# 模擬 (Simulation)

丁培毅

# 運用電腦的速度與記憶體

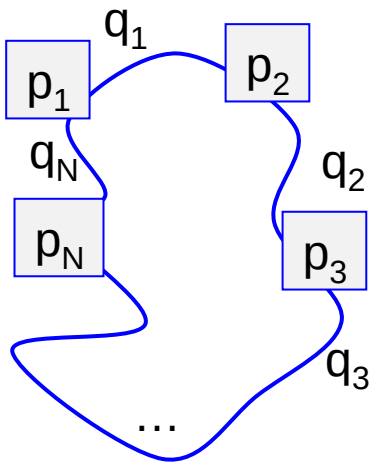
- 很多事情其實**方法很簡單**, 只是需要很大量的計算, 需要紀錄很多的資料, 才能夠完成, 這個時候我們可以盡量運用電腦的速度和記憶體來達成我們的需求
- **模擬 (Simulation) –** 就是當方法很簡單, 只是數量很大, 或是需要相當多的記憶體, 有許多參數需要嘗試時, 讓電腦模擬每一種參數的運作, 嘗試看看來得到結果
- 以下我們用一些例子來說明什麼時候可以用模擬來解決問題, 比較廣義的模擬其實也不要求方法很簡單, 例如一些機率的實驗, 用模擬來做時常常是因爲機率計算太複雜了, 不如用實驗方法來做, 比較常聽到的名字是蒙地卡羅法 (Monte Carlo method); 又例如很多暴力的搜尋法像是數獨、西洋棋、象棋、圍棋一類的如果你可以考慮所有的可能, 其實可以找到最好的答案

# 計算某一天是第幾天

- 指定某一個日期 (例如: 2015/10/10), 請問是由 1970/01/01 算過來第幾天
- 例如: 1970/01/01答案是第 1 天, 1970/01/25 是第 25 天, 1971/01/01 是第 366 天, 2015/10/10 是第 16719 天
- 程式該怎麼算出這個答案呢? 如果是手算的話, 最簡單的概念也就是屈指數一數了, 不過 2015/10/05 真的用手指頭算有點辛苦了, 還是用程式算吧, 也就是模擬一下, 由 1970/01/01 日開始是第 1 天, 一直算到 2014, 每過一年加上 365 (平年) 或是 366 (閏年), 由 1 月開始算到 9 月, 每過一個月加上那個月有幾天 (閏年 2 月是 29 天), 最後再加上 4 (那個月幾號), 如此模擬一下屈指數數的動作

# UVa 11093 Just Finish it Up

- 沿著一環狀道路上，有 N 個加油站，編號從 1 到 N，第 i 個加油站提供 $p_i$ 加侖的汽油。比賽從第 i 個加油站開始，依順時針方向到第 j 個加油站加 $p_j$ 加侖的汽油，假設汽車的油箱永遠足夠大，同時汽車起跑時油箱是空的。你的任務是確定汽車是否能完成比賽，如果它可以完成，從哪一個加油站開始可以完成? 請找到最小的 i 值。
- **輸入**：整數 T 爲測試案例的數量，每個測試案例包含一個整數 N，表示有 N 個加油站，接下來 N 個整數代表加油站 i 提供 $p_i$ 加侖的汽油，隨後的 N 個整數 $q_i$ 是汽車由加油站 i 到加油站 i%N+1 所需的汽油量 (單位加侖)。
- **輸出**：對於每個測試案例，請輸出 “Case t: ”，其中 t 是從 1 到 T，然後顯示是否可能完成一圈。如果不可能，顯示 “Not possible\n”；如果可能，則顯示 “Possible from station x\n”，其中加油站 x 是編號最小而可以完成一圈的加油站。



- T < 25, N < 100001
- 輸入：

```
2
5
1 1 1 1 1
1 1 2 1 1
7
1 1 1 10 1 1 1
2 2 2  2 2 2 2
```

- 輸出：

```
Case 1: Not possible
Case 2: Possible from station 4
```

- 這個題目沒有很簡單的的公式由 $\{p_i\}$ 和 $\{q_i\}$ 這兩組數字直接計算出到底有沒有辦法跑得到, 所以乾脆用模擬 (Simulation) 的方法來做, 由第 i* 個加油站開始, 補充 $p_{i*}$ 的油, 如果油箱裡油量超過 $q_{i*}$, 就足夠跑到下一個加油站, 用掉 $q_{i*}$ 的油, 如此如果一路順利可以跑回第 i* 個加油站, 答案就是 Possible from station i*; 如果還沒有跑回第 i* 個加油站前, 在某一個加油站發現油不夠了, 就表示不能由 i* 開始, 需要由 i*+1 開始…
  上面模擬時, i* 由 1 開始測試, 一找到可以的就直接輸出, 如此就會是編號最小的

