演算法 簡介

- 教授老師: 梁奕智
- 研究室 TC703B
- 教學網頁 http://210.70.110.125/~ezliang
- Email: ezliang@mail.dwu.edu.tw

- 課目概要

  了解演算法的基本原理, 設計策略與分析方法.

- 教學目標

  了解演算法, 能夠實作程式來印證.

- 成績評定

  1) 期中考35% 2) 期末考35% 3) 平常成績30%

- 課堂要求

  期中考/期末考均必須出席.

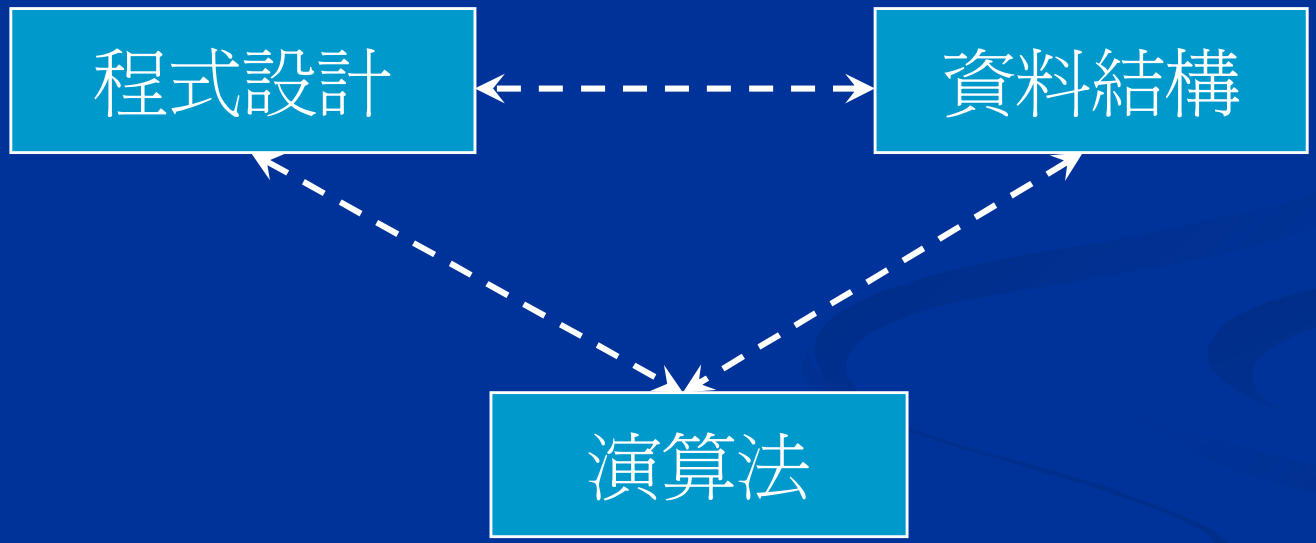
  課後備有作業, 期中期末按時繳交.

