5. Randomized Algorithms

隨機演算法

中國文化大學
資訊工程學系
副教授　張耀鴻

112年度第2學期

Outline

- 5.1 Hiring Problem
- 5.2 Indicator random variables
- 5.3 Randomized algorithm
- Supplementary: Algorithm Experimentation

5.1 Hiring Problem 徵才問題

Senario (情境)

- 請人力仲介公司幫忙找人來應徵
- 仲介公司每天會送一個新人來面試
- 必須當場決定是否雇用新人
- 一旦決定雇用新人必須Fire掉現職員工
- Cost:
  - 每次Interview要給仲介 c_i 元
  - 每次Hire新人要c_h 元(包括Fire掉舊人和付給仲介費用)
  - 假設 $c_h >> c_i$

5.1 Hiring Problem (cont.)

決策原則:

- 只要面試的新人條件比較好，就Fire掉現職員工並雇用新人
- 第1個來面試者一定會被錄用

Goal:

- 最後決定雇用1個員工總共花費是多少?

5.1 Hiring Problem (cont.)

- 演算法:
  1. 從第一個應徵者到最後一個應徵者依序面試
  2. 如果其中某一個應徵者比之前的好，則馬上錄取

```
HIRE-ASSISTANT(n)
best = 0              // candidate 0 is a least-qualified dummy candidate
for i = 1 to n        從第一個應徵者到最後一個應徵者
    interview candidate i              依序面試
    if candidate i is better than candidate best
        best = i                       如果其中某一個應徵者比之前的好
        hire candidate i               則馬上錄取
```

cost model 成本建模

- Cost to interview a candidate c_i 面試一位候選人
- Cost to hire a candidate c_h, suppose $c_h > c_i$
  雇用一位新人, 雇用的花費遠大於面試成本
- Suppose interview n people and m people are hired 假設面試了n個人, 而雇用了其中m個新人
- The cost for Hire-Assistant algorithm is
  徵才問題的成本可用Big-O表示為 $O(c_i n + c_h m)$
- Goal: Want to know the **min** and **max** cost
  欲知最大及最小的成本是多少

面試n人的成本 + 雇用m人的成本

cost analysis 成本分析

- **Worst case analysis 以最壞情況分析**
  - Suppose the quality of candidates come in strictly increasing order 假設每個候選人都比前一個優
  - We then have to hire every candidate 每個人都雇用
  - Cost = $O(c_h n)$
- **Probabilistic analysis 以機率模型分析**
  - Must make assumption about input distribution 需先假設輸入分佈(來應徵者的優劣分佈)
  - The expectation is over this distribution 成本可由機率分佈的期望值算出
  - Analyze average-case running time 分析平均所需花費時間
  - Required skill: Must be able to make a *reasonable* characterization of the input distribution 必須具備根據輸入資料的特徵判斷出合理分佈的能力

有點難

randomized algorithm 隨機演算法

- The input distribution is ***unknown*** *輸入分佈未知*
- Instead, use randomization within the algorithm to impose a input distribution
  在演算法中調整輸入順序以造成隨機效果
- Analyze expected running time 分析執行時間的期望值
- RANDOMIZED-HIRE-ASSISTANT(N) 隨機雇用助理

1. best = 0　一開始假設第0個是最好的人選
2. while (there is someone not interviewed) 還有人沒面試
3. Randomly pick candidate i　隨機選第*i*個候選者
4. if candidate i is better than candidate best
5. best = i　若第*i*個候選者是目前最好的
6. hire candidate i　則立刻FIRE掉現在的，再雇用第*i*個候選者

(指標隨機變數)

5.2 Indicator random variables

- Given a sample space **S** and an event **A**, the **indicator random variable** **I{A}** is defined as

S: 樣本空間

A: 事件

$$\mathrm{I}\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs ,} \quad \text{事件A發生} \\ 0 & \text{if } A \text{ does not occur .} \end{cases}$$

- **Lemma 5.1**
  - Given a sample space S and an event A,
  - let X_A = I{A}.
  - Then $E[X_A]$ = Pr{A}

