前言

2025年9月9日　　下午 03:20

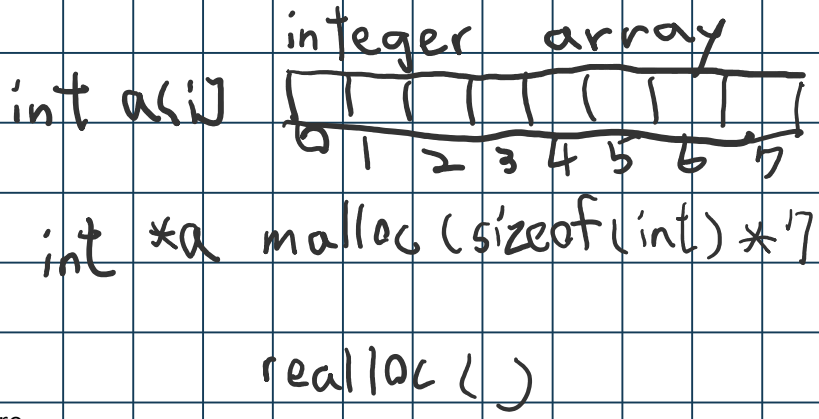
teams聯繫 黃鈺峰　　01_cours...

會用到GitHub及LeetCode 先註冊好帳號

coding style

Comments , naming, documentation

Assignment 1　　02_the_...



Textbook CH.1~CH.5 What is DataStructure

小考

2025年9月15日　　下午 08:33

一定會有紙筆coding
所有上過的都是範圍 有課本額外補充的會先講

Array Practice 1

2025年9月26日　　上午 09:44

Question: Dynamic Array

```
int *array;
int n = 10;

array = (int *) malloc(n * sizeof(int));

for(int i = 0; i < n; i++) {
  array[i] = i + 1;
}

for (int i = 0; i < n; i++) {
   printf("%d ", array[i]);
}

n = n * 2;

array = (int *) realloc(array, n * sizeof(int));

for (int i = n/2; i < n; i++) {
   array[i] = i + 1;   // initialize new elements
}
```

可能要不到空間
check availability
* array → NULL

realloc 不一定拿到一樣的起始位置
→ 資料不一定被覆蓋
array 連續記憶體空間 < 有→新分配 / 沒有→覆蓋

11402 CS203A, COMPUTER SCIENCE & ENGINEERING, YUAN ZE UNIVERSITY　14

☐ **Summary of Issues by Severity:**

☐ **Critical Issues (3):**

Missing malloc error checking - Program crashes if memory allocation fails
Memory leak - Never calls free(array)
Missing realloc error checking - Can lose original array pointer

☐ **Style & Best Practice Issues (4):**

Unnecessary type casting - (int *) not needed in C
Missing headers - No #include <stdio.h> or <stdlib.h>
Inconsistent indentation - Mix of 2 and 4 spaces
Inconsistent spacing - for(vs for (

☐ **Logic & Design Issues (3):**

Magic number - Hardcoded 10
Confusing variable reuse - n means different things
Poor output formatting - No newlines or context

☐ **Advanced Issues (3):**

Potential integer overflow - n * 2 could overflow
No function structure - Everything in global scope
Missing const correctness - Could use const for clarity

☐ **Great for Teaching Because:**

Covers fundamentals: Memory management, error handling
Shows real consequences: Crashes, leaks, undefined behavior
Multiple difficulty levels: From basic syntax to advanced design
Practical skills: Code review, debugging, best practices
Complete solution provided: Shows the "right way" to do it

☐ **Student Learning Outcomes:**

Students will learn to:

Always check malloc/realloc return values
Free all dynamically allocated memory
Write defensive code that handles errors gracefully
Follow consistent coding style
Think critically about code quality and maintainability

