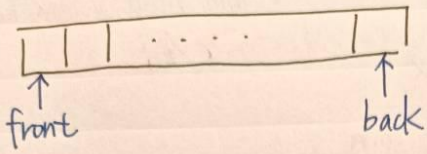
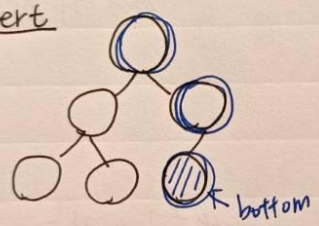
- Queue：排隊，先進先出



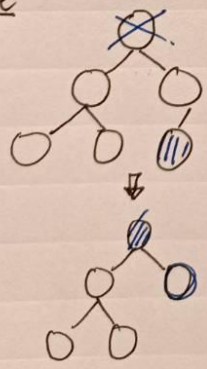
- Heap
  - complete binary tree
  - root 為優先級最高的元素
    - min heap：root 為最小值
    - max heap：root 為最大值

Insert



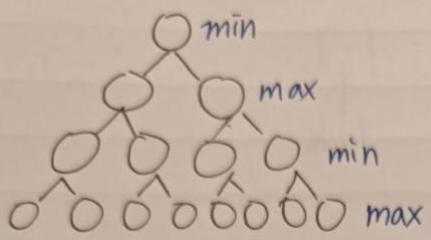
1. 將資料 insert 於 bottom 位置.
2. 往上比較，若優先級較大則往上交換.
3. Efficiency：O(logn)

Delete



1. 刪除 root
2. bottom 移到 root 位置
3. 往下比較/交換
4. Efficiency：O(logn)

min-max heap

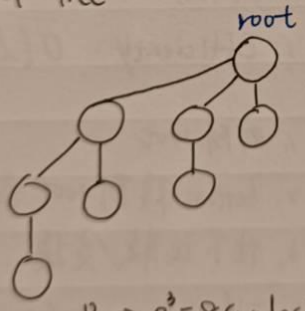


- min heap 和 max heap 交錯

Deap

- root不存放資料
- root左子樹為 min heap
  右子樹為 max heap

Binomial Tree

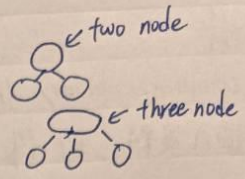


- root有k個孩子
  ⇒ order k (B_k)

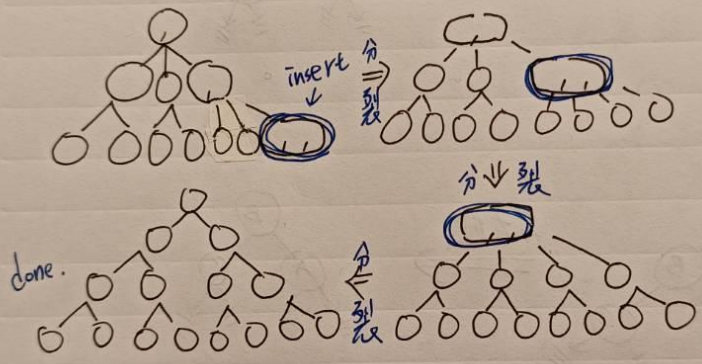
$B_3 \Rightarrow 2^3 = 8$ (nodes)

2-3 樹

- 節點分為 { two-node (1 key) / three-node (2 keys) }



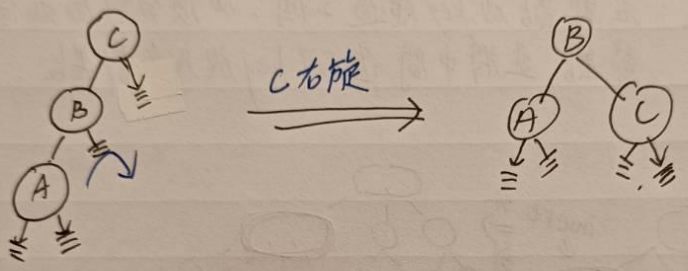
- full tree
- 插入資料：insert 於樹葉節點
- 分裂：若節點內 key 超過 2 個，必須分裂為兩個節點，並將中間值的 key 放入父節點.



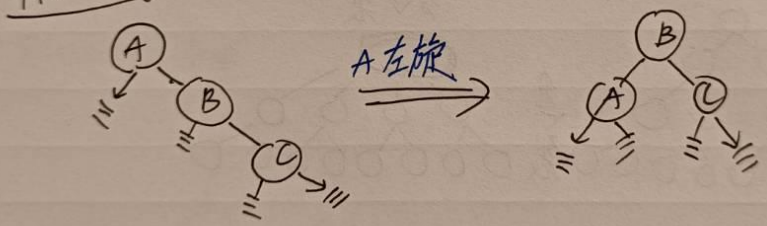
AVL 樹

- complete binary tree
- 透過旋轉保持樹狀結構的平衡

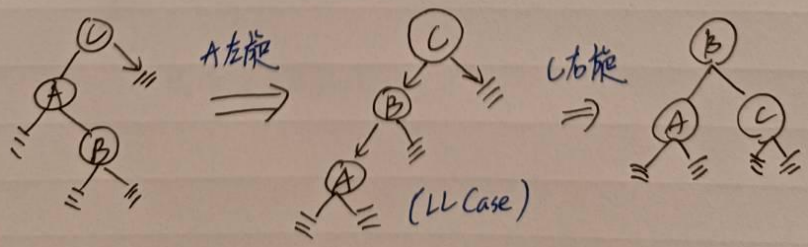
LL Case.



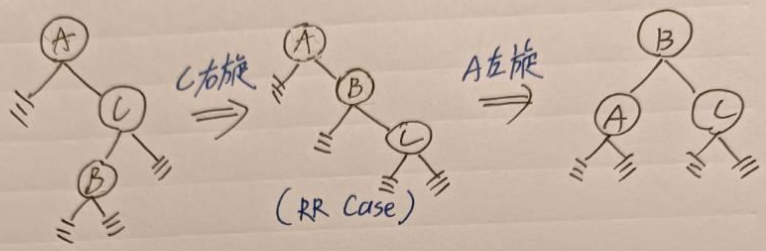
RR Case



LR Case



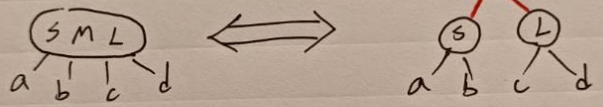
RL Case



2-3-4 樹

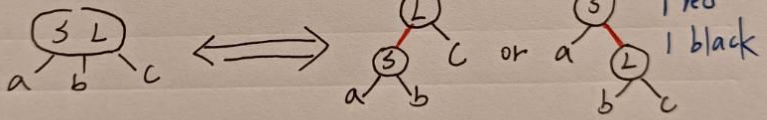
紅黑樹

4-node



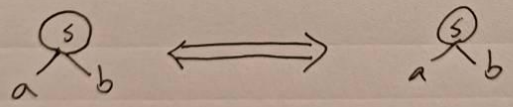
2 red pointers

3-node



1 red
1 black

2-node



2 black pointers

- full tree
- 節點內最多可有3 keys

- complete binary tree
- root 到 leaf 的路徑上不可有連續的紅 pointer
- 每條路徑會有相同數量的黑 pointer