**Leet code刷題紀錄**

2020/06/30

**665. Non-decreasing Array**

沒寫出來

遍歷個元素然後更改值正確。但少考慮何時要修改當前值，何時修改後一值(此關鍵為看當前值與前一值之比較)

2020/-7/01

**633. Sum of Square Numbers**

如果用兩個迴圈會exceed time limit

只能用一個迴圈

最多人使用兩pointer一等於0 一等於sqrt©然後往中間靠

2020/07/04

**154. Find Minimum in Rotated Sorted Array II**

因為有duplicate，所以如果nums[mid] = nums[r]會無法判斷旋轉斷點在哪裡，這時候可以左移一格r

**540. Single Element in a Sorted Array**

可以用^ xor 運算子減少程式碼 nums[mid]==nums[mid^1] 來判斷單一元素在左方或右方

2020/07/05

**4. Median of Two Sorted Arrays**

尋找第(m+n+1)/2與(m+n+2)/2的平均。尋找第k個元素用一recursive function實現，利用二分搜尋法比較num1與num2的第k/2元素。若num1的第k/2元素比較小，則說明第k元素不可能位於num1第k/2元素前，固可將指針增加k/2，並進入下一次遞迴。反之亦然。若其中一array已被搜索完，則直接返回另外一array之第k個元素。若k值等於1，則返回num1及num2兩個初始指針指向之值中的min。

2020/07/06

**O(n logn)：合併排序 (Merge Sort)**

**拆分**

把大陣列切一半成為兩個小陣列

把切好的兩個小陣列再各自切一半

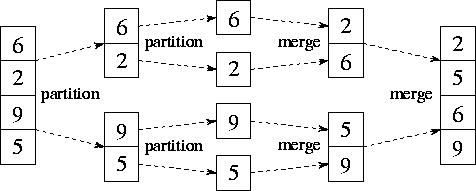
重複步驟二直到每個小陣列都只剩一個元素

**合併**

排序兩個只剩一個元素的小陣列並合併

把兩邊排序好的小陣列合併並排序成一個陣列

重複步驟二直到所有小陣列都合併成一個大陣列



2020/07/07

**75. Sort Colors**

為達到one pass，除了遍歷以外要有兩個指針(一紅指向0，一藍指向2)，red一開始指向頭，當遍歷遇到0時則將此數字與red數字交換，交換玩red指針與i +1；當遍歷遇到2時則將此數字與blue數字交換，叫喚玩blue指針-1；當遇遍歷遇到1時則部動作i+1。

2020/07/08

在Priority queue是以heap實作。**若priority queue內之元素是pair，則以pair中第一個元素之大小作為queue pop的比較依據**

**451. Sort Characters By Frequency**

先用map紀錄出現字數，再放入priority queue。

(如果用內建sort的話可以直接對input sort，但自寫sort function與考慮到若出現次數相同的情況)

**47. Permutations II**

麻煩

2020/07/15

**310. Minimum Height Trees**

原本想法:遍歷每個點然後算出以每個點為root時之樹高，記錄最小樹高以及其root。問題: 跑testcase時會超過time limit，而且若邊沒有出現的數字時樹高為0，會直接變答案，錯誤。

解答建議方式:**利用剝洋蔥法，一層層脫去葉節點，剩下的一個或兩個節點就是答案**。

2020/07/21

**343. Integer Break**

利用dp解。Dp[i]為可得知最大乘積。第一個外圈計算遍歷dp[i]，第二圈內圈計算數字全部拆法(此方法為O(n2))。另外利用找規律(拆出3)

