

โปรแกรมเพื่อช่วยในการแก้ปัญหาการหาเส้นทางที่สั้นที่สุดที่มีความเสี่ยง

นางสาวเจนจิรา แผ่นทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2558

Application for Helping in Shortest Path Problem with Risk

Ms. Chenchira Panthong

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF BACHELOR OF COMPUTER ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY NORTH BANGKOK
ACADEMIC YEAR 2015

ปริญญาบัตรเรื่อง : โปรแกรมเพื่อช่วยในการแก้ปัญหาการหาเส้นทางที่สั้นที่สุดที่มีความ
 เสี่ยง
 ชื่อ : นางสาวเจนจิรา แผ่นทอง
 สาขา : วิศวกรรมคอมพิวเตอร์
 ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์
 คณะ : วิศวกรรมศาสตร์
 อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.ศิริชัย รุจิพัฒน์พงศ์
 ผู้ช่วยศาสตราจารย์ ดร.ชยรัช เผือกสามัญ
 ปีการศึกษา : 2558

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ อนุมัติให้
ปริญญาบัตรนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต สาขา
วิชาวิศวกรรมไฟฟ้า

..... หัวหน้าภาควิชาวิศวกรรมไฟฟ้า
(ผู้ช่วยศาสตราจารย์ ดร.นภคล วิวัชร โกเศศ) และคอมพิวเตอร์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ศิริชัย รุจิพัฒน์พงศ์)

..... กรรมการ


(ผู้ช่วยศาสตราจารย์ ดร.ชยรัช เพื่อกสภามัญ)

..... กรรมการ
(รองศาสตราจารย์ไชยพันธุ์ สวรรณชีวะศิริ)


ลิขสิทธิ์ของภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ


Project Report Title : Application for Helping in Shortest Path Problem with Risk
Name : Ms. Chenchira Panthong
Major Field : Computer Engineering
Department : Electrical and Computer Engineering
Faculty : Engineering
Project Advisor(s) : Asst. Prof. Dr. Sirichai Rujipattanapong
Asst. Prof. Dr. Chayathuch Phuaksaman
Academic Year : 2015

Accepted by the Faculty of Engineering, King Mongkut's University of Technology North Bangkok in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Engineering.


..... Chairperson of Department of Electrical
(Asst. Prof. Dr. Nophadon Wiwatcharagoses) and Computer Engineering


..... Chairperson
(Asst. Prof. Dr. Sirichai Rujipattanapong)


..... Member
(Asst. Prof. Dr. Chayathuch Phuaksaman)


..... Member
(Asst. Prof. Chaiyan Suwanchewasiri)

Copyright of the Department of Electrical and Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology North Bangkok

บทคัดย่อ

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อพัฒนาการแก้ปัญหาทางโลจิสติกส์ในหัวข้อการหาระยะทางที่สั้นที่สุดที่มีความเสี่ยง มีวัตถุประสงค์หลักคือจัดทำระบบให้เป็น Web Server พร้อมทั้งพัฒนาระบบให้สามารถใช้งาน Google Maps ควบคู่กับระบบการคำนวณเพื่อให้ง่ายและสะดวกสบายต่อผู้ใช้ โดยการใช้งาน Google Maps จะใช้ Google API ที่เป็นชุด API ของ Google เป็นอุปกรณ์ในการเรียกใช้แผนที่และชุด Service ต่าง ๆ ในส่วนของการคำนวณจะใช้ Dijkstra Algorithm ในการคำนวณหาระยะทางที่สั้นที่สุดของการเดินทาง

Abstract

This Project Report has been prepared in order to develop the Problem Solution in Logistics of which the title is “Application for Seeking the Shortest Travelling / Transportation Path with Risks”. The main purpose of the project is to set the system to serve as a Web Server and to develop the system to enable the use of Google Maps with calculating system, in order to facilitate the users. By using the Google Maps, the Google API which is the API from the Google will be used to display the maps and their other functional services. The Dijkstra Algorithm will be used to calculate the shortest path for the trip.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สามารถสำเร็จลุล่วงได้ด้วยความรู้ความกรุณาจากท่านอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.ศิริชัย รุจิพัฒน์พงศ์ และท่านอาจารย์ ผู้ช่วยศาสตราจารย์ ดร.ชยรัช เพื่อksamัญ และคณะอาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ให้คำปรึกษา ชี้แนะแนวทางการค้นคว้า และให้ความรู้แก่ผู้จัดทำ จึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

ขอกราบขอบพระคุณบิดา มารดา และขอขอบคุณเพื่อน ที่คอยให้คำปรึกษาและกำลังใจ ในการทำงานจนงานสามารถสำเร็จลุล่วงได้ด้วยดี

เจนจิรา แผ่นทอง

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	จ
บทคัดย่อภาษาอังกฤษ	ฉ
กิตติกรรมประกาศ	ช
สารบัญตาราง	ฅ
สารบัญภาพ	ญ
คำอธิบายสัญลักษณ์และคำย่อ	ฐ
บทที่ 1. บทนำ	1
1.1 ความสำคัญและที่มาของปัญหา	1
1.2 วัตถุประสงค์ของโครงการ	1
1.3 ขอบเขตของการศึกษาโครงการ	2
1.4 ประโยชน์ที่คาดว่าจะได้รับ	2
บทที่ 2. ทฤษฎีและหลักการที่เกี่ยวข้อง	3
2.1 Shortest Path Problem with Risk (SPR)	3
2.2 สมการที่เกี่ยวข้อง	6
บทที่ 3. วิธีการจัดทำโครงการ	9
3.1 อุปกรณ์	9
3.2 วิธีการจัดทำโครงการ	11
3.3 Diagram ของระบบ	30
บทที่ 4. ผลการทดลอง	31
4.1 การทดสอบระบบ	31
บทที่ 5. สรุป วิจัยรณัผล และข้อเสนอแนะ	38
เอกสารอ้างอิง	39
ประวัติผู้แต่ง	40

สารบัญตาราง

ตารางที่	หน้า
3-1 การคำนวณค่าน้ำหนักของเส้นทางจากสมการ Sum of Weight	16

สารบัญภาพ

ภาพที่	หน้า
2-1 ตัวอย่างเส้นทาง	4
2-2 ตัวอย่างเส้นทางเส้นที่ 1	4
2-3 ตัวอย่างเส้นทางเส้นที่ 2	5
2-4 ตัวอย่างเส้นทางเส้นที่ 3	5
2-5 Pseudocode	7
3-1 ตัวอย่าง Google Maps เมื่อเรียกใช้งานผ่านเว็บไซต์	9
3-2 ตัวอย่างการค้นหาเส้นทาง	10
3-3 แผนที่บนหน้าเว็บไซต์และการปักจุดบนแผนที่	11
3-4 ค่า Latitude และ Longitude ที่ทำการปักหมุดบนหน้าเว็บไซต์	12
3-5 ระยะทาง และระยะเวลาในการเดินทางที่ร้องขอจาก Google API	12
3-6 ปุ่มที่ใช้ในการเก็บข้อมูลลงในไฟล์ Excel	12
3-7 การนำข้อมูลเก็บลงในไฟล์ Excel	13
3-8 ปุ่มที่ใช้ในการเลือกไฟล์ Excel เพื่อทำการอ่านไฟล์	13
3-9 การนำข้อมูลที่อ่านจากไฟล์ Excel แสดงบนหน้าเว็บ	14
3-10 ส่วนหนึ่งของโปรแกรมที่ทำงานตามหลักการของ Dijkstra Algorithm	14
3-11 ตัวอย่างของไฟล์ Input ที่ใช้ในการทดสอบโปรแกรม	15
3-12 ผลลัพธ์ของโปรแกรมที่คำนวณจากไฟล์ input	15
3-13 ไฟล์ Output ของโปรแกรมที่คำนวณจากไฟล์ input	16
3-14 กราฟ Dijkstra Algorithm	17
3-15 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node B	17
3-16 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node B แล้ว	18
3-17 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node C	18
3-18 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node C แล้ว	19
3-19 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node D	19
3-21 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node E	20

สารบัญภาพ (ต่อ)

ภาพที่	หน้า
3-22 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node E แล้ว	21
3-23 กราฟของ Dijkstra Algorithm เมื่อ Node A เป็น Super Node	21
3-24 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node C	22
3-25 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node C	22
3-26 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node C แล้ว	23
3-27 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node D	23
3-28 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node D	24
3-29 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node D แล้ว	24
3-30 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node E	25
3-31 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node E	25
3-32 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node E แล้ว	26
3-33 กราฟ Dijkstra Algorithm เมื่อ Node B เป็น Super Node	26
3-34 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ D แล้ว	27
3-35 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ C แล้ว	28
3-36 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ E แล้ว	28
3-37 กราฟ Dijkstra Algorithm และผลลัพธ์ของปัญหา	29
3-38 การทำงานของระบบ	30
4-1 หน้าเว็บไซต์ที่ทำการเรียก	31
4-2 map บนหน้าเว็บไซต์ที่ทำการปักจุดแล้ว	32
4-3 ค่า Latitude และ Longitude ที่มาจากการปักหมุดบน map ในหน้าเว็บไซต์	32
4-4 ตารางที่แสดงหลังจากกดปุ่มตาราง	33
4-5 การแจ้งเตือนเมื่อค่าปัจจัยผิดพลาด	33
4-6 การแจ้งเตือนเมื่อจุด start ผิดพลาด	34
4-7 การแจ้งเตือนเมื่อจุด end ผิดพลาด	34
4-8 การบันทึกข้อมูลลงในไฟล์ input_SPR.xlsx	35

สารบัญภาพ (ต่อ)

ภาพที่	หน้า
4-9 ข้อมูลในไฟล์ input_SPR.xlsx ที่เรียกมาแสดงบนเว็บไซต์	35
4-10 หน้าเว็บไซต์ในส่วนของ Output	36
4-11 หน้าเว็บแสดงเส้นทางและตารางของผลลัพธ์	37
4-12 หน้าเว็บแสดงเส้นทางในการเดินทาง	37

คำอธิบายสัญลักษณ์และคำย่อ

สัญลักษณ์และคำย่อ

SPR

คำอธิบาย

Shortest Path Problem with Risk

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหา

โลจิสติกส์เชิงธุรกิจเป็นการจัดกิจกรรมทั้งหลายที่เกี่ยวข้องกับการสนับสนุนการผลิตและการกระจายสินค้า การขนส่งจึงมีบทบาทและความสำคัญกับบริษัท ไปจนถึงระดับประเทศ ซึ่งต้องใช้การขนส่งเพื่อเคลื่อนย้ายผลิตภัณฑ์ การขนส่งเป็นปัจจัยพื้นฐานการพัฒนาเศรษฐกิจและสังคมของประเทศ ความก้าวหน้าของเทคโนโลยีการขนส่งทำให้การเดินทางทั้งสินค้าและผู้โดยสารรวดเร็วและมีต้นทุนต่ำ ส่งผลให้เกิดการเปลี่ยนแปลงที่สำคัญกับโลกในหลาย ๆ ด้าน

แต่ปัจจุบันการแก้ไขปัญหาทางการขนส่งเป็นไปอย่างยากลำบาก เนื่องจากการสำรวจหาข้อมูลของเส้นทางยังคงใช้แรงคนในการสำรวจแต่ละเส้นทางเพื่อหาค่าความเสี่ยงของเส้นทางนั้น ๆ เมื่อต้องการสำรวจทุกเส้นทางจึงทำให้ใช้เวลาในการเก็บข้อมูลนาน และในการคำนวณหาเส้นทางที่จะเลือกใช้สำหรับการขนส่ง ยังไม่มีเครื่องมือที่รองรับการคำนวณแบบเต็มรูปแบบที่สามารถหาผลลัพธ์ได้เลย มีเพียงส่วนเสริมที่ช่วยสนับสนุนการคำนวณให้ง่ายขึ้นเท่านั้น ทำให้เวลาที่ใช้ในส่วนของการคำนวณใช้เวลานาน จึงมีการพัฒนาเครื่องมือขึ้นมาเพื่อสนับสนุนการคำนวณให้ได้ผลลัพธ์ที่รวดเร็วยิ่งขึ้น

ในการนี้ จึงขอเสนอโครงการเพื่อช่วยในการแก้ปัญหาค้นหาเส้นทางที่สั้นที่สุดที่มีความเสี่ยง ซึ่งจะทำให้การคำนวณหาผลลัพธ์ของการหาเส้นทางเป็นไปอย่างรวดเร็ว และใช้งานได้ง่าย

1.2 วัตถุประสงค์ของโครงการ

1.2.1 สามารถสร้างระบบที่ช่วยในการแก้ปัญหาค้นหาเส้นทางที่สั้นที่สุดที่มีความเสี่ยง ให้สามารถแก้ปัญหาได้อย่างรวดเร็ว

1.2.2 สามารถสร้างระบบที่ใช้งานได้ง่าย

1.2.3 สามารถสร้างระบบที่ใช้งาน Google map ได้ในหลาย ๆ ด้าน

1.3 ขอบเขตของการศึกษาโครงการ

- 1.3.1 สามารถปักจุดตำแหน่งการเดินทาง บน google map ในหน้าเว็บได้
- 1.3.2 สามารถหาค่าระยะทางและระยะเวลาในการเดินทางหลังจากการปักจุดเพื่อไปคำนวณได้
- 1.3.3 ผู้ใช้สามารถกรอกความเสี่ยงของแต่ละเส้นทางได้
- 1.3.4 สามารถนำข้อมูลตำแหน่ง, ระยะทาง, เวลา, ความเสี่ยง ไปเก็บในไฟล์ Excel เพื่อคำนวณได้
- 1.3.5 นำผลลัพธ์ที่ได้จากการคำนวณมาแสดงผลที่หน้าเว็บบน google map และตารางบนหน้าเว็บได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

- 1.4.1 ได้เรียนรู้และฝึกทักษะการหาข้อมูล
- 1.4.2 ได้เรียนรู้และฝึกทักษะการเขียนโปรแกรมด้วยภาษา Java, JavaScript, PHP และHTML
- 1.4.3 ได้เรียนรู้และฝึกทักษะการเขียนโปรแกรมเพื่ออ่านและเขียนไฟล์ Excel
- 1.4.4 ได้เรียนรู้และฝึกทักษะการแก้ปัญหาเฉพาะหน้า

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

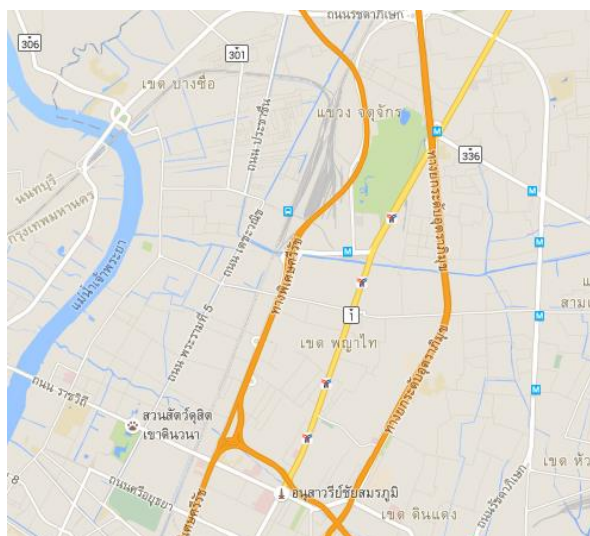
2.1 Shortest Path Problem with Risk (SPR)

