C/C++ 進階班
演算法

複雜度分析 (Complexity)

李耕銘

課程大綱

- 為什麼要評估複雜度？
- 如何評估複雜度？
- 複雜度的估計法？
- Big-O的運算與證明
- 極限的複習與證明
- 複雜度還有哪些估計符號？
- 遞迴的複雜度

Warning ：A little math in this chapter！

為什麼要評估複雜度？

為什麼要評估複雜度

- 電腦並非無所不能
- Why we care?
  - Computers may be fast, but they are still limited.
  - Memory may be cheap, but it is not free
- 常常我們無法得到最佳解，越有**效率**的算法越能帶我們找到較佳解

如何評估複雜度？

效能評估

- Criteria
  - 正確性
  - 可讀性
- Performance Analysis
  - 空間複雜度：記憶體/硬碟需求
  - 時間複雜度：運算時間
- Performance Measurement

Space Complexity

$$S(I) = C + S_P(I)$$

- $S(I)$：Total space required
- C ：Fixed Space Requirements
  - Independent of the inputs
  - Ex：constants、global variables
- $S_P(I)$：Variable Space Requirements
  - Dependent on the inputs
  - Ex：recursive stack space、local variables

Space Complexity

$$S(I) = C + S_P(I)$$

```
int Fibonacci (int n)
{
    if (n<=2)
        return 1;
    else
        return Fibonacci(n-1)+Fibonacci(n-2);
}
```

$$S_P(I) \propto 2^n$$

$$\begin{aligned} S(I) &= C + S_P(I) \\ &= C + k2^n \end{aligned}$$

Time Complexity

$$T(I) = C + T_P(I)$$

- $T(I)$：Total time required
  - compile time and run time
- C ：Fixed time Requirements
  - Independent of the inputs
- $T_P(I)$ ：Variable time Requirements
  - Dependent on the inputs

Time Complexity

$$T(I) = C + T_P(I)$$

```
int sum = 0;
for(int i=1;i<=N;i++)
    sum += i;
```

$$T_P(I) \neq 0$$
$$T_P(I) \propto \mathrm{N}$$
$$T(I) = C + T_P(I)$$
$$= C + kN$$

```
int sum = N*(N+1)/2;
```

$$T_P(I) = 0$$
$$T(I) = C$$

Time Complexity

$$T(I) = C + T_P(I)$$

```
int Fibonacci (int n)
{
    if (n<=2)
        return 1;
    else
        return Fibonacci(n-1)+Fibonacci(n-2);
}
```

$$T_P(I) \propto 2^n$$

$$T(I) = C + T_P(I)$$
$$= C + k2^n$$

複雜度的估計法？

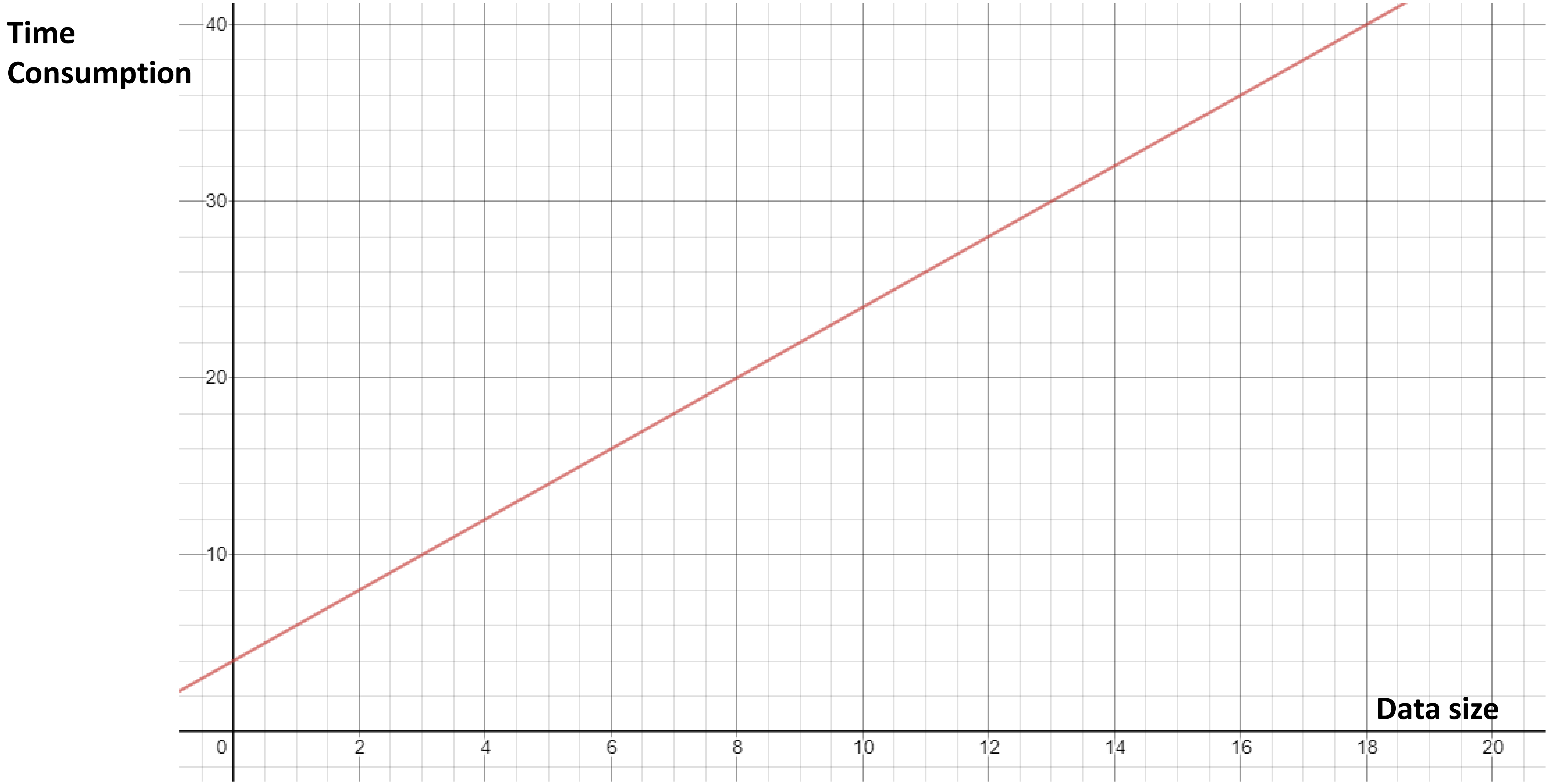
複雜度的估計法？

- 複雜度的概念─假設所有的運算都花費一樣的時間
  - 加減乘除取餘數
  - 位運算、存取記憶體
  - 判斷、邏輯運算子
  - 賦值運算子
- 把所有運算需要的「次數」都計算出來
  - 看數量級的大小 (複雜度)，評估執行需要的時間

