Algorithms

Asymptotic Notation

ปรับปรุงจาก slide รายวิชา 204451 ของอ.เบญจมาศ ปัญญางาม

การเปรียบเทียบการทำงานของอัลกอริทึม

เราจะเปรียบเทียบการทำงานของอัลกอริทึม 2 อันว่าอันไหนดีกว่าอันไหนได้อย่างไร ตัวอย่างเช่น

Input Size	อัลกอริทึม A	อัลกอริทึม B
n	1000n	1.1^n
10	10000	3
100	100000	13781
1000	1000000	$2.5x10^{41}$
1000000	10^9	$4.8x10^{41392}$

Growth of the function

- เราสนใจการวัดประสิทธิภาพของอัลกอริทึมโดยพิจารณาจากอัตราการเติบโตของฟังก์ชัน (Growth of the function) เฉพาะกับข้อมูลเข้าขนาด n **ที่มีค่ามาก ๆ**
- เราพิจารณาภาพรวมของเวลาการทำงานเราจะ
  - ไม่สนใจ low order term (เทอมของ n ที่มีค่าน้อย)
  - ไม่สนใจค่า coefficient ที่เป็นค่าคงที่

ตัวอย่างเช่น $n^3+1000n^2+5000 \approx n^3$

Growth Function

Constant = a

Log ≈ log(n)

Linear ≈ n

Log-linear ≈ n log n

Quadratic ≈ n^2

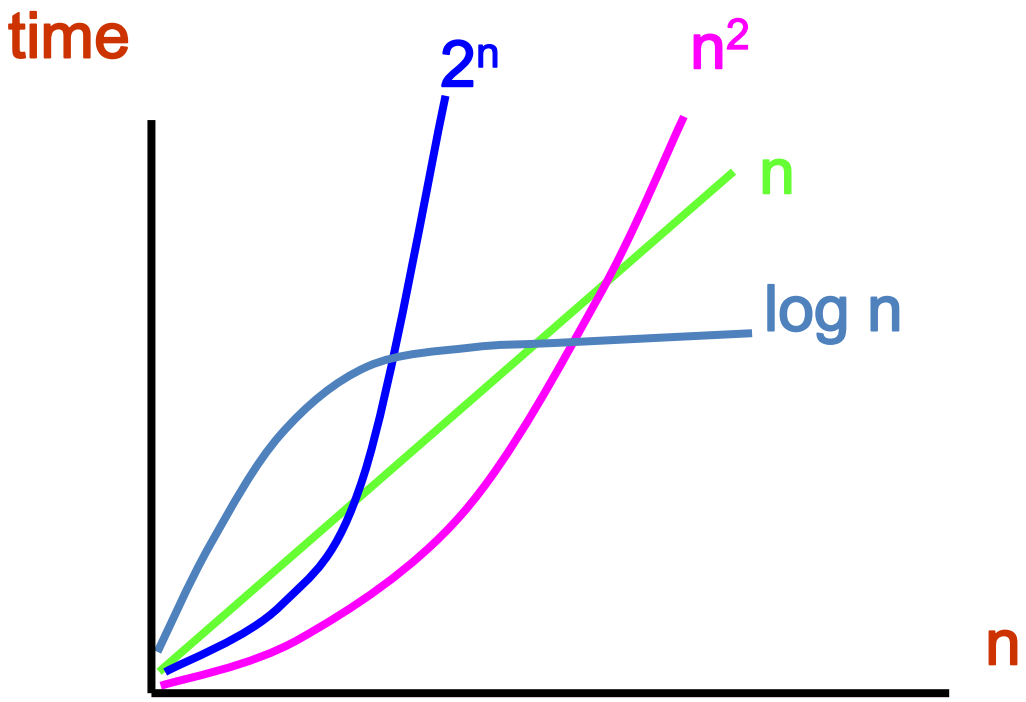
Cubic ≈ n^3

Polynomial ≈ n^k

Exponential ≈ a^n , 2^n ….

…

อัลกอริทึมต่างๆ จะมีฟังก์ชันที่ใช้แทนเวลาในการทำงานของอัลกอริทึมที่แตกต่างกัน



Asymptotic Notation

เราจะใช้สัญกรณ์เชิงเส้นกำกับ (Asymptotic notation) ในการวัดความซับซ้อนของอัลกอริทึม

