บทที่ 4

การออกแบบและการสร้างโปรแกรม

ในบทนี้ จะกล่าวถึงการออกแบบโปรแกรมในส่วนที่สำคัญต่อการสร้างข่ายงานเครื่องแลกเปลี่ยนความร้อนตามที่ได้กล่าวมาในบทที่แล้ว และตัวอย่างการใช้โปรแกรมด้วย อย่างไรก็ตามเนื่องจากโปรแกรมการออกแบบข่ายงานนี้เขียนด้วยภาษาเชิงวัตถุ ดังนั้น จึงขอกล่าวความสำคัญของภาษาเชิงวัตถุก่อนที่จะกล่าวถึงการออกแบบโปรแกรมต่อไป

4.1 ภาษาเชิงวัตถุ

การเขียนโปรแกรมสมัยก่อนนั้น ผู้เขียนโปรแกรมจะใช้ภาษาเครื่องอันประกอบไปด้วยเลข 0 และ 1 เท่านั้น การพัฒนาโปรแกรมจะประสบปัญหาที่ยุ่งยากมากมาย ต่อมามีภาษาแอสแซมบลี ข้อดีคือสามารถทำงานได้รวดเร็ว และติดต่อกับระดับฮาร์ดแวร์ของเครื่องได้สะดวก แต่ก็ยังยากต่อผู้ใช้ทั่วไปที่จะเข้าใจได้ ต่อมาเริ่มเกิดภาษาชั้นสูงซึ่งใช้คำสั่งประกอบด้วยคำพูดหรือสัญลักษณ์ที่ใช้กันในชีวิตประจำวันและคนทั่วไปเข้าใจได้ง่ายขึ้น เช่น ภาษาฟอร์แทรน ซึ่งมีผู้นิยมใช้กันมากแต่ก็เหมาะกับงานเฉพาะด้านการคำนวณเท่านั้น ในที่สุดได้มีผู้พัฒนาภาษาที่ใช้งานได้คล่องตัวและกว้างขึ้นกว่าเดิม ตัวอย่างเช่น ภาษาเบสิก ปาสคาล และ ภาษา C เป็นต้น

ซึ่งประสบความสำเร็จมากและยังเป็นภาษาที่นิยมใช้กันจนถึงปัจจุบันนี้ ภาษาชั้นสูงเหล่านี้เป็นภาษาที่เน้นโครงสร้าง การออกแบบโปรแกรมจะเน้นไปที่วิธีการและหน้าที่ส่วนต่างๆ ของโปรแกรม (ฟังก์ชัน หรือ สับโปรแกรม) โดยจะแตกโปรแกรมขนาดใหญ่เป็นโปรแกรมขนาดเล็กๆ ที่จัดการได้ง่าย จากนั้นจึงค่อยมาคำนึงถึงตัวแปรที่ใช้ อันเป็นลักษณะการออกแบบโปรแกรมจากบนลงล่าง ถ้าโปรแกรมมีขนาดเล็ก การออกแบบโปรแกรมแบบนี้จะไม่มีปัญหาอันใด แต่เมื่อโปรแกรมมีขนาดใหญ่และซับซ้อนขึ้น การพัฒนาและการแก้ไขปรับปรุงโปรแกรมจะทำได้ลำบาก ประสิทธิภาพในการทำงานจะลดลงอย่างมาก

ต่อมาได้มีการปรับปรุงการเขียนโปรแกรมแนวใหม่ที่เรียกว่า การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Program หรือ OOP) การเขียนโปรแกรมแบบนี้ไม่เน้นอัลกอริธึม แต่จะให้ความสำคัญแก่ลักษณะข้อมูลและตัวแปรก่อน ข้อมูลจะถูกใช้เป็นหลักในการแบ่งโปรแกรมออกเป็นส่วนย่อยๆ การออกแบบโปรแกรมเชิงวัตถุจะเกี่ยวข้องกับการนิยามคลาส (Class) และวัตถุ (Object) เท่านั้น ซึ่งในส่วนการทำงานของโค้ดจะยังไม่ถูกคำนึงถึงเลยในการออกแบบ โดยเป้าหมายหลักของการพัฒนาโปรแกรมเชิงวัตถุ คือ

- ทำให้การพัฒนาโปรแกรมใช้เวลาสั้นลง และต้นทุนต่ำลงโดยการใช้คลาสที่สามารถนำกลับมาใช้ได้อีก และสร้างสับคลาส (Subclass) ขึ้นมาเพื่อใช้ในการแก้ปัญหา
- ทำให้ต้นทุนในการบำรุงรักษาโปรแกรมต่ำลง เพราะสามารถหาจุดที่ต้องการเปลี่ยนแปลงในโปรแกรมได้ และการเปลี่ยนแปลงไม่ทำให้เกิดผลกระทบไปยังภายนอกคลาสเลย

ระบบโปรแกรมเชิงวัตถุมีความน่าเชื่อถือมาก เพราะการประกอบส่วนต่างๆ ของโปรแกรมขึ้นเป็นระบบได้ทำในระดับสูงตั้งแต่การออกแบบ ซึ่งสามารถตรวจสอบได้ก่อนที่รายละเอียดส่วนใหญ่ในระดับต่ำจะถูกเขียนขึ้น สิ่งเหล่านี้จึงเป็นตัวพิสูจน์ความน่าเชื่อถือเป็นอย่างดี ต่อไปจะได้กล่าวถึงสิ่งสำคัญอันเป็นคุณสมบัติของโปรแกรมเชิงวัตถุ

4.1.1 คลาสและวัตถุ

คลาสเป็นข้อกำหนดที่ระบุรายละเอียดของข้อมูล (Type) ให้แก่วัตถุ ซึ่งวัตถุคือ ข้อมูลจริงที่ถูกสร้างขึ้นตามข้อกำหนดนั้น โดยทั่วไป นอกจากที่คลาสจะกำหนดลักษณะข้อมูลที่ใช้เพื่อสร้างเป็นตัววัตถุแล้ว ยังสามารถกำหนดวิธีปฏิบัติสำหรับดำเนินงานกับข้อมูลนั้นอีกด้วย ยกตัวอย่างเช่น ถ้าเราต้องการสร้างโปรแกรมวาดภาพกล่องสี่เหลี่ยมสามมิติขึ้นมา สิ่งแรกที่ต้องทำก็คือ จะต้องสร้างคลาสขึ้นมาคลาสหนึ่ง โดยคลาสนี้จะกำหนดรายละเอียดต่างๆ ที่จำเป็นต่อการวาดรูปไว้ ส่วนที่เป็นข้อมูลอาจประกอบไปด้วยข้อมูลของตำแหน่ง ความสูง ความกว้าง ความยาว สี ทิศทาง หรือมุมมอง เป็นต้น ส่วนข้อกำหนดที่เป็นการปฏิบัติจะประกอบไปด้วยวิธีการเคลื่อนที่ การเปลี่ยนขนาด การวาด หมุน เปลี่ยนสี และคัดลอกรูป เป็นต้น จากนั้นจึงจะสร้างโปรแกรมขึ้นมาเพื่อวาดรูปกล่อง โปรแกรมจะสร้างวัตถุที่มีลักษณะตามที่ได้กำหนดไว้ในคลาสขึ้นมา โดยวัตถุนั้นจะมีข้อมูลของรูปครบตามที่กำหนดไว้ในคลาส ถ้าต้องการวาดรูปกล่องสองรูปที่เป็นอิสระต่อกัน โปรแกรมจะสร้างวัตถุขึ้นมาสองตัวสำหรับกล่องแต่ละรูป โดยใช้นิยามคลาสเดียวกัน