Step count table

Code	Steps	Frequency	Sum of steps
`int sum(int *p, int len)`	0	1	0
`{`	0	1	0
`    int sum = 0;`	1	1	1
`    if(len > 0){`	1	1	1
`        for(int i=0;i<len;i++)`	1	len+1	len+1
`            sum += *(p+i);`	1	len	len
`    }`	0	1	0
`    return sum;`	1	1	1
`}`	0	1	0
Total steps		2len+4	

Step count table

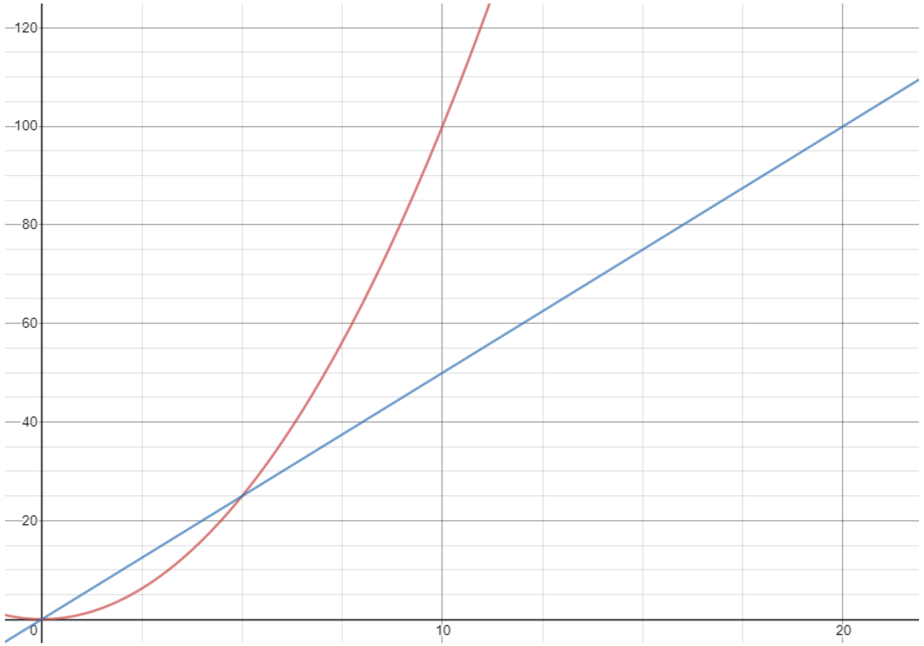


複雜度的估計法？

- 資料小跟資料大的處理速度可能會不一樣！

```
int sum = 0;
for(int i=1;i<=N;i++)
    sum += i;
```

```
int sum = N*(N+1)/2;
```

