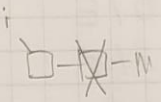
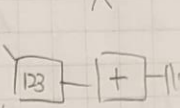
A * (B+C) * D

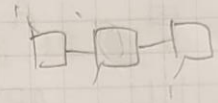
1* (2+3) * 5



A A



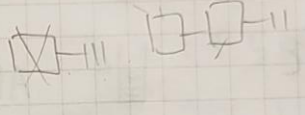
*  A

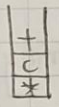


( A incoming > instack

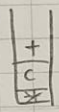
(> *

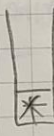
B  AB



\+  AB incoming > instack

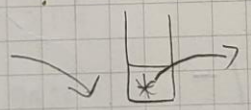
\+ > (

C  ABC

)  ABC+ incoming ≤ instack

) ≤ +

(



* ABC+* 找到左括號就停止

incoming ≤ instak

* ≤ *

D  ABC+*D

=> ABC+*D*

A+B*C-D/E

1+2*4-3/5

A		A
B		AB
*		AB
C		ABC
-		ABC*+
D		ABC*+D
/		ABC*+D
E		ABC*+DE
		ABC*+DE/-

incoming > instack

incoming ≤ instak

incoming > instack

ABC*+-D/E

AB*C

a+b*d+c/d

d/c+d*b+a

E/D-C*B+A

ED/CB*-A+

+AB

/d

/dc

+dc/d

dc/b+a+

+a+*b/cd

計算（後序）

ab-cd+

ab-(c+d)

ab*c+d*

ab*c+d*

abc-d*

ab-c*d

ABC+* = 246+*

A=2

B=4

C=6

新增「運算元」到堆疊中

遇到運算子 用兩個運算元

246+*

46+*

6+*

+*

pop 2次

*

*

pop 2次

6+4=10 結果再

2*10=20

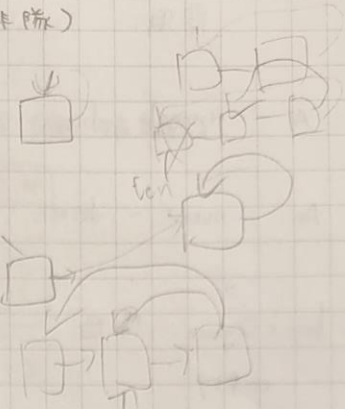
123+*5*

6-4=2

Queue（佇列，排隊）

最後一個 back，rear

第一個 front



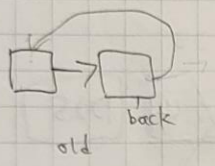
米 先進來先服務

1. 把 int 轉 string

2. 判斷迴文 Queue front = stack top ?

