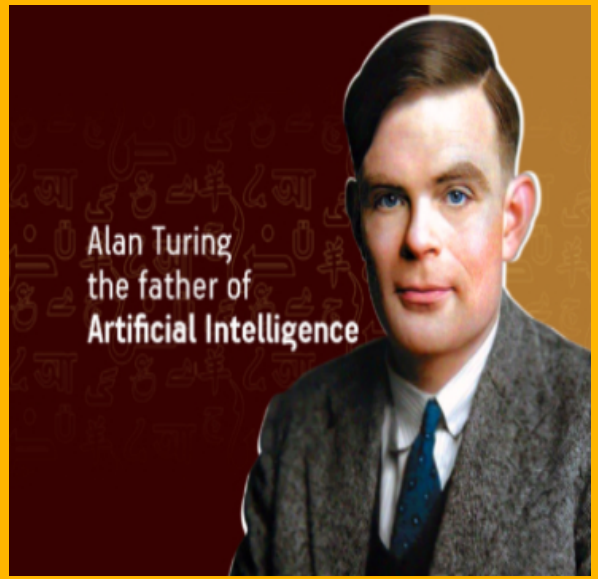


Introduction to Computer Science Fall 2022 #13

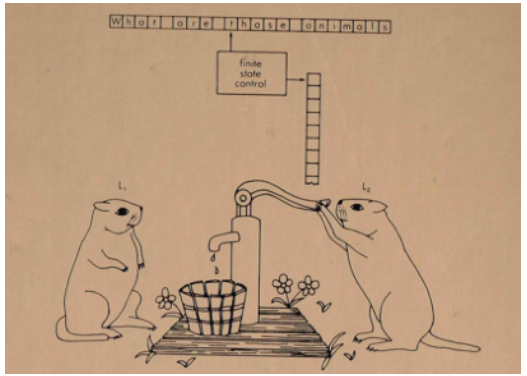
Chi-Jen Wu

Topics

- An Introduction to Computer Science
- The Shapes of Computers Today
- Computer Organization and Architecture
- Operating system
- Networking & The Internet
- Database Systems
- Software Engineering
- **Theory of Computation**
- Cloud Platform/Cloud Shell Editor (gcc/g++/Makefile)
- Cloud Platform/Cloud Source Repositories (git)
- C/C++ Programming

Theory of Computation

- **What is computation theory**
- **Functions and Their Computation**
- Universal Programming Languages
- **Turing Machines**
- **A Noncomputable Function**
- **Complexity of Problems**



What is computation theory

- What **can** and **can not** be computed
- Speed of such computations
  - **Time complexity**
- The amount of memory in use during such computations
  - **Space complexity**
- Programming language, complier and algorithm

Importance of Theory of computation

- Writing efficient algorithms that run in computing devices
- **Programming language** research and their development
  - 使程式設計更方便，更直覺(對人來說)
  - 使電腦看的**高階語法**！
- Efficient **compiler** design and construction
  - 編譯出更有效率的程式

Functions and Their Computation

- Function (program)
  - A correspondence between
    - A collection of possible **input values**
    - A collection of possible **output values**
- 程式和計算之間的關係
  - **可算否？**
  - 如何算才有效？
  - 怎樣才叫有效？

The problem being solved?

- **Computing** a function:
  - Determining the output value associated with a given set of input values
- **Noncomputable** function:
  - A function that cannot be computed by any algorithm
  - 質數有幾個？

The function that converts measurements in yards into meters

1 yards = 0.9144 meters

x yards = f_meter(*x*) {
x * 0.9144
}

Yards (input)	Meters (output)
1	0.9144
2	1.8288
3	2.7432
4	3.6576
5	4.5720
.	.
.	.
.	.

The problem can be solved!

有無限多個質數：無法計算

- 假設所有質數只有 n 個
  - P1、P2、P3…Pn
- P1 × P2 × P3 × … × Pn = P*
- P^ = P* + 1 = APi = BPi + 1,
- B = P1 × P2 × P3 × Pi-1 × Pi+1 × … × Pn
- P^ 無法被 n 個質數裡的任何一個整除
  - P^ 和 P* 互質, (A-B)Pi = 1, A-B = 1, Pi = 1
- 存在 n 個質數的**假設互相矛盾**
- **哥德巴赫猜想, 1742**
  - 大於2的任何偶數都可以表示為兩個質數之和？

歐幾里得

哥德巴赫猜想, 1742

- 大於2的任何偶數都可以表示為兩個質數之和？
  - $N = p^1 + p^2$
- **世界近代三大數學未解難題之一**
- 4 = 2 + 2
- 10 = 3 + 7 = 5 + 5
- 14 = 3 + 11 = 7 + 7
- 陳氏定理, 1966
  - 1+2 —> 62 = 7+5×11 = 43+19
  - $N = p^1 + p^2 * p^3$ $N = p^1 + p^2$