# The 3n+1 Problem

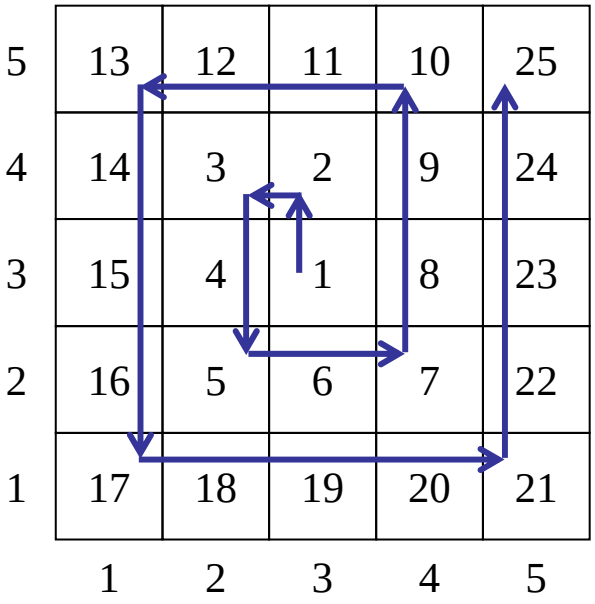
- 右側這個演算法當輸入是 22 時, 印出的數字爲
  22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
- 大家相信這個演算法不管輸入是多少應該是會結束的, 不過並沒有證明
- 不過至少對於小於1000000 的整數 n 來說, 都有驗證過

```
1. input n
2. print n
3. if n=1 then STOP
4.      if n is odd then n←3n+1
5.      else n←n/2
6. GOTO 2
```

- 對某一個整數 n 來說, 印出來的數字個數稱爲其 cycle-length, 例如 22 的 cycle-length 爲 16
- 請寫一個程式, 輸入兩個整數 i 及 j, 尋找 i, i+1, i+2, …, j 這些數字的 cycle-length 的最大值
- 這個問題也沒有什麼可以推導的, 最直接的方法就是模擬了, 對於每一個 n 值, 執行一次這個演算法就知道 cycle-length 是多少了

# Spiral Tap

- Spiral Tap 這個遊戲的介面是一個邊長爲奇數 n 的正方形棋盤, 棋盤上標示著 1 到 $n^2$ 之間的數字, 標示的方法如下圖, 最中間是 1, 反時針逐漸向外排列直到 $n^2$, 遊戲者輸入一個 1 到 $n^2$ 之間的數字, 需要很快地轉換成它的座標, 例如 23 位於第 3 列、第 5 行, 所以對應到座標值 (x,y)=(5,3)
- 請撰寫一個程式能夠很快地完成這個數字到座標的轉換工作



# Spiral Tap – 題目簡化

- 這個程式可以不需要使用陣列的語法
- 目標: 製作一個能夠將 target 數字轉換爲座標 (ix, iy) 的程式
- 一眼看起來不太容易, 可以先思考右邊這個簡單一點的問題

formula:
```
iy = (target-1) / 5 + 1;   // target 在哪一列?
ix = (target-1) % 5 + 1; // 在那一行呢?
```

iy

| | | | | | |
|---|---|---|---|---|---|
| 5 | 21 | 22 | 23 | 24 | 25 |
| 4 | 16 | 17 | 18 | 19 | 20 |
| 3 | 11 | 12 | 13 | 14 | 15 |
| 2 | 6 | 7 | 8 | 9 | 10 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| | 1 | 2 | 3 | 4 | 5 |

ix

- 複雜一點的問題呢?

| | | | | | |
|---|---|---|---|---|---|
| 5 | 25 | 23 | 19 | 13 | 5 |
| 4 | 22 | 18 | 12 | 4 | 9 |
| 3 | 17 | 11 | 3 | 8 | 16 |
| 2 | 10 | 2 | 7 | 15 | 21 |
| 1 | 1 | 6 | 14 | 20 | 24 |
| | 1 | 2 | 3 | 4 | 5 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | 13 | 12 | 11 | 10 | 25 |
| 4 | 14 | 3 | 2 | 9 | 24 |
| 3 | 15 | 4 | 1 | 8 | 23 |
| 2 | 16 | 5 | 6 | 7 | 22 |
| 1 | 17 | 18 | 19 | 20 | 21 |
| | 1 | 2 | 3 | 4 | 5 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | 9 | 16 | 21 | 24 | 25 |
| 4 | 8 | 15 | 20 | 22 | 23 |
| 3 | 7 | 14 | 17 | 18 | 19 |
| 2 | 6 | 10 | 11 | 12 | 13 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| | 1 | 2 | 3 | 4 | 5 |

