INFOPRESS GROUP

IDC PREMIER e-Book VERSION

คู่มือเรียน โครงสร้างข้อมูล และ อัลกอริทึม

Data Structure & Algorithm

• FREE CD
Slide
Source Code

อ่านเข้าใจง่าย
อธิบายอย่างละเอียด เป็นขั้นตอน
มีตัวอย่างประกอบทุกหัวข้อ

เรียนรู้การทำงาน
ด้วยการเขียนโปรแกรม
ภาษา C และภาษา Java

เหมาะสำหรับ
นักเรียน นักศึกษา
และผู้ที่สนใจ

วิษณุ ช้างเนียม บรรณาธิการ กิตินันท์ พลสวัสดิ์

Special Thanks

Infopress Group ขอขอบคุณทุกการสนับสนุน
ที่มีเสมอมา จากประสบการณ์การถ่ายทอด
ความรู้ผ่านตัวหนังสือ เราจะมุ่งมั่นสร้างสรรค์งาน
ที่มีคุณภาพเพื่อ คุณผู้อ่านตลอดไป ภายใต้ชื่อที่ว่า

" The Best of Knowledge Books "

คู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm)

ผู้แต่ง วิษณุ ช้างเนียม data-algo@hotmail.com,
facepage : www.facebook.com/DataStructandAlgorithm

บรรณาธิการ กิตินันท์ พลสวัสดิ์ kitinan_p@idcpremier.com

ออกแบบปก วสันต์ พึ่งพูลผล

ออกแบบและจัดรูปเล่ม วุฒิพันธ์ สมพระเมฆ

พิสูจน์อักษร สุนทรี บรรลือศักดิ์, มนฤดี ศรีอุทโยภาส

E-book Publishing สุรีย์รัตน์ จิอู่, จิราภรณ์ โสภา

ข้อมูลทางบรรณานุกรม

วิษณุ ช้างเนียม
คู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm)
นนทบุรี : ไอดีซีฯ, 2559
296 หน้า
1. โครงสร้างข้อมูล
I ชื่อเรื่อง
005.73
978-616-200-652-4

ISBN (E-book)

สำหรับร้านค้าและตัวแทนจำหน่าย
โทรศัพท์ 0-2962-1081-3 ต่อ 112-114
โทรสาร 0-2962-1084

สมาชิกสัมพันธ์
โทรศัพท์ 0-2962-1081-3 ต่อ 121
โทรสาร 0-2962-1084

จัดพิมพ์และจัดจำหน่ายโดย

บริษัท ไอดีซี พรีเมียร์ จำกัด
200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนลทาวเวอร์
ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120
โทรศัพท์ 0-2962-1081 (อัตโนมัติ 10 คู่สาย) โทรสาร 0-2962-1084

ราคา E-book 200 บาท

บรรณาธิการ

การเรียนในสาขาวิชาที่เกี่ยวข้องกับคอมพิวเตอร์ ไม่ว่าจะเป็นสาขาวิศวกรรมคอมพิวเตอร์ วิทยาการคอมพิวเตอร์ เทคโนโลยีสารสนเทศ และอื่นๆ คงปฏิเสธไม่ได้ถึงความสำคัญในรายวิชาโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) เนื่องจากเป็นความรู้พื้นฐานที่สำคัญอย่างยิ่งในการศึกษาต่อทางแขนงวิชาต่างๆ ไม่ว่าจะเป็นหลักการการจัดเก็บข้อมูล การจัดเรียงข้อมูล และการค้นหาข้อมูล

ทางสำนักพิมพ์จึงมีความตั้งใจที่จะจัดทำหนังสือเล่มนี้ เพื่อเป็นหนังสือสำหรับใช้ประกอบการเรียนการสอนในรายวิชาโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) เพื่อเน้นอธิบายพื้นฐานความคิดเกี่ยวกับโครงสร้างข้อมูลแบบต่างๆ และเพิ่มความเข้าใจในการทำงานของอัลกอริทึม โดยมีตัวอย่างพร้อมคำอธิบายการทำงานอย่างละเอียด ซึ่งผู้อ่านสามารถเลือกศึกษาได้ทั้งภาษา C และภาษา Java

สุดท้ายนี้ทางสำนักพิมพ์หวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะมีประโยชน์สำหรับผู้อ่านที่ใช้ประกอบในการเรียนการสอนไม่มากก็น้อย และหากหนังสือเล่มนี้มีข้อผิดพลาดประการใด ทางสำนักพิมพ์ต้องขออภัยมา ณ โอกาสนี้ด้วย

ด้วยความเคารพยิ่ง
กิตินันท์ พลสวัสดิ์

คำนำ

หนังสือเล่มนี้ เรียบเรียงขึ้นสำหรับใช้ประกอบการเรียนการสอนในรายวิชาโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) สำหรับนักเรียน นักศึกษา เพื่อให้ความรู้ความเข้าใจในเรื่องโครงสร้างข้อมูลและอัลกอริทึม

หนังสือเล่มนี้จะเน้นอธิบายตั้งแต่พื้นฐานเป็นสำคัญ ตั้งแต่อธิบายให้รู้จักกับโครงสร้างข้อมูลและอัลกอริทึม ตัวการวัดประสิทธิภาพของอัลกอริทึม เครื่องมือที่ใช้จัดเก็บข้อมูลแบบอาร์เรย์และลิงค์ลิสต์ วิธีการเก็บข้อมูล และการนำข้อมูลออกมาในรูปแบบต่างๆ ทั้งสแตก คิว ทรีและอื่นๆ รวมไปถึงการจัดเรียงข้อมูล การค้นหาข้อมูล และเรื่องของกราฟ

ในทุกๆ หัวข้อจะมีตัวอย่างประกอบการอธิบายเพื่อเพิ่มความเข้าใจ มีตัวอย่างโค้ดโปรแกรมทั้งภาษา C และภาษา Java สำหรับทดสอบการทำงานของอัลกอริทึม พร้อมคำอธิบายโค้ดอย่างละเอียด

สุดท้ายนี้ ขอขอบพระคุณคุณพ่อคุณแม่และครอบครัวของผู้เขียนที่ให้กำลังใจและให้การสนับสนุนมาตลอด ขอขอบคุณอาจารย์ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับผู้เขียน ประโยชน์ของหนังสือเล่มนี้ขอมอบให้แด่อาจารย์ผู้ประสิทธิประสาทวิชาทุกท่าน บิดา-มารดา และครอบครัว

วิษณุ ช้างเนียม

คำนิยม

ในการศึกษาด้านวิทยาการคอมพิวเตอร์ ความรู้ทางด้านโครงสร้างข้อมูลและอัลกอริทึมถือเป็นพื้นฐานที่สำคัญต่อการพัฒนาโปรแกรมคอมพิวเตอร์ ซึ่งเราจะเห็นได้จากการแข่งขันเขียนโปรแกรมต่างๆ อาทิ การแข่งขันคอมพิวเตอร์โอลิมปิก การแข่งขันโปรแกรมมิงโลกระดับมหาวิทยาลัย ACM-ICPC เป็นต้น ที่ต้องอาศัยความรู้ทางด้านนี้ทั้งสิ้น หัวใจหลักของการศึกษาโครงสร้างข้อมูลและอัลกอริทึมคือ การประมวลผลข้อมูลอย่างมีประสิทธิภาพ ซึ่งหยั่งรากลึกมาตั้งแต่ยุคเริ่มต้นของพัฒนาการด้านคอมพิวเตอร์ มีการศึกษาและพัฒนาโครงสร้างข้อมูลในหลากหลายรูปแบบ รวมถึงการออกแบบอัลกอริทึมเพื่อแก้ปัญหาที่ซับซ้อนต่างๆ ในปัจจุบันถือได้ว่าความรู้ทางด้านโครงสร้างข้อมูลและอัลกอริทึมก้าวหน้าไปมาก แต่ก็ยังมีปัญหาที่เรายังไม่สามารถหาคำตอบที่ดีได้

หนังสือคู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) เล่มนี้ ผู้แต่ง อาจารย์วิษณุ ช้างเนียม ได้อธิบายความรู้พื้นฐานในเรื่องโครงสร้างฐานข้อมูลตั้งแต่ระดับง่ายไปจนถึงระดับยาก ในเล่มมีการใช้รูปภาพประกอบคำอธิบาย แสดงเป็นลำดับ เป็นขั้นเป็นตอน ทำให้ง่ายต่อการทำความเข้าใจ อ่านแล้วไหลลื่น ไม่ติดขัด ไม่ได้รู้สึกว่ากำลังอ่านเรื่องวิชาการที่ยากเกินไป ซึ่งสะท้อนถึงประสบการณ์ในการสอนของผู้แต่งได้เป็นอย่างดี