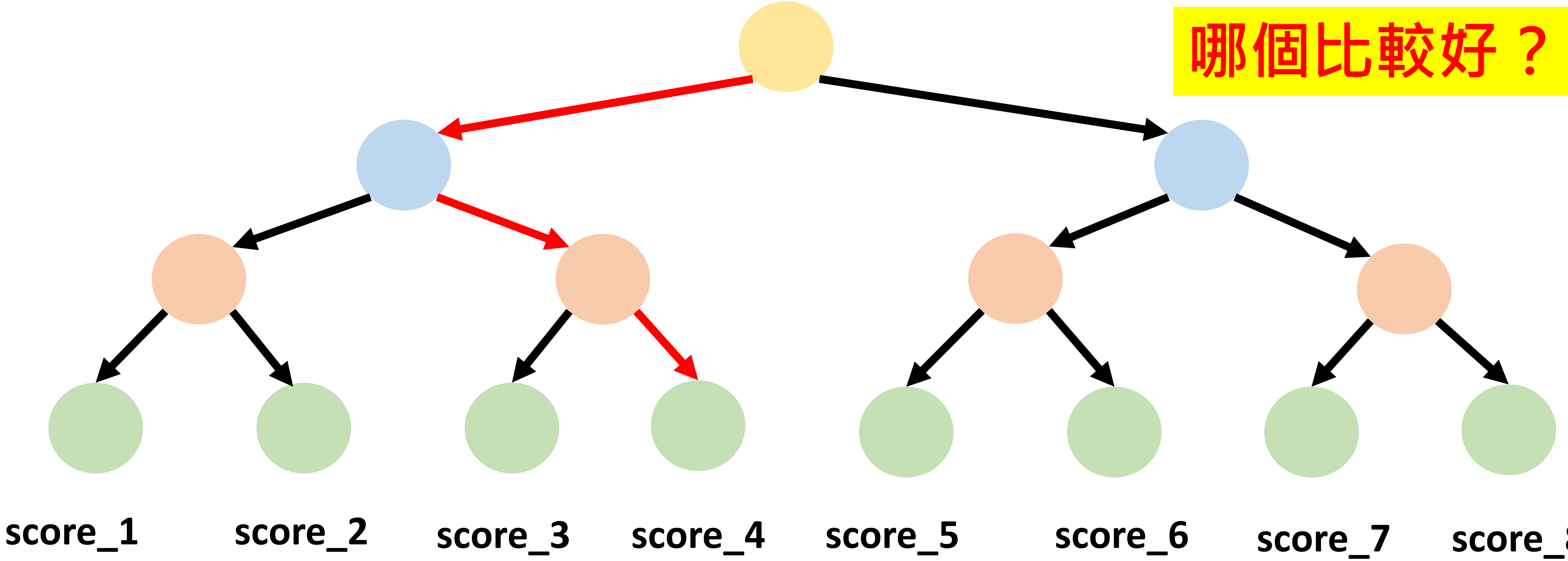


通常我們在意**資料量大**的時候

複雜度的估計法？

score_1 score_2 score_3 score_4 score_5 score_6 score_7 score_8

哪個比較好？



複雜度的估計法？

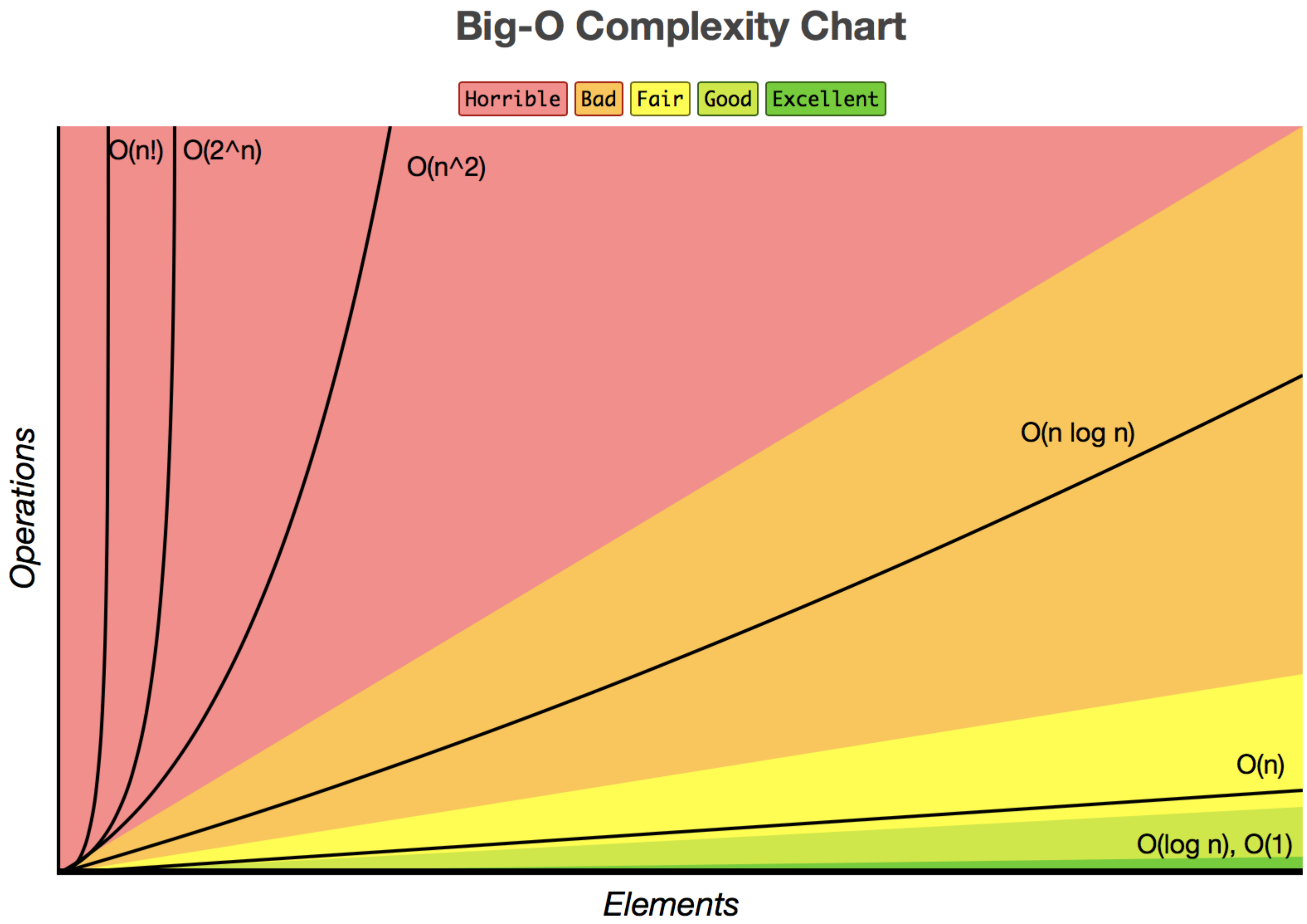
- 究竟是哪個好？
  - 時間花的越少 (CPU運行速度越快) ？
  - 記憶體空間花的越少？
  - 精準度或正確率較高？
  - 開發成本低、速度快 (現在越來越重要了)
- 平衡是很重要的事情
  - 抽樣代替普查
  - 通常 CPU 的運算資源比較珍貴→看時間複雜度
- 時間換取空間、空間換取時間

複雜度的估計法？

Time consumption(ms)

Data size	1	N	N^2	N^3	2^N
1	1	1	1	1	1
10	1	10	10^2	10^3	1024~10^3
100	1	10^2	10^4	10^6	~10^{30}
1000	1	10^3	10^6 (~15分鐘)	10^9 (~12天)	~10^{300}
10000	1	10^4	10^8 (~25小時)	10^{12} (~30年)	~10^{3000}

複雜度的估計法？



Big-O Complexity Chart
Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

Example

計算下列程式碼的**迴圈執行次數**

```
for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)
        for(int k=0;k<M;k++)
            cout << i*j*k << " ";
```

Example

計算下列程式碼的迴圈執行次數

```
for(int i=1;i<N;i++)
    for(int j=1;j<N;j+=2)
        cout << i << " " << j;
```

$$1 + 2(n - 1) < N$$

$$n < \frac{N + 1}{2}$$

Example

計算下列程式碼的**迴圈執行次數**

```
for (i = M; i > 1; i/=2) {
    cout << i << endl;
}
```

$$\frac{M}{2^{n-1}} > 1;\ \ M > 2^{n-1}$$

$$n < \log_2 M + 1$$

Practice

計算下列程式碼的時間與空間用量

```
for (j = 0; j < M; j++) {
    cout << j << endl;
}

for (i = 0; i < N; i++) {
    cout << i << endl;
}
```

Time: $M + N$

Practice

計算下列程式碼的**迴圈執行次數**

```
for(int j=1;j<N;j*=2)
          cout << j;
```

$$2^{n-1} < N$$

$$n < \log_2 N + 1$$

Practice

計算下列程式碼的迴圈執行次數

```
for(int i=1;i<N;i++)
    for(int j=1;j<N;j*=2)
        cout << i << " " << j;
```

$$n < (N-1)(\log_2 N + 1)$$

Big-O的運算與證明

複雜度的估計法？

演算法「工作效率」之函數，就稱為Complexity(複雜度)

- 演算法會依資料散布情形不同而有不同的處理次數
- Worst-case
  - upper bound
- Average-case
- Best-case
  - lower bound

複雜度的估計法

Search

score_1 | score_2 | score_3 | score_4 | score_5 | score_6 | score_7 | score_8

- Worst-case：Len
  - upper bound
- Average-case：(Len+1)/2
- Best-case：1
  - lower bound

Upper bound：Len

Lower bound：1

複雜度的估計法？

- 通常我們會想知道在最壞情形下會跑多久
  - Worst-case
- 或是資料量增加時，所需時間的成長幅度
  - Growth rate
- 不受單位影響
- 不在乎小資料時的狀況

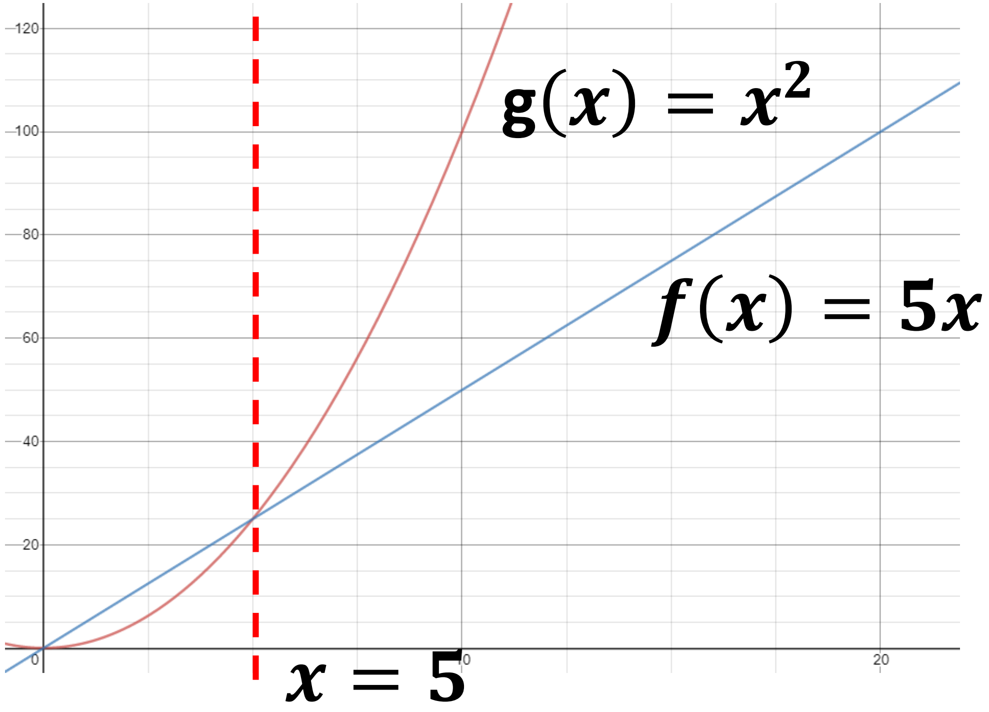
$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, f(n) \leq cg(n)$$

