 กราฟ โดยที่โคลนไม่เปลี่ยนแปลง ซึ่งจะทำให้กราฟที่ปะติดอาจจะมีได้มากกว่า 1 แบบ การที่ได้มากกว่า 1 แบบนั้น ก็หมายความว่ากราฟที่ปะติดกันแล้วนั้น แต่ละกราฟจะไม่สมสัณฐานกัน (isomorphism) แต่กราฟที่ปะติดกันแล้วนั้นจะมีกี่แบบนั้นขึ้นอยู่กับลักษณะของแต่ละกราฟเริ่มต้น (original graph) รวมถึงลักษณะของโคลนด้วย

2. **การปะติดกราฟต้นไม้แผ่ทั่วที่น้อยที่สุด**

การปะติดโครงข่าย 2 โครงข่าย โดยที่แต่ละโครงข่ายสามารถหาต้นไม้แผ่ทั่วที่น้อยที่สุดได้ ซึ่งจะทำให้การปะติดโครงข่าย 2 โครงข่ายนั้นเกิดการปะติดได้ 2 กรณี

กรณีที่ 1 ทุกคู่เส้นของโคลนตรงส่วนที่ปะติดกัน จะมีอย่างน้อย 1 เส้นมาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

จากกรณีที่ 1 การปะติดโครงข่าย 2 โครงข่ายเข้าด้วยกันนั้น โดยการปะติดกันของ 2 โครงข่ายนั้น ไม่ว่าโคลนจะมีกี่เส้นก็ตาม แต่ละคู่เส้นของโคลนจะต้องมีเส้นจะต้องมีเส้นที่มาจากคำตอบที่ดีที่สุดของโครงข่ายใดโครงข่ายหนึ่ง หรืออาจจะเป็นเส้นที่มาจากคำตอบที่ดีที่สุดของทั้ง 2 โครงข่ายเลยก็ได้ จากการปะติดแบบกรณีที่ 1 นี้ ก็จะทำให้ได้ว่าน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่ายที่ปะติดกันแล้ว จะเท่ากับ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของทั้ง 2 โครงข่ายรวมกันแล้วลบออกด้วยน้ำหนักตรงส่วนที่โคลน จึงทำให้ได้ทฤษฎีบทหนึ่งขึ้นมาและได้ทำการพิสูจน์เพื่อยืนยันว่าทฤษฎีบทนี้เป็นจริง โดยมีทฤษฎีว่า กำหนดให้ G_1 และ G_2 เป็นกราฟเชื่อมโยงที่มีน้ำหนัก และ H ไม่เป็นวงของ $G_1 \underset{H}{\diamond} G_2$ โดย T_1 และ T_2 เป็นต้นไม้แผ่ทั่วที่น้อยที่สุด (MST) ของ G_1 และ G_2 ตามลำดับ ถ้าทุกๆ $e \in H, e \in T_1$ หรือ $e \in T_2$ แล้ว $w_{T^*} = w_{T_1} + w_{T_2} - w_H$

โดยมี T^* เป็น MST $G_1 \underset{H}{\diamond} G_2$

กรณีที่ 2 มีบางคู่เส้นของโคลนตรงส่วนที่ปะติดกัน ทั้ง 2 เส้นไม่ได้มาจากคำตอบที่ดีที่สุดจากกราฟที่สัมพันธ์กัน

จากกรณีที่ 2 นี้ การปะติดโครงข่าย 2 โครงข่ายเข้าด้วยกันนั้น โดยการปะติดกันของ 2 โครงข่ายนั้น ไม่ว่าโคลนจะมีกี่เส้นก็ตาม ถ้ามีโคลนอยู่คู่เส้นหนึ่งที่ทั้ง 2 เส้นไม่ได้มาจากคำตอบที่ดีที่สุดแล้วคู่เส้นอื่นๆ จะมีเส้นที่มาจากคำตอบที่ดีที่สุดก็ตาม ก็จะจัดว่าอยู่ในกรณีที่ 2 ทั้งนี้ ผลของน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่ายที่ปะติดกันแล้ว ก็จะไม่ได้ผลเหมือนกับกรณีที่ 1 นั่นคือ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่ายที่ปะติดกันแล้ว จะไม่เท่ากับ น้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุดของทั้ง 2 โครงข่ายรวมกัน แล้วลบออกด้วยน้ำหนักตรงส่วนที่โคลน

3. **การใช้กราฟปะติดไปประยุกต์ใช้กับปัญหาอี-โลจิสติกส์**

การนำกราฟปะติดไปประยุกต์ใช้กับปัญหาอี-โลจิสติกส์ เปรียบเสมือนกับการที่มีโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ เช่น โครงข่ายสายโทรศัพท์ ไฟฟ้า น้ำประปา หรือระบบอินทราเน็ต โดยโครงข่ายเหล่านี้จะช่วยส่งเสริมกิจกรรมทางด้านอี-โลจิสติกส์ การนำกราฟปะติดไปประยุกต์ใช้ก็คือ การที่นำโครงข่าย 2 โครงข่าย มาทำการปะติดกันโดย โครงข่ายแรกอาจจะมีการสร้างไว้แล้วและมีโครงข่ายที่ 2 ที่ได้ทำการศึกษาไว้แล้วแต่ยังไม่ได้ทำการสร้างจริง โดยบริษัทผู้ทำการก่อสร้างจะพิจารณาดูว่ามีเส้นทางใดบ้างที่สามารถทำการปะติดกันได้ โดยบริษัทผู้ทำการก่อสร้างมีจุดประสงค์ที่จะขยายกิจการให้ใหญ่ขึ้นและจะทำการเชื่อมต่อโครงข่ายเพื่อที่จะลดต้นทุนในการก่อสร้างโครงข่ายสาธารณูปโภคพื้นฐานต่างๆ ลง