ปัญหาการค้นหาเส้นทางที่มีระยะทางที่สั้นที่สุดโดยมีความเสี่ยง หรือ Shortest Path Problem with Risk วัตถุประสงค์หลักของปัญหา คือ ต้องการหาเส้นทางที่มีระยะทางสั้นที่สุดและใช้ระยะเวลาน้อยที่สุด รวมทั้งยังต้องเป็นเส้นทางที่มีความสะดวกสบายในการเดินทางและความปลอดภัยมากที่สุดด้วย ซึ่งต้องพิจารณาระยะเวลาเดินทาง ระยะทาง และความเสี่ยงในการเดินทางไปยังจุดปลายทาง

ในการแก้ปัญหาการหาเส้นทางที่สั้นที่สุด โดยทั่วไปเพื่อต้องการหาระยะทาง ระยะเวลา หรือ ค่าใช้จ่ายในการเดินทางที่น้อยที่สุด ซึ่งวิธีการหาคำตอบของปัญหาดังกล่าวนี้นี้มีเงื่อนไขและวัตถุประสงค์ที่หลากหลาย เช่น ต้องการหาเส้นทางที่มีระยะทางและระยะเวลาในการเดินทางที่น้อยที่สุด มีความปลอดภัยมากที่สุด รวมทั้งยังต้องเป็นเส้นทางที่มีความสะดวกสบายในการเดินทางมากที่สุดด้วย อีกทั้งยังมีความไม่แน่นอนของระยะเวลาเดินทางในแต่ละเส้นทาง หรือ กล่าวคือ ระยะเวลามีความแปรปรวนแตกต่างกัน ซึ่งอาจจะเกิดขึ้นได้จากปัจจัยต่าง ๆ เช่น สภาพอากาศ อุบัติเหตุ ความต้องการในการเดินทางของผู้ใช้เส้นทางในแต่ละวัน โดยทั่วไปความไม่แน่นอนของระยะเวลาเดินทางที่เกิดขึ้น จะอาศัยหลักการทางสถิติมาประยุกต์ใช้เพื่อแก้ปัญหา

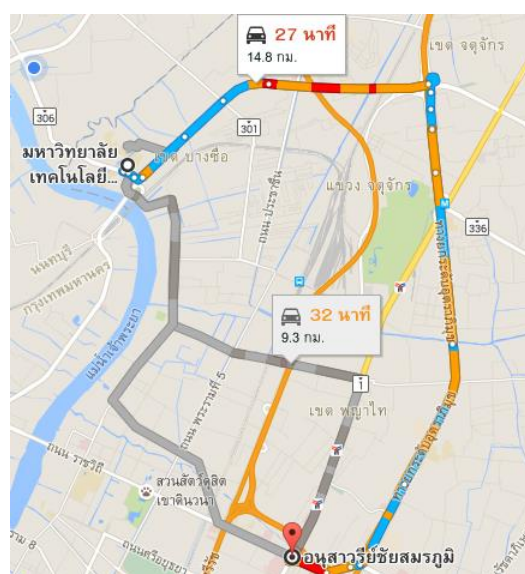
โดยรวมแล้วการแก้ปัญหาเรื่องการหาเส้นทางที่สั้นที่สุดนั้น จำเป็นต้องคำนึงถึงระยะทางและเวลาในการเดินทางที่น้อยที่สุด โดยต้องคำนึงถึงความปลอดภัยของเส้นทาง ความสะดวกสบายในการเดินทาง

2.1.1 ตัวอย่างของปัญหา

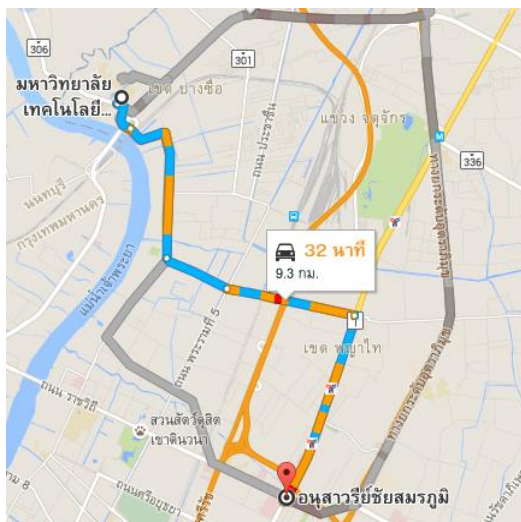


ภาพที่ 2-1 ตัวอย่างเส้นทาง

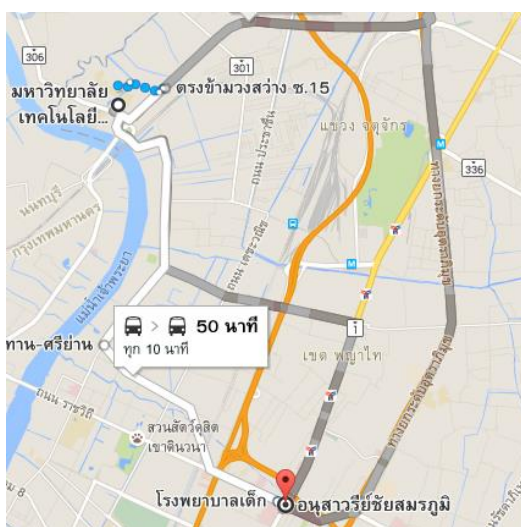
ถ้าต้องการหาเส้นทางจาก มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ไปยัง
อนุสาวรีย์ชัยสมรภูมิ



ภาพที่ 2-2 ตัวอย่างเส้นทางเส้นที่ 1



ภาพที่ 2-3 ตัวอย่างเส้นทางเส้นที่ 2



ภาพที่ 2-4 ตัวอย่างเส้นทางเส้นที่ 3

เส้นทางที่ไปได้จะมีทั้งหมด 3 เส้นทาง แต่ละเส้นทางนั้นจะมีระยะทาง ความ สะดวกสบาย และเวลาเดินทางที่แตกต่างกันจึงจำเป็นต้องทำการคำนวณความเสี่ยงต่าง ๆ ของแต่ละเส้นทางเพิ่มเข้าไป เพื่อที่จะได้ทำการเลือกเส้นทางที่สั้นที่สุด สะดวกสบายที่สุด และใช้เวลา น้อยที่สุด เป็นเส้นทางที่ดีที่สุดในการเดินทาง

2.2 สมการที่เกี่ยวข้อง

2.2.1 Sum of Weight

ในการแก้ปัญหา การหาระยะทางที่สั้นที่สุดที่มีความเสี่ยง หลักการสำคัญที่จะใช้คือการคำนวณค่าน้ำหนักของระยะทางแต่ละเส้นทางรวมกับความเสี่ยง เพื่อใช้เลือกเส้นทางที่เหมาะสม มีรูปแบบทั่วไปของแบบจำลองทางคณิตศาสตร์ดังต่อไปนี้

$$\text{Dijkstra Algorithm : Min } C_{ij} X_{ij} \quad (2-1)$$

โดยมีตัวแปรสำคัญ 3 ส่วน

2.2.1.1 ค่าใช้จ่าย คือ ระยะทางจากจุดหนึ่งถึงอีกจุดหนึ่งคูณกับค่าใช้จ่ายต่อกิโลเมตร

2.2.1.2 ระยะเวลา คือ ระยะเวลาในการเดินทางจากจุดหนึ่งถึงอีกจุดหนึ่ง

2.2.1.3 ค่าความเสี่ยง คือ ค่าความไม่ปลอดภัยต่างๆ เช่น ลักษณะของเส้นทาง ความปลอดภัยของสินค้า ความปลอดภัยของคนขับ ฯลฯ จากจุดหนึ่งถึงอีกจุดหนึ่ง

Sum of Weight :

$$\overline{W}_{ij} = (W_1 * \text{ค่าใช้จ่าย}_{ij}) + (W_2 * \text{เวลา}_{ij}) + (W_3 * \text{ค่าความเสี่ยง}_{ij}) \quad (2-2)$$

โดยที่ \overline{W}_{ij} คือ น้ำหนักรวมของเส้นทาง

W_1 คือ ค่าเปอร์เซ็นต์น้ำหนักของค่าใช้จ่ายซึ่งคิดจากระยะทาง

W_2 คือ ค่าเปอร์เซ็นต์น้ำหนักของเวลา

W_3 คือ ค่าเปอร์เซ็นต์น้ำหนักของค่าความเสี่ยง

i คือ จุดเริ่มต้น

j คือ จุดหมายปลายทาง

เมื่อทำการคำนวณค่าตามสมการข้างต้นรวมกับค่าความเสี่ยงจะได้ค่าน้ำหนักของแต่ละเส้นทาง หลังจากนั้นเลือกค่าน้ำหนักของเส้นทางที่น้อยที่สุดตามเงื่อนไขของสมการ

ในการเลือกต้องคำนึงถึงต้นทางและปลายทาง เพื่อให้ได้เส้นทางที่เชื่อมโยงกันแล้วสามารถเดินทางไปถึงจุดหมายปลายทางได้ หลังจากนั้นจะได้เส้นทางที่ดีที่สุดสำหรับการเดินทาง

2.2.2 Dijkstra Algorithm

Dijkstra's algorithm ถูกคิดค้นขึ้นโดยนักวิทยาการคอมพิวเตอร์ชาวดัตช์ที่มีนามว่า แอ็ดส์เกอร์ ไคก์สตรา (Edsger Dijkstra) ในปี 1959 เพื่อแก้ไขปัญหาการหาเส้นทางที่สั้นที่สุดจากจุดหนึ่งใด ๆ สำหรับกราฟที่มีความยาวของเส้นเชื่อมไม่เป็นลบ สำหรับขั้นตอนวิธี Dijkstra's Algorithm นี้จะหาระยะทางสั้นที่สุดจากจุดหนึ่งไปยังจุดใด ๆ ในกราฟโดยจะหาเส้นทางที่สั้นที่สุดไปที่ละจุดยอดเรื่อย ๆ จนครบตามที่ต้องการ

2.2.2.1 Pseudocode

```

dist[s] ← 0                                (distance to source vertex is zero)
for all v ∈ V - {s}
do dist[v] ← ∞                             (set all other distances to infinity)
S ← ∅                                       (S, the set of visited vertices is initially empty)
Q ← V                                       (Q, the queue initially contains all vertices)
while Q ≠ ∅                                (while the queue is not empty)
do u ← mindistance(Q, dist)               (select the element of Q with the min. distance)
   S ← S ∪ {u}                             (add u to list of visited vertices)
   for all v ∈ neighbors[u]
   do if dist[v] > dist[u] + w(u, v)        (if new shortest path found)
      then d[v] ← d[u] + w(u, v)          (set new value of shortest path)
                                          (if desired, add traceback code)

return dist

```

ภาพที่ 2-5 Pseudocode

2.2.2.2 แนวคิด

2.2.2.2.1 เนื่องจากค่าน้ำหนักในเส้นเชื่อมโยงไม่มีค่าติดลบ แสดงว่ายิ่งเดินทางผ่านโหนดมากเท่าไร ค่าน้ำหนักก็จะยิ่งเพิ่มมากขึ้นเท่านั้น

2.2.2.2.2 จากโหนดเริ่มต้น ถ้ามองออกไปรอบ ๆ ที่เพื่อนบ้าน หากเลือกเส้นเชื่อมโยงที่มีค่าน้อยที่สุด เราเลือกเส้นนั้นและจะเป็นเส้นทางที่สั้นที่สุดจากจุดเริ่มต้น

2.2.2.2.2.1 เมื่อเส้นเชื่อมโยงไม่มีค่าติดลบ ดังนั้นเส้นทางที่ถูกเลือกแล้วจะไม่มีทางถูกเปลี่ยนได้แน่นอน

2.2.2.2.3 ผนวกกับโหนดที่ไปถึงแล้วเข้ามาเป็น Super Node พร้อมกับเส้นเชื่อมโยงของโหนด จากนั้นทำไปเรื่อย ๆ จนไม่พบโหนดใหม่ในกราฟอีก

2.2.2.2.4 ทำการเลือกเส้นทางจากจุดสุดท้ายที่ต้องการไปหาจุดเริ่มต้นโดยเลือกจากค่าน้ำหนัก จะได้เส้นทางที่ดีที่สุด

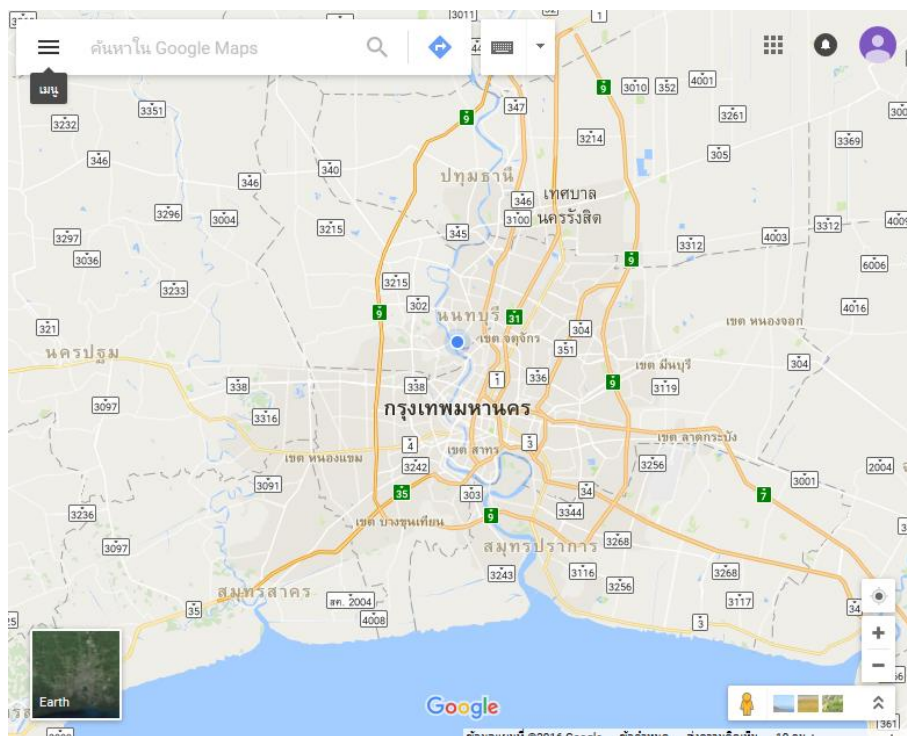
บทที่ 3

วิธีการจัดทำโครงการ

3.1 อุปกรณ์

3.1.1 Google Maps

Google Maps คือ บริการของ Google ที่ให้บริการด้านแผนที่ที่ให้บริการฟรี และสามารถใช้งานได้ง่าย โดยจะให้ข้อมูลเส้นทางการขับขี่ ที่ตั้งของธุรกิจท้องถิ่น และข้อมูลต่าง ๆ สิ่งดึงดูดใจให้มีผู้ใช้งานอย่างแพร่หลายคือ แผนที่และภาพถ่ายทางดาวเทียมคุณภาพดี ซึ่งครอบคลุมทั่วโลก



ภาพที่ 3-1 ตัวอย่าง Google Maps เมื่อเรียกใช้งานผ่านเว็บไซต์

Google Maps API เป็นชุด API ของ Google ใช้สำหรับการพัฒนา web application และ mobile application (Android, iOS) ซึ่งจะใช้ในการเรียกใช้แผนที่และบริการต่าง ๆ ของ Google เพื่อพัฒนา Application ให้ทำงานได้คล้ายกับ Google Maps โดยแผนที่ยังมีคุณสมบัติต่าง ๆ มากมายที่น่าสนใจให้เรียกใช้

