



# 演算法簡介

## 這堂課預計上課重點

- 課程配分與進行方式
- 上課教材
- 何謂演算法
- 演算法的評估方式

# 課程配分與進行方式

# 課程配分與進行方式

## □ 學期成績計算

□ 期中考考試：50% (筆試)

□ 期末考考試：50% (筆試)

□ 課堂練習：視情況調整，最高50%

## □ 課程進行方式

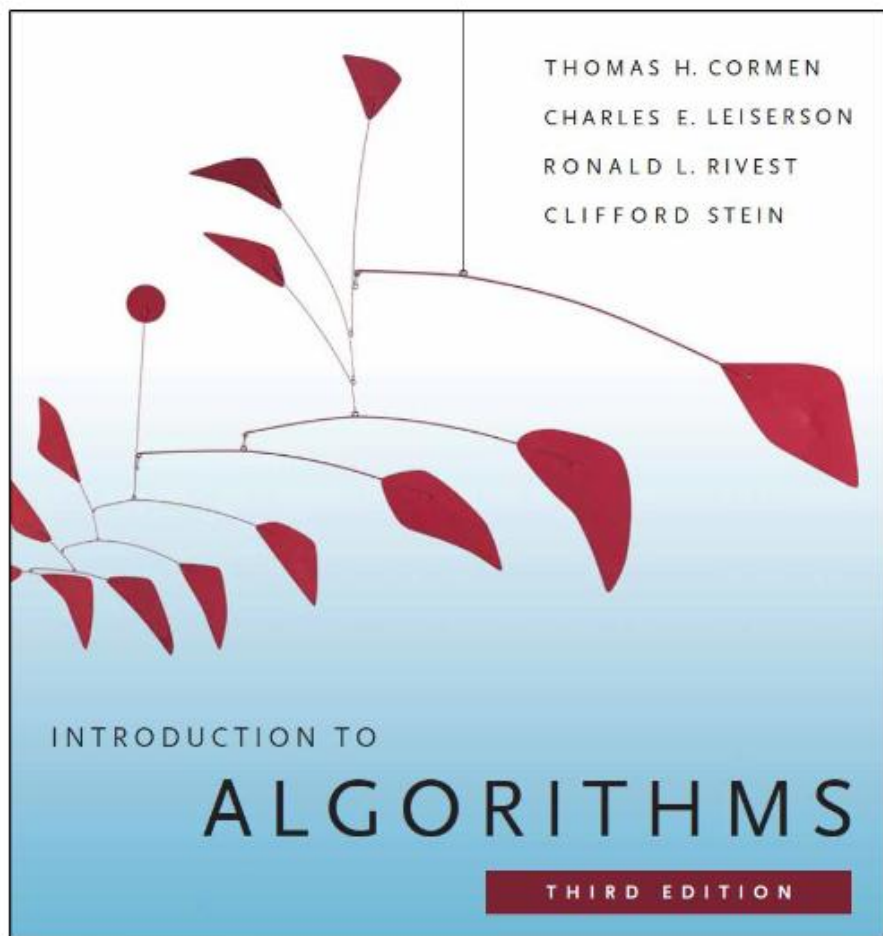
□ 課程簡報

□ 上課內容練習

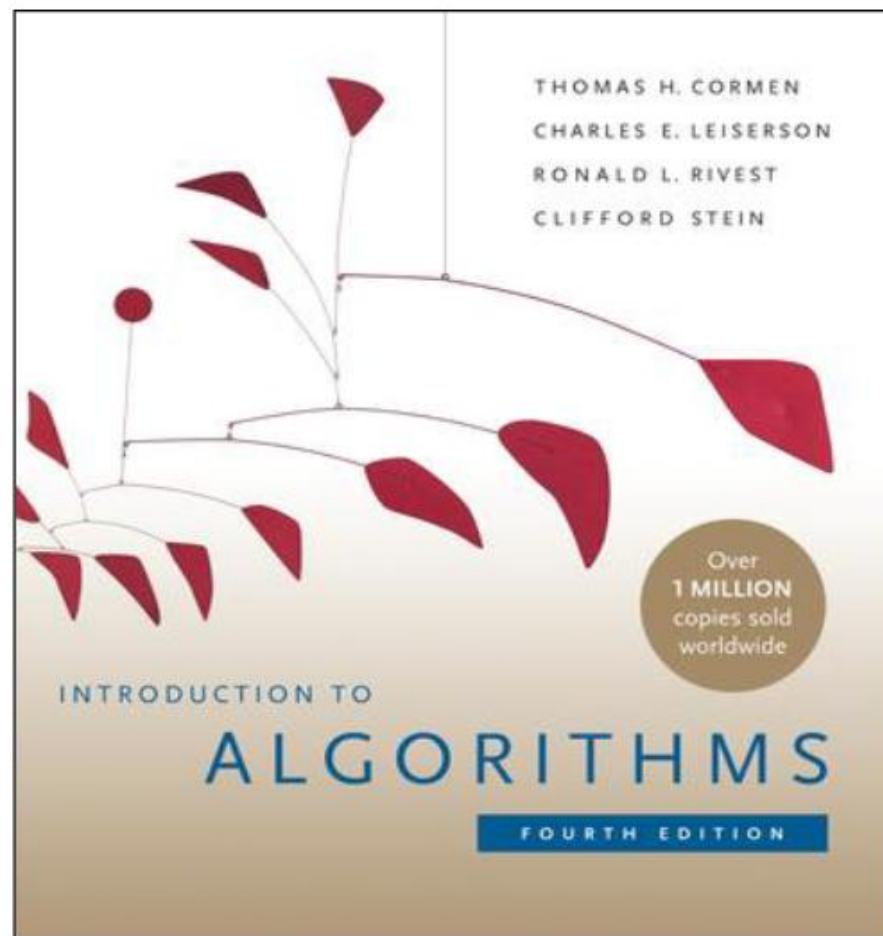
□ 虛擬碼閱讀

□ 虛擬碼轉成實際程式 (C語言) ———→ 視情況而定

# 上課教科書



<https://www.tenlong.com.tw/products/9780262033848>



<https://www.tenlong.com.tw/products/9780262046305>

# 演算法定義

- 一組明確、有限的步驟或規則，用來解決特定問題或完成某項任務
- 特性
  - 有限性 (Finiteness)：步驟數量是有限的，不能無限進行。
  - 明確性 (Definiteness)：每一步都有清楚的定義，不會有歧義。
  - 輸入 (Input)：演算法可以有零個或多個輸入。
  - 輸出 (Output)：至少會產生一個結果。
  - 有效性 (Effectiveness)：每一步都是可行的，能在有限時間內完成。