**646. Maximum Length of Pair Chain**

好像不用dp比較快

2020/07/22

**494. Target Sum**

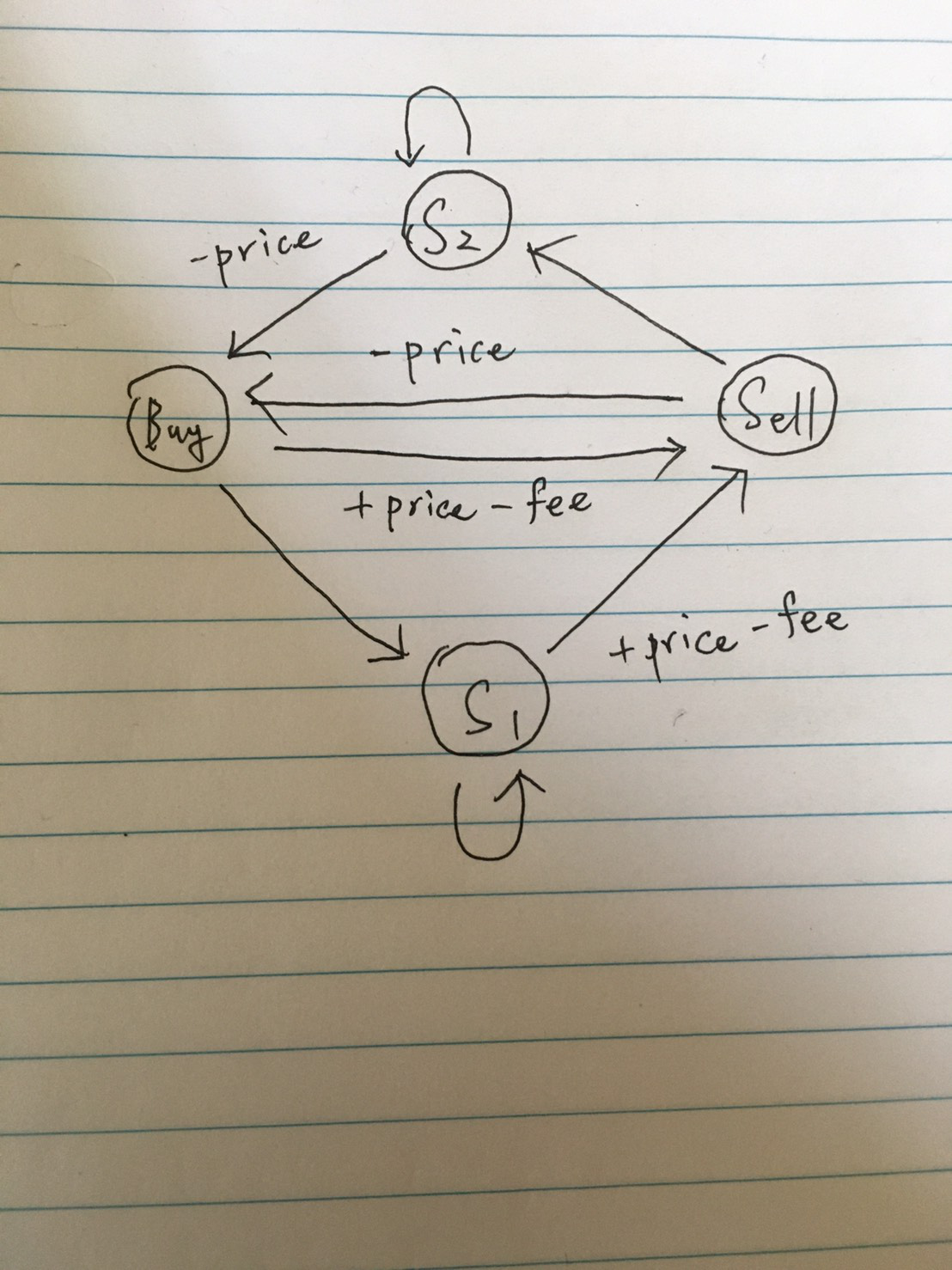
有用recursive function 解出來

其實可以用dp解

2020/07/23

**714. Best Time to Buy and Sell Stock with Transaction Fee**

畫finite state machine解dp



**932. Beautiful Array**

略難，思路:

因为2\*A[k]是偶数，如果要求2\*A[K]!=A[I]+A[J]那么可以构造位置在A[k]左边的全部放奇数，位置在A[k]右边的全部放偶数。这样就保证了对于K位置而言，这个性质是满足的。

然後就從{1}開始擴大Beautiful array，左半邊 x 2 – 1，右半邊 x 2。

**312. Burst Balloons**

要利用 dp 解，暴力遍歷recursion會超過time limit。d[i][j]為在i ~ j 區間可以得到的最多金幣，然後輪流把I ~J 間的氣球當作最後一個打爆的氣球，看哪一個最後打爆可以獲得最多金幣並更新。Dp遞迴關係式:

dp[left][right] = max(dp[left][right], nums[left - 1] \* nums[i] \* nums[right + 1] + dp[left][i - 1] + dp[i + 1][right])

注意: dp 便利要從長度為一的dp[1][1]，dp[2][2]…開始，到長度為二長度為三依序下去。

2020/07/27

**462. Minimum Moves to Equal Array Elements II**

想像把數字畫在數線上，若數字為奇數個，則最短距離數為移動到中間的那個數上；若為偶數個，則最短距離為移動到最中間兩個數之間的區間之任一數。總步數算法為兩端數字為兩兩一對，其距離的數字總和。

