2019年1月5日 星期六

LeetCode by C++

190105 1st course

- 刷一題5分; 至少刷14題。1/28期限。
- C++ LeetCode課程網站：https://docs.google.com/document/d/e/2PACX-1vRCFxWdo98_buluoXBp7acfID8L-GIUsOONeJ5K7WIVP-ToGHQb1XrHMM16eD6lADmBK_FoYJHvRQ_z/pub
- C++基礎課程網站 : https://docs.google.com/document/d/1nCaieh2ttE-Ym1jU9yLeso5Ynbxd6ZoVD63Rk4dBmMM/pub
- C++資料結構課程：http://c.csie.org/~kez/ds233/
- Latched Top interview list : https://leetcode.com/explore/interview/card/top-interview-questions-easy/
- Leetcode介面跑code超過預期時間會顯示 Time Limit Exceeded ：
  - class Solution {
    public:
    string reverseString(string s) {
    reverse(begin(s), end(s));
    while (true) {}
    return s;
    }
    };
- C++ online doc for String: http://www.cplusplus.com/reference/string/string/
  - 注意class object的constructor, destructor
  - string有預設constructor ：http://www.cplusplus.com/reference/string/string/string/
    (1) empty string constructor (default constructor)
    Constructs an empty string, with a length of zero characters.
  - string[] call string.operator[] 為operator override
- K&R的C過舊? 須看C89 spec的定義：https://port70.net/~nsz/c/c89/c89-draft.html
- 你腦袋的C更新了嗎？by良葛格：https://www.ithome.com.tw/voice/108806
- cout is a object, type is 'ostream' : http://www.cplusplus.com/reference/iostream/cout/?kw=cout
- swap in <utility>：http://www.cplusplus.com/reference/utility/swap/?kw=swap

- ‘complexity’ of time speed for a libarary function is described in cpp-doc : http://www.cplusplus.com/reference/vector/vector/operator[]/

190106 2nd course

- long long 一般是指 8 bytes的int
- 只有 int / int 的整數除法才能取餘數%
- int 最大的 3的次方為116226
- double不能用'==', 一定有誤差epsilon, 小於誤差就算正確
- 記憶體用程式碼無法用完: 會爆，OS會擋掉出現未定義行為或disk會一直swap
- i * j loop O(M*N), i-loop abd j-loop O(M+N)
- big-O分析是N很大的時候才有參考價值，否則機器碼指令數是comiplier決定，無法估算。N為外在要素(參數)，並且是最在意那些的factor。可用於時間複雜度(cpu)或空間複雜度(memory)。
- O(1) better than O(logN) b.t.O(N) b.t. O(NlogN) b.t. O(N^2)
- 能將N簡化成logN通常都是利用二分法，只看一半資料。
- 最有名的複雜度問題是NP and P問題。
- while3的例子可以說成O(logN) 或局限於int的limit O(2^16) ; 空間記憶體為constant: O(1)
- 371題：int有32bits, 所以<<最多32次，O(1); 1st ex.為偽遞回，很容易寫成loophttps://
- docs.google.com/presentation/d/e/2PACX-1vQdW-5FNQdTveLd_TQrsJ2rskjZhSaV4H0cYtI7wWd42b_WG_6fbMLXq4AI-5ETqVVNQd9lyBS-afGH/pub?start=false&loop=false&delayms=3000&slide=id.g4c22e02ed6_0_202: O(k*n)
- https://docs.google.com/presentation/d/e/2PACX-1vQdW-5FNQdTveLd_TQrsJ2rskjZhSaV4H0cYtI7wWd42b_WG_6fbMLXq4AI-5ETqVVNQd9lyBS-afGH/pub?start=false&loop=false&delayms=3000&slide=id.g4c22e02ed6_0_216 : O()
- 考慮空間複雜度時input parameter的空間不用考慮，名為in-place
- CPP有參照&語法，C沒有，僅能傳pointer。好處是可以改函數傳進來的參數call by reference.壞處是CPP需多確認看函式的參數型別，才能確定會不會改變main傳入的參數。raw pointer的問題是取值時一定可以取到值，但取到未定義行為，在生命週期外匯有問題。但參照&沒有這個問題，C++強制在宣告的int &a=b，強制規定a的可視範圍較b小。
- C-pointer算是隨機iterator，CPP靠限制iterator能力獲得安全。實作上iterator是個物件，內含pointer。