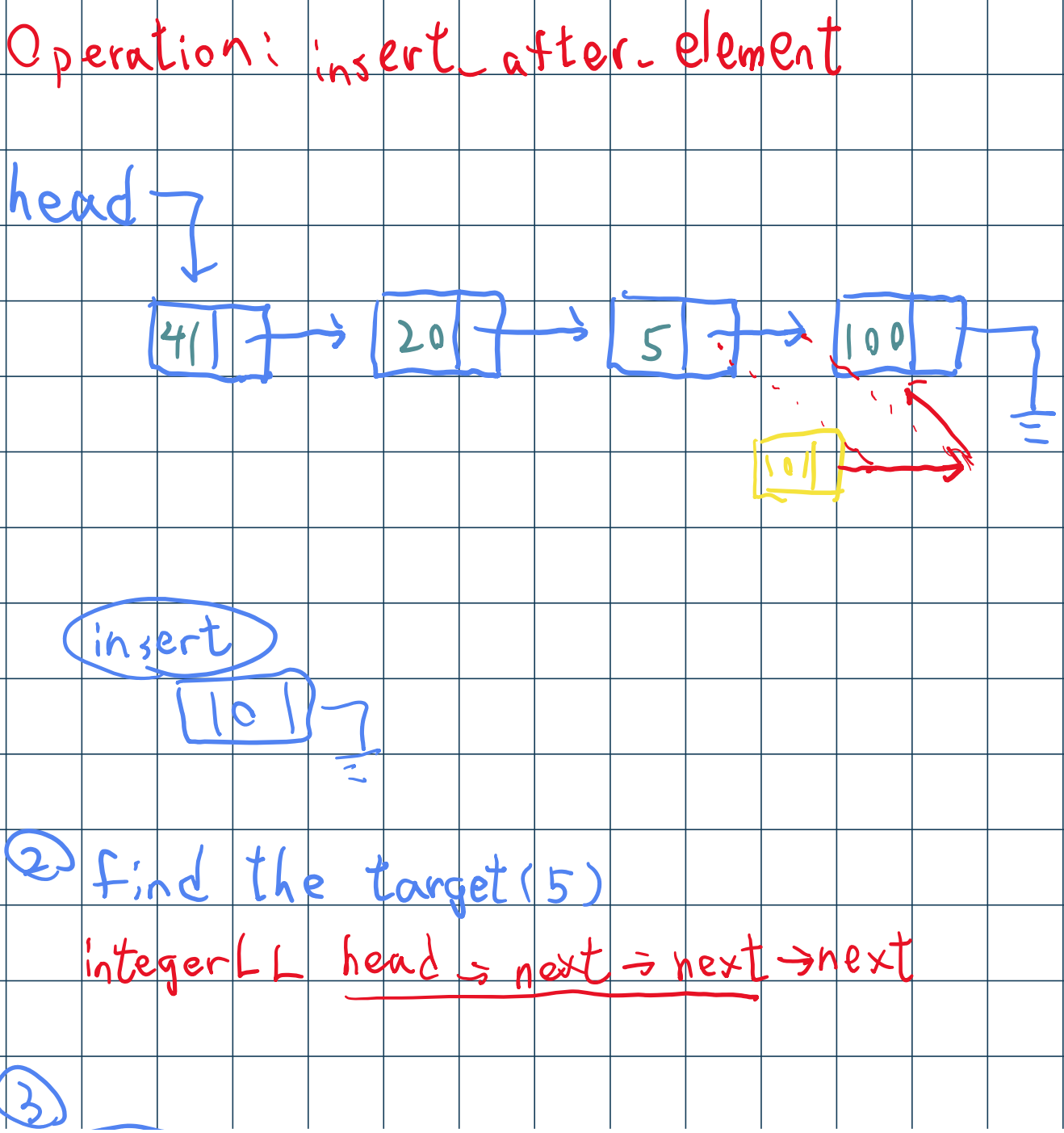
9/30從array到linklist

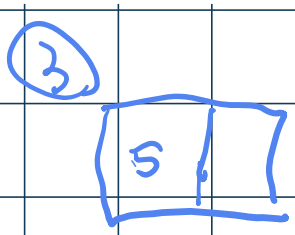
2025年9月23日　下午 04:54

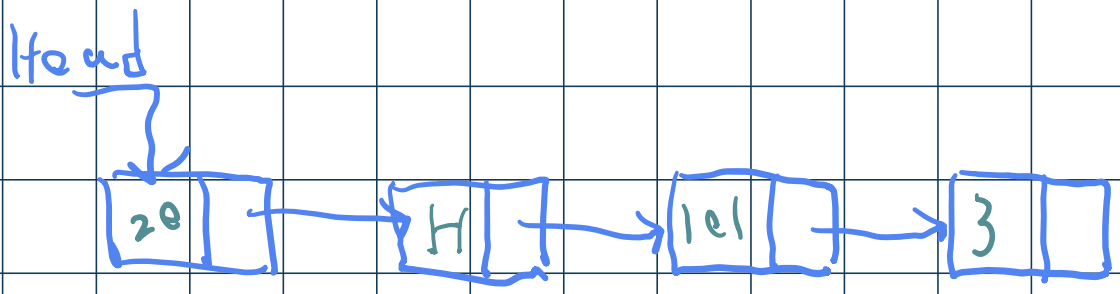
Linklist相較Array差異:

1. 多了Head node和end node(null reference)
2. 記憶體不一定連續 而是用reference(pointer)連成陣列
3. 可以在runtime時resize
4. 任何元素的存取都要從head開始

Array:
隨機存取較有效率，需先和系統要好記憶體大小





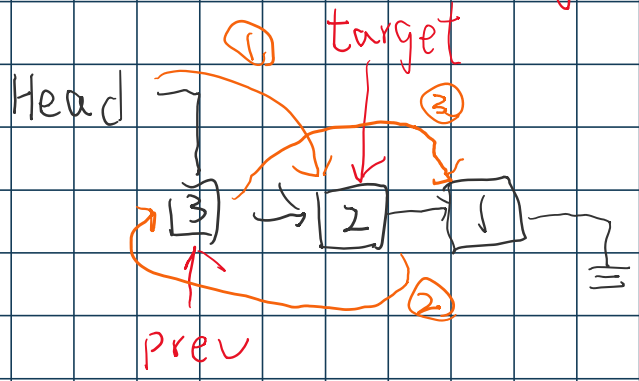


Doubly Linked List
Head往回指 end node往後指 都是NULL

10/07 LinkList characteristics &Sorting

2025年9月30日　下午 03:28

goal swap (prev, target)



prev

prev → next

target:

target → next

① Head = prev ⇒ Head = target

② target → next = prev

③ prev → next = target → next

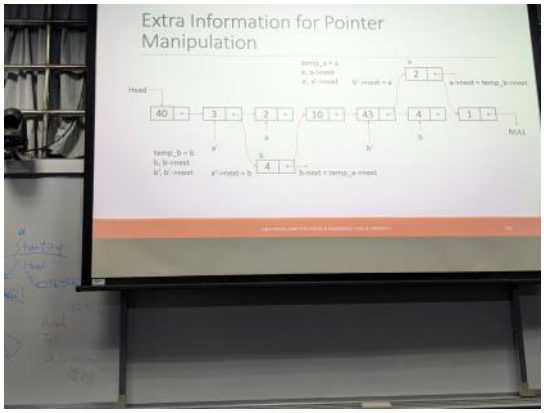
Assignment III

2025年10月13日　上午 09:41

Selection Sort using Linklist

- swap by value → possibly causing unstable sort
  e.g 1 2 2 3 4 ...
  not guarantee the same numbers' order
- swap by pointer
  1. sorted Tail
  2. min
  3. min_prev

「不」兩值互換 如果 a b adjacent(相鄰)?



小考II複習

2025年10月20日　　下午 10:15

trade-offs 權衡

Retrieve 檢索

Singly / Circular / Doubly Linked List

單/雙指標的

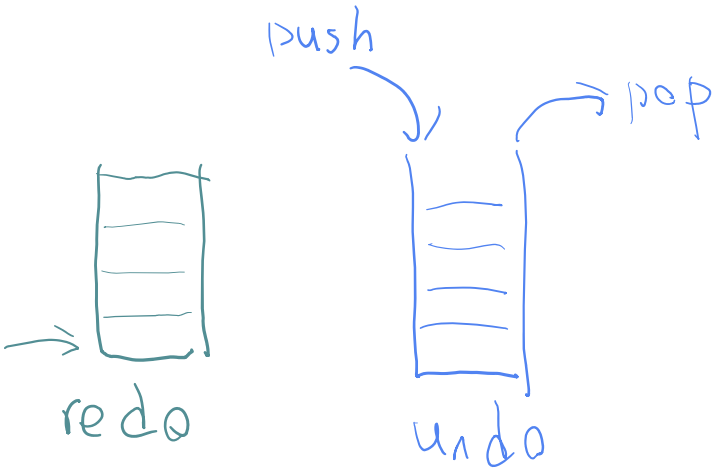
插入

刪除

交換

stack / Quene

redo/undo 通常是用stack



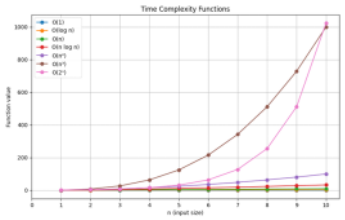
1027期中review

2025年10月27日　上午 09:08

Basic Concept

DS　　ADT：specification of operation for a data structure

complexity < Time performance evaluation
space



Pointer Manipulation

Data Structure

- Array

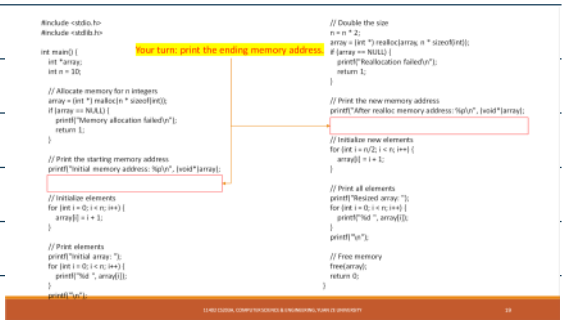
get value by index [0][1][2][3][4][5]　memory continuous

↓

random access

create < dynamic malloc realloc　　# temp
static → int a[5]

realloc(array, size)
↓ pointer

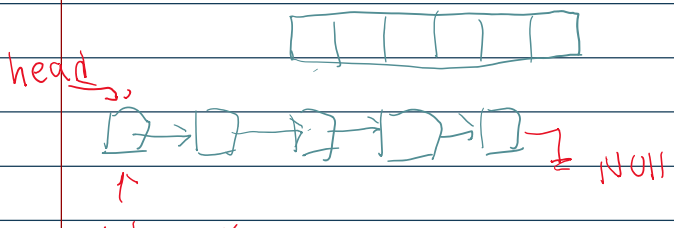


insertion O(1)

deletion O(1)　algorithm < selection sort
bubble sort
insertion sort

search O(n)

- Linked list



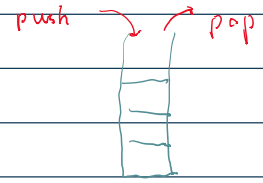
start → head　　data
pointer to next

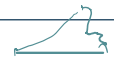
end → NULL

	array	Linked list	
insertion	O(n)	→ O(1)	random access
deletion	O(n)	→ O(1)	

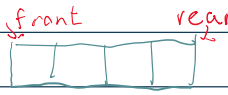
(due to data shiftity)

