การเติบโตของฟังก์ชัน 3

จากตัวอย่างที่ได้นำเสนอในบทที่ 1 เราได้แสดงให้เห็นถึงเวลาการทำงานของโปรแกรมที่เป็นฟังก์ชันของจำนวนข้อมูล เช่น การหาค่าน้อยสุด การหาตัวหมู่มากเป็นต้น แน่นอนว่าถ้าจำนวนข้อมูลมากขึ้น ย่อมต้องกินเวลาการทำงานมากขึ้นตาม แต่จะมากขึ้นแค่ไหน เท่าไรนั้น ก็คงขึ้นกับลักษณะของฟังก์ชันซึ่งแทนเวลาการทำงาน สิ่งที่เราจะสนใจกันในบทนี้ก็คือการเติบโตของฟังก์ชันเหล่านี้ ที่จะใช้เป็นตัวเปรียบเทียบประสิทธิภาพของอัลกอริทึม โดยจะพิจารณาการเติบโตของฟังก์ชันเฉพาะกรณีที่ข้อมูลมีจำนวนมากๆ เพื่อศึกษาภาพรวมของการเติบโต โดยจะใช้สัญกรณ์จำนวนหนึ่งที่ช่วยบรรยายลักษณะการเติบโต เพื่อใช้ในการเปรียบเทียบได้ง่าย

อัตราการเติบโต

หากย้อนกลับไปดูตัวอย่างการหาค่าน้อยสุดในแถวลำดับในบทที่ 1 ที่เราแสดงโปรแกรมตัวอย่างให้สามโปรแกรมนั้น พบว่ามีประสิทธิภาพเชิงเวลาดังนี้

โปรแกรมที่ 1 $T_1(n) = t_5 n + t_6(n-1) + t_3 + t_7$

โปรแกรมที่ 2 $T_2(n) = t_{11} n + (t_{12} + t_{13})(n-1)$

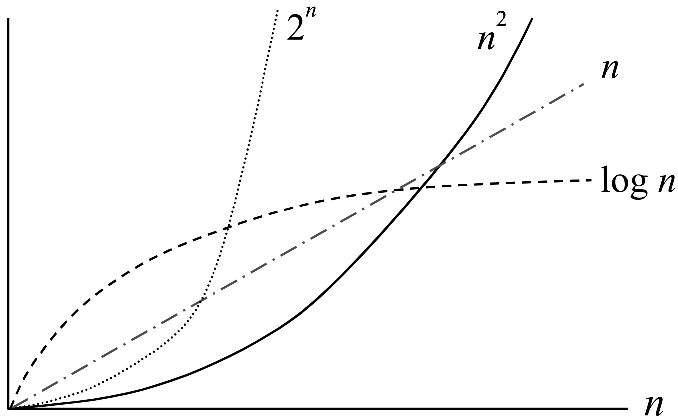
โปรแกรมที่ 3 $T_3(n) = t_{22} n + (t_{22}+t_{23}+t_{24}+t_{25}+t_{26})(n-1)$

ทั้งสามโปรแกรมนี้มีฟังก์ชันที่มีการเติบโตเป็นเชิงเส้นด้วยความชันที่แตกต่างกัน ฟังก์ชันจริงๆ จะเป็นเช่นไร ก็คงขึ้นกับว่าเวลาเขียนเป็นโปรแกรมจริง แปลเป็นคำสั่งที่ทำงานจริง และใช้งานบนเครื่องจริงแล้ว จะเป็นอย่างไร ซึ่งขึ้นกับปัจจัยหลายอาทิ ประสบการณ์ของผู้เขียนโปรแกรม ภาษาและตัวแปลที่เลือกใช้ อีกทั้งเครื่องที่ใช้ทำงานจริงด้วย ดังนั้นการจะเปรียบเทียบการ

ทำงานจริงนั้นคงจะลำบาก แต่สิ่งที่เราสรุปได้อย่างหนึ่ง ซึ่งไม่ขึ้นกับปัจจัยที่กล่าวถึงนี้ ก็คือทั้งสามโปรแกรมนี้ใช้เวลาการทำงานเป็นเชิงเส้น หมายความว่าเวลาการทำงานจะเพิ่มขึ้น k เท่าเมื่อเราเพิ่มปริมาณข้อมูลอีก k เท่า

หรือถ้าจะดูตัวอย่างการหาค่าน้อยสุดอันดับที่สองที่แสดงในบทที่ 1 นั้นเราพบว่าแบบที่หนึ่งนั้นจะใช้ฟังก์ชันเชิงเส้นในการหาตัวน้อยสุดอันดับที่สอง ในขณะที่แบบที่เป็นส่วนขยายของโปรแกรมที่สามนั้นจะใช้เวลาเป็นฟังก์ชัน log ฐานสองของจำนวนข้อมูล โดยที่เราไม่ได้แสดงให้เห็นรายละเอียดของโปรแกรม แต่เราสรุปได้ว่าถ้าเราเพิ่มข้อมูลเป็น 2 เท่า โปรแกรมที่ทำงานแบบเชิงเส้นจะใช้เวลาเพิ่มเป็น 2 เท่าเช่นกัน ในขณะที่โปรแกรมที่มีพฤติกรรมการเติบโตของเวลาการทำงานเป็นฟังก์ชัน log ฐานสองนั้นจะใช้เวลาเพิ่มอีก 1 หน่วยเวลาเท่านั้น อันนี้เห็นได้ว่าอัตราการเติบโตของฟังก์ชันทั้งสองนี้ต่างกันมากๆ

ดังนั้นสิ่งที่เราสนใจจะเปรียบเทียบก็คืออัตราการเติบโตของฟังก์ชันที่แทนประสิทธิภาพของอัลกอริทึม รูปที่ 3–1 แสดงตัวอย่างฟังก์ชันที่มีอัตราการเติบโตซึ่งแตกต่างกัน



รูปที่ 3–1 ตัวอย่างฟังก์ชันที่มีอัตราการเติบโตแตกต่างกัน

บางคนอาจแย้งว่าก็เล่นเขียนฟังก์ชันแบบคร่าวๆ บางทีโปรแกรมที่ใช้เวลาเป็นแบบ $\log n$ อาจมีค่าคงตัวข้างหน้า $\log n$ ที่มีค่ามากก็ได้ เพราะมีจำนวนคำสั่งต่อรอบการทำงานมากมายเหลือเกิน อันนี้เป็นข้อสังเกตที่ดี ก็ขอยกตัวอย่างให้ดู ก็แล้วกัน สมมติว่าโปรแกรม P_1 ใช้เวลา $1000\cdot\log n$ ในขณะที่โปรแกรม P_2 ใช้เวลา n ถามว่า P_1 ใช้เวลามากกว่า P_2 เมื่อ n มีค่าเท่าใด ทดลองเปลี่ยนค่า n สักครู่ก็รู้ว่า $1000\cdot\log n > n$ เฉพาะเมื่อ $n < 3551$ เท่านั้น อันนี้เป็นสิ่งที่แสดงให้เห็นว่าถ้าฟังก์ชัน f โตเร็วกว่าฟังก์ชัน g แล้วค่าของ f ต้องมากกว่าค่าของ g เมื่อ $n \geq n_0$ ดังนั้น

อัตราการเติบโตของฟังก์ชันจะใช้เปรียบเทียบอัลกอริทึมได้อย่างสื่อความหมาย ก็เมื่อเรากำลังพูดถึงกรณีที่ข้อมูลมีจำนวนมาก

