Unit 1 優先佇列

* Review Sorting Algorithm

Algorithm	Average	Worst
Selection sort	n^2	n^2
Bubble sort	n^2	n^2
Insertion sort	n^2	n^2
Merge sort	$n * \log n$	$n * \log n$
Quick sort	$n * \log n$	n^2
Radix sort	n	n

* Basic of Priority Queue

① Selection sort : Unsorted List

pqInsert() : $O(1)$ → 加入速度快

pqDelete() : $O(n)$ → 太慢、效率不佳

② Insertion sort : Sorted List

pqInsert() : $O(n)$ → 太慢，費時

pqDelete() : $O(1)$ → 已排列好

③ Tree sort : Binary Search Tree

pqInsert() : $O(\log n)$ → 與樹高有關

pqDelete() : $O(\log n)$ → 走至最左邊，(一條路)

 True humility is not thinking less of yourself, it is thinking of yourself less. by C.S. Lewis.

My Questions
Problems & Difficulties needing exploration

My learning weather report

* Application of Priority Queue (找距離最短)
↑ Nearest Neighbor (NN)

Step1、劃分矩形(左上+右下).分區

Step2、比較區域距離 PQ C, A, D, B

Step3、取出區域中最近的城市 PQ A, D, Chicago

Step4、直至第一順位為城市 PQ Buffalo, D, Chicago

* Heap (資料可重新調整, 找最小) complete tree

pq Insert() = < O(n)

pq Delete() = < O(n)

Balanced Binary Tree

① min-Heap (數值愈低愈優先)

保證樹根最小, 其餘不可保證

My Thoughts

pq Insert() : O(log n) ← worst

pq Delete() : O(1)

② max-heap

pq Insert() = O(log n)

pq Delete() = O(1) ← 不適當, 因要改根 ⇒ O(log n)

密碼 cipher key

My Notes
Important Concepts worth keeping

Today: / /

* What is a heap?
① it is a complete binary tree → 資料較緊密
由上而下,由左而右填滿,只可缺右下角
② the value stored at a node is greater (smaller) or equal to the values stored at the children (heap property)

* How to build a heap?
void ReheapDown(int, int);
void ReheapUp(int, int);

* bottom: 下一個要新增的節點.(資料量)

* The ReheapUp function 向上比較(只有一條路)
max-heap pqInsert(): $O(\log n)$

* insert a new element into a heap
step.1 加入 bottom 的位置
step.2 呼叫 ReheapUp()

 不要為明天憂慮,因為明天自有明天的憂慮,一天的難處一天當就夠了。《聖經》

My Questions
Problems & Difficulties needing exploration

My learning weather report

* The ReheapDown function 向下調整 (有2選擇)

  max-heap pqDelete(): $O(\log n)$ worst case. +判斷左or右.

  step1、將 bottom 搬上去 (不影響完整樹)