temp

enqueue



環狀 linked list

沒有 null

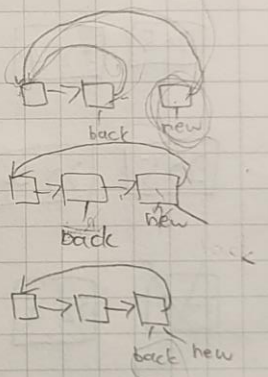
合併

↓

new → next = back → next

back → next = new 的 ptr

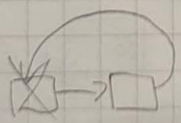
back = new



dequeue

① back → next = back

back = null



② back → next = temp → next

delete temp

應用 simulation

A time - driven simulation

An event - driven simulation

20	5
22	4
23	2
30	3

bankQueue 隊伍

empty

20 5

20 5

20 5

20 5 | 22 4

20 5 | 22 4

20 5 | 22 4 | 23 2

~~20 5~~ | 22 4 | 23 2

22 4 | 23 2

~~22 4~~ | 23 2

(非序) anEventList 處理

A 20 5

Empty

D 25

~~A 22 4~~ | D25

D 25

~~A 23 2~~ | D 25

D 25

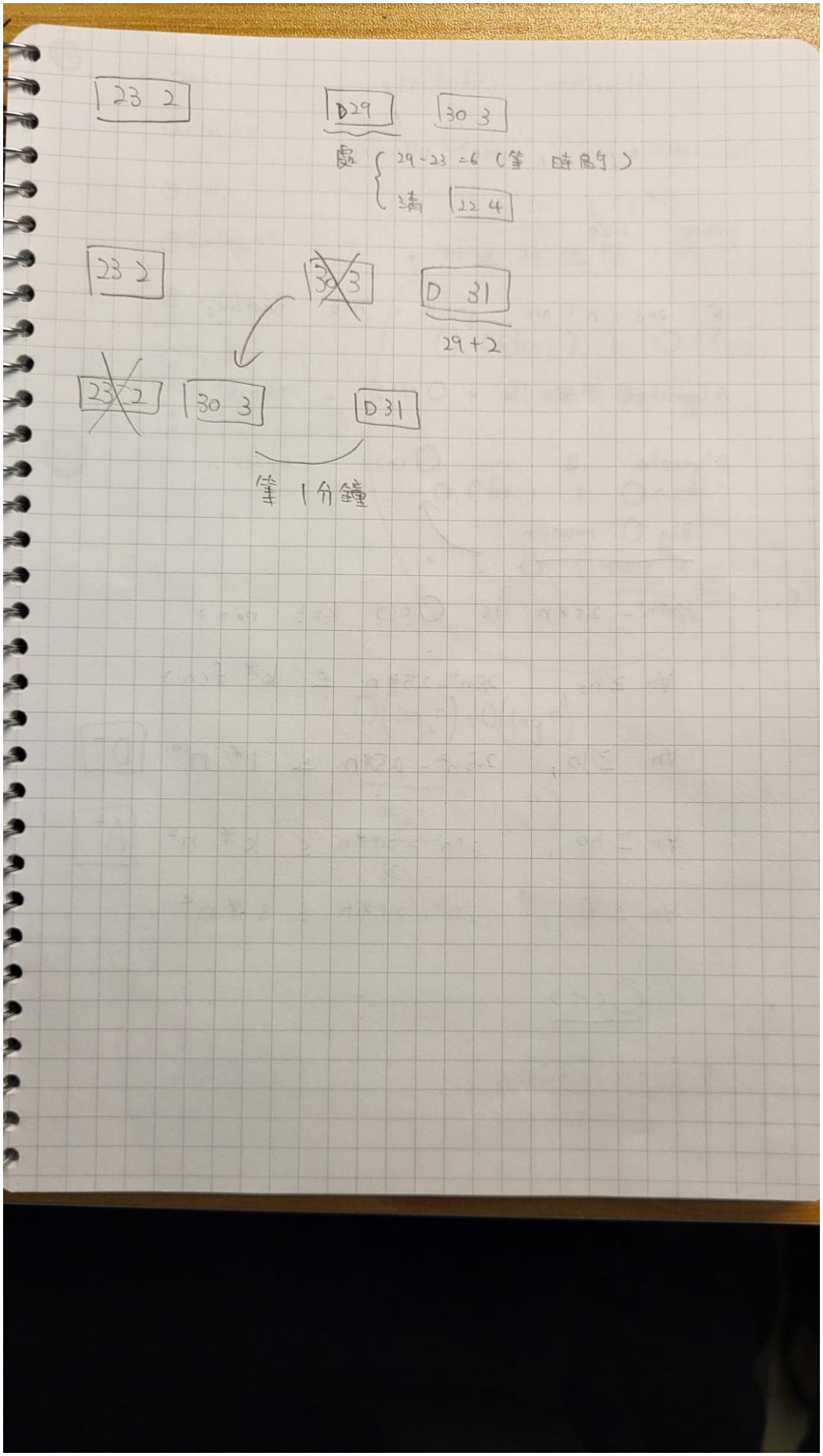
D 25 | 30 3

處理 (算 20 5 25-22=3 (時間 wait))