4. **โปรแกรมคอมพิวเตอร์เพื่อทำการปะติดโครงข่าย**

โปรแกรมคอมพิวเตอร์ที่ทำการสร้างขึ้นนี้จะใส่ค่าและแสดงผลของแต่ละโครงข่ายในลักษณะของ Edge List โดยการใส่ค่าของจุดนั้นให้ใส่ในลักษณะของเลข 4 ตำแหน่ง คือ

---'--- -

โปรแกรมจะรับค่าในแต่ละเส้นทั้งหมด 7 ตำแหน่ง โดย 1 จุดจะรับค่า 3 ตำแหน่ง ซึ่ง 6 ตำแหน่งแรกจะเป็นการรับจุดปลาย (end node) ของแต่ละเส้น ตำแหน่งแรกของแต่ละจุดคือชื่อโครงข่าย (โครงข่าย *A* ใช้หมายเลข 1 โครงข่าย *B* ใช้หมายเลข 2) ตำแหน่งที่ 2 และตำแหน่งที่ 3 ของแต่ละจุดคือหมายเลขจุด (ใส่ค่าได้ 100 จุด) ตำแหน่งที่ 7 คือน้ำหนักบนเส้นเชื่อม (กรณีที่เลือกโปรแกรมแบบไม่มีน้ำหนักโปรแกรมจะไม่รับค่าตรงตำแหน่งที่ 7) และแต่ละโครงข่ายที่จะนำมาใส่ในโปรแกรมจะต้องเป็นโครงข่ายที่เชื่อมโยง โดยโปรแกรมได้ถูกออกแบบมาเพื่อให้ผู้ใช้งานเลือกใช้งานได้ตามความต้องการ ซึ่งจะมีลักษณะของโปรแกรมอยู่ 4 แบบ ดังนี้

- *โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก*

โครงข่ายที่จะนำมาใส่ค่าในโปรแกรมจะต้องเป็นโครงข่ายที่ไม่มีน้ำหนัก โดยโปรแกรมจะรับค่าจำนวนเส้นของโครงข่ายที่ 1 ก่อน จากนั้นโปรแกรมจะขึ้นมาว่าเส้นที่ 1 คือคู่จุดอะไร อย่างนี้ไปเรื่อยๆ จนกว่าจะครบจำนวนเส้นของโครงข่ายที่ 1 จากนั้นก็จะรับจำนวนเส้นของโครงข่ายที่ 2 และโปรแกรมจะรับค่าเหมือนโครงข่ายแรก ลำดับต่อไปจะรับค่าจำนวนเส้นที่โคลนว่ามีกี่เส้น และ

จะให้ใสค่าไปว่า คู่จุดใดของโครงข่ายที่ 1 ปะติดกับคู่จุดใดของโครงข่ายที่ 2 เป็นต้น ตรงส่วนโคลนนี้จะให้ผู้ใช้โปรแกรมเลือกที่จะปะติดโครงข่ายตามความต้องการ เมื่อใส่จำนวนเส้นที่จะทำการปะติดครบแล้ว โปรแกรมจะแสดงผลออกมาเป็นโครงข่ายที่ทำการปะติดกันแล้ว

- *โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก*

โครงข่ายที่จะนำมาใส่ค่าในโปรแกรมจะต้องเป็นโครงข่ายที่มีน้ำหนัก ลักษณะของโปรแกรมจะเหมือนกับโปรแกรมที่ 1 จะต่างกันอยู่ตรงที่โปรแกรมนี้จะรับค่าน้ำหนักของแต่ละเส้นด้วย โดยโปรแกรมจะให้ใส่น้ำหนักของเส้นหลังจากใส่คู่จุดของเส้นนั้นๆ แล้ว และตรงส่วนโคลนให้ใส่ค่าเหมือนกับโปรแกรมที่ 1 ไม่ต้องใส่น้ำหนักของเส้นลงไปตรงส่วนโคลน (เส้นที่จะนำมาปะติดกันนั้นจะต้องมีน้ำหนักของเส้นเท่ากัน) เมื่อใส่จำนวนเส้นที่จะทำการปะติดครบแล้ว โปรแกรมจะแสดงผลออกมาเป็นโครงข่ายที่ทำการปะติดกันแล้ว พร้อมทั้งน้ำหนักของแต่ละเส้นด้วย