指標隨機變數X_A的期望值即為事件A發生的機率

- e.g.定義X_H = I{H}: 丟一次銅板，預期會看到幾個頭
  - S = {H, T}, Pr{H} = Pr{T} = $\frac{1}{2}$
  - $E[X_H]$ = Pr{H} = $\frac{1}{2}$

analysis of randomized hiring problem 分析隨機徵才問題

- Assume qualified candidates arrive in a random order
  假設候選人以隨機順序到達
- Define random variable X = #hired 定義隨機變數X=雇用總人數
- Define indicator random variables $X_1, X_2, \ldots X_n$
  定義指標隨機變數 X_i = 第i個候選人被錄取
- X_i = I{candidate i is hired}
- Useful properties: 總共雇用人數X=雇用第1人+雇用第1人+…
  - ✓ $X = X_1 + X_2 + \ldots + X_n$ 第i人被雇用的期望值=雇用第i人的機率
  - ✓ **Lemma 5.1 → $E[X_i]$ = Pr{candidate i is hired}**
- Want to know: $E[X] = ?$

analysis of randomized hiring problem

候選人抵達順序是以隨機方式

- Candidate i is hired, iff candidate i is better than other 1..i-1 candidates
  「第i個候選人被錄取」換句話說「第i個候選人比前i-1人好」
- Because candidates arrive in random order
  ➔any candidate is equally be the best so far
  ➔ Thus, Pr{candidate i is the best so far} = 1/i
  ➔ implies $E[X_i] = 1/i$

候選人是以隨機順序抵達

第i個面試者是目前最好的機率

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \sum_{i=1}^{n} 1/i \\
&= \ln n + O(1)
\end{aligned}
$$

(by Equation C.21, 期望值有線性的特性)

(by Equation A.7)

- Thus, the expected hiring cost is $\boldsymbol{O(c_h \ln n)}$ much better than $\boldsymbol{O(c_h\, n)}$

Lemma 5.2:

- Algorithm HIRE-ASSISTANT has average-case hiring cost of $\boldsymbol{O(c_h \ln n)}$ 徵才問題的平均雇用成本

hat-check problem 衣帽間問題

- (Ex. 5.2-5 of CLRS)
- Each of n customers gives a hat to a hat-check person
- The hat-check person gives the hats back in random order
  n個客人把帽子交給衣帽間保管，管理員以隨機方式將帽子還給客人
- **Question**: What is the expected number of customers who get back their own hat?

  (預期有幾個客人可以拿回自己的帽子?)

Define random variable X = "#customer get back their hat." WANT TO KNOW: E[X] =?

隨機變數X="拿回自己帽子的人數", 欲知E[X]

Define indicator random variable X_i:

X_i = I{customer i gets back his own hat}

Then, X = X1 + X2 + … + Xn. 指標隨機變數X_i=「第i個人拿回自己帽子」

Since the order of hats is random

➔ Pr{Xi = 1} = 1/n ➔ E[Xi] = 1/n (By Lemma 5.1)

Thus,

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right]$$

$$= \sum_{i=1}^{n} E[X_i] \qquad \text{(linearity of expectation, C.21)}$$

$$= \sum_{i=1}^{n} 1/n$$

$$= 1$$

expect that exactly 1 customer gets back his own hat

5.3 Randomized algorithm (隨機演算法)

- Randomization is now in the algorithm, not in the input distribution.
  將隨機過程加入演算法中, 與輸入分佈無關
- Each time we run the algorithm, we can get a different hiring cost.
  每次執行可能得到不同的結果
- No particular input always elicits worst-case behavior.
  最壞情況不太可能發生
- Bad behavior occurs only if we get "unlucky" numbers from the random number generator.
  除非真的有夠衰……

pseudo code for randomized hiring problem

RANDOMIZED-HIRE-ASSISTANT(n)

1. randomly permute the list of candidates
2. Hire-Assistant(n)

1. 預先把 candidate 作亂數排列
2. 呼叫 HIRE-ASSISTANT(n)

- **Lemma 5.3** 隨機徵才問題的預期雇用成本

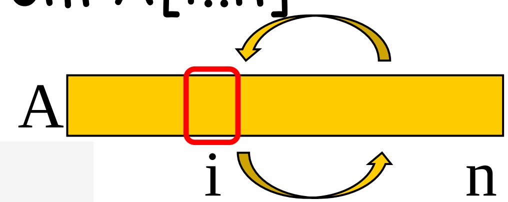
  The expected hiring cost of RANDOMIZED-HIRE-ASSISTANT is $O(c_h \ln n)$.

