COMPUTER SCIENCE: BEYOND PROGRAMMING
電腦科學: 比「寫程式」更多

林鶴哲
台大資訊系 B07, R11 (2018.9 – Current)

2022.11.06

https://reurl.cc/DXgxpE

computer noun

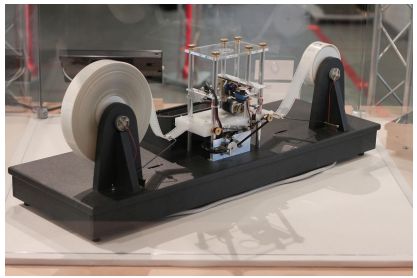
com·put·er kəm-ˈpyü-tər

often attributive

: one that computes

specifically : a programmable usually electronic device that can store, retrieve, and process data

> using a *computer* to design 3-D models



https://reurl.cc/eWxmVQ

只要你想做的事情需要**存資料**或是做**超出人力範圍的加減乘除**，電腦科學都可以摻一腳。

剩下十八分鐘，想帶你了解...

- 你不會說外文系在做的事情是學英文(或其他語言的) 單字
  那也不該說資工系就是寫程式
- 用「資訊腦」思考問題的例子
  - 把問題抽象化
  - 在寫程式之前必須設計好可以算出答案的方法(演算法)
- 資訊科學的問題不見得有「標準答案」
  - 運算時間 v.s. 儲存空間、運算時間v.s. 答案精準度、身份認證v.s. 隱私...

2015 高一的我想像中的資工系學生

Live Submission

Auto update

#		Problem Title	Username	Verdict		Submit Time	Style
487957	130	Similar Strings	B11902xxx	WA (score: 70)	C	about a minute ago	
487954	98	Food Ingredients in Common	B11902xxx	AC (72 ms, 512 KB)	C	11 minutes ago	
487953	72	Fill the Array	B11902xxx	AC (81 ms, 256 KB)	*	11 minutes ago	
487951	98	Food Ingredients in Common	B11902xxx	WA (score: 60)	C	17 minutes ago	

So they do lots of coding, then dive into AI, big data, wow 酷爆了!!!!!!!!!!

https://reurl.cc/zr8M1p

但事實上，online judge 只是砌磚頭的功夫

Live Submission

Auto update

#		Problem Title	Username	Verdict		Submit Time	Style
487957	130	Similar Strings	B11902xxx	WA (score: 70)	C	about a minute ago	
487954	98	Food Ingredients in Common	B11902xxx	AC (72 ms, 512 KB)	C	11 minutes ago	
487953	72	Fill the Array	B11902xxx	AC (81 ms, 256 KB)	*	11 minutes ago	
487951	98	Food Ingredients in Common	B11902xxx	WA (score: 60)	C	17 minutes ago	

程式語言<-> 磚頭
刷題 <-> 培養砌磚能力

Judge 系統能夠幫助我們知道程式語言的基礎使用規則

Beyond Programming: 在砌磚頭之外

https://reurl.cc/rZx1Kx

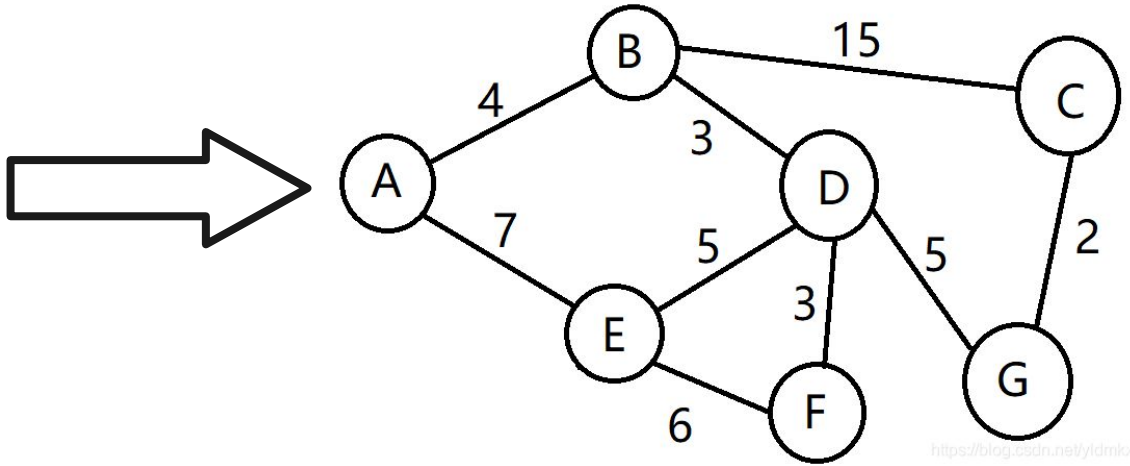
- 我們的究級目標是蓋房子 (那些很炫炮的應用)
  - 砌磚頭 (寫程式) 只是基本功、達到目標的方法。
- 所以在寫程式之外，我們要考慮什麼問題？例如
  - 程式概念的正確性
  - 系統有沒有滿足一開始發想的需求？
- 我會利用自己大學課堂的 Project 說明這兩個不同層次的對應