- 推導公式不見得很直覺, 至少需要相當多時間, 爲什麼不直接模擬排放這些數字的過程, 由 1, 2, 3, …一直排到 target, 過程中就可以知道每一個數字的座標點

```
for (i=1, ix=iy=1; i<target; i++) {
    ix = ix+1;
    if (ix==6) ix=1, iy = iy+1;
}
```

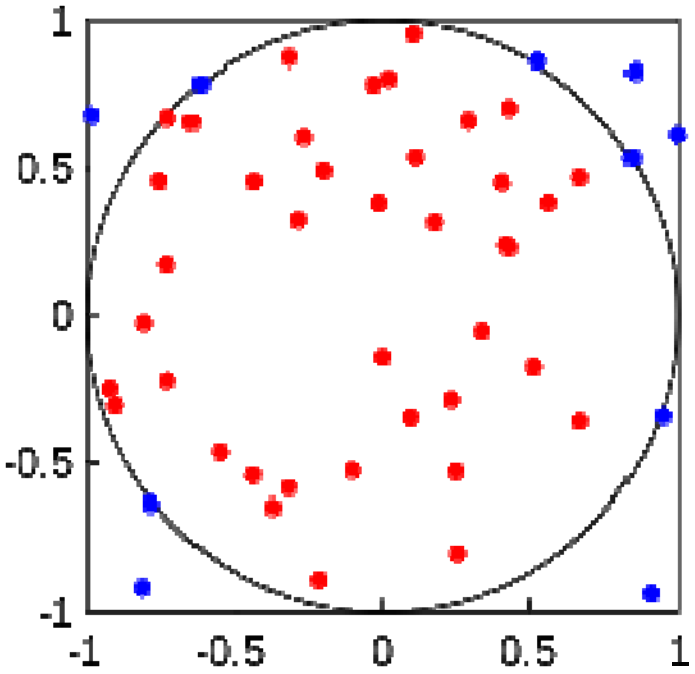
# 機率事件的模擬

- 有些機率事件描述起來很容易, 但是要計算發生的機率就有點複雜, 可以運用模擬的方式把那個事件的環境製作出來, 並且度量那個機率事件的發生頻率, 例如:
  - 三個不透明的袋子各裝有兩個球: 紅紅, 紅白,白白, 請問挑選一個袋子, 抽出第一個球是紅球的情況下, 第二個球也是紅球的機率是多少
  - 五個人玩撲克牌, 一開始每個人發五張牌, 請問其中一個人手上拿到 同花順的機會是多少?
  - 三個人輪流丟兩個骰子, 如果丟出的點數比前一個人丟出的點數少 1, 就算是贏家, 請問三個人贏得此賽局的機率各爲多少

  。 。 。

# Monte Carlo 積分

- 右圖是一個簡單範例, 假設要估計內切圓的面積, 由於正方的面積很容易計算, 藉由均勻產生的亂數分佈在封閉區間裡的比例, 可以直接估計封閉區間的面積
- 通常是用來估計那些不容易運用積分來計算面積的區域



# 其它範例

- 醉漢過橋
- Conway's Game of Life, https://www.youtube.com/watch?v=E8kUJL04ELA
- UVa 10409, Die Game (模擬骰子滾動)
- UVa 10415, Eb Alto Saxophone Player (薩克斯風按鍵模擬計算每一指按壓次數)
- UVa 118, Mutant Flatworld Explorers (模擬機器人的運動以及學習)

