LAB 12

運用優先權佇列之最小生成樹演算法

長榮大學

資訊工程學系

班級：資工 2B

姓名：郭智榮

學號：109B30612

日期：2022/06/06

版本校定紀錄：

版本	更新紀錄	發佈日期
0.0.0.0	初版完成	2022/06/06

一. 實驗需求：

(一). 題目說明

描述：

輸入一個無向圖 G=(V, E, w)，其中點以 0~n-1 編號，而邊的權重是非負整數。計算 G 的最小生成樹的成本。兩點之間可能有多個邊。

輸入說明：

第一行是兩個正整數 n 與 m，代表節點數與邊數，節點是以 0~n-1 編號。

接著有 m 行，每一行三個整數 u, v, w 代表一條無向邊(u,v)的長度是 w。其中 n 不超過 10^4，m 不超過 10^5，w 是不超過 10^4 的非負整數。

輸出說明：

輸出最小生成樹的成本，如果不存在則輸出 -1。

(二). 演算法

i. 虛擬碼

定義 using namespace 型態的 std；//無此定義則 vector 宣告將報錯
定義 max 為 10000000；

int main(){
宣告兩個整數(int)型態變數 n, m，並用於儲存「頂點數量及無向邊數量」;

輸出提示訊息「請輸入頂點數量及無向邊數量」;
輸入兩整數代表測資頂點數量及無向邊數量，並存入 n 和 m 中；
宣告 vector 型態變數 vertex 並存放兩個整數(int)型態資料，且大小為 n，用於存放節點可前往的節點及該邊長(權重)；

建立一個整數陣列 distance 用於紀錄由 0 到某頂點的距離，預設全為 max，陣列大小為 n；
將 distance[0]設為 0，因 0 到自己的距離為 0；

建立一個布林陣列 final 用於紀錄某頂點是否已完成，預設全為 false，陣列大小為 n；

```
for i = 0 to i < m (for i++){
    宣告三個整數(int)型態變數 u, v, w，分別用於儲存「該條無向邊的兩端及邊長(權重)」;

    輸出提示訊息「請輸入該條無向邊的兩端及邊長」;
    輸入無向邊的一端存入 u，另一端存入 v 及無向邊邊長存入 w;

    宣告布林變數 check 並設為 false，用於紀錄所輸入的兩端是否已有連結的邊;

    for j = 0 to j < vertex[u].size() (for j++){
        如果 vertex[u][j].first 等於 v，則 u 已與 v 有無向邊{
            將 check 改為 true;
            如果(vertext[u][j].second 大於 w{
            // 因原本的邊長大於目前輸入的 w，故改為 w
                vertext[u][j].second 設為 w;
                break;
            }
        }
    }
    for j = 0 to j < vertex[v].size() (for j++){
        如果 vertex[v][j].first 等於 u，則 v 已與 u 有無向邊{
            如果(vertext[v][j].second 大於 w{
            // 因原本的邊長大於目前輸入的 w，故改為 w
                vertext[v][j].second 設為 w;
                break;
            }
        }
    }

    如果 check 不為 true，代表 u 與 v 間目前沒有無向邊連結{
        將{v,w}放到 vertex[u]內的最後;
        將{u,w}放到 vertex[v]內的最後;
```

```
    }
}

宣告優先佇列變數 queue 儲存兩個整數，用於紀錄要造訪的頂點；
將 distance[0] 及 0 放入 queue 中；

迴圈 (queue 不為空){
    宣告整數變數 now_vertex 並放入 queue 中最先的頂點；
    因已取出最先的資料，故將 queue 用 pop 丟出最先資料；

    如果 final[now_vertex]為 true，代表已被訪問完成{
        continue；
    }
    將 final[now_vertex]設為 true；

    for auto next 依序存入 vertex[now_vertex] {
        如果 next.second 小於 distance[next.first]{
            如果 final[next.first] 不為 true{
                distance[next.first] 為 next.second；
                queue 存入{ (distance[next.first] * -1),
next.first }；
            }
        }
    }
}

宣告整數變數 cost 儲存「最小生成樹的成本」；
for i = 0 to i < n (for i ++){
    如果 distance[i]等於 max{
        將 cost 設為 -1；
        break;
    }
    否則{
        cost 加上 distance[i]；
    }
}
```

```
        輸出「最小生成樹的成本為：cost」;

        return 0;
    }
```

