PA3 report

1. Cycle breaking 實作所使用的資料結構與演算法

(1) 讀檔存 edges 與 vertices 資訊：在程式中我設計了三個自定義的 class，分別是 Graph、vertices 與 edges 這三個，在 Graph 中的 data member 與 function 主要就是負責實作整張圖的演算法還有一些 helper function，vertices 與 edges 主要就是存一些實作演算法時會用到的重要資訊，例如 predecessor、adjacent list、weight 等。在讀取 input file 的時候，我會根據讀進來的是 undirected graph 或是 directed graph 去拆分成兩個不同的 case 去實作(詳細方法在後面)，但讀取時紀錄資訊的方法都是先將所有 vertices 存進 Graph 中，edges 的關係與權重也先存到 graph 中但先不要接到 vertices 上(也就是先讓所有 vertex 的 adjacent list 為空)，以便後面的演算法實作。

(2) **找到 cycles breaking problem 的 solution 所使用的演算法：**從這裡開始 undirected graph 與 directed graph 的方法會不一樣，主要是針對這兩種圖各自的特性去採取相對應的措施，以下方開討論。

Undirected graph：

Undirected 的部分我所使用的演算法是 Prim's algorithm，因為要找到一個 total weight 最大 acyclic graph 其實就是找到這個 graph 的 maximum spinning tree，所以這邊我藉由在一開始所存的資訊，將所有 vertices 的 adjacent list 填上，接著自己修改了 PA1 所寫過的 MaxHeapify、HeapExtractMax 與 HeapIncreaseKey 這些 helper function，一開始先將所有的 vertices 都存進 MaxHeap 裡面，把 v.d 當成是 MaxHeap 的 key，一開始初始化時先將除了 source 之外所有 vertex 的 d 這個 property 設為-101，將 source 的 d 設為 0。接著，在 while 迴圈中每次都 extract max 一次將 v.d 最大的那個 vertex 取出 MaxHeap，然後看這個 vertex 的 adjacent list，藉由 HeapIncreaseKey 更新這些 vertex 的 d。此外，加入一個判斷條件，若是這時該 vertex 已不在 MaxHeap 中(已經在 MST 中)，則代表會形成一個 cycle，這時不把這個 edge 放進 MST 之餘，就把這個 edge 記錄下來，最後存在這個 vector 中的所有 edges 與 total cost 就是我們所要的答案。

Directed graph：

Directed 的部分我所使用的演算法是 Depth first search，一開始我也是先引用了 PA1 所寫過的 QuickSort，把所有的 edges 先全部由大排到小 sort 過一次，然後再根據這個順序一條一條 edges 插入到 graph 之中，假設這時要插入的 edge 是(u,v)，就先對 v 做 dfs_visit()，看 v 能不能走到 u，

若是 v 可以走到 u，代表當(u,v)這條 edges 被插入到 graph 當中的話，就會形成 cycle，因此就不能把這條插入到 graph 之中，反而應該把這條 edge 記錄下來，因為它就是其中一條 breaking cycles 的 edge，若是 v 不能走到 u，代表當(u,v)這條 edges 被插入到 graph 當中的話不會形成 cycle。最後當每一條 edges 照順序被檢查能否插入過後，就能得到我們所要的答案。

2. Cycle breaking 實作時的發現

在這次的 PA3 中我覺得比較有趣的發現是與同組的同學答案有些微的不同，經討論過後發現是因為 Quicksort 不是一個 unstable 的 sorter，而且 Partition 的時候是 randomize 去取要與其他資料比對的值，所以同樣 weight 的 edges 排列順序就會與本來 input 的順序不同，而且每次跑的時候順序也都不同，所以會造成每個 graph 會固定出現幾個 cost，不過這些答案都是符合 acyclic 的條件。舉例來說，在 public_cace_7.in 這個 case 的結果就會有-576、-1860、-745、-1485 這些 total cost 的結果，而同組同學用 stable sorter 去做只會得出-576 這個結果。除此之外，在 public_cace_8.in 這個 case 的結果有-27837、-27756、-27606、-27785、-27932、-27459 這些結果，而同組同學只會得出-27606 這個結果。由此而猜測應為 Quicksort randomization 的關係。