บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ได้สรุปพื้นฐานความรู้ที่สำคัญเกี่ยวกับ ตัวแบบการบริหารระบบ และกรณีตัวอย่าง (system administration model) การเจริญเติบโตของฟังก์ชัน (growth of function) สัญลักษณ์แบบอะซิมโทติก (asymptotic notation) การนับเบื้องต้น (counting) ทฤษฎีความซับซ้อน (complexity theory) และขั้นตอนวิธีเชิงประมาณ (approximation algorithm) ซึ่งใช้ในการศึกษาความซับซ้อนของปัญหาการคัดลอกข้อมูลที่น้อยที่สุดในบทต่อไป

2.1 ตัวแบบการบริหารระบบ (System Administration Model)

ในส่วนนี้จะนำเสนอตัวแบบการบริหารระบบ ซึ่งตัวแบบนี้ได้ออกแบบมาเพื่อเลียนแบบระบบคอมพิวเตอร์ที่ทำงานจริงในปัจจุบันและรวมถึงสภาวะแวดล้อมของระบบด้วย แต่ถึงอย่างไรก็ตาม ก็ไม่มีตัวแบบตัวไหนที่จะอธิบายสภาวะจริงของระบบคอมพิวเตอร์ได้อย่างสมบูรณ์ ดังนั้นตัวแบบที่นำเสนอในวิทยานิพนธ์ฉบับนี้จึงเป็นการนำเสนอบางส่วนของระบบจริงที่ต้องการศึกษา ซึ่งจากนี้ไปจะนำเสนอนิยามของตัวแบบและนิยามที่เกี่ยวข้อง สำหรับเนื้อหาตั้งแต่นี้เป็นต้นไป กำหนดให้ $\mathbb{N} = \{1, 2, 3, \dots\}$ เป็นเซตของจำนวนธรรมชาติ

นิยาม 2.1 ตัวแบบการบริหารระบบ (System Administration Model)

กำหนดให้ n e m r f s และ d เป็นจำนวนธรรมชาติใดๆ ตัวแบบการบริหารระบบ M เป็น 7 คู่อันดับ (C, L, U, P, S, O, A) โดยที่

1. $C = \{c_1, c_2, c_3, \dots, c_n\}$ เป็นเซตจำกัดของ n **คอมพิวเตอร์** (computer)
2. $L = \{l_1, l_2, l_3, \dots, l_e\}$ เป็นเซตจำกัดของ e **ข่ายเชื่อมโยงการสื่อสาร** (connection link) ระหว่างคอมพิวเตอร์ 2 เครื่องใดๆ และแต่ละ $l_k \in L$ ให้ $l_k = \{c_i, c_j\}$ โดยที่ c_i และ c_j เป็นสมาชิกของ C ซึ่ง $i \neq j$
3. $U = \{u_1, u_2, u_3, \dots, u_m\}$ เป็นเซตจำกัดของ**ผู้ใช้** m คนในระบบ (user)
4. $P = \{p_1, p_2, p_3, \dots, p_r\}$ เป็นเซตจำกัดของกฎ r ข้อ ซึ่งเซตนี้เรียกว่า **นโยบาย** (policy)
5. $S = \{s_1, s_2, s_3, \dots, s_f\}$ เป็นเซตจำกัดของ**สถานะ**ที่สนใจของระบบ f สถานะ (state)
6. $O = \{o_1, o_2, o_3, \dots, o_s\}$ เป็นเซตจำกัดของ**ตัวดำเนินการ** (operator) โดยที่แต่ละ o_k เป็นฟังก์ชันที่ซึ่ง $o_k: S \mapsto S$ สำหรับ $1 \leq k \leq s$

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

7. $A = \{a_1, a_2, a_3, \ldots, a_d\}$ เป็นเซตจำกัดของ**การกระทำของผู้ใช้** (user activity) ซึ่งแต่ละ a_k เป็น 3 คู่อันดับ (triple) โดยที่ $a_k \in U \times O \times \mathbb{N}$ สำหรับ $1 \leq k \leq d$

