

資料結構 la

Petingo

Outline

- 有 6 人投票說想聽資料結構
- 因為大家 4 最佳 py 黨
- 預設大家對 python 有基本的知識

Outline

- 資料結構是三小
- 為什麼要學這ㄍ
- 記憶體
- Array(陣列)
- Linked list(鏈結串列)
- Queue(佇列)
- Stack(堆疊)
- Tree(樹)
- Graph(圖)

為什麼要學資料結構？

- Google 面試這次惹爭議了：雖然我們公司 90%的工程師都用你開發的工具，但我們還是不聘用你
- https://www.bnext.com.tw/ext_rss/view/id/758790

為什麼要學資料結構？

- 這個故事告訴我們兩件事：
 1. 就算你的寫出很厲害的程式，不會資結還是進不了 Google

為什麼要學資料結構？

- 這個故事告訴我們兩件事：
 1. 就算你的寫出很厲害的程式，不會資結還是進不了 Google
 2. 既使你不會複雜的資料結構你一樣可以寫出改變世界ㄉ程式

先來看看這《

- 陳鍾誠——用十分鐘學會《資料結構、演算法和計算理論》
- <https://www.slideshare.net/ccckmit/ss-56891871/1>

首先看看《資料結構》

- 顧名思義

- 就是學習如何安排《程式》

- 所需要用到《資料》

- 的《結構》

同樣的、對於資料結構

- 你只要學會下列三種結構
 - 鏈結串列
 - 二元樹
 - 雜湊表
- 就差不多夠用了！

Array

Array

- Python 的 List 就是一種比較厲害的 Array
- [10, 20, 30, 40]
- 可以說是資料結構裡面最基本的單位

先來講講記憶體

65	80	80	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	1	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

Array

65	80	80	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	1	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

A[0] = (0x18 + 0) 裡面的東西 = 10

A[1] = (0x18 + 1) 裡面的東西 = 20

...

Linked List

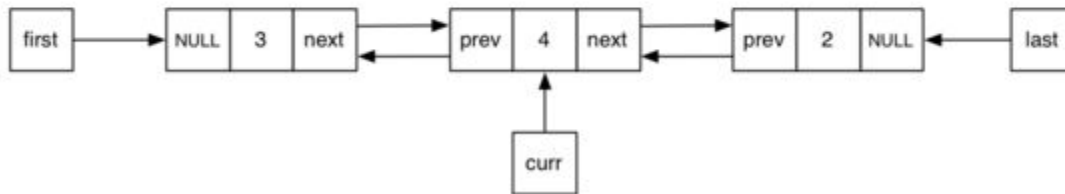
Link List

- 跟陣列一樣，就是一大串東西

Singly-linked List



Doubly-linked List



Link List

10	80	30	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	40	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

first: 要找 A 的話要從 0x00 開始找喔！

(0x00): A 的這一項是 10, 下一項在 0x19

(0x19): A 的這一項是 20, 下一項在 0x02

(0x12): A 的這一項是 30, 下一項在 0x13

(0x13): A 的這一項是 40, 已經沒有下一項了！

Link List

- 既然都是一大串，用 Array 不就好了嗎？
- 有時候陣列沒有辦法滿足我們的需求
- 頻繁插入、刪除的時候

假設今天要把 20 刪掉

- 原本 $A = [10, 20, 30, 40]$
 - $A[0] = 10$
 - $A[1] = 20$
 - $A[2] = 30$
 - $A[3] = 40$
- 刪掉 20 之後中間不能空掉ㄅ
 - $A[0] = 10$
 - $A[1] = 30$
 - $A[2] = 40$

如果是一般的 Array 的話

- 我要把 20 後面的全部往前挪一格

65	80	80	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	1	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

如果是一般的 Array 的話

- 我要把 20 後面的全部往前挪一格

10	80	30	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	40	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	30	40	?	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

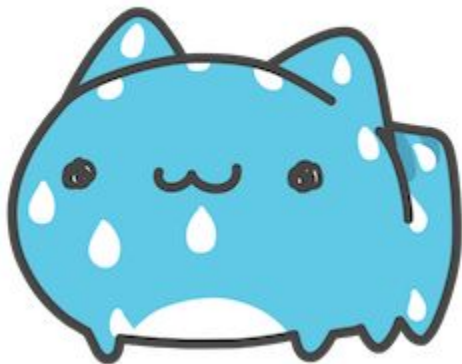
如果我的 Array 很長很長，我又要一直刪東西



貓貓貓貓蟲咖波

如果我的 Array 很長很長，我又要一直刪東西

- 假設 $\text{len}(A) = 100000$
- 我要刪掉當中的 10 個東西
- 電腦大概就需要跑 $100,000 * 10 = 1,000,000$ 個迴圈



相較之下 Link List

10	80	30	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	40	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

first: 要找 A 的話要從 0x00 開始找喔！

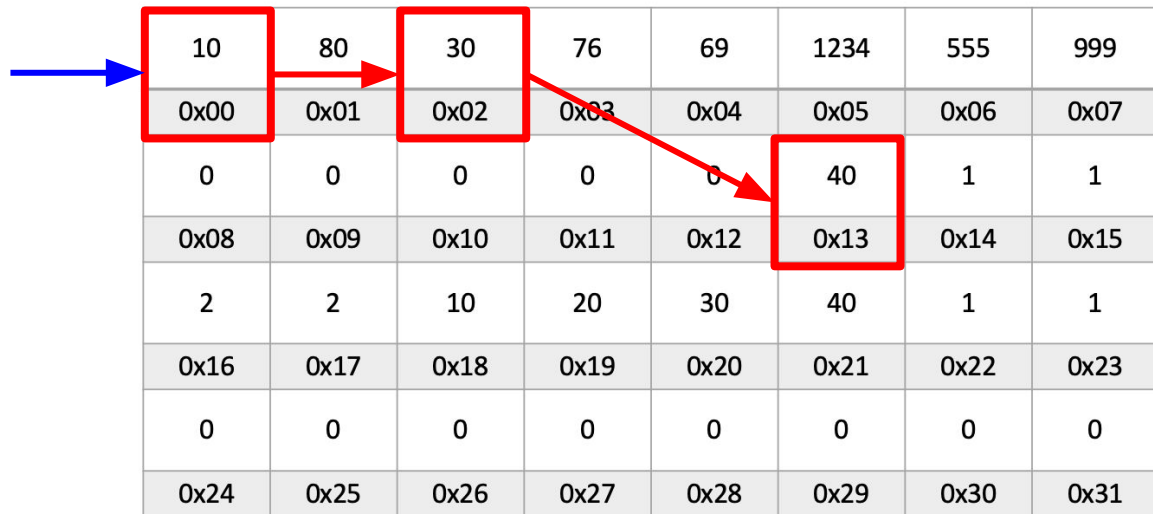
(0x00): A 的這一項是 10, 下一項在 0x19

(0x19): A 的這一項是 20, 下一項在 0x02

(0x12): A 的這一項是 30, 下一項在 0x13

(0x13): A 的這一項是 40, 已經沒有下一項ㄌ！

相較之下 Link List



10	80	30	76	69	1234	555	999
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0	0	0	0	0	40	1	1
0x08	0x09	0x10	0x11	0x12	0x13	0x14	0x15
2	2	10	20	30	40	1	1
0x16	0x17	0x18	0x19	0x20	0x21	0x22	0x23
0	0	0	0	0	0	0	0
0x24	0x25	0x26	0x27	0x28	0x29	0x30	0x31

