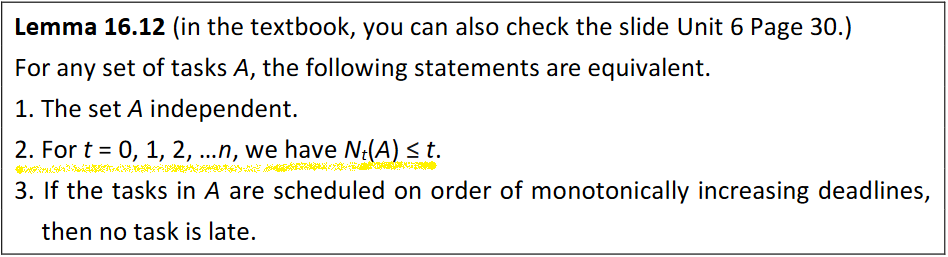
Q1

Define Nt(A):

For t=0,1,2,....,n，let Nt(A) denote the number of tasks in A whose deadline is t or earlier. Note that N0(A)=0 for any set A.

Lemma:



Solution:

宣告一個陣列並初始化B[n]={0,0,0,...,0}。對任意元素 a∈A，B[a.deadline]+=1，若B[a.deadline]> a.deadline，則此集合非獨立結構。檢查A中所有元素，都沒有以上情形則此集合是獨立結構。

Pseudocode:

A[i]=各元素的deadline

B[ length(A)]={0,0,0,0,...,0}

Is\_Independent(A)

for i=1 to length(A)

if( (B[A[i]]++)>A[i] )

return False

return True

Q2.

先把執行時間ti由小排到大，ti最小的task最先做，可得min Ci

Proof ：average Ci =(C1+C2+…+Cn)/n

* (t1+(t1+t2)+…+(t1+t2+…+tn))/n
* (n\*t1+ (n-1)\*t2+…+1\*tn)/n

由於t1出現最多次，所以t1應排最小執行時間，得證

Time complexity：O(n\*logn) //排序所花的時間

Q3

void Machine()

　　　int job[] //sorted by deadline

total\_time = 0

**for i=0** **to size of job**

push job into heap //value of heap = profit

total\_time += job.time

**if** total\_time > job.deadline

heap.pop() // pop out the top in heap total\_time -= remove\_job.time

Sort: O(nlogn)

Heap: O(nlogn)

heap pus一次為 O(logn)，最多做n次push和pop

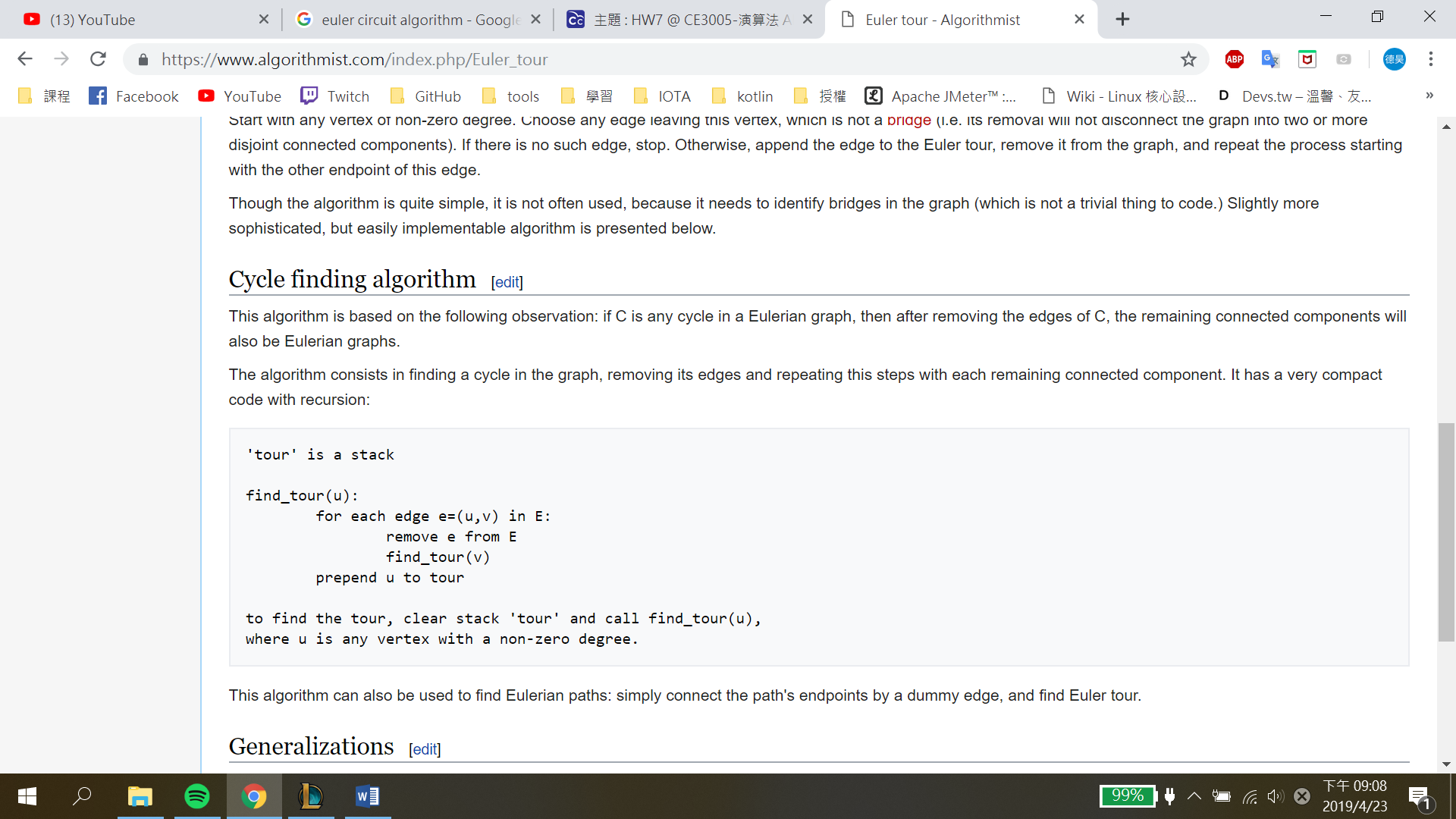
Total：O(nlogn)