二. 完整程式碼：

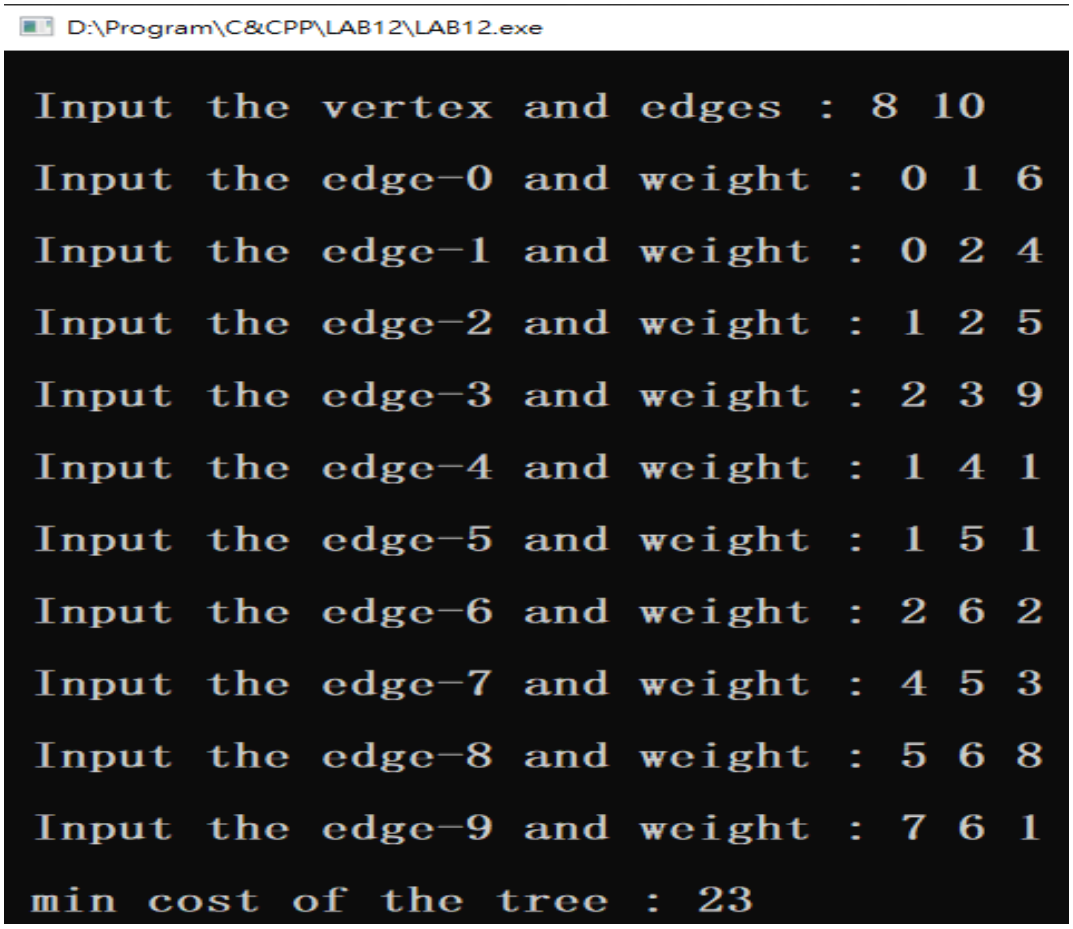
```
#include<bits/stdc++.h>
using namespace std;
#define max 10000000
// 定義通用的最大距離 max 為 10000000

int main(){
    int n, m;
    // 宣告整數 n, m，分別用於儲存「頂點數量及無向邊數量」
    printf("\n Input the vertex and edges : ");
    // 輸出提示訊息
    scanf("%d%d", &n, &m);
    // 將輸入的頂點數量存入 n, 無向邊數量存入 m
    vector< pair<int, int> > vertex[n];
    // 宣告 vector 型態變數 vertex 儲存資料為兩個整數

    int distance[n];
    // 宣告一個大小為 n 的整數陣列，用於儲存各點與頂點 0 的距離
    for(int i = 0 ; i < n ; i++){
        distance[i] = max;
        // 將 distance 內所有節點的值設為 max
    }
    distance[0] = 0;
    // 將頂點 0 與自己的距離設為 0

    bool final[n];
    // 建立大小為 n 的布林矩陣，紀錄該頂點是否已完成
    for(int i = 0 ; i < n ; i++){
        final[i] = false;
        // 將 final 內所有節點的值設為 false
    }

    for(int i  = 0 ; i < m ; i++){
        int u, v, w;
        // 宣告整數變數，分別用於儲存「該條無向邊的兩端及邊長(權重)」
```