30 3

D 29 | 30 3

25+4


23 2
D29
30 3
題 { 29-23 =6 (筆 時鳥9)
猜 22 4
23 2
30 3
D 31
29+2
23 2
30 3
D31
等 1分鐘

Algorithm efficiently

- Time efficiency , space efficiency
- problem size

n^2 and n are growth-rate functions

Algorithm A is $O(n^2)$ - order n^2

Algorithm B is $O(n)$ - order n

Big O notation

Ex $2.5n^2 - 2.5*n$ is $O(?)$ $k=?$ $n_0=?$

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

$\forall n \geq 10$, $2.5n^2 - 2.5*n \leq 1 * n^{10}$ D⁻

$\forall n \geq n_0$, $2.5n^2 - 2.5*n \leq k * n^2$ A⁺

$\forall n \geq 0$, $2.5n^2 - 2.5*n \leq 3 * n^2$

$O(n^2)$

好 1. $O(1)$

2. $O(\log_2 n)$

3. $O(n)$

4. $O(n\log_2 n)$

5. $O(n^2)$

6. $O(n^3)$

Bad 7. $O(2^n)$

技巧

ex: $O(n^3+3n)$

$n^3+3n \leq ? n^3$

- 忽略低位階

$O(5f(n)) = O(f(n))$

- 忽略常數

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

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

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

$\frac{1}{\log_2 10}(\log_{10} n)$

k

✓ • Worst case $O(n)$

✓ • Average Case $O(n)$

• Best Case $O(1)$

2
8
1
9
5
n

Average: $\frac{1+6}{2} = 3.5$

$(n+1)/2$

$0.5n \leq kn$

二元搜尋

• Worst Case $O(\log_2 n)$

1
2
5
n
8
9

$6/2 = 3$
$3/2 = 2$
$2/2 = 1$
} 3

$n = 2^k$

$\log_2 n = k$

#排序並不會有太大的影響

排序

- internal sort (內部)
  - stable
    - insertion
    - merge
    - radix
  - selection
    - quick
    - heap
- external sort (外部)

- stable sort (相同值維持不變的排序)

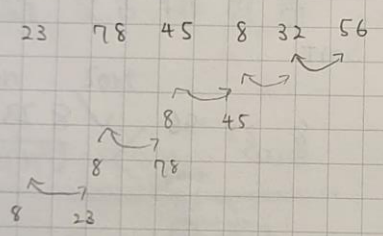
8a 28 14 5a 8b 26 2 6 29 5b

↓

2 5a 5b 6 8a 8b 14 26 28 29

慢 stable

bubble sort (兩兩位置比較，小的往前移)