\_\_\_\_\_\_C\_\_\_\_\_A\_\_\_\_\_P\_\_\_\_\_\_\_B\_\_\_\_\_\_D\_\_\_\_\_\_

移動距離合為，|D – C| + |B – A|。

**169. Majority Element**

原本方法是先sorting 後(O(nlogn)再累加當前數字，若超過半數則返回答案，若沒有則換下一個數字。

Optimal solution: Boyer-Moore Majority Vote可O(n) 時間 O(1)空間

这种投票法先将第一个数字假设为过半数，然后把计数器设为1，比较下一个数和此数是否相等，若相等则计数器加一，反之减一。然后看此时计数器的值，若为零，则将下一个值设为候选过半数。以此类推直到遍历完整个数组，当前候选过半数即为该数组的过半数。

2020/07/30

在array裡搜尋missing element或重複element時，可以試看使用**二進位either or ^ 的性質，x ^ 0s = x , x ^ 1s = ~x, x ^ x =0。**

**476. Number Complement**

注意在使用 << 1 這種方式時，有可能會overflow。

**260. Single Number III**

將所有數字either or 後的數字為兩個single number的 either or，並找出此數字第一位為一的數字(用diff ^= -diff) 用這個數字來對數字分類必可分出此二數字。

2020/08/03

**218. The Skyline Problem**

跳過 ㄏㄏ

2020/08/04

map:

map.find() 時間為O(1)，若沒有找到東西則回傳map.end()\* 回傳map的最後一個元素的下一個，這個元素不存在，通常用於表示資料已經到達結尾。

Hash map vs map: 實現方法不同(hash table vs binary tree)

**Hash map**

Pro: 收尋容易

Con: 刪除新增麻煩

**Map**

Pro: 刪除新增容易

Con: 收尋麻煩

2020/08/05

**503. Next Greater Element II**

使用單調棧(simple stack)。因為此為環形array，所以先將0 ~ (n-2)之元素複製加在nums後面。同時先把n-2 ~ 0的元素先放入stack中。接著開始比較，若current element >= stack.top()則 pop()， < 則為next greater number。(自己想ㄉ) 網上解答為travelsal nums兩遍 然後做大同小異的事。

**217. Contains Duplicate**

用unordered\_set 若沒看過就放入set中，有看過則return true。也可排序後，看左右是否有相同，好處為space complexity 為 O(1)，壞處為sort會增加時間。

2020/08/07

**697. Degree of an Array**

原本思路是先遍歷一次找出各數字出現的次數，與各數字第一次出現與最好一次出現，再用最大degree中找出最短子array。**網路最好解法(一次遍歷)**：用兩個map紀錄數字出現數字與首次出現位置，並用當前位置減去首次出現位置即可找出子array

**594. Longest Harmonious Subsequence**

我用hash map先第一次loop紀錄每個數字出現幾次，在第二次loop然後每次看key + 1存不存在，如果存在的話把出現次數加上key出現次數，然後loop出最大值。網路最好解法(一次遍歷): 用一次loop 將map存入map中，同時觀察key + 1 與key – 1存不存在，存在則比較有沒有比較大的，邊加入map邊更新。

2020/08/08

**287. Find the Duplicate Number**

網路詳解: **利用快慢指針的概念(fast = nums[nums[fast]], slow = nums[slow])其duplicate number為環的起點。**所以第一個while當fast = slow 時，起點到相遇點 = 環長度，所以這時再把快指針移回原點並改成慢速，再次相遇時即為環點(duplicate number)。※去複習Linked List Cycle

2020/08/10

**313. Super Ugly Number**

因super ugly number都由質數表primes來的。所以下一個super ugly number一定由現有的super number \* prime 而來。故設立一ptr vector，紀錄每一個prime已經乘到哪一個super ugly number。找下一個super ugly number的方式就是每個prime \* 此prime還沒乘過的super ugly number，然後比較找最小值，此值就是next super ugly number，找到後更新ptr (有可能同時有兩個最小值，需同時更新ptr)

※此題同時可使用priority queue解。

**870. Advantage Shuffle**

利用greedy algorithm(策略: 對於每個B[i]，找一個A[i]中大於它且最小的一個值，若沒有則配最小的A[i]給它)。使用multiset作為A之容器(sorted and duplicates allowed) ※若使用兩個迴圈掃會超過time limit O(n2) ，所以再找大於B[i]且最小的一個值可利用multiset.upper\_bound() 來找，且time complexity 為 O(nlogn)

2020/08/11

**160. Intersection of Two Linked Lists**

使用兩個pointer從兩個linklist開始遍歷，遍歷完就去遍歷另外一個，到最後若兩個linklist有交集必會同時達到交點(若無交集則同時最終達到null)