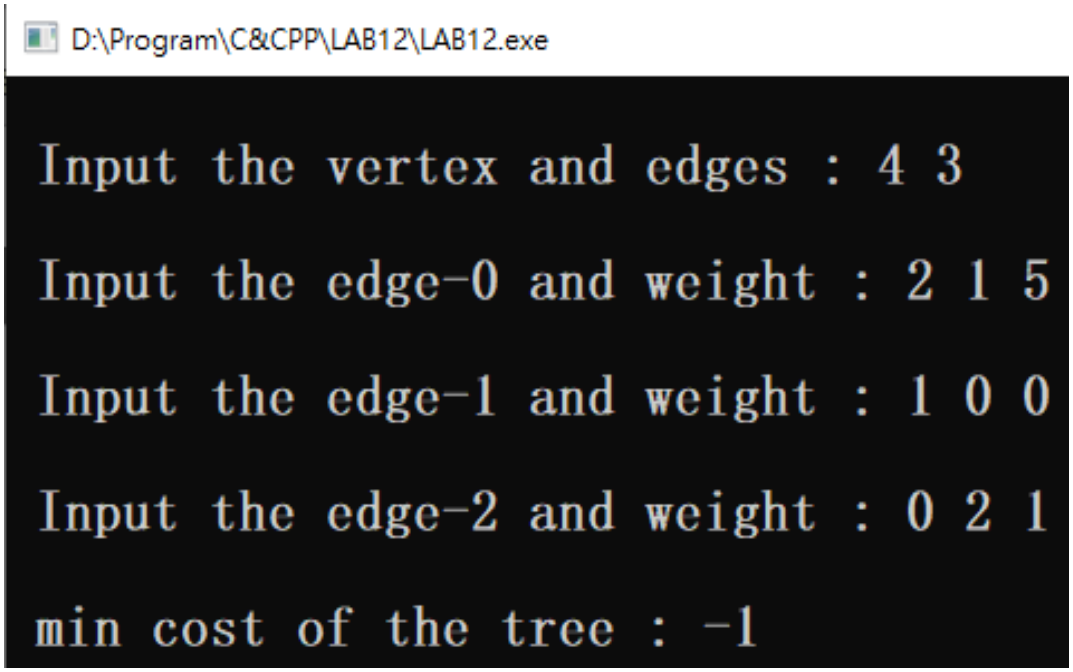
```
printf("\n Input the edge-%d and weight : ", i);
// 輸出提示訊息
scanf("%d%d%d", &u, &v, &w);
// 輸入有向邊起點存入 s 、有向邊終點並存入 t 及有向邊長存入 w
bool check = false;
// 宣告布林變數 check 紀錄所輸入的兩端是否已有連結的邊

for(int j = 0 ; j < vertex[u].size() ; j++){
// 搜尋 vertex[u] 內的所有資料
    if (vertex[u][j].first == v){
    // 若 vertex[u][j] 為 v，代表該條邊連結 u 與 v
        check = true;
        // 將兩端已有連結線變更為 true
        if(vertex[u][j].second > w){
        // 若 u 與 v 的無向邊長大於當前輸入的邊長則進入 if 內
            vertex[u][j].second = w;
            // 將 u 與 v 的邊長改為 w
            break;
            // 離開迴圈
        }
    }
}

for(int j = 0 ; j < vertex[v].size() ; j++){
// 搜尋 vertex[v] 內的所有資料
    if (vertex[v][j].first == u){
    // 若 vertex[v][j] 為 u，代表該條邊連結 v 與 u
        if(vertex[v][j].second > w){
        // 若 u 與 v 的無向邊長大於當前輸入的邊長則進入 if 內
            vertex[v][j].second = w;
            // 將 u 與 v 的邊長改為 w
            break;
            // 離開迴圈
        }
    }
}
```

```
    if(!check){
    // 若 u 與 v 並不存在無向邊則進入 if 內
        vertex[u].push_back({v, w});
        // 紀錄 u 節點可以通往 v 節點且邊長為 w
        vertex[v].push_back({u, w});
        // 紀錄 v 節點可以通往 u 節點且邊長為 w
    }
}

priority_queue< pair<int, int> > queue;
// 宣告優先佇列變數 queue 儲存兩個整數，用於紀錄要造訪的頂點
queue.push({distance[0], 0});
// 將 distance[0] 及 0 放入 queue 中

while(!queue.empty()){
// 若 queue 不為空則進入 while 內
    int now_vertex = queue.top().second;
    // 宣告整數變數 now_vertex 並放入 queue 中儲存最前的頂點
    queue.pop();
    // 因取出最前的資料，故將其用 pop 丟出
    if(final[now_vertex]){
    // 如果 now_vertex 已被訪問完成，則進入 if 內
        continue;
    }
    final[now_vertex] = true;
    // 將目前造訪的頂點設為 true 以記錄該頂點已完成

    for(auto next : vertex[now_vertex]){
    // 將當前造訪頂點所連結的資料依序存入 next
        if(next.second < distance[next.first]){
        // 到下個節點的邊長小於下個節點目前的抵達邊長，則進入 if 內
            if(!final[next.first]){
            // 如果下個節點尚未完成訪問，則進入 if 內
                distance[next.first] = next.second;
                // 下個節點的最小抵達邊長設為目前節點的抵達邊長
```

```
                    queue.push({(distance[next.first] * -1),
next.first });
                    // 將由 0 到下個頂點的距離及下個頂點傳入 queue 中
                    // 在 queue 中同頂點的資料，若距離較大會排在前
                    // 故需 * -1 讓較短距離排在更前面
                }
            }
        }
    }

    int cost = 0;
    //  宣告整數 cost 儲存最小生成樹的成本

    for(int i = 0 ; i < n ; i++){
        if(distance[i] == max){
        // 若節點的最小抵達邊長為 max，則進入 if 內
            cost = -1;
            // 將成本設為 -1，代表沒有形成最小生成樹
            break;
            // 因無法生成，故結果以定，因此跳出迴圈
        }
        else{
            cost += distance[i];
            // 將最小生成樹的成本加上抵達當前節點的成本
        }
    }

    printf("\n min cost of the tree : %d", cost);
    // 輸出樹的最小成本

    return 0;
}
```