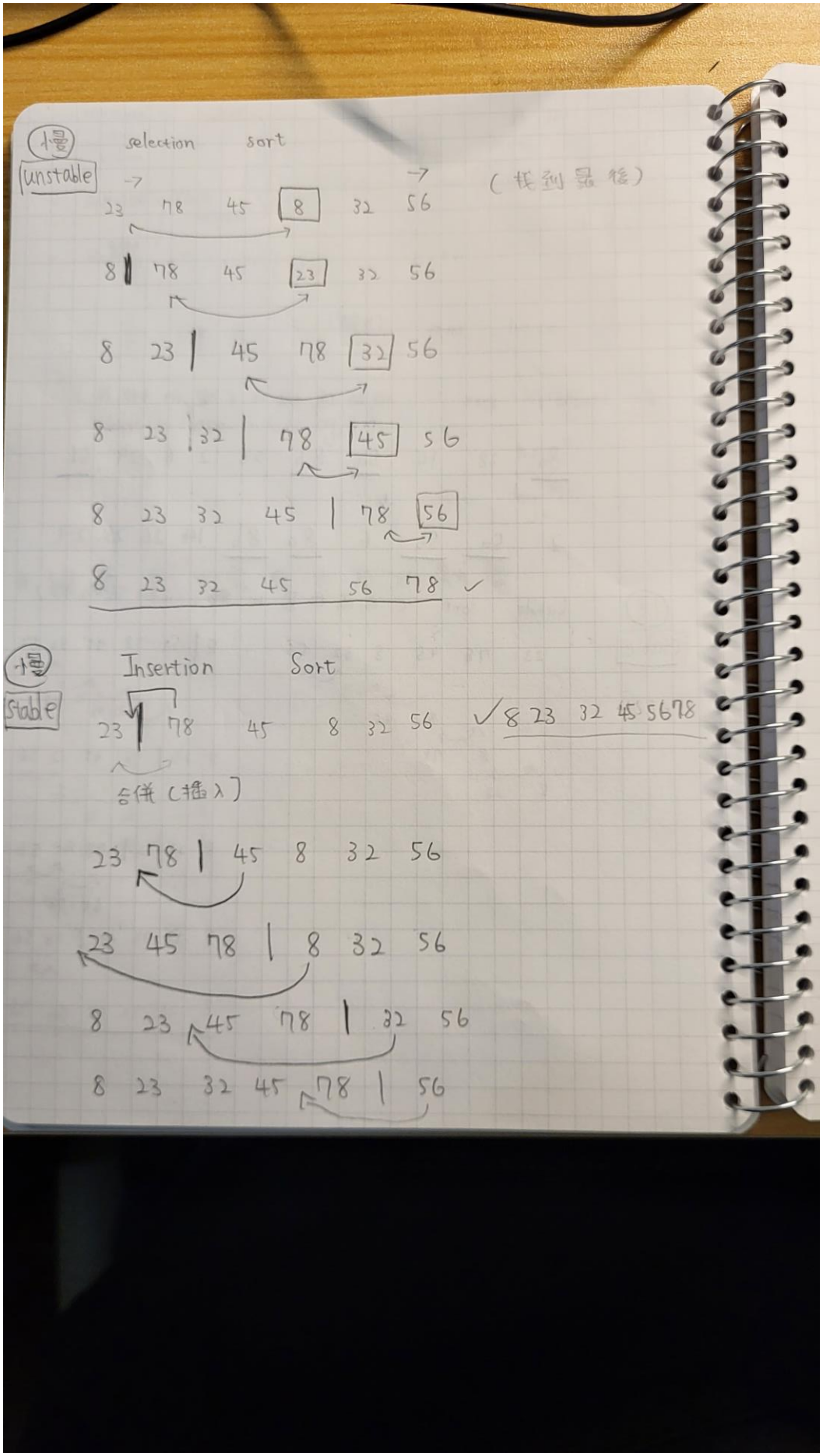
8 | 23 78 45 32 56

8 23 | 78 45 32 56

8 23 32 | 78 45 56

8 23 32 45 | 78 56

8 23 32 45 56 78


慢
selection sort
unstable
(找到最後)
23 78 45 8 32 56
8 78 45 23 32 56
8 23 45 78 32 56
8 23 32 45 78 56
8 23 32 45 56 78 ✓
快
Insertion Sort
stable
23 78 45 8 32 56 ✓8 23 32 45 5678
合併(插入)
23 78 45 8 32 56
23 45 78 8 32 56
8 23 45 78 32 56
8 23 32 45 78 56

- Bubble sort

$n + \sum (n-pass+1) + \sum (n-pass)$

$= n + 2\left[n*(n-1) - n*(n-1)/2\right] + (n-1)$

$= n^2 + n - 1 \Rightarrow O(n^2)$

核心比較

if (A[index] > A[index+1])

swap (A[index], A[index+1])

$4*n*(n-1)/2 = 2n^2 - 2n \rightarrow O(n^2)$

Worst case

判斷是否排好序 => Best Case $O(n)$

- selection sort

$O(n^2)$ < worst / Best case

Swap $O(n)$ 搬動

→ swap $O(1)$ (best case)

加條件(排序好)

