Today:

My Notes
Important Concepts worth keeping

CH 5.

Stack → last in first out.

pop → 自取從 stack 中取出

push → 放進 stack

執行在一步驟時 push > pop 時 ⇒ 符法、不存在

遞迴 ⟷ 堆疊

My Notes
Important Concepts worth keeping

CH6

Today: / /

Queue → FIFO
(佇列)=(排隊)

front →最早被加入的資料 (第1個被取出)

back, rear →最晚被加入的資料

* simulation 模擬

資料產生出來和大自然 or 人類行為很像

⇒ 這個過程稱為 模擬

ADT Queue Operation

- create empty queue
- destroy "
- empty?
- Add → enqueue
- Remove → dequeue
- Retrieve → getfront

Begin with the end in mind.
– Stephen Covey.

My Questions
Problems & Difficulties needing exploration

環狀陣列. ⇒ back + 1 = front (滿)

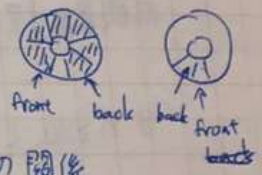
~~不論陣列長大小 ⇒ 無法判斷~~

sol ⇒ 設定機制. (多留1個空間)
or
count
or
多留一個空間

不論 array 為全空 or 全滿.

or

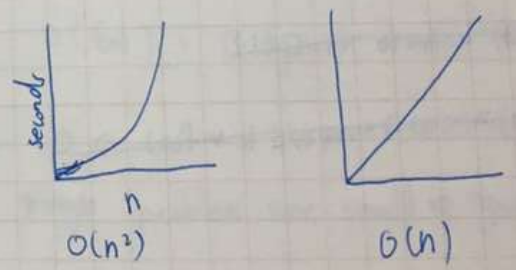
全空 or 全滿時 back & front 的相對位置 都差一



Algorithm growth rate → 程式與 problem size 的關係

* 通常較關注 large problem size

My Opinions
Thoughts, inspirations, and suggestions



O(n) 優於 O(n²)

↳ Big O notation

My Notes
Important Concepts worth keeping

CH7

Today: / /

big O表示的是時間的上限

實際執行≤ big O

Big O不在乎細微的差距. ex: 0.2n ✓ 0.3n ⇒ 細微

用顯著的差異來判斷程式的好壞

$\forall n \geq 10, (2.5n^2 - 2.5*n) \leq 1*n^{10} \Rightarrow O(n^{10})$

$\forall n \geq n_0, (2.5n^2 - 2.5*n) \leq k*n^2 \Rightarrow O(n^2)$

→每個演算法都應找出其能找出最佳 big O的計算方法

~~ex: (n+1)*(c+a) + n*w is O(1)~~

~~$\forall n \geq n_0 \ (n+1)*(c+a)+n*w \leq k*f(n) \Rightarrow$~~

My Questions
Problems & Difficulties needing exploration

$O(1)$

$O(\log_2 n) = O(\log n)$

$O(n)$ $\quad = \frac{1}{\log_2 10}(\log n)$ 常數

$O(n \log_2 n)$

$O(n^2)$

$O(n^3)$

$O(2^n)$

優 ↓ 劣

$O(n^3 + 3n) \Rightarrow O(n^3)$ 忽略低位階 → 分析程式最花時間的地方

瓶頸

$O(5f(n)) \Rightarrow O(f(n))$ 忽略常數

My Opinions
Thoughts, inspirations, and suggestions

$O(f(n)) + O(g(n)) = O(f(n) + g(n))$

~~progb~~ problem size small ⇒ ignore an algorithm's efficiency

My Notes
Important Concepts worth keeping

Today: / /

CH 7.

- big-O notation 辨別程式好壞的標準.

效率 < 時間?
空間

- 在不同情況(不同程式、不同演算法、不同電腦)相同結果.

comparison, assignment, write (cout)

- 費時的or時常使用的 程式通常會計算 big-O

- 影響時間的原因：電腦 程式、資料

Page 1.

- 不是所有程式都可以計算出 big-O

(不只n-個變數...多變數)

Less can be more, slower might go faster.

My Questions
Problems & Difficulties needing exploration

best case

average case

worst case

	Sequential search	binary search
Worst case	O(n)	O(log₂n)
averge case	O(n)	O(n)
best case	O(1)	O(1)

1 7 5 2 3 4

My Opinions
Thoughts, inspirations, and suggestions

My Notes
Important Concepts worth keeping

Today: / /

Efficiency of sorting Algorithm

Sort key 排序鍵
↳ The part of data item that we consider when sorting a data collection

- 內 or 外

internal sort

external sort，"透過記憶体之外的地方 → require secondary memory
大量的資料

相同值維持不變

stable sort	unstable sort
bubble	selection
insertion	quick
merge	heap
radix	

All the roads worth walking are uphill.
– John Maxwell

My Questions
Problems & Difficulties needing exploration

- bubble sort

由最小開始 兩兩比較 (第1&第2)
(第2&第3)

每輪比較の資料縮減1.
範圍

用 swap (互換)の次数決定 時間

- Selection Sort

My Opinions
Thoughts, inspirations, and suggestions

在未排序の範圍裡找出最小的 未排序的
與最左的元素交換
↳ unstable の主因

未排序の範圍縮小1個

凡是值得走的路都是上坡路。

My Notes
Important Concepts worth keeping

Today: / /

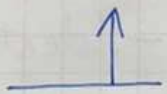
- Insertion Sort

開始：⇒第1個定為已排序

自未排序內第一個元素插入已排序內

⇒適當的位置，並定義為已排序

較不好の
排序方法



My Questions
Problems & Difficulties needing exploration

Merge Sort

- 策略：divide & conquer

①分組

②合併

11-16

Radix Sort.

* Radix = the base of a system of numbers.

My Opinions
Thoughts, inspirations, and suggestions

* 策略：decompose the sort key by the radix

My Notes
Important Concepts worth keeping

CH8

Today: / /

- Terminology

  * Tree 由 nodes 和 edges 組成

  * Parent / child child

    Ancestor / descendant descendant

  * subtree: Any nodes and its descendant.

  * General Tree: children node 可為 0 或任意數字

  * Root: the only node in the tree with no parent.

  * ~~bad~~ Leaf: node with no children

    樹的末梢

  * siblings: node with a common parents

  * Ancestor of node B.: A node on the path from the root to B

  * Descendant of node B: A node on a path from B to a leaf.

The opposite of success is not failure, but doing nothing.

My Questions
Problems & Difficulties needing exploration

* binary tree → 最多又能有2個 children node
  每一個 node

* Height: 從 root 到 leaf 的最長距離

* Full binary tree: 除了 leaf 以外 其它 node 皆有 2 個 children

  empty tree也算一種 full binary tree

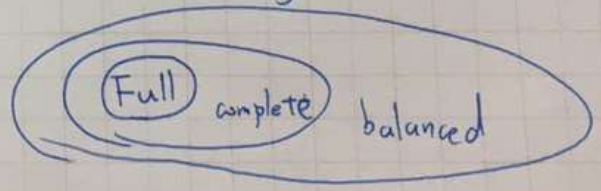
* complete binary tree: ① All node at level <= h-2 have two children

  ② level h-1 has children, all nodes to its left at the same level have two children each.

  ③ when a node at level h-1 has one child, it's a left child

My Opinions
Thoughts, inspirations, and suggestions

* balanced binary tree: 左右子樹高度差不超過 (<=1)



成功的反面不是失敗，而是什麼都不做。

My NoTes

Important Concepts worth keeping

* Traversal

① Preorder root → left → right

② Inorder left → root → right

③ Postorder left → right → root