4.1.2 เอนแคปซูเลชัน (Encapsulation)

คุณสมบัติเอนแคปซูเลชันของคลาสที่กำหนดให้วัตถุ คือ

1. กำหนดขอบเขตที่ชัดเจนให้แก่วัตถุ
2. กำหนดส่วนการติดต่อภายนอกกับวัตถุตัวอื่นๆ

หมายความว่า เราสามารถกำหนดให้ข้อมูลหรือฟังก์ชันในวัตถุหนึ่ง สามารถที่จะให้โปรแกรมภายนอกหรือวัตถุอื่นๆ เรียกใช้งานหรือไม่ก็ได้ ซึ่งจะป้องกันการใช้และแก้ไขข้อมูลที่ไม่ถูกต้อง

4.1.3 อินเฮอริแทนซ์ (Inheritance)

อินเฮอริแทนซ์ หมายถึง การที่เราสามารถสร้างคลาสใหม่โดยสืบทอดคุณสมบัติจากคลาสเก่า เช่น เราสร้างคลาสของจักรวาลเป็นคลาสเริ่มต้น ต่อมาสร้างคลาสใหม่เป็นคลาสเกี่ยวกับระบบสุริยะสืบต่อจากคลาสจักรวาล และในที่สุดอาจสร้างคลาสของดาวเคราะห์สืบต่อจากคลาสระบบสุริยะ เพื่อแบ่งรายละเอียดต่อไปอีก เป็นต้น

4.1.4 โพลีมอร์ฟิซึม (Polymorphism)

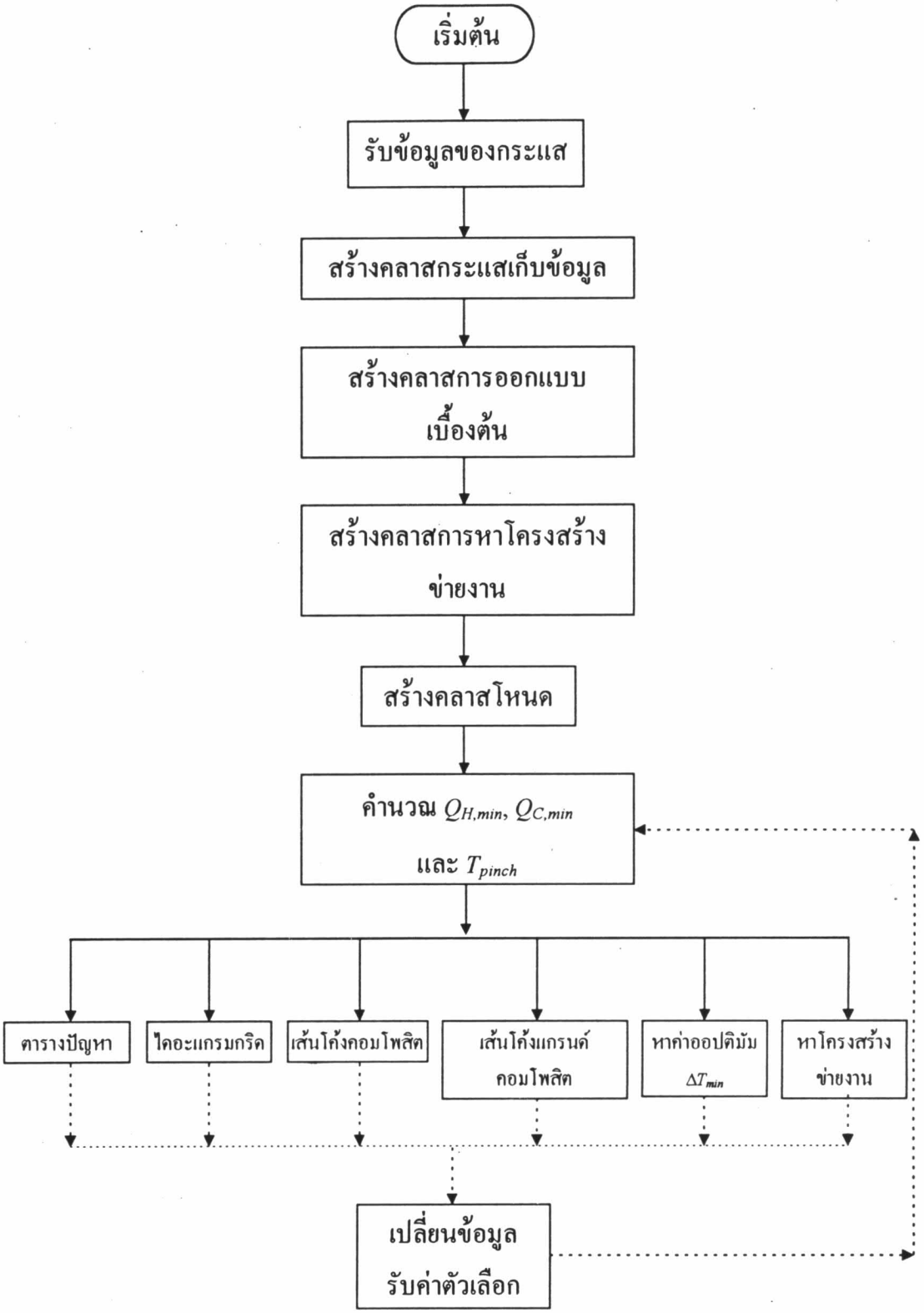
โพลีมอร์ฟิซึม ในการเขียนโปรแกรมก็คือ ผู้ใช้สามารถใช้ชื่อต่างๆ ซ้ำกันได้ แต่จุดประสงค์ของการใช้นั้นจะต่างกันไป ตัวอย่างเช่น สร้างฟังก์ชันสำหรับวาดเส้นชื่อ *LINE* ฟังก์ชันนี้จะใช้วาดเส้นตรงบนผิวระนาบใดๆ ต่อมาในกรณีที่จำเป็นต้องสร้างฟังก์ชันวาดเส้นตรงบนผิวทรงโค้งใดๆ เช่น ทรงกลม กรณีดังกล่าวนี้ก็ยังสามารถสร้างฟังก์ชันใหม่ให้มีชื่อว่า *LINE* เช่นเดียวกันได้ แม้ฟังก์ชันทั้งสองจะมีชื่อซ้ำกัน แต่ตัวแปรหรือข้อมูลภายในจะต่างกัน ซึ่งจะทำให้คอมพิวเตอร์สามารถเลือกฟังก์ชันให้ผู้เขียนโปรแกรมได้ถูกต้อง

4.2 การออกแบบคลาสในข่ายงาน

ในโปรแกรมจะมีคลาสสองคลาสใหญ่ๆ คือ *คลาสการออกแบบเบื้องต้น* และ*คลาสการออกแบบโครงสร้าง* คลาสแรกมีหน้าที่ในการเตรียมหาค่า $Q_{H,min}$ $Q_{C,min}$ และอุณหภูมิพินช์ แสดงผลของตารางปัญหา ไดอะแกรมกริด เส้นโค้งคอมโพสิต เส้นโค้งคอมโพสิตสมดุล เส้นโค้งแกรนด์คอมโพสิต เลือกโหมดการคำนวณแบบยืดหยุ่น หรือแบบไม่ยืดหยุ่น และหาค่า ΔT_{min} ที่เหมาะสมกับข่ายงาน เป็นต้น คลาสที่สองทำหน้าที่หาโครงสร้างของข่ายงานที่เป็นไปได้ทั้งหมดออกมาและแสดงผล รวมทั้งหาข่ายงานที่ประหยัดค่าใช้จ่ายที่สุดในกรณีที่ได้คำตอบหลายโครงสร้างด้วยกัน สำหรับการทำงานของโปรแกรมโดยรวม ให้ดูรูปที่ 4.0