哥德巴赫

資工系 | 1 | 離散數學 Discrete Mathematics 詳細資料 Detail

寫一個程式去判斷哥德巴赫猜想是否為真！？**會怎樣？**

The history of solved problems

- 所有問題是否可以被計算機解決？
  - **不行，已被證明**
    - **不是所有問題都可以被計算機解決！！**
  - **Halting problem**
- 哪些是不可以被計算的？
  - 如何被歸納，證明**不可被計算**？
- 那些是可以被解決 (可計算的)？
  - 評估他們是不是**有效率的計算**？
  - 怎麼設計更有效率的算法！

Factors that influence program **efficiency**

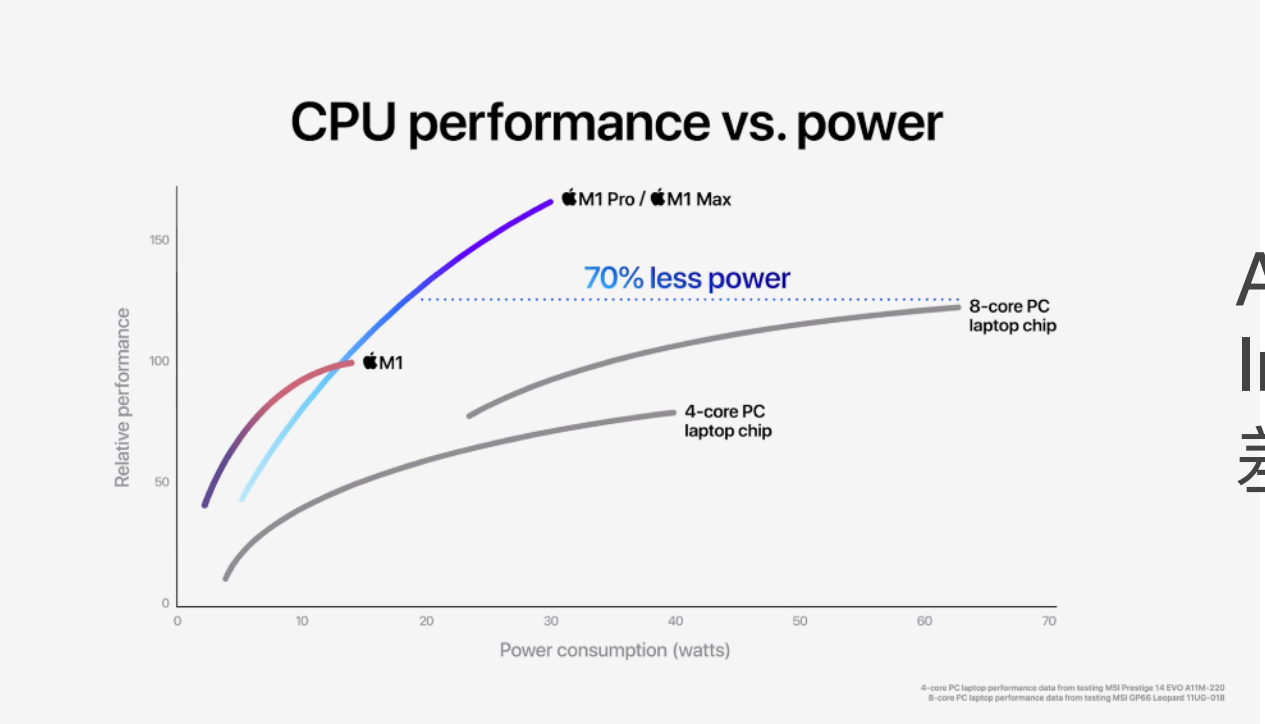
- The problem being solved
  - **確定是可以計算的，(可以在有限的時間計算出來)**
- The algorithm used to build the program
  - Quick sort, bubble sort
- Computer hardware
  - CPU, RAM
- Programming language used
  - python vs C/C++

What is the important factors that influence program efficiency?