3.1.1.1 การปรับแต่งแผนที่ (Styled Map)

3.1.1.2 ชุดควบคุมแผนที่ (Map Control)

3.1.1.3 ชุดเครื่องมือวาดภาพบนแผนที่ (Drawing)

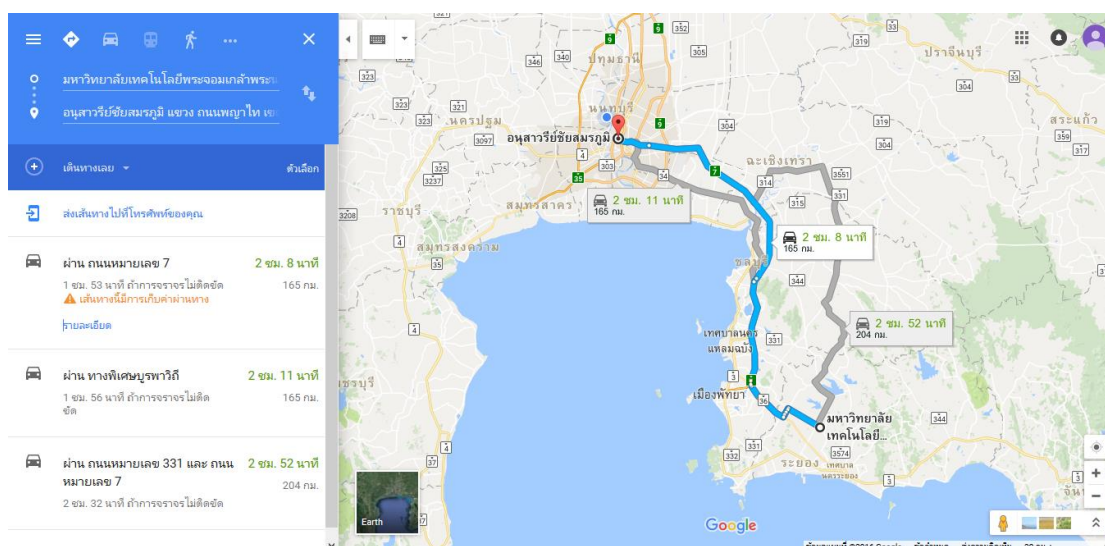
3.1.1.4 การนำทางจากจุดหนึ่งไปยังอีกจุดหนึ่ง (Directions Service)

3.1.1.5 การคำนวณความสูงของจุดพิกัด (Elevation Service)

3.1.1.6 การแปลงที่อยู่เป็นพิกัด Latitude และ Longitude (GeoCoding Service)

3.1.1.7 การดึงข้อมูล POI (Point of Interest) คือข้อมูลสถานที่ต่าง ๆ ที่ Google รวบรวมไว้ให้ เช่น โรงแรม ห้างสรรพสินค้า โรงเรียน –สถานที่ราชการต่าง ๆ และอื่น ๆ อีกมากมาย (Places API) มาใช้งานใน application เรา

3.1.1.8 Street View

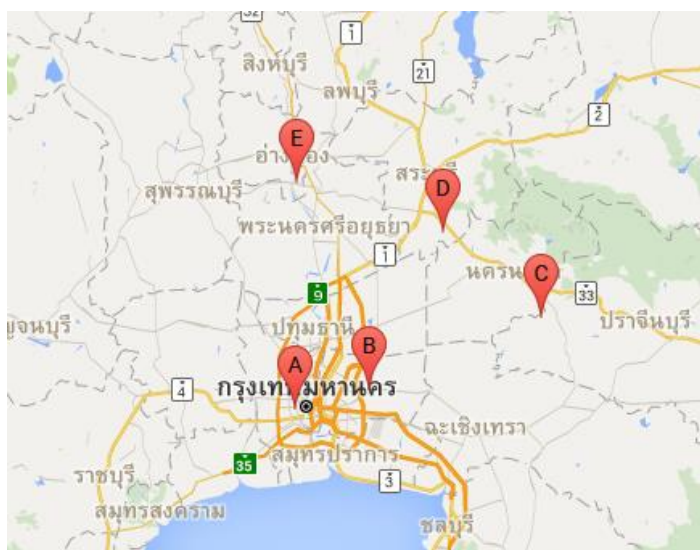


ภาพที่ 3-2 ตัวอย่างการค้นหาเส้นทาง

3.2 วิธีการจัดทำโครงงาน

3.2.1 ขั้นตอนที่ 1 เริ่มจากการศึกษาการทำงานของ Google API โดยทดลองใช้งานผ่านทางหน้าเว็บไซต์ มีขั้นตอนการศึกษาดังนี้

3.2.1.1 ทำการเขียนโปรแกรมด้วยภาษา JavaScript และ HTML เพื่อเรียก google map ผ่านทางหน้าเว็บไซต์ และทำให้ map นั้นสามารถปักจุดได้



ภาพที่ 3-3 แผนที่บนหน้าเว็บไซต์และการปักจุดบนแผนที่

3.2.1.2 เขียนโปรแกรมเพิ่มเติมให้สามารถทำการรับค่า Latitude และ Longitude จากการปักจุดบนหน้าเว็บไซต์ และทำการแสดงค่า Latitude และ Longitude บนหน้าเว็บไซต์

ตำแหน่งทางเดิน
A Latitude: 13.748055898499004 Longitude: 100.46653747558594
B Latitude: 13.810076180474649 Longitude: 100.7171630859375
C Latitude: 14.053995128467742 Longitude: 101.30218505859375
D Latitude: 14.333581907130224 Longitude: 100.96710205078125
E Latitude: 14.501167248306615 Longitude: 100.469970703125

ภาพที่ 3-4 ค่า Latitude และ Longitude ที่ทำการปักหมุดบนหน้าเว็บไซต์

3.2.1.3 เขียนโปรแกรมเพิ่มเติมให้สามารถทำการร้องขอเส้นทางการเดินทาง ระยะทาง และเวลาในการเดินทาง จากการปักจุดบนแผนที่ในหน้าเว็บไซต์ โดยร้องขอจาก Google API

ต้นทาง	ปลายทาง	ระยะทาง	เวลาในการเดินทาง
(13.7484833, 100.466253600000007)	(13.8103802, 100.716585000000001)	46.6 กม.	54 นาที
(13.7484833, 100.466253600000007)	(14.0542218, 101.302150799999994)	141 กม.	2 ชั่วโมง 28 นาที
(13.7484833, 100.466253600000007)	(14.3312715, 100.965998799999997)	115 กม.	1 ชั่วโมง 47 นาที
(13.7484833, 100.466253600000007)	(14.5016731, 100.469637599999994)	105 กม.	1 ชั่วโมง 33 นาที
(13.8103802, 100.716585000000001)	(14.0542218, 101.302150799999994)	99.3 กม.	1 ชั่วโมง 43 นาที
(13.8103802, 100.716585000000001)	(14.3312715, 100.965998799999997)	98.2 กม.	1 ชั่วโมง 23 นาที
(13.8103802, 100.716585000000001)	(14.5016731, 100.469637599999994)	104 กม.	1 ชั่วโมง 19 นาที
(14.0542218, 101.302150799999994)	(14.3312715, 100.965998799999997)	59.7 กม.	1 ชั่วโมง 16 นาที
(14.0542218, 101.302150799999994)	(14.5016731, 100.469637599999994)	127 กม.	2 ชั่วโมง 20 นาที
(14.3312715, 100.965998799999997)	(14.5016731, 100.469637599999994)	81.7 กม.	1 ชั่วโมง 18 นาที

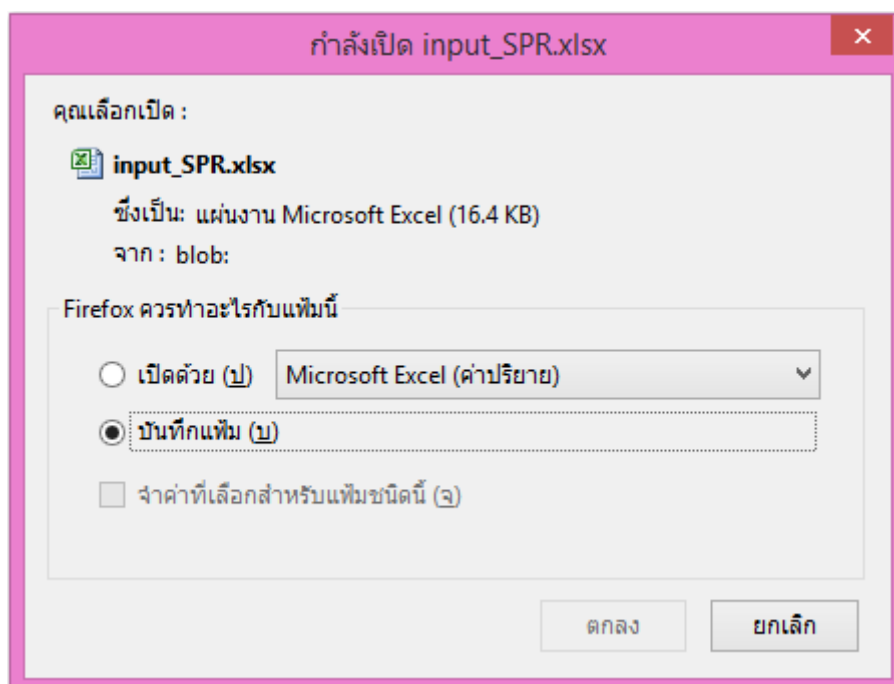
ภาพที่ 3-5 ระยะทาง และระยะเวลาในการเดินทางที่ร้องขอจาก Google API

3.2.2 ขั้นตอนที่ 2 ทำการศึกษาการเขียนและอ่านไฟล์ Excel ผ่านทางหน้าเว็บไซต์

3.2.2.1 เขียนโปรแกรมด้วยภาษา JavaScript และ HTML เพื่อเก็บข้อมูลลงในไฟล์ Excel และให้หน้าเว็บไซต์แสดงปุ่มที่จะทำการเก็บข้อมูล โดยเมื่อกดปุ่มบันทึกจะนำข้อมูลที่จัดไว้บันทึก ลงในไฟล์ input_SPR.xlsx

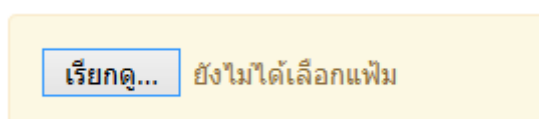
Export to Excel!

ภาพที่ 3-6 ปุ่มที่ใช้ในการเก็บข้อมูลลงในไฟล์ Excel



ภาพที่ 3-7 การนำข้อมูลเก็บลงในไฟล์ Excel

3.2.2.2 เขียนโปรแกรมเพิ่มเติมให้สามารถอ่านข้อมูลในไฟล์ Excel และแสดงข้อมูลที่อ่านบนหน้าเว็บไซต์



ภาพที่ 3-8 ปุ่มที่ใช้ในการเลือกไฟล์ Excel เพื่อทำการอ่านไฟล์

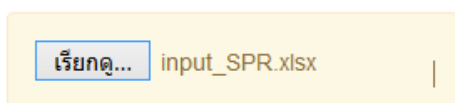


Table 1								
The weight of the factors								
cost (%)	time (%)	risk (%)						
37	13	50						
start	14.20048839	99.94354226	finish	13.64598681	101.141052			
Table 2								
start	Latitude	Longitude	end	Latitude	Longitude	distance(m)	time(s)	risk
A	14.2003603	99.9436171	B	14.1366159	101.0751871	155594	9019	0
A	14.2003603	99.9436171	C	13.6457972	101.1416877	192972	9769	0
B	14.1366159	101.0751871	C	13.6457972	101.1416877	73379	5125	0

ภาพที่ 3-9 การนำข้อมูลที่อ่านจากไฟล์ Excel แสดงบนหน้าเว็บ

3.2.3 ขั้นตอนที่ 3 ทำการศึกษา Dijkstra Algorithm และเขียนโปรแกรมด้วยภาษา Java เพื่อให้โปรแกรมสามารถทำงานได้ตามหลัก Dijkstra Algorithm อย่างมีประสิทธิภาพ โดยส่วนของ Java จะมีการอ่านไฟล์ input_SPR.xlsx เพื่อรับค่า input หลังจากนั้นจะทำการคำนวณตามหลักของ Dijkstra Algorithm และทำการนำผลลัพธ์เขียนลงในไฟล์ output_SPR.xlsx

```

////////////////// dijkstra ////////////////////

public void d(int start){
    q = new LinkedListQueue<Vertex>();
    v = new Vertex[mat.length];
    for(int i=0; i<mat.length; i++){
        v[i] = new Vertex(i);
    }
    if(start<mat.length){
        v[start].set(null, 0);
        q.enqueue(v[start]);
        //System.out.println(q);
    }
    while(!q.isEmpty()){
        Vertex a = (Vertex) q.dequeue();
        for(int i=0; i<mat[0].length; i++){
            if(mat[a.index][i] > 0){
                if(v[i].dist == -1){
                    v[i].set(a, mat[a.index][i]+v[a.index].dist);
                    q.enqueue(v[i]);
                }
                else if(v[i].dist > mat[a.index][i]+v[a.index].dist){
                    v[i].set(a, mat[a.index][i]+v[a.index].dist);
                    q.enqueue(v[i]);
                }
            }
        }
    }
}

```

ภาพที่ 3-10 ส่วนหนึ่งของโปรแกรมที่ทำงานตามหลักการของ Dijkstra Algorithm

3.2.3.1 ทำการตรวจสอบความถูกต้องของโปรแกรม โดยทำการสมมติตัวอย่างของไฟล์

Input ดังภาพที่ 3-11 กำหนดให้จุด A เป็นจุดเริ่มต้น และจุด E เป็นจุดปลายทาง

	A	B	C	D	E	F	G	H	I
1	table 1								
2	The weight								
3	cost (%)	time (%)	risk (%)						
4	33	35	32						
5	start	14.06199	99.50592	finish	14.85189	101.6895			
6	table 2								
7	start	Latitude	Longitude	end	Latitude	Longitude	distance	time	risk
8	A	14.06213	99.50542	B	13.88603	100.6045	141941	8045	23
9	A	14.06213	99.50542	C	14.23677	101.092	217021	10984	43
10	A	14.06213	99.50542	D	14.65862	100.8774	191276	10104	65
11	A	14.06213	99.50542	E	14.85199	101.6894	346823	27184	85
12	B	13.88603	100.6045	C	14.23677	101.092	82296	4908	65
13	B	13.88603	100.6045	D	14.65862	100.8774	104192	5703	23
14	B	13.88603	100.6045	E	14.85199	101.6894	190970	9357	43
15	C	14.23677	101.092	D	14.65862	100.8774	61084	3569	54
16	C	14.23677	101.092	E	14.85199	101.6894	136930	6808	12
17	D	14.65862	100.8774	E	14.85199	101.6894	118634	5789	54

