作業四貼文

(一)簡介:

任務一：建立相鄰串列 adjacency lists　　先新增發訊者再一個一個新增收訊者

首先，先確認是否有先前建立的資料庫，如果有便刪掉，先尋找同名的二進位檔，如果不存在同名二進位檔，會顯示錯誤訊息並請使用者再次輸入，成功找到同名二進位檔後，以二進位方式讀檔，並建立資料庫，紀錄發訊者學號、收訊者學號和量化權重。

接著開始建立相鄰串列，先以一個迴圈按照順序，取出資料庫裡的一筆一筆資料以第一個 ID（putID）去建立發訊者學號的主陣列，確認好發訊者學號(不管是找不到而新增的還是找到的)後，再以那筆資料的第二個 ID（getID）去放入收訊者學號，一筆一筆資料跑迴圈去建立相鄰串列，最後，再將過程同時紀錄的主陣列的 putID 的數量和整個串列的 getID 的總量印出並將資料庫寫以 adj 為延伸檔名的文字檔。

建立發訊者的主陣列:

首先，先判斷目前的相鄰串列是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id 和權重，同時也設定此 node 的收訊者、下一個發訊者和後面任務二要走訪用的布林值，最後直接 return 更新完的相鄰串列。

如目前的相鄰串列不是空的，就以一個迴圈去跑相鄰串列的主陣列直到符合三個條件（1.相同的學號字串 2.學號比要新增的學號還大 3.跑到底了）之一，接著判斷如果是因為 2.學號比要新增的學號還大或 3.跑到底了，便再接著做以下的判斷：　　主陣列是依照學號字串由小到大排序

1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在相鄰串列的尾端

2.如果是因為學號比要新增的學號還大且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）

3.如果是在中間找到學號比要新增的學號還大，就從中間插入將資料丟進的同時也同步更新權重（因為是發訊者因此沒有權重，權重都設定為-1）和發訊者數量，最後回傳更新後的相鄰串列

＊因相同學號字串而跳出迴圈者不用做任何行為，因為代表原相鄰串列已有此發訊者學號，他要做的工作在後面放入收訊者資料的部分

放入收訊者：

首先，先判斷目前指到的主陣列的 get 之 node 是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id 和權重，同時也設定此 node 的 get 收訊者(此陣列的下一個收訊者)、後面任務二要走訪用的布林值和更新 node 總量，最後直接 return。

如目前指到的主陣列的 get 之 node　不是空的，就以一個迴圈去跑此相鄰串列

主陣列之收訊者陣列直到符合兩個條件（1.量化權重比待新增的量化權重還小 2.跑到底了）之一，便再接著做以下的判斷： 量化權重是由大到小排序

1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在相鄰串列的尾端

2.如果是因為量化權重比待新增的量化權重還小且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）

3.如果是在中間找到量化權重比待新增的量化權重還小，就從中間插入

將資料丟進的同時也同步更新權重和收訊者數量

任務二 ： 以數量計算影響力 influence

先一個一個新增收訊者再將收訊者接回發訊者

首先要先判斷是否執行過任務一否則沒有相鄰串列去進行任務二，接著一樣去判斷是否有先前建立的資料庫，如果有便刪掉，如果有任務一的資料庫且沒有 dfs 的資料庫便開始建立寬度優先走訪相鄰串列的資料庫。

先以一個迴圈從第一個發訊者學號出發以寬度優先 BFS 走訪相鄰節點，走訪過程的第一步會先將整個相鄰串列的 visited 都先初始化（設為 false），接著宣告一個裝相鄰節點的 queue 將待走訪的 node 丟進去並初始化發訊者的影響力。