存在 c >0、$\boldsymbol{n_0}$，使得所有的 $\boldsymbol{n > n_0}$ 滿足$\boldsymbol{f(n) \leq cg(n)}$

複雜度的估計法？

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, f(n) \le cg(n)$$

<u>存在 c >0、n_0，使得所有的$n > n_0$滿足$f(n) \le cg(n)$</u>



$$5x \le cx^2$$

$$n_0 = 5$$

$$c = 1$$

$$\forall x > 5, f(x) \le g(x)$$

$$f(x) \in O\big(g(x)\big)$$

$$5x \in O(x^2)$$

Example

證明或否證：$5x \in O(x)$

$$n_0 = 0$$

$$c = 6$$

$$\forall x > 0, f(x) \leq cg(x)$$

$$\forall x > 0, 5x \leq 6x$$

$$\boldsymbol{5x \in O(x)}$$

Example

證明或否證：$5x^2 \in O(x)$

$$5x^2 \le cx \to 5x \le c \to x \le \frac{c}{5}$$

$$no\ matter\ what\ the\ value\ is\ \mathrm{c}$$

$$if\ x > \frac{c}{5},$$

$$5x^2 \le cx\ \ could\ not\ be\ satisfied$$

so,

$$\boldsymbol{5x^2 \notin O(x)}$$

Example

證明或否證：$100x^2 \in O(x^3 - x^2)$

$$100x^2 \le c(x^3 - x^2)$$
$$c = 100$$
$$100x^2 \le 100(x^3 - x^2)$$
$$200x^2 \le 100x^3$$
$$2 \le x$$

$$x_0 = 2$$
$$c = 100$$
$$\forall x > 2, 100x^2 \le 100(x^3 - x^2)$$

$$\boldsymbol{100x^2 \in O\left(x^3 - x^2\right)}$$

Practice

請試著：

(1)**證明或否證** ：$3x^2 \in O(x^2)$

(2)**證明或否證** ：$x \in O(\sqrt{x})$

$$3x^2 \le cx^2$$
$$let\ c = 1,$$
$$n_0 = 0$$

$$x \le c\sqrt{x}$$
$$x^2 \le cx$$
$$x \le c$$
$$no\ matter\ what\ c\ is,$$
$$if\ n > c, then\ x > c\sqrt{x}$$

Another view

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, f(n) \leq cg(n)$$

divided by g(n)

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, \frac{f(n)}{g(n)} \leq c$$

equals to ：

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$$

極限的複習與證明

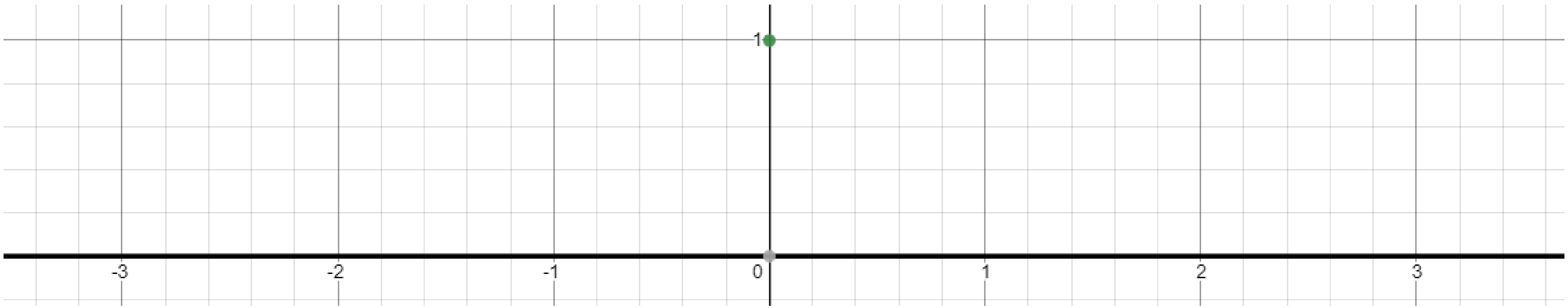
極限的複習與證明

$f(x)$在 x逼近 a時的極限為 L：

$$\lim_{x \to a} f(x) = L$$

$$f(x) = \begin{cases} 0, & \forall x \neq 0 \\ 1, & if\ x = 0 \end{cases}$$

$$f(0) = 1,$$
$$but$$
$$\lim_{x \to 0} f(x) = 0$$



極限的複習與證明

$$Given\ \lim_{x\to\infty} f(x) = L\,,\ \lim_{x\to\infty} g(x) = K\,,\ \lim_{x\to\infty} h(x) = \infty$$

1. $\lim_{x\to\infty}(f(x) + g(x)) = L + K$
2. $\lim_{x\to\infty}(f(x) - g(x)) = L - K$
3. $\lim_{x\to\infty}(f(x) \times g(x)) = L \times K$
4. $\lim_{x\to\infty}(f(x) \div g(x)) = L \div K$
5. $\boldsymbol{\lim_{x\to\infty}\left(\frac{1}{h(x)}\right) = 0}$

Example

$$\boldsymbol{Please\ calculate}:\ \lim_{n\to\infty}\left(\frac{\mathbf{3}\boldsymbol{n}+\mathbf{2}}{\mathbf{2}\boldsymbol{n}+\mathbf{5}}\right)$$

$$\lim_{n\to\infty}\left(\frac{\mathbf{3}\boldsymbol{n}+\mathbf{2}}{\mathbf{2}\boldsymbol{n}+\mathbf{5}}\right)=\lim_{n\to\infty}\left(\frac{\frac{\mathbf{3}\boldsymbol{n}}{\boldsymbol{n}}+\frac{\mathbf{2}}{\boldsymbol{n}}}{\frac{\mathbf{2}\boldsymbol{n}}{\boldsymbol{n}}+\frac{\mathbf{5}}{\boldsymbol{n}}}\right)=\lim_{n\to\infty}\left(\frac{\mathbf{3}+\mathbf{0}}{\mathbf{2}+\mathbf{0}}\right)=\frac{\mathbf{3}}{\mathbf{2}}$$

極限的複習與證明

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, f(n) \leq cg(n)$$

divided by g(n)

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0, \frac{f(n)}{g(n)} \leq c$$

equals to ：

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$$

極限的複習與證明

$$f(n) \in O\big(g(n)\big) \Leftrightarrow \exists c > 0, \lim_{n\to\infty} \frac{f(n)}{g(n)} \le c$$