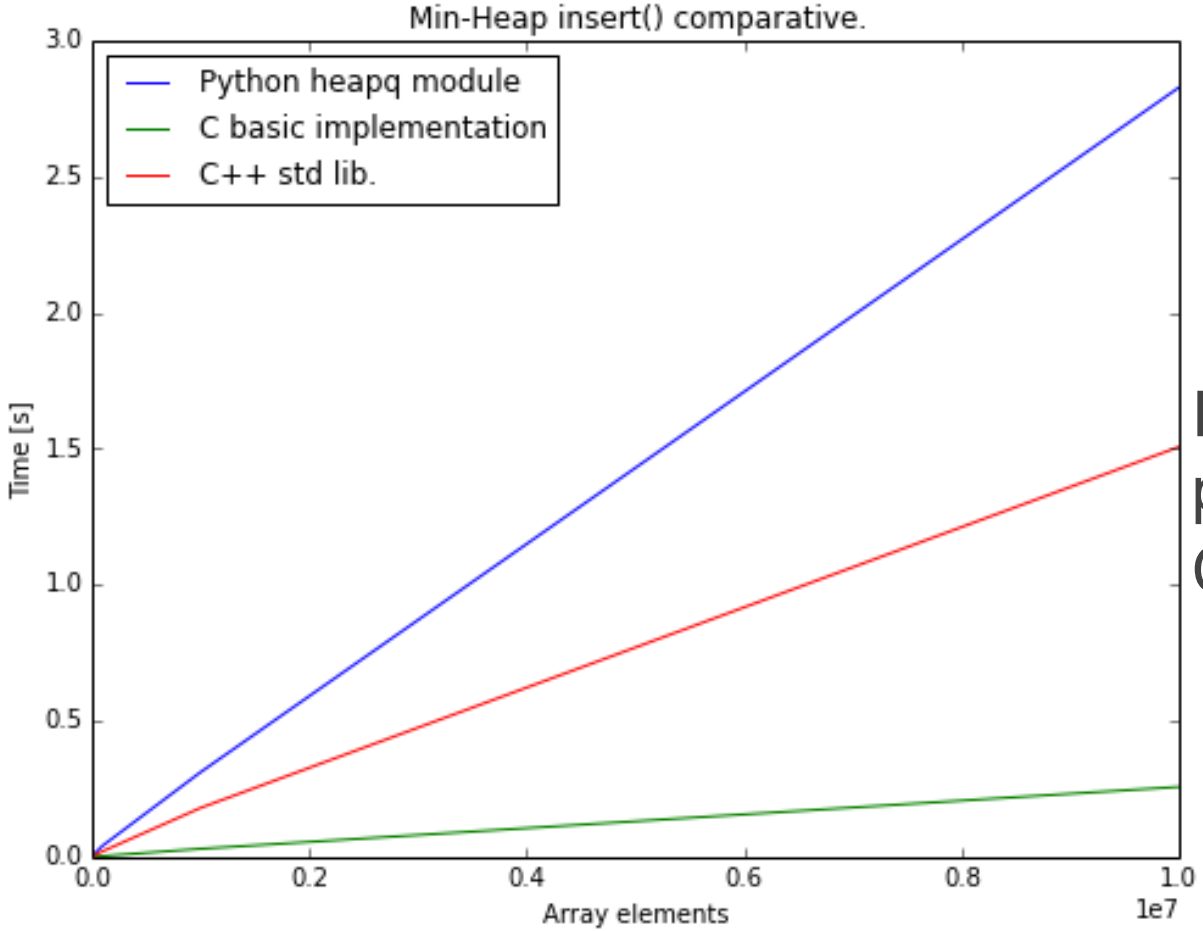
- The algorithm used to build the program
- Computer hardware
- Programming language used