- Iterator用法像是C的pointer。但為了取代raw pointer。CPP的目標是只要通過compiler，就不會出現pointer在C-code中可能出現的未定義行為。
- string算是一種容器。取iterator是用begin(string)
- reverse(bidirection iterator): 因此可以用在array / vector / list ; 因為random iterator向下支援bidirectional iter.
-

190112 3th course

- 2-4：C指預設提供array陣列的資料結構: 要考慮時間複雜度，access time需為const。[]裡面是代表距離，所以是從0開始。資料結構會定義其資料的操作方式，以及保證其時間複雜度。
- 動態陣列會有新的陣列元素無法配置在連續處後面，或是原本指標的頭無法再使用。
- 複製新的動態陣列，只少時間複雜度會O(n)
- C++內動態陣列vector，capacity會久久乘以2/次realloc()，平均起來O(n/n)=O(1)
-
- 插入新元素還是O(n)
- 2-5:vector樣板的<int>的class內類別的成員vector<int>::iterator
- C++11後可用auto會依照begin回傳值判定其type
- rotate array簡化演算法用reverse 時間O(n/2+n/2+n/2)=O(n); 空間O(1) in-place
- C++裡的iterator的begin包含頭，end是不包含尾巴，指向尾巴下一個。
- reverse(作用在容器上) 可vector/string 稱為泛型程式/演算法。
- i < 3 == (bool) false; //C++, 但false隱性轉型為0，為了跟C向下相容
- (x<0)會隱形轉型為int 0或1
- C指標或C++的iterator的型別為unsigned_int，都可以相減，表示兩者距離
- C++11後begin()裡面可以丟C-array，以實現泛型(泛用型別)，容器/iterator為一種Design-Pattern
- 2-8:loop終止條件為相撞或fast先到尾吧
- 選1/2是因為cycle要是odd或even。所以可證明第二個循環一定會撞在一起。
- 3-1要小心loop的邊界條件，會不會產生例外。
- 3-2:
- link list不用保證data的連續性，所以插入新元素比vector/array連續的好。
- 最簡單的做法的是比較操作前v.s.操過後的差異，把會更動的元素先備份snapshot起來。
- 刪除節點，先新創一個備份節點在欲刪除節點之後....
- 3-3: C++11內list表示雙向鏈結，forward_list表單向鏈結

- 因為不知道選哪一個演算法或容器實測上會比較快，泛型程式的實作部分可以不用改，只是套用不同容器/資料結構。
- forward_list.reverse(); 因為無法用泛型的reverse(雙向iterator),所以客製化到class的成員函式內。
- C++支援for (int v : a) 語法糖(syntax sugar)　range-base for loop
- 其實C中的[]是語法糖，本質要用pointer * 來理解 3[b] // *(3+b) b[3] // *(b+3)

190113 4th course

- STL(標準模板)內含容器(for資料結構)

	array	vector	forward_list	list(bi-direction)	
operator	O(1)	O(1)	n.a(只能靠 iterator)	n.a(只能靠 iterator)	
resize()	n.a	worst caseO(N)	O(1)	O(1)	
push_back()	n.a	avg.O(1)	O(1)	O(1)	
begin() / end() 給出 iterator	random access/ +3 -3	r.a	Forward / ++	Bi / ++ / —	

- array v.s link_list是兩個極端：造成插入慢(快)存取快(慢)
- STL裡面<deque>有中間選項：一個節點存一片段array
- 3-4: 單向鏈結串鏈是一個一元樹，唯一樹的特例。
- 樹每個單元被0~1個人指到
- 任兩個node的最短路徑只有一條，透過是共有的root
- N個節點，高度最長為N ,ex. linked_list
- N個節點的平衡x元樹，高度為logx(N)
- heap(基本上保證平衡)實務上可稱作piority queue
- 具平衡性binary tree 找最小值/最大值 O(logN)
- 3-4:看到樹，就要想對遞迴實作。
- 遞回的思路是divide and conquer，已知子問題的答案，有一個formula可以知道母問題的答案
- 邊界條件規定遞迴如何結束，樹的例子為樹葉(指向nullptr)，指到空就不再遞迴下去，且葉子規定高度為0
- Bi heap : priority_queu(in C++): 插入新元素為O(logN)
- Bi search tree 找下一個比現在root還大一些，要找右子樹往下找，沒有的話往上找一層。
- 找幾次要看樹的鏈結有幾個邊，如同link_list ，答案為N-1，所以traverse time= O(N)
- 順便限定子樹的最大最小值check
- 3-5: set(in C++)為二元平衡搜尋樹： 插入排序O(logN)，尋找用set.count() 為O(logN)
  set同值只算一個元素。