ข้อสังเกตจากตัวแบบ ข้อแรกจะเห็นว่าเมื่อรวมคอมพิวเตอร์และข่ายเชื่อมโยงการสื่อสารเข้าไปในตัวแบบ ทำให้คอมพิวเตอร์แต่ละเครื่องได้ถูกเชื่อมโยงเข้าด้วยกันซึ่งจะกำหนดนิยามต่อไปในนิยาม 2.2 สังเกตว่าส่วนประกอบของตัวแบบทั้งสองนี้ทำให้เกิดเป็นโครงข่ายคอมพิวเตอร์ ข้อที่สองคือตัวแบบได้กำหนดเซตของผู้ใช้ไว้ ผู้ใช้แต่ละคนสามารถใช้เครื่องคอมพิวเตอร์ในระบบตามสิทธิ์ที่ได้รับ ข้อสังเกตต่อมาคือเซตของกฎสำหรับระบบ หรือนโยบายที่ทำหน้าที่บังคับให้ระบบทำงานอยู่ในกรอบที่สามารถพยากรได้ ข้อสังเกตที่สี่ แต่ละส่วนประกอบของระบบต่างมีสถานะของตนเอง ซึ่งในตัวแบบที่นำเสนอได้กำหนดเซตของสถานะที่สนใจศึกษาไว้ ข้อที่ห้าการทำงานของผู้ใช้ หรือแม้กระทั่งผู้บริหารระบบเองส่วนมากจะทำงานกับตัวระบบ โดยใช้ตัวดำเนินการต่างๆ ในตัวแบบที่นำเสนอจึงได้กำหนดเซตของตัวดำเนินการเหล่านี้ให้เป็นพารามีเตอร์หนึ่ง ข้อสังเกตสุดท้ายคือ การกระทำของผู้ใช้แต่ละคนได้กำหนดไว้ในตัวแบบโดยที่ตัวแปร $\mathbb{N}$ หมายถึงขั้นเวลาที่ไม่ต่อเนื่องสำหรับแต่ละการกระทำของผู้ใช้

ต่อไปจะนำเสนอนิยามของการสื่อสาร โดยที่คอมพิวเตอร์สองเครื่องใดๆ จะติดต่อสื่อสารกันได้ก็ต่อเมื่อมี*การเชื่อมโยง*ระหว่างสองเครื่องนั้น ซึ่งการเชื่อมโยงในที่นี้มีนิยามดังต่อไปนี้

นิยาม 2.2 สภาพเชื่อมโยง (connectivity)

คอมพิวเตอร์ x y ใดๆ ที่เป็นสามาชิกของเซตคอมพิวเตอร์ C จะเรียกว่าเชื่อมโยงกัน ก็ต่อเมื่อ มีลำดับ $< c_1, c_2, c_3, \ldots, c_k >$ ซึ่งและ $c_1 = x$ และ $c_k = y$ โดยที่ $\{ c_i, c_{i+1} \} \in L$ สำรับทุก $1 \leq i \leq k$-1

2.2 กรณีตัวอย่างของตัวแบบการบริหารระบบ (Instance of System Administration Model)

เพื่อให้เข้าใจตัวแบบการบริหารระบบได้ดียิ่งขึ้น ในหัวข้อนี้จะนำเสนอกรณีตัวอย่างของตัวแบบ สมมติว่ากำลังจะศึกษาในหัวข้อของหน่วยเก็บข้อมูล (storage) ของตัวแบบ ดังนั้นสถานะที่กำลังสนใจก็คือสถานะของฮาร์ดดิสก์ (hard disk)

1. สมมติว่ามีคอมพิวเตอร์อยู่ 6 เครื่องในระบบ นั่นคือ $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ และกำหนดให้คอมพิวเตอร์ c_6 เป็นเครื่องบริการ (server)
2. สมมติให้ในระบบมีการเชื่อมโยง $L = \{\{c_1, c_6\}, \{c_2, c_6\}, \{c_3, c_6\}, \{c_4, c_6\}, \{c_5, c_6\}\}$ จะเห็นว่าคอมพิวเตอร์แต่ละเครื่องในตัวแบบนี้เชื่อมโยงโดยตรงกับเครื่องบริการ และสื่อสารกับเครื่องอื่นในระบบผ่านเครื่องบริการนี้