จุดเด่นอีกประการหนึ่งของหนังสือเล่มนี้คือ การแสดงตัวอย่างโปรแกรมด้วยภาษาโปรแกรมมิง 2 ภาษาด้วยกันคือ ภาษา Java และภาษา C ซึ่งถือได้ว่าเป็นตัวเลือกที่ดีและเหมาะสมสำหรับยุคปัจจุบัน เนื่องจากภาษา Java เป็นที่นิยมสำหรับผู้ที่ต้องการพัฒนาโปรแกรมในลักษณะ Object Oriented Programming (OOP) ส่วนภาษา C ถือได้ว่าเป็นบรรทัดฐานของการพัฒนาโปรแกรมแบบโครงสร้างในยุคปัจจุบัน

ข้าพเจ้าหวังว่าหนังสือคู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) จะช่วยให้ผู้อ่านได้ทั้งความรู้ ความเข้าใจในพื้นฐานและการพัฒนาโครงสร้างข้อมูลและอัลกอริทึม และที่สำคัญเป็นประกายไฟทางความคิดในการแก้ปัญหาทางด้านคอมพิวเตอร์ต่อไป

สุดท้ายนี้ข้าพเจ้าต้องขอขอบคุณทางสำนักพิมพ์ บรรณาธิการ และผู้แต่งที่ให้เกียรติข้าพเจ้าเป็นอย่างสูงสำหรับการเขียนคำนิยมนี้

ดร.อัครา ประโยชน์
หัวหน้าภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

คำนิยม

หนังสือคู่มือเรียนโครงสร้างข้อมูลและอัลกอริทึม (Data Structure & Algorithm) เล่มนี้ ถือได้ว่าเป็นหนังสือที่เหมาะสำหรับการเรียนการสอนในรายวิชาโครงสร้างข้อมูลและอัลกอริทึมในระดับอุดมศึกษา เนื่องจากเป็นหนังสือที่ครอบคลุมตั้งแต่แนวคิด ทฤษฎี และหลักการอย่างครบถ้วนสมบูรณ์ รวมถึงมีการสอดแทรกตัวอย่างโจทย์และแบบฝึกหัดเพื่อเสริมความเข้าใจ ซึ่งอธิบายด้วยโค้ดโปรแกรม 2 ภาษาคือ ภาษา C และภาษา Java ทำให้อาจารย์และนักศึกษาสามารถเลือกภาษาที่เหมาะสมสำหรับการเรียนการสอนได้เอง

อีกทั้งหนังสือเล่มนี้ได้เรียบเรียงและนำเสนอเนื้อหาอย่างเป็นลำดับ ด้วยสำนวนการเขียนที่อ่านเข้าใจง่าย พร้อมภาพประกอบที่เป็นขั้นเป็นตอน ทำให้ง่ายต่อการเข้าใจ ซึ่งบ่งบอกถึงความเชี่ยวชาญและประสบการณ์การสอนในรายวิชาโครงสร้างข้อมูลและอัลกอริทึมของผู้เขียนได้เป็นอย่างดี

สุดท้ายนี้ดิฉันเชื่อว่าหนังสือเล่มนี้จะช่วยให้ผู้อ่านได้ความรู้ ความเข้าใจเกี่ยวกับโครงสร้างข้อมูลและอัลกอริทึมเป็นอย่างดี และดิฉันขอขอบคุณทางสำนักพิมพ์ที่ให้เกียรติสำหรับการเขียนคำนิยมนี้

ผู้ช่วยศาสตราจารย์สุดา เธียรมนตรี
อาจารย์ประจำสาขาวิชาคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์ มหาวิทยาลัยทักษิณ

Contents

CHAPTER 01

รู้จักกับโครงสร้างข้อมูลและอัลกอริทึม

หนังสือเล่มนี้เป็นหนังสือที่สอนในเรื่องโครงสร้างข้อมูลและอัลกอริทึม มุ่งเน้นในเรื่องวิธีการและขั้นตอนการแก้ไขปัญหาต่างๆ โดยใช้คุณสมบัติและความสามารถของโครงสร้างข้อมูล ซึ่งถือได้ว่าเป็นรายวิชาหลักในการเรียนในสาขาคอมพิวเตอร์ ดังนั้น ผู้อ่านควรศึกษาเนื้อหาและตัวอย่างต่างๆ ในหนังสือให้เข้าใจ เพื่อใช้เป็นพื้นฐานในการพัฒนาโปรแกรมต่อไป

โครงสร้างข้อมูลและอัลกอริทึมคืออะไร

โครงสร้างข้อมูล (Data Structure) คือ การจัดการข้อมูลในหน่วยความจำภายในเครื่องคอมพิวเตอร์ หรือในบางครั้งเป็นการจัดการข้อมูลในดิสก์ให้มีความสัมพันธ์กันภายในกลุ่มข้อมูลให้มีรูปแบบและข้อกำหนดที่ชัดเจนในการกำหนดคุณสมบัติเพื่อสร้างความสัมพันธ์ภายในกลุ่มข้อมูล รูปแบบการเก็บข้อมูลที่อยู่ในรูปแบบโครงสร้างข้อมูล เช่น อาร์เรย์ (Array), ลิงค์ลิสต์ (Link-list), สแตก (Stack), ไบนารีทรี (Binary tree) เป็นต้น

อัลกอริทึม (Algorithm) หรือเรียกอีกอย่างว่า **ขั้นตอนวิธี** เป็นวิธีการแสดงลำดับขั้นตอนในการทำงานหรือการแก้ไขปัญหาอย่างใดอย่างหนึ่ง เช่น การกำหนดขั้นตอนเพื่อแก้ไขปัญหาการจัดเรียงเอกสารในแฟ้มข้อมูล หรือการกำหนดอัลกอริทึมในการค้นหาข้อมูลในแฟ้มข้อมูลทั้งหมด เป็นต้น

อัลกอริทึมหรือขั้นตอนวิธีเป็นสิ่งที่พบได้ในชีวิตประจำวัน เช่น ขั้นตอนการเติมเงินในโทรศัพท์มือถือผ่านตู้เติมเงินอัตโนมัติ, ขั้นตอนการใช้งานตู้กดเงินอัตโนมัติ (ATM) เพื่อทำธุรกรรมทางการเงิน เป็นต้น

ตัวอย่างที่ 1.1 แสดงอัลกอริทึมหรือขั้นตอนวิธีการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงินดังนี้

1. ใส่บัตร ATM
2. ป้อนรหัสผ่านของบัตร ATM
3. ในหน้าบริการ เลือกบริการรายการโอนเงิน
4. เลือกรูปแบบการโอนเงินว่าจะโอนเงินเข้าบัญชีอื่นธนาคารเดียวกัน หรือธนาคารอื่นๆ
5. กดหมายเลขบัญชีที่ต้องการโอนเงินเข้าบัญชี
6. ตรวจสอบเลขที่บัญชีที่ป้อนถูกต้องหรือไม่ ถ้าถูกต้องให้กดตกลง
7. กรอกจำนวนเงินที่ต้องการโอนเงิน แล้วกดตกลง
8. ชื่อบัญชีและจำนวนเงินที่ต้องการโอนจะปรากฏขึ้นเพื่อยืนยันความถูกต้องในการโอนเงิน เมื่อตรวจสอบบัญชีและจำนวนเงินถูกต้อง ให้กดตกลง เป็นการเสร็จสิ้นการโอนเงิน
9. รับบัตร ATM
10. รับใบสลิปการโอนเงิน

ตัวอย่างที่ 1.2 แสดงอัลกอริทึมการหาข้อมูลในอาร์เรย์ขนาด n ข้อมูลดังนี้

1. รับข้อมูลตัวเลขที่ต้องการค้นหา
2. เปรียบเทียบข้อมูลในอาร์เรย์ทีละตัวตั้งแต่ข้อมูลในตำแหน่งที่ 0 จนถึงตำแหน่งที่ n-1
3. ถ้าข้อมูลในอาร์เรย์ตรงกับข้อมูลที่ต้องการค้นหา แสดงว่าเจอข้อมูล และจบการค้นหาข้อมูล
4. ถ้าเปรียบเทียบข้อมูลจนถึงตำแหน่งที่ n-1 แล้วไม่พบข้อมูลตัวใดในอาร์เรย์เลย แสดงว่าไม่มีข้อมูลที่ต้องการค้นหาในอาร์เรย์