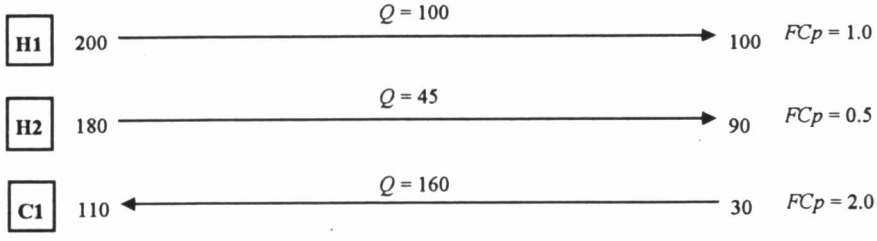
นอกจากนี้ ยังมีคลาสที่สำคัญอีกสองคลาส คือ *คลาสกระแส* และ *คลาสโหนด* สำหรับ *คลาสกระแส* เป็นคลาสที่ใช้เก็บข้อมูลที่จำเป็นของกระแสไว้ เช่น อุณหภูมิขาเข้า อุณหภูมิขาออก ความแปรปรวน ความร้อน และผลคูณของอัตราการไหลและความจุความร้อนจำเพาะ เป็นต้น ซึ่งเวลาใช้งานจะใช้คลาสนี้สร้างวัตถุกระแสร้อน และวัตถุกระแสเย็นตามนิยามของ*คลาสกระแส*ขึ้นมา

สำหรับ*คลาสโหนด* เป็นคลาสที่เก็บสภาวะของการแลกเปลี่ยนความร้อนในข่ายงานไว้ หรือเรียกได้ว่าเป็นคลาสของเครื่องแลกเปลี่ยนความร้อนก็ได้ นอกจากจะมีพารามิเตอร์ที่สำคัญๆ หลายตัวแล้ว จะยังมี*คลาสกระแส*เป็นสมาชิกอยู่ด้วย ซึ่งก่อนที่ได้จะอธิบายต่อไป จะขอกล่าวถึงหลักการการออกแบบ*คลาสโหนด*ก่อน



รูปที่ 4.1 แสดงการทำงานของโปรแกรมโดยรวม

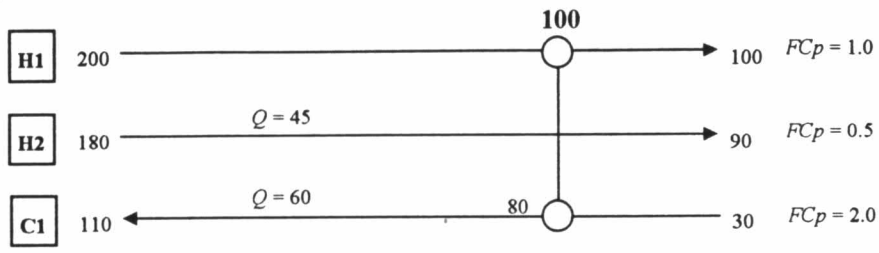
ตัวอย่างที่ 4.1 พิจารณาการแลกเปลี่ยนความร้อนระหว่างกระแสที่มีข้อมูลดังรูปที่ 4.2



รูปที่ 4.2 ข้อมูลกระแสสำหรับสร้างข่ายงานแลกเปลี่ยนความร้อน

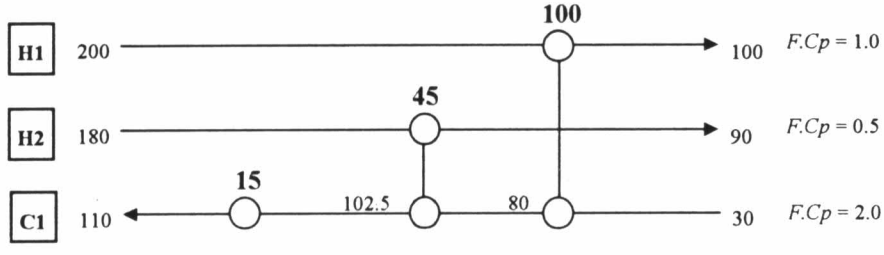
การออกแบบจะมองว่า ณ สภาวะการแลกเปลี่ยนความร้อนหนึ่งๆ หรือเมื่อใดก็ตามที่ได้ใช้กระสวนการจับคู่สร้างเครื่องแลกเปลี่ยนความร้อน, ฮีตเตอร์ หรือคูลเลอร์ขึ้น ที่สภาวะนั้นๆ ถือเป็นโหนดๆ หนึ่ง และจะใช้คลาสโหนดเก็บข้อมูลของทุกกระแสที่สภาวะนั้นไว้ โดยโหนด 0 จะเป็นโหนดเก็บข้อมูลเริ่มต้นของทุกกระแสก่อนการออกแบบ และจะสร้างโหนดที่ 1, 2, 3 เป็นลำดับต่อไป สำหรับเก็บข้อมูลของทุกกระแสหลังการใช้กระสวนการจับคู่นั้นๆ

ในที่นี้ข่ายงานเป็นปัญหาทางด้านความร้อน (สร้างฮีตเตอร์ในข่ายงาน) ให้ ΔT_{min} มีค่าเท่ากับ 10° เริ่มการจับคู่แลกเปลี่ยนความร้อนระหว่างกระแส H1 และ C1 โดยใช้แพทเทิร์นที่ 2 คือแพทเทิร์น A[H] (ในที่นี้ แพทเทิร์นที่ 1 ไม่สามารถใช้ได้) ดังรูปที่ 4.3 สถานะของกระแสที่ถูกจับแลกเปลี่ยนความร้อนจะเปลี่ยนไปนั่นคือ กระแส H1 ความร้อนเหลือ = 0, กระแส C1 ความร้อนเหลือ = 140 และอุณหภูมิขาเข้าคำนวณใหม่ได้เป็น 80°



รูปที่ 4.3 โหนดที่ 1 ใช้แพทเทิร์น A[H]