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

3. ผู้ใช้ระบบคือ $U = \{u_1 =$ กิตติ, $u_2 =$ ขจร, $u_3 =$ คมกฤช, $u_4 =$ จเรย์, $u_5 =$ ฉัตรชัย$\}$
4. นโยบายของระบบ $P = \{p_1, p_2, p_3, p_4, p_5\}$ กำหนดดังต่อไปนี้

   p_1 : มีเฉพาะ $[u_1]$ ที่สามารถตั้งรหัสผ่านใหม่ให้กับผู้ใช้คนอื่นได้
   p_2 : มีเฉพาะ $[u_2]$ ที่สามารถแก้ไขฐานข้อมูลบนคอมพิวเตอร์ $[c_6]$ ได้
   p_3 : ผู้ใช้ $[u_3, u_4, u_5]$ สามารถใช้งานคอมพิวเตอร์ $[c_1, c_2, c_3]$ เท่านั้น
   p_4 : ผู้ใช้ $[u_3, u_4, u_5]$ สามารถใช้คำสั่ง [INSERT, UPDATE, QUERY] ได้
   p_5 : มีเฉพาะ $[u_1]$ ที่สามารถใช้คำสั่ง [DELETE]

   ตัวอย่างจากเซตของนโยบายจะเห็นได้ว่ามีเฉพาะกิตติเท่านั้นที่สามารถกำหนดรหัสผ่านใหม่ให้ผู้ใช้คนอื่นได้ และมีแค่เขาเพียงคนเดียวที่สามารถลบข้อมูลออกจากระบบ
5. เนื่องจากในการในที่นี้ต้องการศึกษาเกี่ยวกับหน่วยเก็บข้อมูล ดังนั้นเซตของสถานะจึงหมายถึงเซต $S = \{s_1, s_2, s_3, \ldots, s_f\}$ ของสถานะทั้งหมดที่เป็นไปได้ของหน่วยเก็บข้อมูล ยกตัวอย่างเช่น s_i อาจจะเป็นสายอักขระของบิตในหน่วยเก็บข้อมูล ณ เวลาหนึ่งๆ
6. ตัวดำเนินการของระบบ $O = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ กำหนดดังต่อไปนี้

   o_1 : INSERT
   o_2 : UPDATE
   o_3 : DELETE
   o_4 : ALTER TABLE
   o_5 : QUERY
   o_6 : RESET PASSWORD

   ตัวอย่างเช่นตัวดำเนินการ INSERT จะเพิ่มข้อมูลบางอย่างเข้าไปในฐานข้อมูล เมื่อตัวดำเนินการถูกใช้อย่างไม่ถูกต้องหรือเกิดข้อผิดพลาด เช่น เมื่อต้องการลบของมูลบางอย่างจากฐานข้อมูลที่ว่างอยู่แล้ว ตัวแบบจะหยุดทำงานโดยอัตโนมัติ
7. สำหรับตัวอย่างนี้สมมติว่ามีการกระทำของผู้ใช้อยู่ 6 เหตุการณ์นั่นคือเซต $A = \{a_1=(u_1, o_5, 1), a_2=(u_2, o_4, 1), a_3=(u_3, o_1, 2), a_4= (u_5, o_2, 2), a_5=(u_4, o_1, 2), a_6=(u_1, o_3, 3)\}$ ซึ่งทำให้ทราบการกระทำของผู้ใช้ในระบบ ตัวอย่างเช่น ในลำดับเวลาที่ 2 คมกฤชและจเรย์ทำการเพิ่มข้อมูลบางอย่างเข้าสู่ฐานข้อมูล และฉัตรชัยแก้ไขข้อมูลบางอย่างในฐานข้อมูล

จะสังเกตุว่าตัวแบบการบริหารระบบในวิทยานิพนธ์ฉบับนี้ สามารถใช้สำหรับศึกษาแต่ละส่วนประกอบทั้ง 7 และพฤติกรรมขององค์ประกอบเหล่านี้ ซึ่งในบทต่อไปจะนำตัวแบบนี้มาใช้เพื่อศึกษาในด้านทฤษฎีความซับซ้อน ซึ่งพื้นฐานความรู้ที่สำคัญในเรื่องของทฤษฎีความซับซ้อนจะกล่าวถึงต่อไปในส่วนที่ 2.6

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

2.3 การเจริญเติบโตของฟังก์ชัน (Growth of Function)

สำหรับส่วนนี้จะสรุปการวิเคราะห์เวลาในการทำงานของขั้นตอนวิธี (algorithm) ซึ่งสามารถศึกษาเพิ่มเติมได้จาก [1, 6] การเปรียบเทียบประสิทธิภาพของขั้นตอนวิธีนั้นสามารถวัดได้จากทรัพยากรที่ขั้นตอนวิธีนั้นใช้ซึ่งในที่นี้จะพิจารณาเฉพาะ*เวลา*ในการทำงาน เพราะเวลาเป็นทรัพยากรที่มีค่ามากที่สุด สำหรับวิทยานิพนธ์ฉบับนี้จะพิจารณาเวลาในรูปของจำนวนครั้ง (step) ในการทำงาน เพราะหากสมมติว่าแต่ละครั้งในการทำงานใช้เวลาเท่ากัน ถ้าจำนวนครั้งในการทำงานมากก็จะทำให้เวลาในการทำงานมากตามไปด้วย