ประโยชน์ของโครงสร้างข้อมูลและอัลกอริทึม

โครงสร้างข้อมูลและอัลกอริทึม เป็นวิธีการจัดการกับข้อมูลที่ทำให้คอมพิวเตอร์สามารถจัดการกับข้อมูลเพื่อนำมาใช้งานได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น

- การจัดเก็บข้อมูลจำนวนมากในรูปแบบของโครงสร้างข้อมูลที่เหมาะสม จะทำให้สามารถนำข้อมูลไปใช้ได้อย่างมีประสิทธิภาพ โดยใช้ทรัพยากรที่มีได้อย่างคุ้มค่าที่สุด (ใช้พื้นที่หน่วยความจำน้อยที่สุด) อีกทั้งยังใช้เวลาในการประมวลผลน้อยที่สุดด้วย ส่งผลให้คอมพิวเตอร์ทำงานเร็วขึ้นนั่นเอง
- การอ่านข้อมูลเพื่อใช้ในการประมวลผล คอมพิวเตอร์จะอ่านข้อมูลมาเก็บในหน่วยความจำหลัก (หน่วยความจำแรม) และในกรณีที่ข้อมูลมีขนาดมากกว่าขนาดของหน่วยความจำ คอมพิวเตอร์จะใช้โครงสร้างข้อมูลและอัลกอริทึมเข้ามาช่วยในการจัดการกับข้อมูลเหล่านั้น เพื่อให้สามารถจัดสรรหน่วยความจำได้อย่างเหมาะสมและสามารถทำงานได้อย่างต่อเนื่อง
- การพัฒนาโปรแกรมโดยใช้โครงสร้างข้อมูลและอัลกอริทึมที่เหมาะสม สามารถเพิ่มประสิทธิภาพการทำงานของโปรแกรมได้เช่นกัน (ใช้หน่วยความจำน้อยและประมวลผลได้รวดเร็ว)

ทั้งนี้ประโยชน์ของโครงสร้างข้อมูลและอัลกอริทึมที่กล่าวมาเป็นเพียงตัวอย่างเล็กน้อยเท่านั้น ซึ่งโครงสร้างข้อมูลและอัลกอริทึมยังสามารถประยุกต์เพื่อใช้งานได้อีกมากมาย ไม่ว่าจะเป็นการเรียงลำดับข้อมูล การค้นหาข้อมูล เป็นต้น

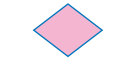
ผังงาน (Flowchart)

ผังงาน หรือเรียกว่า **โฟลว์ชาร์ต (Flowchart)** เป็นเครื่องมือที่ใช้ออกแบบระบบงานด้วยสัญลักษณ์ ช่วยให้มีโครงสร้างของระบบงานที่เป็นลำดับขั้นตอนและเข้าใจได้ง่าย

การนำผังงานมาใช้ในการออกแบบโปรแกรม สามารถตรวจสอบได้ว่ามีลำดับขั้นตอนการทำงานถูกต้องหรือไม่ และสามารถเปลี่ยนแปลงแก้ไขข้อผิดพลาดของระบบงานภายในผังงานได้ง่ายกว่าการหาข้อผิดพลาดที่เกิดจากการเขียนโปรแกรม อีกทั้งยังช่วยลดความสับสนในการพัฒนาโปรแกรมอีกด้วย

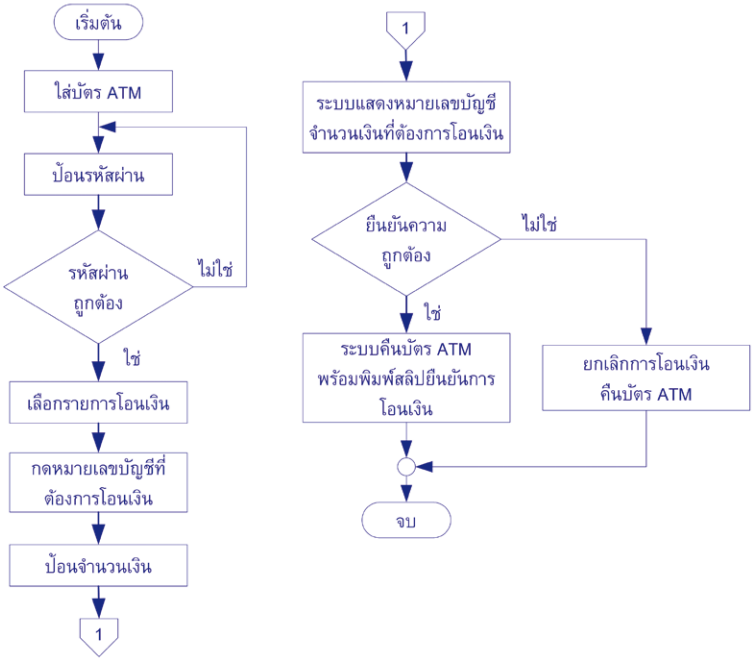
วัตถุประสงค์ที่สำคัญที่สุดในการนำผังงานมาเป็นเครื่องมือในการพัฒนาโปรแกรมคือ เพื่อให้บุคคลอื่นสามารถทำความเข้าใจถึงลำดับขั้นตอนการทำงานของโปรแกรม และผลลัพธ์ที่ได้จากการทำงานของโปรแกรมที่พัฒนาขึ้นได้

สัญลักษณ์ที่นำมาใช้ในการเขียนผังงานมีอยู่ 8 สัญลักษณ์ดังนี้

สัญลักษณ์	ความหมายสัญลักษณ์ในการใช้งานในผังงาน
 Terminator	จุดเริ่มต้นและจุดสิ้นสุดของโปรแกรม
 Process	การประมวลผลหรือการคำนวณของโปรแกรม
 Data	รับข้อมูลเข้ามาในโปรแกรม หรือส่งค่าออกไปจากโปรแกรม
 Decision	ตรวจสอบเงื่อนไข แล้วเลือกการทำงานของโปรแกรม
 Document	แสดงผลออกทางเอกสาร
 On-page reference	จุดเชื่อมต่อหลายเส้นทางของโปรแกรมให้เหลือการเข้ามาเพียงเส้นทางเดียว
Off-page reference	ขึ้นหน้าใหม่ในกรณีที่ผังงานมีความยาวเกินกว่าที่จะแสดงพอในหนึ่งหน้า
	ลูกศรแสดงทิศทางการทำงานของโปรแกรมและการไหลของข้อมูล

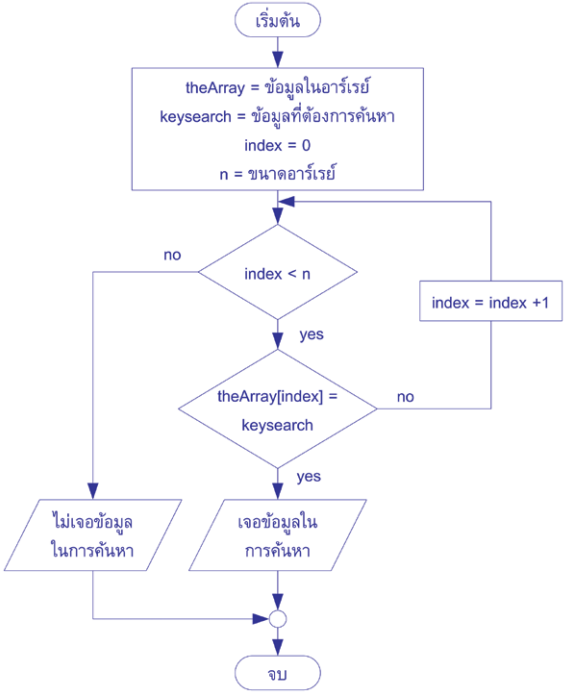
ตัวอย่างที่ 1.3 แสดงผังงานขั้นตอนการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงินดังนี้