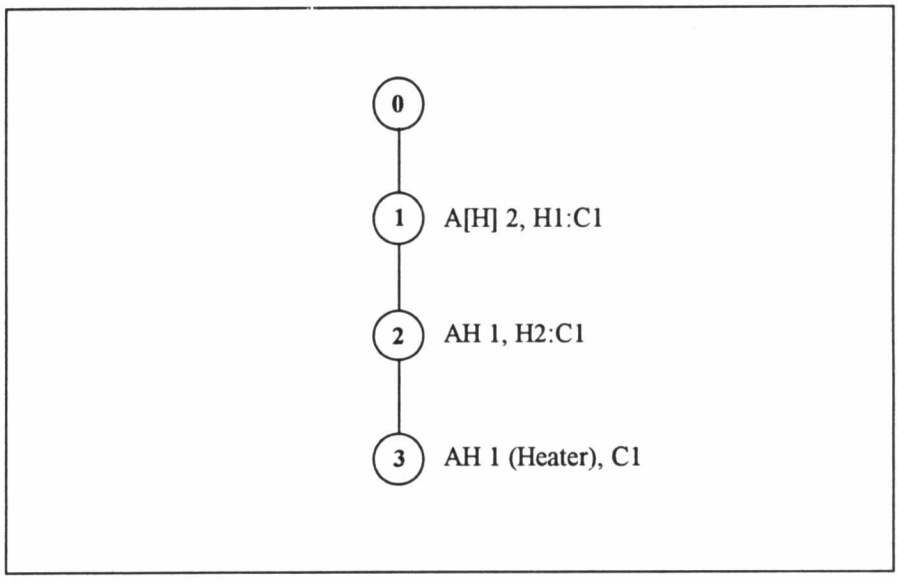
ที่สถานะนี้ จะสร้างโหนดที่ 1 เก็บข้อมูลต่างๆ ไว้ โดยมีโหนด 0 เป็นโหนดแม่ และถ้าสามารถสร้างโหนดที่ 2 ได้ โหนดที่ 1 ก็จะมีสถานะเป็นโหนดแม่ของโหนดที่ 2 อีก และโดยการใช้แพทเทิร์น AH ทำการแลกเปลี่ยนความร้อนระหว่างกระแส H2 และกระแส C1 จะได้โหนดที่ 2 และโหนดที่ 3 ดังรูปที่ 4.4



รูปที่ 4.4 แสดงโครงสร้างที่สมบูรณ์ของข่ายงาน

เนื่องจากโหนดจะเก็บสถานะของกระแสต่างๆไว้ โดยปกติจะใช้ 1 โหนดต่อ 1 แพทเทิร์นการจับคู่ แต่จะสังเกตได้ว่าแพทเทิร์น AH (หรือ BH) นี้ พิเศษกว่าแพทเทิร์นอื่นๆ ที่มีการสร้างฮีตเตอร์ (หรือคูลเลอร์) ด้วยภายในตัวแพทเทิร์นเอง แต่ในที่นี้การออกแบบโปรแกรม

จะทำการสร้างโหนดเมื่อได้สร้างเครื่องแลกเปลี่ยนความร้อนใดๆ ขึ้น (รวมทั้งฮีตเตอร์หรือคูลเลอร์) ดังนั้นในแพทเทิร์นดังกล่าว จะสร้างโหนดไว้สองโหนดต่อ 1 แพทเทิร์น ขณะที่แพทเทิร์นอื่นๆ จะสร้างโหนดเพียงโหนดเดียว สำหรับโหนดที่ได้สร้างมาทั้งสี่โหนดนี้สามารถสร้างเป็นไดอะแกรมได้ดังรูปที่ 4.5



รูปที่ 4.5 แสดงลำดับและสถานะของโหนดต่าง ๆ โดยตัวเลข 0 ถึง 3 แสดงหมายเลขประจำโหนด หมายเลขหลังแพทเทิร์นหมายถึงลำดับแพทเทิร์นตามที่ได้กล่าวไว้ในบทที่ 3

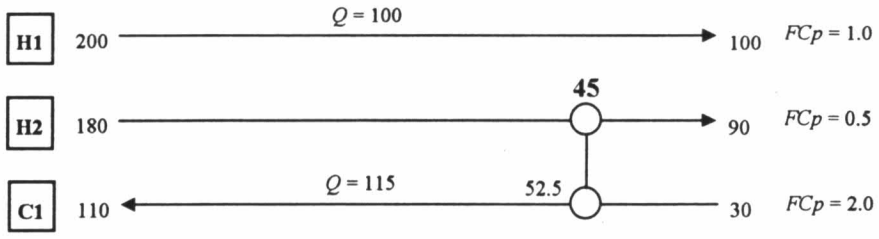
ในการสร้างโหนดใหม่ (คิดจากบนลงล่าง) สำหรับการจับคู่ระหว่างกระแสร้อนและกระแสเย็นคู่ใหม่คู่ใดๆ นั้น จะใช้แพทเทิร์นเริ่มต้นจาก AH หรือจากลำดับที่ 1 ไปเสมอ แต่ถ้าเป็นการสร้างโหนดใหม่โดยย้อนกลับไปหาโหนดเก่าโหนดใดๆ (หรือโหนดแม่) ที่มีอยู่แล้ว จะ

ใช้แพทเทิร์นและลำดับกระแสถัดไปจากโหนดลูกล่าสุดของโหนดแม่นั้นๆ (ที่ทำเช่นนี้เพื่อหาโครงสร้างของข่ายงานอื่นๆที่เป็นไปได้อีก เนื่องจากข่ายงานอาจมีได้หลายคำตอบ)

ตัวอย่างเช่น สำหรับโหนดที่พบคำตอบของข่ายงานที่สมบูรณ์แล้ว หรือสำหรับโหนดที่ไม่สามารถหาคำตอบต่อไปได้อีก (และจะถือว่าโหนดๆ นั้นตายแล้ว) ซึ่งโหนดในลักษณะดังกล่าวไม่สามารถสร้างโหนดในทิศทางจากบนลงล่างได้ ให้ทำการย้อนทิศทางการสร้างโหนดโดยย้อนกลับจากล่างขึ้นบนจากโหนดดังกล่าวไปที่โหนดแม่ (Back tracking) และที่โหนดแม่นี้จะเป็นโหนดเริ่มในการสร้างโหนดใหม่อีกครั้ง โดยจะใช้แพทเทิร์นและลำดับของคู่กระแสถัดไปจากโหนดลูกล่าสุด โดยปกติจะย้อนกลับเพียง 1 โหนดไปที่โหนดแม่ แต่ถ้าโหนดที่กำลังจะย้อนนั้นเป็นฮีตเตอร์หรือ คูลเลอร์ให้ทำการย้อนโหนดไปสองโหนด นอกจากนี้ถ้าโหนดที่เป็นคำตอบนั้น เป็นคำตอบที่ซ้ำกับคำตอบอื่นๆ ก็ให้ทำการย้อนโหนดแล้วสร้างโหนดใหม่ที่ต่างออกไปเช่นกัน สำหรับโหนดที่ให้คำตอบซ้ำ และโหนดที่ไม่สามารถให้คำตอบได้อีกนี้ จะต้องถูกลบออกไปก่อนที่จะสร้างโหนดใหม่ต่อไป เพื่อประหยัดหน่วยความจำของคอมพิวเตอร์ (โดยโหนดที่ถูกลบนี้จะต้องไม่มีโหนดลูกใดๆ ที่เป็นคำตอบอยู่ก่อนแล้ว)