ภาพที่ 3-11 ตัวอย่างของไฟล์ Input ที่ใช้ในการทดสอบโปรแกรม

เมื่อนำไฟล์ input_SPR.xlsx ใสลงในโปรแกรม แล้วทำการรันโปรแกรม โปรแกรมจะทำการคำนวณหาผลลัพธ์ และนำผลลัพธ์ที่ได้เขียนลงไฟล์ Excel พร้อมทั้งปริ้นค่าแสดงผลที่คำนวณได้

way : 0 3 4

ภาพที่ 3-12 ผลลัพธ์ของโปรแกรมที่คำนวณจากไฟล์ input

โดย 0 => A, 3 => D และ 4 => E

	A	B	C	D	E
1		Latitude	Longitude		
2	0	14.062129	99.505422		
3	3	14.658618	100.8773721		
4	4	14.851991	101.6894104		
5					
6					

ภาพที่ 3-13 ไฟล์ Output ของโปรแกรมที่คำนวณจากไฟล์ input

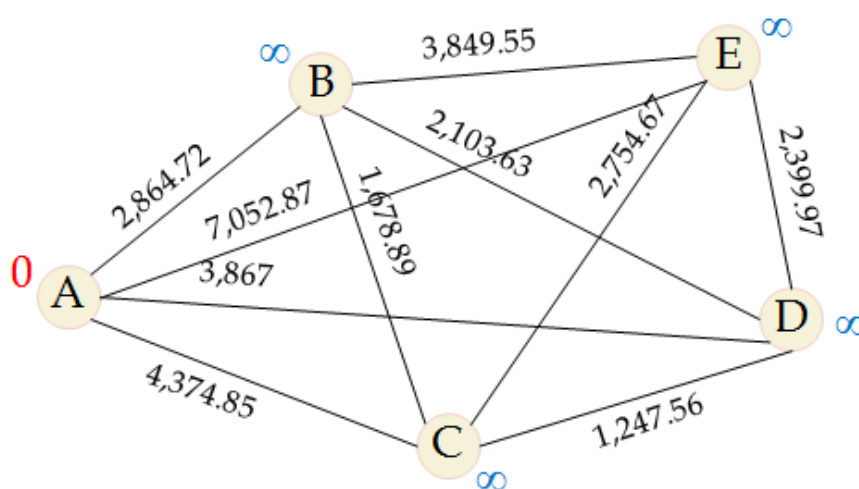
3.2.3.1.1 นำค่าที่อ่านได้จากไฟล์ Input ที่สมมติขึ้น นำมาทำการคำนวณตามสมการ Sum of Weight เพื่อใช้เป็นค่าน้ำหนักใน Dijkstra Algorithm โดยปรับระยะทางให้เป็นกิโลเมตร (km) แล้วคูณด้วยค่าใช้จ่ายต่อกิโลเมตร ปรับเวลาจากวินาทีเป็นนาที แล้วคำนวณตามสมการ จะได้ผลการคำนวณดังตารางที่ 3-1

ตารางที่ 3-1 แสดงการคำนวณค่าน้ำหนักของเส้นทางจากสมการ Sum of Weight

Start	End	distance(m)	cost(60/km)	time(s)	time(m)	risk	\bar{W}_{ij}
A	B	141,941	8,516.46	8045	134.0833	23	2,864.72
A	C	21,7021	13,021.26	10984	183.0667	43	4,374.85
A	D	191,276	11,476.56	10104	168.4	65	3,867
A	E	346,823	20,809.38	27184	453.0667	85	7,052.87
B	C	82,296	4,937.76	4908	81.8	65	1,678.89
B	D	104,192	6,251.52	5703	95.05	23	2,103.63
B	E	190,970	11,458.2	9357	155.95	43	3,849.55
C	D	61,084	3,665.04	3569	59.4833	54	1,247.56
C	E	136,930	8,215.8	6808	113.4667	12	2,754.67
D	E	118,634	7,118.04	5784	96.4	54	2,399.97

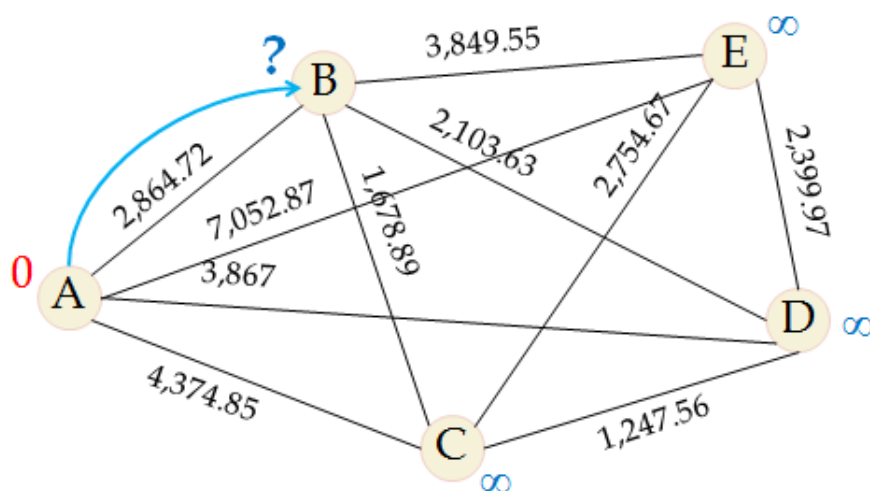
3.2.3.1.2 ทำการคำนวณหาผลลัพธ์จากไฟล์ input ที่สมมติตัวอย่างขึ้นมา ตามหลักของ Dijkstra Algorithm เพื่อหาเส้นทางที่เป็นผลลัพธ์

นำค่าที่คำนวณได้จากตารางที่ 3-1 กำหนดให้เป็นค่าน้ำหนักของแต่ละเส้นทางบนกราฟ Dijkstra Algorithm โดยเริ่มต้นกำหนดให้ค่าน้ำหนักของแต่ละ Node มีค่าเท่ากับ Infinity และจุดเริ่มต้นมีค่าน้ำหนักเท่ากับ 0 ดังภาพที่ 3-14



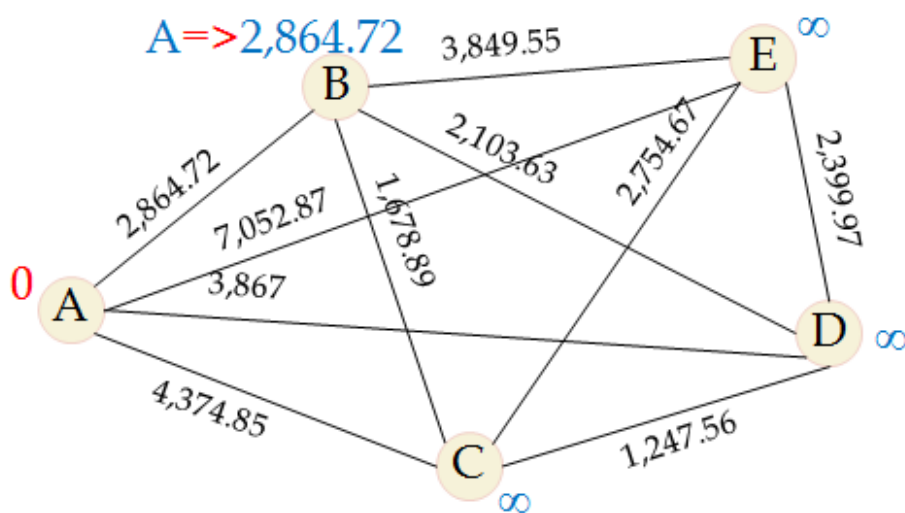
ภาพที่ 3-14 กราฟ Dijkstra Algorithm

Node A เริ่มพิจารณาเพื่อนบ้าน โดยพิจารณาที่ Node B ก่อน ดังภาพที่ 3-15



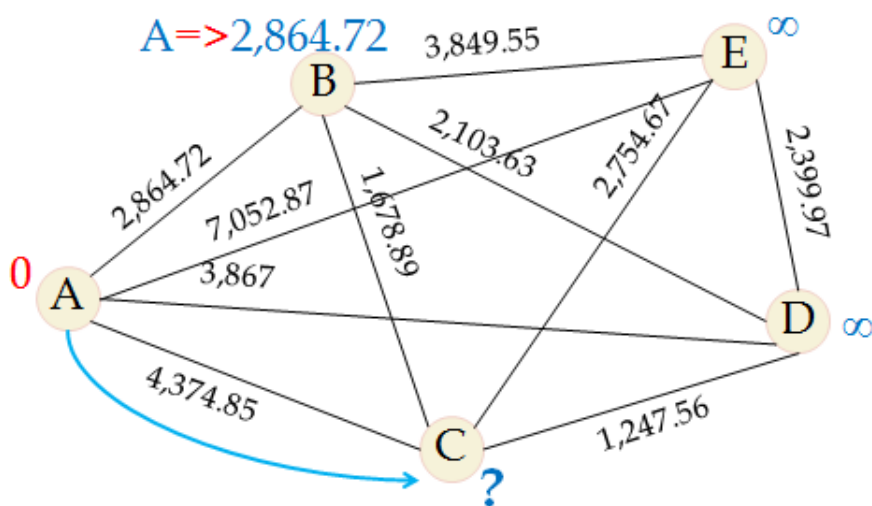
ภาพที่ 3-15 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node B

จะเห็นได้ว่าค่าน้ำหนักจาก Node A ไปยัง Node B มีค่าเท่ากับ 2,864.72 ซึ่งมีค่าน้อยกว่า Infinity ทำให้ค่าน้ำหนักที่ Node B เปลี่ยนเป็น 2,864.72 ดังภาพที่ 3-16



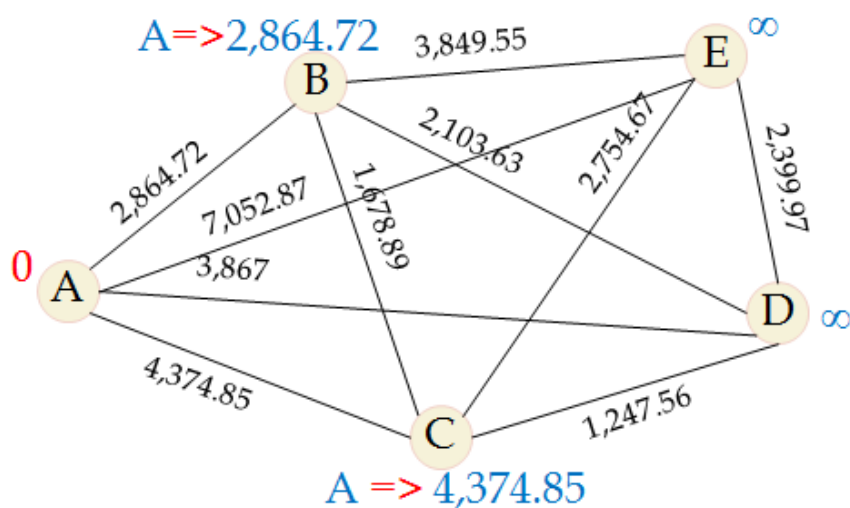
ภาพที่ 3-16 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node B แล้ว

หลังจากนั้น Node A จะพิจารณาที่ Node ถัดไปคือ Node C ดังภาพที่ 3-17



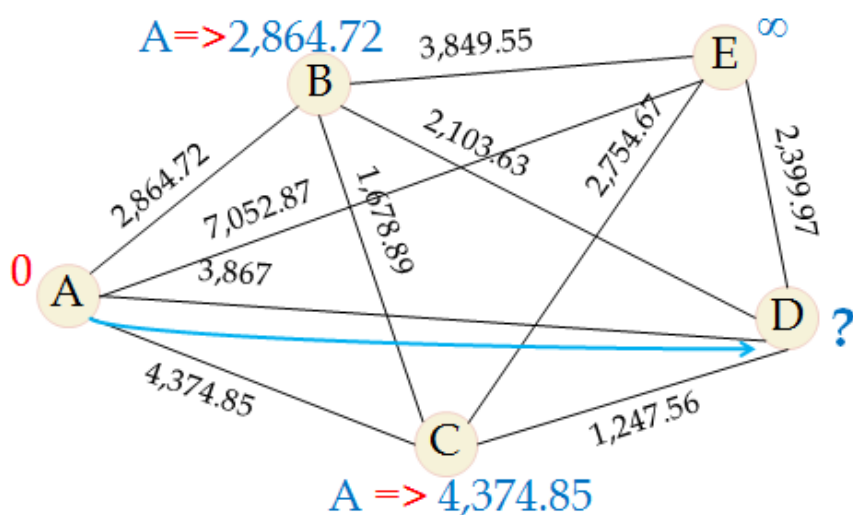
ภาพที่ 3-17 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node C

จะเห็นว่าค่าน้ำหนักจาก Node A ไปยัง Node C มีค่าเท่ากับ 4,374.85 ซึ่งมีค่าน้อยกว่า Infinity ทำให้ค่าน้ำหนักที่ Node C เปลี่ยนเป็น 4,374.85 ดังภาพที่ 3-18



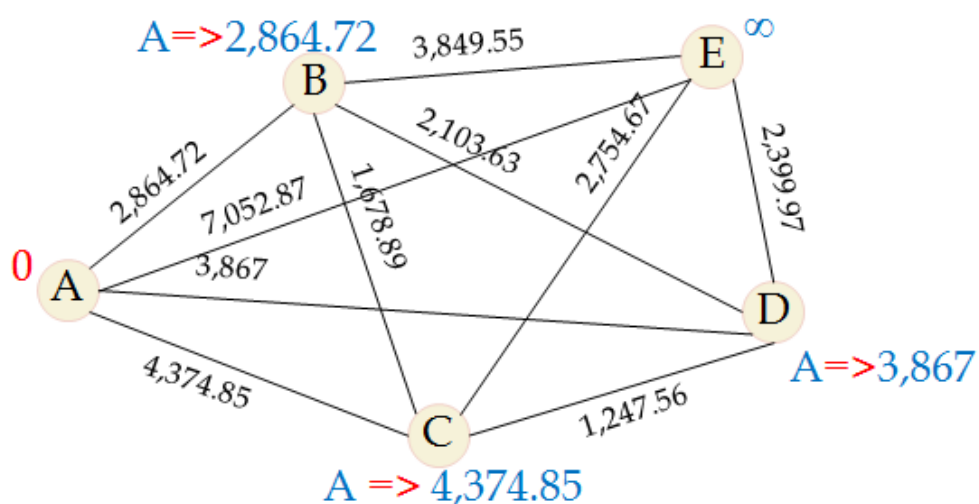
ภาพที่ 3-18 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node C แล้ว

หลังจากนั้น Node A จะพิจารณาที่ Node ถัดไปคือ Node D ดังภาพที่ 3-19



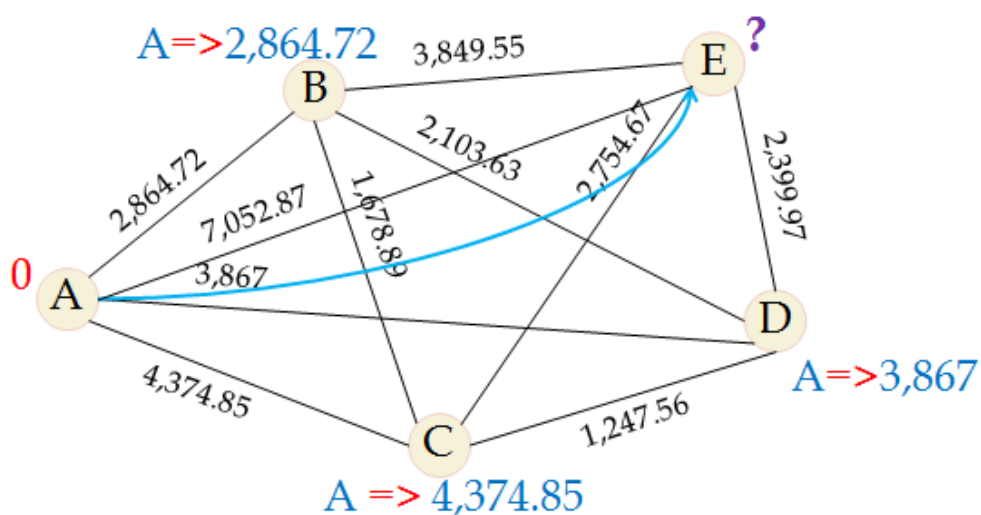
ภาพที่ 3-19 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node D

