PA3 report

B08901041　電機三　王昱翔

Data Sturcture

在儲存edge資料時我用一個 3xE 的二維矩陣，之後用struct *DisjointSet*來表示vertex之間的關係，並且使用Union by Rank和Path Compression來改善complexity。另外第三種case額外用一個叫做*graph*的class來紀錄整張圖，用Linked list來紀錄要加選取的edge，其中Linked list用的資料結構是`vector<list<int> >`（Disjoint Set只考慮Vertex所以無視edge是否directed、class *graph*會考慮edge是否directed）

Algorithm

使用課本教的Kruskal's algorithm的概念，一開始先把所有的edge load進來，並且對他們的weight進行sort，我使用的是PA1中自己寫的HeapSort，因為他的worst case complexity是 $O(nlgn)$，一開始先對每個vertex做Make-Set，之後再不斷用Union-Find的方式逐漸合併

undirected：

從weight最大的edge開始選取，如果兩個vertex在同一個set中就不進行Union，而且把那個edge加到vector *del*中，代表我要把他delete掉

directed：

一樣先從weight最大的edge開始選取，首先先判斷兩個vertex是不是在同一個set中：

- 不同Set → 代表為了達成WCC一定要把他加進去，所以就把此edge加入graph中
- 同一個Set → 判斷edge weight：
  - edge weight ≤ 0 → 刪掉這個edge會使 total cost 更小，所以刪掉他
  - edge weight > 0 → 判斷是否有cycle：
    - 有cycle → 不符合acyclic的要求，刪除此edge
    - 沒有cycle → 把此edge加入graph中

用此判斷式依序由大到小跑完每一條edge即可

Finding

這次PA最滿意的部分就是我把code寫的滿模塊化的，因此整個架構比較易懂，此外也分成三個cpp檔和一個h檔，分開的寫法使我debug時方便許多，更快找出錯誤在哪。

另外也發現同學用了別的方法，但是某些case比我好，某些比我差，我在猜是sorting時相同weight的edge的排列順序不同所以會有影響，不過我也不知道該如何優化才能每個case都比同學的好。

這次PA也試著使用了以往不熟的結構，像是class, list, vector還有Linked list和Disjoint Set，也因此對這些結構的實作更熟悉了。