เวลาในการทำงานโดยมากจะขึ้นอยู่กับสถาปัตยกรรมของเครื่องและขนาดของข้อมูลเข้า (input) ในวิทยานิพนธ์ฉบับนี้จะวิเคราะห์ขั้นตอนวิธีโดยมีสมมติฐานคือ เครื่องคอมพิวเตอร์สามารถทำงานกับคำสั่ง (instruction) ได้ทีละคำสั่ง และสามารถเข้าถึงหน่วยความจำได้แบบสุ่ม (random access memory)

เป็นที่แน่ชัดว่าหากข้อมูลเข้ามีขนาดที่ใหญ่มาก เวลาในการทำงานก็จะมากขึ้นตามเสมอ นั่นคือเวลาในการทำงานเป็นฟังก์ชันของขนาดข้อมูลเข้า ตัวอย่างเช่น หากข้อมูลเข้ามีขนาดเท่ากับ n เวลาในการทำงานของขั้นตอนวิธีก็คือ $T(n)$ ซึ่งฟังก์ชัน $T(n)$ อาจจะมีรายละเอียดที่มากเกินกว่าที่ต้องการ เช่น $T(n) = an^2 + bn + c$ เป็นต้น เนื่องจากเมื่อวิเคราะห์เวลาในการทำงานนั้นผู้วิเคราะห์ต้องการจะทราบเพียงแค่ว่า *ถ้ามูลมีขนาดใหญ่ขึ้นจะใช้เวลาในการทำงานจะมากขึ้นเท่าไร* นั่นก็คือการวิเคราะห์เวลาในการทำงานนั้นต้องการทราบเฉพาะการเจริญเติบโตของฟังก์ชัน $T(n)$ หรือเทอมที่ใหญ่ที่สุดของ $T(n)$ นั่นเอง จากตัวอย่างนี้ก็คือ an^2 และเรามักจะไม่สนใจค่าคงที่ที่คูณอยู่ด้วย เนื่องจากหาก n มีค่าใหญ่มากค่าคงที่นั้นจะไม่มีผลกระทบต่อการเจริญเติบโต

ตัวอย่าง 2.1

สมมติว่ามีขั้นตอนวิธีสำหรับจัดเรียงข้อมูลอยู่ 2 แบบคือ A และ B โดยที่เวลาในการทำงานในกรณีที่แย่ที่สุดของ A เท่ากับ $T_A(n) = 2n^2+200n$ และเวลาในการทำงานที่แย่ที่สุดของ B เท่ากับ $T_B(n) = 30n^2+350$ เมื่อพิจารณาอัตราการเจริญเติบโตของทั้ง 2 วิธีจึงสรุปได้ว่าเมื่อขนาดของข้อมูลเข้ามีขนาดใหญ่มาก ทั้ง 2 วิธีจะใช้เวลาในการทำงานที่ใกล้เคียงกันดังนั้นจึงเลือกใช้ขั้นตอนวิธีใดก็ได้

2.4 สัญลักษณ์แบบอะซิมโทติก (Asymptotic Notation)

จากส่วนที่ 2.3 ได้กล่าวถึงการพิจารณาเวลาในการทำงานของขั้นตอนวิธีโดยดูจากอัตราการเจริญเติบโตของฟังก์ชันเวลาของขั้นตอนวิธีนั้น โดยไม่สนใจเทอมที่มีขนาดเล็ก และค่าคงที่ใน

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

ฟังก์ชัน การพิจารณาในลักษณะนี้เรียกว่าพิจารณาเชิงอะซิมโทติก สัญลักษณ์อะซิมโทติกนั้นมีหลายแบบ แต่ในวิทยานิพนธ์ฉบับนี้จะนำเสนอเฉพาะสัญลักษณ์ที่สำคัญและใช้เป็นประจำในการพิจารณาเวลาในการทำงานของขั้นตอนวิธี ซึ่งสัญลักษณ์อะซิมโทติกมีนิยามได้ดังต่อไปนี้