- *โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด*

โครงข่ายที่จะนำมาใส่ค่าในโปรแกรมจะต้องเป็นโครงข่ายที่มีน้ำหนัก จะใส่ค่าเหมือนกับโปรแกรมที่ 2 ในเมื่อโครงข่ายที่นำมาใส่ค่าในโปรแกรมเป็นโครงข่ายที่เชื่อมโยง ดังนั้นก็จะสามารถหาต้นไม้แผ่ทั่วที่น้อยที่สุดได้ ซึ่งโปรแกรมที่ 3 นี้จะทำการหาต้นไม้แผ่ทั่วที่น้อยที่สุดของแต่ละโครงข่าย รวมทั้งบอกผลรวมของต้นไม้แผ่ทั่วที่น้อยที่สุดด้วยว่ามีค่าเท่ากับเท่าใด โดยโปรแกรมจะแสดงโครงข่ายของต้นไม้แผ่ทั่วที่น้อยที่สุดหลังจากใส่ค่าของแต่ละโครงข่ายครบแล้ว รวมถึงโครงข่ายที่ทำการปะติดกันแล้วโปรแกรมก็จะแสดงโครงข่ายของต้นไม้แผ่ทั่วที่น้อยที่สุด

- *โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์*

โปรแกรมนี้ถูกออกแบบมาให้เหมาะกับผู้ใช้งานเป็นอย่างยิ่งโดยมีลักษณะดังนี้ โครงข่ายแรกนั้น จะให้ใส่ค่าโครงข่ายที่มีการสร้างไว้แล้วโดยไม่จำเป็นต้องเป็นโครงข่ายที่ดีที่สุดก็ได้ โครงข่ายที่ 2 จะใส่ค่า โครงข่ายที่ได้ทำการศึกษาไว้แล้วแต่ยังไม่ได้ทำการสร้างจริง โดยโปรแกรมจะทำการหาคำตอบที่ดีที่สุดให้ โดยการใส่ค่านั้นจะใส่ค่าเหมือนกับโปรแกรมที่ 3 ในส่วนของโคลน จะให้ผู้ใช้โปรแกรมพิจารณาดูเองว่าจะนำเส้นทางใดบ้างมาทำการปะติดกัน โดยผู้ใช้โปรแกรมสามารถลองปะติดดูหลายๆ เส้นทางที่เป็นไปได้ว่าการปะติดแบบใดจะเกิดประโยชน์มากที่สุด การแสดงผลโปรแกรมจะหาต้นไม้แผ่ทั่วที่น้อยที่สุดของโครงข่ายที่ทำการปะติดกันแล้วพร้อมทั้งผลรวมน้ำหนักของต้นไม้แผ่ทั่วที่น้อยที่สุด

5.2 ข้อเสนอแนะ

เนื่องจากกราฟปะติดเป็นเรื่องใหม่ และการนำกราฟปะติดมาประยุกต์ใช้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดนั้นก็เป็นเรื่องใหม่ ทั้งนี้ผู้วิจัยได้พบทฤษฎีที่เกี่ยวกับการประยุกต์ครั้งนี้และอาจจะมีทฤษฎีอื่นๆ อีกก็ได้ที่ผู้วิจัยยังไม่ได้คิด ยกตัวอย่างเช่น ต้นไม้แผ่ทั่วที่น้อยที่สุดของแต่ละโครงข่ายอาจจะมีได้มากกว่า 1 แบบ เมื่อปะติดกันแล้วจะมีกี่แบบ เราอาจจะหาขอบเขตของปัญหานี้ได้ หรือกรณีที่โคลนเป็นแบบวง แล้วต้นไม้แผ่ทั่วที่น้อยที่สุดของกราฟปะติดจะได้ผลอย่างไร เป็นต้น และเราสามารถนำกราฟปะติดไปประยุกต์ใช้กับปัญหากราฟได้อีกหลายปัญหา เช่น ปัญหาทางเดินที่สั้นที่สุด (Shortest Path Problem)

เอกสารอ้างอิง

[1] กมลชนก สุทธิวาทนฤพุฒิ, ศลิษา ภมรสถิต และจักรกฤษณ์ ดวงพัสตรา 2003. "การจัดการโซ่อุปทานและโลจิสติกส์ (Supply Chain and Logistics Management)", บริษัท สำนักพิมพ์ท้อป จำกัด, กรุงเทพ

[2] กรรณิกา กวักเพชูรย์ 2542. "หลักคณิตศาสตร์ (Principles of Mathematics)", โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย

[3] ฉัฐไชย์ ลีนาวงศ์ 2547. "ตรรกะ...แห่งการพิสูจน์ (How to Read and Do Proofs)", บริษัท สำนักพิมพ์ท้อป จำกัด

[4] พัฒนี อุดมกะวานิช 2541. "หลักคณิตศาสตร์ (Principles of Mathematics)", ห้างหุ้นส่วนจำกัด พิทักษ์การพิมพ์

[5] วนิดา เหมะกุล 2521. "คณิตศาสตร์ ดิสครีต (Discrete Mathematics)", บริษัท ซีเอ็ดยูเคชั่น จำกัด, กรุงเทพ

[6] C. Promsakon and C. Uiyyasathian. 2006. "Chromatic Number of Glued Graph" *Thai Journal of Mathematics Special Issue*, 75-81

[7] J. Gross and J. Yellen. 1999. "Graph Theory and Its Applications", CRC Press, Boca Raton

[8] M. Chip. 2002. "The Expected Complexity of Prim's Minimum Spanning Tree Algorithm" *Information Processing Letters*,81, 197-201

[9] M. ZHU., G. HU., Q. ZHENG and H. PENG. 2005. "Multiple Sequence Alignment Using Minimum Spanning Tree" *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,Guangzhou*, 3352-3356

[10] R. Haggarty. 2001. "Discrete Mathematics for Computing", Pearson Education Limited