Q4

寫出 Euler Circuit的充分必要條件，並設計一演算法找出 Euler Circuit。

無向圖：所有頂點 degree(邊的數量)必為偶數，且互相連通。

有向圖：所有頂點的 in-degree 和 out-degree 必相同，且互相連通。



Time Complexity = O(E)

Q5

(a)

To compute the numbers of out-degrees of every vertex, we need to visit each vertex which takes O(V) time. And then calculate the length of the corresponding list containing adjacent edges (. Therefore, it would take:

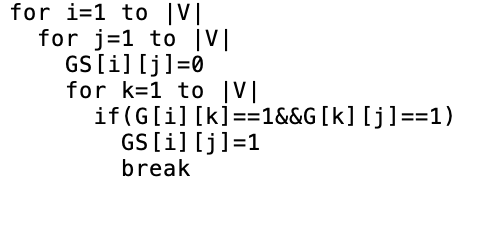
.

(b)

To compute the in-degree of each vertex, we need to visit the whole adjacency-list to count how many times we encounter each vertex. Thus, this operation would take:

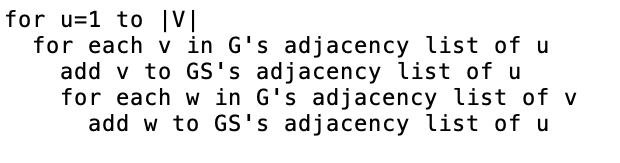
Q6

**Adjacency Matrix**



對於GS[i][j]來說，如果可以找到一個k使得G[i][k]=1且G[k][j]=1的話，GS[i][j]就為1，其實可以看作Ｇ矩陣自己乘自己，上述的psuedocode為O(V^3)，如果改用Strassen去做乘法的話可以更快O(V^(lg 7))。

**Adjacency List**



利用adjacency list的想法為，將G的adjacency list u的所有v加入GS的adjacency list u，再將G的adjacency list v的所有w加入GS的adjacency list u，直到讀完G的adjacency list的所有點，時間複雜度為O(|V|+|E|)\*O(V-1)可化簡為O(|V|\*|E|)，因為|E|大於|V|。

Q7

1. Build a graph whose nodes are wrestlers and edges are rivalries, with adjacency list representation
2. Find a spanning tree with an arbitrary root via BFS. T(n)=O(n+r)
3. For nodes of even level in the spanning tree, labeling them as babyfaces.
4. For nodes of odd level in the spanning tree, labeling them as heels.
5. Finally, check if for each edge, its 2 coincident nodes are babyface and heel respectively. T(n)=O(r)

If true, then it is solution.

Time Complexity : T(n)=O(n+r)+O(r)=O(n+r)