**----- DS\_Final\_Project Report -----**

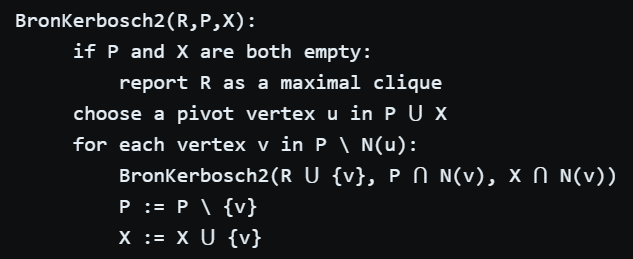
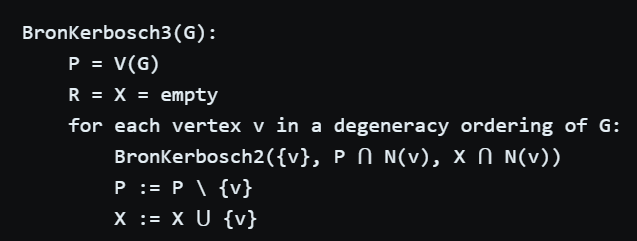
107062313 黃寶萱

**Q1) How you implement your code**

1. For K-core (find coreness)

<方法一> 利用兩個list的二維陣列，一個的index為vertex ID，每個ID的list裡面存和那個vertex相鄰的點，另一個陣列的index為degree，即將每個點放在其對應的degree list裡，接著從degree=1開始，依序將的list裡的所有點刪除，同時也要刪掉其鄰居，接著再更新每個點其所在的degree list，過程中要同時記錄每個點最後屬於哪個degree，其值就是那個點的coreness，便將它存在一維陣列中。  
<方法二>首先，先用set的二維陣列記錄每個點的neighbor，用一維陣列紀錄每個點的degree，從degree最小(min)的點開始移除，將degree<=min的點放進set裡，接著跑過set裡的所有點，同時將其和neighbor的edge移除，重複直到找到所有的coreness。

1. For clique

接續上方找coreness的結果，因為已將每個點的coreness存進一維陣列，再利用網路上找到的演算法求maximum clique，分成兩個部分，With vertex ordering(左圖)以及 With pivoting(右圖)，

R: 目前的clique。

P: 記錄可以增大目前clique的點set，即接下來要列舉的點

（與目前clique上所有點皆相鄰的點所構成的集合）

X: 可以增大目前clique的點集合，但是為已經列舉過的點。用來避免重複取

到相同的點。

其餘的部分只需利用set交集、聯集、差集的用法即可以完成。

**Q2) Challenge I encounter in this project**

一開始是用較直接的方法來找coreness，但因為最初版本的寫法需要跑多層for(int i=0; i<MAX; i++)的迴圈才能找到正確的coreness，跑完open\_testcase\_ds.txt耗時大約介於100~120秒之間，找完coreness後再用Bron–Kerbosch algorithm來找出最大的clique，此情況能符合final project(basic)的時間限制(180 seconds)，但為了完成bonus將時間壓在30秒內，我修正了找coreness的算法，避免跑多層迴圈，新的算法加快了許多，同樣再利用Bron–Kerbosch algorithm，便能在30內找完coreness和最大的clique。除了時間限制上的問題，另一個遇到困難是關於Bron–Kerbosch 演算法，執行過程會有遞迴，但因整個遞迴完成的時間會超過30秒，而超過30秒會終止程式，因此無法在main function 中將目前找到的maximum clique寫入output檔(clique.txt)，我一開始想到的解決方法是除了在main function中可以寫入output檔外，在void signalHandler(int signum)這個function中也可以寫檔，但我不知道為甚麼在signalHandler寫檔當程式終止時會出現Segmentation fault，問過同學後發現他們都將寫檔拉出到另一個function完成，我也試著這樣做便沒有Segmentation fault，請問助教這可能是因為什麼?

**Q3) Reference**

* <https://en.wikipedia.org/wiki/Bron%E2%80%93Kerbosch_algorithm#With_vertex_ordering> (Bron–Kerbosch algorithm)
* <https://iq.opengenus.org/bron-kerbosch-algorithm/>

(Bron–Kerbosch algorithm)

* <https://www.itread01.com/content/1547405646.html>

(c++ set 交集 並集 差集)

* <https://www.itread01.com/content/1546682975.html>

(C++中set用法詳解)