ภาคผนวก

โปรแกรมนี้ได้ทำการเขียนบน Microsoft visual studio 2005 ด้วยภาษา C++ บนระบบปฏิบัติการ Windows XP เขียนขึ้นโดยนายจิรพงษ์ เมฆเวียน

โปรแกรมคอมพิวเตอร์ที่ทำการสร้างขึ้นจะแบ่งออกเป็น 4 แบบ ได้แก่

4.1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก

4.2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก

4.3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด

4.4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

Code โปรแกรมคอมพิวเตอร์

```
#include "stdafx.h"
#include "stdio.h"
#include "conio.h"

int glue1_graph();
int glue2_weight_graph();
int glue3_MST();
int glue4_e_logistic();

int SortEdges(int n);
int SelectEdges(int n);
int CountDot(int n);
int e1[100][3], e2[100][3];

int _tmain(int argc, _TCHAR* argv[])
{
	int program;
	do {
		printf("\n\t\tPlease select a program with you want\n\n");
		printf("\n1. Program for gluing unweighted graphs\n");
		printf("\n2. Program for gluing weighted graphs\n");
		printf("\n3. Program for solving minimum spanning tree problem in
glued graphs\n");
		printf("\n4. Program for applying glued graphs in solving e-logistic
network problem\n");
		printf("\n5. exit\n");
		printf("\n\tSelect Program : "); //รับค่าโปรแกรม
		scanf("%d",&program);

		switch(program){
			case 1 :
			printf("\n\n1. Program for gluing unweighted graphs\n\n");
```

```
                        glue1_graph();
                        break;
                case 2 :
                printf("\n\n2. Program for gluing weighted graphsh\n\n");
                        glue2_weight_graph();
                        break;
                case 3 :
                printf("\n\n3. Program for solving minimum spanning tree
 problem in glued graphs\n\n");
                        glue3_MST();
                        break;
                case 4 :
                printf("\n\n4. Program for applying glued graphs in solving e-
 logistic network problem\n\n");
                        glue4_e_logistic();
                        break;
                case 5 :
                        break;
                default :
                        break;
                }
        }while(program != 5);
        printf("\nPress any key to exit....");
        getch();
        return 0;
}
```

4.1 โปรแกรมเพื่อปะติดโครงข่าย แบบไม่มีน้ำหนัก

```
int glue1_graph()
{
        int A[100][2], B[100][2], C[100][2], D[100][2];
        int E1[100][2];
        int i,j,k,l,n;
        char Chk;
        int a, b, c;

        printf("This program needs 2 connected input networks i.e. Network A &
B\n\n");
//รับค่าจำนวนเส้นของโครงข่าย A
        printf("Network A : Number of edges = ");
                        scanf("%d",&a);
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<a;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d",&A[i][0],&A[i][1]);
        }
```

```
        printf("\nNetwork A\n");
        for(i=0;i<a;i++){
                        printf("(%d,%d)\n",A[i][0],A[i][1]);
                }
//รับค่าจำนวนเส้นของโครงข่าย B
        printf("\nNetwork B : Number of edges = ");
        scanf("%d",&b);
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<b;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d",&B[i][0],&B[i][1]);
        }

        printf("\nNetwork B\n");
        for(i=0;i<b;i++){
                        printf("(%d,%d)\n",B[i][0],B[i][1]);
                }
//รับค่าจำนวนของเส้นที่ปะติดกัน
        printf("\nNumber of edges in the Clone = ");
        scanf("%d",&c);
//รับค่าของแต่ละเส้นที่ปะติดกัน
        for(i=0;i<c;i++){
                printf("Edge#%d.\tNetwork A's edge : ",i+1);
                        scanf("%d,%d",&C[i][0],&C[i][1]);
                        printf("\tto glue with\n");
                        printf("\tNetwork B's edge : ",i+1);
                        scanf("%d,%d",&D[i][0],&D[i][1]);
        }

        for(i=0;i<b;i++){
                for(j=0;j<2;j++){
                        for(k=0;k<c;k++){
                                for(l=0;l<2;l++){
                                        if (B[i][j] == D[k][l]){
                                                B[i][j] = C[k][l];
                                        }
                                }
                        }
                }
        }

        printf("\n");
        printf("\nIn the glued graph, \nAll vertices in the clone will use the vertex
names from network A. ");
        printf("\nSo,Network A glued B is\n");

        n=0;
```

```
        for(i=0;i<a;i++){
                        E1[n][0] = A[i][0];
                        E1[n][1] = A[i][1];
                        E1[n][2] = A[i][2];
                        n++;
        }

        for(i=0;i<b;i++){       // โครงข่าย B โดยตัดเส้นที่ปะติดออกแล้ว และนำค่าจุดที่
ตรงกัน จาก โครงข่าย A แทนที่ โครงข่าย B แล้ว
                Chk='N';
                for(j=0;j<a;j++){ // ตรวจดูว่าเป็นเส้นที่ปะติดกันหรือไม่
                        if((B[i][0]==A[j][0]) && (B[i][1]==A[j][1])){
                                Chk='Y';
                        }
                }
                if (Chk=='N'){ // ถ้าไม่ใช่เส้นที่ปะติดกัน เลือกเส้นจากโครงข่าย B ที่เหลืออยู่
                        E1[n][0] = B[i][0];
                        E1[n][1] = B[i][1];
                        E1[n][2] = B[i][2];
                        n++;
                }
        }

        for(i=0;i<n;i++){
                printf("(%d,%d)\n",E1[i][0],E1[i][1]);
        }

        printf("\nPress any key to exit....");
        getch();

        return 0;
}
```