預先把 candidate 的 rank 作一次亂數排列，
那麼我們就可以合理的假設每次都是得到
一個 average case 的 input

randomly permuting arrays 隨機排列陣列

產生一個均等隨機排列

- **Goal**: Produce a uniform random permutation
- **Idea**: 在第 *i* 次循環時, 從第 *i* 個到最後一個元素中, 隨機挑一個與第 *i* 個位置對調.
  - ✓ In iteration i, choose A[i] randomly from A[i..n]
  - ✓ Will never alter A[i] after iteration i




```
RANDOMIZE-IN-PLACE(A)
n ← length[A]
for i ← 1 to n
      do swap A[i] ↔ A[RANDOM(i, n)]
```

從陣列第1個開始到最後一個, 依序指定*i*值, 迴圈中每次隨機與第*i*個後面的陣列元素對調

- Running time: O(1) per iteration ➔ ***O*(*n*)** total

Techniques and principles

ALGORITHM EXPERIMENTATION

演算法實證

Experimental setup 研究實驗設計

1. Choose question 選擇問題
2. Decide what to measure 決定要量測的對象
3. Generate test data 產生測試用的輸入數據
4. Coding and experiment 實作演算法並分析實驗輸出數據

1. Choose question

- 估計平均執行時間 average case asymptotic running time
- 比較幾個演算法在某範圍內執行效率
- 找出某個帶有參數的演算法之最佳參數
- 針對試圖求得某function之min或max的演算法，測試其與理想值接近程度

2. Decide what to measure

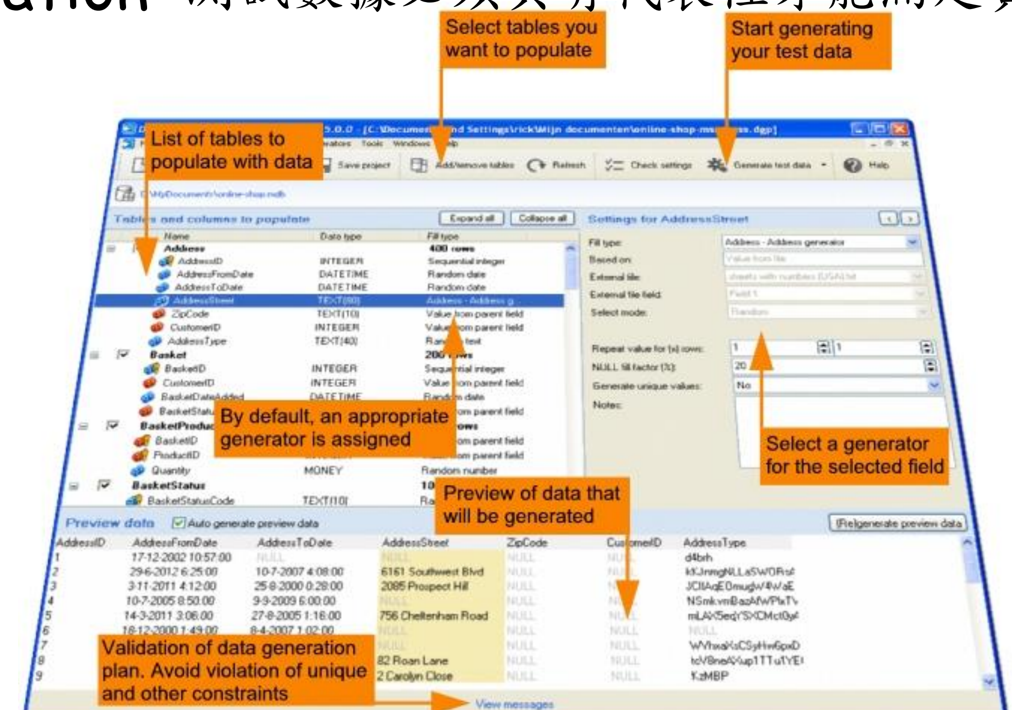
- ✓ Quantitative measurement 量化量測
- ✓ "Wall clock time" vs "CPU time"
- ✓ 常用基本分析
  - Memory reference 記憶體參照次數
  - Comparisons 比較次數
  - Arithmetic operations 算術運算次數

```
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
… /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER
```

- CPU Time: Time spent by the CPU
- Wall time: Time including disk I/O
- CPU time != Wall time? Likely waiting for disk

3. Generate test data

- ✓ Generate enough sample，使其平均能達統計上有效結果
- ✓ 產生不同大小的輸入樣本，to enable educated guess.
- ✓ Generate test data that is representative of practical expectation 測試數據必須具有代表性才能滿足實務上的期望