รูปที่ 1-1 ผังงานขั้นตอนการใช้ตู้กดเงินอัตโนมัติ (ATM) เพื่อโอนเงิน



ตัวอย่างที่ 1.4 แสดงผังงานอัลกอริทึมค้นหาข้อมูลในอาร์เรย์ขนาด n ข้อมูลดังนี้

รูปที่ 1-2 ผังงานอัลกอริทึมค้นหาข้อมูลในอาร์เรย์ขนาด n ข้อมูล



โค้ดรหัสเทียม (Pseudo code)

โค้ดรหัสเทียม (Pseudo code) เป็นโครงสร้างรหัสที่รวมทั้งภาษาเขียนกับภาษาคอมพิวเตอร์เข้าไว้ด้วยกัน เพื่อใช้ในการอธิบายโครงสร้างและลำดับขั้นตอนการทำงานของโปรแกรมโดยไม่อ้างอิงภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง เพื่อเป็นสื่อกลางแทนการเขียนด้วยโค้ดโปรแกรม (Code Program) ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.5 แสดง Pseudo code การค้นหาข้อมูลในอาร์เรย์ดังนี้

```
1  +searching(in theArray:arrayType,in KeySearch:keyType,in MaxData:integer):boolean
2    for (n = 0 to MaxData-1){
3      if (theArray[n] == KeySearch){
4        data is found data then return true
5      }
6    }
7    if (n == MaxData-1){
8      data is not found then return false
9    }
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1 ประกาศฟังก์ชันหรือเมธอด searching โดยกำหนดให้มีตัวแปรอินพุต 3 ตัวคือ

- theArray เป็นข้อมูลทั้งหมดที่เก็บในอาร์เรย์
- KeySearch เป็นข้อมูลที่ต้องการค้นหา
- MaxData เป็นขนาดของอาร์เรย์ที่เก็บข้อมูล

และกำหนดให้ส่งคืนค่าผลการค้นหาข้อมูลเป็นข้อมูลชนิด boolean

บรรทัดที่ 2 วนลูปตั้งแต่ตัวแปร n เท่ากับ 0 จนถึงค่าในตัวแปร MaxData-1

บรรทัดที่ 3-4 ถ้าข้อมูลใน theArray ตำแหน่งที่ n เท่ากับค่าข้อมูล KeySearch แสดงว่าเจอข้อมูลที่ต้องการค้นหา ให้ฟังก์ชันคืนค่าเป็น true

บรรทัดที่ 7-8 ถ้าค่าตัวแปร n เท่ากับ MaxData แสดงว่าไม่พบข้อมูลในอาร์เรย์ให้คืนค่าเป็น false

ตัวอย่างที่ 1.6 แสดง Pseudo code แสดงผลการเรียน (Grade) จากคะแนนที่รับเข้ามาดังนี้

```
1  +showgrade(in score:double)
2    if (score >= 80)
3        output "A"
4    else if (score >= 75)
5        output "B+"
6    else if (score >= 70)
7        output "B"
8    else if (score >= 65)
9        output "C+"
10   else if (score >= 60)
11       output "C"
12   else if (score >= 55)
13       output "D+"
14   else if (score >= 50)
15       output "D"
16   else output "F"
```

อธิบายการทำงานของโปรแกรม

บรรทัดที่ 1	ประกาศฟังก์ชันหรือเมธอด showgrade กำหนดให้รับค่าคะแนนไว้ในตัวแปร
บรรทัดที่ 2-3	ถ้า score มากกว่าเท่ากับ 80 ให้แสดงเกรด “A”
บรรทัดที่ 4-5	แต่ถ้า score มากกว่าเท่ากับ 75 ให้แสดงเกรด “B+”
บรรทัดที่ 6-7	แต่ถ้า score มากกว่าเท่ากับ 70 ให้แสดงเกรด “B”
บรรทัดที่ 8-9	แต่ถ้า score มากกว่าเท่ากับ 65 ให้แสดงเกรด “C+”
บรรทัดที่ 10-11	แต่ถ้า score มากกว่าเท่ากับ 60 ให้แสดงเกรด “C”
บรรทัดที่ 12-13	แต่ถ้า score มากกว่าเท่ากับ 55 ให้แสดงเกรด “D+”
บรรทัดที่ 14-15	แต่ถ้า score มากกว่าเท่ากับ 50 ให้แสดงเกรด “D”
บรรทัดที่ 16	ถ้า score น้อยกว่า 50 ให้แสดงเกรด “F”

Abstract Data Type

Abstract Data Type หรือเรียกว่า **ADT** เป็นการประกาศถึงคุณสมบัติของโครงสร้างข้อมูลและกลุ่มตัวดำเนินการที่กระทำกับโครงสร้างข้อมูล

คุณสมบัติของโครงสร้างข้อมูลที่กำหนดใน ADT เป็นการแสดงถึงลักษณะของโครงสร้างข้อมูลที่นำมาใช้งาน และกลุ่มตัวดำเนินการจะเป็นฟังก์ชันที่กำหนดการทำงานของโปรแกรมที่กระทำกับโครงสร้างข้อมูล ซึ่งแต่ละกลุ่มตัวดำเนินการจะทำงานอย่างอิสระไม่เกี่ยวเนื่องกัน (ไม่มีการส่งค่าระหว่างกลุ่มตัวดำเนินการ) และมีลักษณะการทำงานที่ชัดเจน

รายการ ADT คือ การรวบรวมลำดับการทำงานที่ชัดเจน โดยใช้ตัวเลขในการอ้างอิงตำแหน่งดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.7 รายการ ADT แสดงการจัดการรายการสิ่งของในร้านค้าดังนี้

1. Create an empty list. (สร้างรายการว่างเปล่า)
2. Determine whether a list is empty. (สนใจรายการที่ว่างเปล่า)
3. Determine the number of items on a list. (สนใจตัวเลขสิ่งของในรายการ)
4. Add an item at a given position in the list. (เพิ่มสิ่งของในตำแหน่งที่ให้มาในรายการ)
5. Remove the item at a given position in the list. (ลบสิ่งของในตำแหน่งที่ให้มาในรายการ)
6. Remove all the items from the list. (ลบสิ่งของทั้งหมดในรายการ)
7. Retrieve (get) the item at a given position in the list. (นำสิ่งของกลับคืนมาในตำแหน่งที่ให้มาในรายการ)

จากรายการ ADT นำไปเขียนเป็น Pseudo code ได้ดังนี้

```
+createList()
  //สร้างรายการว่างเปล่า
+isEmpty():boolean{query}
  //สนใจรายการที่ว่างเปล่า
+size():integer{query}
  //คืนค่าตัวเลขรายการทั้งหมด
+add(in index:integer, in item:ListItemType)
  //เพิ่มข้อมูล (item) ในตำแหน่ง index ในรายการ ถ้า 0 <= index < size()
  //ถ้า index = size()+1 ไม่เพิ่มข้อมูลในรายการเนื่องจากรายการเต็ม
+remove(in index:integer)
  //ลบข้อมูลในตำแหน่งของ index ในรายการ ถ้า 0 <= index < size()
  //ถ้า index < 0 ไม่ลบข้อมูลในรายการเนื่องจากรายการว่างเปล่า
+removeAll()
  //ลบข้อมูลทั้งหมดในรายการ
+get(in index:ListItemType){queue}
  //คืนค่า item ในตำแหน่ง index ของรายการ ถ้า 0 <= index < size()
  //ให้เลื่อน index ไปทางซ้าย (ลดค่า index ลงหนึ่งตำแหน่ง) และไม่คืนค่าถ้า index เกินขอบเขตที่กำหนด
```

การออกแบบ ADT เป็นการพัฒนามาจากกระบวนการแก้ไขปัญหา ตัวอย่างเช่น ต้องการตรวจสอบวันหยุดทั้งหมดในหนึ่งปี ซึ่งมีวิธีการคือ ดูปฏิทินว่ามีวันหยุดวันใดบ้างในหนึ่งปี ด้วยการตรวจสอบทีละวันว่าวันใดเป็นวันหยุด เป็นต้น

เพื่อให้เกิดความเข้าใจให้ศึกษาตัวอย่างต่อไปนี้

ตัวอย่างที่ 1.8 การออกแบบรายการ ADT จากฟังก์ชันตรวจสอบวันหยุดที่มีคำสั่งดังนี้

