Algorithm PA3

姓名：謝皓宇　學號：B12901041 系級：電機二

1. Data structure

   First, I declare a new data type **Edge** using Struct.

```
struct Edge{
    int u;
    int v;
    int weight;
    bool chosen;
};
```

   For undirected graph, I used an one-dimension vector (data type is Edge) to store all edge. Since each Edge has two endpoints and weight, I can get the information of graph I need in this vector.

   For directed graph, I store it into an one-dimension vector (as mentioned above), and an adjacent list. For example, for **Adj_list[u],** I store the vertex **u** points to, and the edge weight.

2. Algorithm

   For undirected graph, I used Kruskal Algorithm. I sorted all the edge by their weight, and I choose the edge by weight **in decreasing order** to get maximum spanning tree. If chosen, I set the edge.chosen to be **true**.

   At the end, I added up the weight of edges that are not chosen (edge.chosen == false), and output the information of these edges.

   For directed graph, I solved it in two steps.

   First step, I solved it as undirected graph at the beginning.

   Second step, since Kruskal has sorted the edge by weight, we can add the edge that are not chosen in first step by weight in decreasing order(Choose the biggest edge to smallest edge), until the edge

weight is negative. After adding the edge in the graph, I used DFS to detect whether there exists a cycle in this graph. If the cycle exists, I remove this edge, and if not, I add this edge into the graph.

3. README
   compile: make
   run: ./bin/cb inputs/<input_file_name> outputs/<output_file_name>