4.2 โปรแกรมเพื่อปะติดโครงข่าย แบบมีน้ำหนัก

```
int glue2_weight_graph()
{
        int A[100][3], B[100][3], C[100][2], D[100][2];
        int E1[100][3];
        int i,j,k,l,n;
        char Chk;
        int a, b, c;
        printf("This program needs 2 connected input networks i.e. Network A &
B\n\n");
//รับค่าจำนวนเส้นของโครงข่าย A
        printf("Network A : Number of edges = ");
                        scanf("%d",&a);
```

```
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<a;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&A[i][0],&A[i][1],&A[i][2]);
        }

        printf("\nNetwork A\n");
        for(i=0;i<a;i++){
                        printf("(%d,%d) = %d\n",A[i][0],A[i][1],A[i][2]);
                }
//รับค่าจำนวนเส้นของโครงข่าย B
        printf("\nNetwork B : Number of edges = ");
        scanf("%d",&b);
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<b;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&B[i][0],&B[i][1],&B[i][2]);
        }

        printf("\nNetwork B\n");
        for(i=0;i<b;i++){
                        printf("(%d,%d) = %d\n",B[i][0],B[i][1],B[i][2]);
                }
//รับค่าจำนวนของเส้นที่ปะติดกัน
        printf("\nNumber of edges in the Clone = ");
        scanf("%d",&c);

//รับค่าของแต่ละเส้นที่ปะติดกัน
        for(i=0;i<c;i++){
                printf("Edge#%d.\tNetwork A's edge : ",i+1);
                        scanf("%d,%d",&C[i][0],&C[i][1]);
                        printf("\tto glue with\n");
                        printf("\tNetwork B's edge : ",i+1);
                        scanf("%d,%d",&D[i][0],&D[i][1]);
        }

        for(i=0;i<b;i++){
                for(j=0;j<2;j++){
                        for(k=0;k<c;k++){
                                for(l=0;l<2;l++){
                                        if (B[i][j] == D[k][l]){
                                                B[i][j] = C[k][l];
                                        }
                                }
                        }
                }
        }
```

```
        printf("\n");
        printf("\nIn the glued graph, \nAll vertices in the clone will use the vertex
 names from network A. ");
        printf("\nSo,Network A glued B is\n");

        n=0;
        for(i=0;i<a;i++){
                        E1[n][0] = A[i][0];
                        E1[n][1] = A[i][1];
                        E1[n][2] = A[i][2];
                        n++;
        }

        for(i=0;i<b;i++){
                Chk='N';
                for(j=0;j<a;j++){
                        if((B[i][0]==A[j][0]) && (B[i][1]==A[j][1])){
                                Chk='Y';
                        }
                }
                if (Chk=='N'){
                        E1[n][0] = B[i][0];
                        E1[n][1] = B[i][1];
                        E1[n][2] = B[i][2];
                        n++;
                }
        }

        for(i=0;i<n;i++){
                printf("(%d,%d) = %d\n",E1[i][0],E1[i][1],E1[i][2]);
        }

        printf("\nPress any key to exit....");
        getch();

        return 0;
}
```

