**บทที่ 4**

**ผลการดำเนินโครงงาน**

จากการออกแบบโปรแกรมแสดงผลข้อมูลประเภทกราฟ และได้นำมาพัฒนา จนได้เป็นโปรแกรมแสดงผลข้อมูลประเภทกราฟซึ่งมีส่วนประกอบหลักๆ ที่สามารถนำมาแสดงผลให้เห็นได้ดังนี้

4.1 ส่วนแสดงผลกราฟ

4.2 ส่วนจัดการกับกราฟ

4.3 ส่วนดำเนินการกับไฟล์

4.4 ส่วนอัลกอริทึม

**4.1 ส่วนแสดงผลกราฟ**

ส่วนประกอบหลักสำคัญที่สุดของโปรแกรมคือส่วนของการแสดงผลกราฟ จัดอยู่ที่ส่วนแสดงผลหลักของโปรแกรมเพื่อรองรับการจัดการกับกราฟต่างๆ ซึ่งมีความาสามารถและข้อจำกัดดังนี้

1. สามารถใช้เมาส์ในการจัดวางตำแหน่งของจุดตัดของกราฟได้

2. สามารถกำหนดจุดตัดที่เน้นหรือเส้นเชื่อมที่เน้นได้

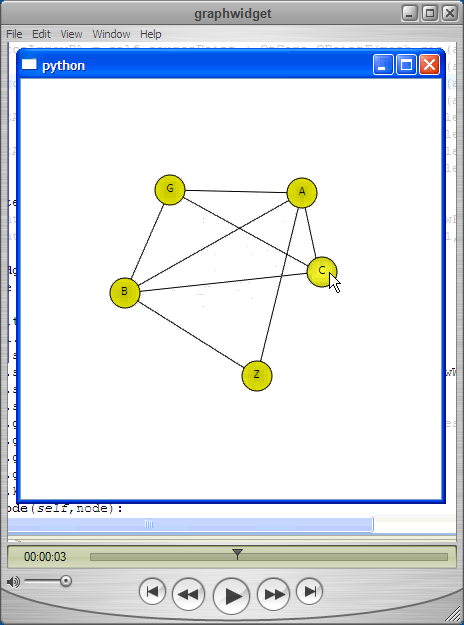
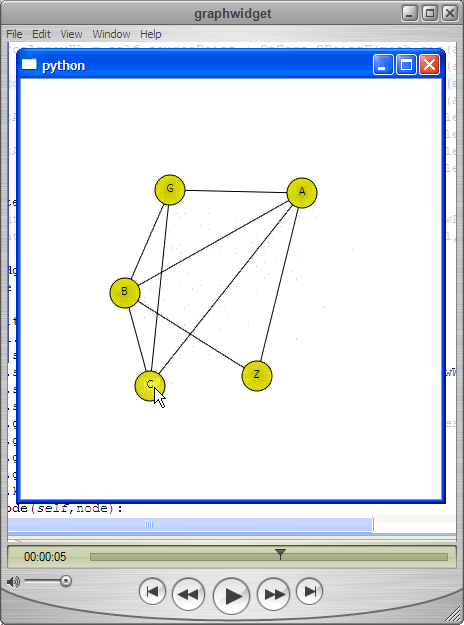
3. สามารถแสดงแบบย่อ/ขยายได้ (Zoom)

4. รูปร่างของจุดตัดและเส้นเชื่อมไม่สามารถเปลี่ยนแปลงได้

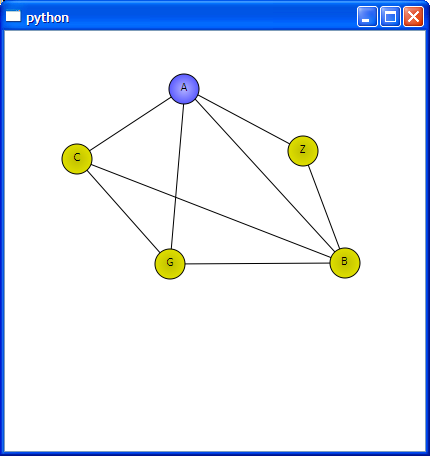
5. จะถูกแทนที่ถ้ามีการโหลดกราฟใหม่เข้ามาแทนที่

6. ชื่อที่กำหนดลงในแต่ละจุดตัดกำหนด จำกัดความยาวเพียง 3 ตัวอักษร

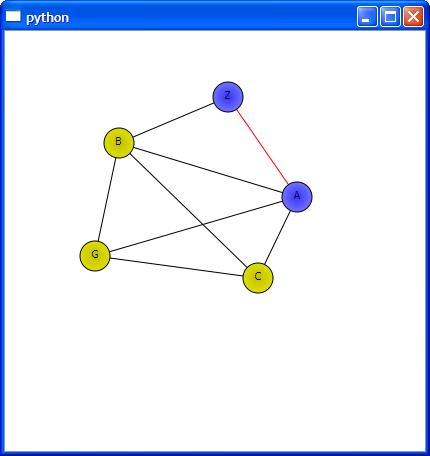
7. สีของจุดตัดและเส้นเชื่อมถูกกำหนดไว้แล้ว

