演算法 PA3

電機二　B12901075 賴禹衡

一、資料結構

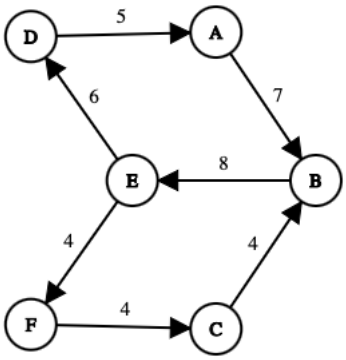
這次的作業比較特別，我使用二個 vector 來分別對應有向圖與無向圖的邊，原因是我最一開始在做有向圖的 cb 時，不知為何只要是 `vector<pair<int, pair<int, int>>>`的儲存方式就容易出現問題（滿有可能是我沒有做好版本管理，而且實測出 performance 會比 bucket queue 弱，不過那跟優先處理的邊有關，可能是 greedy 要再加上討論同樣重的邊應該要如何判斷），但在重做許多遍以後都沒辦法得到好的 performance，因此無向圖就繼續保留 `vector<pair<int, pair<int, int>>>`的儲存方法，但有向圖的處理我則改用 bucket queue：`vector<pair<int,int>> edges[201];`

這樣的情況下 w = index -100，也不需再排序。

二、演算法

首先針對無向圖就很簡單的要做最大生成樹，而要完成最大生成樹，我之前在學習 mst 時，有在網路上看到用 disjoint set 來完成，只需要把邊從原本最小的邊開始，改成最大的邊即可，所以我就用了最一般的 disjoint set 來實施，而針對有向圖的 greedy 方法，延續無向圖的做法，我想到一個絕對不是正解但滿好實施的方法（也

滿好寫的），就是先把圖先視為無向圖，找到最大生成樹（方法跟無向圖的一樣），然後從權重最大的邊一次放回一條，如果產生環（用 dfs 來判斷）就記下來當作要 removed 掉的邊，權重小於零的也直接全部 removed，因為這樣才有最小反饋弧，這樣雖然在一些特殊的 case 會壞掉像是下面這個例子，但在大多 case 都應該能有不錯的表現。



三、複雜度分析

分析一下無向圖：

排序的時間複雜度：$O(mlgm)$

找出最大生成樹，與一般使用 disjoint set 來找最小生成樹方法無異，總共的時間複雜度為$O\left(mlgm\ +\ m\left(a(n)\right)\right)$

有向圖：

建立最大生成樹，$O\left(m\left(a(n)\right)\right)$（因為使用 bucket queue）

處理 deferred_edges，最多要做 m 次 DFS，$O\big(m(m+n)\big)$

總共大概是$O\big(m(m+n)\big)$

最壞的情況是$O(n^4)$