案例一：考慮紅綠燈的最速導航

- 想找「捷運科技大樓站」到「資訊系館」的最短路徑
- 行進時可能會遇到「連鎖紅綠燈」的問題
- 我想要導航系統考慮「最短時間」而非「最短路徑」

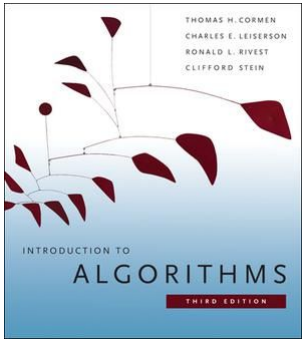
翻譯問題：從「地圖」到「圖」

https://reurl.cc/AyqRkd



問題抽象化是資訊腦具備的能力！

https://reurl.cc/bG5kV6

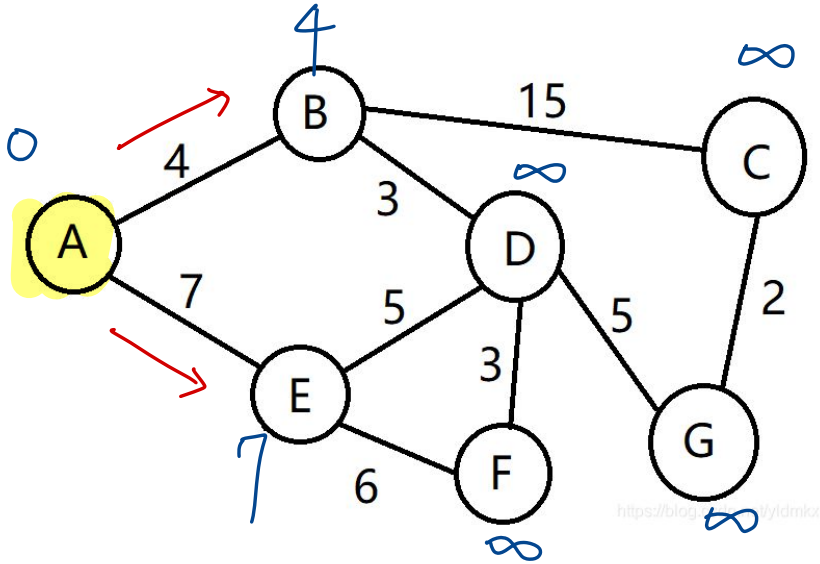


站在巨人的肩膀上: Dijkstra 發明的方法