- 教科書

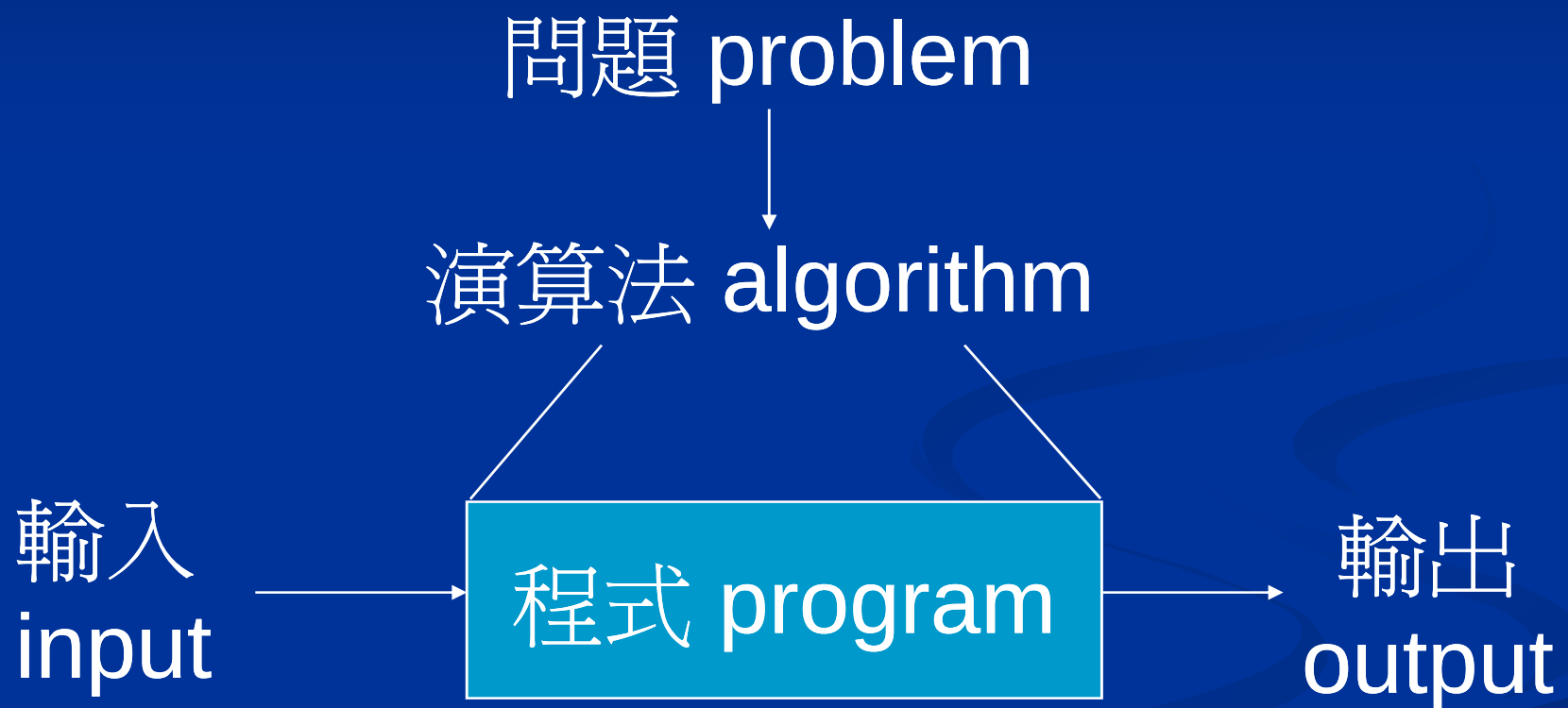
  書名: 演算法概論

  作者: 蔡郁彬等　　出版社: 學貫

課程關聯



什麼是演算法？

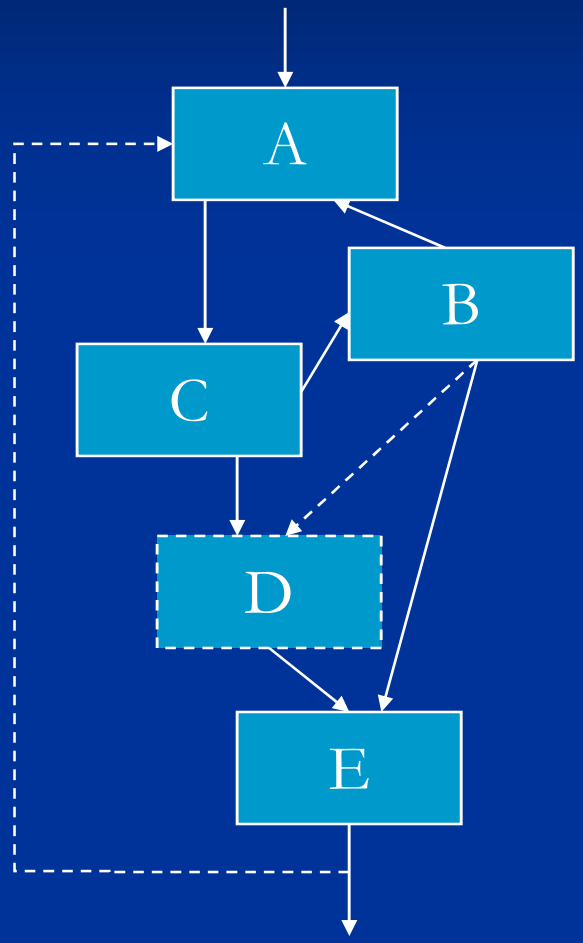


什麼是演算法？

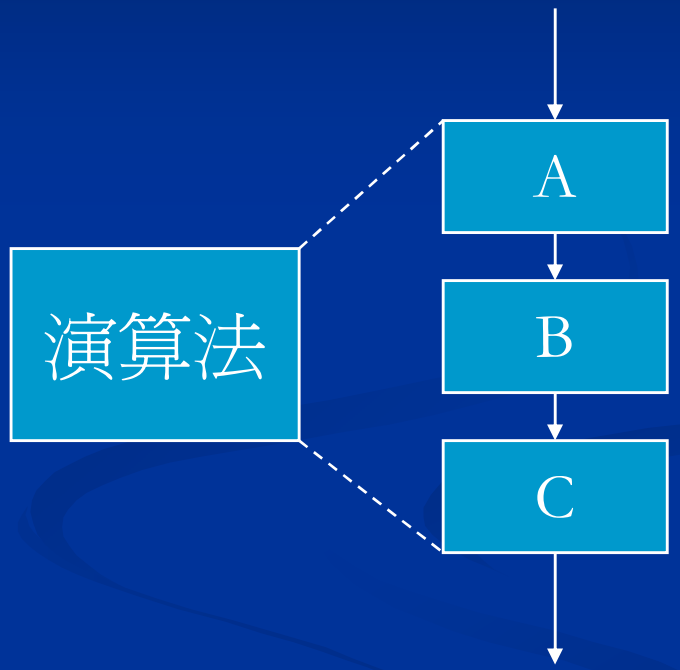
- 任何有良好定義的計算程序