first: 要找 A 的話要從 0x00 開始找喔！

(0x00): A 的這一項是 10, 下一項在 ~~0x19~~ 0x02

(0x12): A 的這一項是 30, 下一項在 0x13

(0x13): A 的這一項是 40, 已經沒有下一項 ㄌ！

相較之下 Link List

- 不管整條 List 多長, 只要操作一次就可以刪掉一個元素

既然這樣，大家都用 Linked List 就好啦？

- Linked List 沒辦法像 Array 一樣，一次就拿到第 n 項
- 必須從頭一個一個找

所以說

- Array 可以快速的對第 n 項進行操作
 - 但如果要插入元素或是刪除元素，會比較慢
-
- Linked List 可以快速的進行循序的取值賦值、新增或刪除元素
 - 但沒辦法快速的拿到第 n 項

Hash Table

神奇ㄉ陣列！

- Python 的 Dictionary 就是一種 Hash Table 的應用
- {cat: “meow”, dog: “woof”}

我們知道

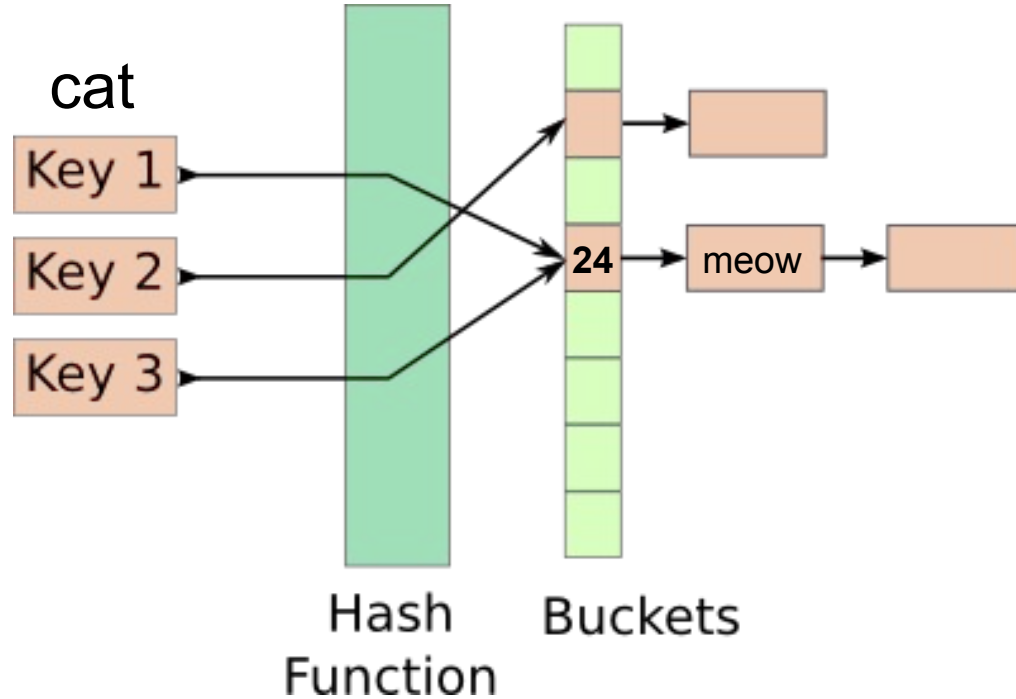
- 陣列只能用數字當 index
 - 例如 `sound = ["meow", "woof", "AAAAAA"]`
 - `sound[0] = meow`
 - `sound[1] = woof`
 - `sound[2] = AAAAAA`
-
- `Sound["cat"] ???`

Sound[“cat”] = “meow”

- 那就把 cat 當作是一個數字就好ㄌ呀！
- $c = 3, a = 1, t = 20$
- $3 + 1 + 20 = 24$
- $\text{sound}[24] = \text{“meow”}$