- Stack

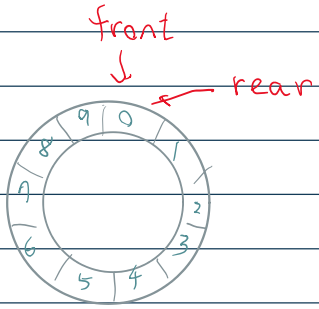




- Quene



linear → circular



ADT

basic { create, push, pop

status { isEmpty, isFull

basic { create, enquene, dequene

status { isEmpty, isFull

Study Note

2025年11月4日　　下午 03:48

HW4

2025年11月17日　　上午 09:43

依循Hash Function模板 實作hash function
有沒有其他function來讓效果更均勻打散同時
用tempfile把老師的code clone下來
1.程式碼
2.README包含hash function 設計原理 邏輯 執行結果貼進去 用老師已經給的測資看pattern(看不出來 要看如何增加資料去更具體)

3.截圖 如果key出來是產生同樣index 解釋出現的pattern
Bonus 額外做一版修改版程式
index分佈狀況

Readme或是程式中直接寫下不同IDE環境開發造成的修改(老師用Vscode C++23)

截圖:
截最終版就好
然後中間有其他提交版本可以再增加一個檔案?

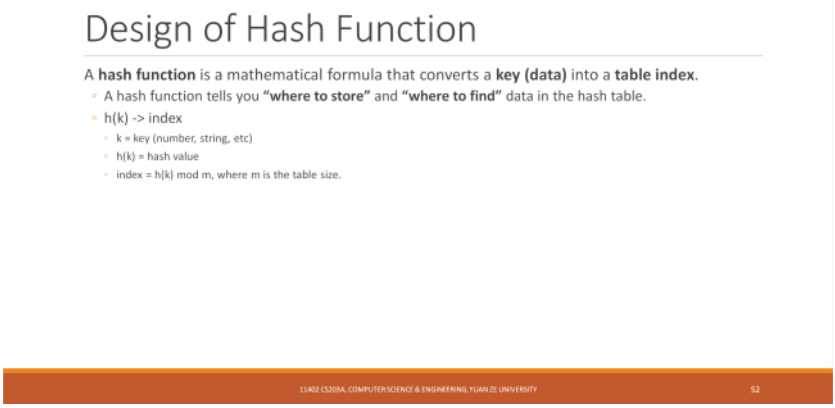
C++版記得把main cpp file中的.cpp改成.h

測資extend去觀察會產生什麼樣的pattern

至少三次commit (抓下來 開發過程 最終版)
*commit messenge要寫清楚(自己看得懂) 把最終版標清楚

如果makefile complier說C++23版本太新 把flag底下兩行拿掉即可

如果有參考網路上程式記得附上reference連結 coding style記得要改



Data locality

2025年11月18日　下午 03:25

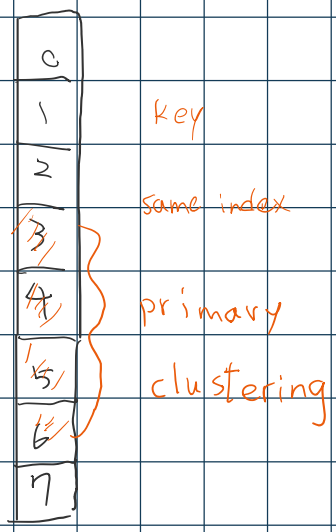
What is probing

空間利用率（擁塞）

70%以下 尤佳　$\alpha < 0.7$

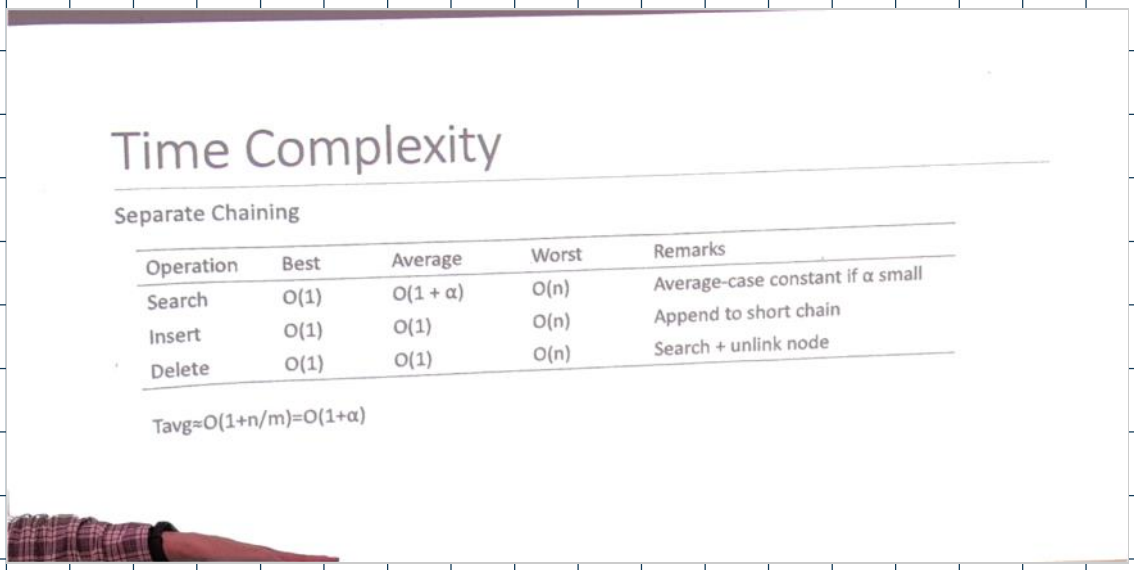
collision次數

Simulation 次數



Quadratic Probing

5855073...