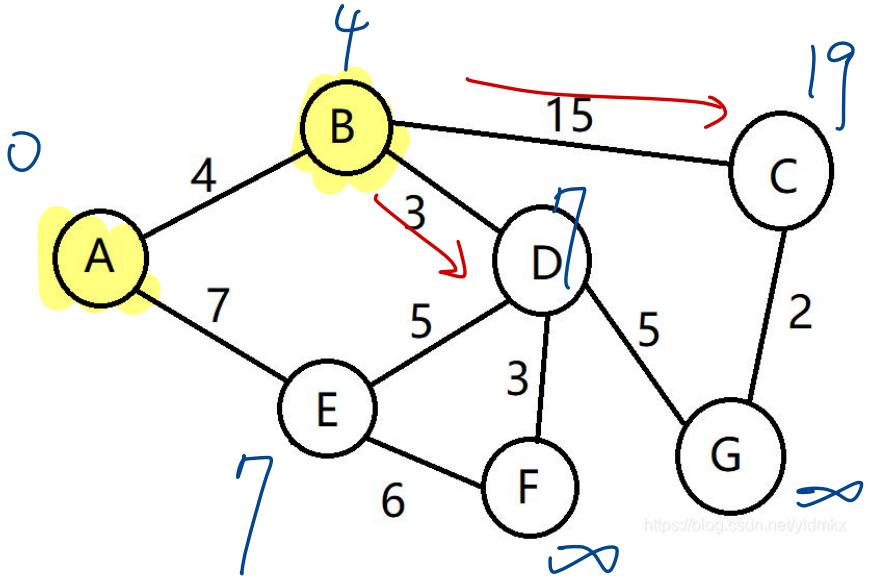
- Dijkstra's Algorithm: 給定一張「圖」(graph)，可以用某一套有規則的方法找出某兩點的**最短路徑**
  - Key Idea: 一直去找當前和起始點最近，但是還沒有算出最短路徑的點
  - 這個方法將會在大二上的演算法課程說明

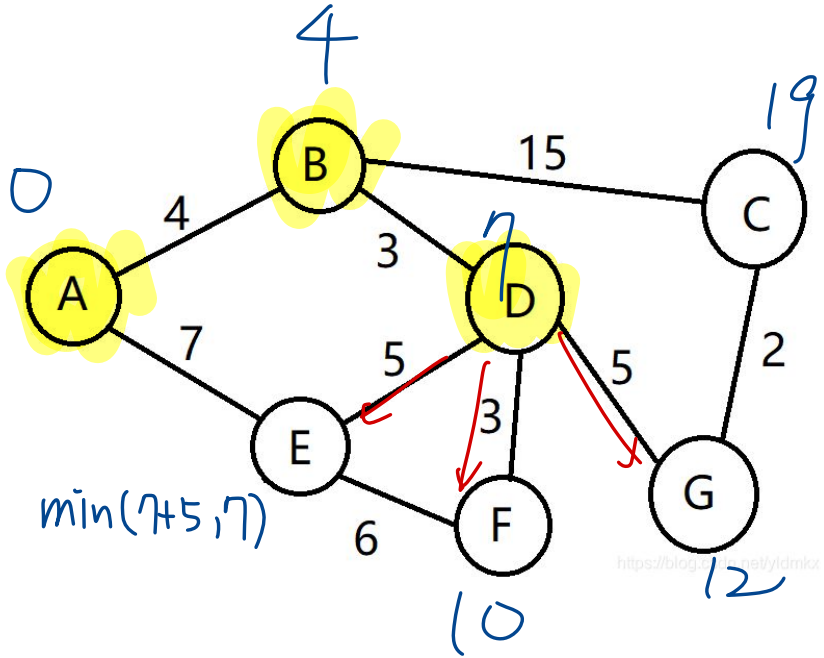
- 困難點：要考慮紅綠燈時間！
  - 後人的想法：把等待紅綠燈的時間放在「圖」中的各個節點上