- 問題？哪一個是最重要的因素？

Computer hardware



Apple M1 Pro vs Intel performance 差異！

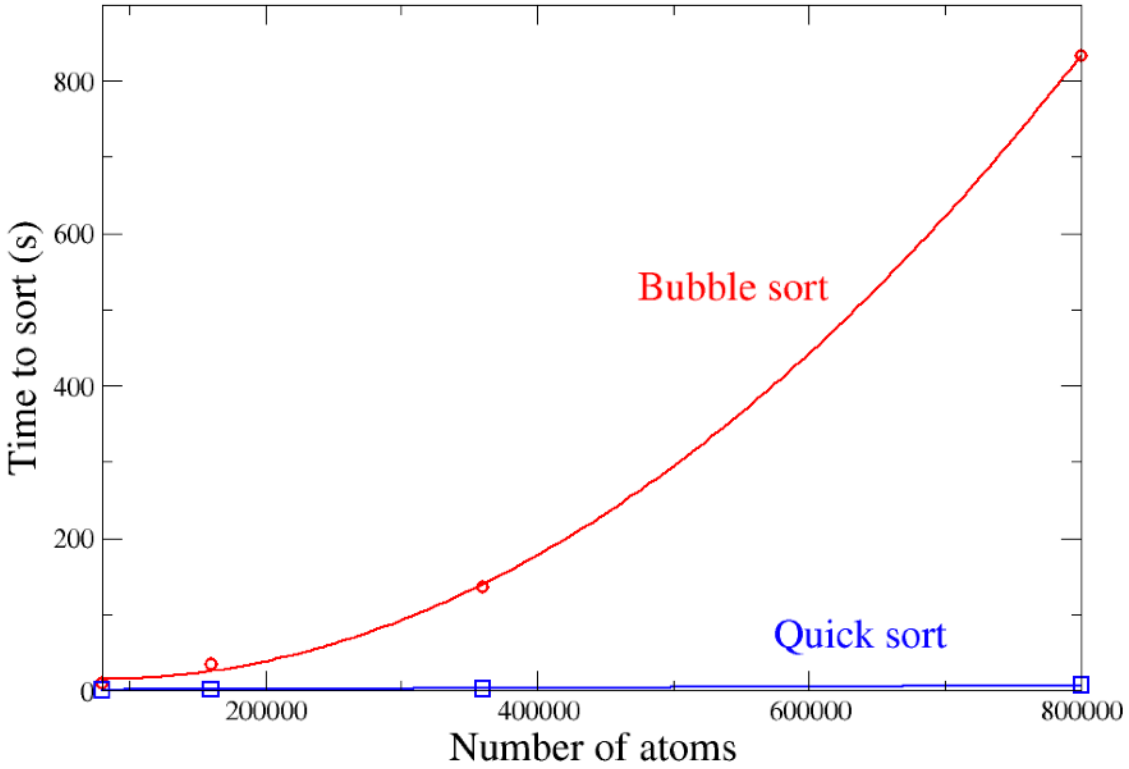


- **Programming language**

Python vs C/C++ performance

Compiler 差異！？

algorithm

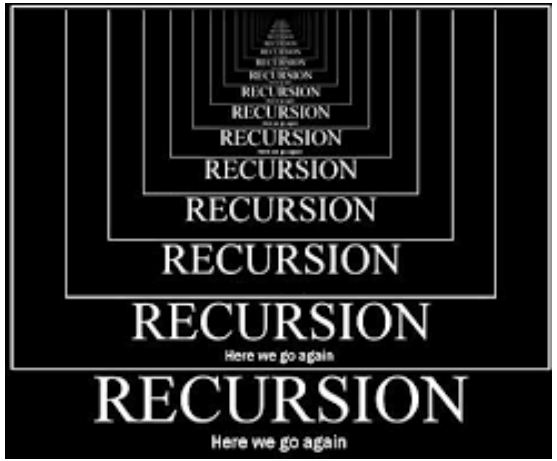


Quick sort, bubble sort performance 差異！？

怎麼差那麼多？

What is the important factors that influence program efficiency?

- The algorithm used to build the program
- Computer hardware
- Programming language used
- 問題？哪一個是最重要的因素？
  - 都滿重要的，看你從哪一個角度去看！
  - 其實隨著科技進展，重要因素也會改變
    - 量子電腦的出現！
    - 程式語言遞迴的出現！
    - 演算法的改進！

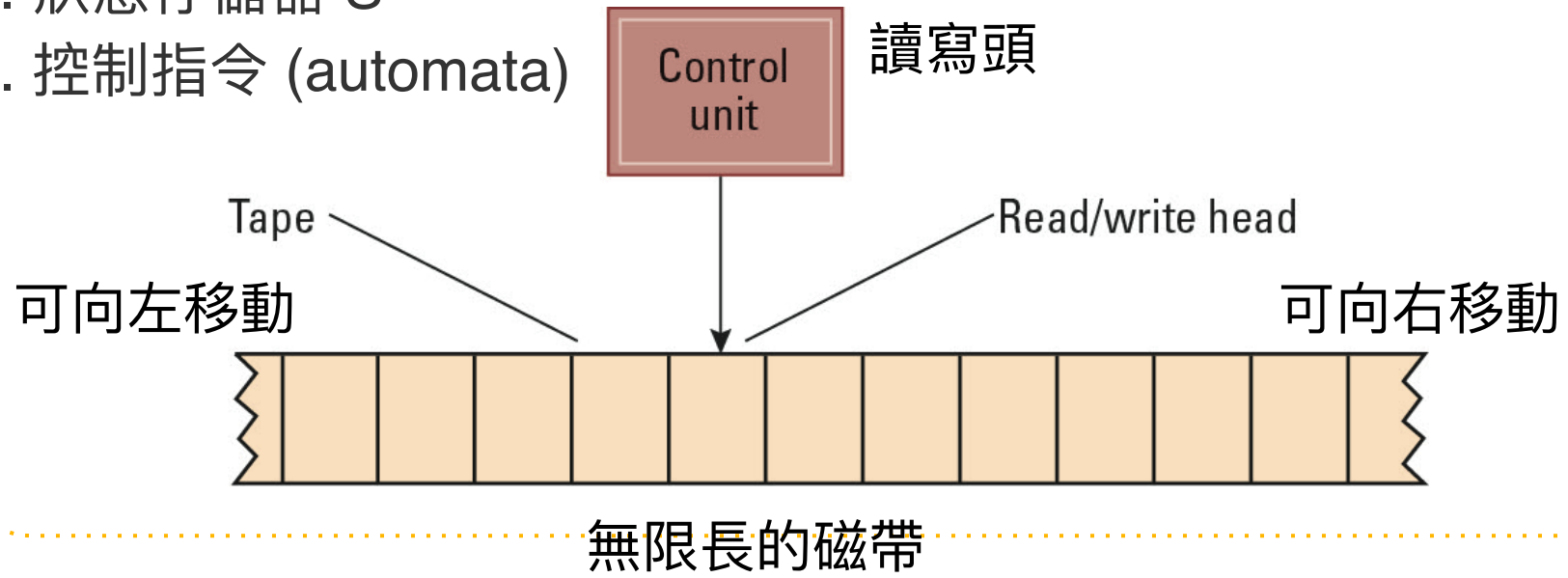


Turing Machines

- Theoretical computing machines
  - Alan Turing,1936
- Used as a tool for studying the power of algorithmic processes
- 一個最基礎的計算機器
  - 但是可以執行所有電腦可以運算的程式
  - 和原子和電子類似，組成萬物