**รูปที่ 4-1** รูปการเลื่อนจุดตัดได้ของโปรแกรม (ในรูปคือการเลื่อนจุดตัด C)



**รูปที่ 4-2** รูปการเน้นจุดตัด (A)



**รูปที่ 4-3** รูปการเน้นเส้นเชื่อม (A-Z)

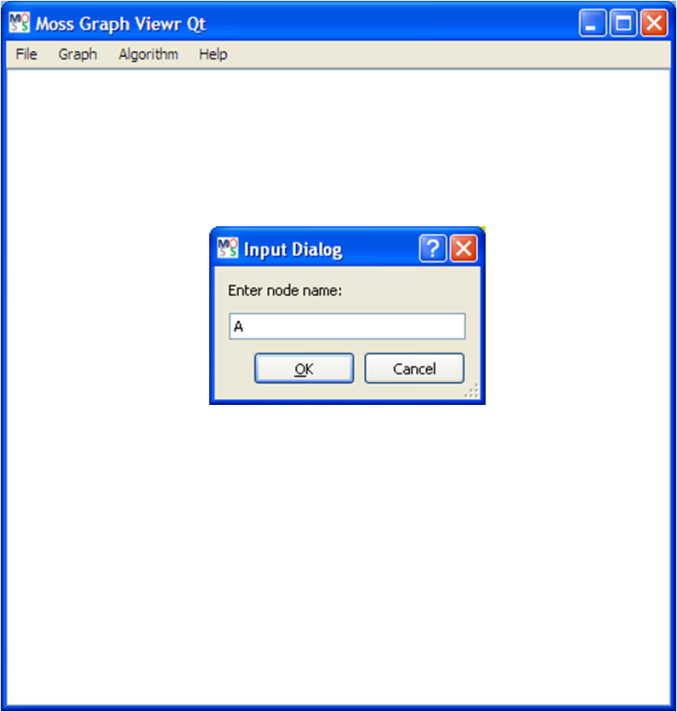
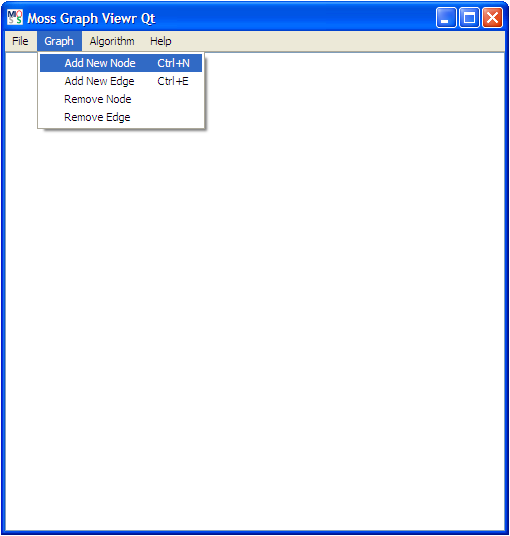
**4.2 จัดการกับกราฟ**

ส่วนที่ใช้ในการจัดการกับข้อมูลของกราฟเช่นการเพิ่มลดขนาดของกราฟ ทำให้มีความยืดหยุ่นในการปรับแก้หรือว่าทดสอบกราฟต่างๆ ผู้จัดทำได้จึงได้เพิ่มความในการจัดการกับกราฟขึ้นมา 4 อย่าง คือ

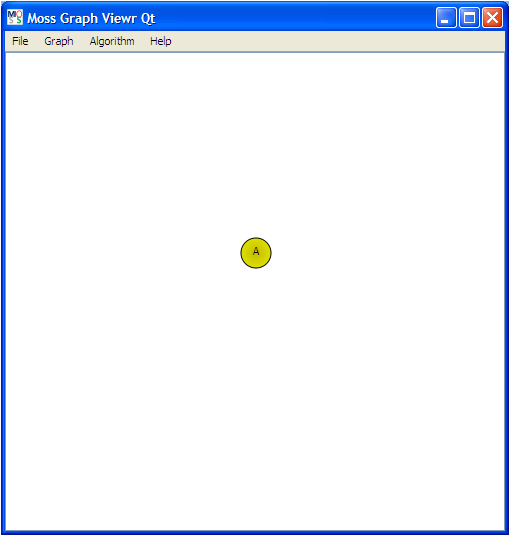
1. การเพิ่มจุดตัด
2. การเพิ่มเส้นเชื่อม
3. การลบจุดตัด
4. การลบเส้นเชื่อม

**4.2.1 การเพิ่มจุดตัด**

การเพิ่มจุดตัดหรือการเพิ่ม Node กำหนดให้ต้องมีการกำหนดชื่อเพื่อที่จะให้มีตัวกำหนดความแต่ต่างระหว่างแต่ล่ะจุดตัด ผู้พัฒนาได้สร้างวิธีในการเพิ่มจุดไว้สองแบบคือการใช้คีย์ลัดและเมนู



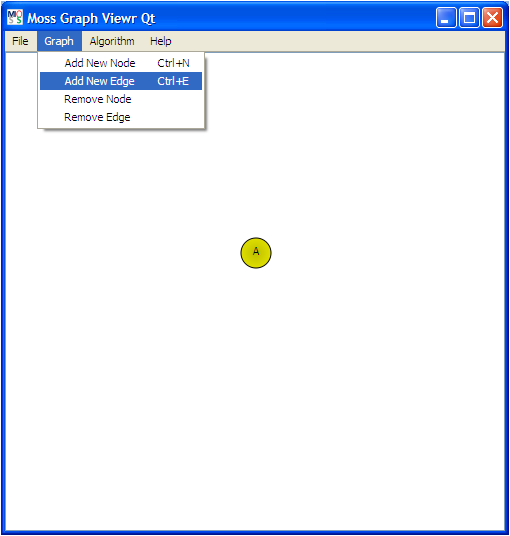
**รูปที่ 4-4** รูปตัวอย่างการเพิ่มจุดตัด (Node) ตามลำดับซ้ายขวา