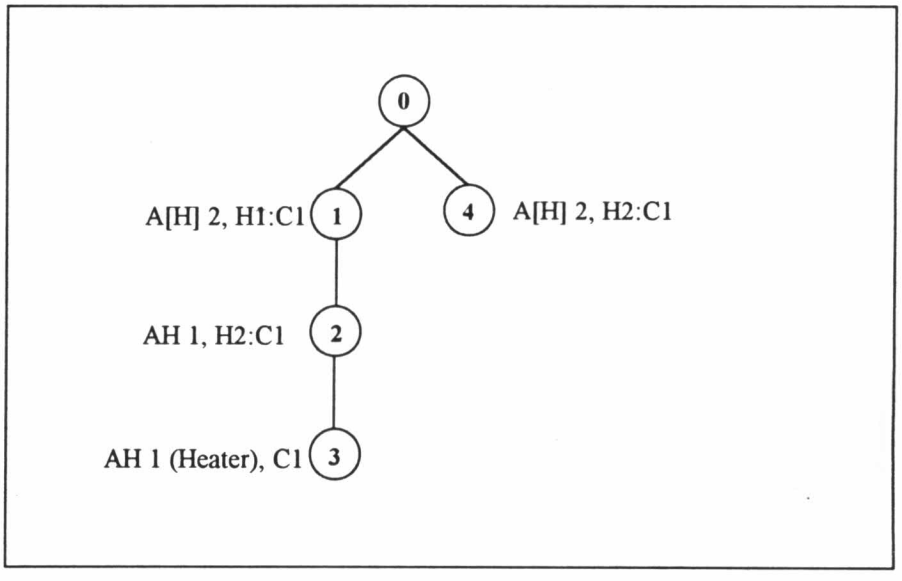
จากรูปที่ 4.5 จะทำการย้อนจากโหนดที่ 3 (ฮีตเตอร์) ไปยังโหนดที่ 1 และใช้แพทเทิร์นที่สอง (A[H]) จับคู่กระแส ในที่นี้ให้คำตอบซ้ำและไม่สามารถหาโครงสร้างใหม่ได้อีก จึงย้อนกลับจากโหนดที่ 1 ไปยังโหนดที่ 0 แล้วทำการค้นหาใหม่ และเนื่องจากโหนด 0 มีโหนดที่ 1 เป็นโหนดลูกล่าสุด ดังนั้นการสร้างโหนดที่ 4 จะทำการค้นหาแพทเทิร์นและลำดับคู่กระแสถัดไปจากโหนดที่ 1 โดยโหนดที่ 1 นี้สังเกตได้ว่าลำดับคู่ของกระแสยังไม่หมด เพราะเป็นการจับคู่

ระหว่าง H1 และ C1 ยังเหลือ H2 และ C1 อีก จึงใช้แพทเทิร์นที่ 2 จับคู่กระแส H2 และ C1 ในการสร้างโหนดที่ 4 ซึ่งเป็นโหนดลูกของโหนด 0 ดังรูปที่ 4.6



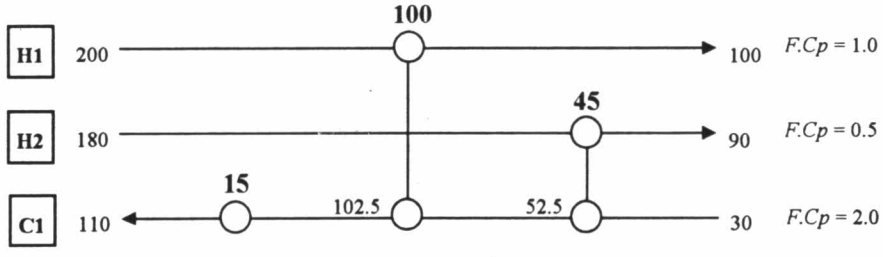
รูปที่ 4.6 โหนดที่ 4 ใช้แพทเทิร์น A[H]

รูปที่ 4.7 แสดงไดอะแกรมของโหนด จะสังเกตได้ว่าโหนดที่ 4 เป็นโหนดลูกของโหนด 0 และลำดับของกระสวนการจับคู่ (หรือแพทเทิร์น) และคู่ลำดับของกระแสจะใช้ต่อจากโหนดลูกของโหนด 0 ล่าสุดที่มีอยู่ ในที่นี้คือโหนดที่ 1

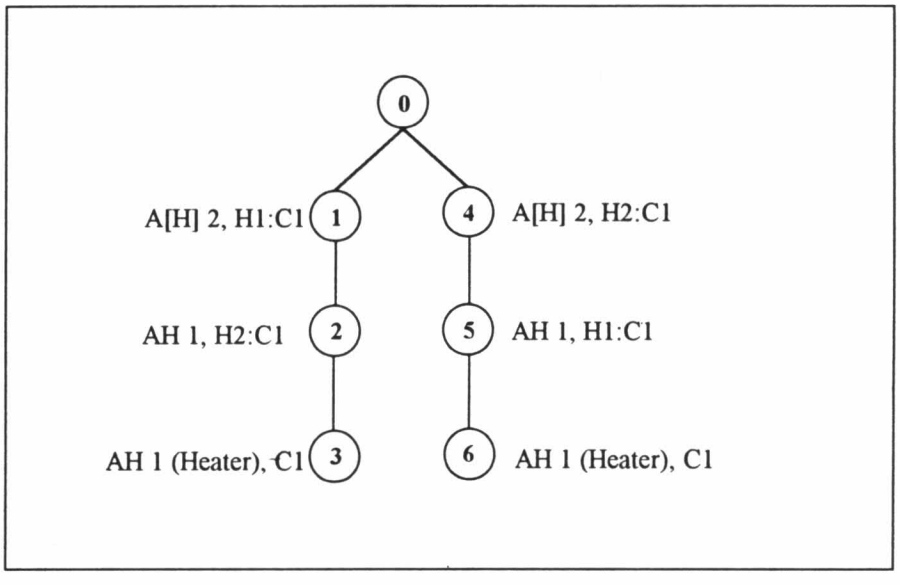


รูปที่ 4.7 แสดงโหนดที่ 4 ซึ่งเป็นโหนดลูกของโหนด 0

การสร้างโหนดใหม่ต่อจากโหนดที่ 4 ต่อไปนั้น จะเริ่มจากการใช้แพทเทิร์นที่ 1 (AH) แลกเปลี่ยนความร้อนระหว่างกระแส H1 กับ C1 ซึ่งจะสร้างโหนดที่ 5 และ 6 ได้คำตอบดังรูปที่ 4.8 แสดงไดอะแกรมของโหนดในรูปที่ 4.9

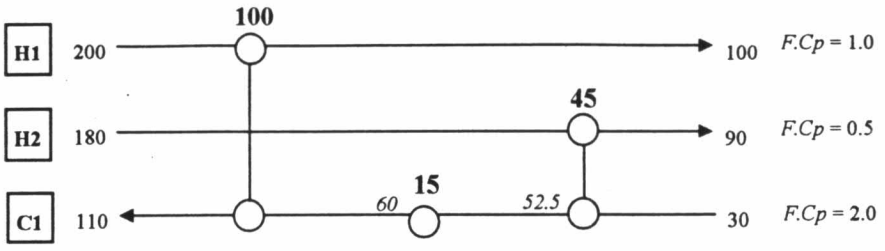


รูปที่ 4.8 แสดงข่ายงานคำตอบที่สอง (เทียบคำตอบแรกในรูปที่ 4.4)