只要$\frac{f(n)}{g(n)}$的極限值不是無限大，則$f(n) \in O\big(g(n)\big)$

Example

$\underline{Prove\ or\ disapprove：5x \in O(x)}$

$$\exists c > 0, \lim_{x \to \infty} \frac{5x}{x} \leq c$$

$\underline{Prove\ or\ disapprove：100x^2 \in O(x^3 - x^2)}$

$$\exists c > 0, \lim_{x \to \infty} \frac{100x^2}{x^3 - x^2} \leq c$$

李耕銘

Example

證明或否證：$5x^2 \in O(x)$

$$\lim_{x \to \infty} \frac{5x^2}{x} = \infty$$

$$\boldsymbol{5x^2 \notin O(x)}$$

Practice

請試著：

(1)證明或否證：$3x^3 + 5x^2 + 2x + 6 \in O(x^3)$

(2)證明或否證：$f(x) \in O(x^2) \Leftrightarrow f(x) \in O(x^2 + x)$

$$\lim_{x\to\infty} \frac{3x^3 + 5x^2 + 2x + 6}{x^3} = 3$$

$$\lim_{x\to\infty} \frac{f(x)}{x^2} = c, f(x) = cx^2$$

$$\lim_{x\to\infty} \frac{f(x)}{x^2 + x} = \frac{cx^2}{x^2 + x} = c$$

$$\lim_{x\to\infty} \frac{f(x)}{x^2 + x} = c, f(x) = c(x^2 + x)$$

$$\lim_{x\to\infty} \frac{f(x)}{x^2} = \frac{c(x^2 + x)}{x^2} = c$$

複雜度的符號

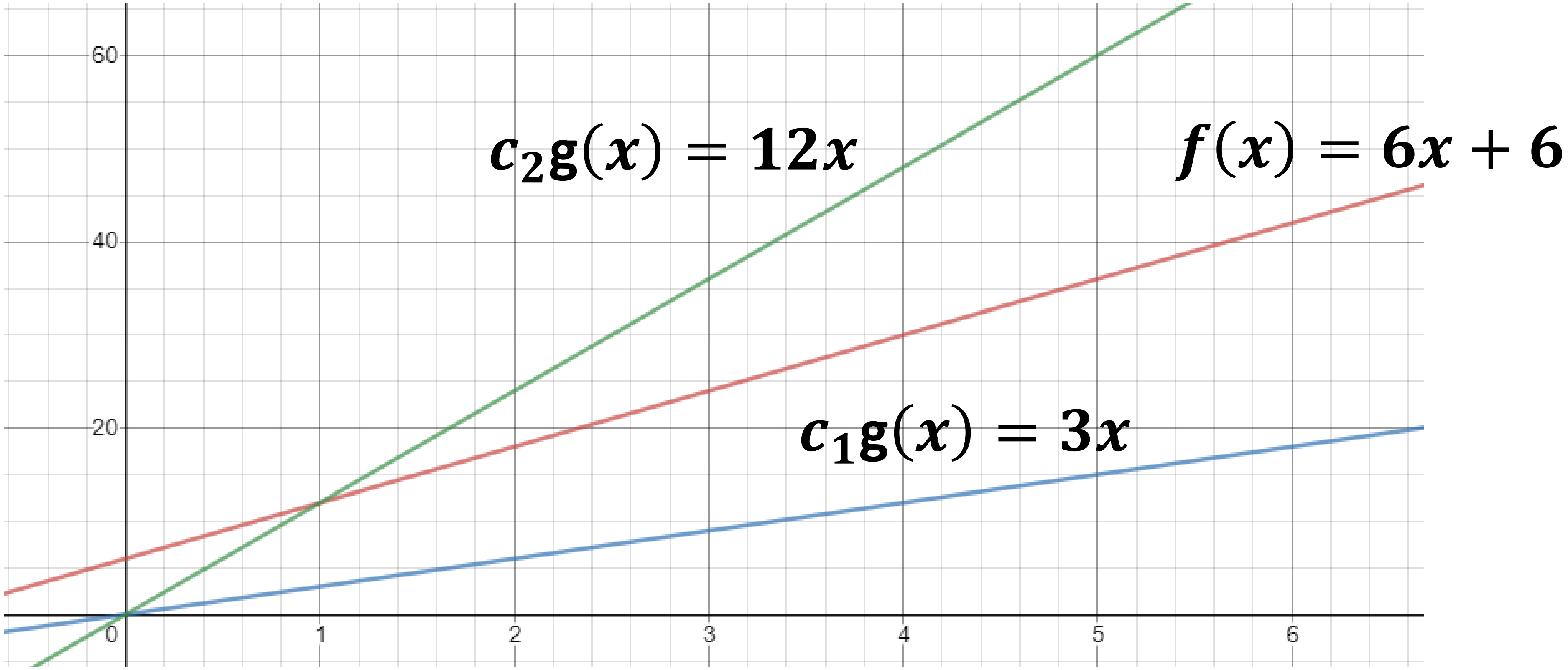
Another view

What'S wrong with Big-O？

$$\underline{\boldsymbol{f(x) \in O(x^2)}}$$
$$f(x) \in O(x^2 + x)$$
$$f(x) \in O(3x^2 + 2x)$$
$$f(x) \in O(x^3 + 1)$$
$$f(x) \in O(2x^4 + x)$$
$$f(x) \in O(x^5 + x^2 + 2)$$
…

Big-Theta : Θ

$$f(n) \in \Theta\big(g(n)\big) \Leftrightarrow \exists c_1 、c_2 > 0, \exists n_0, \forall n > n_0$$
$$s.t.\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n)$$