Hash Table

$$3 + 1 + 20 = 24$$



但是但是

- cat、act、tac 加起來不是都是 24 嗎？



碰撞

- 這個撞在一起的行為叫做碰撞
- 選比較好的 Hash Function
- 當碰撞發生時需要妥善處理

補充

<http://alrightchiu.github.io/SecondRound/hash-tableintrojian-jie.html>

Queue

Queue

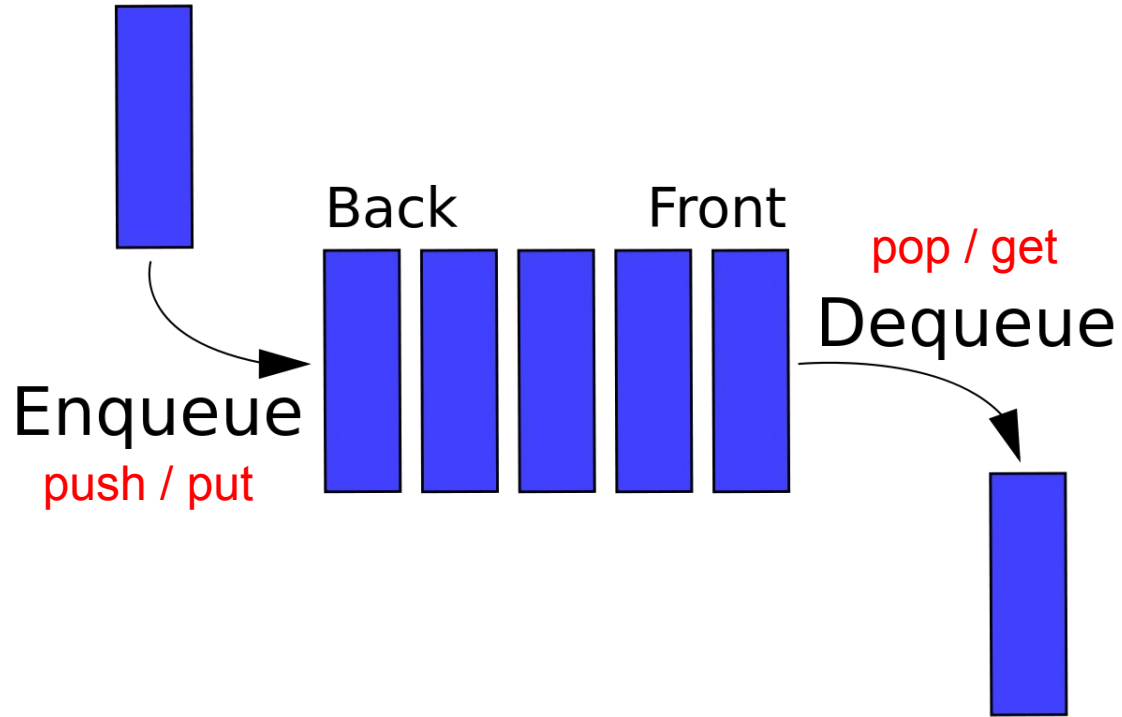
- 排隊
- 先來的先處理



Queue

- 印表機排程
- 台鐵訂票系統
- 硬碟寫入

Queue



Queue - practice

- 我找不到什麼應用 QQ

```
import queue
q = queue.Queue()

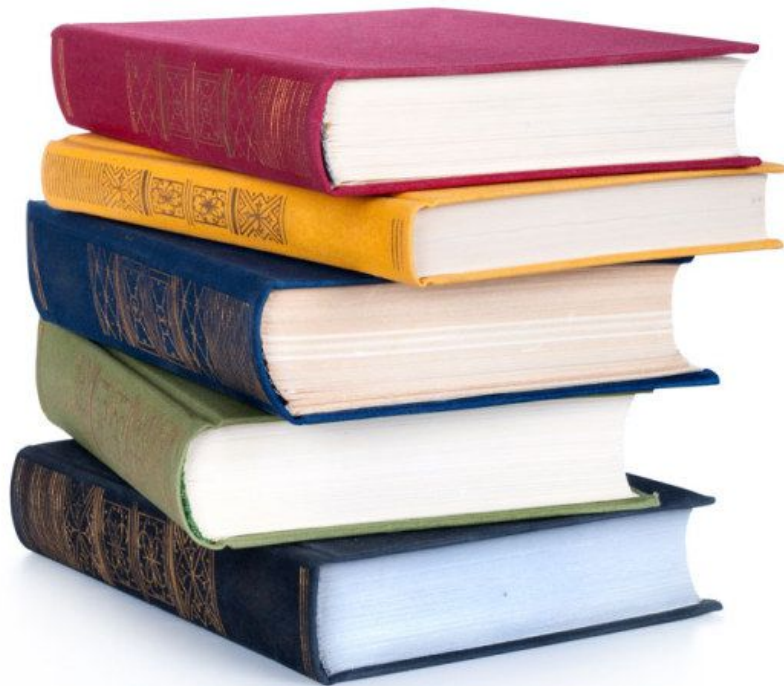
q.put(1)
q.put(2)
q.put(3)

print(q.get())
>> 1
```


Stack

Stack

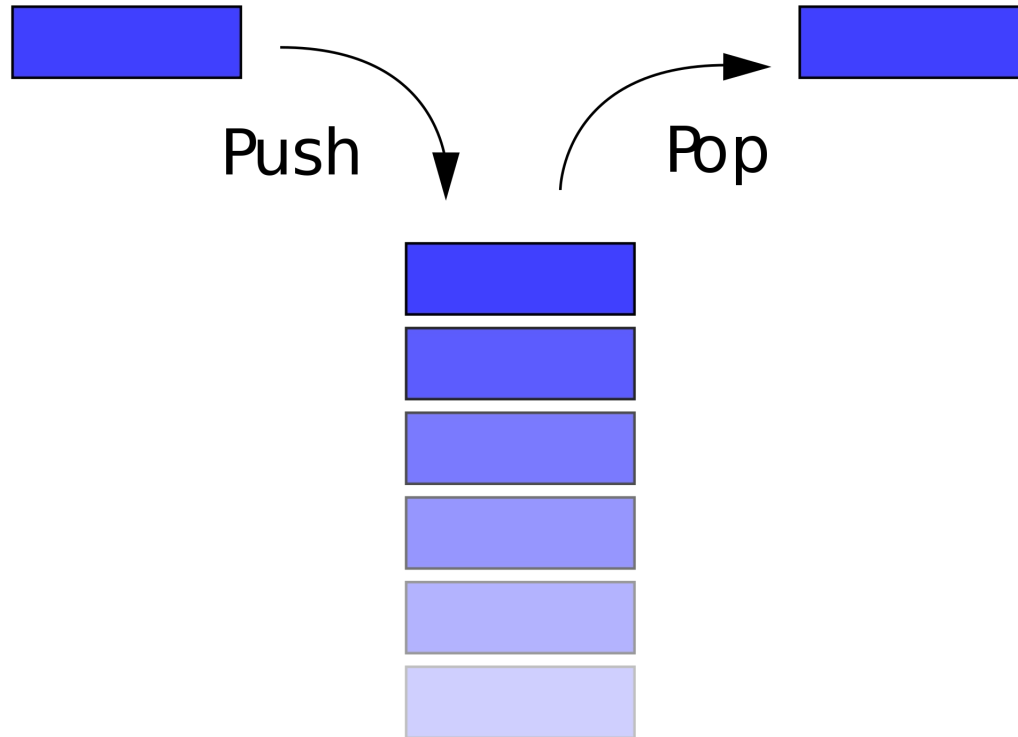
- 腔腸動物門
- 品客
- 一疊書
- 越晚放的東西越早拿出來



Stack

- 電腦內部構造常常用到
- 函數呼叫
- 後序運算

Stack



Stack - practice

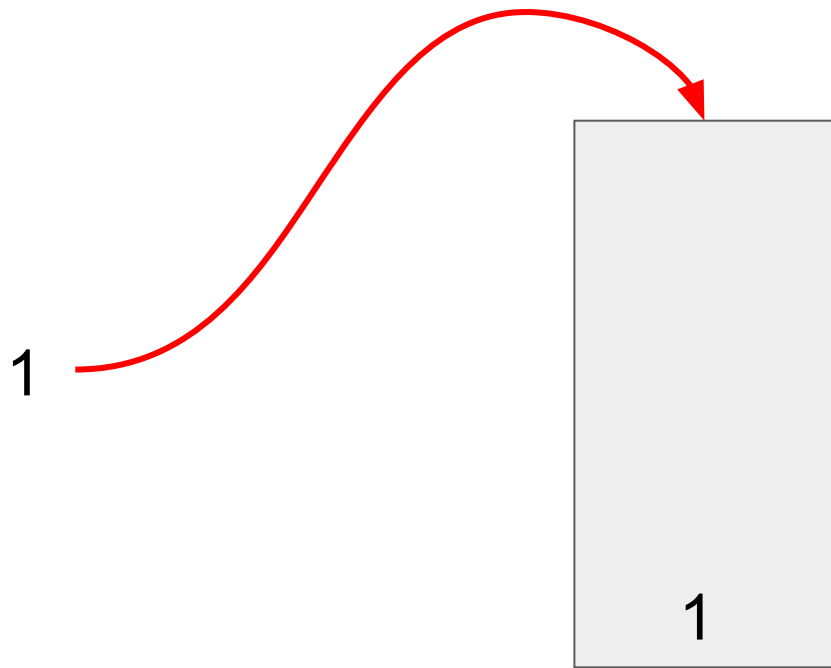
- 電腦在進行四則運算的時候，會把數字轉成「後序式」在進行操作
 - 所謂後序式就是 $+$ $-$ $*$ $/$ 在後面的表示法
 - Ex. $1 + 2 * 3$ 的後序式是 $1\ 2\ 3\ *\ +$
-
- 這個要怎麼轉換有點複雜，我們來計算就好了！