# 暴力搜尋 (Brute-Force Search)

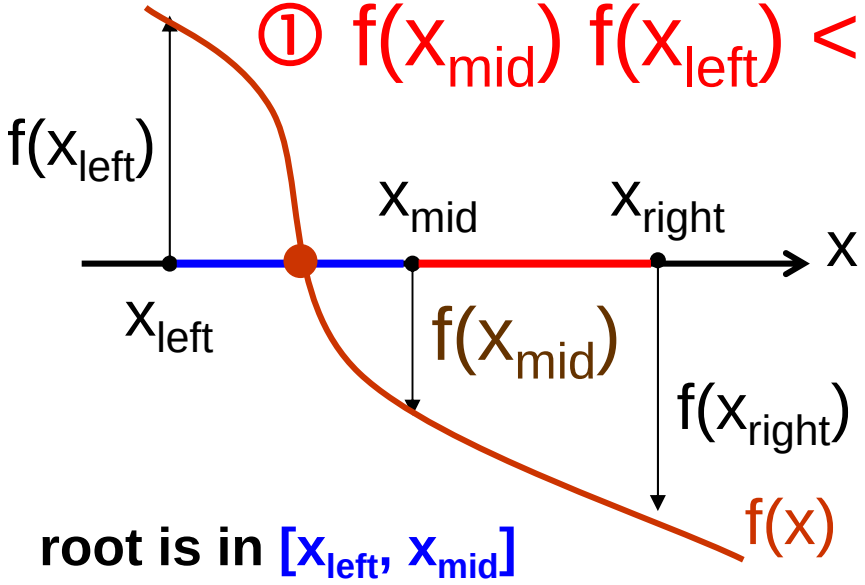
- 其實暴力搜尋或是地毯式的搜尋廣義來看也是一種模擬, 不是模擬實際世界的某一個系統, 而是模擬人手動解題 XD
- 很多時候一個問題的答案空間可以很快地確定下來, 例如
  - Travelling Salesman Problem (TSP): 有 n 個城市, 兩兩之間有一條公路, 距離都是已經量測出來的, 請尋找一條路徑, 能夠經過每一個城市恰好一次, 而且總距離是最短的 (或是小於某一指定的距離的路徑是否存在)
  - Subset Sum Problem: 有 n 個物品, 各自標示其重量, 是否存在一個子集合, 所包含的物品的總重量爲指定的 w
  - The Partition Problem: 有 n 個物品, 各自標示其重量, 是否存在一種切割方法, 把這些物品分爲相同重量的兩群
  - 西洋棋、象棋、圍棋、數獨、分解因數、離散對數…
- 列舉所有可能的答案, 一一測試 (NP 問題) 是一種很直覺的解法, 人來做太慢了, 寫成程式模擬各種狀況會快許多

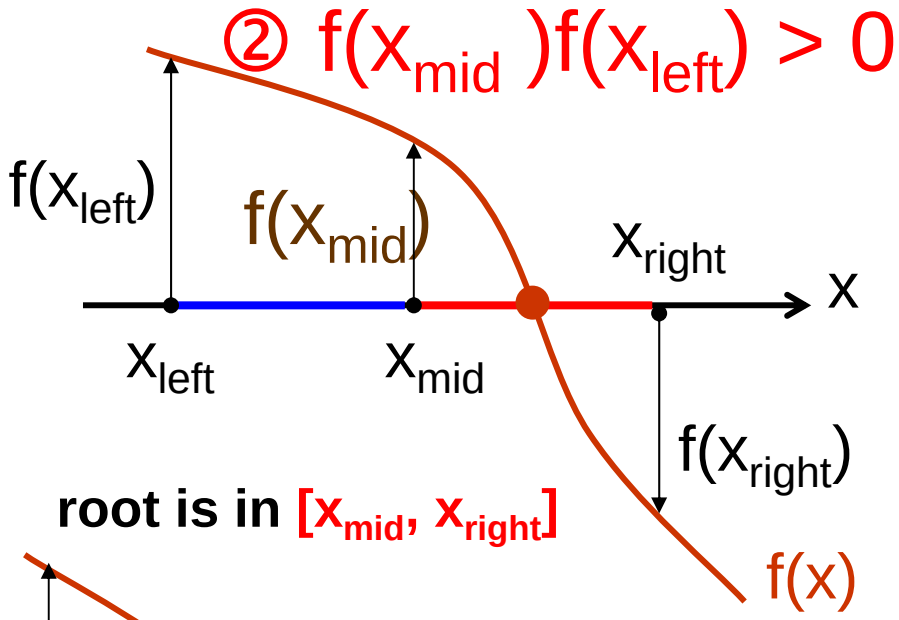
# 尋找完全數 (Perfect Number)

- 一個正整數是 完全數 (perfect number) 如果這個數字等於所有小於它本身的因數的和, 例如 6 是一個 完全數 因爲 6 = 1 + 2 + 3
- 請寫一個程式, 使用者由鍵盤輸入一個正整數, 程式判斷小於等於這個整數的所有正整數中哪幾個數字是 完全數
- 撰寫這個程式的時候你需要想像如果你是電腦的話你要怎樣來完成這個題目的要求? (你可能沒有很好的公式或是很好的數學定理可以讓你很快地完成這個工作, 但是你有很快的運算速度, 可以用很簡單但是很暴力的作法來達成上面的要求)
  - 先弄清楚範圍 (先讀取使用者要求的數字上限)
  - 一個數字一個數字來判斷是否爲完全數 (也許需要把每個數字的真因數都找到)

