Home work #4 B10815057 廖聖郝

1. Shortest Path

節點class，每條邊使用pair<Node<N,W>\*,W>來儲存邊的終點指標與權重，父節點則是為了讓之後能夠印出路徑，所以要儲存路徑中上個節點的指標，以便之後能夠遞迴找到整條路徑。



找尋節點的函式



初始化新節點的函式



輸入最終節點，使用父節點往上遞迴直到起始節點，然後印出路徑



處理輸入資料，建立節點與所有邊



Dijkstra’s Algorithm，從起始節點開始走，把往外連接的節點賦予自己的權重加上邊的權重，然後加入可能列表，如果該節點曾經被賦予權重，就要比較新的權重與舊的權重，選小的更新到此節點的權重，最後還要連接最佳路徑的父節點。



印出最終節點跟起始節點之間的路徑與權重，並釋放記憶體



1. Path with maximum Probability

邊的class，節點並無實體的class，而是用string代表，每條邊儲存2個端點的名稱(string)與邊的probability。



因為是用vector儲存所有集合，所以該函式回傳節點位於哪個集合之中(以陣列index表示)



處理輸入，建立所有邊，存入vector(all\_edge)當中，all\_node用於儲存所有遇到的節點名稱，因為集合的特性，所以不用處理重複insert的狀況



先把所有節點建立一個集合，集合內只有該節點，每個集合都放到vector(all\_set)當中



使用C++內建的排序函式，因為排序的資料型別(Edge)沒有定義如何判斷大小，所以建立一個lambda函式來判斷2個Edge的大小，函式內部直接比較probability即可。



排序後，all\_edge內的資料變為由大到小，依序由大到小的處理每個邊，若邊上的2個節點屬於同一個集合，加入這條邊後會形成一個環，就不符合最小生成樹的規則，所以直接忽視這條邊，反之，若處於不同集合，連接這條邊就是合法的，把結果乘上該邊的probability，然後合併2個集合。