จะเห็นว่าค่าน้ำหนักจาก Node A ไปยัง Node D มีค่าเท่ากับ 3,867 ซึ่งมีค่าน้อยกว่า Infinity ทำให้ค่าน้ำหนักที่ Node D เปลี่ยนเป็น 3,867 ดังภาพที่ 3-20



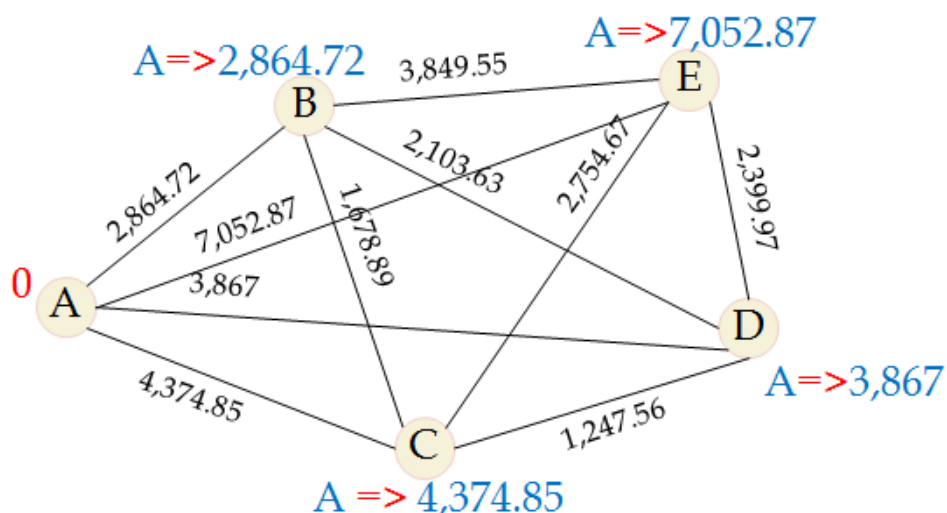
ภาพที่ 3-20 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node D แล้ว

หลังจากนั้น Node A จะพิจารณาที่ Node ถัดไปคือ Node E ดังภาพที่ 3-21



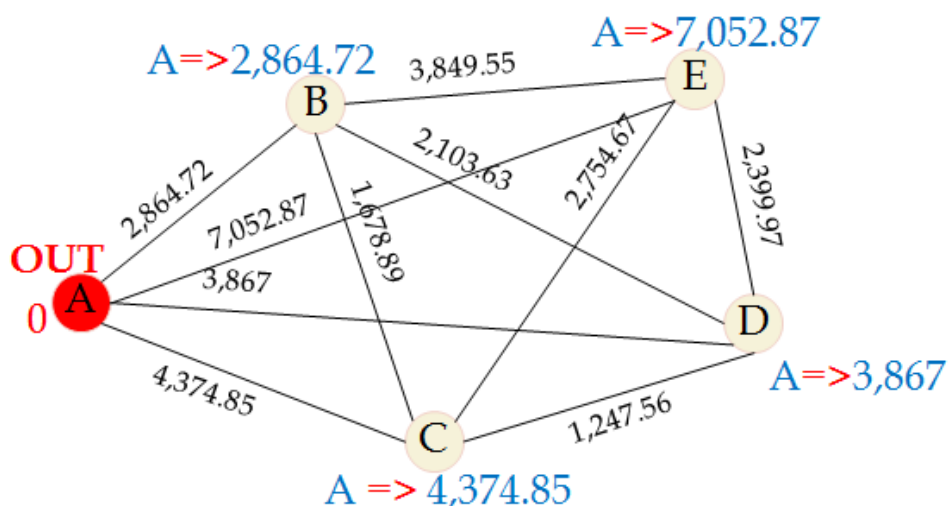
ภาพที่ 3-21 กราฟ Dijkstra Algorithm เมื่อ Node A เริ่มพิจารณา Node E

จะเห็นว่าค่าน้ำหนักจาก Node A ไปยัง Node E มีค่าเท่ากับ 7,052.87 ซึ่งมีค่าน้อยกว่า Infinity ทำให้ค่าน้ำหนักที่ Node E เปลี่ยนเป็น 7,052.87 ดังภาพที่ 3-22



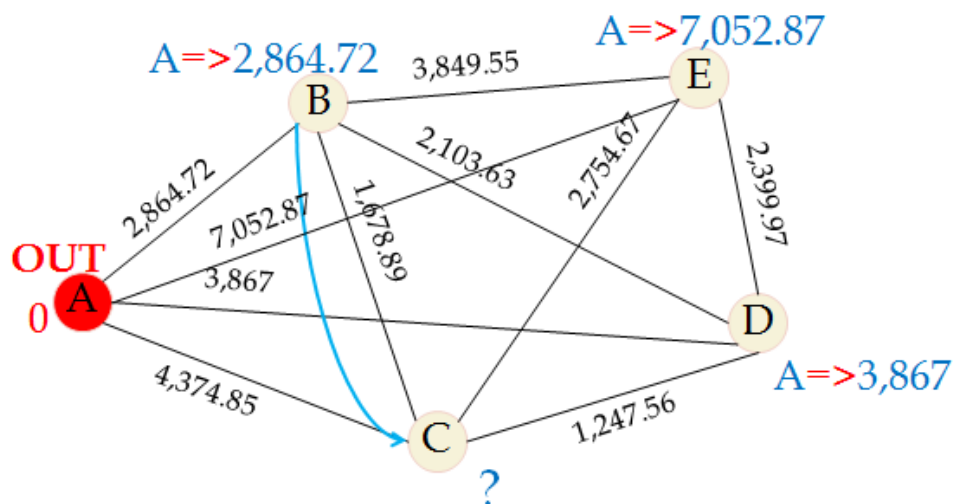
ภาพที่ 3-22 กราฟ Dijkstra Algorithm เมื่อ Node A พิจารณา Node E แล้ว

เมื่อพิจารณาเพื่อนบ้านที่ใกล้ Node A ครบทุก Node แล้ว จะทำให้ Node A กลายเป็น Super Node ดังภาพที่ 3-23



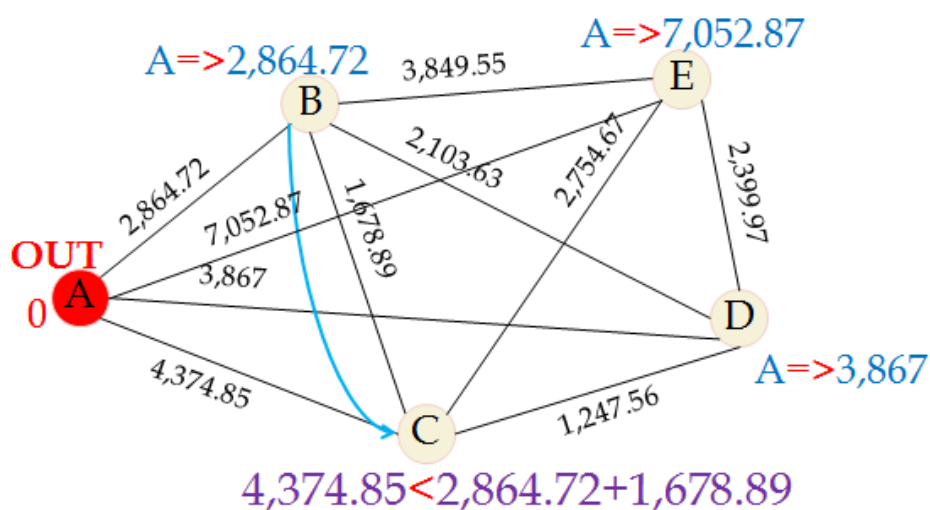
ภาพที่ 3-23 กราฟของ Dijkstra Algorithm เมื่อ Node A เป็น Super Node

หลังจากนั้น Node B จะเริ่มพิจารณา โดยพิจารณาที่ Node C ก่อน ดังภาพที่ 3-24



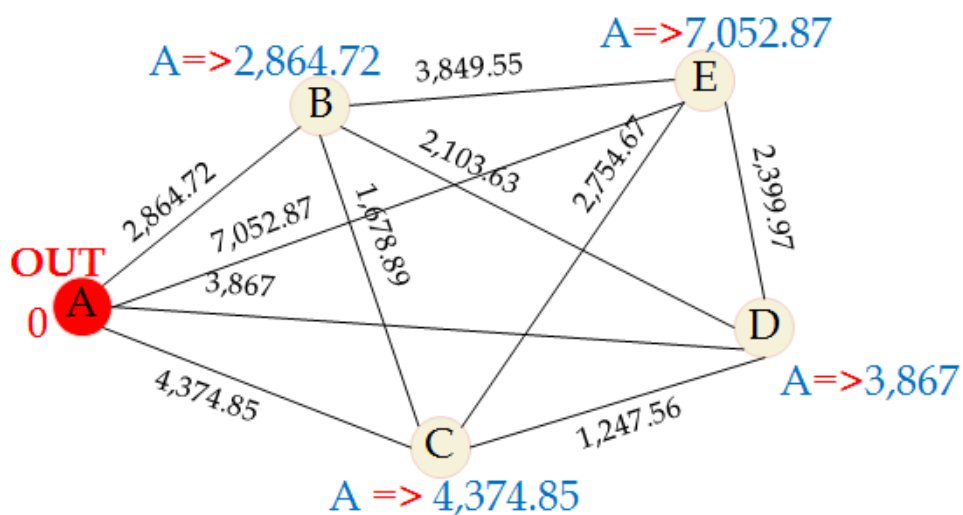
ภาพที่ 3-24 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node C

จะเห็นว่าค่าน้ำหนักจาก Node B ไปยัง Node C มีค่าน้ำหนักทั้งหมดเท่ากับ $2,864.72 + 1,678.89 = 4,543.61$ ซึ่งมากกว่าค่าน้ำหนักเดิมที่ Node C ดังภาพที่ 3-25



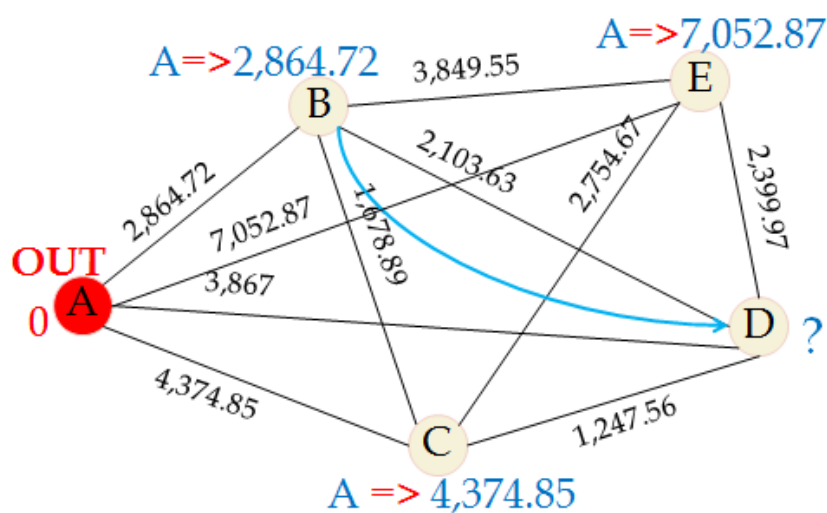
ภาพที่ 3-25 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node C

จะเห็นว่า ค่าน้ำหนักที่ Node C จะยังคงมีค่าเท่ากับ 4,374.85 ไม่เปลี่ยนแปลง
 ดังภาพที่ 3-26



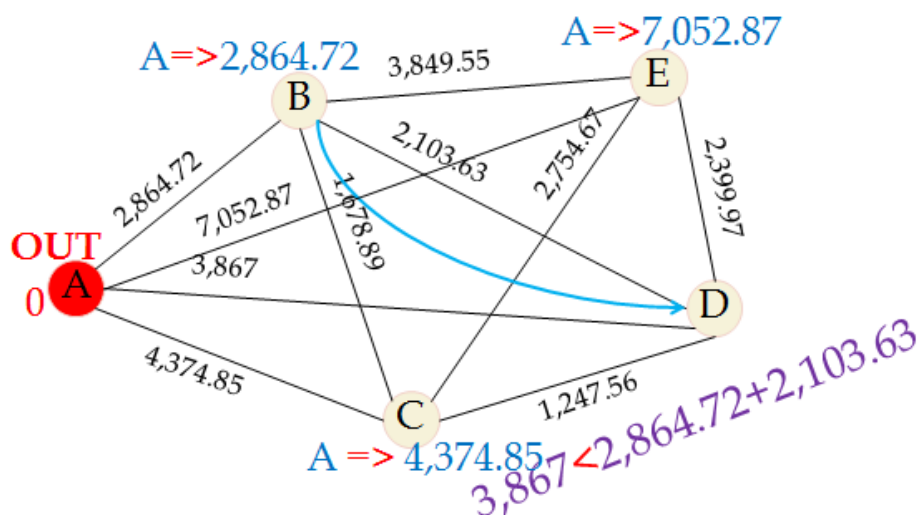
ภาพที่ 3-26 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node C แล้ว

หลังจากนั้น Node B จะพิจารณาที่ Node ถัดไปคือ Node D ดังภาพที่ 3-27



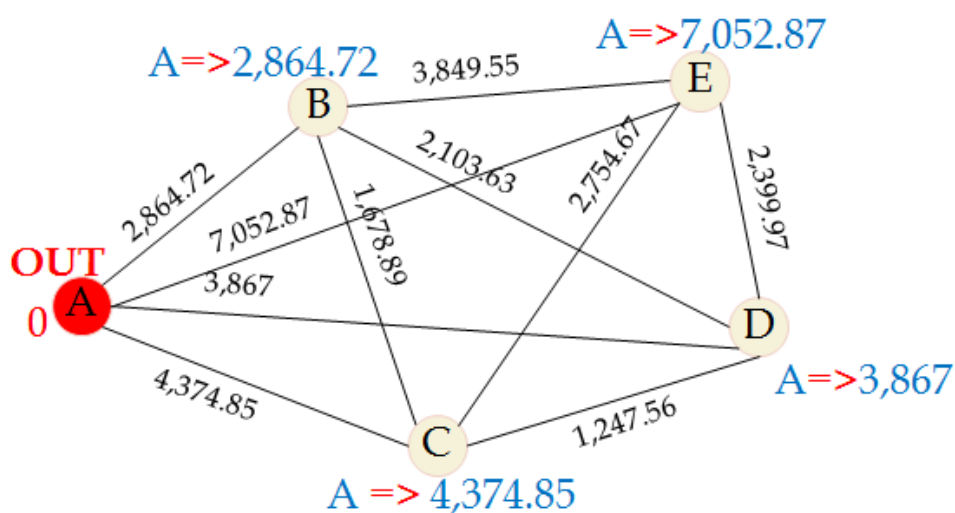
ภาพที่ 3-27 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node D