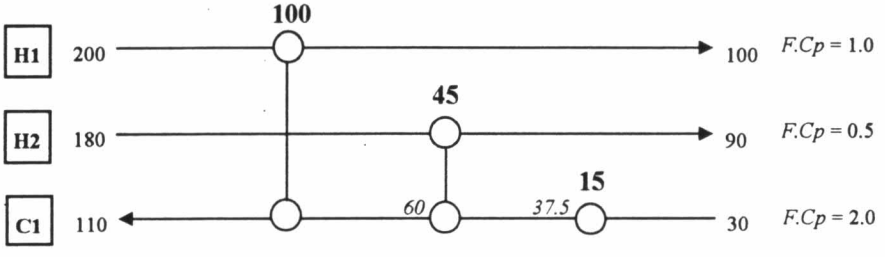


รูปที่ 4.9 ไดอะแกรมของโหนดคำตอบข่ายงานคำตอบที่ 2 ที่เพิ่มขึ้นมา

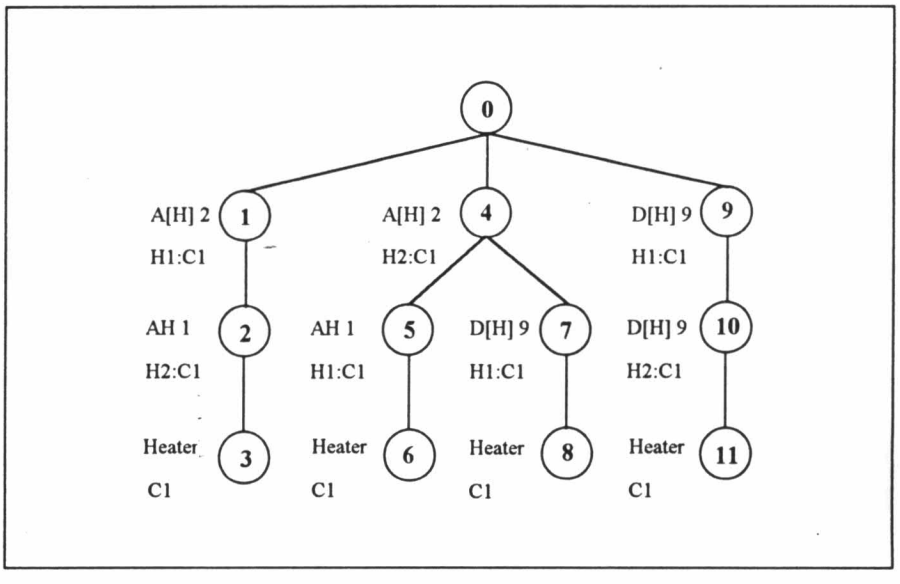
โดยใช้วิธีเช่นนี้ จะได้คำตอบเพิ่มขึ้นมาอีกสองคำตอบ ดังรูปที่ 4.10 และ 4.11 โดยมีโหนดไดอะแกรมที่สมบูรณ์ในรูปที่ 4.12



รูปที่ 4.10 แสดงข่ายงานคำตอบที่สาม



รูปที่ 4.11 แสดงข่ายงานคำตอบที่สี่



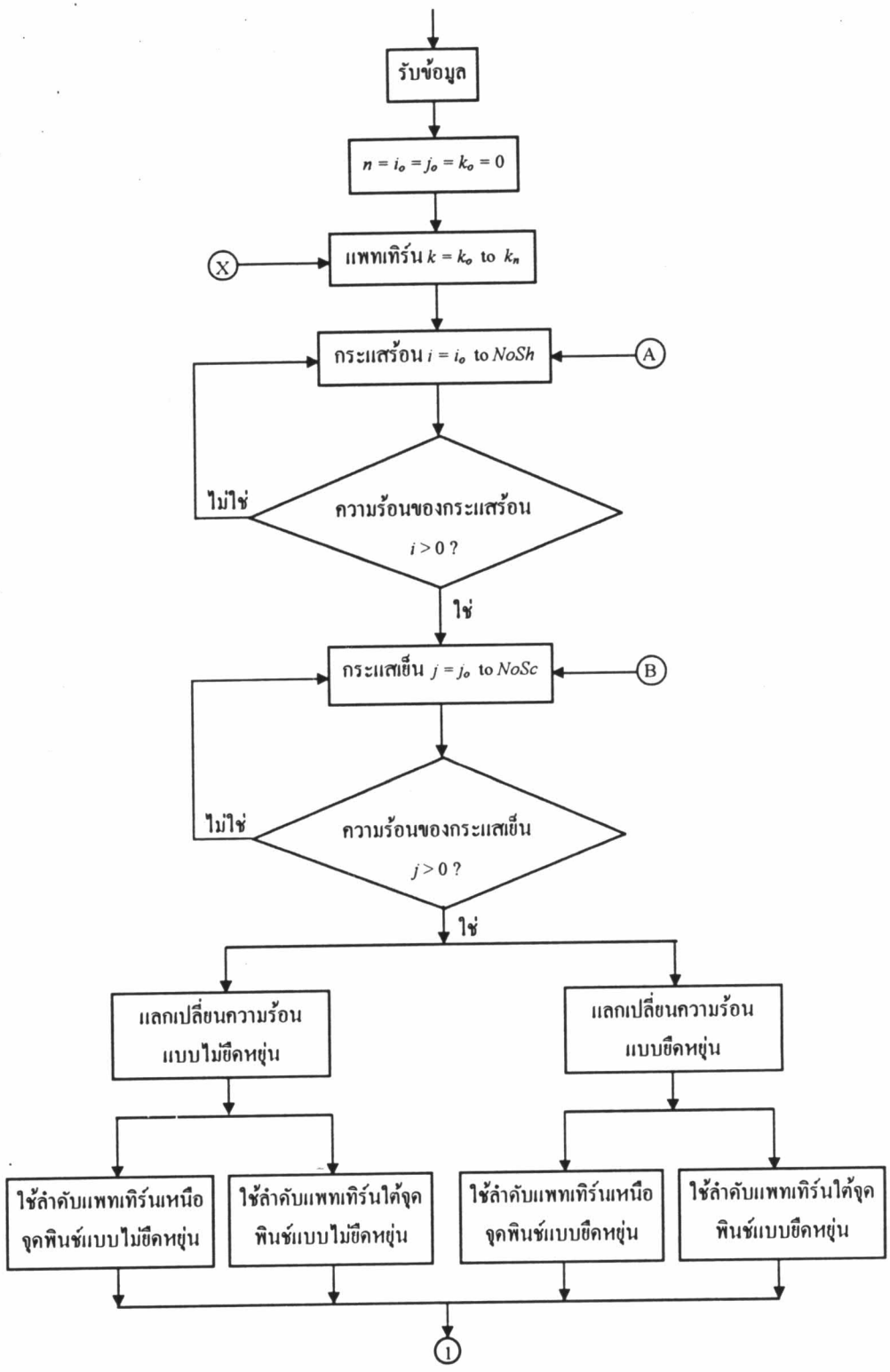
รูปที่ 4.12 ไดอะแกรมของโหนดทั้งสี่คำตอบที่สมบูรณ์ในตัวอย่างที่ 4.1