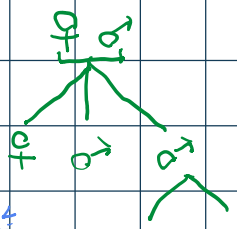
mod值可以改

Double Hashing

Study Note - Tree

2025年11月24日 上午 09:33

家族樹



電腦檔案總管

C:
D:
E:

- Windows
- Program Files
- Users

生物親代子代

domain IP

ip
domainname
140.138.14x.xxx
www.yzu.edu.tw

Linked List

sequential

結構 水平

子代 唯一

Tree

non linear

no cycle

階層 (hierarchy)

垂直

left, right (0~n個)

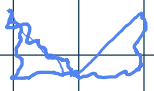
Tree → binary tree → binary search tree

degree限制
child node 0~2

< parent node
child node
reletionship

左子節點數 ≤ 母節點數
右子節點數 ≥ 母節點數

平衡左右子數



AVL (Adelson-Velsky and Landis) Tree

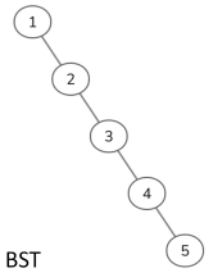
An AVL tree is a **binary search tree** in which for every node in the tree, the height of the left and right subtrees differ by at most 1.

AVL tree was named after inventors Georgy **A**delson-**V**elsky and Evgenii **L**andis, is a self-balancing binary search tree.

Balancing strategy:

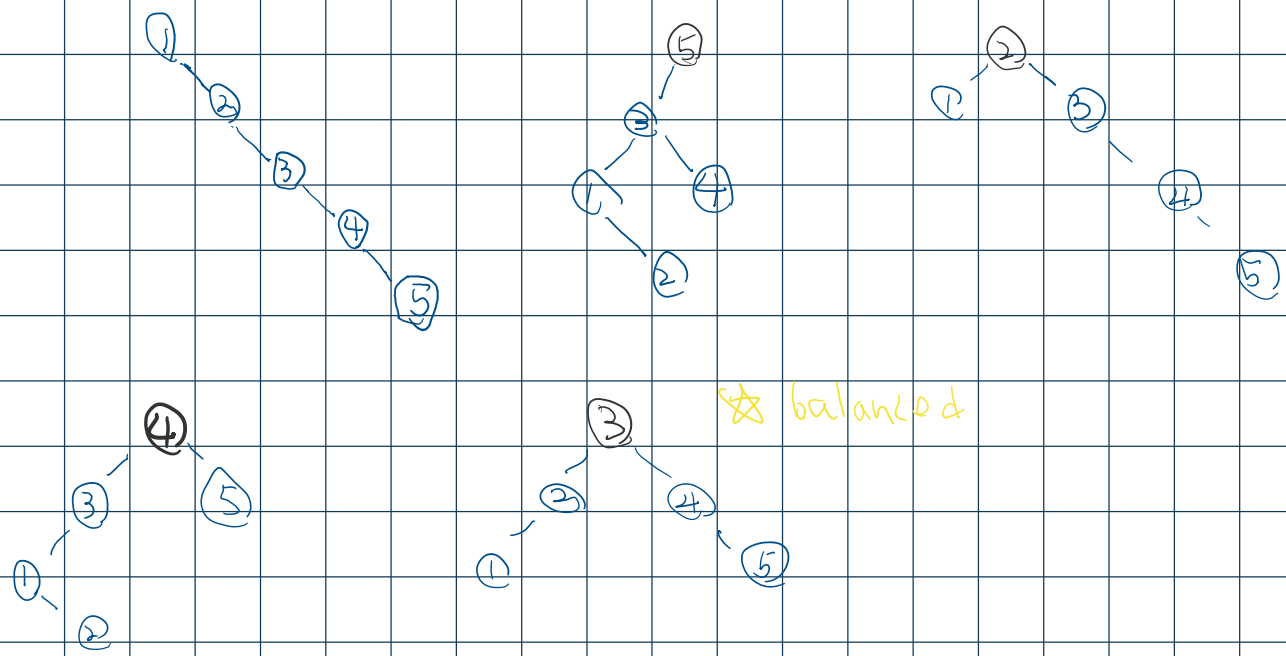
balance = let complexity ≤ O(logN)