นิยาม 2.3 สัญลักษณ์บิ๊กเธทต้า (Θ-notation**)**

$\Theta(g(n)) = \{f(n) \mid$ มีค่าคงที่บวก c_1, c_2 และ n_0 โดยที่ $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ สำหรับ $n \ge n_0\}$

ฟังก์ชันใดๆ $f(n)$ จะเป็นสมาชิกของเซต $\Theta(g(n))$ เมื่อมีค่าคงที่บวก c_1 และ c_2 ที่ทำให้ $c_1 g(n) \le f(n) \le c_2 g(n)$ จะเห็นว่า $\Theta(g(n))$ เป็นเซตของฟังก์ชันแต่โดยมากมักจะนิยมเขียน "$f(n) = \Theta(g(n))$" แทน "$f(n) \in \Theta(g(n))$" เพื่อความสะดวก แต่ก็ยังมีความหมายว่า $f(n)$ เป็นสามาชิกของเซต $\Theta(g(n))$ เหมือนเดิม

นิยาม 2.4 สัญลักษณ์บิ๊กโอ (O-notation)

$O(g(n)) = \{f(n) \mid$ มีค่าคงที่บวก c และ n_0 โดยที่ $0 \le f(n) \le cg(n)$ สำหรับ $n \ge n_0\}$

สัญลักษณ์บิ๊กโอมักจะใช้เพื่ออธิบายขอบเขตบน (upper bound) ของฟังก์ชันภายใต้ตัวประกอบ (factor) ที่เป็นค่าคงที่ โดยทั่วไปจะเขียน $f(n) = O(g(n))$ แต่มีความหมายว่า $f(n) \in O(g(n))$ ในการเวลาในการทำงานของขั้นตอนวิธี จะใช้สัญลักษณ์นี้เพื่อวิเคราะห์เวลาในการทำงานสำหรับกรณีที่แย่ที่สุด (worst case) ของขั้นตอนวิธี

นิยาม 2.5 สัญลักษณ์บิ๊กโอเมก้า (Ω-notation)

$\Omega(g(n)) = \{f(n) \mid$ มีค่าคงที่บวก c และ n_0 โดยที่ $0 \le cg(n) \le f(n)$ สำหรับ $n \ge n_0\}$

สัญลักษณ์บิ๊กโอเมก้าใช้เพื่ออธิบายขอบเขตล่าง (lower bound) ของฟังก์ชันภายใต้ตัวประกอบที่เป็นค่าคงที่ และมักจะเขียน $f(n) = \Omega(g(n))$ โดยให้ความหมายว่า ว่า $f(n) \in \Omega(g(n))$ ในการวิเคราะห์ขั้นตอนวิธี สัญลักษณ์นี้จะใช้บอกว่าขั้นตอนวิธีนั้นจะต้องใช้เวลาในกาทำงานอย่างน้อยเท่าไร นั่นก็คือเวลาในการทำงานสำหรับกรณีที่ดีที่สุด (best case) ของขั้นตอนวิธี

จากนิยาม 2.3 2.4 และ 2.5 จะเห็นได้อย่างชัดเจนว่าหาก $f(n) = \Omega(g(n))$ และ $f(n) = O(g(n))$ ก็จะได้ว่า $f(n) = \Theta(g(n))$ และในทางกลับกันหากแสดงได้ว่า $f(n) = \Theta(g(n))$ ก็จะสรุปได้ว่า $f(n) = \Omega(g(n))$ และ $f(n) = O(g(n))$

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

2.5 การนับ (Counting)

การวิเคราะห์ขั้นตอนวิธีนั้น จำเป็นต้องมีความเข้าใจในเรื่องของทฤษฎีความน่าจะเป็นและการนับ ในวิทยานิพนธ์ฉบับนี้จะใช้ความรู้บางส่วนจากพื้นฐานการนับสำหรับกำหนดนิยามของปัญหาในบทที่ 4 และใช้สำหรับออกแบบขั้นตอนวิธีในบทที่ 5 ต่อไป ดังนั้นในส่วนนี้จึงจะสรุปใจความสำคัญของพื้นฐานการนับเพื่อให้เข้าใจเนื้อหาส่วนต่อไปได้ดียิ่งขึ้น ซึ่งสามารถศึกษาเพิ่มเติมได้จาก [1, 6, 9] การศึกษาเกี่ยวกับพื้นฐานการนับ จำเป็นต้องมีความรู้พื้นฐานเกี่ยวกับเรื่องเซต (set) ซึ่งจะไม่กล่าวถึงในที่นี้