  ps、2條路選大的 再比較 or 交換

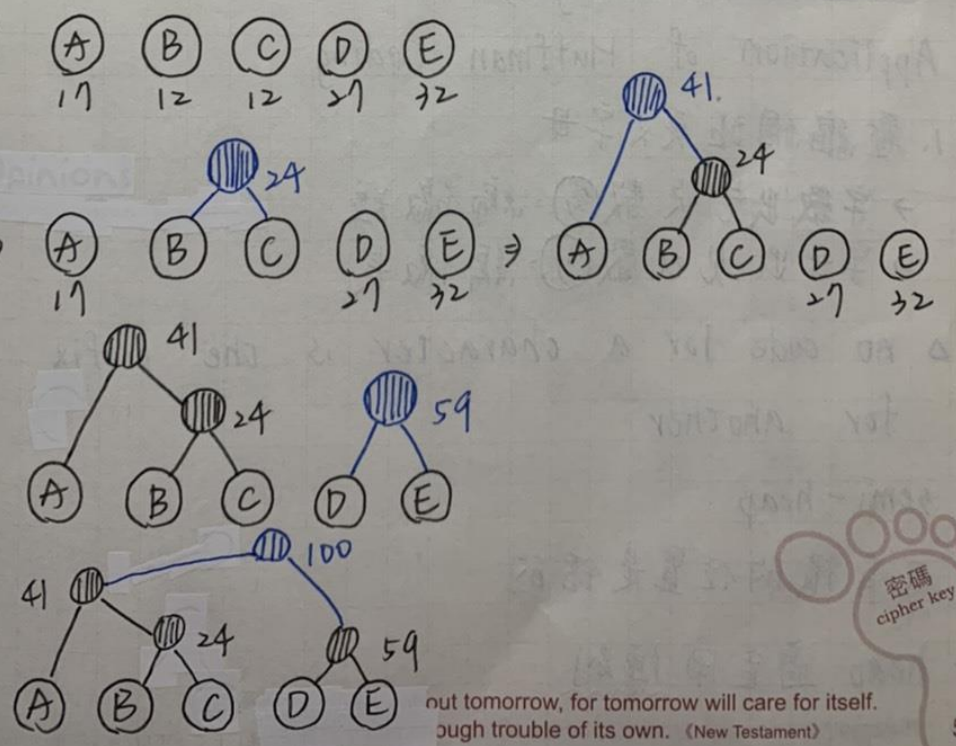
  step2、呼叫 ReheapDown()

* 假設刪中間節點 ⇒ 有取方向!

* Huffman Coding 霍夫曼編碼 (網路 or 影像壓縮)

  取2點最小相加成根.

ex.

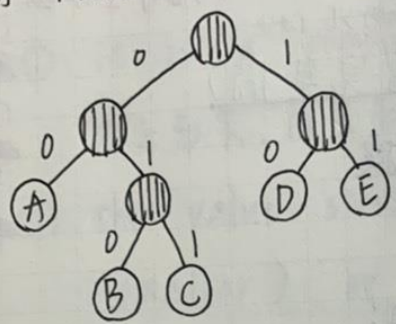


My Notes

Important Concepts worth keeping

Today: / /

* Huffman Coding



A 00

B 010

C 011

D 10

E 11

bit 的壓縮.

* 任一 code 都不能為其餘 code 的 prefix.

* 使用 heap 搭配 Huffman Coding.

* 2個 Delete, 1個 insert, ex. delete 12, 12 insert 24

比排序快. O(log n)

* Application of Huffman Coding

1. 壓縮網站英文字母

→ 字母出現次數多 = 編碼短

→ 字母出現次數少 = 編碼長

△ no code for a character is the prefix for another

* semi-heap

只有根的位置是錯的

* heap 適合用陣列

* heapInsert(): Strategy

1. insert newItem into the bottom of the tree
2. newItem trickles up to an appropriate spot in the tree

Efficiency: O(log n)

。size 就是 bottom

* 找父節點: (n-1)/2 子節點: 2n+1 (左) or 2n+2 (右)

條件: (parent >= 0) && (items[place] > item[parent])

↑ max-heap

* heapDelete(): Strategy

step 1 取 bottom 補 root

step 2 複製 bottom 到 root

step 3、移除 bottom (--size) ⇒ semi-heap.

step 4、將 semi-heap 轉成正常 heap

使用遞迴, heapRebuild(). 往下檢查

Efficiency: O(log n)

* Complete Binary Tree → Heap

call heapRebuild() → 可重複使用

一層一層解決 (本質皆為 semi-heap，根錯而已)

ps、由下往上做.

△ 不可從 0 開始做 ⇒ 做最後位置可修正，且此時左,右子樹已皆為 heap.

