My Notes

Important Concepts worth keeping

Today: / /

Priority Queues

- 和 selection sort 類似 ↗ selection sort : Unsorted List
- 它的 Insert() 為 O(1), Delete 為 O(n)
- 和 Insertion sort 類似 ↗ Insertion sort : sorted List
- 它的 Insert() 為 O(n), Delete 為 O(1)
- Tree sort = Binary search Tree.

Heap

→ Balanced Binary Tree.

- 有 min-heap, max-heap (基本堆積)

What is a heap?

① a complete binary tree.

② a node is greater (smaller) or equal to the values sorted at the children (最大 or 最小)

If you don't know where you're going it doesn't matter what path you take.

– Lewis Carroll

My Questions
Problems & Difficulties needing exploration

- max-heap → 最大的元素一定在樹根
- min-heap → 最小的元素一定在樹根

如何插入元素到 heap 裡？

① 插入新的元素在 next bottom right most place

② 呼叫堆積的 function.

如何移除最大的元素？

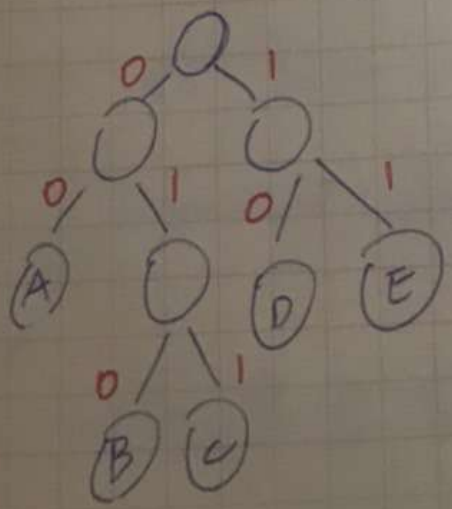
① 先將 bottom 的資料複製到樹根

② 刪除 bottom

③ 呼叫堆積的 function.

My Opinions
Thoughts, inspirations, and suggestions

Huffman coding 霍夫曼編碼



A : 00
B : 010
C : 011
D : 10
E : 11

→ 出現次數越高，編碼的數字要比較少

My Notes
Important Concepts worth keeping

Today: / /

Heap sort 的方法

- 問 n 次，可以知道排序的結果 O(n * logn)（問n次）
- 移除 root，一開始為 swap O(1)，而後來為排序 O(logn)，所以時間複雜度為 O(n)

Min-max Heap

① decide which level → min or max

② 確認需不需要交換

- 效率為 O(logn)
- 如不需交換還是需要往上比.

○ Heap 不適合做搜尋

My Questions
Problems & Difficulties needing exploration

- unsorted array - based
  - → insertion O(1)
  - → Deletion O(n) → shifting data 和資料量成正比
  - → Retrieval O(n) → sequential search 和資料量成正比
- sorted array - based.
  - → insertion & deletion O(n) → shifting data
  - → Retrieval O(logn) → binary search
- unsorted pointer - based
  - → No data shift
  - → Insertion O(1) → 直接加入
  - → Deletion & Retrieval O(n) → sequential search
- sorted pointer - based
  - → No data shift
  - → Insertion、Deletion & Retrieval O(n)

△不一定要先做排序

密碼
cipher key

每當覺得說實話很難，通常正是最應該說實話的重要時刻。

My Notes

Important Concepts worth keeping

Today: / /

△ Binary Search Tree

tree Height 無法保證

→ Maximum - n

→ Minimum : $\log_2(n+1)$

2-3 Tree

• Have 2-nodes and 3-nodes

→ A 2-node has one data item and two children

→ A 3-node has two data item and three children

• 不是二元樹

• 樹比較矮，因有些可放兩個 key.

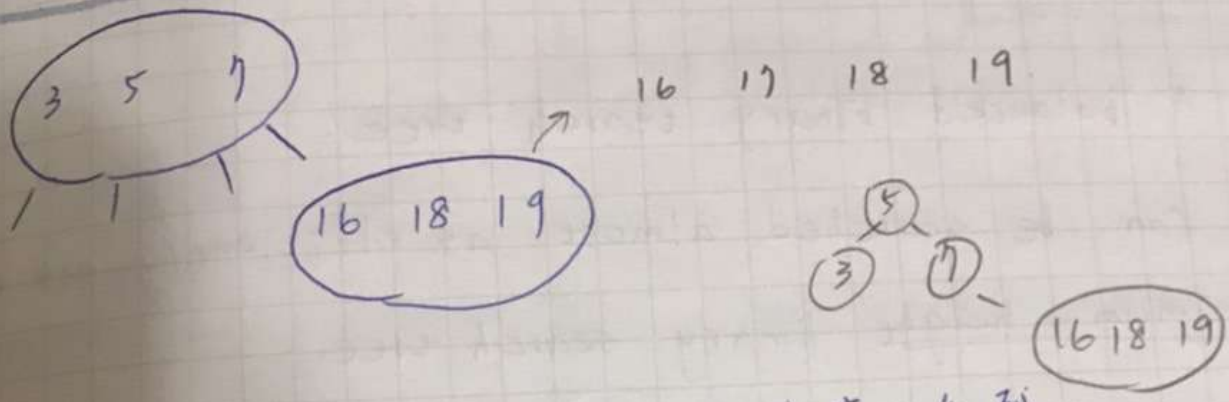
• 左邊小，右邊大.