จะเห็นว่าค่าน้ำหนักจาก Node B ไปยัง Node D มีค่าน้ำหนักทั้งหมดเท่ากับ $2,864.72 + 2,103.63 = 4,968.35$ ซึ่งมากกว่าค่าน้ำหนักเดิมที่ Node D ดังภาพที่ 3-28



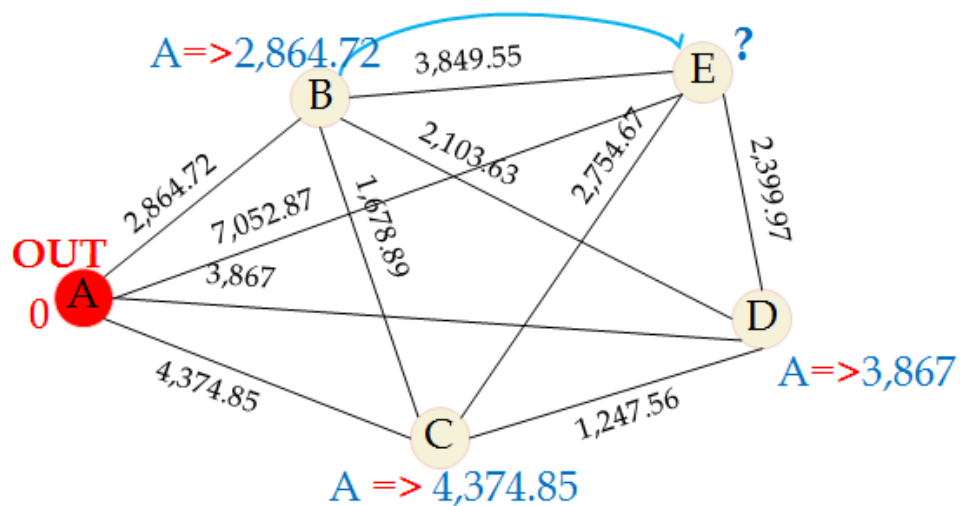
ภาพที่ 3-28 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node D

จะเห็นว่า ค่าน้ำหนักที่ Node D จะยังคงมีค่าเท่ากับ 3,867 ไม่เปลี่ยนแปลง ดังภาพที่ 3-29



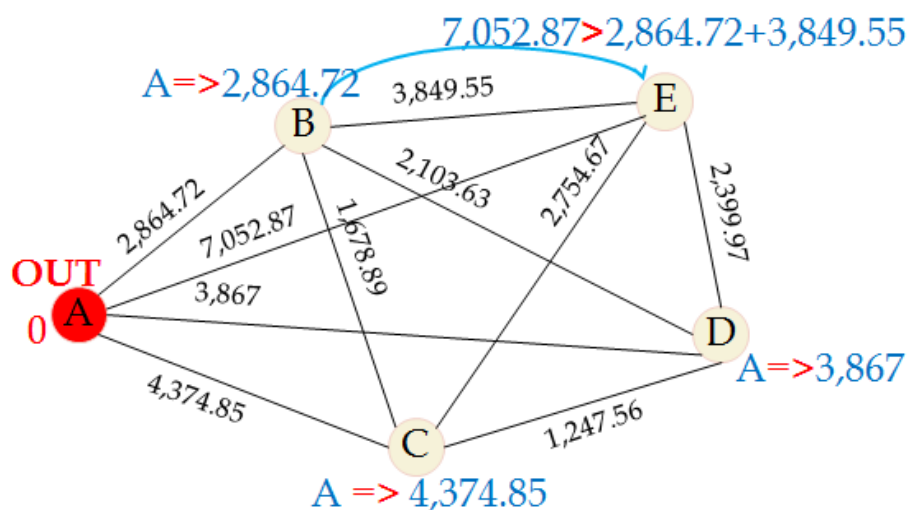
ภาพที่ 3-29 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node D แล้ว

หลังจากนั้น Node B จะพิจารณาที่ Node ถัดไปคือ Node E ดังภาพที่ 3-30



ภาพที่ 3-30 กราฟ Dijkstra Algorithm เมื่อ Node B เริ่มพิจารณา Node E

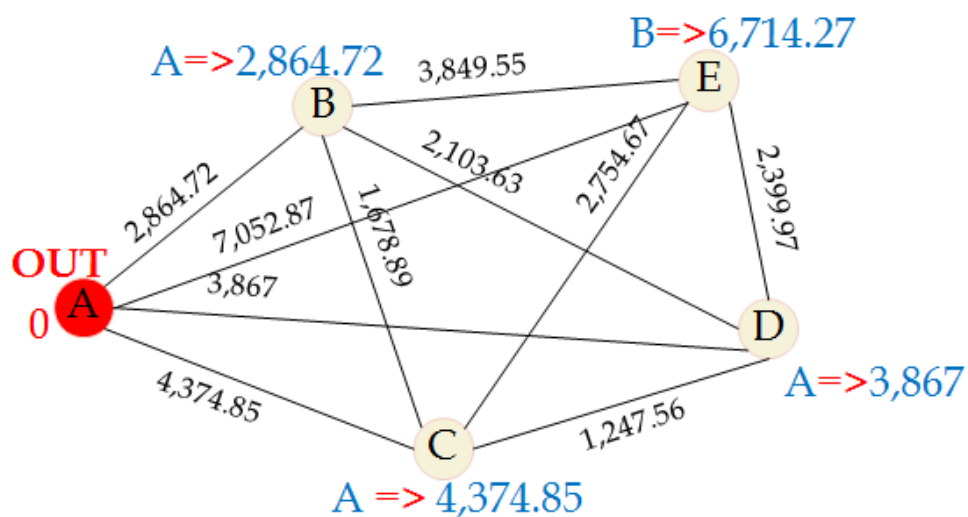
จะเห็นว่าค่าน้ำหนักจาก Node B ไปยัง Node E มีค่าน้ำหนักทั้งหมดเท่ากับ $2,864.72 + 3,849.55 = 6,714.27$ ซึ่งน้อยกว่าค่าน้ำหนักเดิมที่ Node E ดังภาพที่ 3-31



ภาพที่ 3-31 กราฟ Dijkstra Algorithm เมื่อ Node B เปรียบเทียบค่าน้ำหนักที่ Node E

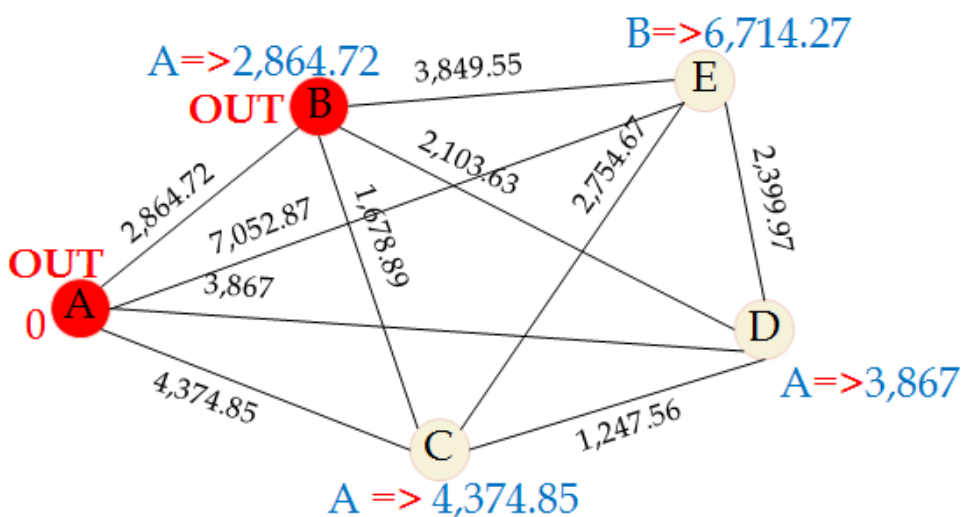
จะเห็นว่า ค่าน้ำหนักที่ Node E จะเปลี่ยนแปลง ทำให้มีค่าเท่ากับ 6,714.27

ดั่งภาพที่ 3-32



ภาพที่ 3-32 กราฟ Dijkstra Algorithm เมื่อ Node B พิจารณา Node E แล้ว

เมื่อพิจารณาเพื่อนบ้านที่ใกล้ Node B ครบทุก Node แล้ว จะทำให้ Node B กลายเป็น Super Node ดั่งภาพที่ 3-33



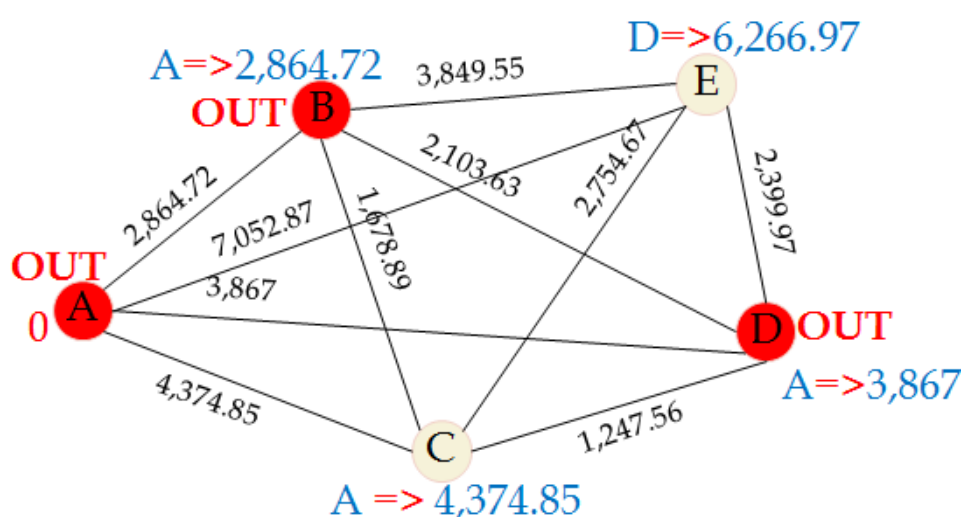
ภาพที่ 3-33 กราฟ Dijkstra Algorithm เมื่อ Node B เป็น Super Node

หลังจากนั้นทำการพิจารณาเพื่อนบ้านที่ใกล้ Node D ซึ่งวิธีการจะคล้ายกับวิธีที่กล่าวมาดังภาพที่ 3-14 ถึงภาพที่ 3-33

จะได้ว่า เมื่อ Node D พิจารณาที่ Node C ค่าน้ำหนักจาก Node D ไปยัง Node C มีค่ามากกว่าค่าน้ำหนักที่ Node C ทำให้ค่าน้ำหนักที่ Node C จะมีค่าน้ำหนักเท่าเดิม ไม่เปลี่ยนแปลง

แต่เมื่อ Node D พิจารณาที่ Node E ค่าน้ำหนักจาก Node D ไปยัง Node E มีค่าน้อยกว่าค่าน้ำหนักที่ Node E ทำให้ค่าน้ำหนักที่ Node E จะมีค่าเปลี่ยนแปลง มีค่าเท่ากับ 6,266.97

และเมื่อพิจารณาครบทุก Node แล้ว จะทำให้ Node D กลายเป็น Super Node ดังภาพที่ 3-34

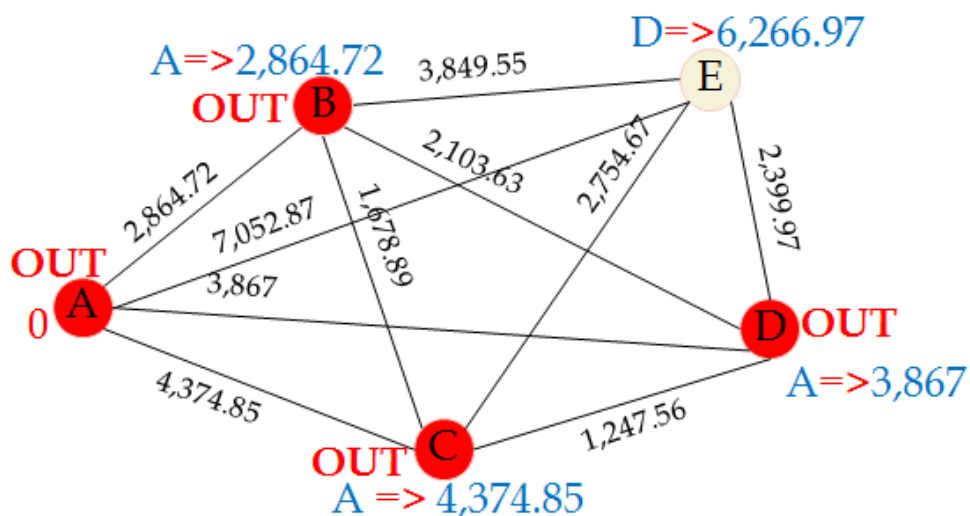


ภาพที่ 3-34 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ D แล้ว

หลังจากนั้นทำการพิจารณาเพื่อนบ้านที่ใกล้ Node C ซึ่งวิธีการจะคล้ายกับวิธีที่กล่าวมาดังภาพที่ 3-14 ถึงภาพที่ 3-33

จะได้ว่า เมื่อ Node C พิจารณาที่ Node E ค่าน้ำหนักจาก Node C ไปยัง Node E มีค่ามากกว่าค่าน้ำหนักที่ Node E ทำให้ค่าน้ำหนักที่ Node E จะมีค่าน้ำหนักเท่าเดิม ไม่เปลี่ยนแปลง

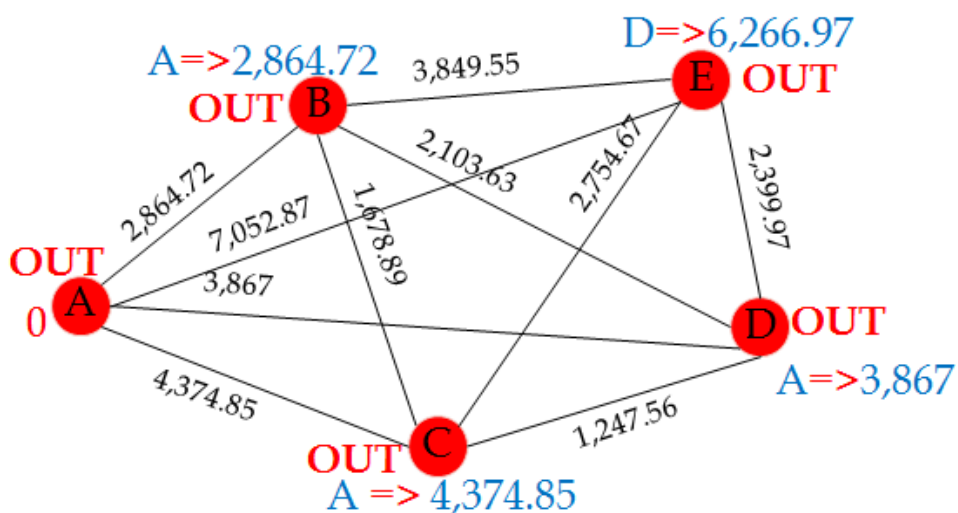
และเมื่อพิจารณาครบทุก Node แล้ว จะทำให้ Node C กลายเป็น Super Node ดังภาพที่ 3-35



ภาพที่ 3-35 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ C แล้ว

หลังจากนั้นทำการพิจารณาเพื่อนบ้านที่ใกล้ Node E แต่ Node E เป็น Node สุดท้ายในกราฟ จึงไม่มี Node เพื่อนบ้านให้พิจารณา