แล้วจะพิจารณากรณีมากสักแค่ไหน ก็เพื่อให้มั่นใจแน่ๆ ก็คิดตอนที่มันมีขนาดมากเข้าใกล้อนันต์เลย เราเขียน $f(n) \prec g(n)$ เพื่อแทนว่า $f(n)$ โตช้ากว่า $g(n)$ โดยมีนิยามดังนี้

$$f(n) \prec g(n) \quad \textit{iff} \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

ในทางกลับกัน ถ้าค่าลิมิตข้างบนนี้มีค่าเป็นอนันต์ เราก็บอกว่า $f(n)$ โตเร็วกว่า $g(n)$ แต่ถ้าค่าลิมิตนี้เป็นค่าอื่นที่ไม่ใช่ 0 และ อนันต์ เราก็จะเรียกว่าทั้ง $f(n)$ และ $g(n)$ โตพอกัน

ต้องขอเน้นตรงนี้ครับว่า ฟังก์ชันที่เราพูดกันในเรื่องของอัลกอริทึมนั้นเป็นฟังก์ชันที่ให้ค่าเป็นจำนวนไม่ติดลบ เราจะไม่พิจารณากรณีประสิทธิภาพติดลบ (ไม่รู้หมายความว่าอะไรเหมือนกัน) จะได้ไม่ต้องมาห่วงกรณี $-\infty$

ในการหาลิมิตข้างบนนี้ ก็ขอให้นึกถึงกฎของโลปิตาล (l'Hôpital's Rule) ที่เคยเรียนกันในวิชาแคลคูลัส จะช่วยได้มากทีเดียว ขอเขียนทบทวนให้ดูโดยไม่พิสูจน์ที่มาดังนี้

กฎของโลปิตาล ถ้า $f(n)$ และ $g(n)$ เป็นฟังก์ชันที่หาอนุพันธ์ได้ โดยที่ $\lim_{n\to\infty} f(n) = \infty$ และ $\lim_{n\to\infty} g(n) = \infty$ แล้ว $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$

ตัวอย่างที่ 3-1 จงเรียงลำดับฟังก์ชันต่อไปนี้ตามอัตราการเติบโต : $0.5^n,\ 1,\ \log n,\ n,\ 10^n$

พอสรุปได้ดังนี้

- $0.5^n \prec 1 \prec \log n$ เพราะว่า 0.5^n เป็นฟังก์ชันซึ่งนอกจากจะไม่โตแล้ว ยังมีค่าลดลงเรื่อยๆ แต่ 1 นั้นเป็นฟังก์ชันนิ่งๆ ไม่เพิ่มไม่ลด ในขณะที่ $\log n$ เป็นฟังก์ชันที่โต
- มาดู $\log n$ กับ n จากกฎของโลปิตาลจะได้ว่า $\lim_{n\to\infty} \frac{\log n}{n} = \lim_{n\to\infty} \frac{(1/\ln 10)(1/n)}{1} = 0$ ดังนั้น $\log n \prec n$
- จากข้อบนย่อมได้ว่า $10^{\log n} \prec 10^n$ ดังนั้น $n \prec 10^n$

ดังนั้น $0.5^n \prec 1 \prec \log n \prec n \prec 10^n$

ตัวอย่างที่ 3-2 จงเปรียบเทียบอัตราการเติบโตของ $\ln^9 n$ กับ $n^{0.1}$

ใช้กฎของโลปิตาลได้ดังนี้

$$\begin{aligned}\lim_{n\to\infty}\frac{\ln^9 n}{n^{0.1}} &= \lim_{n\to\infty}\frac{(9\ln^8 n)(1/n)}{0.1n^{(0.1-1)}} = \lim_{n\to\infty}\frac{9\ln^8 n}{0.1n^{0.1}}\\ &= \lim_{n\to\infty}\frac{9\cdot 8\ln^7 n}{(0.1)^2 n^{0.1}}\\ &\dots\\ &= \lim_{n\to\infty}\frac{9\cdot 8\cdots 1\ln^0 n}{(0.1)^9 n^{0.1}}\\ &= 0\end{aligned}$$

สรุปได้ว่า $\ln^9 n \prec n^{0.1}$

อยากให้ผู้อ่านลองไปทำต่อเพื่อแสดงให้เห็นจริงว่า $\log^a n \prec n^b$ สำหรับจำนวนจริง $b > 0$ และขอให้สังเกตด้วยว่าฐานของ log จะเป็นฐานอะไรก็ได้ที่มีค่ามากกว่า 1 ก็ทำให้ $\log^a n \prec n^b$ เนื่องจากเราสามารถเปลี่ยนจากฐานหนึ่งไปอีกฐานหนึ่งได้โดยการคูณค่าคงตัวค่าหนึ่ง ซึ่งก็ไม่ทำให้ค่าของลิมิตเปลี่ยนไปแต่อย่างใด

ตัวอย่างที่ 3-3 จงเปรียบเทียบอัตราการเติบโตของ n^{10} กับ 2^n

จากความรู้ที่ว่า $\log^a n \prec n^b$ แสดงว่า $\lg^{10} n \prec n$ แทน $\lg n$ ด้วย n จะได้ว่า $n^{10} \prec 2^n$

(อ้อลืมบอกไปว่า คนในวงการเขาชอบใช้ lg แทน log ฐาน 2 เนื่องจากมันเป็นฐานที่พบบ่อยในคอมพิวเตอร์ ก็เลยจะใช้บ้างในที่นี้)

และในทำนองเดียวกันกับตัวอย่างที่แล้ว อยากให้ผู้อ่านลองไปทำต่อเพื่อแสดงให้เห็นจริงว่า $n^a \prec b^n$ สำหรับจำนวนจริง $b > 1$ จากสองตัวอย่างข้างต้นนี้ ขอสรุปอัตราการเติบโตของฟังก์ชันที่ใช้มากดังนี้

- $\log^a n \prec n^b,\ b > 0$ หมายความว่าฟังก์ชัน polylogarithmic โตช้ากว่าฟังก์ชัน polynomial
- $n^a \prec b^n,\ b > 1$ หมายความว่าฟังก์ชัน polynomial โตช้ากว่าฟังก์ชัน exponential

สัญกรณ์เชิงเส้นกำกับ

มีวิธีแทนความสัมพันธ์ของฟังก์ชันในแง่ของอัตราการเติบโตอีกแบบหนึ่ง คือการใช้สัญกรณ์เชิงเส้นกำกับ (asymptotic notations) การใช้สัญกรณ์ในลักษณะนี้จะช่วยทำให้การเขียนบรรยายฟังก์ชันกระทำได้ง่าย เนื่องจากเป็นการแทนพฤติกรรมของฟังก์ชัน $f(n)$ เมื่อ n มีค่ามากๆ อีกทั้งทำให้การจัดการฟังก์ชัน (เช่น การหาผลรวม การหาค่ามาก ค่าน้อย และอื่นๆ) กระทำได้ง่ายขึ้น เราจะอธิบายในรายละเอียดของสัญกรณ์ 5 ตัวดังนี้ o, ω, Θ, O และ Ω (ขอเน้นตรงนี้หน่อยครับว่าเรื่องนิยามของสัญกรณ์ต่างๆ ที่จะพูดต่อไปนี้นั้น หนังสือหลายๆ เล่มจะให้นิยามแตกต่างกันไป ขึ้นกับว่าผู้เขียนจะเครียดหรือจู้จี้ขนาดไหน สำหรับผมเอาง่ายๆ ก็คือให้จำไว้เสมอว่าสัญกรณ์ที่จะใช้ต่อไปนี้ใช้กับฟังก์ชันที่ให้ค่าไม่ติดลบ เพราะเรากำลังพูดถึงฟังก์ชันที่แทนประสิทธิภาพของอัลกอริทึม)