# 演算法的評估方式

## □ 演算法評估方式

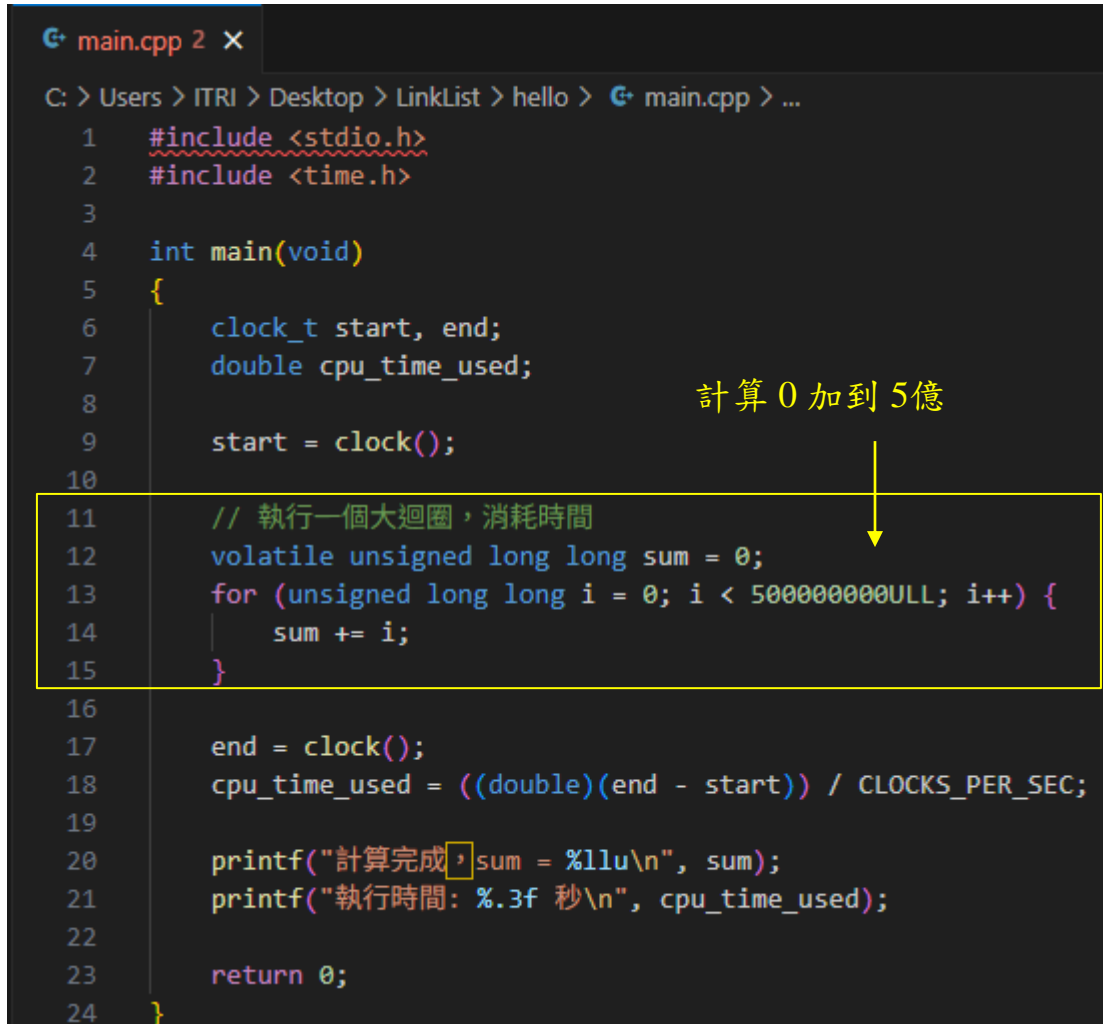
□ 佔用空間

□ 執行時間

} 好的演算法: 佔用空間愈少愈好，執行時間愈快愈好

# 演算法的評估方式 (執行時間)

□ 這個程式執行3次，印出的執行時間會相同嗎？



```
main.cpp 2 x
C: > Users > ITRI > Desktop > LinkList > hello > main.cpp > ...

1  #include <stdio.h>
2  #include <time.h>
3
4  int main(void)
5  {
6      clock_t start, end;
7      double cpu_time_used;
8
9      start = clock();
10
11     // 執行一個大迴圈，消耗時間
12     volatile unsigned long long sum = 0;
13     for (unsigned long long i = 0; i < 5000000000ULL; i++) {
14         sum += i;
15     }
16
17     end = clock();
18     cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
19
20     printf("計算完成, sum = %llu\n", sum);
21     printf("執行時間: %.3f 秒\n", cpu_time_used);
22
23     return 0;
24 }
```

計算 0 加到 5億



# 演算法的評估方式 (執行時間)

□ 同 1 台電腦執行結果 → 為何會有此現象 ??

```
C:\Users\ITRI\Desktop\LinkList x + v - □ x
計算完成，sum = 124999999750000000
執行時間：0.264 秒

Process returned 0 (0x0)   execution time : 0.282 s
Press any key to continue.
```

第1次執行

```
C:\Users\ITRI\Desktop\LinkList x + v - □ x
計算完成，sum = 124999999750000000
執行時間：0.246 秒

Process returned 0 (0x0)   execution time : 0.274 s
Press any key to continue.
```

第2次執行

```
C:\Users\ITRI\Desktop\LinkList x + v - □ x
計算完成，sum = 124999999750000000