* Heap Sort Approach

△ 排序，跑 n 次可得結果 : $O(n * \log n)$

1、刪除 ⇒ the end of the unsorted elements、$O(1)$

2、Reheap the remaining unsorted elements、$O(\log n)$

△ 不會受到資料是否有排序影響

Unit2 堆積變形

* Variations of Heap

△ Double-ended Priority Queues (DEPQ)

- Min-max heap
- Double-ended Heap (DEAP)

△ Forest (union) of Heaps

- Binomial Heap
- Fibonacci Heap

* Min-max Heap → Complete Binary Tree

△ Double-ended Priority Queue (DEPQ)

找最小 ⇒ 樹根 root 找最大 ⇒ root 的子節點之一

* Min-max Heap: Insert

1、決定層數 ⇒ min or max

2、確任是否和其父節點交換

△ No: ReheapUp from the current node 比父大

△ Yes: ReheapUp from its parent 比父小

ps、祖父節點: $(n-1)/2$ 再 $(n-1)/2$

密碼 cipher key

My Notes
Important Concepts worth keeping

Today: / /

* Min-max heap: Delete the smallest
  1、將 bottom 搬至 root
  2、確認子節點(與小的做檢查)
    △ No: ReheapDown from the root (recursion) 根小
    △ Yes: ReheapDown from the root (recursion) 根大
* Min-max heap: Delete the largest
    △ No: ReheapDown from the current node
    △ Yes: ReheapDown from the current node
* 判斷 min-max heap 的 level
  取 log 2!
  看奇、偶
  level = ((int) floor(log2(i+1)) %2)?
* grandparent of item [i]
  if ((i-1)/2 >2)
    grandparent = (i-3)/4
* grandchildren of item [i]
    grandchildren = item [i*4+j] for j = 3、4、5、6.

* Main idea in Min-max Heap

△ Three 4-way trees

- max-heap + min-heap + max-heap
- each node in max-heap has its parent in min-heap

* Double-ended Heap (DEAP) 左右對應，左小右大

△ Insert

1、Left < Right

2、ReheapUp is necessary (recursion)

ps、若遇此數無相對應資料：

☆找右邊父節點比較，看是否需換

☆ step1 左右 check step2、上面 check

△ Delete the smallest

1、Replace the root of min-heap with the last element

2、ReheapDown if necessary

☆ step1. 上下 check.

step2、走到完→確認左右節點關係

△ Delete the largest

1、Replace the root of max-heap with the last element

2、ReheapDown if necessary

3、corresponding nodes: Left < Right

△ 先往下換，再找對應

(若往下還有一層，檢查左heap點的child，若小於其child，則需交換)

* Main Idea in DEAP

△ Two heaps

- Pseudo root + min-heap + max-heap
- each node in max-heap corresponds to one in min-heap、
- Left < Right

* correspond node: levelNo = (int)floor(log2(i+1));

* Which type of heaps does item[i] belong to?

```
levelNo = (int) floor ( log2(i+1) )
```
判斷左右

```
leftOfMaxHeap = exp2( levelNo - 1) * 3 - 1;
type = ( i < leftOfMaxHeap )
```

☆ 考慮邊界

* Where is the node corresponding to item[i]?

```
displacement = exp2 ( levelNo - 1 )
cj = j + displacement * ((type == MIN) ? 1 : -1 );
```

* Application of Double-ended Priority Queues
  - △ External sort
    - Large amount of data on secondary storage
      - eg、quicksort + heapsort
  - △ Merge of priority queues
    - multiple servers : job queues (load balance)

密碼
cipher key

My Notes
Important Concepts worth keeping

Today / /

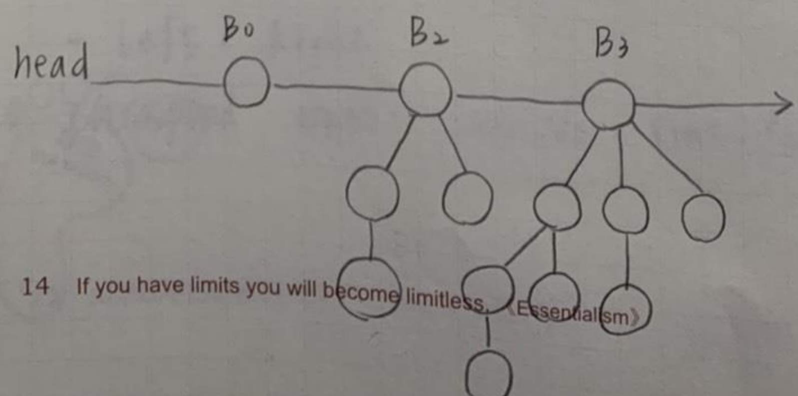
* Binomial Heap : Definition

- A binomial heap is a collection of binomial trees that satisfy the heap property and have distinct orders
  - Two binomial trees of the same order can be merged

ps、k 的意思 → 根的 child 有幾個

- any number = $a_0 2^0 + \cdots + a_k 2^k$
  where $a_0, \ldots, a_k$ are 0 or 1
  Given the number of nodes
  → a unique structure