A type of effective method which has a definite list of well-defined instructions for completing a task.

演算法的重要性



隨機拼湊、不斷試誤的程式



演算法作爲範本的程式

演算法的設計考量

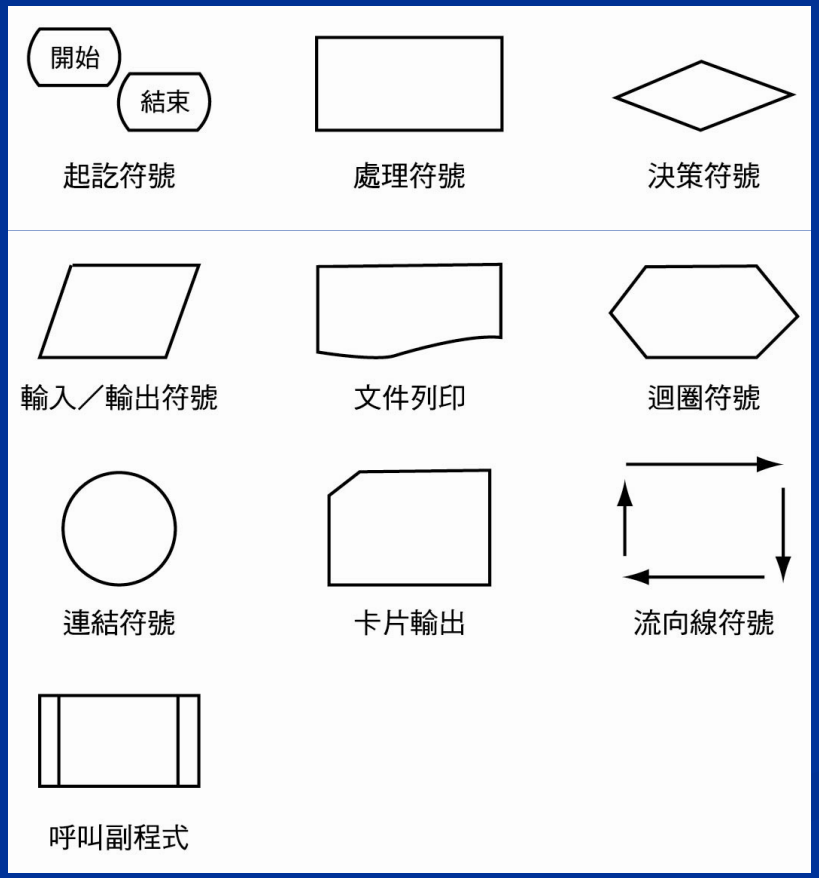
- 如何表示演算法？
- 如何設計演算法？
- 如何證明問題解決的正確性？
- 如何評估效能？
- 如何進行最佳化？