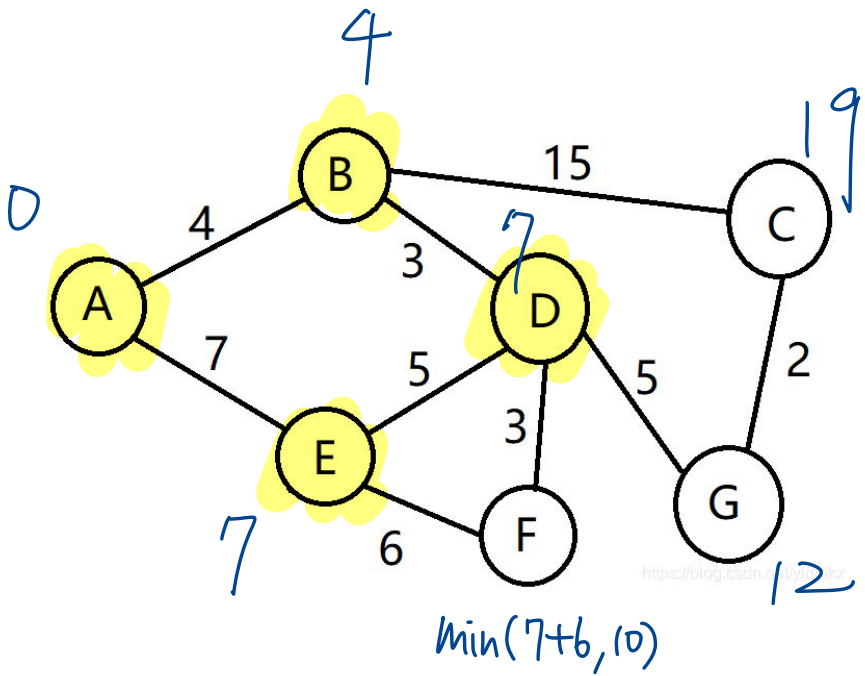
第一輪：拿A當作起點。



⇒ B是未塗成黃色的點中最近的！


A
B
C
D
E
F
G
4
15
4
3
7
7
5
5
2
3
6
0
4
19
7
∞
∞


4
19
B
15
C
0
4
3
7
A
D
7
5
5
2
3
E
G
min(7+5,7)
6
F
12
10


4
19
B
15
C
0
4
3
7
A
D
7
5
5
2
3
E
G
7
6
F
12
min(7+6,10)

我在這個 Survey Project 中沒有寫任何程式碼

- 為什麼沒寫程式還是 Computer Science?
  - 因為寫程式的成本很高，用精神去思考、感覺的成本很低 (x)
  - 抽象化的思考可以讓同個方法在不同情境下可以使用
- 針對問題提出演算法 <-> 設計蓋房子牆面、樑柱的工法
- 有正確答案，沒有「正確作法」

https://reurl.cc/nZzEDn

還有很多問題要處理

- 我們只給出了一個會正確運作的方法
  - 算一次要多久？
  - 可以支援台北到高雄的導航嗎？
  - 中途遇到紅綠燈壞掉/車禍事故要怎麼做路徑更新？

- 最速路徑的計算是在手錶上的晶片算，還是丟給水果公司的中央系統電腦算？
  - 本地計算：儲存空間夠存全台灣紅綠燈資料？
  - 給水果公司算：隱私問題（被查水表？？）

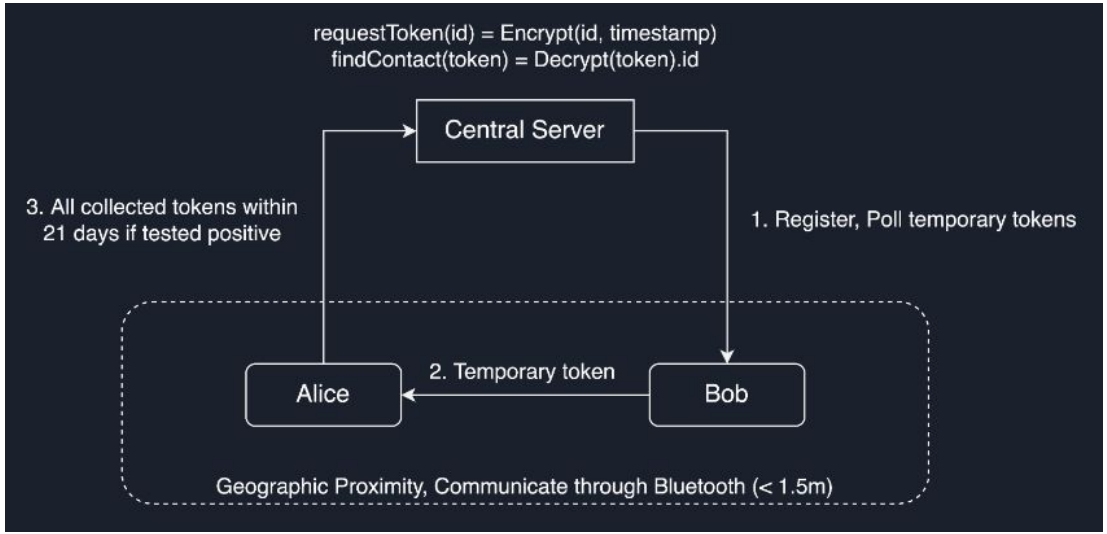
案例二：社交距離 APP

- （情境對白）CECC：能不能設計一個顧及隱私的狀況抓出密接者的系統？
- 這比「最速路徑」的問題更加「白話文」
  - 很「文組」，很「不量化」
- 這是設計系統（system）、協議（protocol）或是機制（mechanism）時會遇到的問題
  - 就好比蓋房子時設計整棟建築物，我們是建築師。
  - 常常用的網路協議（IP, https）設計也都有這些問題要考慮