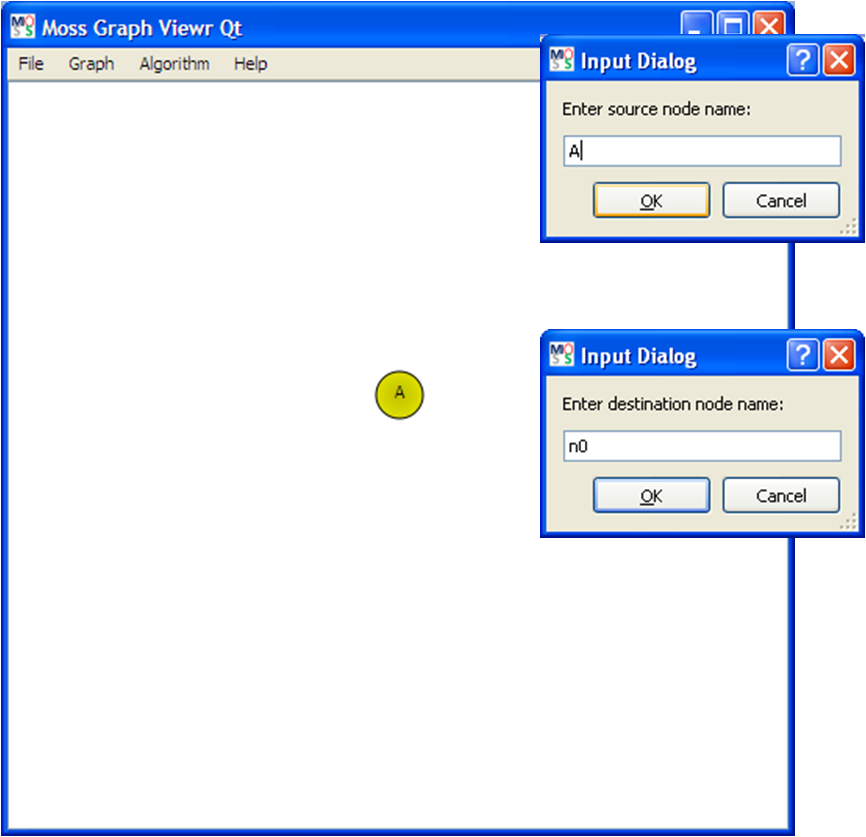
**รูปที่ 4-5** รูปตัวอย่างผลของการเพิ่มจุดตัด (Node)

**4.2.2 การเพิ่มเส้นเชื่อม**

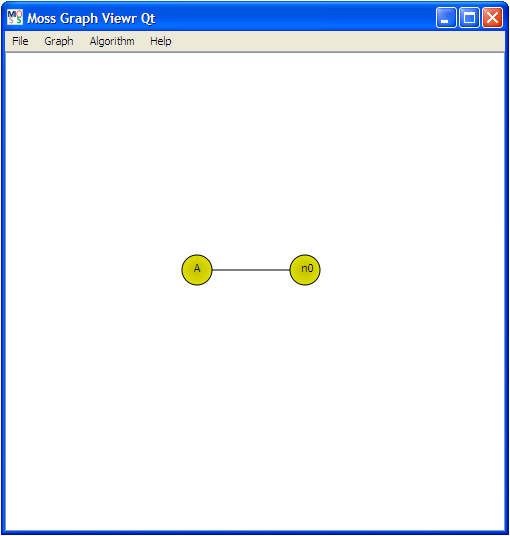
การเพิ่มเส้นเชื่อมหรือการเพิ่ม Edge กำหนดให้ต้องมีการกำหนดชื่อของสองจุดตัดที่จะใช้ในการเชื่อมต่อกัน ผู้พัฒนาได้สร้างวิธีในการเพิ่มจุดไว้สองแบบคือการใช้คีย์ลัดและเมนู เช่นเดียวกับการเพิ่มจุดตัด แต่ว่าถ้าเกิดมีการเพิ่มเส้นเชื่อมไปยังจุดตัดที่ยังไม่มีอยู่โปรแกรมจะสร้างจุดตัดนั้นให้โดยอัตโนมัติ



**รูปที่ 4-6** รูปตัวอย่างผลของการเลือกเมนูเพิ่มเส้นเชื่อม (Edge)



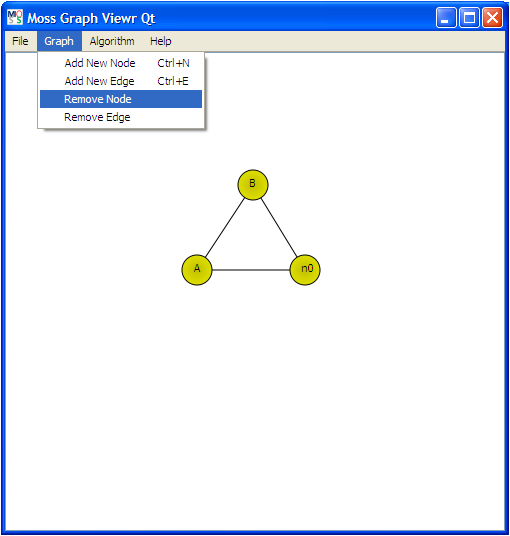
**รูปที่ 4-7** รูปตัวอย่างผลของการเพิ่มเส้นเชื่อม กรอกจุดเริ่มต้นและจุดปลาย