Stack - practice

- 運算規則：
- 遇到數字的話，就把數字 push 進 stack 的頂端
- 遇到 $+$ $-$ $*$ $/$ 的時候，把 stack 最上面的兩個數字拿出來運算
- 然後再把結果丟進 stack 的頂端

例子

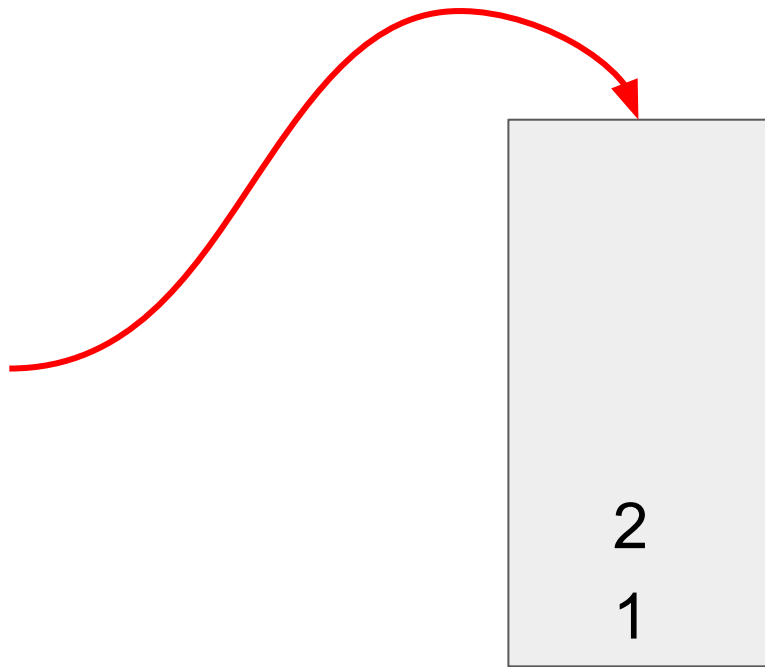
1 2 3 * +



例子

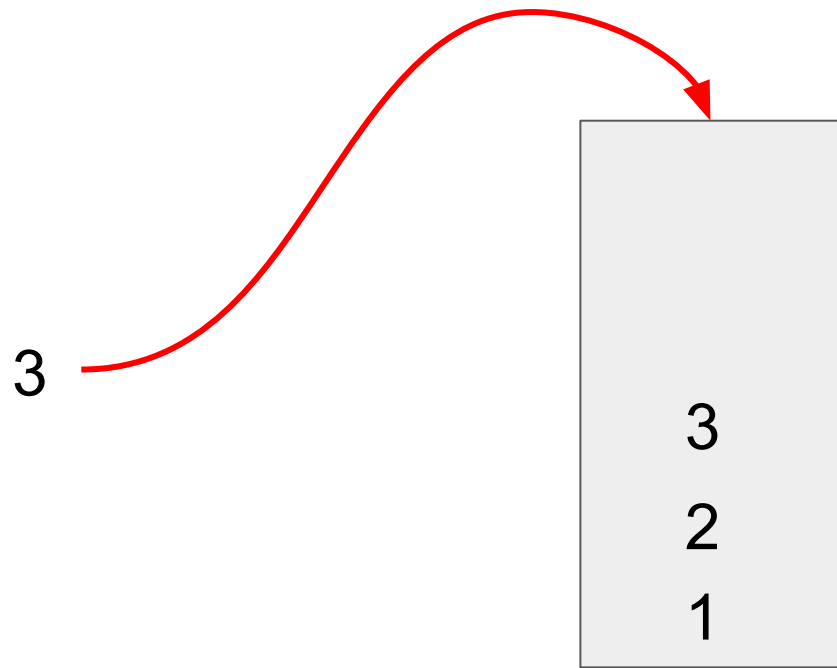
1 2 3 * +

2



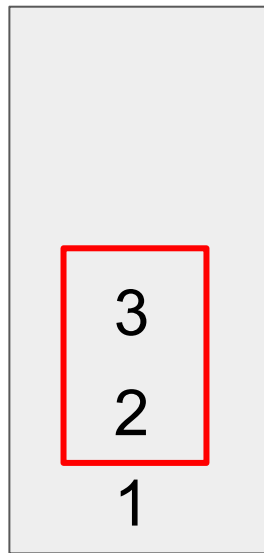
例子

1 2 3 * +



例子

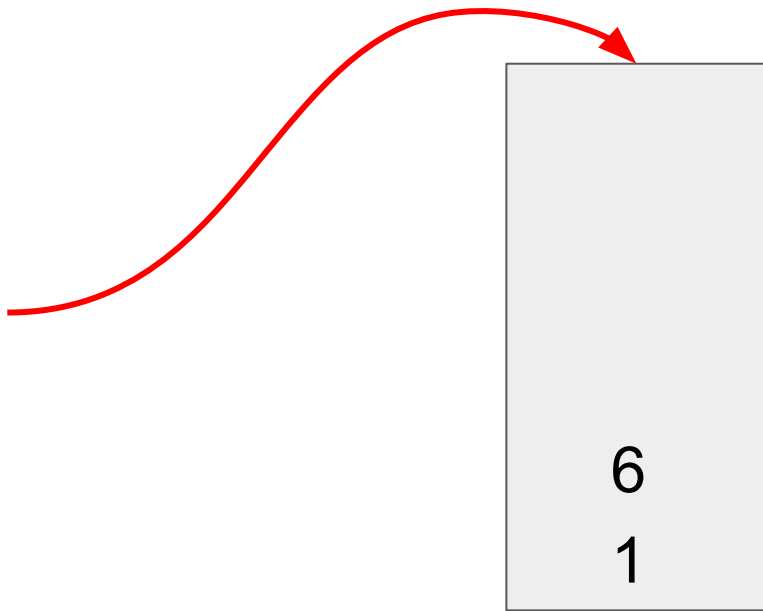
1 2 3 * +



例子

1 2 3 * +

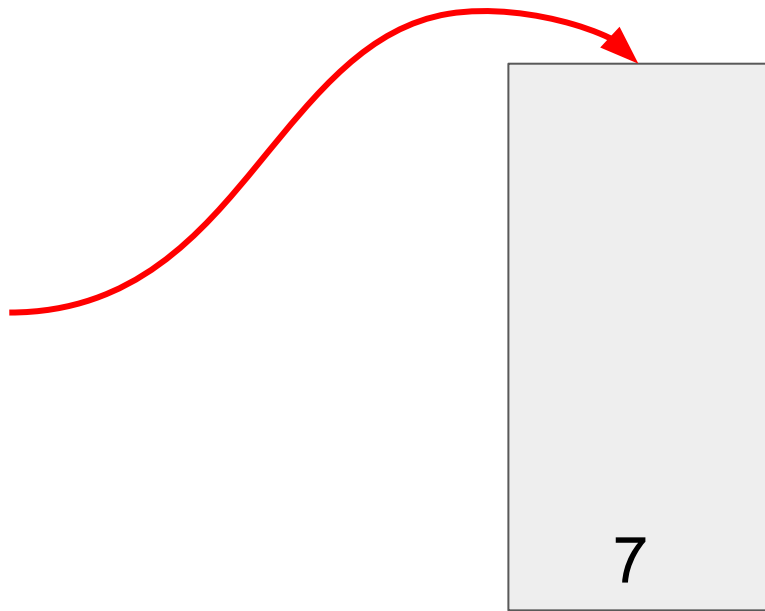
2 * 3 = 6



例子

1 2 3 * +

6 + 1 = 7



來 44 看ㄅ

- 我們就算 * 跟 + 就好ㄅ
- Python 官方說如果要寫 stack 的話用 list 就可以ㄅ

- 1 2 3 * + (Ans: 7)
- 3 3 + 4 * (Ans: 24)
- 1 1 1 1 + + + (Ans: 4)