- 先備知識：這種系統大多是藉由手機間的藍芽進行資料互傳

From Hsieh et al., Privacy-Preserving Contact Tracing with Adaptive Undeniability, 2022, CNS project at NTU

現今作法 1: BlueTrace in Singapore

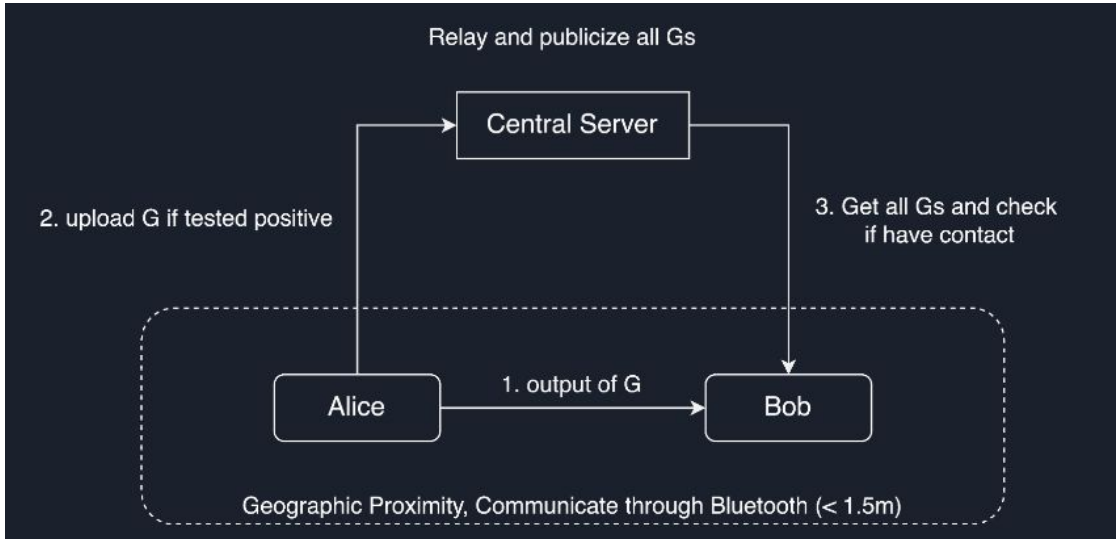


1. 中央系統分配每人一個ID
2. 密切接觸時交換ID
3. 確診者中獎時上傳從別人得來的ID給政府
4. 政府去找到那些ID的主人

中央集權，可以追蹤密接者，但是沒有隱私...

From Hsieh et al., Privacy-Preserving Contact Tracing with Adaptive Undeniability, 2022, CNS project at NTU

現今作法 2: Apple+Google in Taiwan



1. 每人自己產生 ID
2. 密切接觸時交換ID
3. 確診者中獎時上傳自己的ID給政府
4. 政府給大家確診者的ID，大家自行比對

沒有上傳到 central server, 密接者可裝死。

這個 Project 不僅沒有程式碼，也沒有標準答案

- 我們的方法是利用one-way function
  - 給定 x，很好計算 f(x)，但是給定 y 很難計算哪個 x 使得 f(x) = y
  - 小感覺：
    中央系統拿到“y=f(id)” 時，不會知道 id 對應到的是誰，比較有隱私性

- 重點是思考系統本身的特性
  - 在「隱私」與「可追蹤性」之間的取捨(trade-off)
  - 這個機制可以處理全國幾千萬人的傳送規模嗎？
- 這樣的思考層次是在程式碼之外的

https://reurl.cc/Z1OrM3

在程式碼之外

- 蓋房子不僅止於堆磚頭
  - 寫程式碼 <-> 堆磚頭
  - 設計演算法 <-> 建築工法
  - 系統設計分析 <-> 設計圖、災防分析、承重分析...

- 有些人努力堆磚頭
- 有些人努力研究工法
- 有些人擘畫出一棟棟建築
- 有些人致力於上述之間的連結和互動

這是電腦科學的匠人精神！！

這是大學畢業的我對於「電腦科學」的詮釋和理解