โอเล็ก

เราเริ่มด้วยสัญกรณ์โอเล็ก เขียนแทนด้วย o ซึ่งมีนิยามดังนี้

$$\mathrm{o}(g(n)) = \left\{ f(n) \mid \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \right\}$$

พูดง่ายๆ ก็คือว่า $\mathrm{o}(g(n))$ ก็คือเซตของฟังก์ชันทั้งหลายที่โตช้ากว่า $g(n)$ ดังนั้นจากตัวอย่างก่อนหน้านี้เราจะได้ว่า $\log^{1000} n \in \mathrm{o}(n^{0.00001})$, $n^{1000} \in \mathrm{o}((1.0001)^n)$ เป็นต้น

โอเมกาเล็ก

ในทางกลับกัน เรานิยามให้ $\omega(g(n))$ ก็คือเซตของฟังก์ชันทั้งหลายที่โตเร็วกว่า $g(n)$ ดังนี้

$$\omega(g(n)) = \left\{ f(n) \mid \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \right\}$$

ซึ่งก็เห็นได้ชัดเจนว่า $f(n) \in \omega(g(n))$ ก็ต่อเมื่อ $g(n) \in \mathrm{o}(f(n))$

ทีตาใหญ่

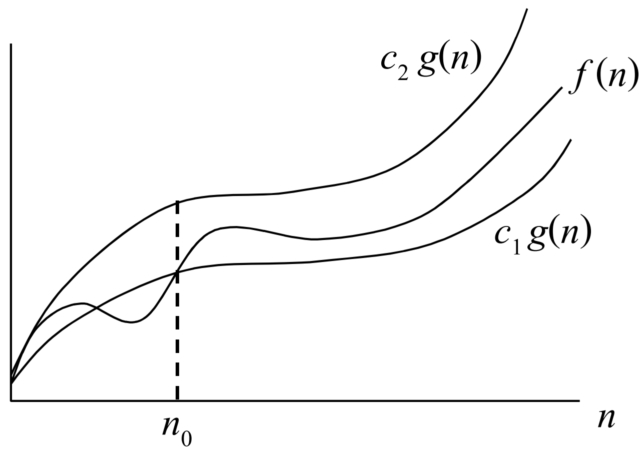
สำหรับกรณีที่ฟังก์ชันโตด้วยอัตราเดียวกัน เราก็มีสัญกรณ์ให้คือ Θ ซึ่งมีนิยามดังนี้

$$\Theta(g(n)) = \left\{ f(n) \mid \lim_{n\to\infty} \frac{f(n)}{g(n)} = c, c \neq 0, c \neq \infty \right\}$$

หรือจะเขียนนิยามแบบไม่ต้องยุ่งกับลิมิตก็จะได้แบบนี้

$\Theta(g(n))$ = { $f(n)$ | มีค่าคงตัวบวกสามตัวคือ c_1, c_2 และ n_0 ที่ทำให้ $c_1g(n) \le f(n) \le c_2g(n)$ เมื่อ $n \ge n_0$ }

เขียนซะยืดยาวอย่างนี้ ก็เพราะนิยามแบบนี้มองเห็นภาพได้ง่ายกว่า พิจารณารูปที่ 3–2 เราบอกว่า $f(n) \in \Theta(g(n))$ ก็แสดงว่า $g(n)$ เป็นฟังก์ชันที่กำหนดขอบเขตการเติบโตของ $f(n)$ ทั้งขอบเขตบนและขอบเขตล่าง เมื่อ n มีค่ามากพอ (คือเมื่อมีค่าตั้งแต่ n_0 เป็นต้นไป) ค่า c_1 และ c_2 เป็นแค่ตัวคูณ $g(n)$ เพื่อให้ขอบเขตล่างและบนมีรูปแบบการเติบโตคล้าย $g(n)$ เพียงแต่เอียงลงและขึ้นเล็กน้อย รูปที่ 3–2 แสดงให้เห็นว่า $f(n)$ จะไม่หลุดออกนอกขอบเขตล่างและบนนี้เลย เมื่อ $n \ge n_0$ เราเรียก $g(n)$ ว่าเป็นฟังก์ชันกำหนดขอบเขตที่กระชับของ $f(n)$



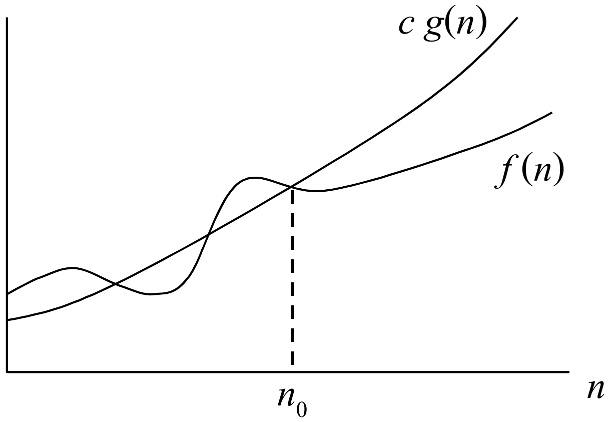
รูปที่ 3–2 $f(n) \in \Theta(g(n))$

โอใหญ่

โอใหญ่ไม่ได้มีไว้เพื่อให้ตรงข้ามกับโอเล็ก เราเขียน $f(n) \in O(g(n))$ เพื่อบอกว่า $f(n)$ เป็นฟังก์ชันที่โต<u>ไม่เร็วกว่า</u> $g(n)$ นั่นคือ $O(g(n)) = o(g(n)) \cup \Theta(g(n))$ คือเป็นเซตที่รวมฟังก์ชันที่โตช้ากว่าและที่โตเท่ากับ $g(n)$ หรือเขียนเป็นนิยามได้อีกแบบหนึ่งดังนี้

$O(g(n))$ = { $f(n)$ | มีค่าคงตัวบวกสองตัวคือ c และ n_0 ที่ทำให้ $f(n) \le cg(n)$ เมื่อ $n \ge n_0$ }

หมายความว่าถ้า $f(n) \in O(g(n))$ แสดงว่าการเติบโตของ $f(n)$ จะถูกกำหนดขอบเขตด้านบนไว้ด้วยลักษณะการเติบโตของ $g(n)$ นั่นคือเราสามารถหาค่าคงตัวบวก c ที่ $f(n) \le cg(n)$ แสดงเป็นตัวอย่างได้ดังรูปที่ 3–3



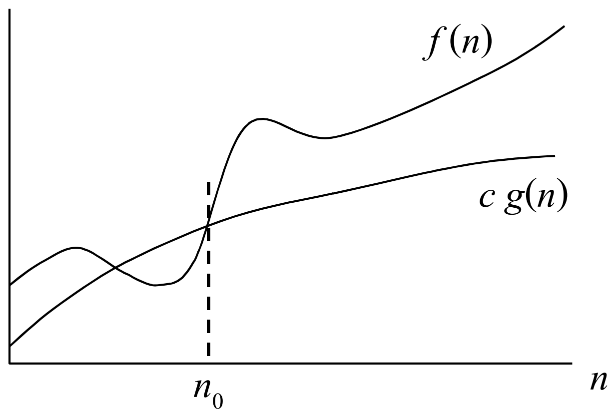
รูปที่ 3–3 $f(n) \in O(g(n))$

โอเมกาใหญ่

เรามีแบบโตช้ากว่า (ο) โตเร็วกว่า (ω) โตเท่ากัน (Θ) และโตไม่เร็วกว่า (O) ก็ต้องปิดท้ายด้วยโตไม่ช้ากว่า นั่นคือเราเขียน $f(n) \in \Omega(g(n))$ เพื่อบอกว่า $f(n)$ เป็นฟังก์ชันที่โต<u>ไม่ช้ากว่า</u> $g(n)$ นั่นคือ $\Omega(g(n)) = \omega(g(n)) \cup \Theta(g(n))$ คือเป็นเซตที่รวมฟังก์ชันที่เร็วกว่าและที่โตเท่ากับ $g(n)$ หรือเขียนเป็นนิยามได้อีกแบบหนึ่งดังนี้

$$\Omega(g(n)) = \{ f(n) \mid \text{ มีค่าคงตัวบวกสองตัวคือ } c \text{ และ } n_0 \text{ ที่ทำให้ } cg(n) \le f(n) \text{ เมื่อ } n \ge n_0 \}$$

หมายความว่าถ้า $f(n) \in \Omega(g(n))$ แสดงว่าการเติบโตของ $f(n)$ จะถูกกำหนดขอบเขตด้านล่างไว้ด้วยลักษณะการเติบโตของ $g(n)$ นั่นคือเราสามารถหาค่าคงตัวบวก c ที่ $cg(n) \le f(n)$ แสดงเป็นตัวอย่างได้ดังรูปที่ 3–4



รูปที่ 3–4 $f(n) \in \Omega(g(n))$

คุณสมบัติของสัญกรณ์เชิงเส้นกำกับ

สัญกรณ์ทั้งหลายที่ได้นำเสนอมานี้มีคุณสมบัติที่น่าสนใจดังนี้

Transitivity :

$f(n) \in \Theta(g(n))$ และ $g(n) \in \Theta(h(n))$ จะได้ว่า $f(n) \in \Theta(h(n))$

$f(n) \in \mathrm{O}(g(n))$ และ $g(n) \in \mathrm{O}(h(n))$ จะได้ว่า $f(n) \in \mathrm{O}(h(n))$

$f(n) \in \Omega(g(n))$ และ $g(n) \in \Omega(h(n))$ จะได้ว่า $f(n) \in \Omega(h(n))$

$f(n) \in \mathrm{o}(g(n))$ และ $g(n) \in \mathrm{o}(h(n))$ จะได้ว่า $f(n) \in \mathrm{o}(h(n))$

$f(n) \in \omega(g(n))$ และ $g(n) \in \omega(h(n))$ จะได้ว่า $f(n) \in \omega(h(n))$

Reflexivity :

$f(n) \in \Theta(f(n))$

$f(n) \in \mathrm{O}(f(n))$

$f(n) \in \Omega(f(n))$

Symmetry :

$f(n) \in \Theta(g(n))$ ก็ต่อเมื่อ $g(n) \in \Theta(f(n))$

Transpose symmetry :

$f(n) \in \mathrm{O}(g(n))$ ก็ต่อเมื่อ $g(n) \in \Omega(f(n))$

$f(n) \in \mathrm{o}(g(n))$ ก็ต่อเมื่อ $g(n) \in \omega(f(n))$

ขอให้ระวังไว้นิดหนึ่งว่า การเปรียบเทียบฟังก์ชันสองฟังก์ชันตามอัตราการเติบโตโดยใช้สัญกรณ์ที่กล่าวมานี้ อาจกระทำได้ไม่เสมอไป อาทิเช่น n กับ $n^{1+\sin(n)}$ เนื่องจากจำนวนยกกำลัง 1+ sin(n) มีค่าแกว่งไปมาระหว่าง 0 กับ 2 เมื่อ n มีค่าเพิ่มขึ้น เป็นต้น

เราสามารถจัดการกับเซตเชิงเส้นกำกับที่กล่าวมาได้ หลากหลายรูปแบบเพื่อเป็นการลดหรือเปลี่ยนให้อยู่ในรูปแบบที่ง่ายขึ้นได้อาทิเช่น (จะไม่ขอแสดงวิธีพิสูจน์ให้ดู)

กำหนดให้ $f_1(n) = \mathrm{O}(g_1(n))$ และ $f_2(n) = \mathrm{O}(g_2(n))$

$$f_1(n) + f_2(n) = \mathrm{O}(g_1(n) + g_1(n))$$

$$f_1(n) + f_2(n) = \mathrm{O}(\max(g_1(n), g_1(n)))$$

$$f_1(n) \cdot f_2(n) = \mathrm{O}(g_1(n) \cdot g_1(n))$$

$$f_1(n)^k = \mathrm{O}(g_1(n)^k)$$

$$\sum_{k=1}^{n} O(f(k)) = O\left(\sum_{k=1}^{n} f(k)\right)$$

ตัวอย่างที่ 3-4 จงแสดงให้เห็นจริงว่า $2n^2 + 500n + 1000\log n = \mathrm{O}(n^2)$

ต้องหาค่า c และ n_0 ที่ทำให้ $2n^2 + 500n + 1000\log n \le cn^2$ เป็นจริงเสมอเมื่อ $n \ge n_0$ ให้ c = 1502 ก็สบายใจได้เลยว่าอสมการนี้เป็นจริงแน่เมื่อ $n \ge 1$

ตัวอย่างที่ 3-5 จงแสดงให้เห็นจริงว่า $2n^2 + 500n + 1000\log n = \mathrm{O}(n^{200})$

จากผลของตัวอย่างที่ 3-4 $2n^2 + 500n + 1000\log n = \mathrm{O}(n^2)$ และความจริงที่แทบไม่ต้องแสดงให้เห็นว่า $n^2 \le n^{200} = \mathrm{O}(n^{200})$ ดังนั้น $2n^2 + 500n + 1000\log n = \mathrm{O}(n^{200})$ บางคนเขาเรียกการระบุขอบเขตบน ซึ่งสูงกว่าที่ควรจะเป็น (เช่นในตัวอย่างนี้) หรือในทางกลับกันขอบเขตล่างซึ่งต่ำกว่าที่ควรจะเป็นว่า *ขอบเขตหลวม* (loose bound)

ตัวอย่างที่ 3-6 จงแสดงให้เห็นจริงว่า $(n/2) \lg (n/2) = \Omega(n \lg n)$

ต้องหาค่า c และ n_0 ที่ทำให้ $cn \lg n \le (n/2) \lg (n/2)$ เมื่อ $n \ge n_0$ เขียนใหม่ได้เป็น

$cn \lg n \le (n/2) \lg n - (n/2) \lg 2$ หารด้วย $n \lg n$ ตลอดได้ $c \le (1/2) - (1/2)(\lg 2)/(\lg n)$ ให้ $n = 4$ จะได้ค่า $c \le (1/2) - (1/2)(\lg 2)/(\lg 4) = 1/4$ ดังนั้นอสมการข้างต้นเป็นจริงเมื่อ $c = 1/4$ และ $n_0 = 4$

ตัวอย่างที่ 3-7 จงแสดงให้เห็นจริงว่า $\sum_{k=1}^{n} k = \Omega(n^2)$

ข้อนี้ง่าย ใครๆ ก็รู้ว่าสูตรสำเร็จของผลบวกในโจทย์คือ $n(n+1)/2 = n^2/2 + n/2 \ge (1/2)n^2$ สำหรับทุกๆ $n \ge 0$ หรือเราจะพิสูจน์โดยใช้กลวิธีการแยกผลบวก โดยไม่ต้องรู้สูตรสำเร็จก็ได้ดังนี้

$$\begin{aligned}
\sum_{k=1}^{n} k &= \sum_{k=1}^{\lfloor n/2 \rfloor} k + \sum_{k=\lfloor n/2 \rfloor+1}^{n} k \\
&\ge \sum_{k=1}^{n/2} 0 + \sum_{k=\lfloor n/2 \rfloor+1}^{n} \frac{n}{2} \\
&\ge (n/2)^2 \\
&= \Omega(n^2)
\end{aligned}$$

ตัวอย่างที่ 3-8 จงแสดงให้เห็นจริงว่า $H_n = \sum_{k=1}^{n} \frac{1}{k} = \mathrm{O}(\log n)$

เราสามารถแยก $\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$ เป็น หลายๆ ชุดในลักษณะดังนี้

$\left(\frac{1}{1+0}\right), \left(\frac{1}{2+0} + \frac{1}{2+1}\right), \left(\frac{1}{4+0} + \frac{1}{4+1} + \frac{1}{4+2} + \frac{1}{4+3}\right), \left(\frac{1}{8+0} + \frac{1}{8+1} + \ldots + \frac{1}{8+7}\right), \ldots$ เขียนใหม่ได้เป็น

$$\begin{aligned}
\sum_{k=1}^{n} \frac{1}{k} &\le \sum_{i=0}^{\lfloor \lg n \rfloor} \sum_{j=0}^{2^i-1} \left(\frac{1}{2^i + j}\right) \\
&\le \sum_{i=0}^{\lfloor \lg n \rfloor} \sum_{j=0}^{2^i-1} \left(\frac{1}{2^i}\right) \\
&\le \sum_{i=0}^{\lfloor \lg n \rfloor} 1 \\
&\le \lg n + 1 \\
&= O(\log n)
\end{aligned}$$

- ทางขวาอาจมีจำนวนพจน์ที่ผลบวกมากกว่าทางซ้าย จึงเป็นขอบเขตบน
- ตัด j ทิ้ง ยังคงเป็นขอบเขตบน
- $\sum_{j=0}^{2^i-1} \left(\frac{1}{2^i}\right) \le 1,\ i \ge 0$

ตัวอย่างที่ 3-9 จงแสดงให้เห็นจริงว่า $H_n = \sum_{k=1}^{n} \frac{1}{k} = \mathrm{O}(\log n)$

คราวนี้หาอีกแบบโดยการประมาณผลบวกด้วยปริพันธ์ นั่นคือ

$$\int_{m-1}^{n} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)dx$$

สำหรับกรณีที่ $f(x)$ เป็นฟังก์ชันที่เพิ่มทางเดียว (monotonically increasing) แต่ถ้าเป็นฟังก์ชันที่ลดทางเดียว (monotonically decreasing) เราจะประมาณได้ดังนี้

$$\int_{m}^{n+1} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m-1}^{n} f(x)dx$$

เนื่องจาก $\frac{1}{k}$ เป็นฟังก์ชันลดทางเดียว $\sum_{k=2}^{n} \frac{1}{k} \le \int_{1}^{n} \frac{1}{k} dx = \ln n$ ดังนั้น $H_n \le 1 + \ln n = \mathrm{O}(\log n)$

ตัวอย่างที่ 3-10 จงแสดงให้เห็นจริงว่า $\sum_{i=1}^{n} i^k = \Theta\left(n^{k+1}\right)$ โดยที่ k เป็นค่าคงตัว

อีกวิธีหนึ่งในการแสดง $f(n) \in \Theta(g(n))$ นอกจากจะใช้การหาลิมิตหรือการหาค่าสามค่าเพื่อแสดงขอบเขตกระชับดังที่นิยามไว้ ก็คือการแสดงให้เห็นว่า $f(n) \in \mathrm{O}(g(n))$ และ $f(n) \in \Omega(g(n))$ หมายความว่า $g(n)$ มีลักษณะการเติบโตที่เป็นทั้งขอบเขตบนและขอบเขตล่างของ $f(n)$ จากโจทย์ เราจะแสดงให้เห็นว่า $\sum_{i=1}^{n} i^k = \mathrm{O}\left(n^{k+1}\right)$ และ $\sum_{i=1}^{n} i^k = \Omega\left(n^{k+1}\right)$

เริ่มด้วยขอบเขตบนก่อน เนื่องจากภายในผลบวกนั้น i มีค่าตั้งแต่ 1 ถึง n แสดงว่า $i \le n$ สรุปได้ว่า $i^k \le n^k$ ดังนั้นเมื่อรวมทุกๆ i ตั้งแต่ 1 ถึง n ย่อมได้ว่า $\sum_{i=1}^{n} i^k \le \sum_{i=1}^{n} n^k = n^{k+1} = \mathrm{O}(n^{k+1})$

สำหรับขอบเขตล่าง ถ้าเราหาผลบวกของ i^k สำหรับทุกๆ i ตั้งแต่ $\lceil n/2 \rceil$ ถึง n ย่อมได้ค่าไม่มากกว่าผลบวกที่ต้องการหา ดังนั้น $\sum_{i=1}^{n} i^k \ge \sum_{i=\lceil n/2 \rceil}^{n} i^k$ ถ้าเราแทน i^k ในผลบวกทางขวาด้วย $(n/2)^k$ จะได้ว่า $\sum_{i=\lceil n/2 \rceil}^{n} i^k \ge \sum_{i=\lceil n/2 \rceil}^{n} (n/2)^k \ge (n/2)^{k+1} = (1/2)^{k+1} n^{k+1} = \Omega(n^{k+1})$

จากที่แสดงให้เห็นว่า $\sum_{i=1}^{n} i^k = O(n^{k+1})$ และ $\sum_{i=1}^{n} i^k = \Omega(n^{k+1})$ ดังนั้น $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$

ตัวอย่างที่ 3-11 จงแสดงให้เห็นจริงว่า $\log n! = \Theta(n \log n)$

เราจะเริ่มด้วยการพิสูจน์ว่า $\log n! = O(n \log n)$ จากนิยามของแฟกทอเรียล $n! = n\cdot (n-1) \ldots 2\cdot 1$ ขอแทนทุกๆ พจน์ทางขวาด้วย n จะได้ว่า $n! \le n^n$ หาค่า log ได้ $\log n! \le n \log n = O(n \log n)$

ต่อมาจะพิสูจน์ว่า $\log n! = \Omega(n \log n)$ จากนิยามของแฟกทอเรียล $n! = n\cdot (n-1) \ldots 2\cdot 1$ คราวนี้ขอแทนพจน์ $n, (n-1), \ldots, (n/2)$ ด้วย $(n/2)$ และแทนพจน์ $(n/2-1), (n/2-2), \ldots, 2, 1$ ด้วย 1 จะได้ว่า $n! \ge (n/2)^{n/2}$ หาค่า log จะได้ $\log n! \ge (n/2) \log (n/2) = \Omega(n \log n)$ (จากตัวอย่างที่ 3-6)

จากขอบเขตบนและล่างที่แสดงให้เห็นจริงแสดงว่า $\log n! = \Theta(n \log n)$

ตัวอย่างที่ 3-12 จงแสดงให้เห็นจริงว่า $\log_a n = \Theta(\log_b n)$ สำหรับค่าคงตัว $a, b > 1$

เนื่องจากเราสามารถแปลงฐานของ log ได้ จาก $\log_a n = (\log_b n) / (\log_b a) = \Theta(\log_b n)$

ตัวอย่างข้างบนนี้ต้องการชี้ให้เห็นว่าเรามักจะไม่ใส่ฐานของ log ในสัญกรณ์เชิงเส้นกำกับ เพราะว่านอกจากจะสะดวกแล้ว ยังไม่มีผลใดๆ ด้วย ดังนั้นในเชิงเส้นกำกับแล้ว $\log_{1.1} n$ กับ $\log_{100} n$ มีอัตราการเติบโตเท่ากัน

ตัวอย่างที่ 3-13 จงแสดงให้เห็นจริงว่า $\log n^a = \Theta(\log n)$ สำหรับค่าคงตัวบวก a

เห็นได้ชัดว่า $\log n^a = a \log n = \Theta(\log n)$

นี่ก็เป็นอีกตัวอย่างที่ต้องการแสดงให้เห็นว่าเลขชี้กำลังภายใน log นั้นไม่มีความหมายใดๆ ต่ออัตราการเติบโต ดังนั้น $\log n^{1000}$ กับ $\log \sqrt{n}$ มีอัตราการเติบโตเท่ากัน (อันนี้จะขัดกับความรู้สึกในครั้งแรกที่พบ แต่ขอให้เข้าใจด้วยว่ามันเป็นพฤติกรรมของฟังก์ชันเมื่อ n มีค่ามาก)

ตัวอย่างที่ 3-14 จงแสดงให้เห็นจริงว่า $a^{\lg n} \ne \Theta(a^{\log n})$

หวังว่าทุกคนคงไม่ลืมเอกลักษณ์ $a^{\log_b n} = n^{\log_b a}$ ดังนั้น $a^{\lg n} = n^{\lg a}$ ในขณะที่ $a^{\log n} = n^{\log a}$ เนื่องจาก lg n ไม่เท่ากับ log n ดังนั้น $n^{\lg a} \neq \Theta(n^{\log a})$ แสดงว่า $a^{\lg n} \neq \Theta(a^{\log n})$

ตัวอย่างข้างบนนี้ต้องการเน้นว่าฐานของ log ที่เป็นเลขชี้กำลังของพจน์อื่น จะมาตัดทิ้งโดยไม่พิจารณาไม่ได้ ดังนั้นกล่าวโดยสรุปว่า ถ้าจะตัดไม่พิจารณาฐานของ log ก็ขอให้ระวังๆ กันหน่อย

ตัวอย่างที่ 3-15 จงแสดงให้เห็นจริงว่า $\sum_{h=0}^{k}\left(\frac{n}{2^h}O(h)\right) = O(n)$ โดยที่ $k = \lfloor \lg n \rfloor$

ถ้ายังจำกันได้ ผลบวกข้างบนนี้คือเวลาของการสร้างฮีปแบบทวิภาค (binary heap) โดยใช้วิธีการค่อยๆ ดันข้อมูลให้ไหลลง (percolate down) ย้อนจากข้อมูลในแถวลำดับตัวสุดท้ายมายังตัวแรก เราสามารถดึงโอใหญ่ซึ่งอยู่ภายในผลบวกออกมาอยู่นอกผลบวกได้ จากนั้นจะทำให้เราจัดการกับผลบวกได้ง่ายขึ้น อีกทั้งเมื่อเรารู้ว่าคำตอบจะเป็นโอใหญ่ซึ่งเป็นขอบเขตบน ทำให้เรากล้าที่จะขยายผลบวกให้รวมพจน์จำนวนมากขึ้นเพื่อลดรูปผลเฉลยที่ได้ให้สวยขึ้น ดังนี้

$$\sum_{h=0}^{k}\left(\frac{n}{2^h}O(h)\right) = O\left(n\sum_{h=0}^{k}\left(\frac{h}{2^h}\right)\right) = O\left(n\sum_{h=0}^{\infty}\left(\frac{h}{2^h}\right)\right) = O(2n) = O(n)$$

ขอใช้เครื่องหมาย = แทน $\in$

บางคนอาจสังเกตเห็น และเกิดความสงสัยมาตั้งแต่อ่านตัวอย่างที่ผ่านมาแล้วว่า ตอนแรกก็นิยามให้สารพัด o, ω, Θ, O และ Ω เป็นเซต แล้วก็เขียนสวยๆ มาตลอดเช่น $f(n) \in \Theta(g(n))$ แล้วอยู่ดีๆ ก็มาใช้เครื่องหมาย = แทน $\in$ ในสองสามหน้าที่ผ่านมา ซึ่งนักเรียนมัธยมก็เห็นชัดๆ ว่าผิด ยอมรับครับว่ามันผิด แต่ก็ยังจงใจจะใช้เครื่องหมาย = ครับ ทั้งนี้ก็เพราะว่าผมเห็นมันแต่ไหนแต่ไรแล้วละครับว่าเขาใช้ = กันทั้งนั้น จึงจะขอใช้ผิดด้วยคน เนื่องจากมันชินตาและสะดวกดี (ไม่ต้องเปลี่ยนฟอนต์บ่อย) แต่มันก็มีเหตุผลบ้างเหมือนกันว่าทำไมเขาถึงใช้มันผิดแบบนี้ ลองอ่านเหตุผลที่ Donald Knuth (ปรมาจารย์ทางคอมพิวเตอร์) เขียนไว้ในหนังสือ Concrete Mathematics ดังนี้

- ใช้มานาน ก็เลยชิน

- ในวงการคอมพิวเตอร์เราใช้เครื่องหมาย = กันผิดๆ อยู่แล้ว (เช่น A = B มีหลายความหมายในภาษาคอมพิวเตอร์) ขอใช้ผิดอีกสักครั้งจะเป็นไร
- ปกติเราอ่าน $f(n) = \mathrm{O}(g(n))$ ว่า $f(n)$ *เป็น* โอใหญ่ของ $g(n)$ คืออ่าน "=" ว่า "เป็น" ซึ่งก็เป็นลักษณะของการเท่ากับทางเดียว (เช่น ลิงเป็นสัตว์ แต่ไม่ได้หมายความว่าสัตว์เป็นลิง) หรืออีกนัยหนึ่ง $f(n) = \mathrm{O}(g(n))$ ไม่ได้หมายความว่า $\mathrm{O}(g(n)) = f(n)$
- (อันนี้เข้าท่าหน่อย) การใช้ = จะทำให้เราจัดการกับนิพจน์ที่มีสัญกรณ์เชิงเส้นกำกับได้อย่างเป็นธรรมชาติ และง่ายขึ้น ดังที่จะกล่าวต่อไป

การใช้สัญกรณ์เชิงเส้นกำกับในสมการ

โดยทั่วไปเราเขียนบรรยายฟังก์ชันแบบละเอียดครบถ้วน เช่น $f(n) = 2n^3 + 3n + 7.5/n$ แต่ในบางครั้ง เราอาจละเลย ไม่อยากลงรายละเอียดพจน์ที่ไม่ค่อยสำคัญ เช่น การใช้เครื่องหมาย ≈ แล้วตัดส่วนที่คิดว่าไม่สำคัญทิ้ง เช่น เขียนเป็น $f(n) \approx 2n^3$ หรืออาจใช้สัญกรณ์เชิงเส้นกำกับช่วยก็จะได้ความหมายที่มากกว่าเครื่องหมาย ≈ เช่น เขียนเป็น $f(n) = 2n^3 + \Theta(n)$ เป็นการบอกว่า $f(n) = 2n^3 + g(n)$ โดยที่ $g(n) \in \Theta(n)$ การใช้เครื่องหมาย = นี้ถึงแม้จะผิดความหมาย แต่จะช่วยให้เราเข้าใจความหมายของ $f(n)$ ได้ง่ายขึ้น กล่าวคือ $f(n)$ ก็คือ $2n^3$ บวกอะไรบางอย่างที่มีอัตราการเติบโตเท่ากับ n ให้สังเกตว่าเราใช้สัญกรณ์เชิงเส้นกำกับกับส่วนของฟังก์ชันที่มีอัตราการเติบโตที่ช้ากว่า ส่วนที่โตเร็วกว่าก็ยังเขียนเหมือนเดิม นั่นคือเราคงไม่เขียน $f(n) = \Theta(n^3) + 3n + 7.5/n$ เพราะเขียนแบบนี้ $3n + 7.5/n$ ไม่เห็นมีความหมายใดๆ เนื่องจากมามันโตช้ากว่า n^3 ดังนั้นจะเหมือนกับเขียน $f(n) = \Theta(n^3)$

บางคนอาจอยากถามว่า แล้วเราไปหาเรื่องเขียนแบบคร่าวๆ ทำไม ถ้าเรารู้ตัวฟังก์ชันจริงๆ อยู่แล้ว อันนี้ถูกต้องถ้าเรารู้ของละเอียด ก็ไม่ต้องทำให้มันหยาบ แต่ที่เราจะใช้สัญกรณ์เชิงเส้นกำกับในสมการในลักษณะนี้นั้น ก็สำหรับกรณีที่เราไม่รู้ของละเอียด หรือถ้ารู้ของละเอียดก็ต้องเขียนกันยืดยาวมากๆ เช่น จากความรู้ในอดีตเกี่ยวกับเรื่องจำนวนฮาร์มอนิค H_n มีนิยามว่า

$$H_n = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n}$$

ที่คนปรารถนาอยากได้ก็คือรูปแบบปิดของ H_n แต่ก็หาไม่ได้ ที่จะพอมีให้เห็นก็เช่น

$$H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{\varepsilon_n}{120n^4}$$

โดยที่ $0 < \varepsilon_n < 1$ และ γ คือค่าคงตัวออย์เลอร์มีค่าเท่ากับ 0.5772156649…

แต่ถ้าเรากำลังสนใจว่า H_n มีแนวโน้มการเปลี่ยนแปลงค่าอย่างไร ก็อาจเขียนได้ดังนี้

$$H_n = \ln n + \gamma + \mathrm{O}(1/n)$$

เพื่อชี้ให้เห็นไปเลยว่าพจน์หลังๆ ที่ไม่ได้เขียนให้ดูนั้นโตไม่เกิน $1/n$ หรือจะเขียนแบบนี้

$$H_n = \ln n + \mathrm{O}(1)$$

ก็ไม่ผิดเหมือนกัน (แต่ละเอียดน้อยลงหน่อย) หรือสุดๆ เลยก็เขียนแบบนี้ก็ยังได้

$$H_n = \mathrm{O}(\log n)$$

และที่เราจะพบบ่อยมากในการใช้สัญกรณ์เชิงเส้นกำกับในสมการ ก็คือการเขียนเป็นส่วนหนึ่งของความสัมพันธ์เวียนเกิด เพื่อแทนประสิทธิภาพของอัลกอริทึม ตัวอย่างง่ายๆ เช่น การเรียงลำดับแบบผสาน (mergesort) นั้นเราแบ่งจำนวนข้อมูลออกเป็นสองส่วนเท่าๆ กัน แล้วไปเรียงลำดับทั้งสองส่วนให้เสร็จก่อนแล้วจึงมาผสานกัน ถ้ากำหนดให้ $t(n)$ คือเวลาในการเรียงลำดับข้อมูลจำนวน n ตัวแบบผสาน เราจะเขียนความสัมพันธ์เวียนเกิดได้เป็น

$$t(n) = 2t(n/2) + \text{เวลาในการผสาน}$$

ก็มาหากันว่าเวลาในการผสานใช้เวลาทั้งสิ้นเท่าไร เป็น $n–1$? เป็น n ? เอ๊ะหรือว่าไม่ใช่ทั้งสอง เพราะอาจมีค่าคงตัวคูณข้างหน้าด้วยก็ได้ แต่ขอให้สังเกตว่าไม่ว่าจะเป็น n หรือเป็น $n–1$ หรือว่ามีค่าคงตัวคูณอยู่ข้างหน้าก็ตาม เราเขียนดังนี้ได้

$$t(n) = 2t(n/2) + \Theta(n)$$

ซึ่งหมายความว่าการผสานข้อมูลนั้นใช้เวลาที่เป็นฟังก์ชันซึ่งโตแบบเชิงเส้น หลังการวิเคราะห์แล้วก็จะได้ $t(n)$ ในรูปแบบของสัญกรณ์เชิงเส้นกำกับด้วย

การใช้สัญกรณ์เชิงเส้นกำกับ

จากที่ได้กล่าวมาเกี่ยวกับความหมายของสัญกรณ์เชิงเส้นกำกับต่างๆ บางคนอาจรู้สึกว่าสัญกรณ์เหล่านี้ฟุ่มเฟือย บางตัวนิยามได้จากตัวอื่น ถึงแม้ว่าจะดูว่าฟุ่มเฟือย เช่น O กับ Ω เพราะมันคู่กันถ้า $f(n) = \mathrm{O}(g(n))$ ก็ต่อเมื่อ $g(n) = \Omega(f(n))$ เป็นต้น แต่เรามักใช้มันในความหมายที่แตกต่างกัน เพื่อให้เข้าใจง่ายขึ้น อาทิเช่น ประโยคต่างๆ ต่อไปนี้

- "Insertion sort ใช้เวลาในการเรียงลำดับข้อมูล n ตัวเป็น $O(n^2)$" หมายความว่าเวลาการทำงานเป็นฟังก์ชันที่โตไม่เร็วกว่า n^2
- "Insertion sort ใช้เวลาในการเรียงลำดับข้อมูล n ตัวเป็น $\Omega(n)$" บอกว่าอย่างน้อย Insertion sort ก็ต้องใช้เวลาที่เป็นฟังก์ชันที่โตแบบเชิงเส้น
- "อัลกอริทึมการเรียงลำดับข้อมูล n ตัวที่ใช้การเปรียบเทียบข้อมูลเป็นหลัก ย่อมใช้เวลาเป็น $\Omega(n \log n)$" หมายความว่าถ้าเราคิดอัลกอริทึมการเรียงลำดับอันหนึ่งที่ใช้เวลาเป็น $O(n \log n)$ แสดงว่าเราได้พบวิธีที่เรียงลำดับที่มีอัตราการเติบโตของเวลาการทำงานที่ดีที่สุดแล้ว

ตัวอย่างต่างๆ ข้างบนนี้ชี้ให้เห็นว่า เรามักบอกภาระขั้นต่ำของอัลกอริทึมที่แก้ปัญหาหนึ่งๆ ด้วยโอเมกาใหญ่ (นั่นคือเป็นขอบเขตล่าง) ในขณะที่เราจะใช้โอใหญ่สำหรับประสิทธิภาพการทำงานของอัลกอริทึม (นั่นคือระบุขอบเขตบน หรือกรณีที่ใช้เวลามากที่สุดของอัลกอริทึมนั้น)