จะทำให้ Node E กลายเป็น Super Node ดังภาพที่ 3-36

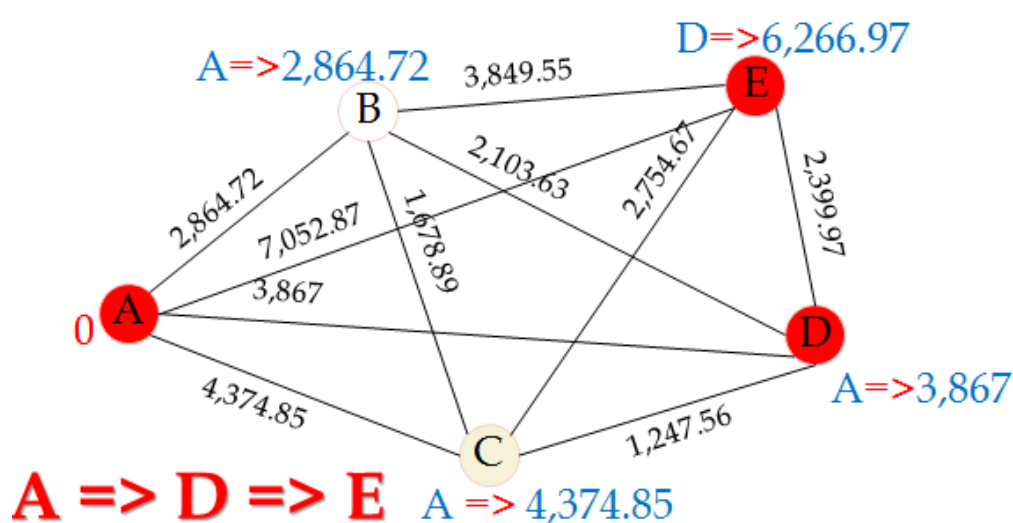


ภาพที่ 3-36 กราฟ Dijkstra Algorithm เมื่อตรวจสอบเพื่อนบ้านที่ใกล้ E แล้ว

เมื่อพิจารณาทุก Node ในกราฟเรียบร้อยแล้ว จะสามารถเลือกเส้นทางที่สั้นที่สุดได้ โดยพิจารณาจาก Node ปลายทางที่เราต้องการไปยัง Node เริ่มต้น

จากตัวอย่าง กำหนดให้ Node E เป็น Node ปลายทาง แล้วเส้นทางที่สั้นที่สุดที่มาถึง Node E มาจาก Node D และเส้นทางที่สั้นที่สุดที่มาถึง Node D มาจาก Node A ซงเป็น Node เริ่มต้น ทำให้ได้เส้นทางที่สั้นที่สุดจาก Node A ไปยัง Node E เป็นเส้นทางดังนี้

คือ $A \Rightarrow D \Rightarrow E$ ดังภาพที่ 3-37



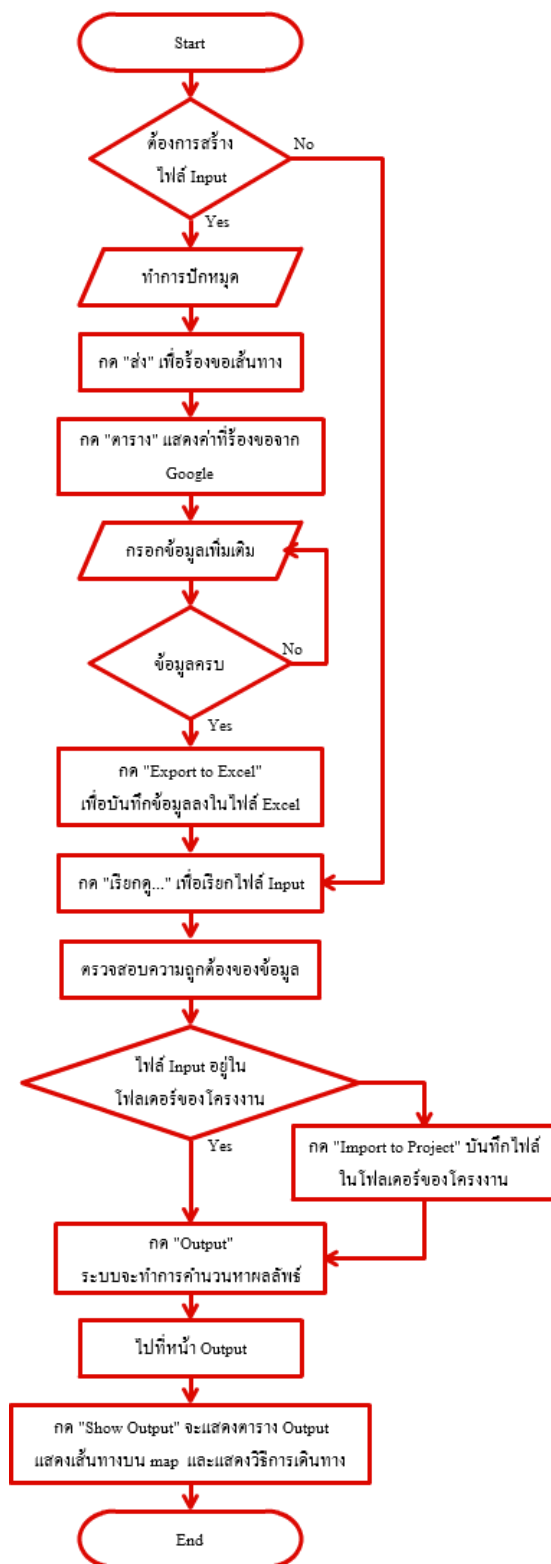
ภาพที่ 3-37 กราฟ Dijkstra Algorithm และผลลัพธ์ของปัญหา

3.2.3.1.2 จากการตรวจสอบจะเห็นได้ว่าผลลัพธ์ของ Dijkstra Algorithm ที่คำนวณด้วยตัวเอง กับผลลัพธ์จากโปรแกรมนั้นเหมือนกัน คือจะได้เส้นทางจาก Node A ไปยัง Node E ดังนี้

$A \Rightarrow D \Rightarrow E$

จึงสรุปได้ว่าการทำงานของโปรแกรมนั้นถูกต้อง

3.3 Diagram ของระบบ



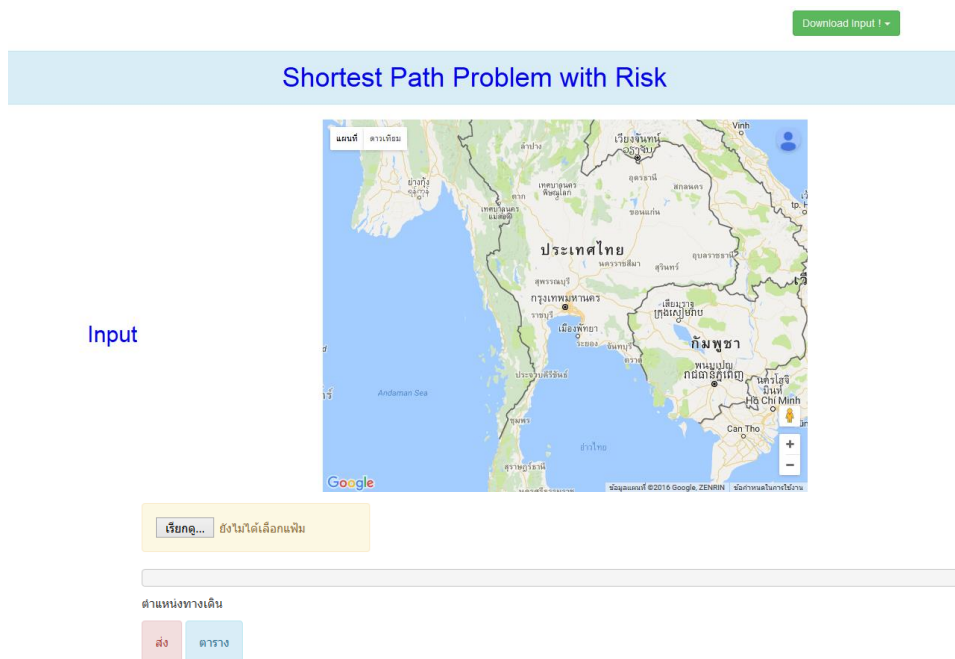
ภาพที่ 3-38 การทำงานของระบบ

บทที่ 4

ผลการทดลอง

4.1 การทดสอบระบบ

4.1.1 เริ่มทดสอบด้วยการเรียนหน้าเว็บไซต์ขึ้นมาจะแสดง map และปุ่มต่าง ๆ



ภาพที่ 4-1 หน้าเว็บไซต์ที่ทำการเรียก

4.1.1.1

Download Input !

ปุ่มสำหรับการดาวน์โหลดไฟล์ Input

4.1.1.2

เรียกดู...

ปุ่มสำหรับเรียกไฟล์ Input

4.1.1.3

ส่ง

ปุ่มสำหรับส่งค่า Latitude และ Longitude เพื่อร้องขอเส้นทาง

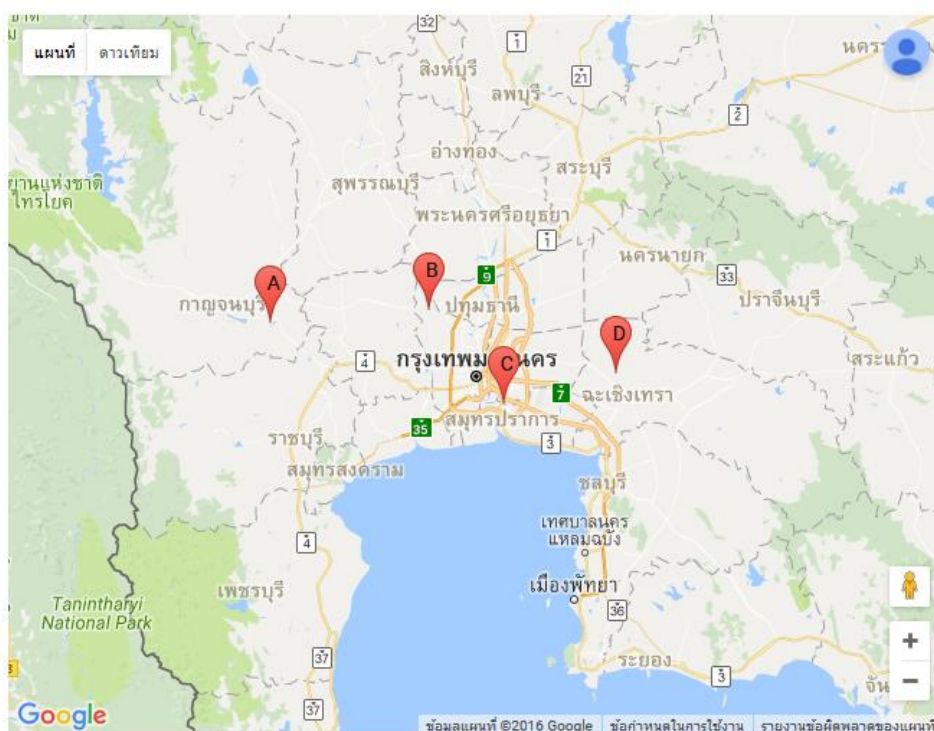
ระยะทาง เวลาการเดินทางจาก Google API

4.1.1.4

ตาราง

ปุ่มสำหรับเรียกตารางเพื่อทำการกรอกข้อมูลในไฟล์ Input

4.1.2 ปักจุดไปที่ map บนเว็บไซต์ โดยสมมติจุดต่างๆ เพื่อทำการทดสอบ



ภาพที่ 4-2 map บนหน้าเว็บไซต์ที่ทำการปักจุดแล้ว

ตำแหน่งทางเดิน

A Latitude: 13.955392274642316 Longitude: 99.71786487847567

B Latitude: 14.003366516945931 Longitude: 100.32211292535067

C Latitude: 13.656662778922 Longitude: 100.60775745660067

D Latitude: 13.768731154711025 Longitude: 101.03622425347567

ส่ง

ตาราง

ภาพที่ 4-3 ค่า Latitude และ Longitude ที่มาจากการปักหมุดบน map ในหน้าเว็บไซต์

4.1.3 หลังจากปักจุดเรียบร้อยแล้ว กดปุ่ม “ส่ง” แล้วทำการกดปุ่ม “ตาราง”

จะแสดงตารางเพื่อทำการกรอกข้อมูลให้ไฟล์ Input

เส้นทาง :

ต้นทาง	ปลายทาง	ระยะทาง	เวลาในการเดินทาง	ความเสี่ยง
A	B	81.5 กม.	1 ชั่วโมง 16 นาที	0
A	C	121 กม.	2 ชั่วโมง 4 นาที	0
A	D	175 กม.	2 ชั่วโมง 58 นาที	0
B	C	64.0 กม.	1 ชั่วโมง 6 นาที	0
B	D	97.1 กม.	1 ชั่วโมง 48 นาที	0
C	D	82.1 กม.	1 ชั่วโมง 16 นาที	0

ปัจจัยค่าใช้จ่าย(%)	ปัจจัยเวลา(%)	ปัจจัยความเสี่ยง(%)
0	0	0

*** บวกกันได้ 100 ***

Point
start
end

Export to Excel!

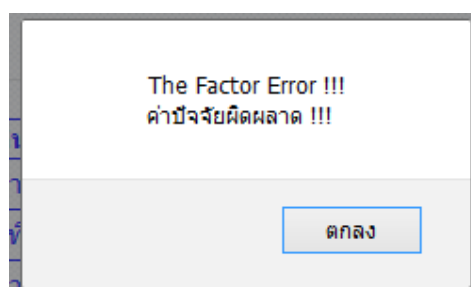
ภาพที่ 4-4 ตารางที่แสดงหลังจากกดปุ่มตาราง

4.1.3.1 กดปุ่ม “Export to Excel!” ปุ่มสำหรับบันทึกข้อมูลลงในไฟล์ Input

4.1.3.2 กรณีที่ทำการกรอกข้อมูลค่าปัจจัยผิด หรือลืมใส่ข้อมูลค่าปัจจัยจะมีการแจ้งเตือน

หลังจากทำการกดปุ่ม “Export to Excel!”

Export to Excel!

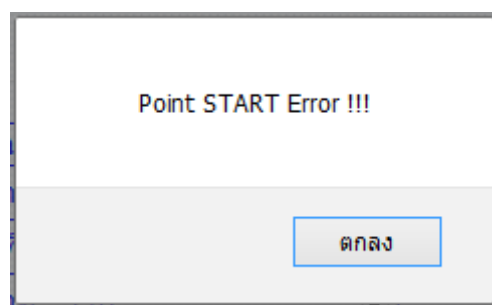


ภาพที่ 4-5 การแจ้งเตือนเมื่อค่าปัจจัยผิดพลาด

4.1.3.3 กรณีที่ทำการกรอกจุด start ผิด หรือลืมใส่จุด start จะมีการแจ้งเตือนหลังจากทำ

การกดปุ่ม “Export to Excel!”

Export to Excel!