換root



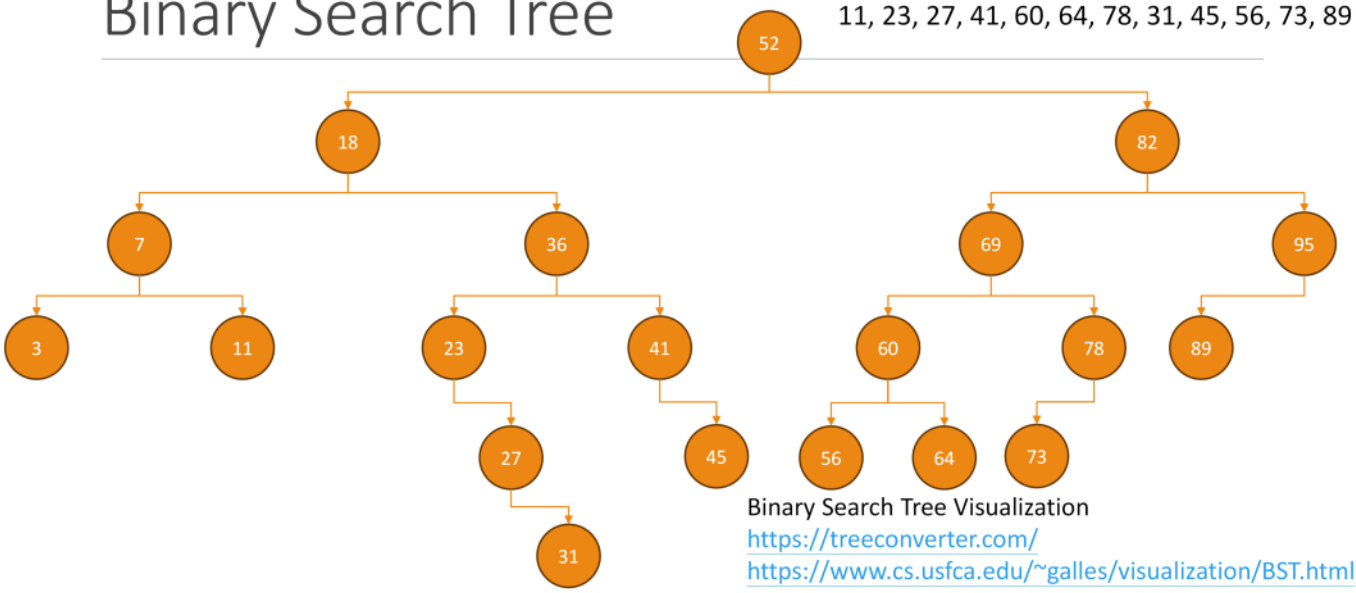
BST

AVL Tree

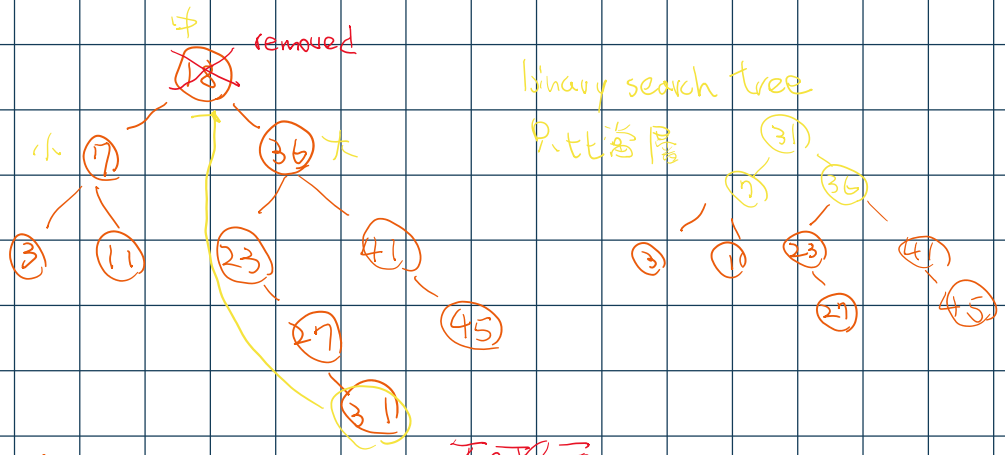


Binary Search Tree

Input integers: 52, 18, 82, 7, 69, 36, 95, 3, 11, 23, 27, 41, 60, 64, 78, 31, 45, 56, 73, 89



Binary Search Tree Visualization
https://treeconverter.com/
https://www.cs.usfca.edu/~galles/visualization/BST.html

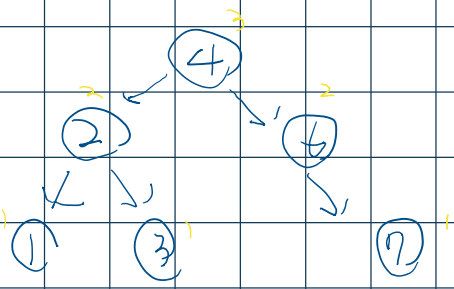


和17行

左子節點中最大數，必小於母節點

⇒找左子節點中最大 or 右子節點中最小

要符合AVL樹兩邊高度差≤1 如何動手?



要有位置(節點高度)

從Binary tree → Binary search tree? → AVL tree?

traverse節點 重建一棵樹

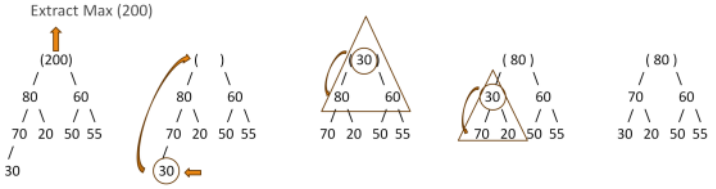
Study Note - Heap

2025年12月8日　上午 09:42

Com

Example: Max Heap (Extract Max)

Extract Max (200)



Array representation: [200, 80, 60, 70, 20, 50, 55, 30]

Array representation: [30, 80, 60, 70, 20, 50, 55] → Replace 200 with 30 (last leaf), heap order violated