演算法的表示

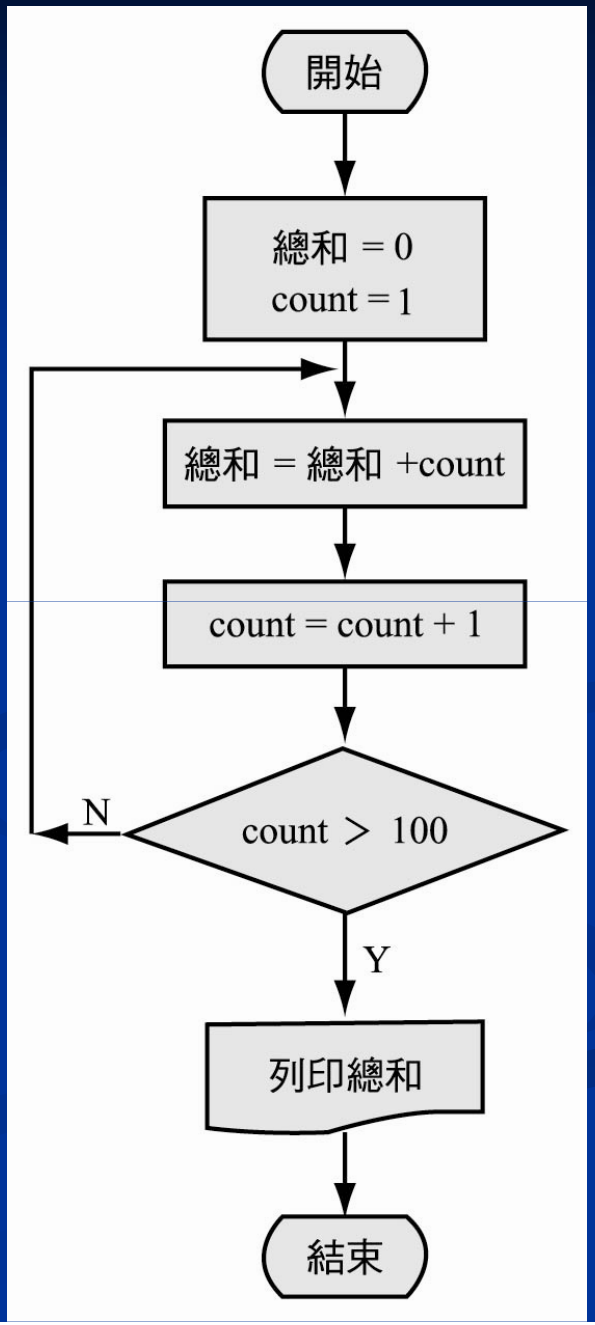
- 文字
  1. 輸入兩個自然數 A, B。
  2. A 除以 B 餘數爲 R。
  3. 如果 R 爲零，則跳至 5.。
  4. B 的值給A (B=A)，R 的值給 B (B=R)，跳至 2.。
  5. B 即爲最大公因數。

演算法的表示

- 流程圖 (Flowchart)



例：1+2+3+…+100=?



演算法的表示

- 虛擬碼 (pseudocode)

  例：1+2+3+…+100=?

  (1) 令 count=1，sum=0

  (2) sum=sum+count

  (3) count=count+1

  (4) if count>100 then

  執行(5)

  else

  執行(2)

  (5) print sum

演算法的表示

- 虛擬碼 (pseudocode)
  例：1+2+3+…+100=?

```
ALGORITHM CountUp(n)
    k ← 1，y ← 0
    while k<=100 do
            y ← y+k
            k ← k+1
    return y
```

演算法的表示

- C 語言

例：1+2+3+…+100=?

```
int count_up(){
    int k,y;
    for(k=1, y=0; k<=100; k++) y=y+k;
    return y;
}
```

