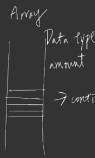


空間利用率

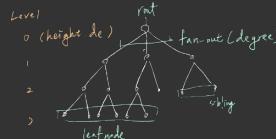
⇒浪費 (hash → collision handling)

static → 固定，已知 size
dynamic → 非固定 continuous) 呈長方式



→ continuous, 位置 + index
Random access
O(1)

o(n) ⇒ sequential search → sorting ⇒ O($\log n$)
↑
binary search
↓
BST (Binary search tree)
if BST imbalance (skewed degree)
⇒ find middle integer (+往數)



Tree

sequential
(linked list)
Hierarchy
(Tree)
□ → □ → A

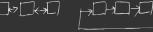
Tree → Binary Tree → Binary Search Tree
degree limitation child and parent related

Linked List

use pointer
→ NULL → link traverse
time complexity

⇒ traverse issue (can't get element by index)

double / circular
O(n)
O(1)
O(1)



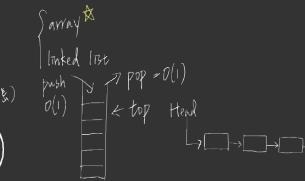
Stack / Queue

- FIFO ⇒ 先進先出 程序

- FILO ⇒ 先進後出 程式 function call



Stack



Hash

⇒ array + linked list



- search, sequential speed up ⇒ Open Addressing or Random Access

空間利用率 < 費
72.9 ± 7.5 best condition

Tree

圖案 for int
general tree → binary tree → binary search tree
Method:
Left child - Right sibling

tree traversal

- preorder
- inorder
- postorder

Heap

priority Queue

Graph

$G(V, E)$
adjacenc Matrix
adjacent List
BFS
DFS
雙向記錄