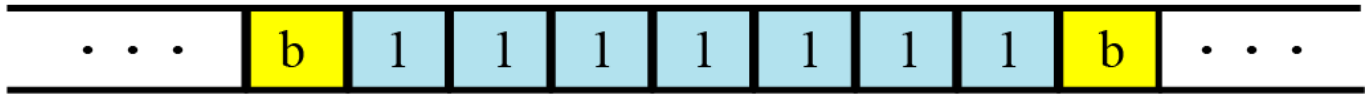
The components of a Turing machine

1. 讀寫頭
2. 磁帶
3. 狀態存儲器 S
4. 控制指令 (automata)



Turing Machine: Tape

- Modern computers use a random-access storage device with **finite** capacity
- Turing machine's memory is **infinite**
- The tape holds a sequence of characters
  - only two symbols:
    - **a blank (b or 0)**
    - **digit 1**



Tape

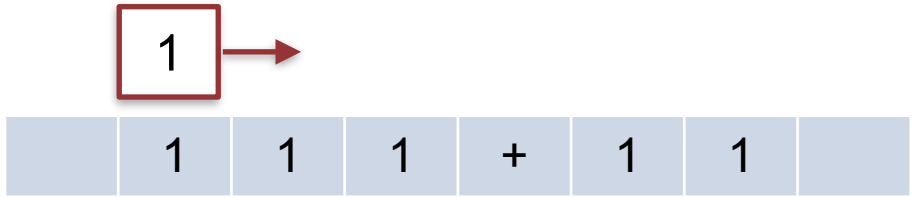
Turing machines 想像

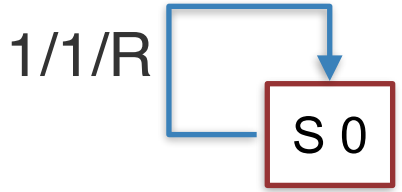
- 可以想像成指令的機器
  - 在紙上寫上或擦除某個符號；
    - incr(X), decr(X)
  - 把讀寫頭從紙的一處移動到另一處
    - 可以向左移動 X 步
    - 可以向右移動 X 步
    - Loop(X)

Turing Machine 第一個例子

- 3 + 2
  - 111 + 11

不是二進位喔

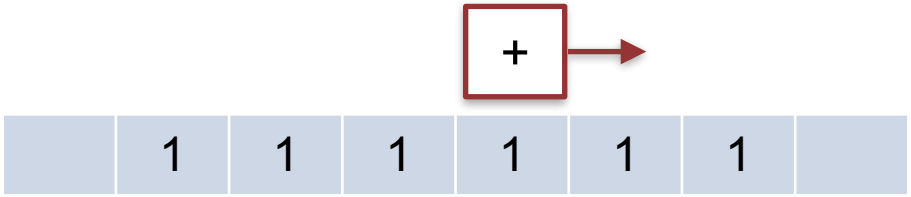




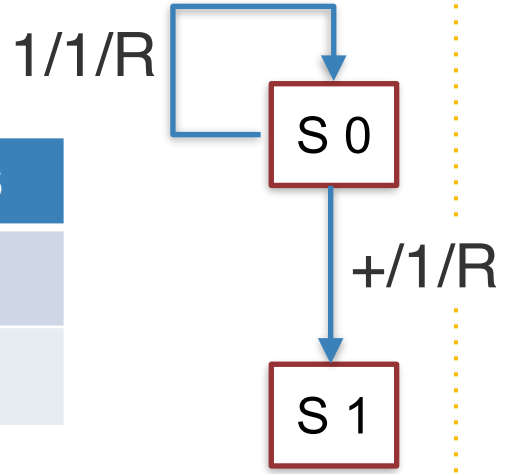
	Read	Op	Move	Next S
S 1	+	1	Move R 1	S 1