```
+listHolidays(in year:integer)
//แสดงวันหยุดทั้งหมดในหนึ่งปีของ year ให้มา
    date = วันที่ของวันแรกของปี
    while(date ไม่ใช่วันแรกของปีถัดไป)
            if(date = วันหยุด){
                write(date + "คือวันหยุด")
            } //end if
       date = วันที่ถัดไป
    }//end while
```

จากฟังก์ชันสามารถเขียนในรูปแบบ ADT เพื่อใช้ในการแก้ไขปัญหาการตรวจสอบวันหยุดในหนึ่งปีได้ดังนี้

1. Determine the date of the first day of a given year. (สนใจวันแรกของปีที่ให้มา)
2. Determine whether a date is before another date. (สนใจวันที่ในการตรวจสอบก่อนวันที่อื่นๆ)
3. Determine whether a date is a holiday. (สนใจวันที่ที่เป็นวันหยุด)
4. Determine the date of the day that follows a given date. (สนใจวันที่ถัดไปของวันที่กำหนดให้มา)

จาก ADT การตรวจสอบวันหยุดในหนึ่งปีนำไปเขียนเป็น Pseudo code ได้ดังนี้

```
+firstDay(in year:integer):Date{query}
//คืนค่าวันแรกของปีที่ให้มา
+isBefore(in date1:Date,date2:Date):boolean{query}
//คืนค่า true ถ้า date1 เป็นวันที่มาก่อน date2 นอกเหนือจากนั้นให้คืนค่า false
+isHoliday(in aDate:Date):boolean{query}
//คืนค่า true ถ้า aDate เป็นวันหยุด นอกเหนือจากนั้นให้คืนค่า false
+nextDay(in aDate:Date):Date
//คืนค่าวันที่ถัดไปจากวันที่กำหนดให้มาในตัวแปร aDate
```

ดังนั้น การออกแบบ ADT จะเป็นการแสดงข้อมูลและการเลือกตัวดำเนินการให้เหมาะสมกับการแก้ไขปัญหา โดยที่ตัวดำเนินการนั้นมีการทำงานที่อิสระในการกำหนดคุณสมบัติของ ADT

ประเภทของอัลกอริทึม

การแยกประเภทของอัลกอริทึมเหมือนกับการแยกประเภทขั้นตอนการแก้ไขปัญหา โดยอธิบายการทำงานของอัลกอริทึมแต่ละประเภทได้ดังนี้

- **Brute Force algorithm** เป็นการแก้ไขปัญหาโดยสั่งให้ทำงานไปเรื่อยๆ จนกระทั่งได้คำตอบของทุกปัญหา การแก้ไขในรูปแบบนี้ไม่เหมาะสำหรับแก้ไขปัญหาที่มีข้อมูลจำนวนมาก ตัวอย่างอัลกอริทึมที่ใช้หลักการ Brute Force algorithm เช่น การจัดเรียงข้อมูลแบบ Bubble sort, Selection sort เป็นต้น
- **Divide and Conquer algorithm** เป็นอัลกอริทึมที่มีหลักการคิดโดยแยกปัญหาออกเป็นสองส่วน คือ ส่วนที่หนึ่งแบ่งปัญหาออกเป็นส่วนเล็กๆ แล้วแก้ไขปัญหาในส่วนเล็กๆ นั้นก่อน และอีกส่วนนำผลที่ได้จากการแก้ไขปัญหาในส่วนเล็กๆ กลับมารวมกันใหม่ ตัวอย่างอัลกอริทึมที่ใช้หลักการ Divide and Conquer algorithm เช่น การจัดเรียงข้อมูลแบบ Quick sort, Merge sort เป็นต้น
- **Decrease and Conquer algorithm** เป็นอัลกอริทึมที่แก้ไขปัญหาด้วยการลดขนาดของปัญหาลง และเลือกขนาดของกลุ่มปัญหาที่ต้องการแก้ไขปัญหา โดยละเว้นปัญหาบางส่วนไว้ก่อนเพื่อจะแก้ปัญหาที่มีขนาดเล็กลงกว่าเดิม เนื่องจากการแก้ไขปัญหาที่มีขนาดเล็กกว่าจะสามารถแก้ไขปัญหาได้ง่ายกว่า ตัวอย่างอัลกอริทึมที่ใช้หลักการ Decrease and Conquer algorithm เช่น การค้นหาข้อมูลแบบไบนารี เป็นต้น
- **Transform and Conquer algorithm** เป็นการแก้ไขปัญหาด้วยการเปลี่ยนรูปแบบของปัญหาที่ต้องการแก้ไขให้อยู่ในรูปแบบอื่นก่อน ด้วยคาดหวังว่าเมื่อเปลี่ยนรูปแบบของปัญหาแล้วจะสามารถแก้ไขปัญหาได้ง่ายและรวดเร็วขึ้น ตัวอย่างในการเปลี่ยนโครงสร้างข้อมูลก่อนการแก้ไข

ปัญหา เช่น นำข้อมูลที่ต้องการค้นหามาจัดเรียงข้อมูลก่อนที่จะค้นหา เพื่อให้สามารถใช้อัลกอริทึมที่มีประสิทธิภาพค้นหาข้อมูลได้ แทนที่จะค้นหาข้อมูลทีละตัวจากข้อมูลที่กระจัดกระจาย

- **Greedy algorithm** หรือ**อัลกอริทึมแบบละโมบ** เป็นอัลกอริทึมที่มีลักษณะของการแก้ไขปัญหาด้วยการเพิ่มประสิทธิภาพของการแก้ไขปัญหาให้เหมาะสมที่สุด (Optimization problems) ซึ่งเป็นรูปแบบอัลกอริทึมที่พิจารณาคำตอบที่ดีที่สุดและคุ้มค่าที่สุดในการแก้ไขปัญหานั้นๆ ตัวอย่างอัลกอริทึมที่ใช้หลักการ Greedy algorithm ในการแก้ไขปัญหา เช่น ปัญหาการทอนเหรียญคือเลือกทอนเหรียญในหน่วยที่มีขนาดใหญ่ที่สุดก่อน เป็นต้น
- **Dynamic programming algorithm** หรือ**อัลกอริทึมโปรแกรมพลวัต** เป็นอัลกอริทึมที่มีลักษณะของการแก้ไขปัญหาด้วยการแบ่งปัญหาเป็นส่วนเล็กๆ แล้วนำผลของปัญหาเล็กๆ ที่ดีที่สุดนำมาแก้ไขปัญหาใหญ่ที่เรียกกันว่า การแก้ไขปัญหาจากล่างขึ้นบน **(Bottom-up approach)** ตัวอย่างอัลกอริทึมที่ใช้หลักการ Dynamic programming algorithm ในการแก้ไขปัญหา เช่น การหาค่าตัวเลข Fibonacci เป็นต้น
- **Backtracking algorithm** หรือ**อัลกอริทึมย้อนรอยถอยหลัง** เป็นอัลกอริทึมค้นหาเส้นทางทุกเส้นทางที่เป็นไปได้ เพื่อหาคำตอบของปัญหาทีละส่วนย่อยว่า คำตอบนั้นเป็นคำตอบที่ถูกต้องหรือไม่ แต่ถ้าคำตอบนั้นไม่ใช่ส่วนหนึ่งของคำตอบจะถอยหลังกลับมาจุดเดิม และยกเลิกคำตอบนั้นแล้วค้นหาคำตอบใหม่ ตัวอย่างอัลกอริทึมที่ใช้หลักการ Backtracking algorithm เช่น การกำหนดสีให้กับเมืองในแผนที่, การคิดความเป็นไปได้ทั้งหมดของการเดินหมากกระดาน เป็นต้น