* Draw a Binomial Heap
  What does a binomal heap of 13 nodes look like?
  $13 = 2^3 + 2^2 + 0 + 2^0 = (1101)_2$



My Questions
Problems & Difficulties needing exploration

My learning weather report

* Binomial Heap: Merge

1、A linked list sorted by the orders of binomial trees (degrees of the roots)

2、Merge two binomial trees of the same order (from left to right)

適合用 pointer 來實做!

* Binomial Heap: Insert

1、Insert into the linked list of the roots

2、Call merge function

* Binomial Heap: Delete

1、Find the minimum from the linked list of the roots、

2、Delete the root having the minimum

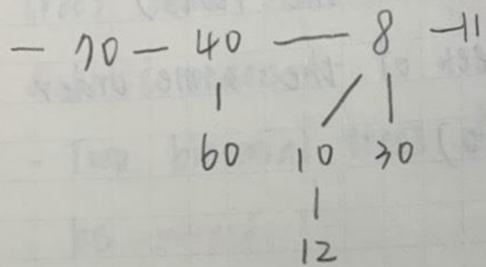
3、Add its children into the linked list

4、call merge function

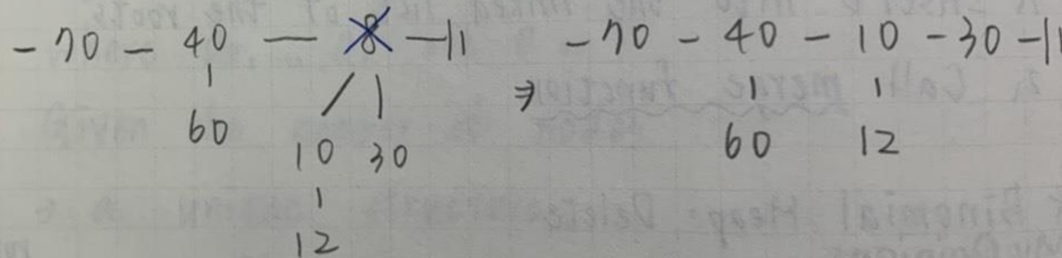
密碼 cipher key

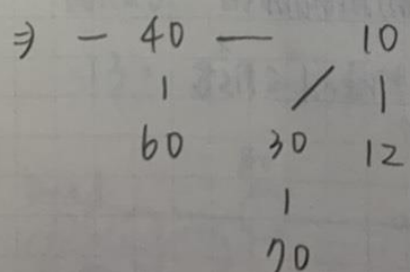
* Insert into a binomial heap

* Input order: 10, 12, 30, 8, 60, 40, 70 (min-heap)



* Delete a binomial heap





\#

My Questions
Problems & Difficulties needing exploration

My learning weather report

* Binomial Tree

△ Binomial Tree of order k (Bk)

- The root has k children
- Merged by two binomial trees of order k-1
- Number of nodes = 2^k
- Tree height = k+1 → O(log n)
- C_i^k nodes at level i, for i = 0…k

* 每個來源必須是 binomial heap.

* Fibonacci Heap: Definition

- Doubly linked list on the siblings (tree roots)
- Doubly linked list between parent and child
- Merge: simply concatenate two lists of tree roots

<table>
<tr><td colspan="3">parent</td></tr>
<tr><td>Llink</td><td>key</td><td>RLink</td></tr>
<tr><td colspan="3">Children</td></tr>
</table>

密碼 cipher key