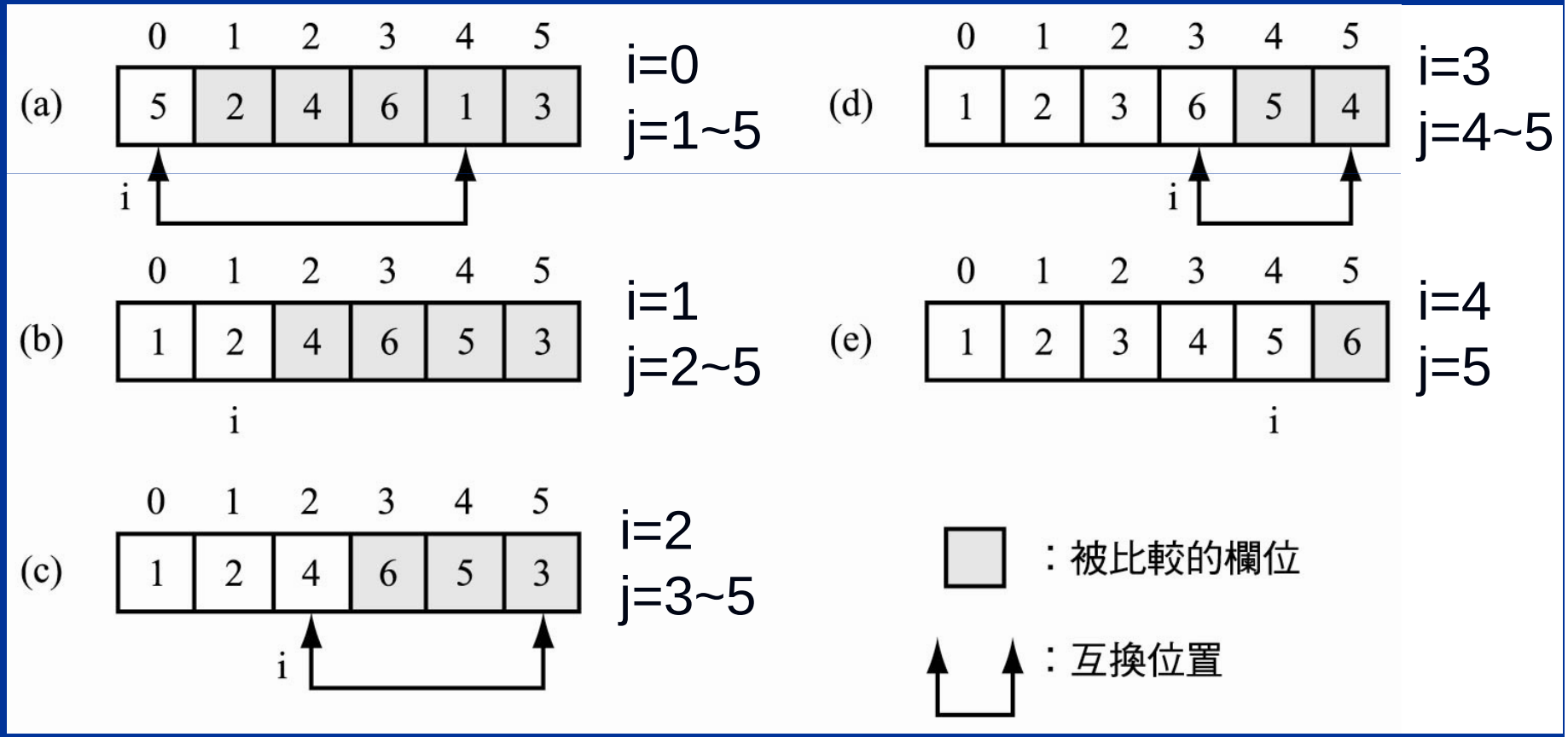
如何設計演算法？

- 瞭解問題
- 分析問題
- 設計解決步驟

演算法設計實例

- 排序 (構想: 選擇排序法)

從全部未排序好的元素中，挑出最小的一個放在最左邊



由演算法設計程式

i=0
j=1~5

i=1
j=2~5

i=2
j=3~5

i=3
j=4~5

i=4
j=5

```
n=6;

for(i=0;i<=n-2;i++)   // i 由 0 到 n-2
{    selected=i;
     for(j=i+1;j<=n-1;j++)  // j 由 i+1 到 n-1
          if (A[j]<A[selected])
               selected=j;
     if (i != selected)
          swap(A, i, selected);
          //交換 i 與 selected 位置
}
```