- **Branch and bound algorithms** เป็นอัลกอริทึมที่เพิ่มประสิทธิภาพในการแก้ไขปัญหาด้วยการนำโครงสร้างทรี (Tree) มาเก็บปัญหาย่อยๆ โดยที่ปัญหาหลักจะอยู่ในตำแหน่งบนสุดของทรี คือ โหนดราก (root node) และในแต่ละโหนดจะแก้ไขปัญหาของตัวเอง และถ้าแก้ปัญหาถูกต้องจะใช้ผลนั้นเป็นข้อมูลในการแก้ไขปัญหาทั้งหมด แต่ถ้าการแก้ไขปัญหาไม่ถูกต้องให้แบ่งปัญหาออกเป็นสองโหนดย่อย เก็บไว้ในตำแหน่งโหนดลูกของโหนดที่แก้ไขปัญหาไม่ถูกต้อง แล้วกลับไปทำใหม่ จนกระทั่งทุกโหนดย่อยในทรีสามารถแก้ไขปัญหาได้ทุกโหนด ตัวอย่างอัลกอริทึมที่ใช้หลักการ Branch and bound algorithms มาใช้ในการแก้ไขปัญหา เช่น ปัญหาในการหาเส้นทางที่เหมาะสมให้กับพนักงานขายสินค้าให้สามารถเดินทางได้ครบทุกที่ได้เร็วที่สุด เป็นต้น
- **Recursive algorithm** หรือ**อัลกอริทึมแบบวนซ้ำ** เป็นการแก้ไขปัญหาขั้นพื้นฐานด้วยการเรียกใช้ตัวเองซ้ำๆ โดยนำข้อมูลปัญหาส่วนย่อยของปัญหาทั้งหมดกลับมาเป็นข้อมูลในการแก้ไขปัญหา ตัวอย่างอัลกอริทึมแบบเรียกซ้ำ เช่น การหาค่า Factorial, อัลกอริทึมบวกข้อมูลตัวเลขที่อยู่ในกลุ่ม เป็นต้น
- **Randomized algorithms** หรือ**อัลกอริทึมแบบสุ่ม** เป็นอัลกอริทึมที่ใช้หลักการสุ่มข้อมูล แล้วนำข้อมูลที่สุ่มเลือกขึ้นมาได้กระทำกับอัลกอริทึมเพื่อให้ได้ผลตามที่ต้องการ ตัวอย่างอัลกอริทึมที่ใช้หลักการ Randomized algorithms ไปใช้งาน เช่น พยายามหาข้อมูลที่สำคัญที่สุดด้วยการเลือกข้อมูลจากการสุ่มด้วยการหาร หรือการจัดเรียงข้อมูลแบบ Quick sort ด้วยการสุ่มตัวเลขที่ใช้เป็นข้อมูลเพื่อใช้ในการเปรียบเทียบ (pivot) ในการจัดเรียงข้อมูล เป็นต้น

สรุปเนื้อหาบทที่ 1

- โครงสร้างข้อมูลคือ การจัดการข้อมูลในหน่วยความจำภายในเครื่องคอมพิวเตอร์
- อัลกอริทึมคือ ลำดับขั้นตอนการทำงานเพื่อใช้ในการแก้ไขปัญหา
- ผังงาน เป็นเครื่องมือที่ช่วยออกแบบขั้นตอนการทำงานด้วยสัญลักษณ์
- โค้ดรหัสเทียม เป็นโครงสร้างรหัสที่มีการรวมกันทั้งภาษาเขียนกับภาษาคอมพิวเตอร์ เพื่อใช้ในการอธิบายโครงสร้างและลำดับขั้นตอนการทำงานของโปรแกรม โดยไม่อ้างอิงภาษาการเขียนโปรแกรมภาษาใดภาษาหนึ่ง
- Abstract Data Type เป็นการบอกถึงคุณสมบัติของโครงสร้างข้อมูลและกลุ่มตัวดำเนินการที่กระทำกับโครงสร้างข้อมูล
- ประเภทของอัลกอริทึมแยกได้เหมือนกับแยกรูปแบบในการแก้ไขปัญหาของโปรแกรม การแยกประเภทของอัลกอริทึมเพื่อแยกรูปแบบอัลกอริทึมในการแก้ไขปัญหานั่นเอง

แบบฝึกหัดท้ายบทที่ 1

ให้เขียนขั้นตอนวิธี ผังงาน และโค้ดรหัสเทียมของโปรแกรมต่อไปนี้

1. คำนวณเลขยกกำลังกำหนดให้รับตัวเลข 2 ค่าคือ ค่าเลขฐานและเลขยกกำลัง เพื่อนำมาคำนวณเลขยกกำลัง
2. เปลี่ยนค่าหน่วยเวลาจากหน่วยวินาทีที่รับเข้ามาให้เป็นหน่วยชั่วโมง นาที และวินาที
3. คำนวณอัตราแลกเปลี่ยนจากข้อมูลตัวเลขที่ป้อนเข้ามา กำหนดให้เป็นสกุลเงินบาทเปลี่ยนเป็นสกุลเงินประเทศสหรัฐอเมริกา ญี่ปุ่น และจีน กำหนดให้อ้างอิงอัตราแลกเปลี่ยนปัจจุบันในการซื้อขาย
4. รับข้อมูลตัวเลขจำนวน 20 ตัว กำหนดให้นำเฉพาะข้อมูลที่เป็นเลขคู่มาบวกกันและแสดงผลการบวก
5. คำนวณการถอนเงิน โดยให้รับข้อมูลตัวเลขเข้ามา 2 จำนวนคือ ยอดเงินที่ลูกค้าซื้อและจำนวนเงินที่ลูกค้าจ่าย แล้วให้คำนวณว่าต้องถอนแบงก์ 1,000 500 100 50 20 เหรียญ 10 5 2 1 .50 และ .25 บาท จำนวนอย่างละเท่าไร
6. รับคะแนนสอบของนักศึกษาจำนวน 20 คนว่ามีนักศึกษาที่ได้เกรดแต่ละเกรดจำนวนกี่คน และหาค่าเฉลี่ยว่านักศึกษากลุ่มนี้มีค่าคะแนนเฉลี่ยอยู่ที่เกรดอะไร โดยกำหนดค่าคะแนนในแต่ละช่วงเกรดดังนี้

คะแนน	เกรด
ต่ำกว่า 50	F
50-54	D
55-59	D+
60-64	C
65-69	C+
70-74	B
75-79	B+
80 ขึ้นไป	A

7. รับข้อมูลอายุจำนวน 20 คน กำหนดให้นับจำนวนคนในแต่ละช่วงวัยและหาค่าเฉลี่ยจากอายุที่รับเข้ามาว่าอยู่ในช่วงวัยไหน โดยแยกแต่ละช่วงวัยดังนี้
   - วัยเด็กอายุน้อยกว่า 15 ปี
   - วัยรุ่นอายุมากกว่าและเท่ากับ 15 ปี ถึง 20 ปี
   - วัยผู้ใหญ่ตอนต้นอายุมากกว่า 20 ปี ถึง 40 ปี
   - วัยกลางคนอายุมากกว่า 40 ปี ถึง 60 ปี
   - วัยชราอายุมากกว่า 60 ปีขึ้นไป
8. รับข้อมูลตัวเลขจำนวนชั่วโมงการทำงานในแต่ละสัปดาห์จำนวน 4 สัปดาห์และประเภทของงานที่ทำงานเพื่อคำนวณค่าจ้าง โดยกำหนดให้ประเภทของงานและอัตราค่าจ้างดังนี้

ลำดับประเภทงาน	ชื่อประเภทงาน	อัตราค่าจ้าง/ชั่วโมง
0	แรงงานระดับใช้กำลัง	20
1	แรงงานระดับฝีมือ	25
2	แรงงานระดับชำนาญเฉพาะด้าน	30
3	แรงงานระดับฝีมือชำนาญเฉพาะด้าน	35