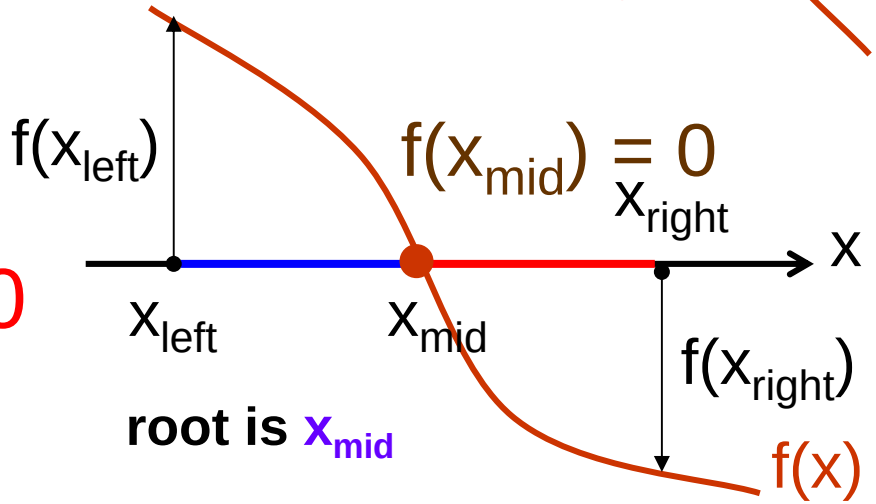
# 二分法勘根

- 用二分逼近的方法來尋找函式 f() 的根, 其實概念很簡單, 就讓電腦用它的速度來幫我們一步一步做一次吧




③ f(x_mid) == 0



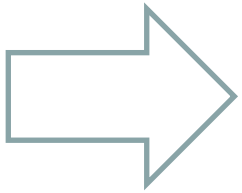
# 數獨遊戲 (Sudoku)

- 在這三張圖片裡, 九列九行的盤面上共有 9 x 9 = 81 格, 這 81 格又切成 9 個大區塊, 每個區塊是 3 x 3 的九宮格。 在盤面上已經有很多格已經填入數字 1~9, 這個遊戲就是要玩家在所有其他空格填入 1~9 的數字, 滿足下列條件
  - 每一個 3x3 的九宮格裡沒有重複的數字
  - 每一列沒有重複的數字
  - 每一行沒有重複的數字

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 6 | 1 | | 3 | 4 | | |
| | | 3 | | | 8 | | | |
| 5 | 4 | | 7 | | | 1 | 2 | |
| | | | 2 | | | | | 4 |
| | 5 | | 3 | | 9 | | 7 | |
| 7 | | | | | 4 | | | |
| | 3 | 7 | | | 5 | | 4 | 2 |
| | | | 8 | | | 7 | | |
| | | 1 | 4 | | 7 | 8 | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | | 8 | | | | 6 | | |
| | 6 | | | | | 4 | | |
| | 9 | | 8 | 5 | | | 1 | |
| 2 | | | 9 | | | | | |
| 4 | | 5 | | | | 9 | | 2 |
| | | | | | 7 | | | 6 |
| | 2 | | | 1 | 3 | | 6 | |
| | | 9 | | | | | 3 | |
| | | 1 | | | | 5 | | 4 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | 8 | 9 | | 2 | | | | 5 |
| 6 | | | | | 5 | | 8 | |
| | | | | | | | 1 | |
| 2 | | | 8 | | | | 5 | 1 |
| | | 5 | 7 | | 1 | 6 | | |
| 9 | 1 | | | | 6 | | | 3 |
| | 7 | | | | | | | |
| | 5 | | 9 | | | | | 6 |
| 8 | | | | 1 | | 5 | 3 | 7 |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | 8 | 9 | 1 | 2 | 4 | 3 | 6 | 5 |
| 6 | 4 | 1 | 3 | 7 | 5 | 2 | 8 | 9 |
| 5 | 2 | 3 | 6 | 8 | 9 | 7 | 1 | 4 |
| 2 | 6 | 7 | 8 | 4 | 3 | 9 | 5 | 1 |
| 4 | 3 | 5 | 7 | 9 | 1 | 6 | 2 | 8 |
| 9 | 1 | 8 | 2 | 5 | 6 | 4 | 7 | 3 |
| 3 | 7 | 4 | 5 | 6 | 8 | 1 | 9 | 2 |
| 1 | 5 | 2 | 9 | 3 | 7 | 8 | 4 | 6 |
| 8 | 9 | 6 | 4 | 1 | 2 | 5 | 3 | 7 |