執行時間：0.225 秒

Process returned 0 (0x0)   execution time : 0.954 s
Press any key to continue.
```

第3次執行

# 演算法的評估方式 (執行時間)

```
main.cpp 2 X
C: > Users > ITRI > Desktop > LinkList > hello > main.cpp > ...

1  #include <stdio.h>
2  #include <time.h>
3
4  int main(void)
5  {
6      clock_t start, end;
7      double cpu_time_used;
8
9      start = clock();
10
11     // 執行一個大迴圈，消耗時間
12     volatile unsigned long long sum = 0;
13     for (unsigned long long i = 0; i < 500000000ULL; i++) {
14         sum += i;
15     }
16
17     end = clock();
18     cpu_time_used = ((double)(end - start)) / CLOCKS_PER_SEC;
19
20     printf("計算完成, sum = %llu\n", sum);
21     printf("執行時間: %.3f 秒\n", cpu_time_used);
22
23     return 0;
24 }
```

執行(5億+1)次判斷

計算 0 加到 5 億

執行(5億)次加法

1 次變數Assign值  
+ (5億+1)\*(每次判斷大小需要時間)  
+ (5億+1)\*(每次加法需要時間)  
+ (5億)\*(每次加法需要時間)  
+ (5億)\*(每次變數Assign值的時間)

# 演算法的評估方式

## □ 佔用空間 → 空間複雜度

□ 衡量演算法在執行過程中需要的額外記憶體。

□ 包含：

□ 程式碼空間（程式本身需要的大小）

□ 輸入空間（輸入資料所需的大小）。

□ 輔助空間（演算法額外需要的暫存變數、堆疊、遞迴深度等）。

## □ 執行時間 → 時間複雜度

□ 衡量演算法執行所需的時間 → 漸近分析 (Asymptotic Analysis)

□ 表示法

□  $O$  (大寫) 表示法 (Big-O Notation) → 最差情況 (Upper Bound)

□  $\Omega$  表示法 (Big-Omega Notation) → 最佳情況 (Lower Bound)

□  $\Theta$  表示法 (Theta Notation) → 平均情況 (Tightly Bound)