4.3 โฟลวชาร์ตการทำงานของโปรแกรม

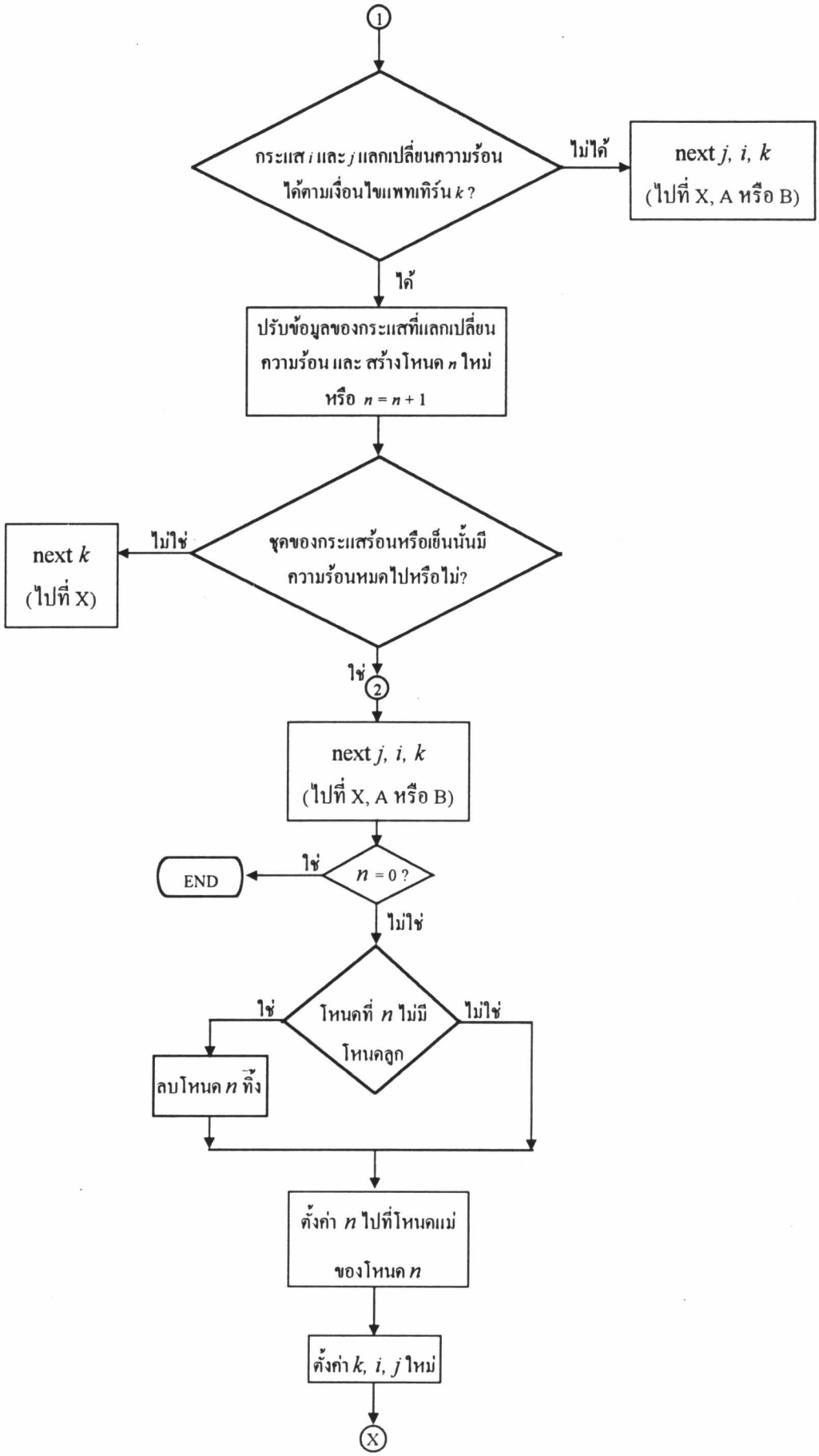
โปรแกรมส่วนการหาโครงสร้างข่ายงานนี้ จะให้ความสำคัญแก่คลาสโหนดและคลาสกระแสเป็นอย่างมาก เพราะว่าจะใช้ประโยชน์ในการคำนวณและการค้นหาคำตอบ (Search) โดยตรง และสำหรับวิธีการค้นหาคำตอบดังกล่าว จะต้องหาให้ครอบคลุมความน่าจะเป็นของโอกาสทั้งหมดที่โครงสร้างข่ายงานสามารถจะมีได้ด้วย โดยไล่คู่ลำดับของกระสวนการจับคู่และกระแสที่เป็นไปได้ให้ครบ ตรวจสอบคำตอบที่ซ้ำและลบโหนดที่ตายออกไป เพื่อประหยัดหน่วยความจำของคอมพิวเตอร์ หรือเมื่อมีการการสร้างโหนดใหม่ไม่ว่าแบบจากบนลงล่างหรือล่างขึ้นบนก็ตาม จะต้องตั้งค่าแพทเทิร์นและลำดับของกระแสต่างๆ สำหรับการคำนวณครั้งใหม่ต่อไปให้ถูกต้อง เพื่อให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพและเร็วที่สุด

จากรูป 4.13 แสดงถึงการไล่ลำดับกระสวนการจับคู่และกระแส เพื่อนำมาแลกเปลี่ยนความร้อน จากโฟลวชาร์ตจะเห็นว่า ได้เลือกอันดับกระสวนการจับคู่ก่อน (k) จากนั้นเลือกกระแสร้อน (i) และกระแสเย็น (j) ตามลำดับ เมื่อได้ข้อมูลของกระสวนการจับคู่และกระแสที่จะนำมาทดสอบการแลกเปลี่ยนความร้อนแล้ว ก็จะตรวจสอบว่าข่ายงานที่กำลังออกแบบนี้เป็นข่ายงานแบบยืดหยุ่นหรือไม่ยืดหยุ่น และเป็นข่ายงานเหนือจุดพินช์หรือใต้จุดพินช์ หรือไม่มีจุดพินช์ โปรแกรมก็จะทำการเลือกฟังก์ชันที่จะใช้งานตามความเหมาะสม

รูปที่ 4.13 (ต่อ) แสดงการตรวจสอบการแลกเปลี่ยนความร้อนระหว่างกระแสร้อนและเย็นในกระสวนการจับคู่ที่เลือกไว้ ถ้าไม่สามารถแลกเปลี่ยนกันได้จะใช้ลำดับการแลกเปลี่ยน