~~$O(5)$~~ ~~$O(0)$~~

只有 $O(1)$

- Insertion Sort

$O(n^2)$ worst case

$O(n)$ best case 排序好

* 希爾排序法 unstable

插入一 20 80 40 25 60 10 15 h=3 間隔3

20 60 40 25 80 10 15

20 60 10 25 80 40 15

15 60 10 20 80 40 25

插入排序 (h=1)

15 60 | 10 20 80 40 25

10 15 60 | 20 80 40 25

10 15 20 60 | 80 40 25

10 15 20 60 80 | 40 25

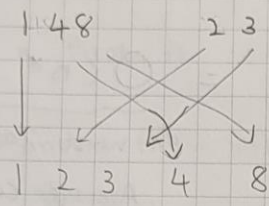
10 15 20 40 60 80 | 25

10 15 20 25 40 60 80 ✓

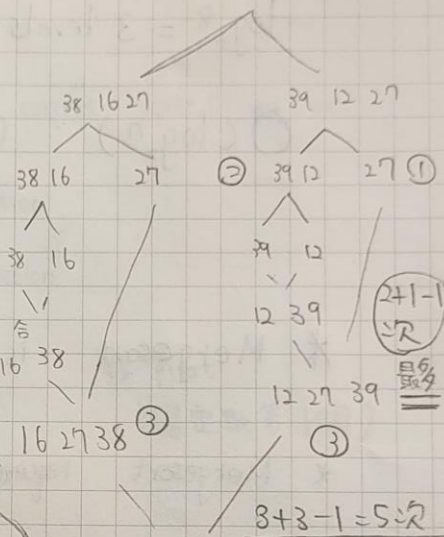
Mergesort 合併排序 (遞迴) Stable

1. 先分組
2. 排序
3. 合併

ex: 8 1 4 3 2



38 16 27 39 12 27

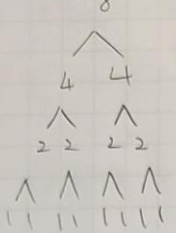


3+3-1=5次

12 16 27 27 38 39

```
mergeSort ( Data Type theArray [],
            int first, int last) {
  if ( first < last ) {
    int mid = (first + last)/2
    mergeSort ( the Array , first, mid)
    mergeSort ( the Array , mid+1, last)
    merge ( the Array , first, mid, last)
  }
}
```

O(n)



$2^0 \times 3 \times 8 - 1$ → $3*n - 2^0$ → $O(n)$

$2^1 \times (3 \times 4 - 1)$ → $3*n - 2^1$ → $O(n)$

$2^2 \times (3 \times 2 - 1)$ → $3*n - 2^2$ → $O(n)$

2^3

$\log_2 8 = 3$ levels

$O(\log_2 n) \cdot O(n) = \underbrace{O(n * \log_2 n)}$

worst case

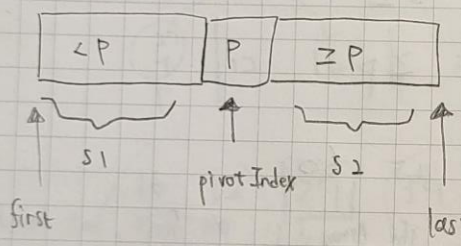
Average case

* Mergesort is an extremely fast algorithm

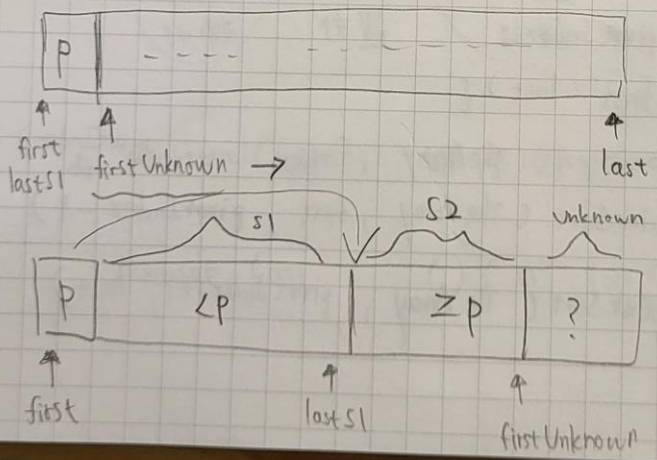
* Mergesort requires a second array as large as and original array