用 queue 會先新增完目前發訊者的所有收訊者再將這些收訊者一個一個當發訊者再去新增他的收訊者們

=> 寬度優先 BFS

開頭會先將第一個發訊者設定為走訪過並將他 push 進 queue 裡，接著進入一個迴圈是只要 queue 裡還有東西就不會跳出來，將 queue 裡的 node pop 出來並取得他的 get 收訊者，以此收訊者去跑迴圈將此主陣列之所有收訊者一個一個去找到他對應的發訊者（相同學號），如找得到且是未曾走訪過的便會將此收訊者新增至一個暫時的收訊者串列（每次接會回傳串列之首），並將對應的發訊者設定為走訪過並將他 push 進 queue 裡 ;如找不到便直接將此收訊者新增至一個暫時的收訊者串列（每次接會回傳串列之首）

將目前的發訊者走訪之收訊者新增完後再將目前的發訊者新增至寬度優先資料庫，同時連接第一個收訊者和此發訊者，最後，再將過程同時紀錄的主陣列的 putID 的數量印出並將資料庫寫以 cnt 為延伸檔名的文字檔。

新增至一個暫時的收訊者串列（每次接會回傳串列之首）:

首先，先判斷此暫時的收訊者串列是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id、影響力（收訊者沒有影響力因此為-1）和此 node 的收訊者，並同時更新此收訊者之發訊者的影響力數量，最後直接 return 更新完的暫時之收訊者串列。

如目前的暫時之收訊者串列不是空的，就以一個迴圈去跑暫時之收訊者串列直到符合三個條件（1.相同的學號字串 2.學號比要新增的學號還大 3.跑到底了）之一，接著判斷如果是因為 2.學號比要新增的學號還大或 3.跑到底了，便再接

著做以下的判斷：　收訊者串列是依照學號字串由小到大排序
1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在暫時之收訊者串列的尾端
2.如果是因為學號比要新增的學號還大且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）
3.如果是在中間找到學號比要新增的學號還大，就從中間插入將資料丟進的同時也同步更新此收訊者之發訊者的影響力數量，最後回傳更新後暫時的收訊者串列指向之首
新增發訊者：
首先，先判斷目前寬度優先串列是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id 和更新前面收訊者串列紀錄的影響力及主陣列（發訊者）總數，同時也設定此 node 的 get 收訊者(此主陣列的收訊者)為前面紀錄的暫時之收訊者串列，最後直接 return。
如目前寬度優先串列不是空的，就以一個迴圈去跑此寬度優先串列直到符合兩個條件（1.影響力比待新增的影響力還小 2.跑到底了）之一，便再接著做以下的判斷：　影響力是由大到小排序
1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在寬度優先串列的尾端
2.如果是因為影響力比待新增的影響力還小且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）
3.如果是在中間找到影響力比待新增的影響力還小，就從中間插入
將資料丟進的同時也同步更新影響力和發訊者數量
任務三 ： 以權重估計影響力 influence
先一個一個新增收訊者再將收訊者接回發訊者
首先要先判斷是否執行過任務一否則沒有相鄰串列去進行任務三，接著一樣去判斷是否有先前建立的資料庫，如果有便刪掉，如果有任務一的資料庫且沒有 dfs 的資料庫便開始建立深度優先走訪相鄰串列的資料庫，使用者輸入門檻。
先以一個迴圈從第一個發訊者學號出發以深度優先 DFS 走訪相鄰節點，走訪過程的第一步會先將整個相鄰串列的 visited 都先初始化（設為 false），接著宣告一個裝相鄰節點的 stack 將待走訪的 node 丟進去並初始化發訊者的影響力。
用 stack 會先 push 目前發訊者的所有收訊者再將這些收訊者一個一個當發訊者再去新增他的收訊者們
=> 深度優先 DFS
開頭會先將第一個發訊者設定為走訪過並將他 push 進 stack 裡，接著進入一個迴圈是只要 stack 裡還有東西就不會跳出來，將 stack 裡的 node top 出來並取得