สมมติว่ามีเซตที่ไม่มีส่วนร่วมกัน A และ B โดยที่เซตทั้งสองมีจำนวนสมาชิกเท่ากับ $|A|$ และ $|B|$ ตามลำดับ หากต้องการเลือกสามาชิกหนึ่งตัวจากเซตใดก็ได้จะมีจำนวนกรณีที่เลือกได้ทั้งหมดเท่ากับ $|A \cup B|$ แต่เนื่องจาก A และ B ไม่มีส่วนร่วมกันดังนั้นจึงได้จำนวนกรณีที่จะเกิดขึ้นได้ทั้งหมดเท่ากับ $|A| + |B|$ สำหรับการเลือกในกรณีนี้เรียกว่า*กฎของผลบวก*

จากเซต A และ B หากต้องการเลือกสมาชิก 1 ตัวจากเซต A และอีก 1 ตัวจากเซต B โดยที่จะต้องเลือกจากเซต A ก่อน จะได้ว่าจะมีเหตุการณ์ที่เกิดขึ้นทั้งหมดเท่ากับ $|A|\cdot|B|$ กรณี ซึ่งการเลือกในกรณีนี้เรียกว่า*กฎของผลคูณ*

สมมติว่าเซต A มีจำนวนสมาชิกทั้งหมด $n = |A|$ ตัว มีคำถามว่าหากต้องการจะเลือกสมาชิกของ A ออกมา k ตัวโดยให้ความสำคัญกับลำดับ จะมีเหตุการณ์ที่ต่างกันเกิดขึ้นได้ทั้งหมดกี่แบบ เริ่มจากพิจารณาเซต A จะเห็นว่าในการเลือกสมาชิกออกมา 1 ตัวในแต่ละครั้งจะทำให้เกิดเซตใหม่ที่มีสมาชิกน้อยกว่าเดิม 1 ตัว ดังนั้นจากกฎของผลคูณจะได้ว่าสามารถเกิดเหตุการณ์ที่ต่างกันทั้งหมดเท่ากับ $(n)(n-1)(n-2)\ldots(n-k+1)$ เหตุการณ์ และในกรณีพิเศษที่ค่า $k = n$ จะได้ว่ามีจำนวนเหตุการณ์ทั้งหมดที่เกิดขึ้นได้เท่ากับ $(n)(n-1)(n-2)\ldots(1)$ หรือเท่ากับ $n!$ เครื่องหมาย ! อ่านว่า “แฟกทอเรียล” (factorial) และเรียกกรณีที่ $k = n$ นี้เรียกว่า *การเรียงสับเปลี่ยน* (permutation)

จากเซต A ที่มีจำนวนสมาชิก $n = |A|$ ตัว มีคำถามว่าหากต้องการจะเลือกสมาชิกของ A ออกมา k ตัวโดยไม่สนใจลำดับ จะมีเหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมดเท่าไร เริ่มพิจารณาจากกรณีที่สนใจลำดับ จะเห็นว่ามีเหตุการณ์ที่ต่างกันทั้งหมด $(n)(n-1)(n-2)\ldots(n-k+1)$ เหตุการณ์ และแต่ละเซตที่เลือกออกมา k ตัว นั้นสมารถเรียงสับเปลี่ยนกันได้ $k!$ วิธี ดังนั้นจะได้ว่าเหตุการณ์ที่ต่างกันในการเลือกสมาชิก k ตัวจากเซต A โดยไม่สนใจลำดับ หรือเรียกอีกอย่างว่า *การจัดหมู่* (combination) จึงเท่ากับ

$$\frac{(n)(n-1)\ldots(n-k+1)}{k!} = \frac{(n)(n-1)\ldots(n-k+1)(n-k)!}{k!(n-k)!} = \frac{n!}{k!(n-k)!} \qquad (2.1)$$

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

ในกรณีนี้หาก $k = 0$ จะได้ว่าจำนวนเหตุการณ์ที่ได้จะเท่ากับ 1 นั่นคือเหตุการณ์ที่เกิดขึ้นจากการเลือกของ 0 สิ่งก็คือไม่ต้องเลือกอะไรเลย ($0! = 1$)

2.6 ทฤษฎีความซับซ้อน (Complexity Theory)

