編譯環境：CODEBLOCKS 16.01

Release 16.01　rev 10702 (2016-01-25 19:50:14)　gcc 4.9.2 Windows/unicode - 32 bit

程式輸出：

利用 Kruskal's Minimum Spanning Tree 輸出最短距離

參考資料：

https://www.geeksforgeeks.org/union-find/

https://www.geeksforgeeks.org/greedy-algorithms-set-2-kruskals-minimum-spanning-tree-mst/

https://www.cdn.geeksforgeeks.org/kruskals-minimum-spanning-tree-using-stl-in-c/

心得：

此次作業算是超出我能力範圍了，**Kruskal 理論本身超級簡單，但是要把它寫成程式就十分困難，光 struct 就要創出 2 個，還有構成 Kruskal 理論的元素** `find`、`Edge`、`Graph`、`Union`，最難的部分就是 `KruskalMST` 主程式的地方，先找出距離由長短做排序，以最短的相連接，並判斷他是否有連出圓圈，如果有就捨棄，沒有就找下一個，一直到連線的線段少於節點`-1`，這短短的敘述打成程式碼真的超級麻煩，上網找了很多很多的參考程式碼，但有一部份都只能處理 `0~9`，不具有處理 `10+`以上的數值能力

至於讀檔的部分依舊是老樣子，`char ch; /***read a character**/`，將頭尾以及當前位置分 3 個 `array` 去存

```
else if(ch==' '&& n==0){   /****start point***/
                        strcpy(ascii, str.c_str());        /***copy string to char array***/
                        start1 = int(a [0])-65;          /***ascii A=0,B=1,C=2...***/
                        src[k]=start1;
                        n++;
                        str.clear();
                        }
else if(ch==' '&& n==1){   /***end point***/
                         strcpy(ascii, str.c_str());        /**copy string to char array***/
                         end1 = int(a[0])-65;             /*** ascii    A=0,B=1,C=2...***/
                         dest[k]=end1;
                         str.clear();
                        }
```

輸出的部分

```
Graph r(V,E);
for(int i=0;i<E;i++){
        r.addEdge(src[i],dest[i],weight[i]);;
        }
        int mst_wt = r.kruskalMST();
        fp<<mst_wt<<endl;
```

此次作業會讀取值與距離並個別分開儲存，避免中間的 spanning tree 跑開

```
for (it=edges.begin();it!=edges.end();it++){
        int u=it->second.first;
        int v=it->second.second;
        int set_u=ds.find(u);
        int set_ =ds.find(v);
        // Check if the selected edge is creating
        // a cycle or not (Cycle is created if u
        // and v belong to same set)
        if (set_u!= set_v){
            // Current edge will be in the MST
            // so print it
            cout<<u<<"---"<<v<<endl;
            // Update MST weight
            mst_wt+=it->first;
            // Merge two sets
            ds.merge(set_u,set_v);
        }
    }
```

為額外加上方便檢測數據

求出無向圖的其中一 tree。若是圖不連通，則是求出其中最小 tree 是 kruskal 理論的判斷方法。圖上每一個點，各自是一棵最小生成子樹 MSS。
圖上所有邊，依照權重大小，由小到大排序。嘗試圖上所有邊，作為最小 tree 的邊：兩端點分別位於兩棵 MSS，也就是產生了橋：用這條邊連結兩棵 MSS，合併成一棵 MSS。這條邊是最小 tree 上的邊。兩端點皆位於同 tree MSS，也就是產生了 circle：捨棄這條邊。

```
vector< pair<int, iPair> > edges;這段也超級特別由於格式一定要
< *<*, *> >不可以少空格
'>>' should be '> >' within a nested template argument list
```