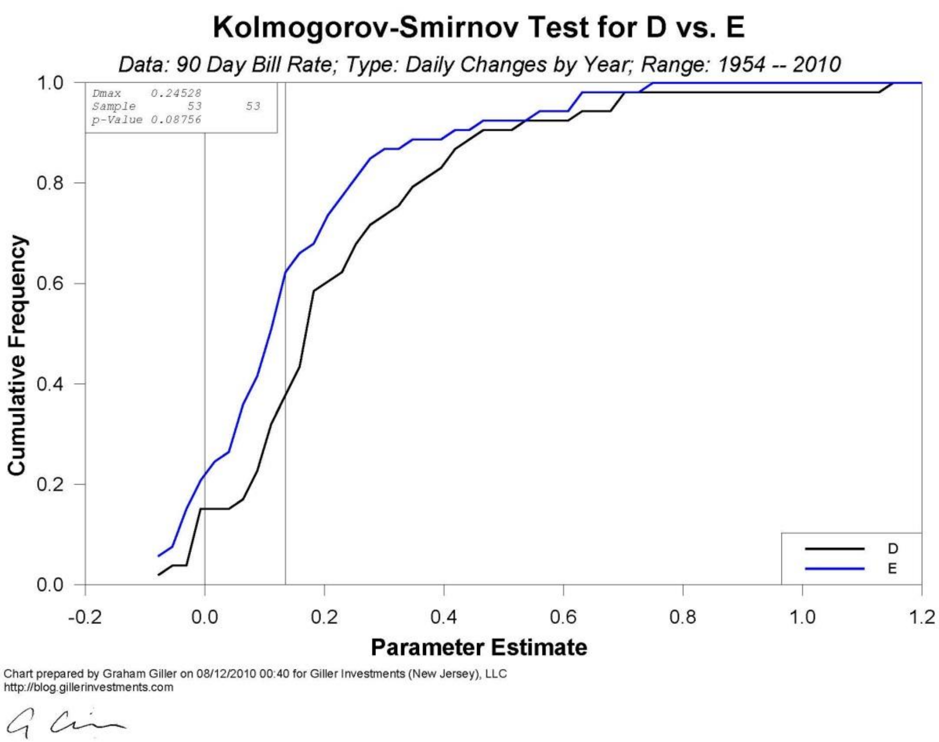
4. Coding and experiment

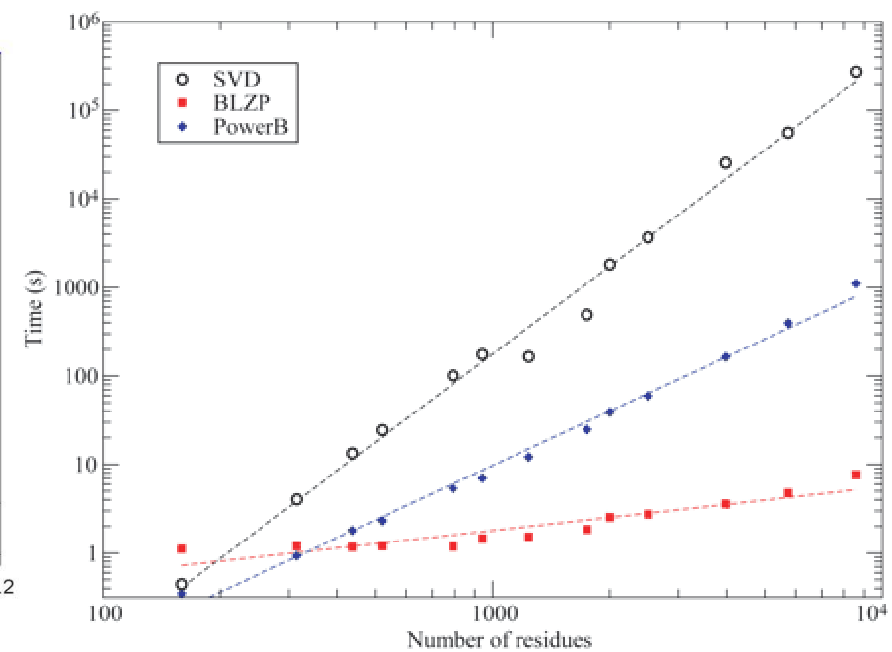
- ✓ 正確而有效的將演算法實作出來
- ✓ 盡力求得 reproducible (可重製) 的結果
- ✓ Perform experiment in a sterile environment
  實驗時需排除不必要的環境干擾