初始位址：最左邊不是空白的位址

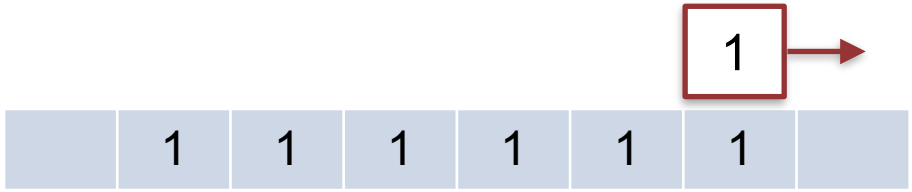
3+2 : 111+11

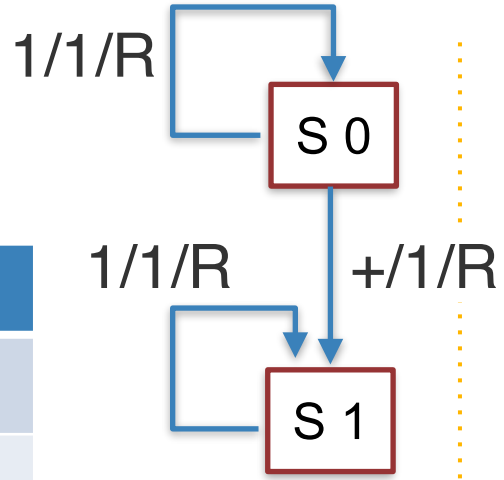


	Read	Op	Move	Next S
S 0	1	1	Move R 1	S 0
S 0	+	1	Move R 1	S 1

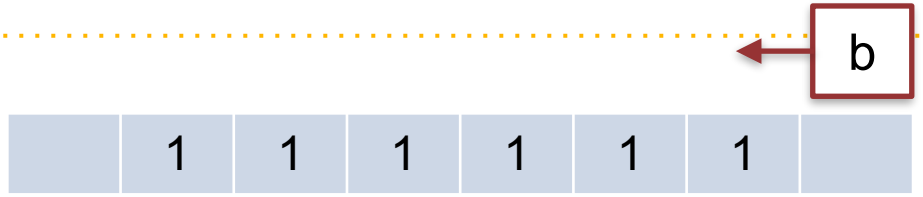


3+2 : 111+11

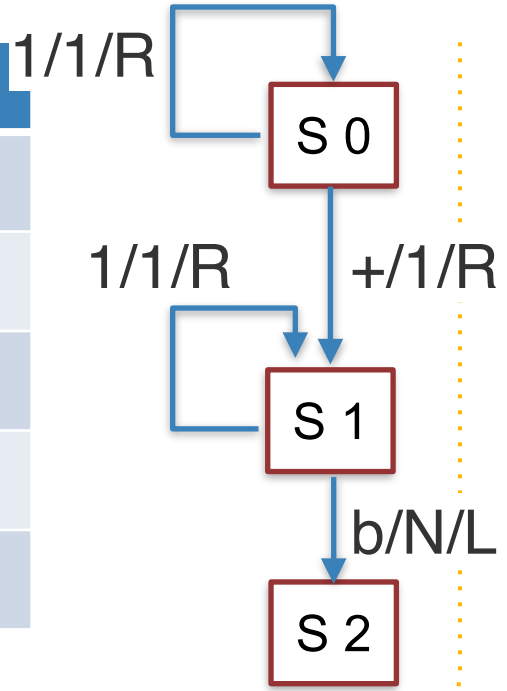


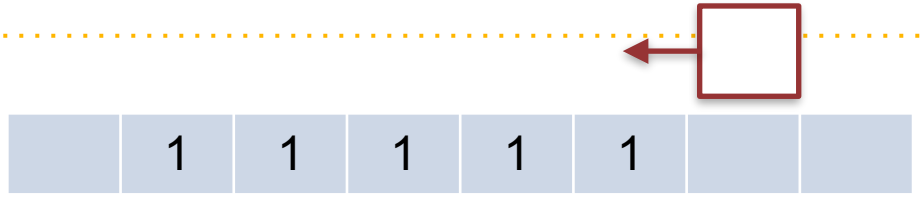


	Read	Op	Move	Next S
S 0	1	1	Move R 1	S 0
S 0	+	1	Move R 1	S 1
S 1	1	1	Move R 1	S 1