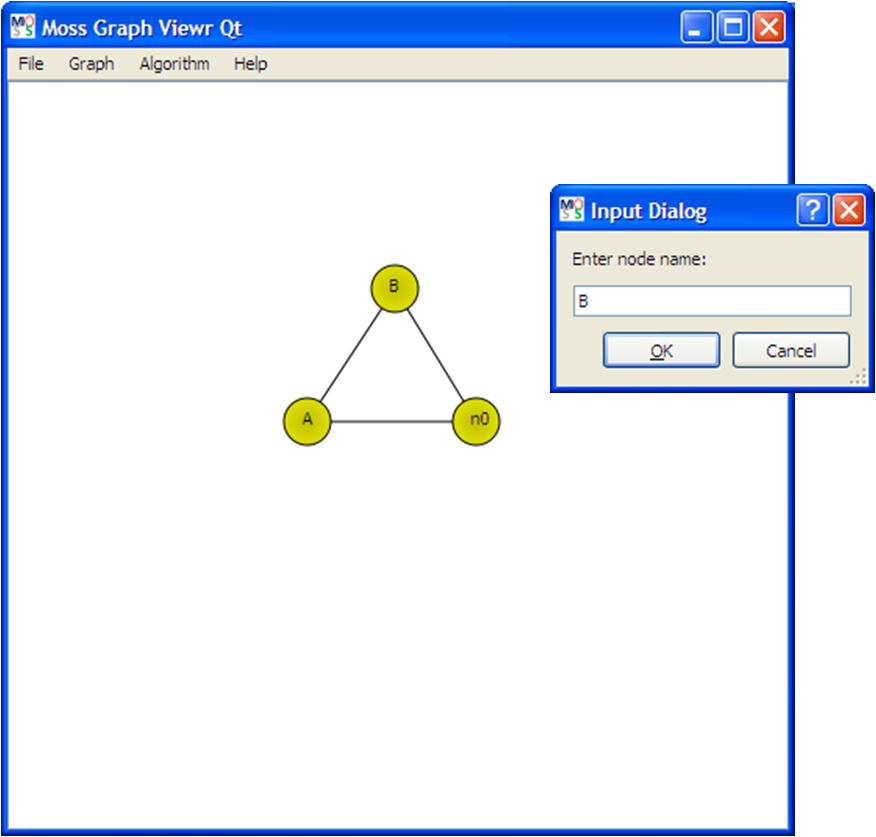
**รูปที่ 4-8** รูปตัวอย่างผลของการเพิ่มเส้นเชื่อม

**4.2.3 การลบจุดตัด**

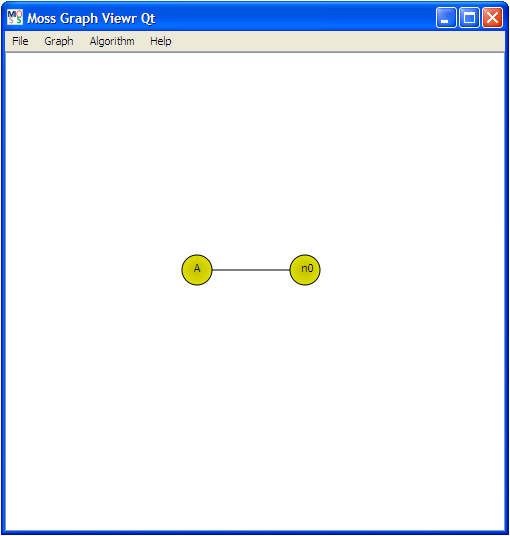
การลบจุดตัดจะต้องมีการกรอกชื่อของจุดตัดที่จะลบ เมื่อมีการลบจุดตัดออกจะต้องมีการพิจารณาว่าจุดตัดนั้นมีเส้นเชื่อมที่เชื่อมมาถึงมันหรือไม่ ถ้ามีเส้นเชื่อมที่เชื่อมมาถึงมัน ก็ต้องลบเส้นเชื่อมนั้นๆออกด้วย เพราะว่าเส้นเชื่อมแต่ล่ะเส้นเชื่อมจำเป็นต้องมีจุดปลายสองจุด



**รูปที่ 4-9** รูปตัวอย่างของการเลือกเมนูลบจุดตัด



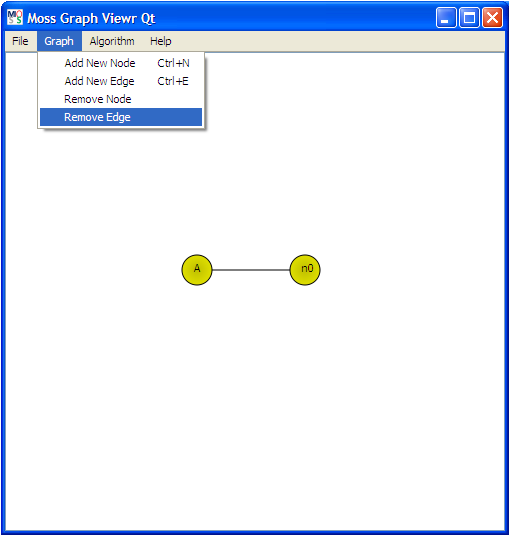
**รูปที่ 4-10** รูปตัวอย่างของการกรอกจุดตัดที่ต้องการลบ



