Algorithms Programming Assignment #3

B10901006 電機三 鄒昀樺

1. Algorithm
   - ✧ Undirected

     The problem can be seen as a **maximum** spanning tree problem for the undirected case, which must contain n-1 edges at the end. Instead of choosing edges by increasing order, sort and choose them by decreasing order to have the minimum cost (total weight of cut edges). Counting sort is used in this program by adding 100 to the weight so that all weights are positive or 0.
   - ✧ Directed

     As for the directed graph, before adding an edge to the tree we need to check if any cycle would be formed. This can be done by creating a transitive closure matrix "t" to record whether paths exist between two vertices. Furthermore, in order to obtain minimum cost, more than n-1 edges could be chosen as long as they will not form any cycle and their weights are positive. In other words, if an edge (u,v) with u,v both belonging to the tree, choose the edge if its weight is positive and don't if it's negative, for it can reduce the cost.
2. Data Structure
   - ✧ Class: disjointSet- vertices
     - ◆ parent[]
     - ◆ rank[]
     - ◆ findSet()
     - ◆ Union()
   - ✧ Class: edge- edges_unsorted, edges
     - ◆ w – weight of the edge
     - ◆ u, v – start, end point
   - ✧ t[][]
     - ◆ t[i][j] = true if there is a path from i to j **in the tree**, false otherwise
     - ◆ Updated each time an edge is added to the tree
3. Result (on edaU12)

Input case	Total weight of cut edges
public_case_0.in	5
public_case_1.in	21
public_case_2.in	-3330
public_case_3.in	-21468
public_case_4.in	0
public_case_7.in	-10515
public_case_8.in	-70938