NTUEE Algorithm Programming Assignment #3

B10901151 電機三 林祐群

Data Structure & Basic Algorithm：

這次程式的基本想法是利用 Maximum Spanning Tree，也就是和課堂上提到的 Minimum Spanning Tree 雷同，但是卻是以最大邊的權重總合做為篩選條件，同時利用類似 Kruskal's Algorithm 的方式，先把邊按權重大小由大到小排，接著由大的邊開始檢查，一路檢查完所有邊，直至最小的邊也檢查完畢，如果邊的兩個端點沒有在同一個 set 裡面的話就需要做 union。

而我便利用 disjoint-set 的想法與 C++中的 vector 來實作：創建 struct Edge(存有每個邊的起始點、終點和權重，)與 struct Graph 來存取多個 edge。並實作其他功能，像是 addEdge(把未來讀進來的每個邊加到 vector 中)、sortEdges(用來使之後的 sorting 可以做大到小的比較)、findParent(找每個 disjoint-set 的代表，而這些代表在初始化時給予值-1)、unionSets(合併兩個原先 disjoint 的 set)、和 `findMaximumSpanningTree`(基本上在做的事情就是尋找該被 removed 的邊，並確保可以讓這些被去除的邊的權重總合為最小。其中的 vector<int> parent 用來記錄各個 set 的代表與之間的 parent 和 children 關係，也就是 disjoint-set 課堂上所說的架構)。

Main function 則是處理讀取資料和輸出到檔案上等等程序，並由輸入的 char u 或 d 判斷為 undirected 或是 directed graph，而因為 unweighted graph 的邊的權重都是 1，所以不用多去考慮這個特例，而可以直接運用原本的程式來實作。而輸入的 weight 的範圍式-100~100，雖然有負值，但不影響 Kruskal Algorithm 等概念的實際運行(也可以說是 MST 這類型的問題不是在比較一個路徑上面所有邊權重加總這類 SP 的問題，所以 negative weight 並不會影響)。

同時，我在處理有向圖和無向圖時使用了不同的邊處理方式(添加邊的方式)，以確保它們的處理是正確的。對於無向圖，邊會被添加到兩個頂點的 Adjacency List 中，而對於有向圖，只有單向的邊會被添加到一個頂點的 Adjacency List 中。這樣就可以確保部會有多餘的邊或是缺少邊在我們的考慮範圍中。

Findings：

這次 PA 中使用了比較多 C++11 之後的功能，讓整體程式作業可以更有效率的運行也使整體程式碼更加簡潔。像是 auto 的使用、lambda 表達式等等，除此之外在嘗試優化程式碼的時候，有想到 std::ios::sync_with_stdio(false)和 cin.tie(NULL)來加速檔案讀取的時間，也在找尋一些資料時發現 emplace_back() 可以更有效率的替代 push_back()，所以在逐步更改程式碼後得到了愈來愈好的檔案 size，從原先 6KB 到 4KB(MobaXterm 上的檔案大小)。