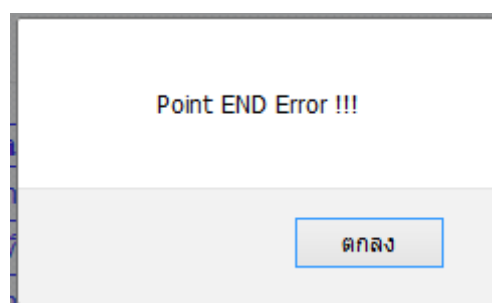


ภาพที่ 4-6 การแจ้งเตือนเมื่อจุด start ผิดพลาด

4.1.3.2 กรณีที่ทำการกรอกจุด end ผิด หรือลืมใส่จุด end จะมีการแจ้งเตือนหลังจากทำ

การกดปุ่ม “Export to Excel!”

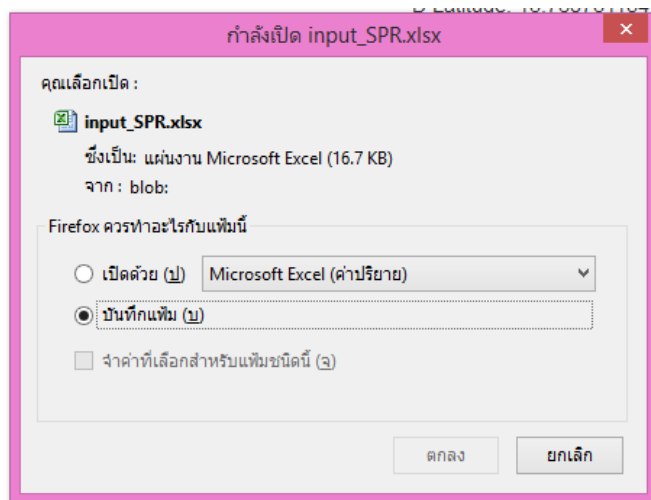
Export to Excel!



ภาพที่ 4-7 การแจ้งเตือนเมื่อจุด end ผิดพลาด

4.1.4 หลังจากการกดปุ่ม “Export to Excel!” จะทำการบันทึกข้อมูลที่อยู่ในตารางหน้าเว็บไซค์รวมถึงข้อมูลที่ทำกรอกลงในไฟล์ input_SPR.xlsx หลังจากนั้นให้ทำการบันทึกไฟล์

Export to Excel!



ภาพที่ 4-8 การบันทึกข้อมูลลงในไฟล์ input_SPR.xlsx

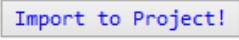
4.1.5 เมื่อทำการบันทึกไฟล์เรียบร้อยแล้ว ให้ทำการกดปุ่ม “เรียกดู...” เพื่อเรียกไฟล์ input_SPR.xlsx ที่ทำการบันทึกค่าไว้ขึ้นมาแสดงที่หน้าเว็บไซต์ และเพื่อตรวจสอบความถูกต้องของไฟล์ input_SPR.xlsx


เรียกดู... input_SPR.xlsx

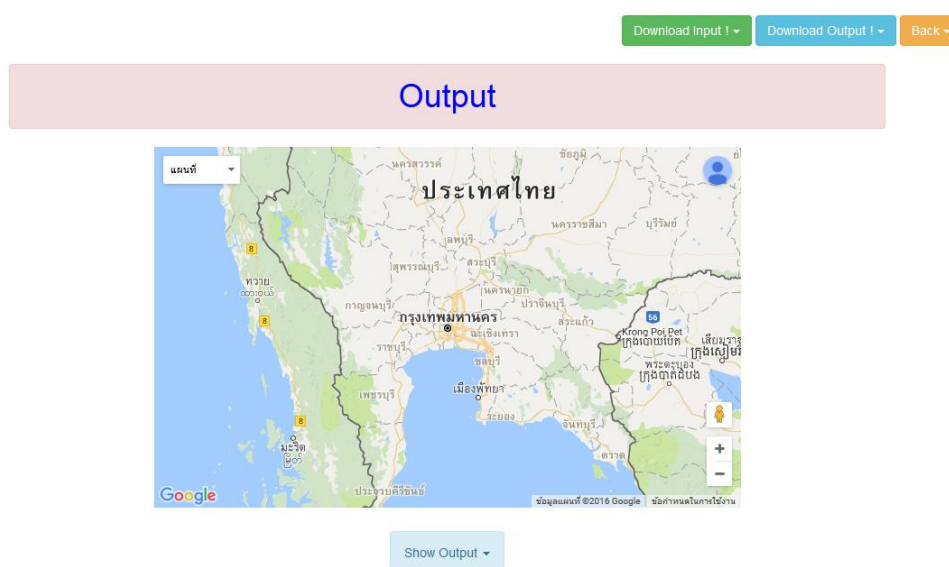
Table 1 The weight of the factors								
cost (%)	time (%)	risk (%)						
37	13	50						
start	13.95539227	99.71786488	finish	13.76873115	101.0362243			
Table 2								
start	Latitude	Longitude	end	Latitude	Longitude	distance(m)	time(s)	risk
A	13.9551759	99.7170403	B	14.0017624	100.3247903	81460	4587	32
A	13.9551759	99.7170403	C	13.6567373	100.6075564	121416	7449	56
A	13.9551759	99.7170403	D	13.7690102	101.038604	174753	10665	13
B	14.0017624	100.3247903	C	13.6567373	100.6075564	64003	3948	52
B	14.0017624	100.3247903	D	13.7690102	101.038604	97127	6454	24
C	13.6567373	100.6075564	D	13.7690102	101.038604	82104	4573	66

Import to Project!
Output




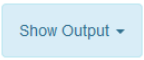
ภาพที่ 4-9 ข้อมูลในไฟล์ input_SPR.xlsx ที่เรียกมาแสดงบนเว็บไซต์

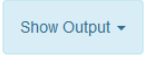
4.1.5.1 กรณีที่ไฟล์ input_SPR.xlsx ไม่ได้อยู่ในโฟลเดอร์ของโครงการต้องทำการกดปุ่ม “Import to Project!”  เพื่อบันทึกไฟล์ input_SPR.xlsx ใหม่ โดยต้องบันทึกไว้ในโฟลเดอร์เดียวกับโครงการ ถึงจะทำการคำนวณหาผลลัพธ์ได้

4.1.6 ทำการกดปุ่ม “Output”  จะทำการคำนวณหาผลลัพธ์ เมื่อคำนวณเสร็จแล้วจะแสดงหน้าเว็บไซต์ในส่วนของ Output

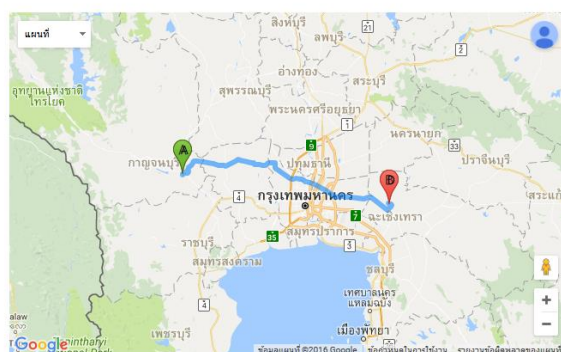


ภาพที่ 4-10 หน้าเว็บไซต์ในส่วนของ Output

- | | | |
|---------|---|--------------------------------------|
| 4.1.6.1 |  | ปุ่มสำหรับการดาวน์โหลดไฟล์ Input |
| 4.1.6.2 |  | ปุ่มสำหรับการดาวน์โหลดไฟล์ Output |
| 4.1.6.3 |  | ปุ่มย้อนกลับเพื่อไปหน้าที่ก่อน Input |
| 4.1.6.4 |  | ปุ่มแสดงผลลัพธ์ของการคำนวณ |

4.1.7 ทำการกดปุ่ม “Show Output”  จะมีการแสดงค่าของผลลัพธ์ที่ map แสดงตารางของผลลัพธ์ และแสดงเส้นทางการเดินทาง



Output



Show Output ▾

point	Latitude	Longitude
A	13.9551759	99.7170403
D	13.7690102	101.038604

ภาพที่ 4-11 หน้าเว็บแสดงเส้นทางและตารางของผลลัพธ์

 ชอย ห้วยหงษ์ 2 ตำบล รังศาลา อำเภอ ท่าม่วง กาญจนบุรี 71110 ประเทศไทย		
175 กม.. ประมาณ 2 ชั่วโมง 58 นาที		
1.	มุ่งหน้าทางใต้ ไปตามชอย ห้วยหงษ์ 2 เข้าสู่ถนนแสงชูโตสายเก่า	0.4 กม.
↱ 2.	เลี้ยวซ้าย เข้าสู่ ถนนแสงชูโตสายเก่า	4.3 กม.
↱ 3.	เลี้ยวซ้าย เข้าสู่ ถนนหมายเลข 3081	10.7 กม.
↱ 4.	เลี้ยวขวา เข้าสู่ ถนนหมายเลข 346	21.8 กม.
↱ 5.	เลี้ยวขวา เข้าสู่ ถนนหมายเลข 321/ถนนหมายเลข 346	0.4 กม.
↱ 6.	เลี้ยวซ้าย เข้าสู่ ถนนหมายเลข 346	22.0 กม.
↱ 7.	เลี้ยวซ้าย เพื่อวิ่งบน ถนนหมายเลข 346	15.5 กม.
↱ 8.	เลี้ยวขวา เข้าสู่ ถนน บางกรวย - กรุงเทพ/ถนนหมายเลข 3215 วิ่งต่อไปตามเส้นทาง ถนน บางกรวย - กรุงเทพ	18.2 กม.
↱ 9.	ชิดขวา เพื่อวิ่งต่อไปบน ถนน ชัยพฤกษ์/ถนน บางกรวย - กรุงเทพ	1.6 กม.
↱ 10.	ชิดขวา เพื่อวิ่งบน ถนน ชัยพฤกษ์/ถนน บางกรวย - กรุงเทพ	5.8 กม.
↱ 11.	ชิดขวา เพื่อวิ่งต่อไปบน ถนนหมายเลข 304	1.2 กม.
↱ 12.	ชิดขวา เพื่อวิ่งบน ถนนหมายเลข 304	27.1 กม.
↱ 13.	เลี้ยวซ้ายเล็กน้อย เพื่อวิ่งบน ถนนหมายเลข 304	9.7 กม.
↑ 14.	ขับตรงไปตลอดเพื่อวิ่งบน ถนนหมายเลข 304	0.6 กม.
↑ 15.	ขับตรงไปตลอดเพื่อวิ่งบน ถนนหมายเลข 304	3.7 กม.
↱ 16.	ชิดขวา เพื่อวิ่งบน ถนนหมายเลข 304	3.2 กม.
↑ 17.	ขับตรงไปตลอดเพื่อวิ่งบน ถนนหมายเลข 304	1.6 กม.
↑ 18.	ขับตรงไปตลอดเพื่อวิ่งบน ถนนหมายเลข 304	18.9 กม.
↱ 19.	เลี้ยวซ้าย เพื่อวิ่งบน ถนนหมายเลข 304	4.8 กม.
↱ 20.	เลี้ยวซ้าย	2.9 กม.
↱ 21.	เลี้ยวซ้าย	0.3 กม.
 Unnamed Road, ตำบล คลองนครเนื่องเขต อำเภอเมืองฉะเชิงเทรา ฉะเชิงเทรา		
ข้อมูลแผนที่ ©2016 Google		

ภาพที่ 4-12 หน้าเว็บแสดงเส้นทางในการเดินทาง

บทที่ 5

สรุป วิจัยรณผล และข้อเสนอแนะ

จากการศึกษาค้นคว้ารวบรวมข้อมูลและทำการทดลองเขียนโปรแกรมในแต่ละส่วนดังที่กล่าวมาข้างต้น การพัฒนาในส่วนย่อยสามารถพัฒนาได้ตามต้องการ คือ สามารถใช้งาน Google Maps ในการปักหมุดและร้องขอข้อมูลของเส้นทางได้ สามารถอ่านและเขียนไฟล์ Excel ผ่านทางหน้าเว็บไซต์ได้ สามารถพัฒนาส่วนของการคำนวณที่ใช้ภาษา Java ให้ทำงานตามหลักของ Dijkstra Algorithm ได้ แต่เมื่อต้องนำส่วนต่าง ๆ มารวมกันเพื่อให้ทำงานสอดคล้องกัน ยังต้องมีการปรับปรุงแก้ไขในการรับส่งข้อมูลเพื่อให้ส่วนต่าง ๆ ทำงานร่วมกันแต่ก็สามารถแก้ไขปัญหาได้ ทำให้ระบบทั้งหมดนั้นทำงานร่วมกันได้อย่างมีประสิทธิภาพ และตรงตามจุดประสงค์ที่ตั้งไว้

เพื่อประสิทธิภาพที่มากขึ้น ควรจะพัฒนาระบบให้ใช้งานได้ง่ายยิ่งขึ้นด้วยการให้ผู้ใช้งานหมุดแค่จุดเริ่มต้นและจุดปลายทางเท่านั้น ในส่วนของการปักหมุดจุดต่าง ๆ ระหว่างทางควรเป็นระบบที่ทำหน้าที่นี้

เอกสารอ้างอิง

1. บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ. คู่มือการทำวิทยานิพนธ์. กรุงเทพฯ : 21 เซ็นจูรี, 2554.
2. Ninenik Narkdee. Google map API v.3 กับ jQuery ลากจุดหา พิกัด ค่า latitude และ longitude (ออนไลน์). สืบค้นจาก : Google map API v.3 กับ jQuery ลากจุดหา พิกัด ค่า latitude และ longitude. สืบค้นจาก http://www.ninenik.com/content.php?arti_id=326 via @ninenik, 2558.
3. Gendreau, Michel., Laporte, Gilbert., Tas, Duygu., Jabali, Ola. (29 กรกฎาคม 2558). The travelling salesman problem with time-dependence service time. International Federation of Operational Research Societies (IFORS). 10-11.
4. Prachuab Ratana. Google Maps. สืบค้นจาก : <http://slideplayer.in.th/slide/2069464/>, 2558.
5. Google. Google Maps APIs(ออนไลน์). สืบค้นจาก : <https://developers.google.com/maps/documentation/javascript/>, 2558.
6. Edsger W. Dijkstra. Dijkstra's algorithm. สืบค้นจาก : https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm, 2558.
7. นายไชยยศ ไชยมั่นคง, และดร.มยุขพันธ์ ไชยมั่นคง. (2557). กลยุทธ์โลจิสติกส์และซัพพลายเชน เพื่อการแข่งขันในตลาดโลก (พิมพ์ครั้งที่ 8). นนทบุรี: วิชั่น พรีเมส จำกัด.

ประวัติผู้แต่ง

ปริญญานิพนธ์เรื่อง : โปรแกรมเพื่อช่วยในการแก้ปัญหาคำหาเส้นทางที่สั้นที่สุดที่มีความเสี่ยง

สาขา : วิศวกรรมคอมพิวเตอร์

ภาควิชา : วิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะ : วิศวกรรมศาสตร์

ชื่อ : นางสาวเจนจิรา แผ่นทอง

ประวัติ

เกิดวันที่ 26 กรกฎาคม พ.ศ. 2537 อยู่บ้านเลขที่ 70/1 หมู่ 2 ตำบลลาดทิพรส อำเภอตาคลี จังหวัดนครสวรรค์ สำเร็จการศึกษาในระดับประกาศนียบัตรวิชาชีพ สาขาไฟฟ้าและอิเล็กทรอนิกส์ โรงเรียนเตรียมวิศวกรรมศาสตร์ วิทยาลัยเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2554 และสำเร็จการศึกษาในระดับปริญญาตรี สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2558