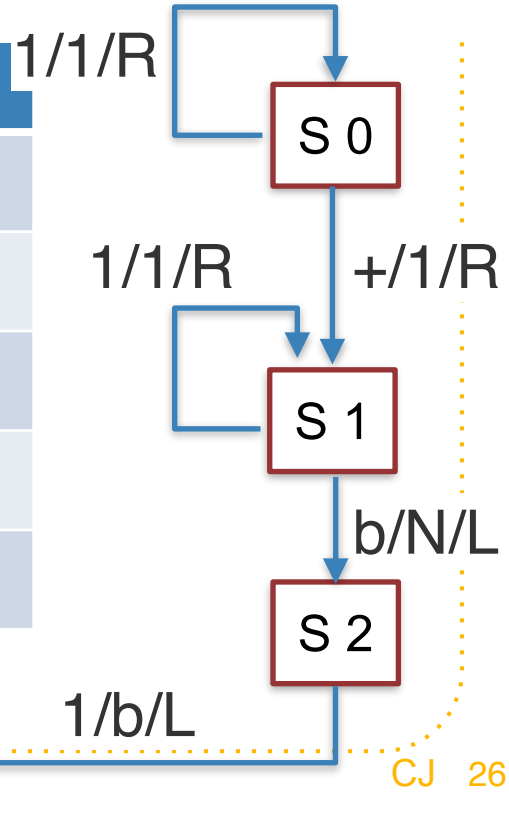


	Read	Op	Move	Next S
S 0	1	1	Move R 1	S 0
S 0	+	1	Move R 1	S 1
S 1	1	1	Move R 1	S 1
S 1	b	N	Move L 1	S 2
S 2	1	b	Move L 1	S stop

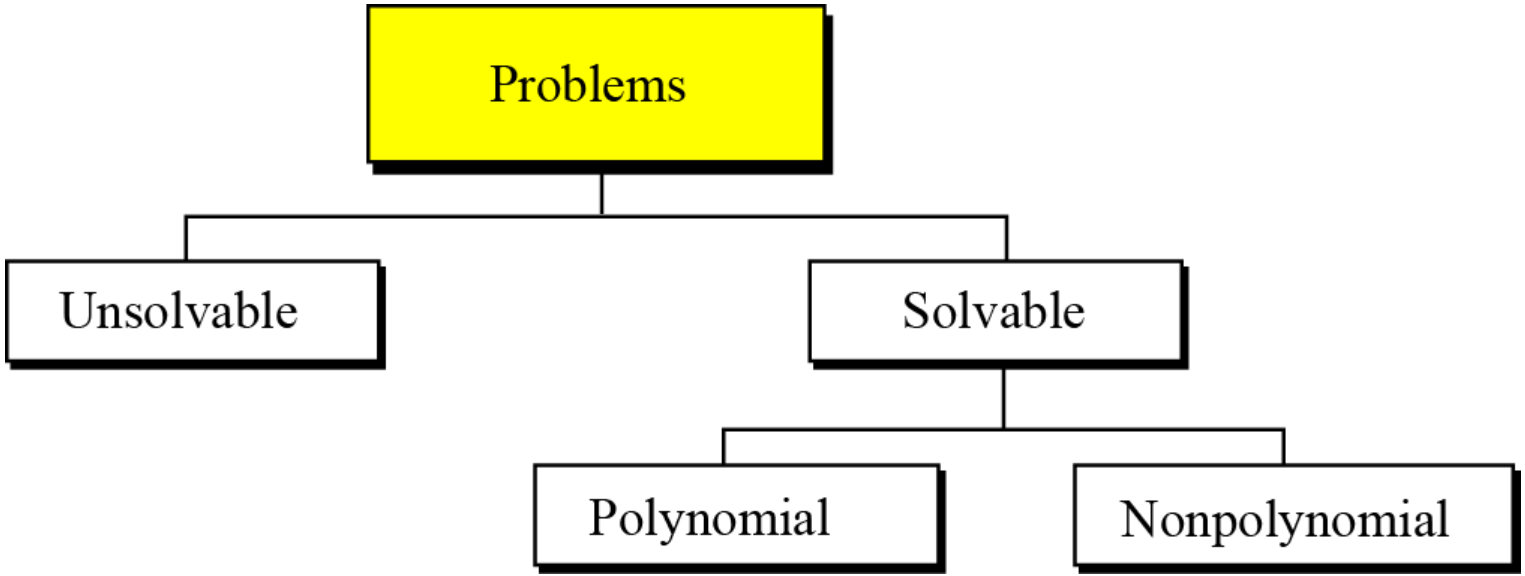




	Read	Op	Move	Next S
S 0	1	1	Move R 1	S 0
S 0	+	1	Move R 1	S 1
S 1	1	1	Move R 1	S 1
S 1	b	N	Move L 1	S 2
S 2	1	b	Move L 1	S stop



Complexity of Problems



A Non computable Function

- The Halting Problem
  - 停機問題
  - 寫一個找哥德巴赫猜想的程式
    - N = p1 + p2，loop forever?!
- 忙碌的海狸
  - 增長的速度太快，快到無法計算
  - 到底是**算不完**還是**無法算**？
- 對電腦來說，**無法分辨算不完和無法算**，因為他會一直算！

The Halting Problem

- Given the **encoded version** of any program
- 1 if the program is self-terminating
- 0 if the program is not

```
{
    計算 pi 的值
}
```

問題：可以寫一個程式判斷這會停嗎？

Solvable problems

- To know how long it takes for the computer to solve that problem
- How complex is the program?
  - The complexity of a program can be measured
    - Computing time (**Time complexity**)
    - Memory space (**Space complexity**)

Big-O notation

- This **simplification** of efficiency
- The number of operations given as a **function inputs**
- O(n) means a program
  - **n operations** for **n inputs**,
- O(n^2) means a program
  - **n^2 operations** for **n inputs**

Complexity of solvable problems

- To measure the complexity of a solvable problem is to find the number of operations executed by the computer
- Three different programs to solve the same problem
  - The first one has a complexity of $O(\log_{10} n)$
  - The second $O(n)$
  - The third $O(n^2)$

問題：哪一個程式跑的比較快？

- Assuming 1 million inputs
- How long does it take to execute each of these programs
  - Executes one instruction in 1 microsecond
  - 1 million instructions per second?

