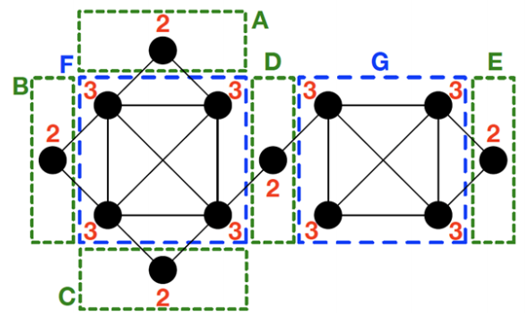
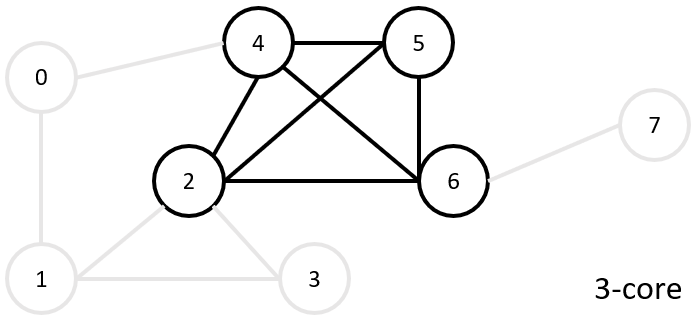
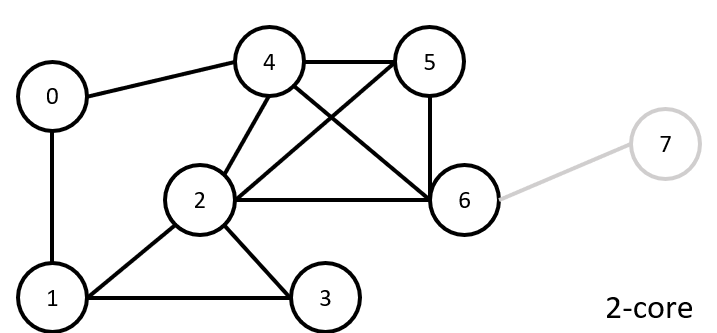
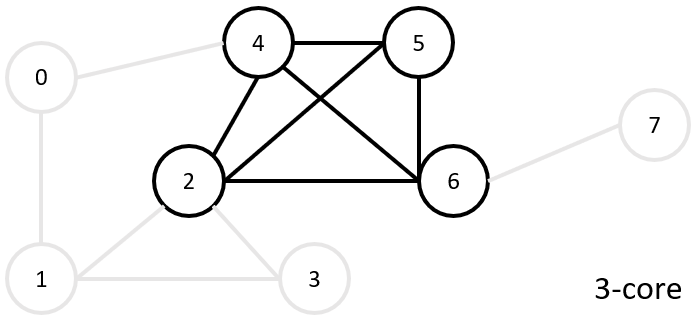
1. How you implement your code

其實一開始我是完全不懂題目給的Kcore例子的。Ppt裡的這張圖片

的藍色F框，我怎麼看這四個點都是4-degree以上，所以應該是4呀，怎麼會全部都是3呢?後來是ppt的P28~29的以下兩張圖給了我很大的啟發。

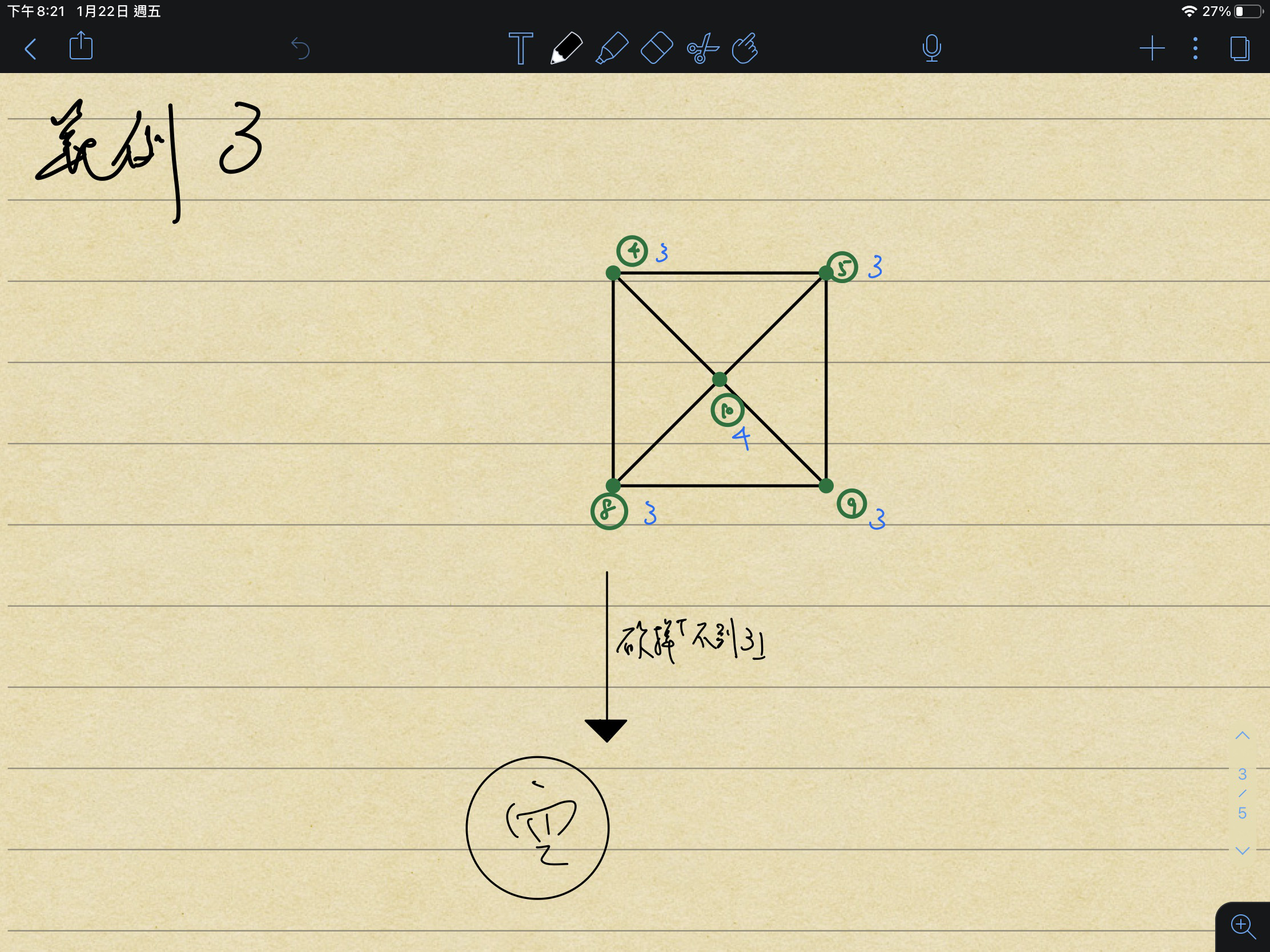


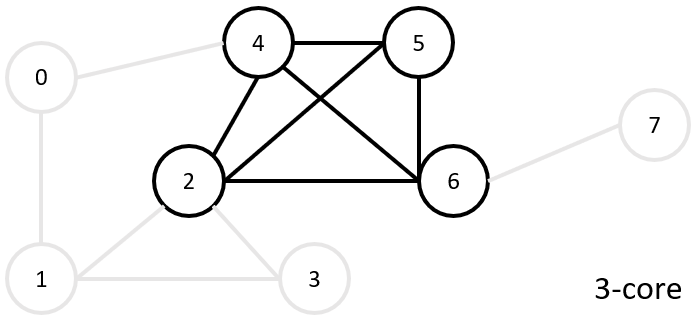
原來是因為在第一輪刪掉的時候，雖然有把不滿k-degree的點全部刪除，但這些點所連出去的edge的終點vertex也會受影響，而這些受影響的終點vertex也許就會因此不滿k-degree，所以也要一併刪除。所以關於K-core的找尋，我就有了結論:「要找特定的k的k-core圖，就把不滿k-degree的vertex刪除，這樣的步驟要一直做，直到某一次做的時候完全沒有把任何點刪除，即可。如果需要計算每個vertex的coreness，那就用迴圈k++把這步驟包起來，得到每個k-core的圖，並且比較、找出k-core和(k+1)-core之間刪掉的點有哪些，即可」。

至於最大Clique，我在觀察好幾個K-core的例子之後(例如[這個網頁](https://www.jianshu.com/p/f495f8219b56))，發現，最大Clique不就會是找到某個圖的有著最大coreness的點所構成的嗎?就例如，