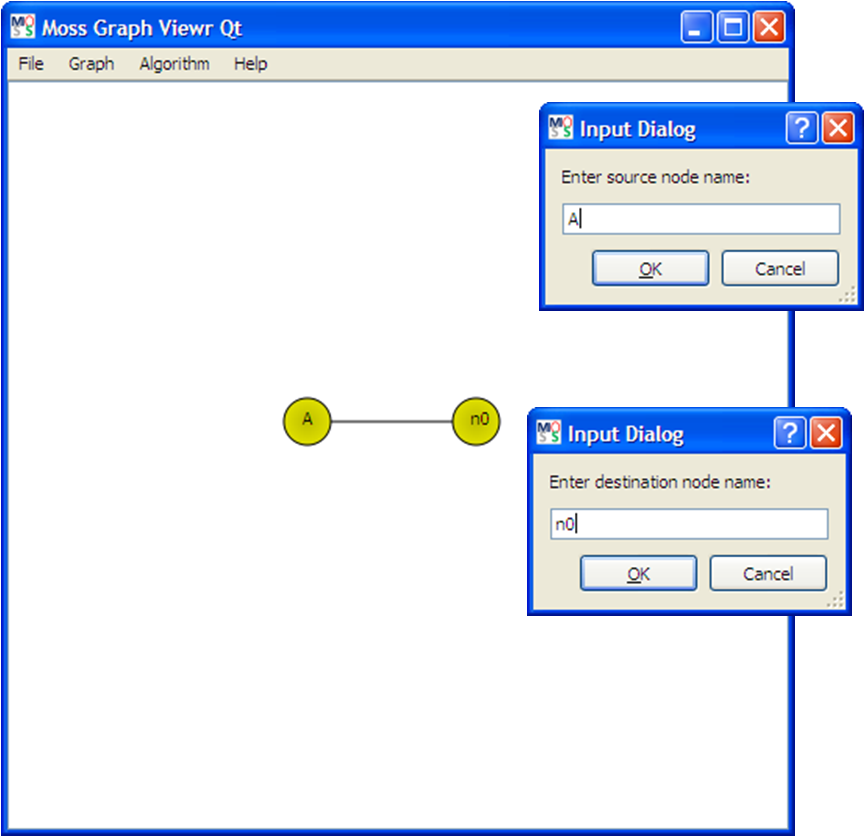
**รูปที่ 4-11** รูปตัวอย่างผลของการเลือกเมนูลบจุดตัด

**4.2.4 การลบจุดเส้นเชื่อม**

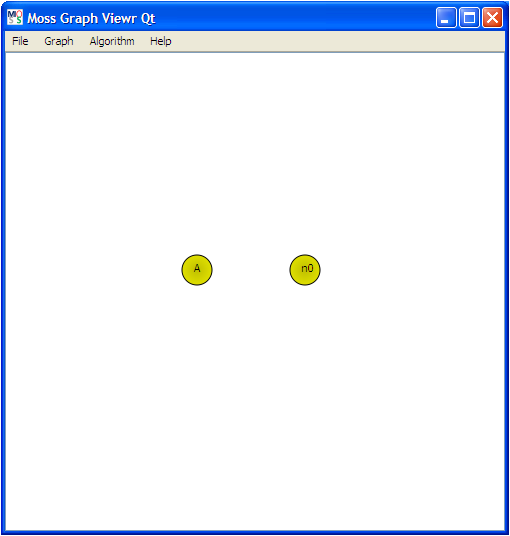
การลบเส้นเชื่อมจะต้องมีการกรอกชื่อของจุดปลายทั้งสองของเส้นเชื่อมที่จะลบ แต่ว่าการเส้นเชื่อมนั้นจะไม่มีผลใดๆ กับจุดตัดใดๆ ทั้งสิ้น



**รูปที่ 4-12** รูปตัวอย่างผลของการเลือกเมนูลบเส้นเชื่อม



**รูปที่ 4-13** รูปตัวอย่างผลของกรอกข้อมูลจุดปลายเพื่อลบเส้นเชื่อม



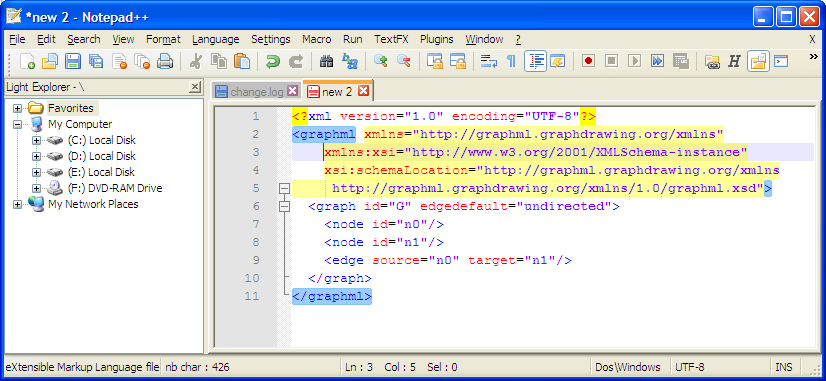
**รูปที่ 4-14** รูปตัวอย่างผลของการเลือกเมนูลบจุดตัด

**4.3 ส่วนดำเนินการกับไฟล์**

ส่วนที่ใช้ในการดำเนินการกับไฟล์เพื่อให้สามารถเก็บข้อมูลกราฟที่สร้างขึ้นมา หรือว่าเปิดอ่านข้อมูลกราฟที่ได้มาจากโปรแกรมอื่นที่ใช้ มาตรฐานเดียวกันในการเก็บข้อมูล ขึ้นมาแสดงผลได้

**4.3.1 การอ่านข้อมูลจากไฟล์**

ไฟล์ที่ใช้ในการเก็บข้อมูลของโปรแกรมนี้ใช้รูปแบบของ GraphML ซึ่งเป็นมาตรฐาน XML ตัวหนึ่งซึ่งใช้ในการเก็บข้อมูลประเภทกราฟโดยเฉพาะซึ่งมีโครงสร้างหลักๆ ดังนี้