選擇排序法

```
void SelectionSort(int A[ ],int n){
    int i, j, selected;
    for(i=0;i<=n-2;i++){    selected=i;
          for(j=i+1;j<=n-1;j++)
                if (A[j]<A[selected]) selected=j;
          if (i != selected) swap(A, i, selected);
    }
}
void swap(int A[ ], int k, int m){
    int tmp;  tmp=A[k]; A[k]=A[m]; A[m]=tmp;
}
```

演算法的表示

```
ALGORITHM SelectionSort(A[0..n-1])
   for i ← 0 to n-2 do
        selected ← i
        for j ← i+1 to n-1 do
                if A[j]<A[selected]
                        selected ← j
        if i != selected
                swap A[i] and A[selected]
```

完整的程式

問題規模　　資料結構: 陣列　　輸入:隨機產生數值

```
main(){
  int n=10;   int A[n];      int k;
  srand(time(0));
  for(k=0; k<n; k++) { A[k]=rand()%n;
      printf("%d ", A[k]);
  } printf("\n");
  SelectionSort(A, n);
  for(k=0; k<n; k++) printf("%d ", A[k]);
}
```

演算法

輸出:排序之結果

測試結果

參考範例 1-1

- n=5　　3 0 4 3 2
  0 2 3 3 4
- n=10　　7 7 8 1 3 9 1 0 7 0
  0 0 1 1 3 7 7 7 8 9
- n=15　　4 9 13 12 6 2 7 11 6 7 3 2 2 13 5
  2 2 2 3 4 5 6 6 7 7 9 11 12 13 13

如何證明問題解決的正確性？

$1+2+\ldots+n=\frac{1}{2}n(n+1)$

- 數學歸納法 (mathematical induction)
  1. 驗証 n=1 時命題成立
  2. 假設 n=k 時命題成立，証明 n=k+1 時命題成立。

  根據1、2 可以推論命題對一切自然數 n 都成立。

如何證明問題解決的正確性？

- 強化歸納法 (strong induction)
  1. 驗證 n=1 時命題成立。
  2. 假設 n=k, n=k-1, ..., n=1 命題均成立，證明此命題在 n=k+1 時成立

  那麼一切自然數 n 來說，命題都成立。

- 給予任意 n 值，演算法結果均正確，則此演算法即能正確地解決問題

演算法的效能

- 依據問題規模所需要的空間與時間

  O (big-oh) 上限
  Ω (big-omega) 下限
  Θ (big-theta) 上、下限

- 演算法效能通常指所需要的時間上限

效能量測

- 使用 bash 的 time 指令

  `$ time 執行檔`

- 使用 clock() 函數

```
#include <time.h>
int t0, t1;
t0 = clock() / (CLOCKS_PER_SEC / 1000);
執行演算法
t1 = clock() / (CLOCKS_PER_SEC / 1000);
printf("%f sec", (t1-t0)/1000.0);
```