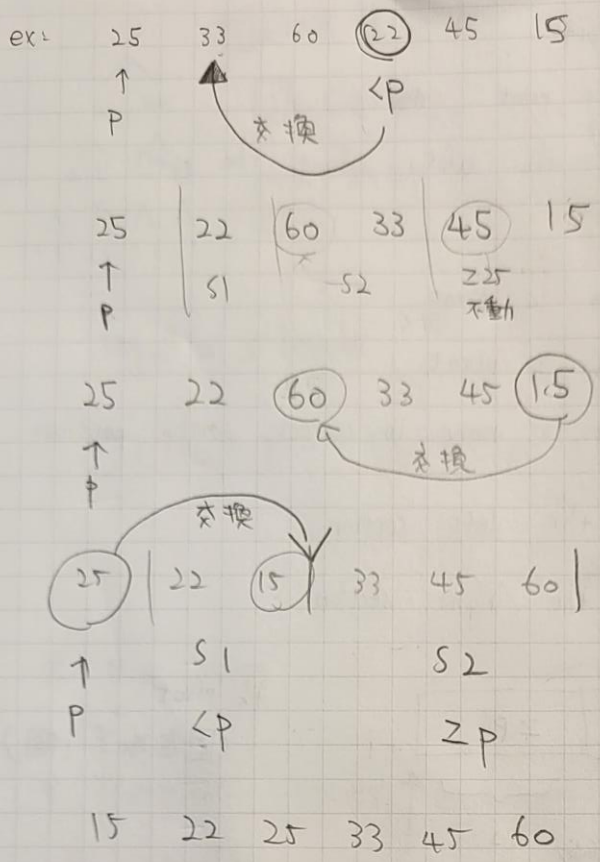
Quicksort

- choose a pivot 軸
- Partition the array about the pivot
  分割
  - items < pivot
  - items ≥ pivot
  - Pivot is now in correct sorted position
- Sort the left section
- sort the right section



找 pivot
(固定取第1個)

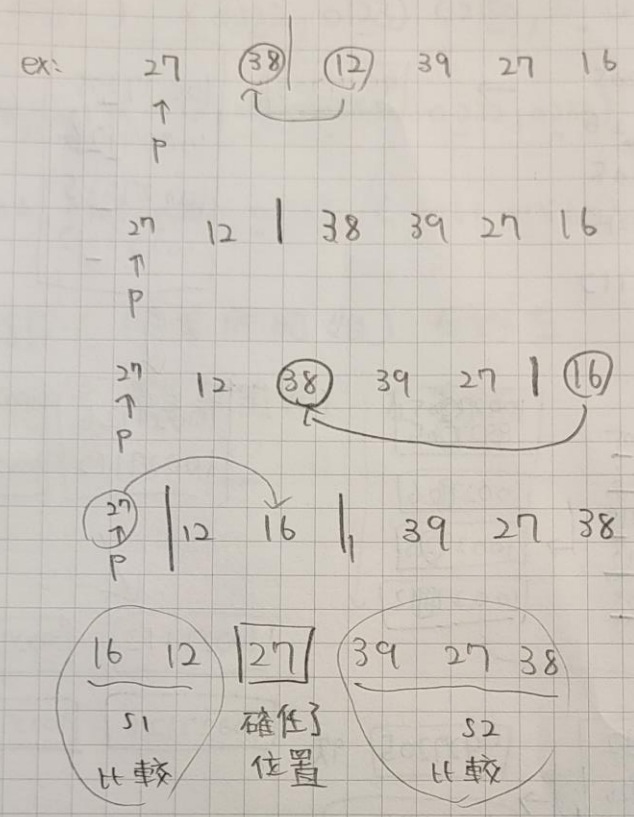





```
void quickSort ( DataType theArray [] int first, int last)
    int pivotIndex;
    if (first < last) {
        partition (TheArray, first, last, pivotIndex)
        quickSort ( TheArray, first, pivotIndex - 1)
        quickSort ( TheArray, pivotIndex + 1, last)
    }
}
```