他的 get 收訊者，如果此收訊者是一位符合權重門檻的收訊者，以此收訊者去跑迴圈將此主陣列之所有收訊者一個一個去找到他對應的發訊者（相同學號），如找得到且是未曾走訪過的便會將此收訊者當發訊者 push 進 stack ;如找不到便 pop 出來將目前的發訊者走訪之收訊者新增完後再將目前的發訊者新增至深度優先資料庫，同時連接第一個收訊者和此發訊者，最後，再將過程同時紀錄的主陣列的 putID 的數量印出並將資料庫寫以 inf 為延伸檔名的文字檔。

新增至一個暫時的收訊者串列（每次接會回傳串列之首）:

首先，先判斷此暫時的收訊者串列是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id、影響力（收訊者沒有影響力因此為-1）和此 node 的收訊者，並同時更新此收訊者之發訊者的影響力數量，最後直接 return 更新完的暫時之收訊者串列。

如目前的暫時之收訊者串列不是空的，就以一個迴圈去跑暫時之收訊者串列直到符合三個條件（1.相同的學號字串 2.學號比要新增的學號還大 3.跑到底了）之一，接著判斷如果是因為 2.學號比要新增的學號還大或 3.跑到底了，便再接著做以下的判斷： 收訊者串列是依照學號字串由小到大排序

1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在暫時之收訊者串列的尾端

2.如果是因為學號比要新增的學號還大且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）

3.如果是在中間找到學號比要新增的學號還大，就從中間插入將資料丟進的同時也同步更新此收訊者之發訊者的影響力數量，最後回傳更新後暫時的收訊者串列指向之首

新增發訊者：

首先，先判斷目前深度優先串列是否是空的，如裡面是沒有資料的就直接 new 一個新的 node 將資料丟進去，紀錄學生 id 和更新前面收訊者串列紀錄的影響力及主陣列（發訊者）總數，同時也設定此 node 的 get 收訊者(此主陣列的收訊者)為前面紀錄的暫時之收訊者串列，最後直接 return。