4.3 โปรแกรมเพื่อแก้ปัญหาต้นไม้แผ่ทั่วที่น้อยที่สุดในกราฟปะติด

```
int glue3_MST()
{
        int A[100][3], B[100][3], C[100][2], D[100][2];
        int SrtA[100][3], SrtB[100][3], SelA[100][3], SelB[100][3];
        int E1[100][3], E2[100][3], E_TMP[3];
        int Sel_Edge[100][3];
        int i,j,k,l,m,n;
        char Chk;
        char f1,f2;
        int a, b, c, y, z;
```

```
        int Sum_MST, Cnt_Dot;

        printf("This program needs 2 connected input networks i.e. Network A &
B\n\n");
//รับค่าจำนวนเส้นของโครงข่าย A
        printf("Network A : Number of edges = ");
                        scanf("%d",&a);
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<a;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&A[i][0],&A[i][1],&A[i][2]);
        }

        printf("\nNetwork A\n");
        for(i=0;i<a;i++){
                        printf("(%d,%d) = %d\n",A[i][0],A[i][1],A[i][2]);
                }

        for(i=0;i<a;i++){
                e1[i][0] = A[i][0];     e1[i][1] = A[i][1];     e1[i][2] = A[i][2];
        }
        SortEdges(a);
        for(i=0;i<a;i++){
                SrtA[i][0] = e2[i][0];  SrtA[i][1] = e2[i][1];  SrtA[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(a);

        for(i=0;i<a;i++){
                e1[i][0] = SrtA[i][0];  e1[i][1] = SrtA[i][1];  e1[i][2] = SrtA[i][2];
        }
        SelectEdges(a);
        for(i=0;i<a;i++){
                SelA[i][0] = e2[i][0];  SelA[i][1] = e2[i][1];  SelA[i][2] = e2[i][2];
        }

        printf("\nMinimum Spanning Tree in network A is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) = %d\n",SelA[i][0],SelA[i][1],SelA[i][2]);
                Sum_MST = Sum_MST + SelA[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);

//รับค่าจำนวนเส้นของโครงข่าย B
        printf("\nNetwork B : Number of edges = ");
        scanf("%d",&b);
```

```
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<b;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&B[i][0],&B[i][1],&B[i][2]);
        }

        printf("\nNetwork B\n");
        for(i=0;i<b;i++){
                        printf("(%d,%d) = %d\n",B[i][0],B[i][1],B[i][2]);
                }

        for(i=0;i<b;i++){
                e1[i][0] = B[i][0];     e1[i][1] = B[i][1];     e1[i][2] = B[i][2];
        }
        SortEdges(b);
        for(i=0;i<b;i++){
                SrtB[i][0] = e2[i][0];  SrtB[i][1] = e2[i][1];  SrtB[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(b);

        for(i=0;i<b;i++){
                e1[i][0] = SrtB[i][0];  e1[i][1] = SrtB[i][1];  e1[i][2] = SrtB[i][2];
        }
        SelectEdges(b);
        for(i=0;i<b;i++){
                SelB[i][0] = e2[i][0];  SelB[i][1] = e2[i][1];  SelB[i][2] = e2[i][2];
        }

        printf("\nMinimum Spanning Tree in network B is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) = %d\n",SelB[i][0],SelB[i][1],SelB[i][2]);
                Sum_MST = Sum_MST + SelB[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);

//รับค่าจำนวนของเส้นที่ปะติดกัน
        printf("\nNumber of edges in the Clone = ");
        scanf("%d",&c);

//รับค่าของแต่ละเส้นที่ปะติดกัน
        for(i=0;i<c;i++){
                printf("Edge#%d.\tNetwork A's edge : ",i+1);
                        scanf("%d,%d",&C[i][0],&C[i][1]);
                        printf("\tto glue with\n");
                        printf("\tNetwork B's edge : ",i+1);
```

```
                scanf("%d,%d",&D[i][0],&D[i][1]);
        }

        for(i=0;i<b;i++){
                for(j=0;j<2;j++){
                        for(k=0;k<c;k++){
                                for(l=0;l<2;l++){
                                        if (B[i][j] == D[k][l]){
                                                B[i][j] = C[k][l];
                                        }
                                }
                        }
                }
        }

        printf("\n");
        printf("\nIn the glued graph, \nAll vertices in the clone will use the vertex
names from network A. ");
        printf("\nSo,Network A glued B is\n");

        n=0;
        for(i=0;i<a;i++){
                        E1[n][0] = A[i][0];
                        E1[n][1] = A[i][1];
                        E1[n][2] = A[i][2];
                        n++;
        }

        for(i=0;i<b;i++){
                Chk='N';
                for(j=0;j<a;j++){
                        if((B[i][0]==A[j][0]) && (B[i][1]==A[j][1])){
                                Chk='Y';
                        }
                }
                if (Chk=='N'){
                        E1[n][0] = B[i][0];
                        E1[n][1] = B[i][1];
                        E1[n][2] = B[i][2];
                        n++;
                }
        }

        for(i=0;i<n;i++){
                printf("(%d,%d) = %d\n",E1[i][0],E1[i][1],E1[i][2]);
        }

        for(i=0;i<n;i++){
                e1[i][0] = E1[i][0];    e1[i][1] = E1[i][1];    e1[i][2] = E1[i][2];
        }
```

```
        SortEdges(n);

        for(i=0;i<n;i++){
                E2[i][0] = e2[i][0];    E2[i][1] = e2[i][1];    E2[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(n);


        for(i=0;i<n;i++){
                e1[i][0] = E2[i][0];    e1[i][1] = E2[i][1];    e1[i][2] = E2[i][2];
        }

        SelectEdges(n);

        for(i=0;i<n;i++){
                Sel_Edge[i][0] = e2[i][0];      Sel_Edge[i][1] = e2[i][1];
        Sel_Edge[i][2] = c2[i][2];
        }

        printf("\nMinimum Spanning Tree in the glued graph is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) =
%d\n",Sel_Edge[i][0],Sel_Edge[i][1],Sel_Edge[i][2]);
                Sum_MST = Sum_MST + Sel_Edge[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);
        printf("\nPress any key to exit....");
        getch();

        return 0;
}
```