รูปที่ 4.13 แสดงโฟลวชาร์ตการทำงานของโปรแกรม
ในการหาโครงสร้างของข่ายงาน

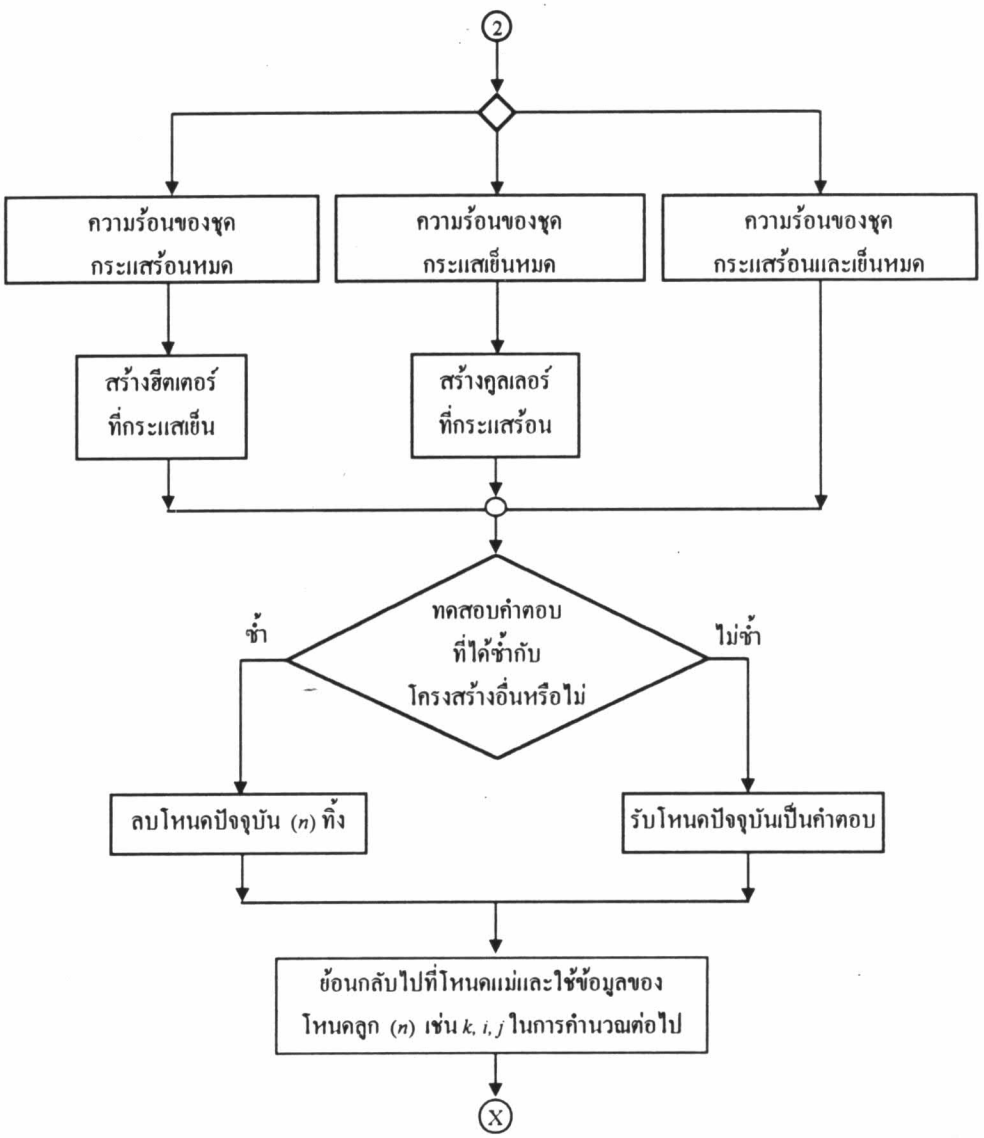

1
กระแส i และ j แลกเปลี่ยนความร้อน ได้ตามเงื่อนไขแพทเทิร์น k ?
ไม่ได้
next j, i, k (ไปที่ X, A หรือ B)
ได้
ปรับข้อมูลของกระแสที่แลกเปลี่ยนความร้อน และ สร้างโหนด n ใหม่ หรือ n = n + 1
ชุดของกระแสร้อนหรือเย็นนั้นมีความร้อนหมดไปหรือไม่?
ไม่ใช่
next k (ไปที่ X)
ใช่
2
next j, i, k (ไปที่ X, A หรือ B)
n = 0 ?
ใช่
END
ไม่ใช่
โหนดที่ n ไม่มีโหนดลูก
ใช่
ไม่ใช่
ลบโหนด n ทิ้ง
ตั้งค่า n ไปที่โหนดแม่ของโหนด n
ตั้งค่า k, i, j ใหม่
X

รูปที่ 4.13 (ต่อ)

ถัดไป เช่น ถ้ายังใช้กระแสเย็นไม่หมด ก็จะนำกระแสเย็นอื่นๆ มาจับคู่กับกระแสร้อนให้หมดก่อน เมื่อกระแสเย็นถูกเลือกมาหมดแล้วและยังไม่ได้ตามเงื่อนไขอีก ก็จะเลือกอันดับกระแสร้อนถัดไป แล้ววนรอบการเลือกกระแสเย็นใหม่ และถ้าเลือกกระแสร้อนหมดแล้วและยังไม่ได้ ก็จะใช้กระสวนการจับคู่ถัดไป แล้วจะตั้งต้นเลือกกระแสร้อนและเย็นใหม่ เป็นต้น แต่ถ้าการแลกเปลี่ยนความร้อนของคู่กระแสในกระสวนการจับคู่นั้นๆ สามารถเป็นไปตามเงื่อนไขได้แล้ว ก็จะทำการสร้างโหนดใหม่ เพื่อเก็บข้อมูลของกระแสที่ทำการแลกเปลี่ยนความร้อนล่าสุดนี้ไว้ แล้วตรวจสอบว่า ความร้อนของกระแสร้อนทั้งหมดหรือกระแสเย็นทั้งหมด ยังมีเหลือไว้แลกเปลี่ยนความร้อนได้ต่อไปอีกหรือไม่ ถ้าความร้อนของกระแสชุดใดหมดลง แสดงว่าสิ้นสุดการหาโครงสร้างข่ายงานนั้นๆ แล้ว และความร้อนของกระแสชุดที่เหลือจะถูกจัดการโดยหน่วยยูทิลิตี จากนั้น โปรแกรมจะเริ่มหาคู่ลำดับใหม่สำหรับการหาโครงสร้างข่ายงานอื่นๆ ที่เป็นไปได้ต่อไปเรื่อยๆ โปรแกรมในส่วนนี้จะหยุดการทำงาน เมื่อสถานะของโหนดนั้นอยู่ที่โหนด 0 และได้ใช้คู่ลำดับของกระสวนการจับคู่และคู่กระแสครบหมดแล้ว

ตัวอย่างเช่น ในการแลกเปลี่ยนความร้อนเหนือจุดพินช์ สมมติว่าความร้อนของกระแสร้อนทุกกระแสถูกใช้หมดแล้ว การค้นหาลำดับการจับคู่กระแสก็จะหยุดลงชั่วคราว (ถ้ายังไม่หมด ก็จะยังทำการค้นคู่ลำดับต่อไป) จากนั้นโปรแกรมจะไปที่ส่วนของการสร้างหน่วยยูทิลิตี ซึ่งในที่นี้จะเป็นการสร้างฮีตเตอร์ ณ กระแสเย็นที่มีความร้อนหลงเหลือ ดูเงื่อนไขการสร้างฮีตเตอร์และคูลเลอร์ได้ในรูปที่ 4.14 ซึ่งเมื่อได้โครงสร้างของข่ายงานออกมาแล้ว ยังต้องตรวจสอบว่า โครงสร้างที่หามาได้นี้ซ้ำกับโครงสร้างเดิมที่เคยมีมาก่อนหรือไม่ โดยตรวจสอบจาก

อุณหภูมิขาเข้าและขาออกของกระแสร้อนและกระแสเย็นในเครื่องแลกเปลี่ยนความร้อนของข่ายงานทุกเครื่อง ซึ่งถ้าไปเหมือนกับของข่ายงานอื่นแล้ว ก็จะทำการลบโหนดปัจจุบันทิ้ง (ไม่ได้ลบโหนดทั้งหมดซึ่งประกอบกันคำตอบที่ซ้ำ) แล้วทำการย้อนโหนดไปที่โหนดแม่ของโหนดปัจจุบัน และจะเริ่มการคำนวณใหม่ แต่ถ้าโครงสร้างข่ายงานนี้ไม่ซ้ำกับโครงสร้างอื่นๆ โปรแกรมก็จะทำการรับโหนดปัจจุบันเป็นคำตอบ จากนั้นโปรแกรมจะวนหาโครงสร้างอื่นๆ ต่อไปจนครบรอบและจบการทำงาน



รูปที่ 4.14 แสดงส่วนของการสร้างหน่วยยูทิลิตีและการตรวจสอบคำตอบ