如目前深度優先串列不是空的，就以一個迴圈去跑此深度優先串列直到符合兩個條件（1.影響力比待新增的影響力還小 2.跑到底了）之一，便再接著做以下的判斷：

影響力是由大到小排序

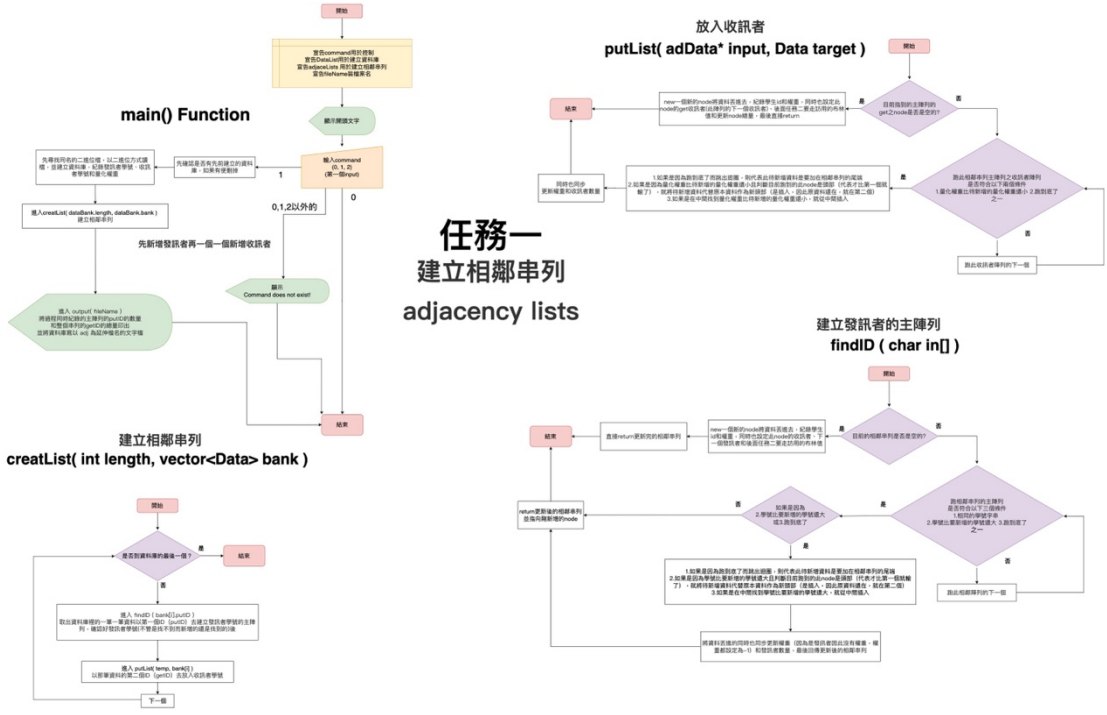
1.如果是因為跑到底了而跳出迴圈，則代表此待新增資料是要加在深度優先串列的尾端

2.如果是因為影響力比待新增的影響力還小且判斷目前跑到的此 node 是頭部（代表才比第一個就輸了），就將待新增資料代替原本資料作為新頭部（是插入，因此原資料還在，就在第二個）

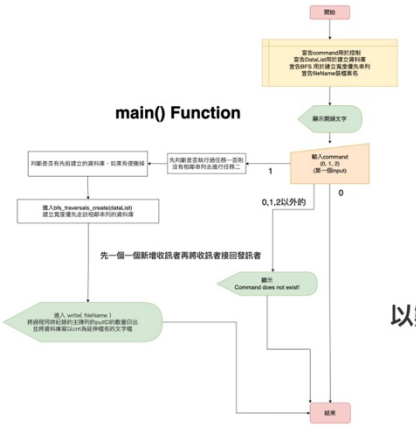
3.如果是在中間找到影響力比待新增的影響力還小，就從中間插入將資料丟進

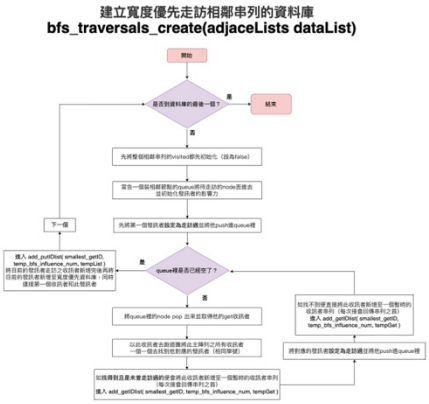
的同時也同步更新影響力和發訊者數量

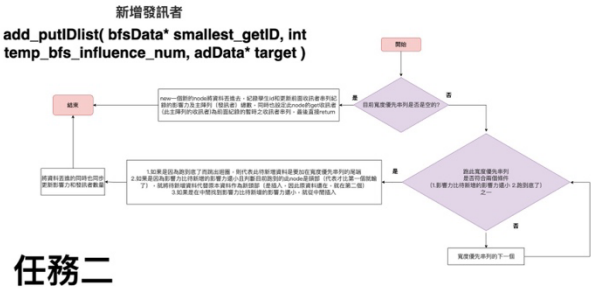
心得:
由於讀檔的部分在上一個任務已使用過我們也相對得心應手因此基本上對於二進位讀檔事沒什麼問題
原本在看完老師影片的我們認為這次的作業可能會比較困難，因為較為抽象不是單靠觀察資料結構便能理解，還要將資料結構轉換成一張有向圖，結果實際寫的時候發現一點都不難且也不需要一定要一個一個將測資畫出有向圖才能完成。
看似困難但其實只要依照老師課程內容建立主陣列及串列，寬度優先走訪用 queue、深度優先走訪用 stack 便能順利完成，因此基本上我們在完成任務的時候幾乎是沒遇到什麼大問題，沒有卡太久的問題也讓我們一致認同這次作業應該是五次之中最好完成的！
(二)圖示:
任務一:



任務二:







任務二

以數量計算影響力 influence