1st program:	n = 1,000,000	$O(\log_{10} n) \rightarrow 6$	Time → 6 μs
2nd program:	n = 1,000,000	$O(n) \rightarrow 1{,}000{,}000$	Time → 1 sec
3rd program:	n = 1,000,000	$O(n^2) \rightarrow 10^{12}$	Time → 277 h

要先確定是在同一台機器，同一個程式語言！

Complexity of Problems

- Time Complexity
  - The number of **instruction** executions required
- 大部分 **complexity** means **time complexity**
- Space Complexity
  - The number of **memory space** required when the program executing

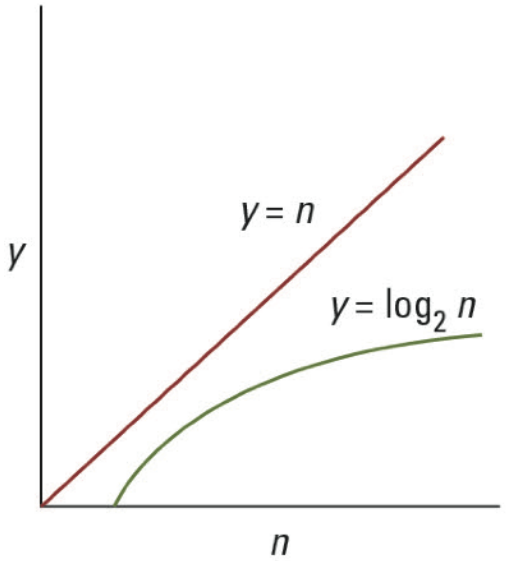
Polynomial problems

- A program has a complexity of
  - O(logn), O(n), O(n^2), O(n^3), O(n^4), or O(**$n^k+n^4+n^3+n^2+n$**) where **k** is a **constant**
- 當下的電腦科技
  - We can get solutions to **polynomial problems** with a **reasonable** number of inputs

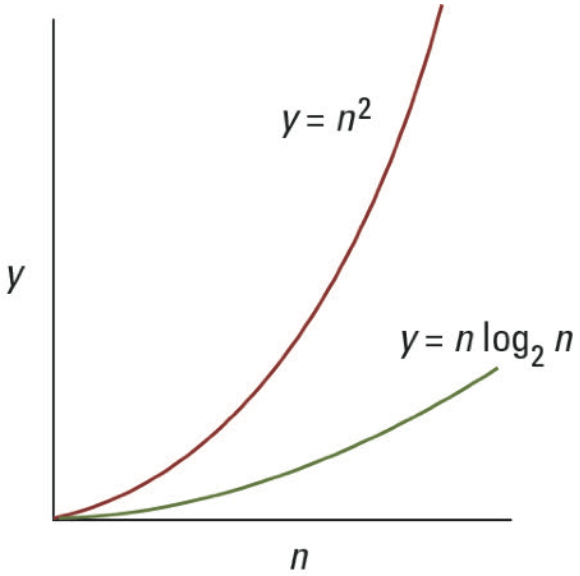
Non-polynomial problems

- A program has a complexity that is greater than a polynomial
  - $O(2^n)$, $O(10^n)$ or $O(n!)$
- it can be solved if the number of inputs is very small
  - such as fewer than 100
  - If the number of inputs is large
    - 要等好幾天，但是還是跑的出來！

Graphs of the mathematical expressions n, $\log_2 n$, $n \log_2 n$, and n^2



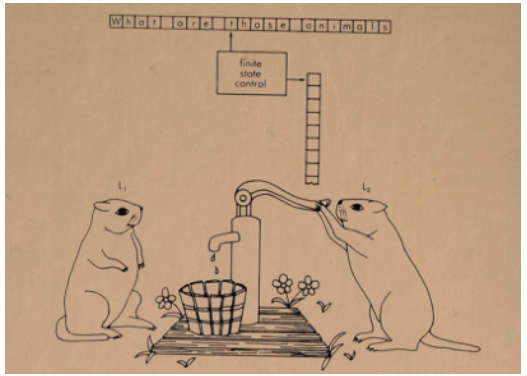
a. n versus $\log_2 n$



b. n^2 versus $n \log_2 n$

Conclusion

- Functions and Their Computation
- Universal Programming Languages
- Turing Machines
- A Noncomputable Function
- Complexity of Problems



Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw