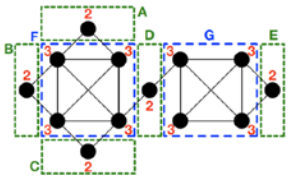
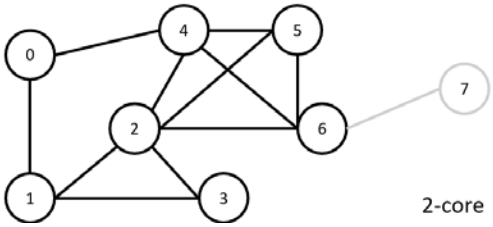
1. How you implement your code

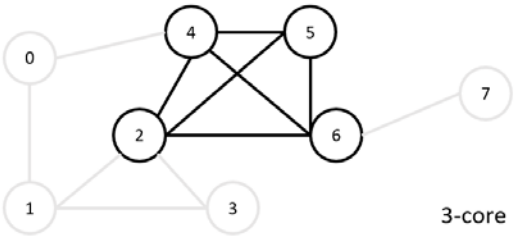
其實一開始我是完全不懂題目給的 Kcore 例子的。Ppt 裡的這張圖片



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。



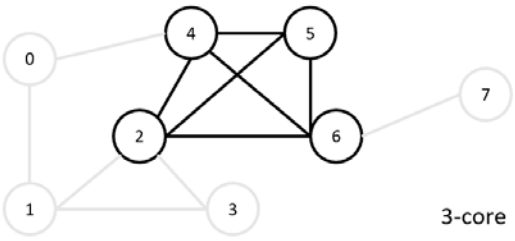
2-core



3-core

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」。

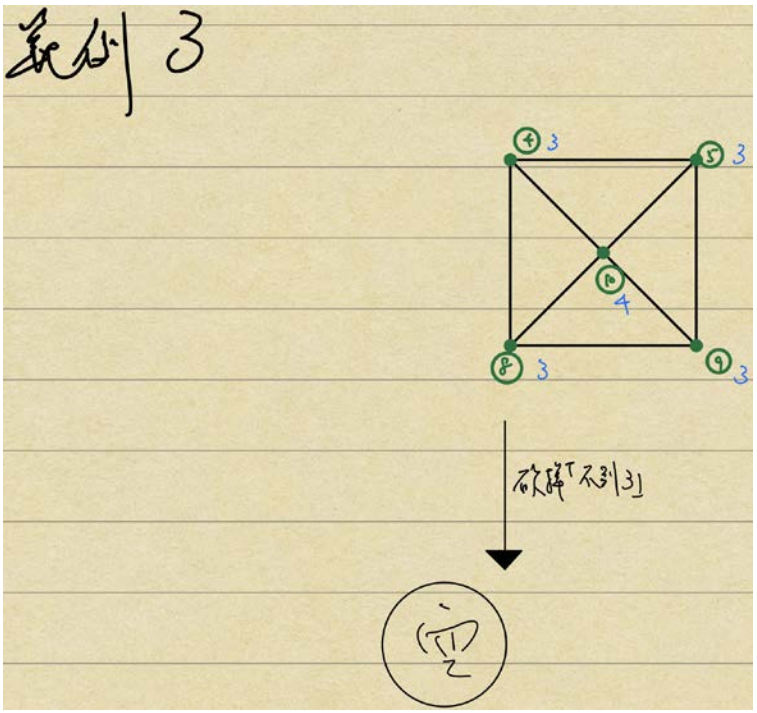
至於最大 Clique，我在觀察好幾個 K-core 的例子之後(例如這個網頁)，發現，最大 Clique 不就會是找到某個圖的有著最大 coreness 的點所構成的嗎?就例如



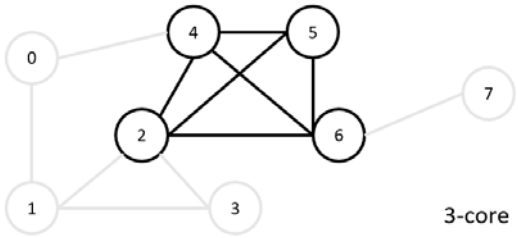
3-core ，

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 3-core)，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。

2. 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。

3. Reference that give you idea

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

(2)我和朋友討論的結論