9. หาผลคูณจากเมตริกซ์ 2 เมตริกซ์ กำหนดเมตริกซ์ที่ 1 มีขนาด n x m และเมตริกซ์ที่ 2 มีขนาด m x n ผลลัพธ์ของการคูณเมตริกซ์ทั้งสองมีขนาด n x n

Data Structure & Algorithm

CHAPTER 02

การวิเคราะห์ประสิทธิภาพของอัลกอริทึม (Performance Analysis)

ในบทนี้เราจะมาศึกษาการวิเคราะห์ประสิทธิภาพของอัลกอริทึม (Performance Analysis) ซึ่งเป็นพื้นฐานที่มีความสำคัญเป็นอย่างยิ่งในการเรียนรู้และใช้งานโครงสร้างข้อมูลและอัลกอริทึม

คณิตศาสตร์พื้นฐานสำหรับการวิเคราะห์ประสิทธิภาพของอัลกอริทึม

สมการคณิตศาสตร์ที่ใช้ในการวิเคราะห์ประสิทธิภาพของอัลกอริทึมมี 3 รูปแบบคือ ลอการิทึม, การหาผลรวม และเลขยกกำลัง โดยสรุปได้ดังนี้

ลอการิทึม (Logarithms)

- ลอการิทึมของค่า y ฐาน b จะเขียนอยู่ในรูปลอการิทึมได้คือ $log_b y = x$
  ดังนั้น ถ้า $\log_b y = x$ แล้ว $b^x = y$ และ $b^{\log_b y} = y$
  ตัวอย่างเช่น $\log_2 1000 = 10$ หาได้จาก $2^{10} = 1000$
- คุณสมบัติของลอการิทึมมีดังนี้
  1. $\log(nm) = \log n + \log m$
  2. $\log(n/m) = \log n - \log m$
  3. $\log(n^r) = r \log n$
  4. $\log_a n = \frac{\log_b n}{\log_b a}$

ผลรวม (Summation)

ผลรวมเป็นการบวกตัวเลขที่อยู่ในช่วงของกลุ่มข้อมูล โดยใช้สัญลักษณ์ซิกมา ($\sum$: Sigma) ซึ่งมีรูปแบบดังนี้

$$\sum_{i=1}^{n} f(i) = f(1) + f(2) + \ldots + f(n-1) + f(n)$$

เลขยกกำลัง (Exponentiation)

เลขยกกำลังในรูปแบบของโครงสร้างข้อมูล จะใช้เพียงการบวกและคูณเลขยกกำลัง

- การบวกเลขยกกำลัง ตัวอย่างเช่น

$$n^2 + 3n + n^2 + n + 1 = (n^2 + n^2) + (3n + n)+1 = 2n^2 + 4n + 1$$

- การคูณเลขยกกำลัง ตัวอย่างเช่น

$$n*(n + 1) = n^2 + n$$

การวัดประสิทธิภาพอัลกอริทึม

วัตถุประสงค์แรกในการพัฒนาโปรแกรมขึ้นมาคือ โปรแกรมที่พัฒนาขึ้นมาต้องทำงานถูกต้องและได้ผลลัพธ์ตามต้องการ เมื่อโปรแกรมสามารถทำงานได้อย่างถูกต้องแล้ว สิ่งที่ต้องพิจารณาตามมาก็คือ ประสิทธิภาพการทำงานของโปรแกรมว่าใช้เวลาในการประมวลผลนานเกินไปหรือไม่ และใช้หน่วยความจำของเครื่องคอมพิวเตอร์มากเกินความจำเป็นหรือไม่ ซึ่งเป็นกระบวนการในการวัดประสิทธิภาพของอัลกอริทึม

เครื่องมือที่ใช้ในการวัดประสิทธิภาพอัลกอริทึมมีอยู่ 2 เครื่องมือคือ

- การวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล (Space Complexity Analysis)
- การวิเคราะห์เวลาที่ใช้ในการประมวลผล (Time Complexity Analysis)

เครื่องมือทั้งสองเครื่องมือนี้ เป็นตัวตัดสินประสิทธิภาพการทำงานของอัลกอริทึมนั้นๆ ว่ามีประสิทธิภาพมากน้อยเพียงใด

การวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล (Space Complexity Analysis)

การวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล (Space Complexity Analysis) คือ การวิเคราะห์หน่วยความจำทั้งหมดที่โปรแกรมใช้ในการประมวลผลของอัลกอริทึม เพื่อให้ทราบถึงขนาดข้อมูลที่สามารถป้อนหรือส่งข้อมูลเข้ามาให้อัลกอริทึมประมวลผลแล้วไม่เกิดข้อผิดพลาด หรือเพื่อให้ทราบถึงขนาดหน่วยความจำของเครื่องคอมพิวเตอร์ว่า มีเพียงพอสำหรับการเก็บข้อมูลทั้งหมดในขณะประมวลผลอัลกอริทึมหรือไม่ และจะมีผลอย่างมากในการทำงานผ่านเครือข่าย เนื่องจากจะทำให้สิ้นเปลืองทรัพยากรและช่องสัญญาณของเครือข่าย

องค์ประกอบของการวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล

ในหัวข้อนี้จะอธิบายถึงองค์ประกอบในการวิเคราะห์หน่วยความจำที่ใช้ในการประมวลผล ซึ่งมีอยู่ 3 ส่วนดังนี้

- **Instruction Space** คือ ขนาดหน่วยความจำที่จำเป็นต้องใช้ขณะคอมไพเลอร์ (Compiler) โปรแกรม
- **Data Space** คือ ขนาดหน่วยความจำที่จำเป็นต้องใช้สำหรับเก็บข้อมูลค่าคงที่ และตัวแปรที่ใช้ในขณะประมวลผลโปรแกรม ซึ่งแยกออกได้ 2 ประเภทคือ
  - **Static memory** เป็นขนาดหน่วยความจำที่ต้องใช้ในการประมวลผลอย่างแน่นอน เช่น หน่วยความจำที่ใช้เก็บค่าคงที่ หรือตัวแปรชนิดอาร์เรย์ เป็นต้น

- **Dynamic memory** เป็นขนาดหน่วยความจำที่ต้องใช้ในการประมวลผลที่ไม่แน่นอนคือ จะรู้ว่าต้องใช้หน่วยความจำเท่าไรก็ต่อเมื่อโปรแกรมต้องใช้งานนั่นเอง เช่น การประกาศตัวแปรพอยน์เตอร์ (Pointer) ในภาษา C หรือการเก็บข้อมูลในรูปแบบลิงค์ลิสต์ที่สามารถเพิ่มหรือลดขนาดการเก็บข้อมูลได้แบบอัตโนมัติ โดยไม่ต้องจองพื้นที่หน่วยความจำไว้ก่อนใช้งาน เป็นต้น

- **Environment Stack Space** คือ หน่วยความจำที่ใช้สำหรับเก็บข้อมูลผลลัพธ์ที่ได้จากการประมวลผล เพื่อรอเวลาที่จะนำกลับไปใช้ใหม่ในโปรแกรม ซึ่งหน่วยความจำประเภทนี้จะเกิดขึ้นเมื่อมีการใช้งานเท่านั้น

ตัวอย่างที่ 2.1 วิเคราะห์หน่วยความจำที่ใช้ในการประมวลผลของโค้ดต่อไปนี้