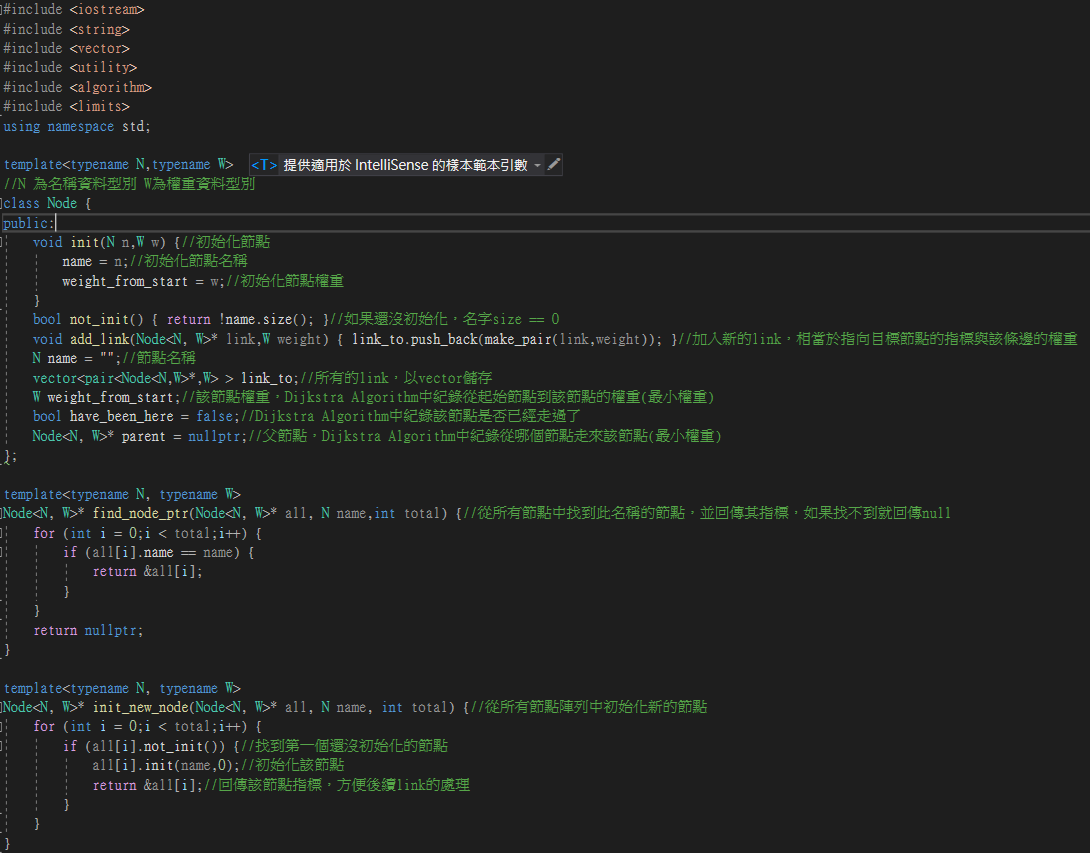
處理輸出，根據題目要求，結果小於0.05視為0，其餘的四捨五入到小數點後四位

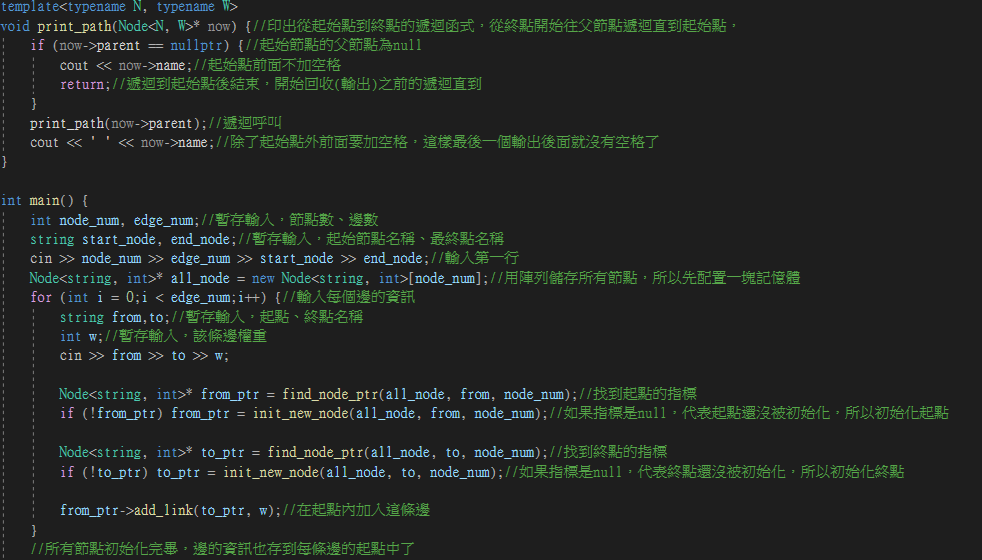


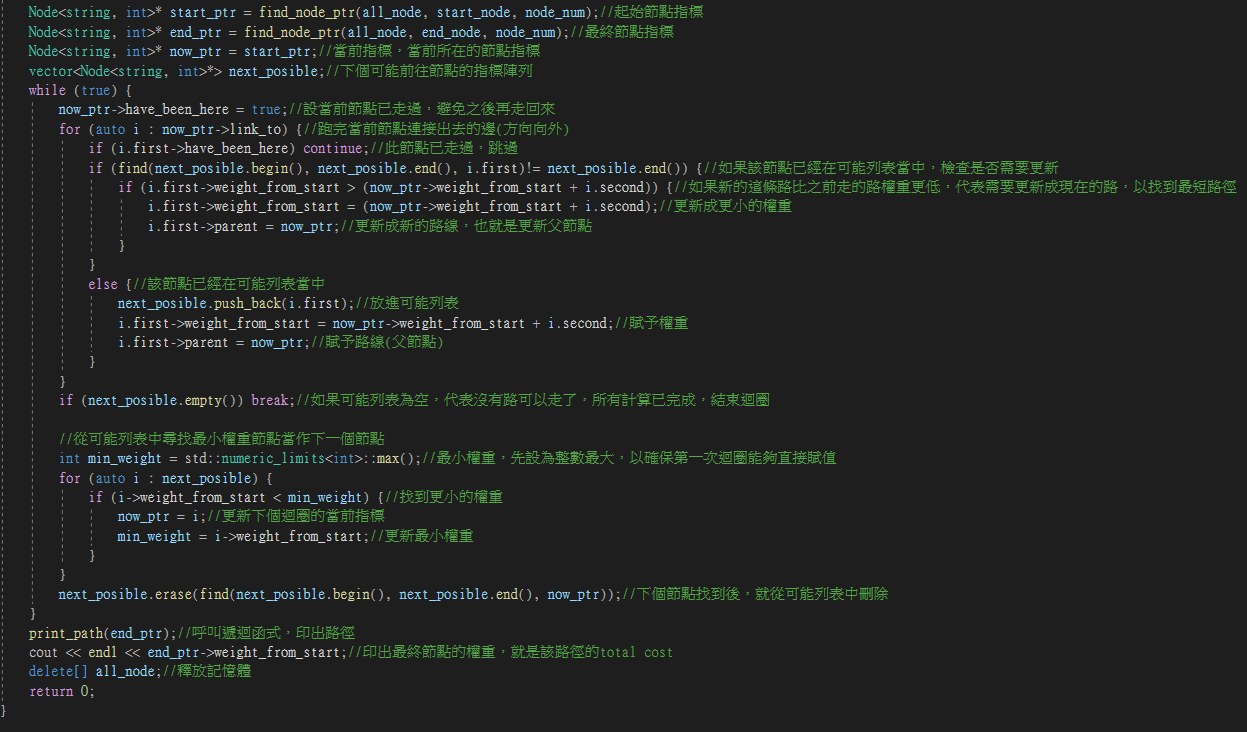
1. Source code

[GIST](https://gist.github.com/frakw/f1f01062b29a9e12816b54f774584270) [IMGUR圖片](https://imgur.com/a/g97xG0d)

4-1







4-2