參考範例 1-2 與 1-3

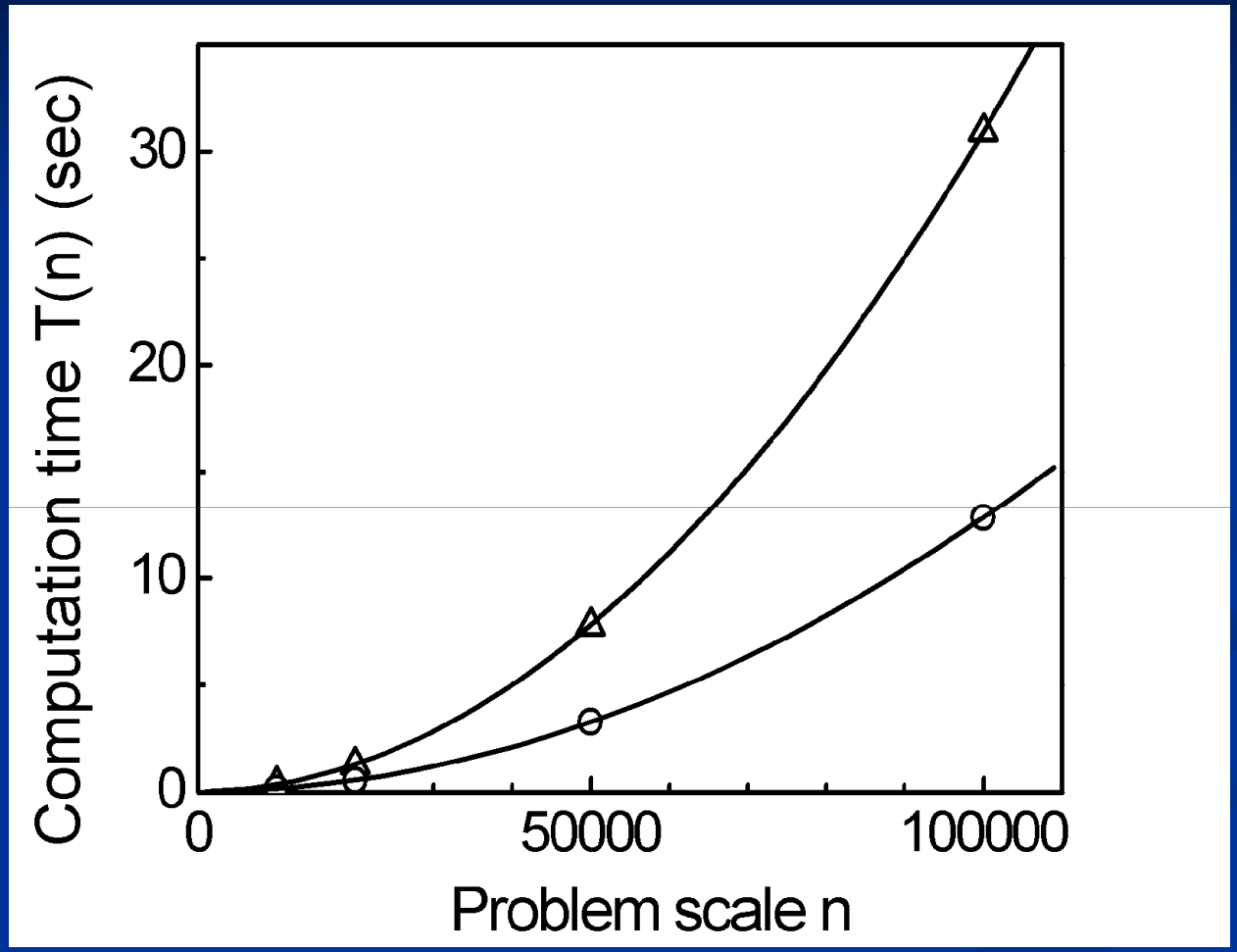
效能量測

- 設定較大的 n 值來測試演算法

n=10000	0.343s	0.155s
n=20000	1.265s	0.546s
n=50000	7.765s	3.254s
n=100000	30.936s	12.822s

(部分使用組合語言)

效能測量



計算時間與問題規模的關係 $T(n)=A+Bn+Cn^2$

效能分析

for(i=0;i<=n-2;i++)　i 由 0 到 n-2

　　for(j=i+1;j<=n-1;j++)　j 由 i+1 到 n-1

迴圈完成約需要 C(n) 個比較

$C(n)= (n-1) + (n-2) + \dots + 1 = \frac{1}{2}(n-1)n$

$= \frac{1}{2}(-n+n^2)$

運算時間

$T(n) = T_0C(n) = Bn+Cn^2$　時間複雜度 $\Theta(n^2)$

效能分析

- 演算法所需要的記憶體空間
- 指令空間 ~ 常數
- 資料空間 ~ 常數 空間複雜度爲 Θ(1)
- 堆疊空間 ~ 常數

如何進行最佳化？

- 增加計算能力: 更快的電腦、更多的 CPU
- 改善程式結構，使用低階語言來增進效能
- 根據問題規模，適當選擇演算法

應當建立的能力

- 由問題構想演算法的能力
- 由演算法設計程式的能力
- 由程式效能驗證演算法理論的能力