**รูปที่ 4-15** รูปตัวอย่างไฟล์ GraphML

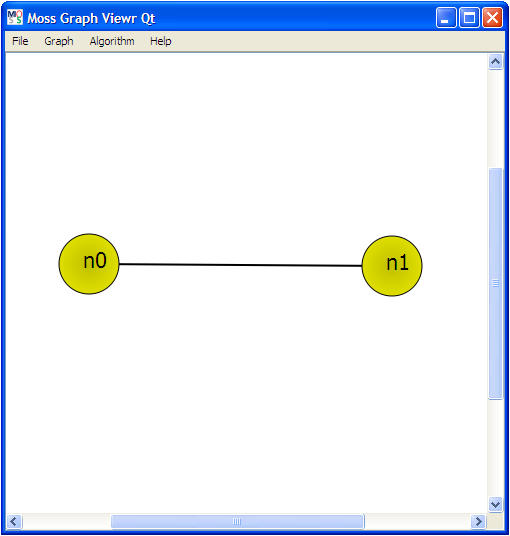
จะเห็นได้ว่าจะมีการประกาศ Namespace โดยแท็ก GraphML แต่ว่าโครงสร้างของกราฟจริงๆ นั้นเริ่มขึ้นที่แท็ก graph ซึ่งเป็นเสมือนเป็นจุดเริ่มต้นของข้อมูลกราฟซึ่งจะกำหนดด้วยว่ากราฟจะมีลักษณะเป็นแบบได้ เช่นมีทิศทาง (directed graph) หรือว่าเป็นกราฟที่ไม่มีทิศทาง (undirected graph) ซึ่งในโปรแกรมรุ่นปัจจุบันจะยังไม่พิจารณาเรื่องทิศทางของกราฟ

ส่วนที่กำหนดโครงสร้างของกราฟคือแท็ก node และ edge โดยที่แท็ก node จะต้องกำหนด id เสมอซึ่งสามารถที่จะกำหนดอย่างอื่นเพิ่มเติมภายในแท็กนั้นๆ ได้ตาม ความต้องการของโปรแกรมต่างๆ ที่อาจจะมีส่วนเสริมเพิ่มเติมในเรื่องนี้ ดูตัวอย่างได้ที่รูปที่ 4-16



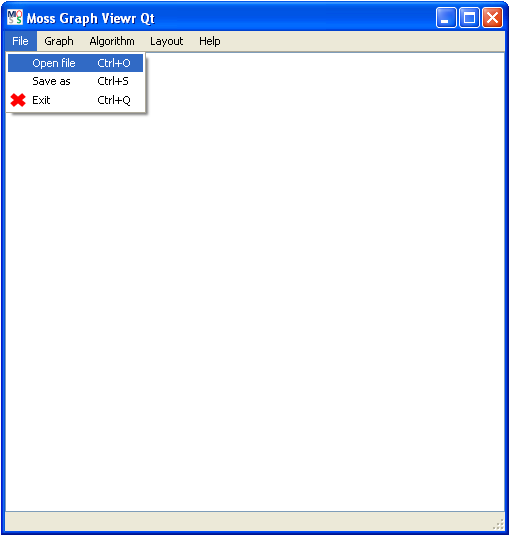
**รูปที่ 4-16** ตัวอย่างไฟล์ GraphML ที่ใช้กับโปรแกรม yWork ที่เพิ่มส่วนเสริมเข้าไป (ในกรอบคือส่วนเสริม)

แท็ก edge กำหนดการเชื่อมโยงด้วย แอตทริบิวต์ source และ target เสมอ แม้ว่าจะเป็นข้อมูลกราฟแบบ ไม่มีทิศทางก็ตาม เพื่อป้องกันความสับสนของผู้ใช้ที่ต้องนำไปใช้กับกราฟหลายแบบ และข้อมูลของแอตทริบิวต์ของ source และ target ก็คือ id ของ node ซึ่งโปรแกรมในส่วนของการดำเนินการกับไฟล์ของโปรแกรมนี้จะใช้ id ของแต่ล่ะ node มาเป็นชื่อของจุดตัด ตัวอย่างการแสดงผลข้อมูลกราฟข้างบนด้วยโปรแกรมจะแสดงออกมาดังรูปที่ 17

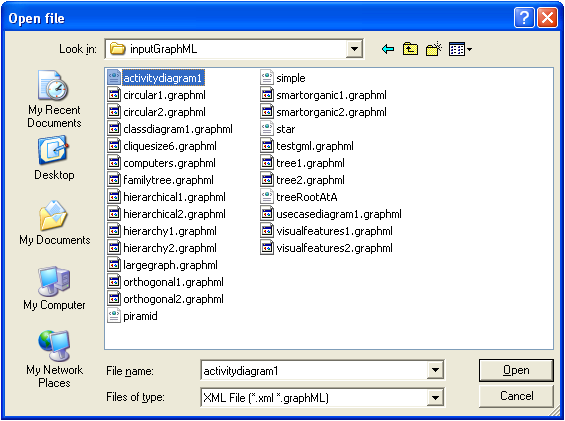


**รูปที่ 4-17** รูปตัวอย่างการแสดงผลข้อมูลกราฟจากไฟล์

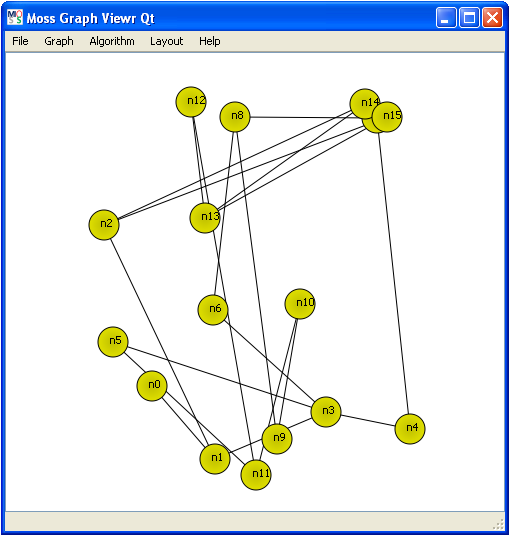
การเปิดหน้าต่างเลือกไฟล์เนื่องจาก PyQt มีความสามารถในการเรียกหน้าต่างเลือกไฟล์ของระบบปฏิบัติการ และสร้างตัวกรอกได้ทำให้ป้องกันปัญหาเรื่องการเปิดไฟล์ผิดรูปแบบได้ในระดับหนึ่ง ลำดับการเปิดเลือกไฟล์เพื่อนำมาแสดงผลดูได้ ในรูปที่ 4-18 ถึง 4-20 ตามลำดับ