4.4 โปรแกรมเพื่อประยุกต์ใช้กราฟปะติดในการแก้ปัญหาโครงข่ายอี-โลจิสติกส์

```
int glue4_e_logistic()
{
        int A[100][3], B[100][3], C[100][2], D[100][2];
        int SrtA[100][3], SrtB[100][3], SelA[100][3], SelB[100][3];
        int E1[100][3], E2[100][3], E_TMP[3];
        int Sel_Edge[100][3];
        int i,j,k,l,m,n;
        char Chk;
        char f1,f2;
        int a, b, c, y, z;
        int Sum_MST, Cnt_Dot;
```

```
        printf("This program needs 2 connected input networks i.e. Network A &
 B\n\n");
 //รับค่าจำนวนเส้นของโครงข่าย A
        printf("input Network A \n ");
        printf("Network A : Number of edges = ");
                scanf("%d",&a);
 //รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<a;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&A[i][0],&A[i][1],&A[i][2]);
        }

        printf("\nNetwork A\n");
        for(i=0;i<a;i++){
                        printf("(%d,%d) = %d\n",A[i][0],A[i][1],A[i][2]);
                }

        for(i=0;i<a;i++){
                e1[i][0] = A[i][0];     e1[i][1] = A[i][1];     e1[i][2] = A[i][2];
        }
        SortEdges(a);
        for(i=0;i<a;i++){
                SrtA[i][0] = e2[i][0];  SrtA[i][1] = e2[i][1];  SrtA[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(a);

        for(i=0;i<a;i++){
                e1[i][0] = SrtA[i][0];  e1[i][1] = SrtA[i][1];  e1[i][2] = SrtA[i][2];
        }
        SelectEdges(a);
        for(i=0;i<a;i++){
                SelA[i][0] = e2[i][0];  SelA[i][1] = e2[i][1];  SelA[i][2] = e2[i][2];
        }

        printf("\nMinimum Spanning Tree in network A is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) = %d\n",SelA[i][0],SelA[i][1],SelA[i][2]);
                Sum_MST = Sum_MST + SelA[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);
//รับค่าจำนวนเส้นของโครงข่าย B
        printf("input Network B \n ");
        printf("\nNetwork B : Number of edges = ");
        scanf("%d",&b);
```

```
//รับค่าจุดปลายของแต่ละเส้น
        for(i=0;i<b;i++){
                printf("Edge#%d is between vertices : ",i+1);
                scanf("%d,%d %d",&B[i][0],&B[i][1],&B[i][2]);
        }

        printf("\nNetwork B\n");
        for(i=0;i<b;i++){
                        printf("(%d,%d) = %d\n",B[i][0],B[i][1],B[i][2]);
                }

        for(i=0;i<b;i++){
                e1[i][0] = B[i][0];     e1[i][1] = B[i][1];     e1[i][2] = B[i][2];
        }
        SortEdges(b);
        for(i=0;i<b;i++){
                SrtB[i][0] = e2[i][0];  SrtB[i][1] = e2[i][1];  SrtB[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(b);

        for(i=0;i<b;i++){
                e1[i][0] = SrtB[i][0];  e1[i][1] = SrtB[i][1];  e1[i][2] = SrtB[i][2];
        }
        SelectEdges(b);
        for(i=0;i<b;i++){
                SelB[i][0] = e2[i][0];  SelB[i][1] = e2[i][1];  SelB[i][2] = e2[i][2];
        }

        printf("\nMinimum Spanning Tree in network B is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) = %d\n",SelB[i][0],SelB[i][1],SelB[i][2]);
                Sum_MST = Sum_MST + SelB[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);

//รับค่าจำนวนของเส้นที่ปะติดกัน
        printf("\nNumber of edges in the Clone = ");
        scanf("%d",&c);

//รับค่าของแต่ละเส้นที่ปะติดกัน
        for(i=0;i<c;i++){
                printf("Edge#%d.\tNetwork A's edge : ",i+1);
                        scanf("%d,%d",&C[i][0],&C[i][1]);
                        printf("\tto glue with\n");
                        printf("\tNetwork B's edge : ",i+1);
```

```
                scanf("%d,%d",&D[i][0],&D[i][1]);
        }


        for(i=0;i<b;i++){
                for(j=0;j<2;j++){
                        for(k=0;k<c;k++){
                                for(l=0;l<2;l++){
                                        if (B[i][j] == D[k][l]){
                                                B[i][j] = C[k][l];
                                        }
                                }
                        }
                }
        }

        printf("\n");
        printf("\nIn the glued graph, \nAll vertices in the clone will use the vertex
names from network A. ");
        printf("\nSo,Network A glued B is\n");

        n=0;
        for(i=0;i<a;i++){
                        E1[n][0] = A[i][0];
                        E1[n][1] = A[i][1];
                        E1[n][2] = A[i][2];
                        n++;
        }

        for(i=0;i<b;i++){
                Chk='N';
                for(j=0;j<a;j++){
                        if((B[i][0]==A[j][0]) && (B[i][1]==A[j][1])){
                                Chk='Y';
                        }
                }
                if (Chk=='N'){
                        E1[n][0] = B[i][0];
                        E1[n][1] = B[i][1];
                        E1[n][2] = B[i][2];
                        n++;
                }
        }

        for(i=0;i<n;i++){
                printf("(%d,%d) = %d\n",E1[i][0],E1[i][1],E1[i][2]);
        }

        for(i=0;i<n;i++){
                e1[i][0] = E1[i][0];    e1[i][1] = E1[i][1];    e1[i][2] = E1[i][2];
```

```
        }

        SortEdges(n);

        for(i=0;i<n;i++){
                E2[i][0] = e2[i][0];    E2[i][1] = e2[i][1];    E2[i][2] = e2[i][2];
        }

        Cnt_Dot = CountDot(n);

        for(i=0;i<n;i++){
                e1[i][0] = E2[i][0];    e1[i][1] = E2[i][1];    e1[i][2] = E2[i][2];
        }

        SelectEdges(n);

        for(i=0;i<n;i++){
                Sel_Edge[i][0] = e2[i][0];      Sel_Edge[i][1] = e2[i][1];
        Sel_Edge[i][2] = e2[i][2];