ในการศึกษาความยากง่ายของปัญหานั้นทำได้โดยพิจารณาความซับซ้อนของปัญหา ซึ่งในวิทยานิพนธ์ฉบับนี้ได้สรุปเพียงส่วนหนึ่งที่สำคัญจาก [6, 8] ปัญหาบางอย่างสามารถหาผลเฉลยได้ภายในเวลาที่เป็นฟังก์ชันพหุนามของขนาดข้อมูลเข้า และขั้นตอนวิธีสำดังกล่าวจะเรียกว่า ขั้นตอนวิธีแบบพหุนาม (polynomial time algorithm) นั่นคือหากข้อมูลเข้า (input) มีขนาดเท่ากับ n ขั้นตอนวิธีที่ใช้จะหยุดทำงาน (halt) และให้คำตอบภายในเวลา $O(n^k)$ โดยที่ k เป็นค่าคงที่ใดๆ คำถามต่อมาก็คือ ทุกปัญหามีขั้นตอนวิธีแบบพหุนามสำหรับหาผลเฉลยหรือไม่ ซึ่งคำตอบที่ได้ก็คือ *ไม่มี* และปัญหาในกลุ่มที่ไม่สามารถหาผลเฉลยในเวลาแบบพหุนามเรียกว่า ปัญหาเอ็นพีคอมพลีท (*NP* – complete problems) ก่อนที่จะอธิบายปัญหาในกลุ่มเอ็นพีคอมพลีท จะเริ่มจากทำความเข้าใจปัญหาในคลาสเอ็นพี (class *NP*) เป็นอันดับแรก

ข้อมูลใดๆ ที่อยู่ในคอมพิวเตอร์ จะถูกเข้ารหัสให้อยู่ในรูปของสายอักระ (string) บนเซต {0, 1} ซึ่งสายอักขระเหล่านี้ข้อมูลเหล่านี้เรียกได้อีกอย่างว่าภาษา (language) เช่นปัญหาที่กำลังสนใจหากต้องการจะใช้คอมพิวเตอร์ในการแก้ปัญหาก็ต้องนำไปเข้ารหัสเพื่อให้คอมพิวเตอร์สามารถคำนวณเพื่อหารผลเฉลยของปัญหานั้นได้ก่อน และสายอักขระจากการเข้ารหัสนี้เรียกว่าภาษาของปัญหา

นิยาม 2.6 คลาสเอ็นพี (class *NP*)

ภาษา L เป็นสามาชิกของคลาสเอ็นพี ก็ต่อเมื่อ มีขั้นตอนวิธีแบบพหุนาม A และค่าคงที่ c โดยที่ $L = \{x \in \{0,1\}^* \mid$ มีคำตอบที่เป็นไปได้ y โดยที่ $|y| = O(|x|^c)$ โดยที่ $A(x, y) = 1\}$

นั่นคือหากมีคำตอบที่เป็นไปได้ (certificate) ขั้นตอนวิธี A จะตรวจสอบคำตอบนั้นภายในเวลาแบบพหุนาม หรือกล่าวได้ว่าปัญหาในคลาสเอ็นพีเป็นปัญหาที่สามารถตรวจสอบคำตอบที่ให้มาได้ภายในเวลาแบบพหุนาม ตัวอย่างเช่นปัญหาการจัดเรียงตัวเลข n ตัวจากน้อยไปหามาก เมื่อกำหนดลำดับตัวเลขใดๆ จะสามารถตรวจสอบลำดับนั้นโดยการไล่ตัวเลขในลำดับจากตัวแรกไปถึงตัวสุดท้าย นั่นคือปัญหาการจัดเรียงตัวเลข n ตัว จัดอยู่ในคลาสเอ็นพี

ภาษาใดๆ $L \in \{0,1\}^*$ สามารถลดรูป (reduce) ไปเป็นปัญหา $L' \in \{0,1\}^*$ ได้ภายในเวลาแบบพหุนามจะใช้สัญลักษณ์ $L \leq_p L'$ เมื่อมีฟังก์ชันที่คำนวณได้ภายในเวลาแบบพหุนาม f ที่ซึ่งทุกกรณีตัวอย่าง (instance) $x \in L$ ฟังก์ชัน $f : x \in \{0,1\}^* \mapsto \{0,1\}^*$ โดยที่ $x \in L$ ก็ต่อเมื่อ $f(x) \in L'$