這個圖最大的coreness是3(因為如果算k-core(k=4)的圖，會發現沒有vertex也沒有edge)，而有著coreness為3的點們，就是2、4、5、6，因此這些點，就直接構成這個圖的最大Clique。

雖然上面的K-core算法非常完美，沒有缺陷，但我後來在自己設計的測資中發現Clique的算法有點問題。



如上圖，這五個點的coreness都是3，所以如果我用上面的方法會把這些點{4, 5, 8, 9, 10}全部輸出，但這並不對，因為這張圖裡面，對角線不互連，例如4不接9、5不接8，但是如果我輸出 {4, 5, 8, 9, 10}，就代表4有接9，與事實不符。事實上，觀察後可以發現，大部分的情況是，如果最大的coreness是k，那麼clique就會有k+1個點(例如)，少部分情況(例如我寫的範例3測資)會是k個點。先確認Clique會是多大之後，就可以由蒐集到的有著最大coreness的點們，找到Clique。由範例3的圖可知，如果最大的coreness是k，則我找到coreness為k的點，可能遠遠大於k個，例如可能k+1000個。因此我多寫了一個函式，來尋找精準的「Clique大小」個點。每要把一個點加入之前，就先確認這個點是否和已經加入的所有點都有連接，如果有，這個點才能加入temp\_clique，如果沒有，這個點就馬上捨棄。直到找到精準、不多也不少、剛剛好「Clique大小」個點之後，Clique的找尋就完成了。

1. Challenge you encounter in this project

這次project主要遇到兩個麻煩。

1. 我這個project用了很多vector和set，但一開始都沒有用。一開始是開了很多很多的陣列，大小都寫死是82168個(甚至是二維的82168x82168)，所以在Visual Studio直接說bad\_alloc。我看明明這樣只有用6GB啊!助教說可以用32GB，為啥給我bad\_alloc?不爽!我就偏要用6GB，所以就到Stack Overflow找方法，結果還真的被我找到。原本安安心心寫完了，Visual Studio也不會跳錯誤，結果用makefile配合g++編譯，又出現一次bad\_alloc。心灰意冷之下，想說改用STL好了，沒想到沒用沒事，一用就驚為天人好用!因為我為了很多資訊維護了很多一維陣列，但STL可以直接用函數叫出來，再加上STL很多算法是很好的(遠比我自己寫的好太多了)，所以時間從本來code剛剛好三分鐘(還很怕到底在助教工作站上會不會三分鐘內沒結果)，縮短到我自己設計最機車的測資也只要15秒左右。雖然可能助教的測資更強更容易把我的code打敗，但是要從15秒的規模，把我的code拖時間到超過3分鐘，應該也不算容易。然後記憶體用量也從原本超貪心的6GB~14GB，縮小到大概300MB~600MB。
2. 在Report的第一個問題中，我有寫到「觀察後可以發現，大部分的情況是，如果最大的coreness是k，那麼clique就會有k+1個點(例如)，少部分情況會是k個點」。這個結論是我在範例3發現bug以後，反覆想反覆想得到的結果。因為大部分的圖都不規律，所以clique是k+1個點，這並沒有問題。少部分像範例3的正n邊形的點才會出現狀況。但我又發現，如果正n邊形而且n是奇數的話，因為沒有真正最中心的點(例如n=3是三角形，中心位置沒有edge經過所以沒有vertex。又例如n=5圖形是正五邊形外加裡面有星星，中心位置也是沒有edge經過所以沒有vertex。)，所以就算存在有不是最外圈的點，則這些點因為degree不足，所以很快就會被刪掉，之後連帶影響到最外圈的點也會被刪掉，因此整個圖會分崩離析。因此只有在正n邊形而且n是偶數的時候，Clique大小才會剛好是k。

因為正n邊形有著每個最外圈的點都有連接很多很多條edge的特性，所以我把助教說最多會有200萬條edge拿來計算，用正2050邊形的情況來約略估計我在第一個問題的最後面寫到的「每要把一個點加入之前，就先確認這個點是否和已經加入的所有點都有連接，如果有，這個點才能加入temp\_clique，如果沒有，這個點就馬上捨棄。」的算法會花多久時間，發現大概也只要15~18秒左右，所以還算滿有信心可以在這次project拿到滿分的。

1. Reference that give you idea

(1) <https://www.jianshu.com/p/f495f8219b56>

(2)我和朋友討論的結論