**รูปที่ 4-18** รูปตัวอย่างการเลือกเมนูเปิดไฟล์



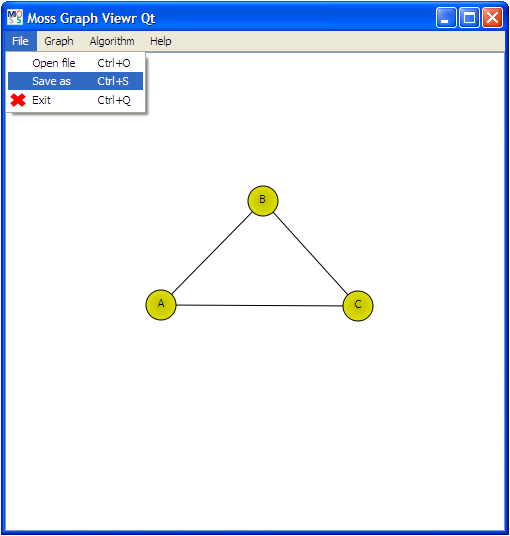
**รูปที่ 4-19** รูปตัวอย่างหน้าต่างเลือกเปิดไฟล์และตัวกรองไฟล์



**รูปที่ 4-20** รูปตัวอย่างไฟล์ที่ถูกเปิดขึ้นมาแสดงผล

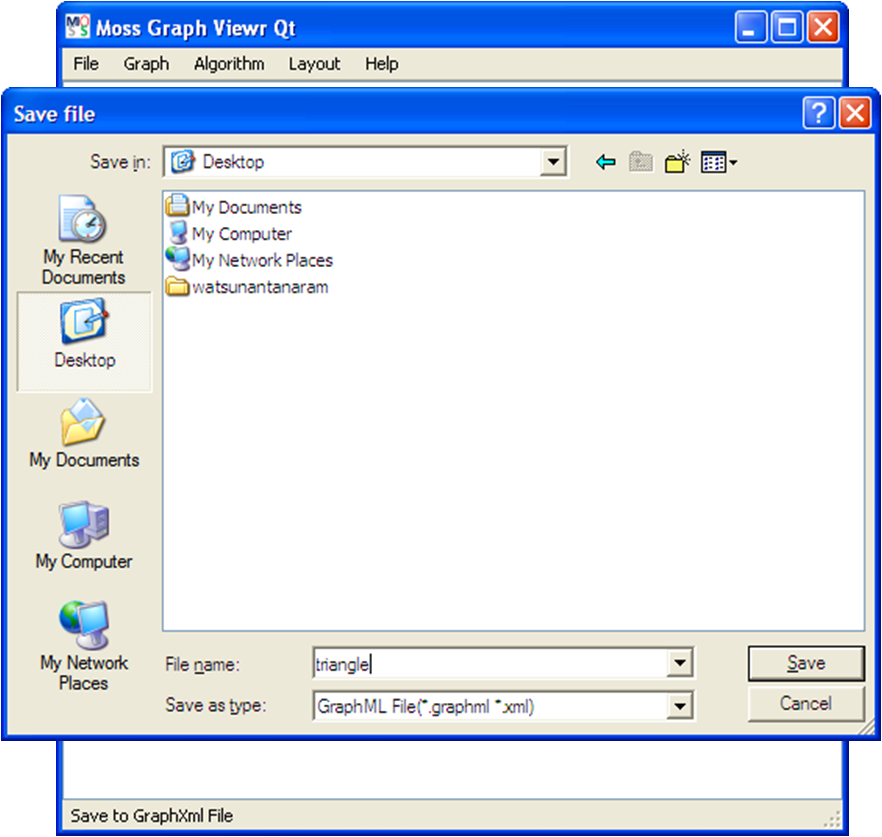
**4.3.2 การเก็บข้อมูล**

การที่จะเก็บข้อมูลของกราฟที่แก้ไขแล้วกลับไปเป็นไฟล์เพื่อที่จะสามารถนำไปใช้ต่อได้ในครั้งต่อไปจะต้องนำข้อมูลกราฟที่แสดงอยู่นั้นออกมาเป็นข้อมูลแล้วเขียนออกมาเป็นไฟล์ในรูปแบบของ GraphML โดยการเก็บไฟล์ของโปรแกรมนี้จะใช้ชื่อของจุดตัดแทนใน แอตทริบิวต์ id ของแท็ก node โปรแกรมจะไม่เก็บพิกัดของจุดหรือว่าสีที่เน้นไว้ แต่ว่าจะเก็บแต่โครงสร้างหลักๆ ของกราฟเท่านั้น รูปต่อไปนี้จะแสดงการบันทึกกราฟเป็นไฟล์ของโปรแกรม



