Homework 4 報告 105503518 資工 2 林季勛第一個是要用 Edmond-Karp algorithm 來找 MAX FLOW 還要找出一個 MIN CUT 作法:

1. 創造一個 allcapa 來儲存所有 edge 的 residual capacity，所以要建造 residual net work。
2. 將 edge 讀取進來放到，adj-list 裡面，順便再創造反向的邊，創造的時候檢查是否已經有創造過了，如果有，僅改變 allcapa 的值而已，而如果只是反向的邊的話，他的 pt->s 值會是 0，表示原本不存在，之後 min-cut 的時候會用到
3. 開始尋找 MAX FLOW

   a.用 BFS 的方式將搜尋出最短 augment path 路徑，可以達到 T 的路徑，過程中我用 sQ 來表示 BFS 的 QUEUE，又創造一個 mQ 來儲存這條路最小的 capacity，並且記錄他們的 parent，如果搜尋到 t 了，就跳出來把從 t 到回去 s 上路過的點的 capacity 以及反向的 capacity 分別減掉 mQ 的值以及加上 mQ 的值，最後回傳 mQ。

   b.回傳的 mQ 值如果是 0 的話表示沒有 augment path 了，就回傳 totflow

   c.如果 mQ 的值>0 的話，就把 totflow 加上 mQ 的值，再繼續 BFS 尋找下一個 augment path。時間複雜度分析:

1. 用的是 BFS 的方式尋找所以會花 O(V+E)的時間。
2. 每次最少一個邊會增加 FLOW，每一條 augment path 上有可能有 E 個邊跟 V 個點組合，所有會 EV 個 augment path，
3. 所以要做 EV 次 BFS=O((V+E)*VE)=O(VE^2)

第二個是要找 flow 增加最大的路徑開始找 augment path 作法:

1. 紀錄所有的點最大的 capacity，再生出 residual network 的時候順便記錄下最大的 capacity 是誰
2. 根據 max_capacity 找一個 2^N 的 delta 剛好最接近 max_capacity 而不超越他
3. 更改上面 edmond-karp 的程式碼，每一次 BFS 放入 Q 中的判斷從至少要大於 0 改成只少要大於 delta。
4. 直到第三點 BFS 找不到 augment path 的時候，把 delta/=2;再進入 BFS 不斷找 augment path，直到最後 delta 變成 0，再變回原本 EDMOND-KARP
5. 最後再輸出 FLOW 時間複雜度分析:

1. 會有 Lg C 次進入 BFS 的階段，以及因為都是找最大值，每一次都會有一條邊滿，最多會有 E 個 augment path。
2. 所以時間會是 O(LgC(E^2))

兩個演算法的比較:

第二個演算法是第一個的改進，因為第一個演算法就是窮舉找出一個最小的路徑，但是 FLOW 的大小能夠保證，所以有可能會每次都只加一點點，第二個演算法就改進了第一個演算法的缺點，先由最大的開始找，就不會有只加一點點的狀況。

Min cut 找法:

1. 跑完兩個方法後，兩個的 allcapa 就會儲存著全部的流量，根據這個流量我做 DFS，把從 S 可以走原有的邊可以 VISIT 到的點都記錄下來，在開頭設置邊的時候，原有的邊 pt->s 會設置為 1，所以看的出來
2. DFS 完以後，在根據所有有被 VISIT 的點的所有原有的邊，找出滿的，終點又是指向沒有 VISIT 過的邊，這個邊就是一個 bottle neck，把它輸出出來。

時間複雜度:

1. 先 DFS ，所以時間是 O(E+V)
2. 然後所有的 E 都比較一次是 O(E)
3. 所以總共 O(E+V)