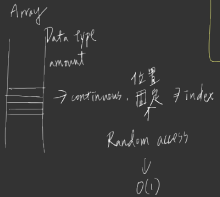


空間利用率

⇒ 浪費 (hash → collision handling)

static → 固定, 已知 size
dynamic 動態 continuous) 宣告方式



Array → integer
insert → $O(n)$ $O(nlogn)$... because of different 演算法
insert
delete
lookup
time, space complexity

$O(n)$ ⇒ sequential search → sorting → $O(nlogn)$

↓
binary search

↓
BST (Binary search tree)

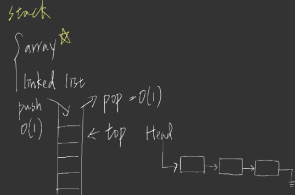
if BST imbalance (skewed degree)
⇒ find middle integer (中位數)

Linked List

use pointer
→ reverse $O(1)$
→ traverse issue (can't get element by index)
time complexity
- search $O(n)$
- insert $O(1)$
- delete $O(1)$
doubly / circular
□ → □ → □ → NULL
□ → □ → □ → □ → □ → □

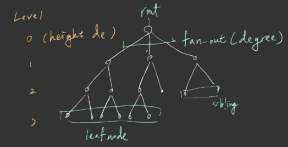
Stack / Queue

- FIFO → 先進先出, 排隊
- LIFO → 先進後出 棧或 function call
Queue array (知道裡面狀態)
array
linked list
front
tail



Tree

sequential (linked list)
□ → □ → □
Hierarchy (Tree)
□
/ \



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

Hash

⇒ array + linked list



- search, sequential speed up → Open Addressing → Random Access
空間利用率 低
72.9 + 75 base condition

Tree

general tree → binary tree → binary search tree
Merkle
left child - Right sibling
complete binary tree
Heap
tree traverse
- preorder
- inorder
- postorder
priority queue

Graph

$G(V, E)$
adjacency Matrix DFE
adjacent list PEG
↔ 雙向記錄