$$\Omega \quad \Theta \quad O$$

Big-O

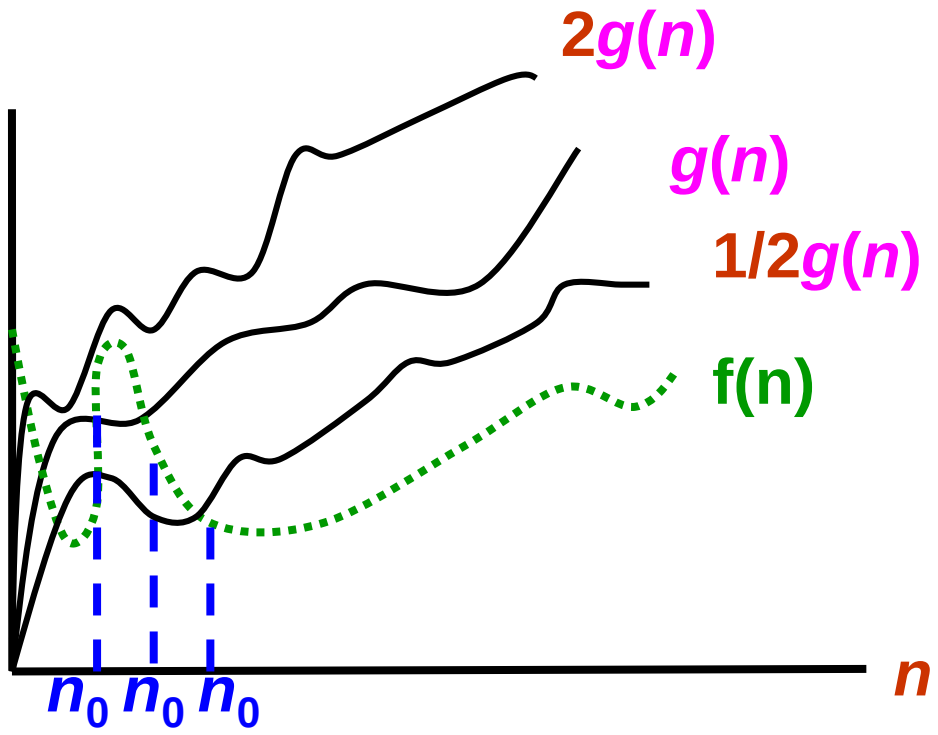
นิยาม

$$O(g(n)) = \{f(n): \exists c>0 \text{ and } n_0>0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n>n_0\}$$

หมายความว่า ฟังก์ชัน f(n) มีอัตราการเติบโตของฟังก์ชันน้อยกว่าหรือเท่ากับฟังก์ชัน g(n)

c Vs. n_0

f(n) = O(g(n))



แสดงว่า g(n) เป็นขอบเขตบน upper bound ของ f(n)

จากรูปพบว่าเราสามารถหาค่า c ได้หลายค่าซึ่งแต่ละค่าจะมีค่า n_0 ที่สอดคล้องที่ทำให้สมการเป็นจริง

$3n^2+2n+1 = O(n^2)$?

$3n^2+2n+1 = O(n^2)$? จงพิสูจน์

<u>วิธีทำ:</u> กำหนดให้ $f(n) = 3n^2+2n+1$, and $g(n) = n^2$

ต้องการพิสูจน์ว่า $f(n) = O(g(n))$

จากนิยามของ $f(n) = O(g(n))$ นั่นคือ ต้องการหาค่า $c > 0$ และ $n_0 > 0$ ที่ทำให้ $3n^2+2n+1 \leq cn^2$ สำหรับทุกๆ $n \geq n_0$

$$3n^2+2n+1 \leq cn^2$$

$$3+\frac{2}{n}+\frac{1}{n^2} \leq c$$

ถ้าเลือก $c = 6$ และ $n_0 = 1$ แล้ว $3n^2+2n+1 \leq cn^2$ สำหรับ $n \geq n_0$

ดังนั้น $3n^2+2n+1 = O(n^2)$

$2^{n+1} = O(2^n)$?

$2^{n+1} = O(2^n)$? จงพิสูจน์

<u>วิธีทำ</u>: กำหนดให้ $f(n) = 2^{n+1}$ และ $g(n) = 2^n$

ต้องการพิสูจน์ว่า $f(n) = O(g(n))$

จากนิยามของ $f(n) = O(g(n))$ นั่นคือต้องการหาค่า $c > 0$ และ $n_0 > 0$ ที่ทำให้ $2^{n+1} \leq c2^n$ สำหรับ $n \geq n_0$

$$2^{n+1} \leq c2^n$$

$$2^{n+1-n} \leq c$$

$$2 \leq c$$

ถ้าเลือก $c = 2$ และ $n_0 = 1$แล้ว $2^{n+1} \leq c2^n$ สำหรับทุกๆ $n \geq n_0$

ดังนั้น $2^{n+1} = O(2^n)$ เป็นจริง

$2^{2n} = O(2^n)$?

$2^{2n} = O(2^n)$? จงพิสูจน์

<u>วิธีทำ</u>: พิสูจน์โดยข้อขัดแย้ง สมมติว่า $2^{2n} = O(2^n)$ เป็นจริง

นั่นคือมีค่าคงที่ $c > 0$ ที่ทำให้ $2^{2n} \leq c2^n$

$2^{2n} \leq c2^n$

จาก $2^{2n} = 2^n * 2^n$ จะได้ว่า $2^n \leq c$

เนื่องจาก n มีค่าเพิ่มขึ้นเรื่อยๆ ทำให้ 2^n ไม่สามารถหาขอบเขตได้ ดังนั้นไม่มีค่าคงที่ c ที่มีคุณสมบัตินี้อยู่จริง

นั่นคือที่เราสมมติว่า $2^{2n} = O(2^n)$ <u>ไม่จริง</u>

Omega

นิยาม

$\Omega(g(n)) = \{f(n): \exists c>0$ and $n_0>0$ such that

$$0 \leq cg(n) \leq f(n) \text{ for all } n>n_0\}$$

“f(n) is asymptotically grows more than or equal to g(n)”

หมายความว่า ฟังก์ชัน f(n) มีอัตราการเติบโตของฟังก์ชันมากกว่าหรือเท่ากับฟังก์ชัน g(n)

Theta

นิยาม

$\Theta(g(n)) = \{f(n): \exists c_1>0, c_2>0 \text{ and } n_0>0 \text{ such that}$

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n>n_0\}$

"f(n) is asymptotically grows equally as g(n)"

หมายความว่า ฟังก์ชัน f(n) และ g(n) มีอัตราการเติบโตของฟังก์ชันเทียบเท่ากัน

$n^2/2-3n = \Theta(n^2)$

$n^2/2-3n = \Theta(n^2)$? จงพิสูจน์

<u>วิธีทำ</u>: กำหนดให้ $f(n) = n^2/2-3n$ และ $g(n) = n^2$

ต้องการพิสูจน์ว่า $f(n) = \Theta(g(n))$

จากนิยามของ $f(n) = \Theta(g(n))$ นั่นคือต้องการหาค่า $c_1, c_2 > 0$ และ $n_0 > 0$ ที่ทำให้ $c_1n^2 \leq n^2/2-3n \leq c_2n^2$ สำหรับ $n > n_0$

$c_1n^2 \leq n^2/2-3n$

$c_1 \leq 1/2-3/n$

นั่นคือ $n = 7, c_1 = 1/14$

$n^2/2-3n \leq c_2n^2$

$1/2-3/n \leq c_2$

นั่นคือ $n = 1/2, n \geq 1$

ถ้าเลือก $c_1 = 1/14, c_2 = 1/2$ และ $n_0 = 7$ แล้ว $c_1n^2 \leq n^2/2-3n \leq c_2n^2$ เป็นจริง

ดังนั้น $n^2/2-3n = \Theta(n^2)$

Exercise

- $f(n)= 3n^3/2$ $g(n)= 2n^2+120n+9$ จงพิสูจน์ว่า $f(n) \in O(g(n))$

- $A=\log_3(n^2)$,$B=\log_2(n^3)$ จงพิสูจน์ว่า $A \in \Theta(B)$

Exercise

- จงพิสูจน์ว่า $3n^2+n = \Omega(n^2)$

- จงพิสูจน์ว่า $100n+50 \neq \Omega(n^2)$

Analyzing the time complexity

เราสามารถจะจัดรูปฟังก์ชันในการวิเคราะห์

กำหนดให้ $f_1(n) = O(g_1(n))$ และ $f_2(n) = O(g_2(n))$

- $cf_1(n) = cO(g_1(n)) = O(g_1(n))$
- $f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$
- $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot (g_2(n))$
- $f_1(n)^k = O(g_1(n)^k)$
- $\sum_{k=1}^{n} O\big(f(k)\big) = O(\sum_{k=1}^{n} f(k))$

Analyzing the time complexity

```
Find_Max(A)
    max := A[1]                                  O(1)
    for i := 2 to n                              Σ_{k=2}^{n} O(1) = O(Σ_{k=2}^{n} 1) = O(n)
        if max < a[i] then max := A[i]           O(1)
    return(max)                                  O(1)
```

T(n) = O(max(1,n,1))=O(n)

จาก $f_1(n) + f_2(n) = O(max(g_1(n),g_2(n)))$

Analyzing the time complexity

Selection_Sort(A)

for i=n downto 2

j=1 O(1)

for (k = 2; k <= i; k++) $\sum_{k=2}^{i} O(1)$ =O(i)

if (A[j] < A[k]) j = k; O(1)

swap(A,i,j) O(1)

max(O(1),O(i),O(1))=O(i)

T(n) =$\sum_{i=2}^{n} O(i) = O(\sum_{i=2}^{n} i)$=O(n(n+1)/2) = O($n^2$)

Analyzing the time complexity

```
for i=1 to n
    A[i]=0
```

$\sum_{i=1}^{n} O(1) = O(\sum_{i=1}^{n} 1)$=O(n)

```
for i=1 to n
    for j=1 to n
        A[i]=A[i]+A[j]
```

$\sum_{j=1}^{n} O(1)$=O(n)

$\sum_{i=1}^{n} O(n) = O(\sum_{i=1}^{n} n)$=O($n^2$)

T(n) = max (O(n),O(n^2)) = O(n^2)

ตัวอย่างการลดรูป

$O(k) = O(1)$, $10^k = O(1)$ สำหรับค่าคงที่ k ใดๆ

$N^2/2 - 3N = O(N^2)$

$1 + 4N = O(N)$

$7N^2 + 10N + 3 = O(N^2)$

$\log_{10} N = \log_2 N / \log_2 10 = O(\log_2 N) = O(\log N)$

$\log N + N = O(N)$

$\sum_{i=1}^{N} O(i) = O(N^2)$

$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

$\sum_{i=1}^{N} O(i^2) = O(N^3)$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$