        }

        printf("\nMinimum Spanning Tree in the glued graph is\n");
        Sum_MST = 0;
        for(i=0;i<Cnt_Dot-1;i++){
                printf("(%d,%d) =
%d\n",Sel_Edge[i][0],Sel_Edge[i][1],Sel_Edge[i][2]);
                Sum_MST = Sum_MST + Sel_Edge[i][2];
        }
        printf("\nwith weight = ");
        printf("%d\n",Sum_MST);
        printf("\nPress any key to exit....");
        getch();

        return 0;
}

int SortEdges(int n)
{
        int i, j, e_tmp[3];

        e2[0][0] = e1[0][0];
        e2[0][1] = e1[0][1];
        e2[0][2] = e1[0][2];
        for(i=1;i<n;i++){

                if (e1[i][2] < e2[i-1][2]){
                        e_tmp[0] = e2[i-1][0];
                        e_tmp[1] = e2[i-1][1];
                        e_tmp[2] = e2[i-1][2];
```

```
                        e2[i-1][0] = e1[i][0];
                        e2[i-1][1] = e1[i][1];
                        e2[i-1][2] = e1[i][2];

                        e2[i][0] = e_tmp[0];
                        e2[i][1] = e_tmp[1];
                        e2[i][2] = e_tmp[2];

                        j=i-1;
                        while(j > 0){
                                if (e2[j][2] < e2[j-1][2]){

                                        e_tmp[0] = e2[j-1][0];
                                        e_tmp[1] = e2[j-1][1];
                                        e_tmp[2] = e2[j-1][2];

                                        e2[j-1][0] = e2[j][0];
                                        e2[j-1][1] = e2[j][1];
                                        e2[j-1][2] = e2[j][2];

                                        e2[j][0] = e_tmp[0];
                                        e2[j][1] = e_tmp[1];
                                        e2[j][2] = e_tmp[2];
                                }
                                j--;
                        }
                }else{
                        e2[i][0] = e1[i][0];
                        e2[i][1] = e1[i][1];
                        e2[i][2] = e1[i][2];
                }
        }
        return e2[100][3];
}

int SelectEdges(int n)
{
        int ETmp1, ETmp2, z, i, j, k, l, pt;
        char ChkFlag1, ChkFlag2;

        e2[0][0] = e1[0][0];    e2[0][1] = e1[0][1];    e2[0][2] = e1[0][2];
        e2[1][0] = e1[1][0];    e2[1][1] = e1[1][1];    e2[1][2] = e1[1][2];
        z = 2;
        for(i=2;i<n;i++){
                ETmp1 = 0;
                ETmp2 = 0;
                ChkFlag1 = ' ';
                ChkFlag2 = ' ';

                for(j=0;j<i;j++){
```

```
if( ChkFlag1 == ' ' && ( e1[i][0] == e2[j][0] || e1[i][0] == e2[j][1] ) ){
                ChkFlag1 = 'Y';
        }

if( ChkFlag2 == ' ' && ( e1[i][1] == e2[j][0] || e1[i][1] == e2[j][1] ) ){
                ChkFlag2 = 'Y';
        }
}

if((ChkFlag1 == 'Y') && (ChkFlag2 == 'Y')){

        ETmp1 = e1[i][0];
                for(j=0;j<i;j++){
                        for(k=0;k<i;k++){
                                if(ETmp2 == 0){
                                        if(e1[i][1] == e2[k][0]){
                                        ETmp2 = e2[k][1];

                                }else if(e1[i][1] == e2[k][1]){
                                        ETmp2 = e2[k][0];

                                        }
                                }else{
                                        if(ETmp1 == ETmp2){
                                                k = i;
                                        }else{
                                                for(l=0;l<i;l++){


if(ETmp2 == e2[l][0]){

ETmp2 = e2[l][1];

}else if(ETmp2 == e2[l][1]){

ETmp2 = e2[l][0];

//pt = l;
                                                                                }

if(ETmp1 == ETmp2){

k = i;
l = i;
                                                                                }

                                                                        }
                                                                }
                                                        }
                                                }
```

```
                                        }
                        if(ETmp1 != ETmp2){
                e2[z][0] = e1[i][0];    e2[z][1] = e1[i][1];    e2[z][2] = e1[i][2];
                                z++;
                        }
                }else{
                e2[z][0] = e1[i][0];    e2[z][1] = e1[i][1];    e2[z][2] = e1[i][2];
                        z++;
                }

        }
        return e2[100][3];
}

int CountDot(int n)
{
        int y, i, j, Amt_Dot[100];;
        char ChkDot;
        y = 0;

        for(i=0;i<n;i++){

                ChkDot = ' ';
                for(j=0;j<y;j++){

                        if(e2[i][0] == Amt_Dot[j]){
                                ChkDot = 'Y';
                        }
                }
                if (ChkDot != 'Y'){
                        Amt_Dot[y] = e2[i][0];
                        y++;
                }

                ChkDot = ' ';
                for(j=0;j<y;j++){

                        if(e2[i][1] == Amt_Dot[j]){
                                ChkDot = 'Y';
                        }
                }
                if (ChkDot != 'Y'){
                        Amt_Dot[y] = e2[i][1];
                        y++;
                }
        }

        return y;
}
```

ประวัติผู้เขียน

ชื่อ – สกุล	นายจิรพงษ์ เมฆเวียน
วัน เดือน ปีเกิด	14 ธันวาคม 2525
สถานที่เกิด	ปทุมธานี
ประวัติการศึกษา	มหาวิทยาลัยราชภัฏสวนดุสิต
วุฒิการศึกษา	ครุศาสตรบัณฑิต
ปีการศึกษาที่จบ	2548