```
1  +SpaceAnalysis(in data1:int,in data2:int):int
2       int temp
3       temp = data1 + data2
4       return temp
```

โปรแกรมนี้มีการประกาศตัวแปรทั้งหมด 3 ตัวแปรคือ data1, data2 และ temp ทั้งสามตัวแปรมีชนิดข้อมูลชนิดเดียวกันคือ integer ซึ่งเป็นชนิดข้อมูลที่ใช้พื้นที่หน่วยความจำเท่ากับ 4 ไบต์ (ภาษา Java ใช้พื้นที่หน่วยความจำ 4 ไบต์, ภาษา C ใช้พื้นที่หน่วยความจำ 2 ไบต์)

ดังนั้น โปรแกรมนี้ใช้พื้นที่หน่วยความจำเท่ากับ 3×4 = 12 ไบต์ (ภาษา C ใช้หน่วยความจำ 3×2 = 6 ไบต์)

ตัวอย่างที่ 2.2 วิเคราะห์หน่วยความจำที่ใช้ในการประมวลผลของโปรแกรมแบบวนซ้ำ ซึ่งมีโค้ดต่อไปนี้

```
1  +Factorial(in n:int):int
2      if (n == 0)
3         return 1
4      else return n * (Factorial(n-1))
```

โปรแกรมนี้มีการประกาศตัวแปร n เพียงตัวแปรเดียว โดยมีชนิดข้อมูลของตัวแปรคือ integer ซึ่งใช้พื้นที่หน่วยความจำเท่ากับ 4 ไบต์ (ภาษา Java)

เมื่อพิจารณาการใช้หน่วยความจำของโปรแกรมแบบวนซ้ำ จะต้องพิจารณาตามจำนวนรอบที่วนซ้ำ ถ้ากำหนดให้ตัวแปร n มีค่าเท่ากับ 3 จะต้องวนซ้ำ 3 ครั้ง ในกรณีนี้จะใช้พื้นที่ในหน่วยความจำเท่ากับ 3×4 = 12 ไบต์ (ภาษา Java)

สรุปได้ว่าถ้ากำหนดให้หาค่าแฟ็กทอเรียล n จะต้องใช้พื้นที่หน่วยความจำเท่ากับ n * 4 ไบต์ (ภาษา Java) ในการคำนวณหาค่าแฟ็กทอเรียล n

การวิเคราะห์เวลาที่ใช้ในการประมวลผล (Time Complexity Analysis)

การวิเคราะห์เวลาที่ใช้ในการประมวลผล (Time Complexity Analysis) คือ การวิเคราะห์ขั้นตอนของการดำเนินการที่ต้องใช้ในการประมวลผลของอัลกอริทึม เพื่อใช้ประมาณการเวลาที่ใช้ประมวลผล ทำให้ทราบถึงประสิทธิภาพการทำงานของโปรแกรมและการแก้ไขปัญหาได้อย่างถูกต้อง อีกทั้งสามารถนำไปใช้เลือกเครื่องคอมพิวเตอร์ที่เหมาะสมกับการประมวลผลของอัลกอริทึมได้อีกด้วย

หลักการพิจารณาเวลาที่ใช้ประมวลผล

สิ่งหนึ่งที่ผู้อ่านจะต้องทำความเข้าใจก่อนวิเคราะห์เวลาที่ใช้ในการประมวลผลมีดังนี้

- เมื่อประมวลผลโปรแกรมในเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพเร็วกว่า ก็จะประมวลผลข้อมูลได้เร็วกว่า
- เมื่อรันโปรแกรมที่มีผลการทำงานเดียวกัน โปรแกรมที่มีโค้ดน้อยกว่าจะทำงานได้เร็วกว่า
- ตัวแปรที่มีขนาดหน่วยความจำเล็กกว่า จะประมวลผลได้เร็วกว่า

ประเภทของเวลาที่ใช้ในการประมวลผล

เวลาที่ใช้ในการประมวลผลโปรแกรมสามารถแบ่งออกได้ 2 ประเภทคือ

- **Compile time** เป็นเวลาที่ใช้ในการตรวจไวยากรณ์ (Syntax) ของโค้ดโปรแกรมว่าเขียนถูกต้องตามหลักโครงสร้างภาษาที่ใช้เขียนโปรแกรมหรือไม่
- **Run time** หรือ **Execution time** เป็นเวลาที่เครื่องคอมพิวเตอร์ใช้ในการประมวลผลอัลกอริทึม ซึ่งขึ้นอยู่กับชนิดข้อมูล จำนวนตัวแปรที่ใช้ในโปรแกรม และจำนวนลูปที่ใช้ในการประมวลผล

ตัวอย่างที่ 2.3 วิเคราะห์เวลาที่ใช้ในการประมวลผลของโค้ดต่อไปนี้

```
1  int n = 20;                                  ← กำหนดค่า 1 ครั้ง
2  int total = 0;                               ← กำหนดค่า 1 ครั้ง
3  while(n != 20){                              ← เปรียบเทียบ n+1 ครั้ง
4           total += n;                         ← คำนวณ n ครั้ง
5     ++n;                                      ← คำนวณ n ครั้ง
6  } //end while
7  System.out.println("ผลรวม = " + total);     ← แสดงผล 1 ครั้ง
```

บรรทัดที่ 1-2 ทำงานบรรทัดละ 1 ครั้ง เป็นการกำหนดค่าและชนิดข้อมูลให้กับตัวแปร

บรรทัดที่ 3 ทำงาน n+1 ครั้ง เพื่อวนรอบตรวจสอบตัวแปร n ตั้งแต่ 0 ถึง 19 เท่ากับ 20 รอบ คือ เท่ากับ n ส่วนที่ต้องทำอีก +1 คือ ตรวจสอบที่ n = 20 เพื่อจบการวนลูป ดังนั้น ในบรรทัดนี้ทำงานทั้งหมด n+1 ครั้ง

บรรทัดที่ 4-5 ทำงาน n ครั้งคือ ทำงานเท่ากับจำนวนรอบที่วนลูป

บรรทัดที่ 7 ทำงาน 1 ครั้งคือ แสดงผล

ถ้ากำหนดให้ f(n) แทนประสิทธิภาพในการวิเคราะห์เวลาที่ใช้ในการประมวลผล และ n แทนจำนวนรอบในการทำงาน จะได้ว่าอัลกอริทึมนี้มีประสิทธิภาพดังนี้

$$f(n) = 1 + 1 + (n + 1) + n + n + 1 = 3n + 4$$

ตัวอย่างที่ 2.4 วิเคราะห์เวลาที่ใช้ในการประมวลผลของโปรแกรมแบบวนซ้ำ ซึ่งมีโค้ดต่อไปนี้

```
1  +Factorial(in n:int):int                     ← ถูกเรียกใช้ n ครั้ง
2     if (n == 0)                               ← ตรวจสอบเงื่อนไข n ครั้ง
3        return 1                               ← คืนค่า 1 ครั้ง
4     else return n * (Factorial(n-1))          ← เรียกใช้ตัวเอง n ครั้ง
```

บรรทัดที่ 1 ฟังก์ชันจะถูกเรียกใช้ตัวเอง n ครั้ง

บรรทัดที่ 2 ตรวจสอบว่า n = 0 หรือไม่ ในบรรทัดนี้ต้องตรวจสอบ n ครั้ง

บรรทัดที่ 3 ทำงาน 1 ครั้ง เพื่อคืนค่าเมื่อ n = 0

บรรทัดที่ 4 เรียกใช้งานตัวเองจำนวน n ครั้ง

ถ้ากำหนดให้ f(n) แทนประสิทธิภาพในการวิเคราะห์เวลาที่ใช้ในการประมวลผล และ n แทนจำนวนรอบในการทำงาน จะได้ว่าอัลกอริทึมนี้มีประสิทธิภาพดังนี้

$$f(n) = n + n + 1 + n = 3n + 1$$

อัตราการเติบโตของอัลกอริทึม (Algorithm Growth Rates)

การวัดประสิทธิภาพด้วยอัตราการเจริญเติบโตของอัลกอริทึม เป็นการวิเคราะห์อัลกอริทึมโดยใช้เครื่องหมายมาเป็นเครื่องมือในการวัดประสิทธิภาพของอัลกอริทึม เช่น Big-O, Big-Omega, Big-Theta, Little-o และ Little-omega เป็นต้น

อัตราการเติบโต Big-O (O)

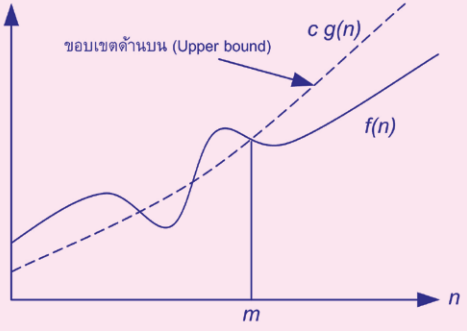
อัตราการเติบโต Big-O เป็นการวัดประสิทธิภาพเชิงเวลาที่ใช้ในการประมวลผลของอัลกอริทึม ซึ่งเป็นฟังก์ชันเวลาขอบเขตบนที่ใช้ในการประมวลผล (Asymptotic upper bounds) โดยมีสัญลักษณ์เป็นตัวโอใหญ่ (O)

ดังนั้น หากใช้สัญลักษณ์ Big-O จะเป็นการรับรองว่าเวลาที่ใช้ในการประมวลผลสูงสุดของอัลกอริทึมนี้จะไม่มากกว่าเวลาที่คำนวณได้จากฟังก์ชัน Big-O เช่น O(n) หมายถึง ฟังก์ชันนี้จะใช้เวลาในการประมวลผลน้อยกว่าหรือเท่ากับ n (≤n) เสมอ

นิยาม Big-O

ฟังก์ชัน $f(n) = O(g(n))$ ก็ต่อเมื่อมีค่าคงที่ m, c ที่ทำให้ $f(n) \leq cg(n)$ เมื่อ $n \geq m$

รูปที่ 2-1
กราฟนิยามอัตราการเติบโต Big-O



จากนิยามสามารถอธิบายได้ว่า ฟังก์ชัน f จะมีอัตราการเติบโตไม่มากไปกว่าฟังก์ชัน g เมื่อค่า n มีค่ามากกว่าหรือเท่ากับ m ดังกราฟที่แสดงให้เห็นได้ว่าฟังก์ชัน f(n) ในตำแหน่งที่ m จะเป็นตำแหน่งที่ค่าเวลาในการประมวลผลไม่มากกว่าฟังก์ชัน cg(n) เสมอดังตัวอย่างต่อไปนี้