数據分析與視覺化

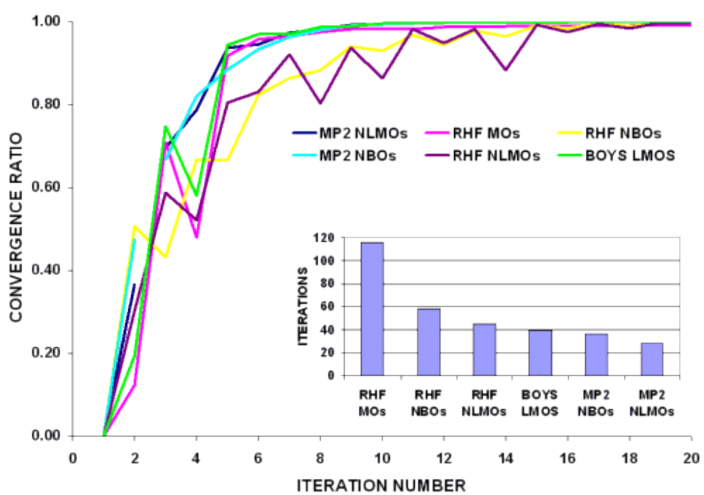
Data analysis and visualization

- Ratio test 比值審斂法: 判別級數對某特定值是否收斂
- Power test 檢定力(迴歸)分析法: 檢定某項假設是否成立

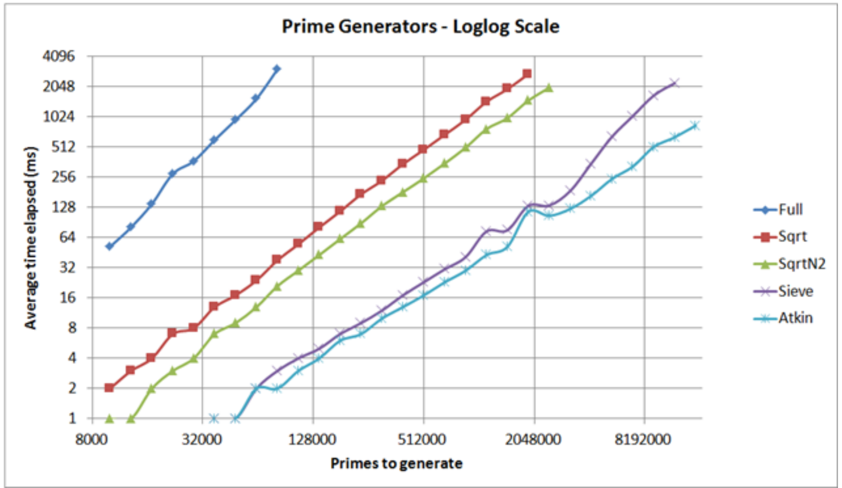




- Ratio test
  - 導出演算法執行時間 $T(n) = n^c$
  - 分析 **average running time** 是否為 $O(n^c)$
  - **蒐集數**所得依不同大小的n值所得的實際執行時間 t(n)
  - 繪出比值 r(n)次實驗= t(n)/T(n)
    - if r(n) grows as n increase ➔ T(n) under estimate
    - if r(n) converges to 0 ➔ T(n) over estimate
    - if r(n) converges to some constant b > 0 ➔ T(n) GOOD



- Power test
  - NO NEED to make a good guess!
  - 從實驗中蒐集(x, y)，其中 y = f(x), x 為輸入樣本大小
  - 轉換 (x, y) → (x', y')，其中 x' = lg x and y' = lg y (對數座標)
  - 繪出所有 (x', y') 並檢查結果
  - if $t(n)=bn^c$, log-log 轉換 implies y' = cx' + b
    - 若繪出結果為直線，則可求得 b 和 c
    - 若繪出結果向上彎，表示 algorithm 為 NP
    - 若繪出結果向下彎，表示 algorithm 為 sub-linear



Summary

- Hiring problem
  - HIRE-ASSISTANT(N) $O(c_i n + c_h m)$
- Cost analysis
  - Worst case analysis
  - Probabilistic analysis
  - Average case analysis
- Indicator random variable $X_i \in \{ 0, 1 \}$
- Hat-check problem
- Randomized algorithm
  - RANDOMIZED-HIRE-ASSISTANT(N) $O(c_h \ln n)$
- Algorithm experimentation
  - Ratio/power test