**รูปที่ 4-20** รูปการเลือกเมนูบันทึกไฟล์

จากรูปมีการสร้างกราฟขนาด 3 จุดตัดขึ้นมา และต้องการที่จะบันทึกเป็นไฟล์เก็บไว้ ซึ่งในส่วนของการบันทึกไฟล์ผู้จัดทำก็ได้ทำตัวกรองไฟล์ไว้เช่นกัน นามสกุลของไฟล์โดยปริยายกำหนดให้เป็น GraphML ซึ่งผู้ใช้สามารถเลือกที่จะบันทึกเป็น xml ก็ได้แต่ว่าภายในไฟล์จะมีเนื้อหาเหมือนกัน



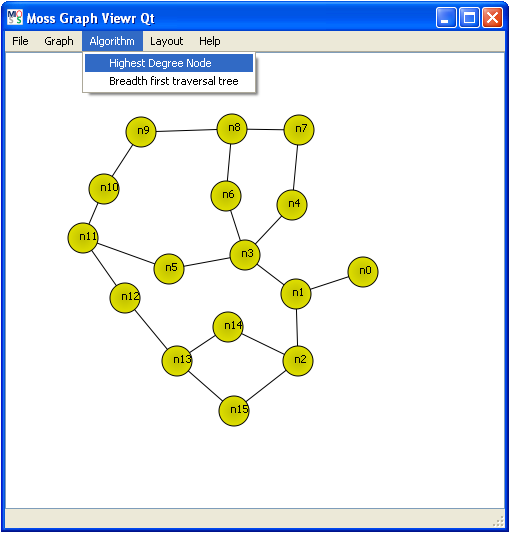
**รูปที่ 4-21** รูปหน้าต่างบันทึกไฟล์

เนื้อไฟล์ที่ได้จากการบันทึกกราฟข้างต้นจะได้ออกมาดังนี้

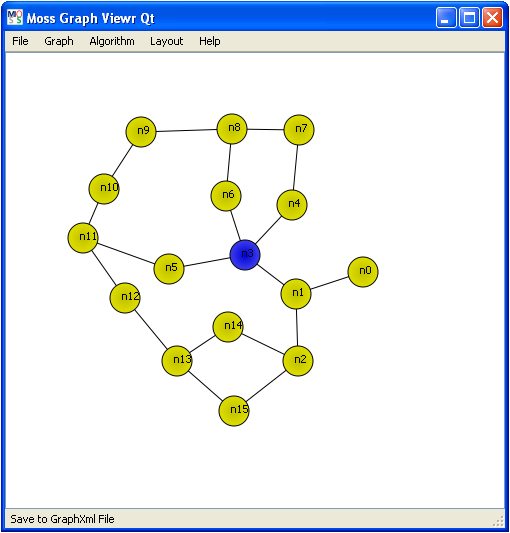
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <graphml xmlns="http://graphml.graphdrawing.org/xmlns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">  <graph id="G" edgedefault="undirected">  <node id="A"/>  <node id="C"/>  <node id="B"/>  <edge source="A" target="B"/>  <edge source="C" target="B"/>  <edge source="A" target="C"/>  </graph>  </graphml> |

**4.4 ส่วนอัลกอริทึม**

ส่วนประกอบที่ทำให้เราสามารถนำกราฟอัลกอริทึมต่างๆ มากระทำกับกราฟแล้วทำให้มันแสดงผลออกมาในรูปแบบกราฟิกได้ โดยการที่เราจะเขียนโปรแกรมคำนวณเกี่ยวกับอัลกอริทึมแยกเป็นอีกส่วนหนึ่ง ของโปรแกรมเมื่อเรียกใช้ก็จะทำการส่งกราฟที่เป็นข้อมูลไปคำนวณและเมื่ออัลกอริทึม ทำงานเสร็จก็จะส่งผลการทำงานกลับมาซึ่งถ้าส่งกลับมาเป็นกราฟย่อยของกราฟที่ส่งไปข้อมูลไปก็จะทำการ เปลี่ยนสีกราฟที่เป็นผล หรือว่าจะแสดงผลเป็นแบบอื่นก็ขึ้นอยู่กับการทำงานในส่วนของแต่ล่ะอัลกอริทึมนั้นๆ แต่ว่าตอนนี้โปรแกรมสนับสนุนแค่การเน้นสีของกราฟเท่านั้น รูปที่ 4-22 และ รูปที่ 4-23 จะแสดงตัวอย่างการใช้อัลกอริทึมง่ายๆ อย่างการหาจุดตัดที่มีดีกรีสูงสุด



**รูปที่ 4-21** รูปกราฟต้นแบบก่อนการกระทำด้วยอัลกอริทึม



**รูปที่ 4-22** รูปผลลัพธ์ของการเลือกใช้อัลกอริทึมหาจุดตัดที่มีดีกรีสูงสุด

จากตัวอย่างข้างต้นคือ อัลกอริทึมหาจุดตัดที่มีดีกรีสูงสุดซึ่งไม่ต้องการข้อมูลอื่นนอกจากกราฟ แต่ว่าสำหรับในบางอัลกอริทึมแล้วต้องการข้อมูลเข้ามากกว่านั้นเช่น อัลกอริทึมในการหาต้นไม้ที่อยู่ในกราฟอย่าง Breadth first tree หรือว่า Breadth first search ที่ต้องการข้อมูลเข้าเป็น root ของต้นไม้และ node ที่ต้องการหา ดังนั้นจึงต้องมีการให้โปรแกรม รับข้อมูลเข้าที่เกี่ยวข้องก่อที่จะทำงานได้ รูปที่ 4-23 และ รูปที่ 4-24 จะแสดงการผลของการใช้อัลกอริทึมที่ต้องการข้อมูลเข้าอื่นๆ โดยปัจจุบันโปรแกรมใช้วิธีการรับข้อมูลเข้าจากกล่องกรอกข้อความ