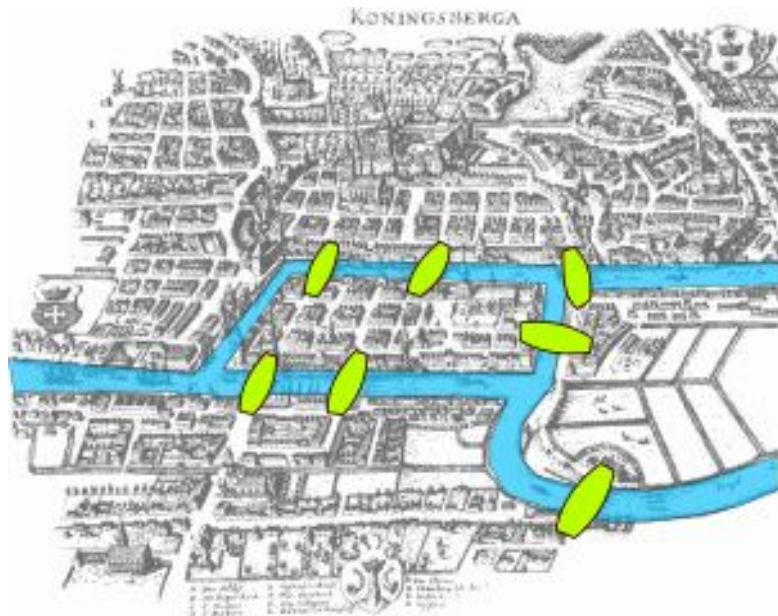
```
stack = []  
stack.append(1)  
stack.append(2)  
stack.append(3)  
  
print(stack.pop())  
>> 3  
print(stack.pop())  
>> 2
```

Graph

柯尼斯堡七橋問題

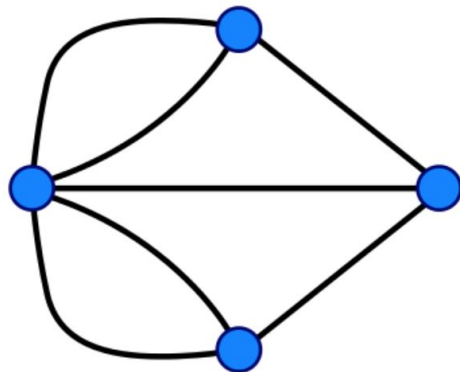
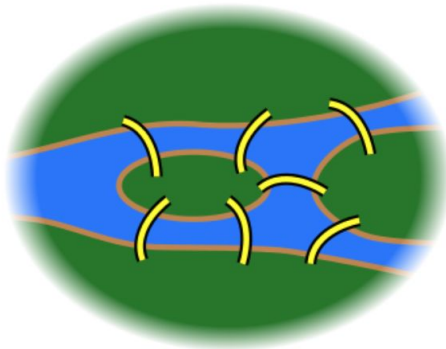
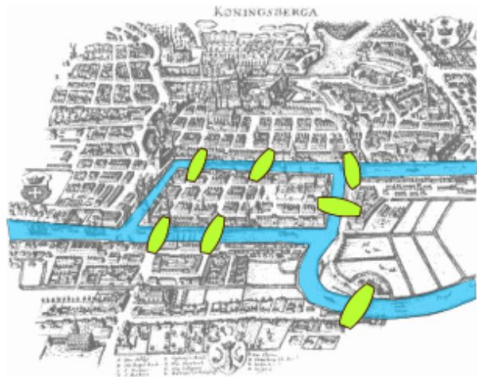
東普魯士柯尼斯堡(今俄羅斯加里寧格勒)市區跨普列戈利亞河兩岸，河中心有兩個小島。

小島與河的兩岸有七條橋連接。在所有橋都只能走一遍的前提下，如何才能把這個地方所有的橋都走遍？

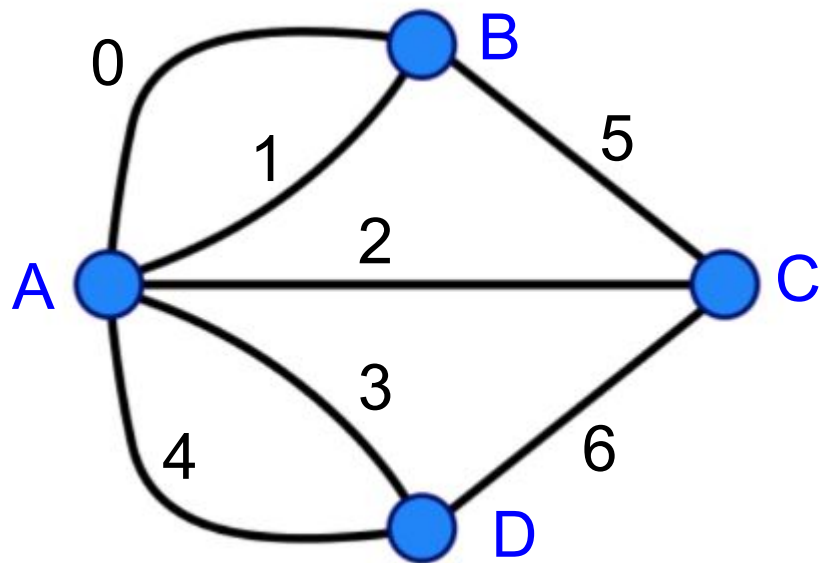


先把問題抽象化

- 把陸地視為一個點，橋當成邊



要怎麼把這東東存在電腦裡面ㄟ？



存邊

- edge [0] = (A, B)
- edge [1] = (A, B)
- edge [2] = (A, C)
- edge [3] = (A, D)
- ...
- edge [6] = (C, D)

存邊

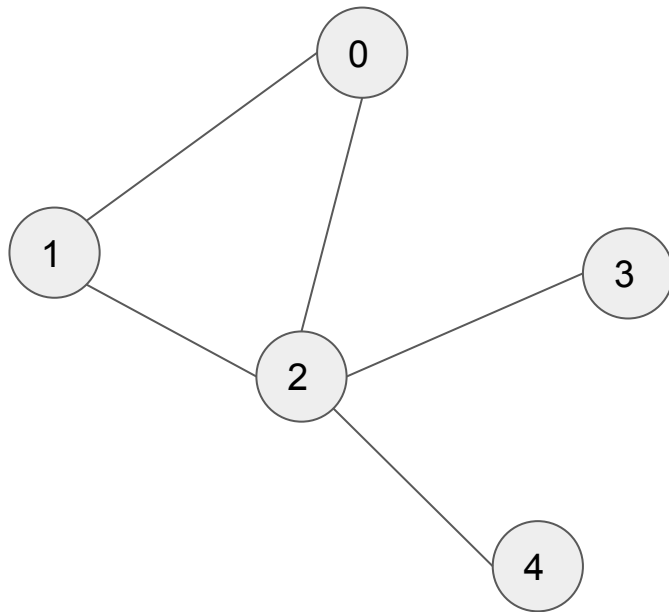
- $\text{edge}[\text{"A"}] = [(0, \text{B}), (1, \text{B}), (2, \text{C}), (3, \text{D}), (4, \text{D})]$
- $\text{edge}[\text{"B"}] = [(0, \text{A}), (1, \text{A}), (5, \text{C})]$
- ...

存一張表

	A	B	C	D
A	[]	[0, 1]	[2]	[3, 4]
B	[0, 1]	[]	[5]	[]
C	[2]	[5]	[]	[6]
D	[3]	[]	[6]	[]

Graph - practice

- 來試試看存個簡單一點的圖
- 兩個點中間最多只會有一個邊, 邊有「長度」
- 印出所有的邊

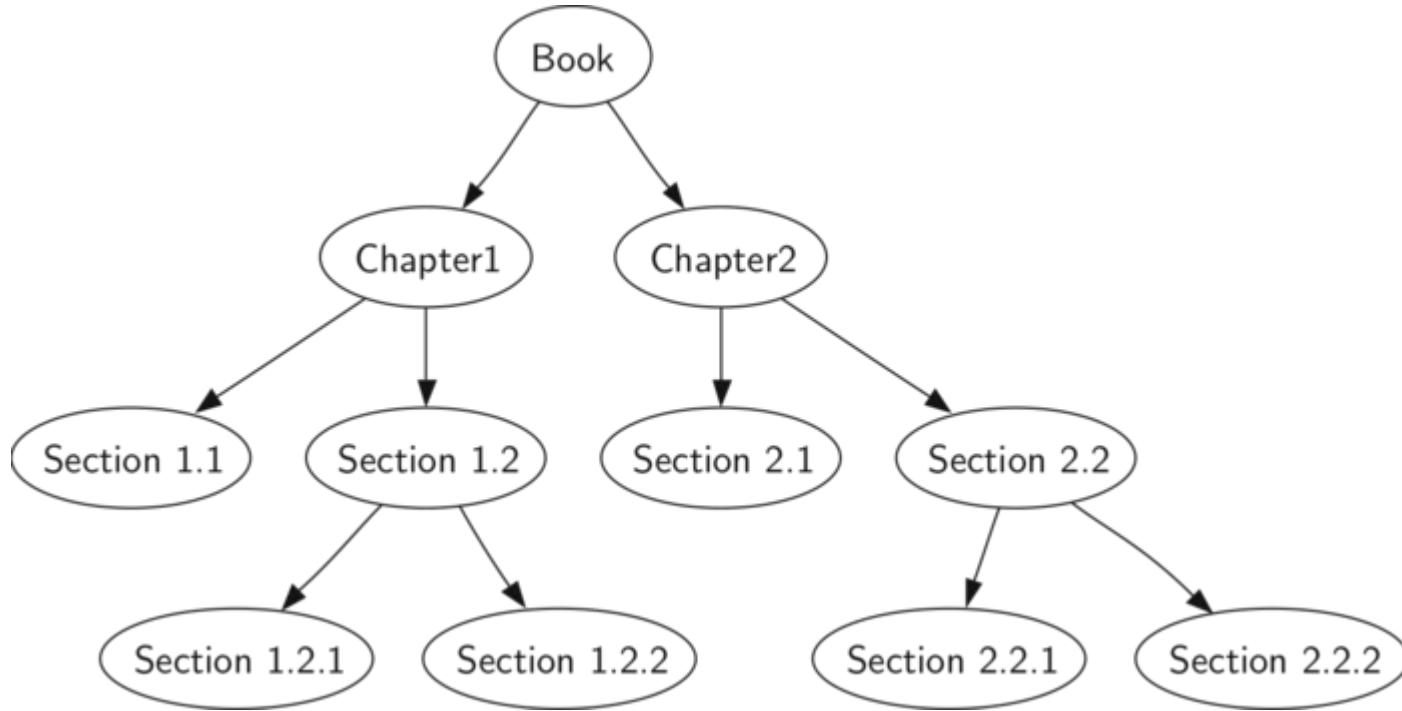


Tree

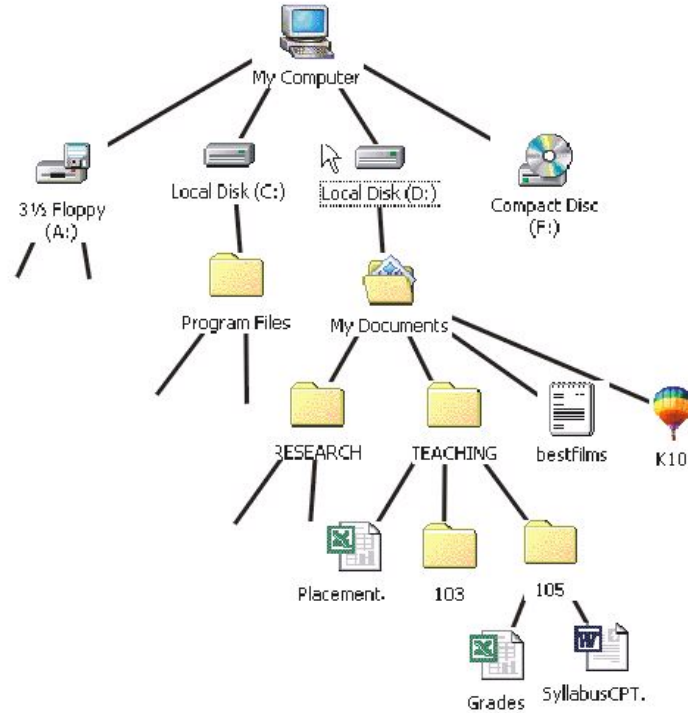
Tree

- 一樣有點點跟線線
- 可以說是某種特殊形狀ㄉ Graph

Tree



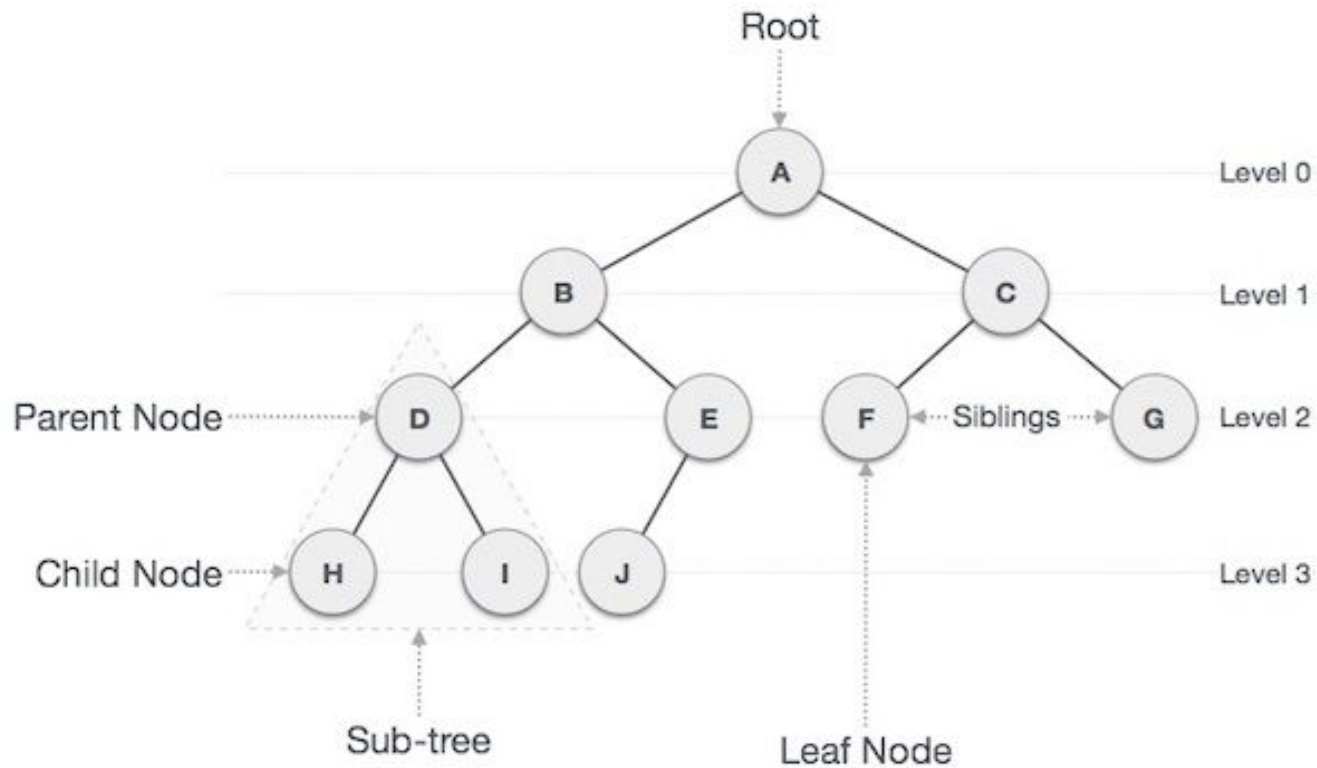
Tree



Tree

- 檔案系統
- 中序轉後序
- Heap
- 資料壓縮 Huffman Tree

Tree



Binary Tree

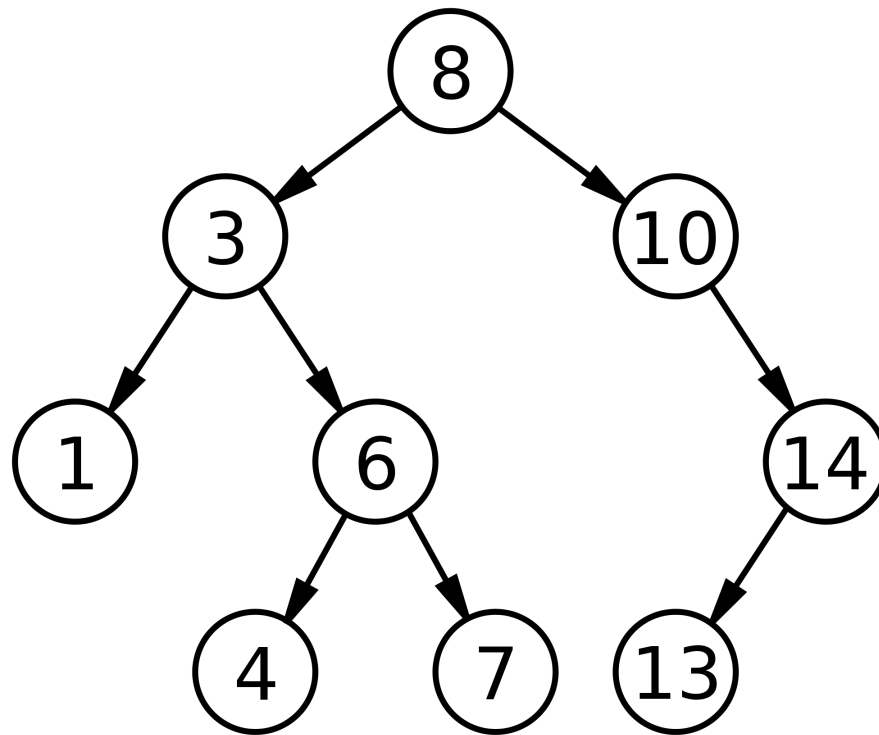
- 電腦很喜歡把東西變成 Binary ㄉ
- Binary Tree 有很多很多應用

Binary Search Tree

- 我有一大坨數字
- 我想知道某個數字有沒有在這坨裡面
- Ex. 14 有在 [8, 3, 10, 1, 6, 14, 4, 7, 13] 裡面嗎？

Binary Search Tree

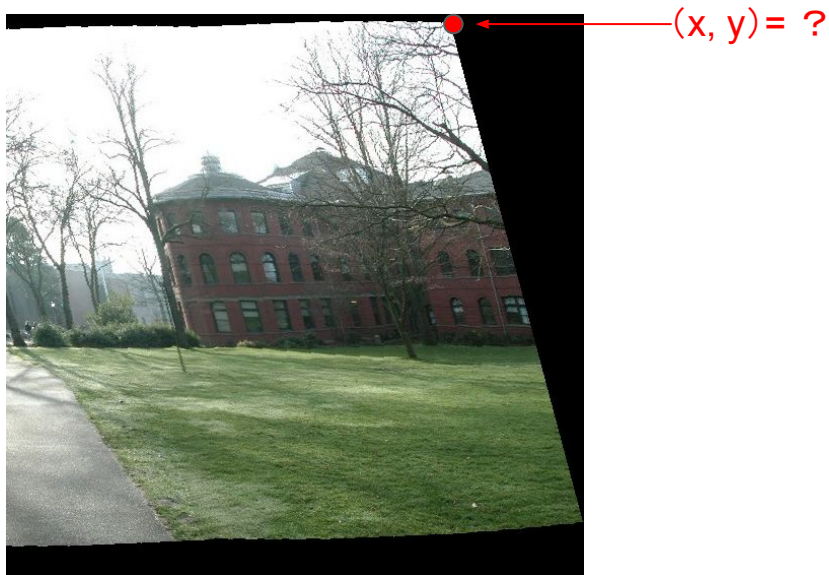
- 右邊的都比較大
- 左邊的都比較小

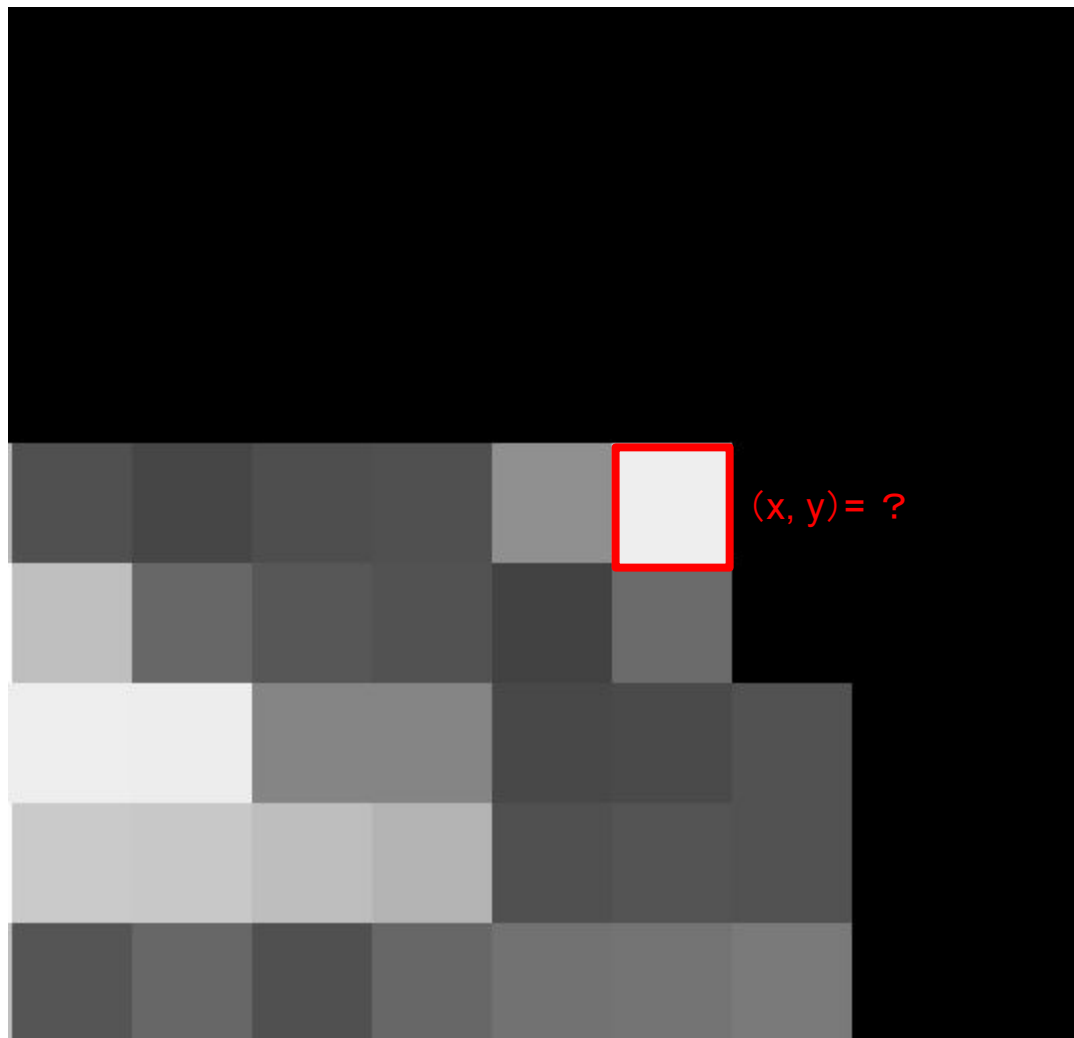


最後，來點應用ㄟ？

影像處理

- 其實是我昨天在寫的作業
- 有一張照片，邊邊有一大片黑色的區域，我想要知道最右上角「不是黑色」的座標是多少

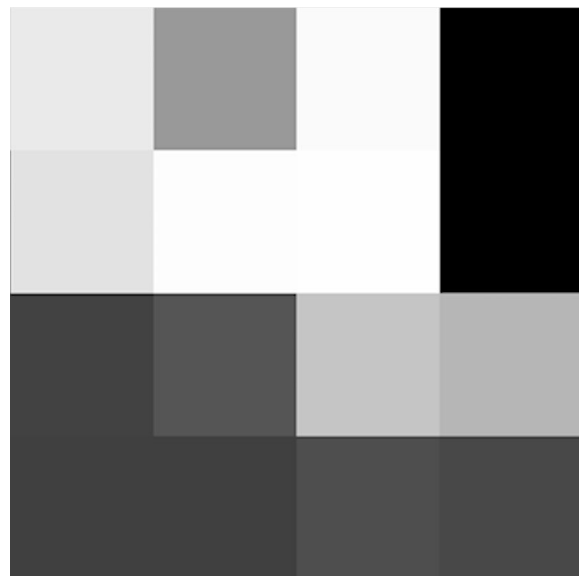




影像處理

- 先來想想看，怎麼把一張灰階照片存在電腦裡面？





0

255