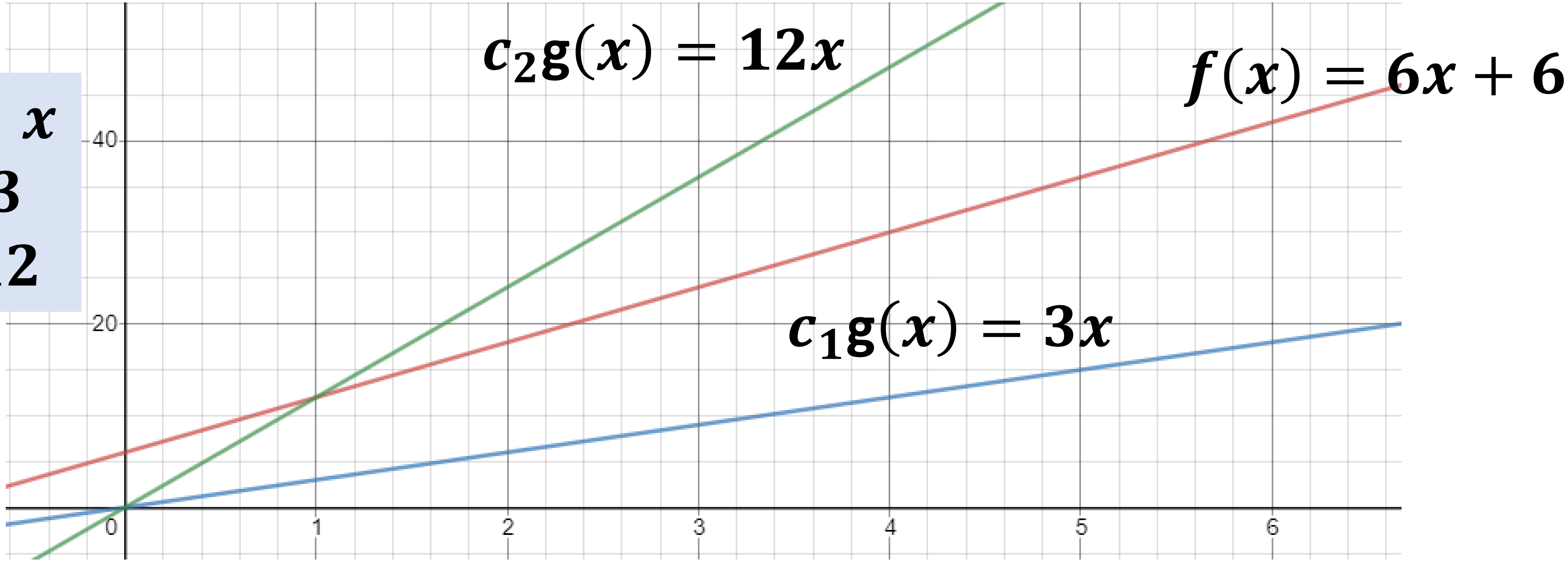


Big-Theta : Θ

$$f(n) \in \Theta(g(n)) \Leftrightarrow \exists c_1、c_2 > 0, \exists n_0, \forall n > n_0$$
$$s.t.\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n)$$

$$\boldsymbol{6x + 6 \in \Theta(x)}$$

$$\boldsymbol{g(x) = x}$$
$$\boldsymbol{c_1 = 3}$$
$$\boldsymbol{c_2 = 12}$$



Big-Theta：Θ

What happened to Big-O？

$$\underline{\boldsymbol{6x+6 \in O\left(x^2\right)}}$$
$$6x+6 \in O(x^2+x)$$
$$6x+6 \in O(3x^2+2x)$$
$$6x+6 \in O(x^3+1)$$
$$6x+6 \in O(2x^4+x)$$
$$6x+6 \in O(x^5+x^2+2)$$
$$\ldots$$

$$\boldsymbol{6x+6 \in \Theta(x)}$$
$$6x+6 \in \Theta(2x)$$
$$\cancel{6x+6 \notin \Theta(3x^2+2x)}$$
$$\cancel{6x+6 \notin \Theta(x^3+1)}$$
$$\cancel{6x+6 \notin \Theta(2x^4+x)}$$
$$\cancel{6x+6 \notin \Theta(x^5+x^2+2)}$$
$$\ldots$$

Big-Theta : Θ

$$f(n) \in \Theta\big(g(n)\big) \Leftrightarrow \exists c_1、c_2 > 0, \exists n_0, \forall n > n_0$$
$$s.t.\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n)$$
$$\textit{divided by } g(n)$$

$$\boldsymbol{0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \le c}$$

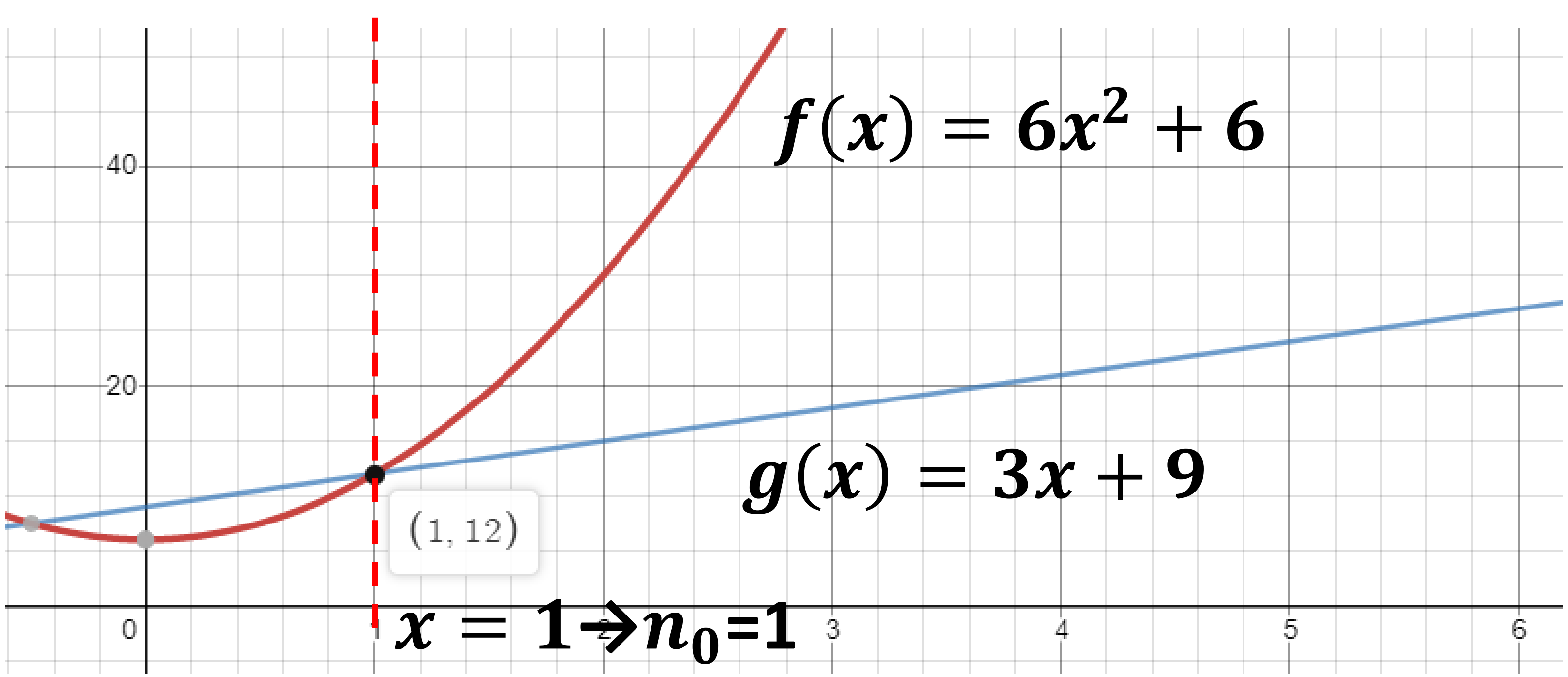
只要$\frac{f(n)}{g(n)}$的極限值不是無限大而且大於0，則$\boldsymbol{f(n) \in \Theta\big(g(n)\big)}$

※$\boldsymbol{g(n)}$只跟$\boldsymbol{f(n)}$的最大次方有關

Big-Omega : Ω

$$f(n) \in \Omega(g(n)) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0,$$
$$s.t.\ 0 \le cg(n) \le f(n)$$

$$\mathbf{6x^2 + 6 \in \Omega(3x + 9)}$$



Big-Omega : $\boldsymbol{\Omega}$

$$f(n) \in \Omega\big(g(n)\big) \Leftrightarrow \exists c > 0, \exists n_0, \forall n > n_0,$$
$$s.t.\ 0 \le cg(n) \le f(n)$$

$$\textit{divided by } g(n)$$

$$\boldsymbol{c \le \lim_{n \to \infty} \frac{f(n)}{g(n)}}$$

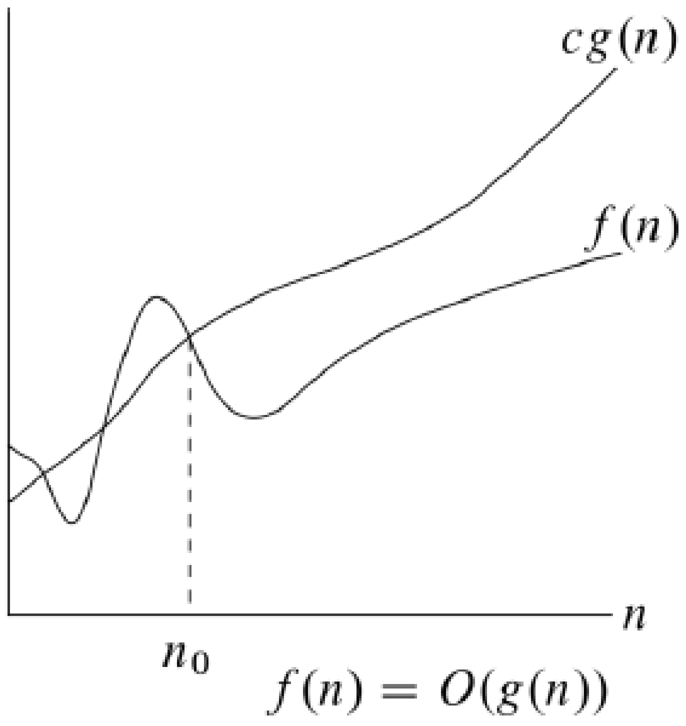
只要$\boldsymbol{\frac{f(n)}{g(n)}}$的極限值大於0，則$\boldsymbol{f(n) \in \Omega\big(g(n)\big)}$

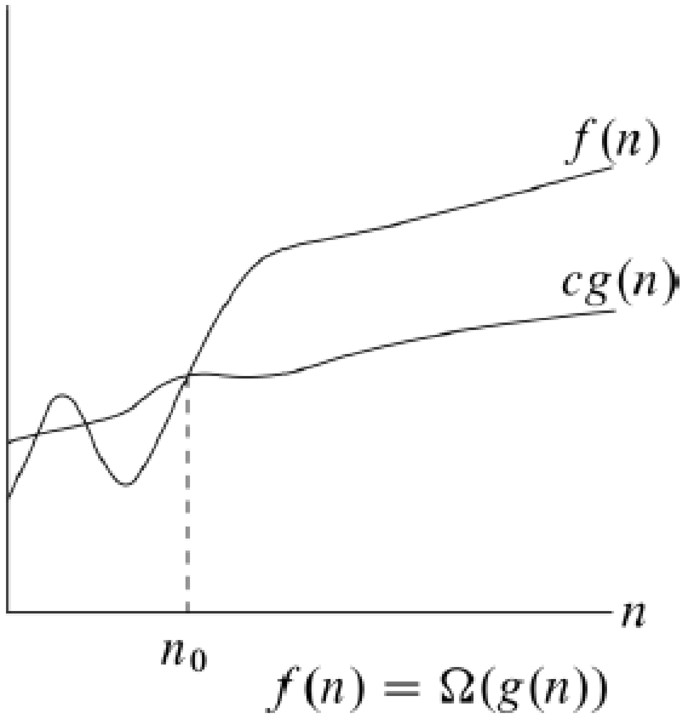
比較

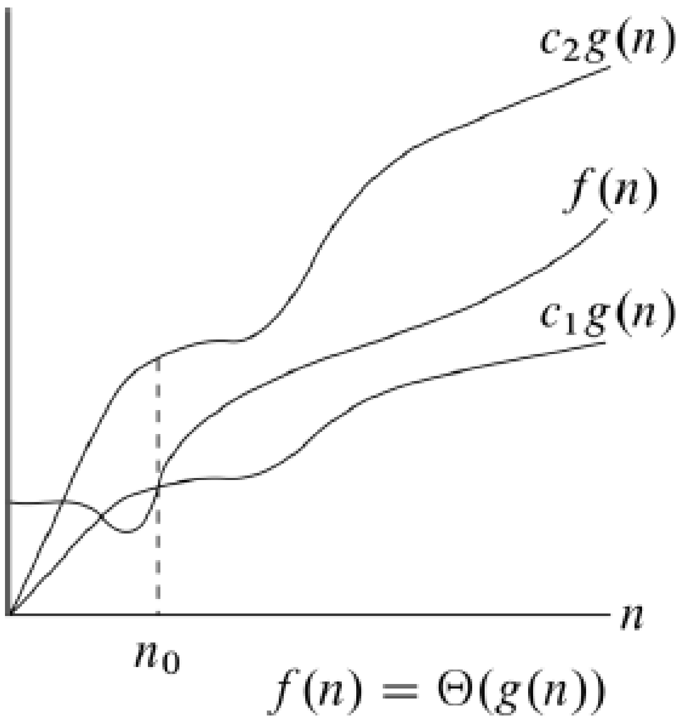
$O(g(n))$：上界

$\Omega(g(n))$：下界

$\Theta(g(n))$：上界+下界

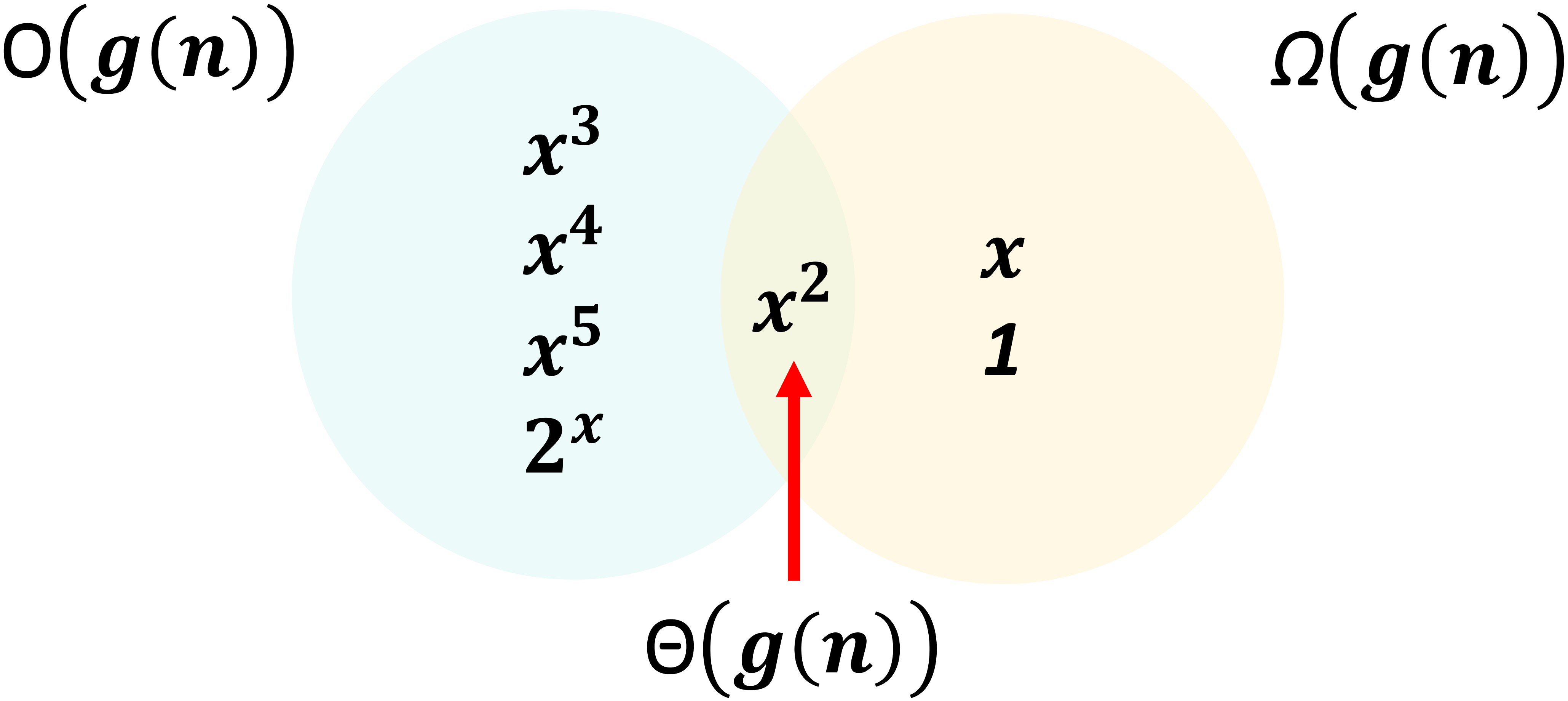






比較

$$f(x) = 3x^2 + x + 5$$



遞迴的複雜度

遞迴的複雜度

1. 數學解法 Mathematics-based Method)
   - 直接以遞迴的觀念算出複雜度
2. 代換法 (Substitution Method)
   - 猜一個數字後帶入
3. 遞迴樹法 (Recurrence Tree Method)
   - 畫出遞迴樹後加總之

Example

計算下列程式碼的時間複雜度

$$T(n) = \begin{cases} T(n-1) + 3, if\ n > 1 \\ 1, otherwise \end{cases}$$

Example

- 數學解法 Mathematics-based Method)
  - 直接以遞迴的觀念算出複雜度

$$T(n) = \begin{cases} T(n-1)+3, if\ n>1 \\ 1, otherwise \end{cases}$$

$$\begin{aligned} &T(n) \\ &= T(n-1)+3 \\ &= T(n-2)+3+3 \\ &= T(n-3)+3+3+3 \\ &= \ldots\ldots \\ &= T(1)+3(n-1) \\ &= 3n-2 \\ &\underline{T(n) \in O(n)} \end{aligned}$$

Example

- 代換法 (Substitution Method)
  - 猜一個數字後帶入

$$T(n) = \begin{cases} T(n-1)+3, if\ n>1 \\ 1, otherwise \end{cases}$$

猜 $T(n) \in O(n)$

$T(n) \le c(n-1)+3$

$T(n) \le cn - c + 3$

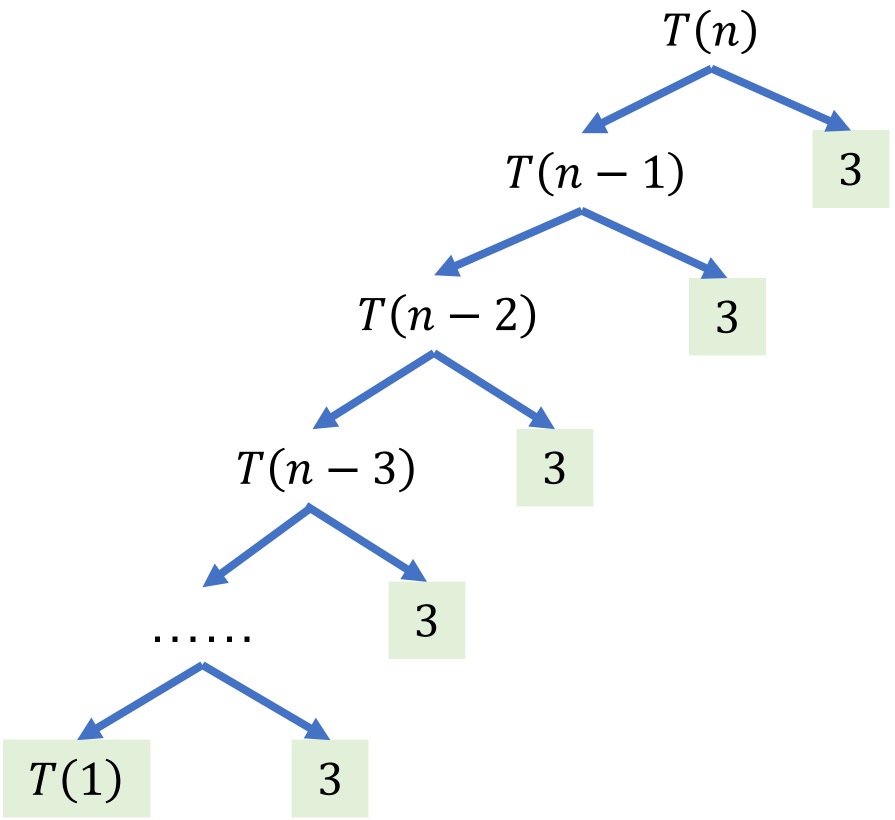
$\underline{T(n) \in O(n)}$

Example

- 遞迴樹法 (Recurrence Tree Method)
  - ➢ 畫出遞迴樹後加總之

$$T(n) = \begin{cases} T(n-1) + 3, if\ n > 1 \\ 1, otherwise \end{cases}$$

$T(n)$
$= T(1) + 3 \times (n-1)$
$= 3n - 2$
$\underline{T(n) \in O(n)}$



Practice

計算下列程式碼的時間複雜度

$$T(n) = \begin{cases} 2T(n-1), if\ n > 0 \\ 1, otherwise \end{cases}$$

Practice

$$T(n) = \begin{cases} 2T(n-1), if\ n > 0 \\ 1, otherwise \end{cases}$$

$$\begin{aligned} &T(n) \\ &= 2T(n-1) \\ &= 2^2T(n-2) \\ &= 2^3T(n-3) \\ &\dots \\ &= 2^nT(0) \\ &= 2^n \in O(2^n) \end{aligned}$$