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved

ปัญหาในคลาสเอ็นพีคอมพลีทจัดว่าเป็นคลาสที่ยังไม่มีขั้นตอนวิธีแบบพหุนามสำหรับหาผลเฉลย ทำให้นักวิทยาการคอมพิวเตอร์ส่วนมากมีความเชื่อว่าเป็นปัญหาที่ยากสำหรับคอมพิวเตอร์ แต่อย่างไรก็ดีความสัมพันธ์ของคลาสเอ็นพีคอมพลีทกับคลาสพี (กลุ่มปัญหาที่สามารถหาผลเฉลยได้ด้วยขั้นตอนวิธีแบบพหุนาม) ก็ยังพิสูจน์ไม่ได้ โดยที่ปัญหาเอ็นพีคอมพลีทมีนิยามดังนี้

นิยาม 2.7 คลาสเอ็นพีคอมพลีท (class *NP*-complete)

ภาษา L อยู่ในคลาสเอ็นพีคอมพลีท ก็ต่อเมื่อ

1. $L \in NP$ และ
2. $L' \leq_p L$ สำหรับทุก $L' \in NP$

จากนิยาม 2.7 ในข้อ 2 หมายความว่าปัญหาที่กำลังสนใจ L มีความยาก “อย่างน้อย” เท่ากับปัญหา L' สำหรับทุก $L' \in NP$ จากข้อ 1 และ 2 จะได้ว่าปัญหา L เป็นปัญหาที่ยากที่สุดในคลาส NP ซึ่งปัญหาในลักษณะนี้มีความเป็นไปได้อย่างมากที่จะไม่มีขั้นตอนวิธีแบบพหุนามสำหรับหาผลเฉลยที่ถูกต้องแม่นยำ ดังนั้นจึงต้องใช้ขั้นตอนวิธีเชิงประมาณเพื่อหาคำตอบโดยประมาณแทน

2.7 ขั้นตอนวิธีเชิงประมาณเบื้องต้น (Approximation Algorithm)

ปัญหาที่พบในชีวิตจริงจำนวนมากเป็นปัญหาในคลาสเอ็นพีคอมพลีท (NP - complete) แต่ปัญหาเหล่านี้กลับเป็นปัญหาที่มีความสำคัญเกินกว่าที่มองข้าม การจัดการกับปัญหาเหล่านี้ทำได้อย่างน้อย 3 วิธีคือ หากขนาดของข้อมูลเข้า (input) มีขนาดเล็ก ขั้นตอนวิธีที่ใช้เวลาในการทำงานแบบเลขชี้กำลัง (exponential running time) ที่ให้คำตอบที่ถูกต้องแม่นยำก็ถือว่าเหมาะสมที่สุด วิธีต่อมาคือ แบ่งศึกษาเฉพาะกรณีตัวอย่าง (instance) ที่สำคัญ และมีขั้นตอนวิธีแบบพหุนามสำหรับหาผลเฉลย วิธีสุดท้ายก็คือ ใช้ขั้นตอนวิธีที่ให้คำตอบที่ใกล้เคียงกับคำตอบที่ดีที่สุด (near optimal solution) ซึ่งขั้นตอนวิธีแบบนี้เรียกว่า ขั้นตอนวิธีเชิงประมาณ (Approximation Algorithm)

ในการวิเคราะห์ขั้นตอนวิธีเชิงประมาณ นอกจากขอบเขตบนของเวลาที่ใช้ในการทำงานจะต้องไม่เกินฟังก์ชันพหุนามของของขนาดข้อมูลเข้าแล้ว ยังต้องพิจารณาเซตคำตอบที่ได้ว่าใกล้เคียงกับคำตอบที่ดีที่สุดเท่าไร โดยพิจารณาจากอัตราส่วนการประมาณ (approximation ratio) ขั้นตอนวิธีสำหรับหาผลเฉลยแบบประมาณของทั้งปัญหาที่ต้องการคำตอบที่มากที่สุด (maximization) และปัญหาที่ต้องการคำตอบที่น้อยที่สุด (minimization) จะเรียกว่ามี อัตราส่วนการประมาณ $\rho(n)$ ก็ต่อเมื่อ ทุกขนาดข้อมูลเข้า n ขั้นตอนวิธีจะให้คำตอบ C ที่มีค่าอย่างมากเป็นอันตราส่วน $\rho(n)$ ของคำตอบที่ดีที่สุด C^* หรือ $\max(\frac{C}{C^*}, \frac{C^*}{C}) \leq \rho(n)$

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved