การวิเคราะห์อัลกอริทึม (ส่วนที่ 1)

ผศ.จรรยา สายนุ้ย

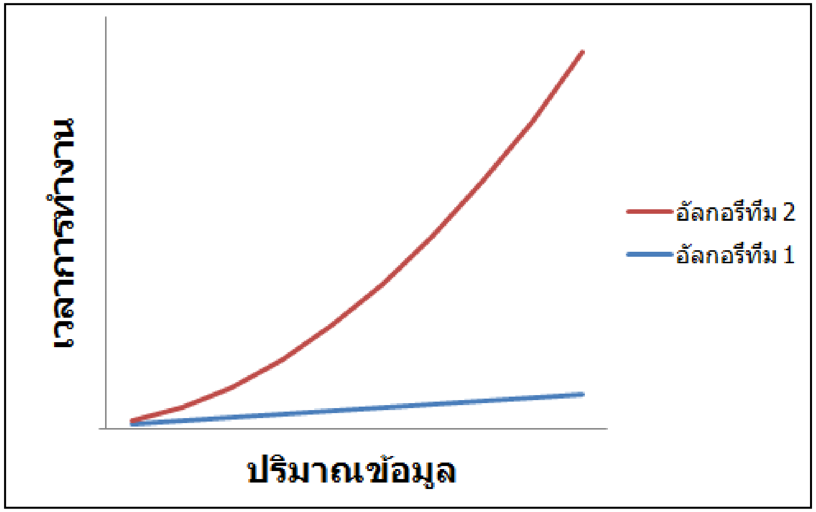
344-111 ชุดวิชาการโปรแกรมและขั้นตอนวิธี

เนื้อหา

- การวิเคราะห์อัลกอริทึม
- การนับจำนวนการทำงานของคำสั่งพื้นฐาน
- อัตราการเจริญเติบโต
- สัญกรณ์เชิงเส้นกำกับ

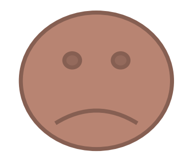
การวิเคราะห์อัลกอริทึม

- ศึกษาประสิทธิภาพของอัลกอริทึม
  - เวลาการทำงาน
  - ปริมาณหน่วยความจำ
- การวิเคราะห์เวลาการทำงาน



การวิเคราะห์เวลาการทำงาน

- Mathematical Analysis
- Experimental Analysis
  - เขียนโปรแกรม จับเวลาการทำงานจริง และบันทึกผล
  - โดยรันการทำงาน T รอบ หาค่าเฉลี่ยเวลาที่ใช้
  - นำผลของเวลาการทำงานเฉลี่ยมาเปรียบเทียบ
  - ข้อเสีย: ขึ้นอยู่กับหลายปัจจัย เช่น ภาษา คอมไพเลอร์ ตัวเครื่อง



การวิเคราะห์แบบคณิตศาสตร์

- ไม่ต้องเขียนโปรแกรม วิเคราะห์จากอัลกอริทึมโดยตรง
- โดยการ**นับจำนวน** **<u>คำสั่งตัวแทน</u>** (คำสั่งที่ทำงานเยอะที่สุด)
- <u>ตัวอย่าง</u>

```
for(k = n; k > 1; k--)
    maxI = 1
    for(i = 2; i <= k; i++)
        if (d[i] > d[maxI])
            maxI = i
        end if
    end for
    d[k] ↔ d[maxI]
end for
```

คำสั่งตัวแทน

$$\sum_{k=2}^{n}\left(\sum_{i=2}^{k}1\right)=\sum_{k=2}^{n}(k-1)=\sum_{j=1}^{n-1}j=\frac{n(n-1)}{2}$$

การวิเคราะห์เวลาการทำงาน

- เวลาการทำงานแปรตามจำนวนคำสั่ง
- จำนวนการทำงานของคำสั่งแปรตาม จำนวนการทำงานของคำสั่งตัวแทน
- การวิเคราะห์เชิงเวลา
  - ดูความสัมพันธ์ระหว่างปริมาณข้อมูล กับจำนวนการทำงานของคำสั่งตัวแทน
- ปริมาณข้อมูล
  - ข้อมูลที่มีผลตัวความเร็วของอัลกอริทึม

เปรียบเทียบการวิเคราะห์

- Algorithm1:

```
for(i=1; i<=n;i++)
    for(j=i+1; j<= n; j++)
        sum = sum + cosine(d[i][j])*i
```

$$\sum_{i=1}^{n}\left(\sum_{j=i+1}^{n} 1\right) = \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n} n - \sum_{i=1}^{n} i = n^2 - \frac{n(n+1)}{2}$$

$$= \frac{n^2}{2} - \frac{n}{2}$$

- Algorithm 2:

```
for(i=1; i<=n; i++)
    for(j=1; j<= n; j++)
        sum = sum + j
```

$$\sum_{i=1}^{n}\sum_{j=1}^{n} 1 = \sum_{i=1}^{n} n = n^2$$

เปรียบเทียบการวิเคราะห์

- Algorithm1:

```
for(i=1; i<=n;i++)
    for(j=i+1; j<= n; j++)
        sum = sum + cosine(d[i][j])*i
```

$$\frac{n^2}{2} - \frac{n}{2}$$

จำนวนการทำงานของคำสั่งตัแทน น้อยกว่า แต่คำสั่งตัวแทนใช้เวลามากกว่า

- Algorithm 2:

```
for(i=1; i<=n; i++)
    for(j=1; j<= n; j++)
        sum = sum + j
```

$$n^2$$

จำนวนการทำงานของคำสั่งตัแทน มากกว่า แต่คำสั่งตัวแทนใช้เวลาน้อยกว่า

- สรุปยากว่าอัลกอริทึมใดเร็วกว่ากัน

เปรียบเทียบอัตราการเติบโต

ขนาดข้อมูล (n)	ใช้เวลา			
	n^2	อัตราการเติบโต	$\frac{n^2}{2}-\frac{n}{2}$	อัตราการเติบโต
10	100		45	
20	400	4	190	4.22
40	1600	4	780	4.11
80	6400	4	3160	4.05
160	25600	4	12720	4.03
320	102400	4	51040	4.01
640	409600	4	204480	4.01
1280	1638400	4	818560	4.00

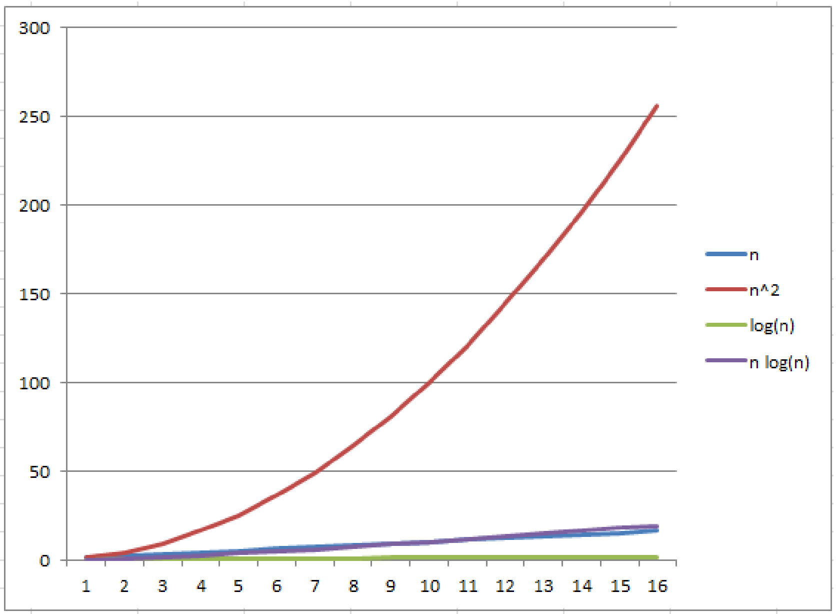
มีอัตราการเติบโตเท่ากัน

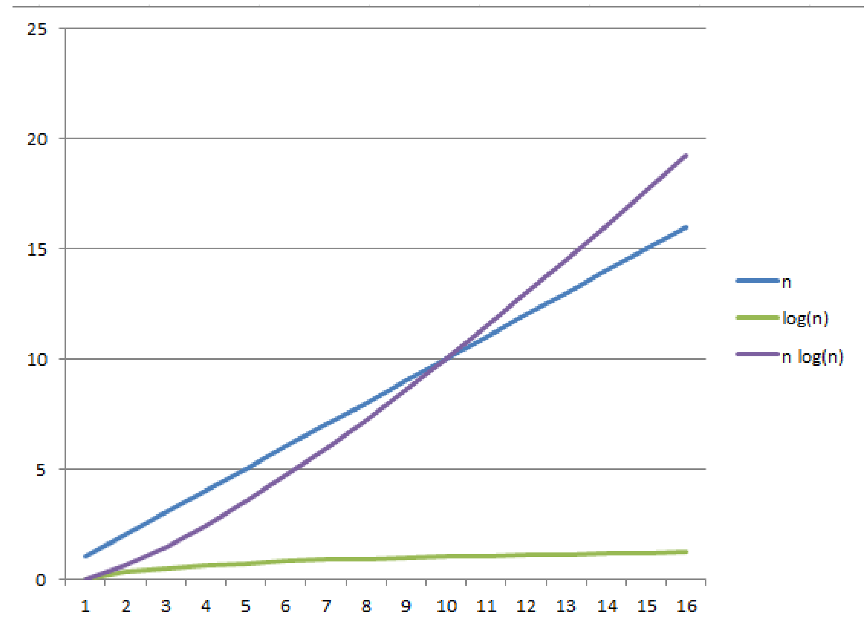
เปรียบเทียบอัตราการเติบโต

ขนาดข้อมูล (n)	ใช้เวลา			
	10n	อัตราการเติบโต	$\frac{n^2}{100}$	อัตราการเติบโต
10	100		1	
20	200	2	4	4
40	400	2	16	4
80	800	2	64	4
160	1600	2	256	4
320	3200	2	1024	4
640	6400	2	4096	4
1280	12800	2	16384	4
5120	51200	2	262144	4

$n^2/100$ มีอัตราการเติบโตเร็วกว่า 10n

เปรียบเทียบอัตราการเติบโต





อัตราการเติบโต (f(n) vs. g(n))

- ให้ f(n) และ g(n) แทนฟังก์ชันแสดงเวลาที่ได้จากการวิเคราะห์การทำงานของอัลกอริทึมที่ 1 และอัลกอริทึมที่ 2 ตามลำดับ
- เมื่อเราต้องการเปรียบเทียบว่า อัลกอริทึมใดมีอัตราการเติบโตเร็วกว่ากัน
- สามารถทำได้โดยนำเอา 2 ฟังก์ชันมาหารกัน แล้วเทค limit เมื่อ n เข้าใกล้ infinity
- ผลออกมามี 3 รูปแบบ ดังนี้

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \begin{cases} 0 & \text{บอกได้ว่า f(n) โตช้ากว่า g(n)} \\ c > 0 & \text{บอกได้ว่า f(n) โตเท่ากับ g(n)} \\ \infty & \text{บอกได้ว่า f(n) โตเร็วกว่า g(n)} \end{cases}$$

ตัวอย่าง

1. $\mathrm{f(n)} = 10\mathrm{n},\ \mathrm{g(n)} = \dfrac{\mathrm{n}^2}{1000}$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{10n}{\left(\frac{n^2}{1000}\right)} = \lim_{n\to\infty}\frac{10000n}{n^2} = \lim_{n\to\infty}\frac{10000}{n} = \mathbf{0}$$

- f(n) โตช้ากว่า g(n)

2. $\mathrm{f(n)} = 2\mathrm{n}^2 - 5\mathrm{n},\ \mathrm{g(n)} = 10\mathrm{n}^2$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{2\mathrm{n}^2-5\mathrm{n}}{10\mathrm{n}^2} = \lim_{n\to\infty}\left(\frac{2n^2}{10n^2} - \frac{5n}{10n^2}\right)$$

$$= \lim_{n\to\infty}\left(\frac{1}{5} - \frac{1}{5n}\right) = \mathbf{\frac{1}{5}}$$

f(n) โตเท่ากับ g(n)

สัญกรณ์เชิงเส้นกำกับ (Asymptotic Notation)

สัญกรณ์เชิงเส้นกำกับ เป็นวิธีการแทนความสัมพันธ์ในแง่อัตราการเติบโตอีกรูปแบบหนึ่ง

- little – o (o)
- little – omega (ω)
- Big – O (O)
- Big – Omega (Ω)
- Big – Theta (Θ)

tittle – o: โตช้ากว่า

- นิยาม:

$$\text{o}(g(n)) = \{f(n) \mid f(n) \text{ โตช้ากว่า } g(n)\}$$

- ตัวอย่าง:

$$n^{0.9} \in o(n)$$

$$10^{5} \in o(n)$$

tittle-omega: โตเร็วกว่า

- นิยาม:

$$\omega(g(n)) = \{f(n) \mid f(n) \text{ โตเร็วกว่า } g(n)\}$$

- ตัวอย่าง:

$$n^{1.01} \in \omega(n)$$

$$n^2 \in \omega(n)$$

$$2^n \in \omega(n)$$

Big-Theta: โตเท่ากัน

- นิยาม:

$$\Theta\big(g(n)\big) = \{f(n) \mid f(n) \text{ โตเท่ากัน } g(n)\}$$

- ตัวอย่าง:

$$10 \log n \in \Theta(\log n)$$

$$10 + \log n^5 \in \Theta(\log n)$$

Big – O: โตไม่เร็วกว่า (ช้ากว่าหรือเท่ากับ)

- นิยาม:

$$O\big(g(n)\big) = \{f(n) \mid f(n) \text{ โตไม่เร็วกว่า } g(n)\}$$

- ตัวอย่าง:

$$n \in O(n^2)$$

$$5n^2 + 2n \in O(n^2)$$

$$\frac{1}{2}n(n-1) \in O(n^2)$$

Big-Omega: โตไม่ช้ากว่า (เร็วกว่าหรือเท่ากับ)

- นิยาม:

$$\Omega\big(g(n)\big) = \{f(n) | f(n) \text{ โตไม่ช้ากว่า } g(n)\}$$

- ตัวอย่าง:

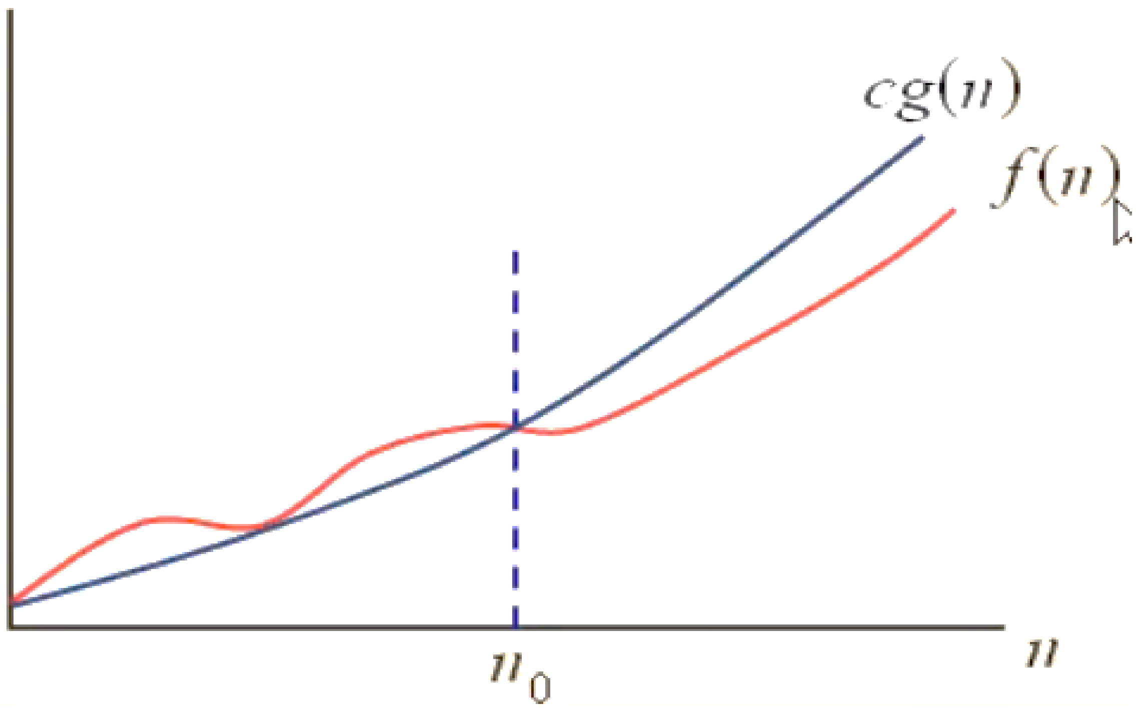
$$5n^3 + 2n \in \Omega(n^2)$$

$$n^3 \in \Omega(n \log n)$$

Big – O: บอกขอบเขตบน

$$f(n) \in O(g(n))$$

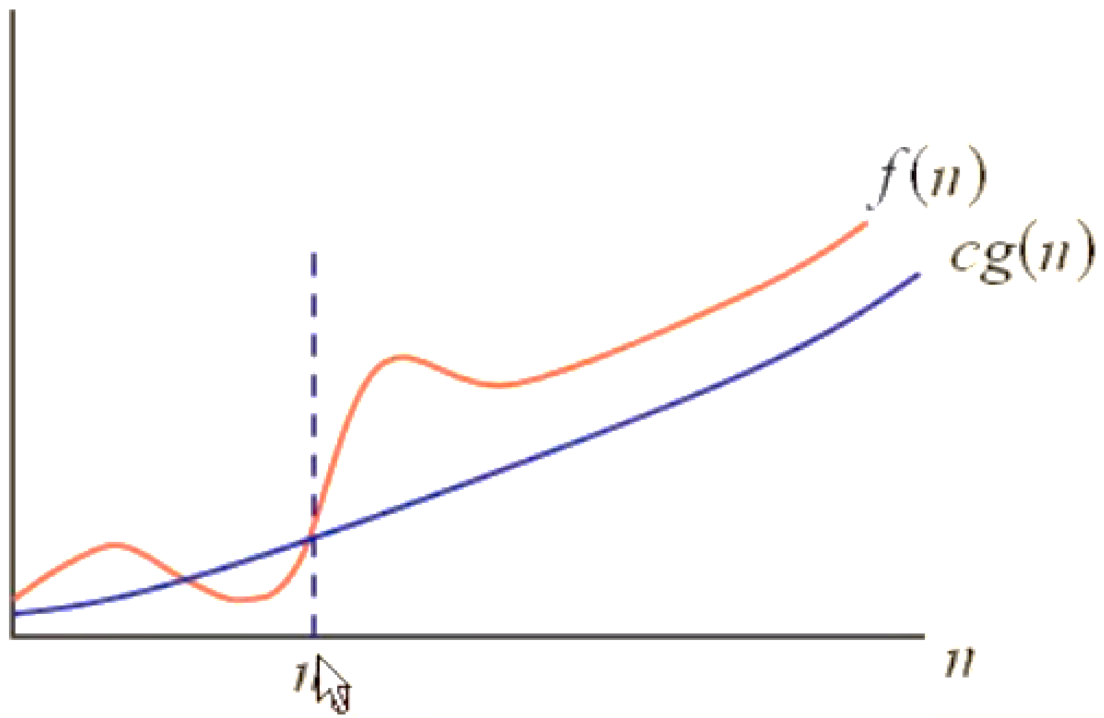
if $f(n) \leq cg(n)$ for all $n \geq n_0$ and $c \in R^+$



Big-Omega: บอกขอบเขตล่าง

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

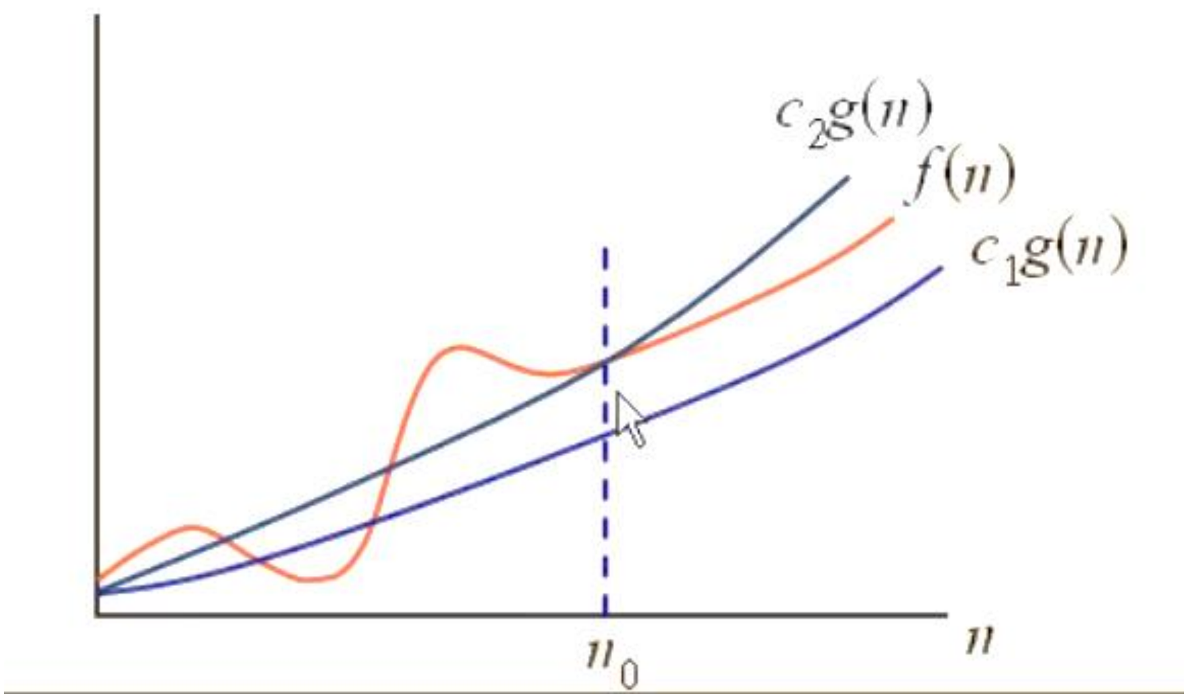
if $f(n) \geq cg(n)$ for all $n \geq n_0$ and $c \in R^+$



Big-Theta: บอกขอบเขตกระชับ

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

if $c_2 g(n) \geq f(n) \geq c_1 g(n)$ for all $n \geq n_0$ and $c_1, c_2 \in R^+$



Summary

- การวิเคราะห์เวลาการทำงานของอัลกอริทึม
  - เขียนโปรแกรมจริง เพื่อจับเวลา
  - วิเคราะห์แบบคณิตศาสตร์
    - นับคำสั่งตัวแทน จากอัลกอริทึม ไม่ต้องเขียนโปรแกรม
      - คำสั่งตัวแทนคือ คำสั่งที่ทำงานเยอะที่สุด
    - เขียนให้อยู่ในรูปของสัญกรณ์เชิงเส้นกำกับ
- การเปรียบเทียบอัลกอริทึม
  - ให้ดูจากอัตราการเติบโตของฟังก์ชัน
    - เราสนใจฟังก์ชันที่มีการเติบโตช้า
  - ใช้การหา limit เข้าช่วยในการเปรียบเทียบ 2 อัลกอริทึม
  - ใช้สัญกรณ์เชิงเส้นกำกับช่วยในการจัดกลุ่มฟังก์ชัน ทำให้เปรียบเทียบเวลาการทำงานได้ง่าย และเข้าใจมากยิ่งขึ้น

อ้างอิง

- สมชาย ประสิทธิ์จูตระกูล, การออกแบบและวิเคราะห์อัลกอริทึม
- Anany Levitin, Introduction to the Design and Analysis of Algorithm, second edition.