- begin(set)的iterator因為已經排序過，traverse是從最小尋訪到最大，尋一次還自O(N)
- map<key, value>與set<key>都是二元平衡搜尋樹(實作為紅黑樹)，key值唯一且要能夠比大小。
- 4-1
- two-sum: time complex= O(N*logN) space = O(logN)
- stable def: 一樣點數的數值排序後的順序如同排序之前。
- counting sort : O(N+值堉), 會多開一個O(值愈)的陣列
- radix sort :
- C++的sort是混合各種方法的比較排序法的sort
- 4-2: cmp要定義傳入兩個值時，第一個是否要排在第二個前面: T/False
- 匿名函式lamda可定義寫在要呼叫的地方
- 242.- 用counting演算法或者直接sorting一遍比較
- unodered_map為hash table的實作(常做新增刪除，讀寫較少)

190119 5th course

- Questions:
  - linked-list的ListNode的 constructor()/destructor()語法，或是舊課程的投影片/網路資源。
  - 刷14題是看題數還是同題不同實作也算? 完成期限日?
  - 互動更進一步的話題：職場面試/自己目前狀況實力/台北軟體產業狀況.....
- 面試1題/hr，LeetCode 4題/1.5hr(2~3題大概程度夠)，每週日有比賽可以抓名次(排名中間safe) https://leetcode.com/contest/weekly-contest-120
- 4-3: 二元搜尋樹是資料結構容器，二元搜尋法式對在array裡面排序好的資料而言(可能就等於二元平衡樹，兩者有曖昧的關係)。
- 尾遞迴==自我呼叫時是只在return 出現或是最後一行出現，終止邊界條件會變反向。
- 二元搜尋法的容器用array是因為需要隨機選取 []很快，用二元搜尋樹的容器實作時一樣的意思，因為此容器已經特化過，往左或右都新的中間值
- 二元搜尋法：要先決定搜尋範圍要不要包含中點
- 69題: O(X^0.5),用二元搜尋法O(logX)
- 5-1: hash tabel 的搜尋最好的達到O(1)
- hash function可定義成 mode 13產生索引值; 兩個不同的key丟到hash裡面，產生同一個索引值(分類)，稱為碰撞collision。平均上來說搜尋 O(1)，worst case O(N)全部都碰撞在一起。
- hash index 最直覺的存法是array，理由[]要快。
- heap-memory / stack-memory 遞迴可以用stack改寫 模擬記憶體裡面呼叫函式的方式
- 資料庫的原理
- 7-1: 排列組合問題用巢狀結構窮舉是第一步，第二步享有沒有辦法”剪支”在某一層提前結束。
- 為了要程式碼可擴充(結構漂亮)，可改寫成遞迴。
- 遞迴一定會掀開 stack 存上一個 caller的變數，所以stack-memory有限，系統會結束，與while(1)不一樣。
- 70: 最後一步有可能是1步，或2步這裏兩種可能。先用數學遞回form想，找到邊界條件，由大往回推。再把他改寫迴圈，通常是顛倒順序，從小往上算(從邊界條件開始出發)。先想減少重複運算，在看空間有沒有機會再省。
- 8.1: O(N)變O(logN)通常是用二元搜尋法，O(N^2)變O(N)可考慮hash或動態規劃DP

- DP精神：有兩個變因，有沒有可能固定一個變因後，問題變簡單。通常是一個確定值，而另外一個為遞迴。
- 121: 變因有兩個買/賣, 固定賣的時機(決定買就變成線性搜尋問題)。思考起點為最後一天賣/或不賣，兩個分支出去。一支變成find()，另一支變成遞迴(天數變少)。每次求最小+callＮ次=>O(N^2)。從大開始想或從小開始想也是一個關鍵。
- 191: 暴力法，每一棟搶/不搶O(2^n);
- Greedy難證明，實作思維與DP相反，只考慮最近的前後兩步。