三. 輸入及輸出結果：


D:\Program\C&CPP\LAB12\LAB12.exe
Input the vertex and edges : 8 10
Input the edge-0 and weight : 0 1 6
Input the edge-1 and weight : 0 2 4
Input the edge-2 and weight : 1 2 5
Input the edge-3 and weight : 2 3 9
Input the edge-4 and weight : 1 4 1
Input the edge-5 and weight : 1 5 1
Input the edge-6 and weight : 2 6 2
Input the edge-7 and weight : 4 5 3
Input the edge-8 and weight : 5 6 8
Input the edge-9 and weight : 7 6 1
min cost of the tree : 23


D:\Program\C&CPP\LAB12\LAB12.exe
Input the vertex and edges : 4 3
Input the edge-0 and weight : 2 1 5
Input the edge-1 and weight : 1 0 0
Input the edge-2 and weight : 0 2 1
min cost of the tree : -1

四. 心得與討論：

本次實作的最小生成樹成本，採用的演算法跟 LAB11 一樣，因此在程式碼上只需要將 LAB11 的儲存部分做更動，便能將這次的實作完成，因此在實作上並沒有太大的問題，稍微轉一下腦袋便能迎刃而解。

透過這次實作能夠更加了解最小生成樹的生成方式跟程式碼，相信經過這幾次的實作，未來若碰上圖或樹的資料結構，都能很輕鬆的理解並實作出來。