อีกประเด็นที่เราได้พูดถึงมาก่อนหน้านี้ ก็คือการใช้สัญกรณ์เชิงเส้นกำกับมาเป็นตัวบอกพฤติกรรมการทำงานของอัลกอริทึม ถ้าอัลกอริทึม A ใช้เวลาการทำงานเป็น $\Theta(n^2)$ ก็แสดงว่าเรารู้ลักษณะการเติบโตของเวลาการทำงานที่แน่นอนว่าเป็นฟังก์ชันยกกำลังสอง แต่ถ้าบอกว่าอัลกอริทึม B ใช้เวลาการทำงานเป็น $O(n^5)$ ก็เพียงรู้ว่าเวลาการทำงานมีลักษณะการเติบโตที่ไม่เลวกว่า n^5 ซึ่งของจริงอาจเป็นเชิงเส้นก็ได้ (เพราะเพียงบอกแค่ขอบเขตบน) ถึงแม้ว่าเราจะรู้ว่าอัลกอริทึมของเรามีประสิทธิภาพเป็น $\Theta(n^2)$ เราก็คงไม่ไปเที่ยวบอกชาวบ้านว่าอัลกอริทึมของเรานั้นเป็น $O(n^{10})$ ซึ่งก็ไม่ผิด ไม่ได้โกหก แต่ก็ไม่น่าทำจริงไหม ? ดังนั้นโดยทั่วไปถ้าเราสามารถวิเคราะห์ประสิทธิภาพได้เป็น Θ จะเป็นการบรรยายพฤติกรรมที่แน่นอนกว่า แต่ถ้าเราวิเคราะห์โดยไม่รู้ขอบเขตล่าง (คือสรุปเป็น Θ ไม่ได้) เราก็ควรเป็นนักการตลาดที่ดีพอที่จะบอกประสิทธิภาพที่เป็นขอบเขตบนที่ต่ำสุดเท่าที่จะต่ำได้ ตัวอย่างเช่น การวิเคราะห์วิธีสร้างฮีปแบบทวิภาคด้วยการค่อยๆ ดันข้อมูลให้ไหลลง เพื่อปรับฮีป (บทที่ 6) ถ้าเราวิเคราะห์แบบลวกๆ ก็คือการดันข้อมูลให้ไหลลงนั้นเสียเวลาครั้งละ $O(\log n)$ ต้องทำ n ครั้งรวมเป็น $O(n \log n)$ ซึ่งก็เป็นคำตอบที่ไม่ผิด เพราะเป็นขอบเขตบน (จะบอกว่า $O(n^5)$ ก็ไม่ผิดเช่นกัน) แต่มันบ่งบอกถึงความเลวของวิธีที่คิดขึ้น แทนที่จะโฆษณาความดี ทั้งๆ ที่ถ้าเราวิเคราะห์ละเอียดหน่อยดังที่ทำในตัวอย่างที่ 3-15 ก็จะพบว่าแท้จริงแล้วมีประสิทธิภาพเป็น $O(n)$ (แต่จะวิเคราะห์ให้ละเอียดอีกสักเท่าใดก็คงไม่ทำให้ขอบเขตบนต่ำกว่า $O(n)$ เป็นแน่)

ขอปิดท้ายด้วยเรื่องการใช้สัญกรณ์เชิงเส้นกำกับอีกเรื่องหนึ่ง มาดูตัวอย่างการเขียนในบรรทัดข้างล่างนี้ว่าแปลกๆ หรือไม่

$$f(n) = n^2 = \Theta(5n^2 + 3n + 3)$$

ที่เขียนข้างบนนี้ก็ไม่มีอะไรผิด (นอกจากการใช้ =) แต่มันแปลกตรงที่ว่าจะไม่ค่อยเห็นใครเขาเขียนกัน ทั้งนี้ก็เพราะว่าจุดประสงค์ของการใช้สัญกรณ์เหล่านี้ก็เพื่อบรรยายฟังก์ชันที่ยุ่งด้วยกลุ่มของฟังก์ชันที่อ่านง่ายกว่า บรรยายอัตราการเติบโตที่ตีความได้ง่ายกว่า ดังนั้นส่วนใหญ่เราจะเห็นฟังก์ชันง่ายๆ อยู่ในวงเล็บของสัญกรณ์เช่น $g(n) = 5n^2 + 3n + 3 = \Theta(n^2)$ เพราะเราต้องการบรรยายฟังก์ชัน $g(n)$ ที่ยุ่งด้วย Θ เพื่อบอกว่ามีการอัตราการเติบโตเท่ากับ n^2 ที่ตีความได้ง่ายๆ

แบบฝึกหัด

1. จงอธิบายความหมายของ $O(1)$ $\Theta(1)$ และ $\Omega(1)$
2. ถ้า $f_1(n) = \Omega(g_1(n))$ และ $f_2(n) = \Omega(g_2(n))$ แล้ว $f_1(n) + f_2(n) = \Omega(\min\{g_1(n), g_2(n)\})$ หรือว่า $f_1(n) + f_2(n) = \Omega(\max\{g_1(n), g_2(n)\})$ พิสูจน์ให้ดูด้วย
3. ถ้า $t_1(n) = O(f(n))$ และ $t_2(n) = O(f(n))$ ข้อใดต่อไปนี้เป็นจริง
   ก) $t_1(n) + t_2(n) = O(f(n))$
   ข) $t_1(n) - t_2(n) = o(f(n))$
   ค) $t_1(n) / t_2(n) = O(1)$
   ง) $t_1(n) = O(t_2(n))$
4. จงเรียงลำดับฟังก์ชันข้างล่างนี้ตามอัตราการเติบโต

$\sqrt{n}$	3^n	$n^2 + \log n$	$n^{0.01}$	$\log \log n$	$n!$
$n / \log n$	$(1.01)^n$	10^6	3^{-n}	n^n	$(\log n)^{10}$

5. จงหาความสัมพันธ์ของ $f(n)$ และ $g(n)$ ต่างๆ ข้างล่างนี้ว่า $f(n) = O(g(n))$ หรือ $g(n) = O(f(n))$
   ก) $f(n) = (n^2 - n) / 7$ $\quad g(n) = 0.5n$
   ข) $f(n) = n^{0.03}$ $\quad g(n) = \lg(n^2+3n)$
   ค) $f(n) = n \log n$ $\quad g(n) = n\sqrt{n}$

ง) $f(n) = 9(\log n)^3$ $g(n) = 9 \log n^3$

6. จงหาว่าข้อย่อยต่อไปนี้ จริงหรือเท็จ

ก) $n \log n = O(n^{1.5})$

ข) $\sqrt{n} + \log n = O(n)$

ค) $2n^2 + \sqrt{n} = O(n + n\sqrt{n} + 20n)$

ง) $\sqrt{n} \log n = O(n)$

จ) $n + \sqrt{n} = O(\sqrt{n} \log n)$

ฉ) $2^n = \Theta(2^{n+1})$

ช) $n! = \Theta((n+1)!)$

7. จงพิสูจน์ หรือพิสูจน์แย้งข้อย่อยต่อไปนี้

ก) $n^{\log n} = O((\log n)^n)$

ข) $n^{\log \log \log n} = O((\log n)!)$

ค) $(n!)! = O(((n-1)!)!\,((n-1)!)^{n!})$

8. กำหนดให้ $f(n)$ และ $g(n)$ เป็นฟังก์ชันที่ให้ค่าบวก จงพิสูจน์ หรือพิสูจน์แย้งข้อย่อยต่อไปนี้

ก) $f(n) = \Theta(f(n/2))$

ข) $f(n) = O((f(n))^2)$

ค) $f(n) + o(f(n)) = \Theta(f(n))$

ง) ถ้า $f(n) = O(g(n))$ แล้ว $2^{f(n)} = O(2^{g(n)})$

จ) ถ้า $f(n) = O(g(n))$ แล้ว $\log f(n) = O(\log g(n))$

9. จงพิสูจน์ว่าสำหรับจำนวนเต็มบวก k ใดๆ $\sum_{i=1}^{n} i^k \lg i = O(n^{k+1} \log n)$

10. จงพิสูจน์ว่า $\sum_{i=1}^{n} \lceil \lg(n/i) \rceil = O(n)$