• a leaf may contain either one or two data items

• 2-3-4 樹效率比 2-3 樹效率好

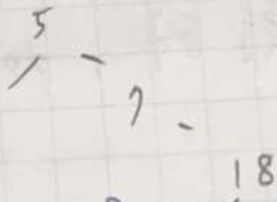
My Questions
Problems & Difficulties needing exploration

2-3-4 樹



→ 避免遞迴，遇到有三個的就分裂

→ 保證 logn 做完



- splits occur only at the path from the root to a leaf
- No recursion (upward)

My Opinions
Thoughts, inspirations, and suggestions

△ 效率好，樹高矮！！！

My Notes

Important Concepts worth keeping

Today: / /

AVL tree

- A balanced binary search tree.
- Can be searched almost as efficiently as a "minimum-height" binary search tree.
- Maintains the tree height close to the minimum

→ 要是平衡的，如果不是要利用 rotate 使它平衡.

△ Balance Factor (BF)

BF(a node) = h(left subtree) - h(right subtree)

→ any node in a binary search tree differ by no more than "1"

→ 如果不是 1 或 -1 代表它不平衡.

My Questions
Problems & Difficulties needing exploration

有四種情況

BF(x) = +2 , BF(x→left) = +1 or 0

① Height of x→left 為 2 , Height of x→left→left 為 1 ⇒ 它為 single rotation with left child (LL)

BF(x) = -2 , BF(x→right) = -1 , 0

② Height of x→right 為 2 , Height of x→right→right 為 1 ⇒ 它為 single rotation with right child (RR)

BF(x) = +2 , BF(x→left) = -1

③ Height of x→left 為 2 , Height of x→left→right 為 1 ⇒ 它為 double rotation with left child (LR)

BF(x) = -2 , BF(x→right) = +1

④ Height of x→right 為 2 , Height of x→right→left 為 1 ⇒ 它為 double rotation with right child (RL)

密碼
cipher key

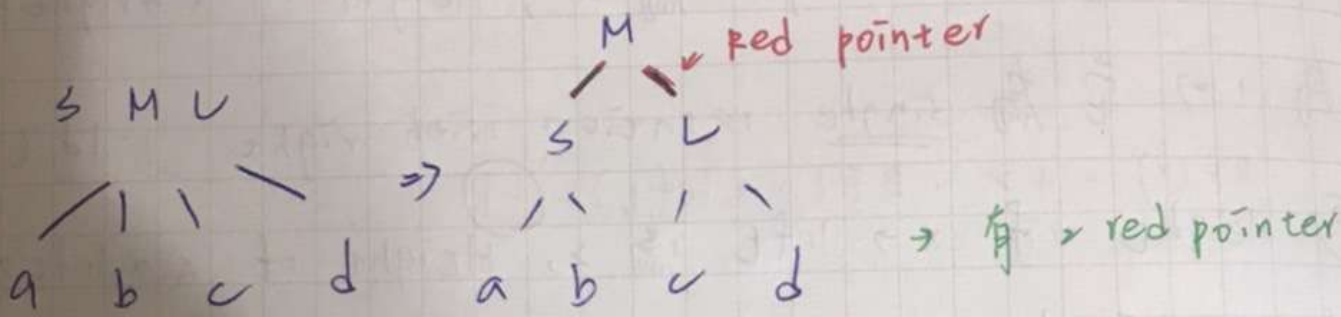
My Notes
Important Concepts worth keeping

Today : / /

red-black tree

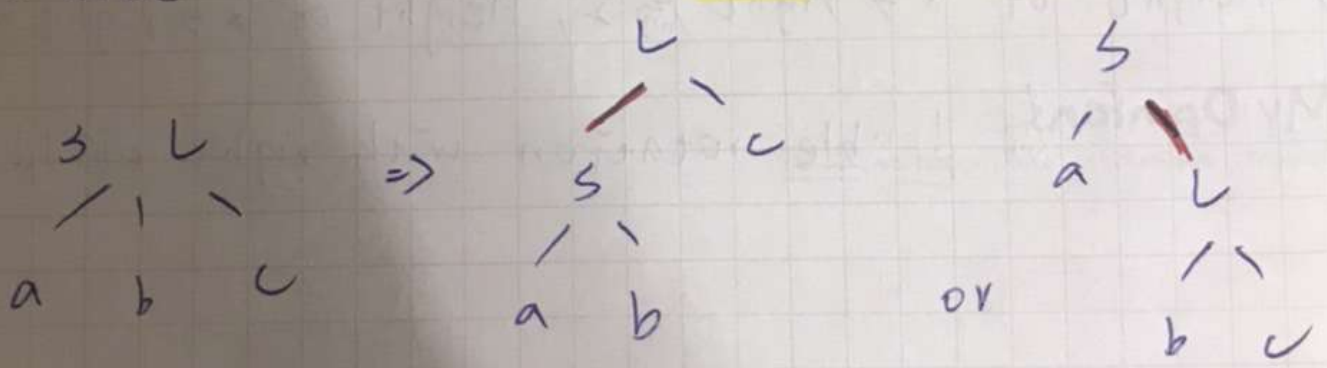
- 每一個 紅黑樹 對應一個 2-3-4 樹

△ 4-node



→ 有 1 red pointer

△ 3-node → Two choices → Not a unique representation



→ 有 1 red 1 black

△ 2-node → 2 black