Array representation: [80, 30, 60, 70, 20, 50, 55]

Array representation: [80, 70, 60, 30, 20, 50, 55]

當 Root 移除(最大值拿掉)

⇒ leaf移上去(最後一個leaf node往上提)　Guarantee maintain binary tree

⇒ 由上往下調整

Study Note - Graph

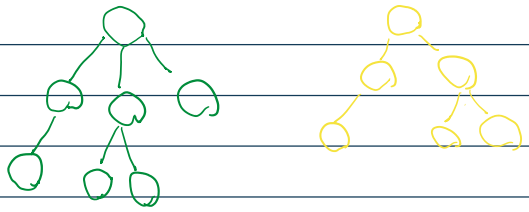
2025年12月8日　　上午 10:00

Tree是Graph的變型　少了hierarchy

HW5

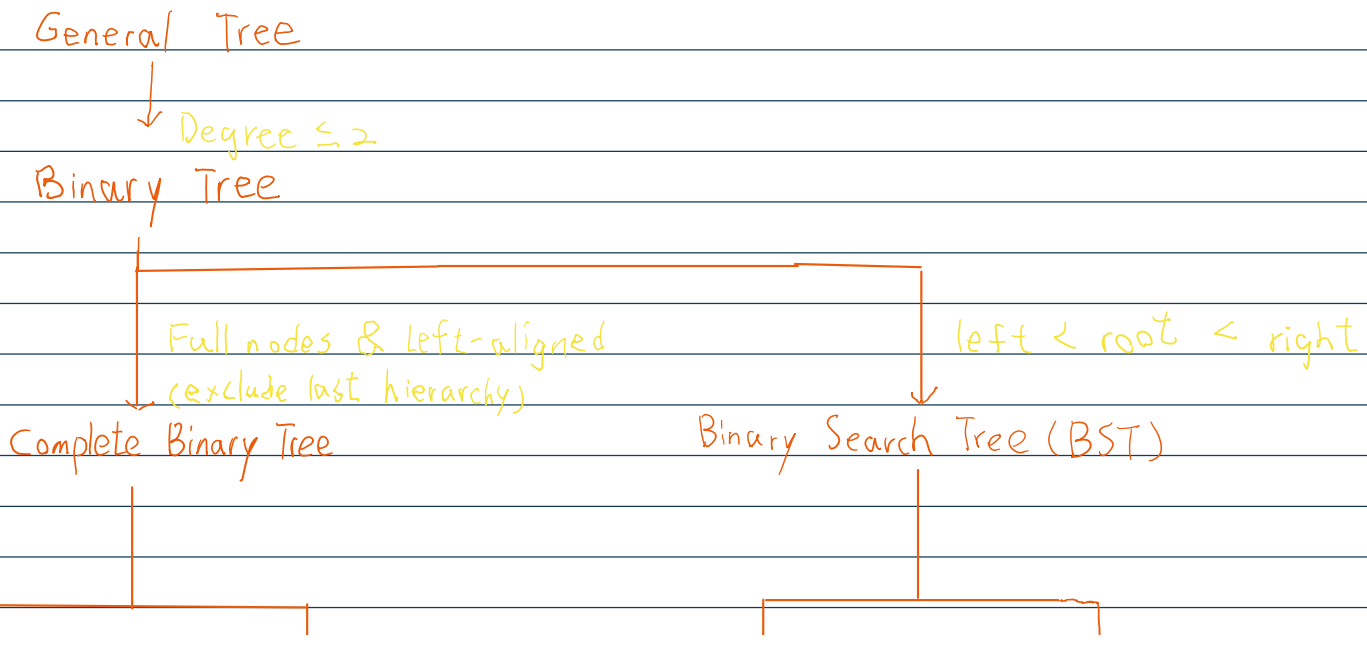
2025年12月13日　下午 07:09

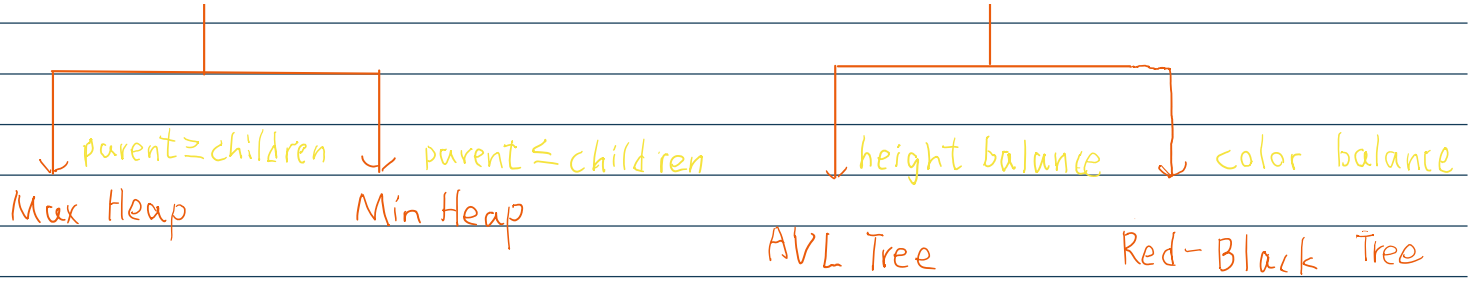
General Tree → Binary Tree → Complete Binary Tree