partition $O(n)$

recursive calls $O(\log n)$



* Worst Case : 排序好的 , $O(n^2)$

Average Case : $O(n^* \log_2 n)$

Radix Sort (快)

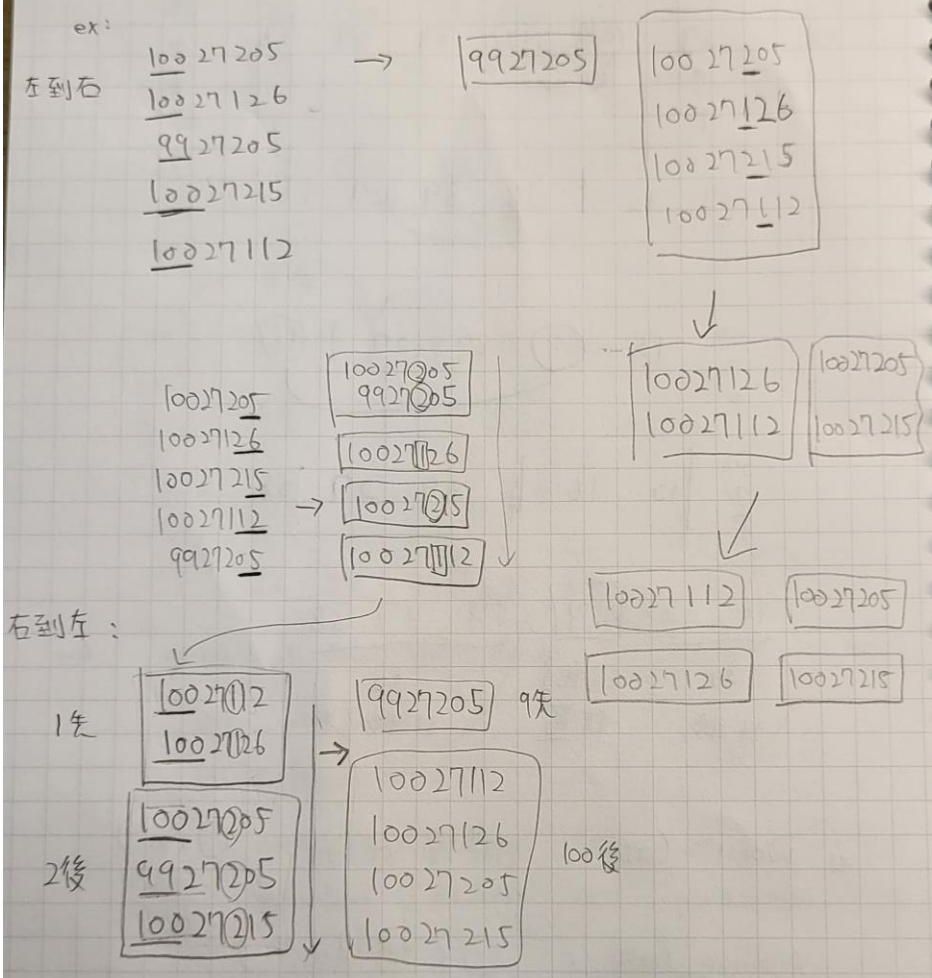
O(n)

分解 取部分值

分配至對應容器

ex:

左到右



右到左：

ex: 0123 , 2154 , 0222 , 0004 , 0283 , 1560 , 1061 , 2150

(1560, 2150) (1061) (0222) (0123, 0283) (2154, 0004)

(0004) (0222, 0123) (2150, 2154) (1560, 1061) (0283)

(0004, 1061) (0123, 2150, 2154) (0222, 0283) (1560)

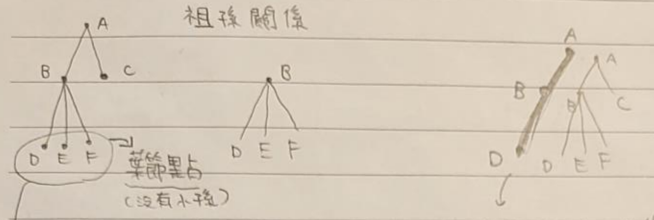
(0004, 0123, 0222, 0283) (1061, 1560) (2150, 2154)

LSD (種要的開始) 右到左

MSD (重要的開始) 左到右

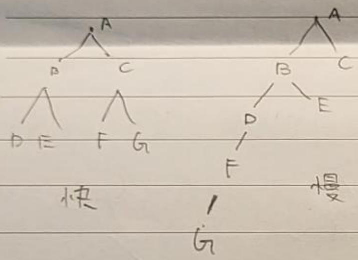
位置導向：list, stack, queue, binary tree
內容導向：sorted list, binary search tree

Tree 親子關係
祖孫關係

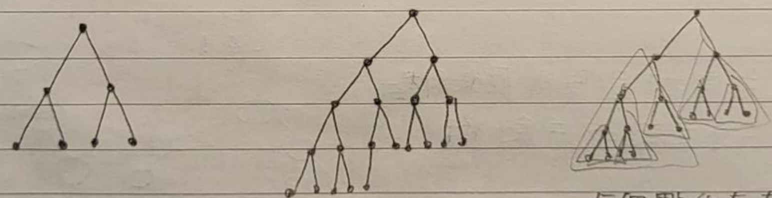


A是B的父節點　　　祖先節點（往上走）
or
兄弟節點　　　子孫節點（往下走）

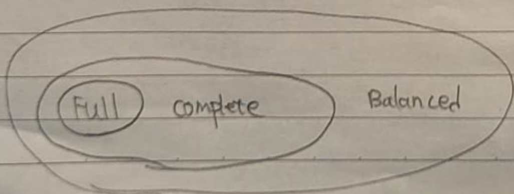
Height of tree（影響速度）
Worst case：O(logn)
Best case：O(1)



1. 完全樹　　2. 完整樹　往左靠　　3. 平衡樹



每個點的左右高度不會超過1



No.
Date

h 樹高　　$2^h - 1$ 資料量

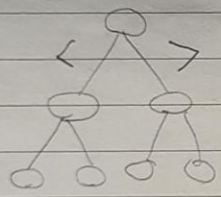
完整二元樹
- $h \geq \log_2(n+1) \Rightarrow h = \lceil \log_2(n+1) \rceil$ 最小樹高
- $h \leq \log_2(n)+1 \Rightarrow h = \lfloor \log_2(n) \rfloor + 1$ 最大樹高

點 − 邊 = 1

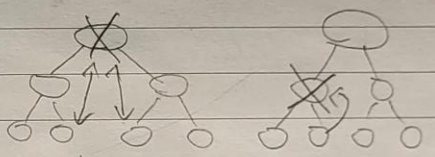
$N_0 = N_2 + 1$ （不會只有1個小孩的）

N_0：下面沒有人

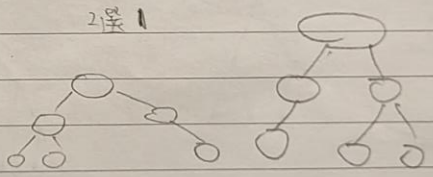
Search（左小右大）



inorder => 排序



2選1



前序柯還原成原來的樹

中序走訪 => 平衡樹

一般樹 => 二元樹

(左)：最左側小孩

(右)：右邊的兄弟

Chey culture