Binary Tree → Binary Search Tree → AVL / Red-Black

Binary Tree → Max Heap / Min Heap




parent ≥ children
parent ≤ children
Max Heap
Min Heap
height balance
color balance
AVL Tree
Red-Black Tree

12/16 & 12/23 總複習

2025年12月16日　下午 03:13

不同 data structure
⇒ Array
- Linked list
- Stack & Quene
- Tree / Heap
- Graph
- Hash Table — Hash function / Seperate chaining / Open Addressing — Random Access

12/16

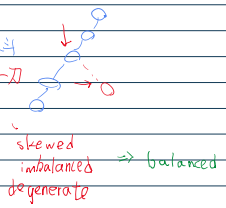
Random Access
⇒ O(1)　O(n)

binary search tree
↑
binary search
↑
⇒ sequential search → sorting → O(logn)

data property

Complexity
- insert
- delete/update
- lookup　O(1)

Sorting — O(n²)　O(logn) 演算法



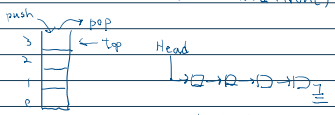
適用於資料數量固定 行為有跡可尋 空間利用率高
優勢:有index Random Access O(1)
關鍵字Hash OpenAddress Collision handling

Linked list

用多少算多少
Traverse issue 只能sequential access O(N)
Sorting無效

☆ insert / delete O(1)
∵ 已知位置

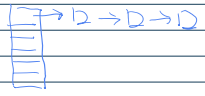
Stack
e.g. function call　more often
implement by Array　4之負責 front, end or cicular array avoid 空間利用率差
Linked list　Head (front)　tail



Array　Linked list

both: push, pop　O(1)

Hash
Array + linked list



bucket

Sequential Search 太慢? 那試試Open Addressing(via random access implemented by array)

linear probing　primary cluster
又作第二次Hash　secondary cluster
重要 index　$\alpha = \frac{n}{m}$
空間利用率

Tree 重要!

general ⇒ binary ⇒ binary search
binary ⇒ complete binary — Heap - priority Queue
degree ≤ 2

Graph
O(V, E)　DFS　BFS
Adjacent Matrix

Graph

O(V,E)

Adjanct Matrix

Ajanct Linked list 有向

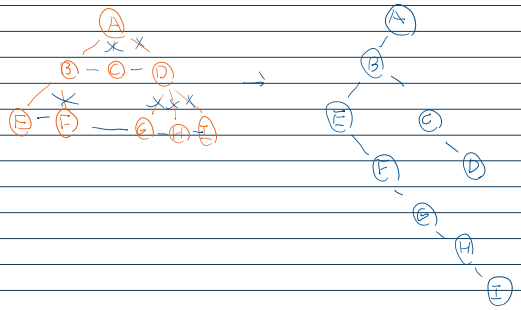
無向要記 both side

DFS

BFS

Tree → Binary Search Tree

如何將 Degree > 2 之分支重新排序？



12/23

Graph

⇒ Graph traversaliy → Path

DFS, BFS

memory path vertex visited

stack queue

特殊性優先順序

Time complexity

Space complexity

Adjacenty: Matrix (2D Array) / List (Linked list)

Tree

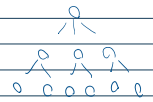
hierarchical data structure

Graph特殊性質 ⇒ no cycle

Root, child, node

tree traversal

BFS / DFS] memory path stack queue



general tree → binary tree

BST

complete binary tree → Heap

Space complexity

Time complexity logn

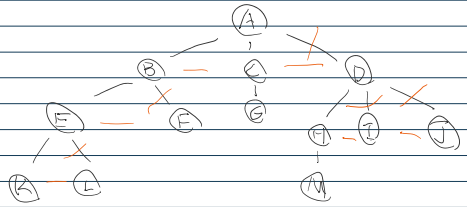
考點整理

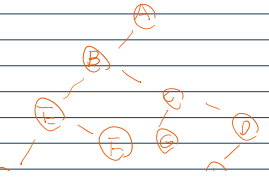
Tree ; General Tree → Binary Tree

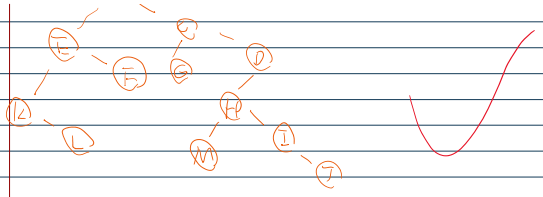
題型

簡答

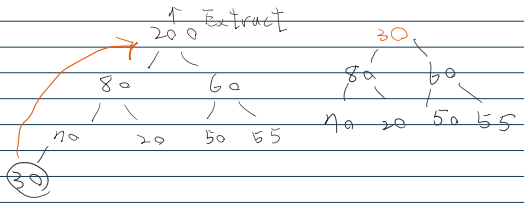
設計







Tree ← shape-based 形狀導向
criteria-based 數值導向



Heap ← Max
Min

	insert	extract		
最下最右 parent交換	Root移除 會最下最右	和	child swap (大)	
最下最右 parent交換	Root移除 會最下最右	和	child swap (小)	

左child 右child

Graph: visited ← Lazy Marking
← Eager Marking