EE4033 Algorithms

Programming Assignment #3

B11901110 電機三 陳璿吉

1. Data Structure:

I designed a class for the edges of a graph, which consists of 3 properties: `starting point`, `ending point` and `weight`. This design aligns with the input format.

```
class edge
{
    public:
        void set_start(int s){start = s;}
        int get_start(){return start;}
        void set_end(int des){end = des;}
        int get_end(){return end;}
        void set_weight(int w){weight = w;}
        int get_weight(){return weight;}

    private:
        int start;
        int end;
        int weight = 1;
};

bool operator>(edge e1, edge e2)
{
    return e1.get_weight() > e2.get_weight();
}

bool operator<(edge e1, edge e2)
{
    return e1.get_weight() < e2.get_weight();
}
```

The class of graph has two members: number of vertices and an edge list that store the edge specified in the input. To facilitate problem solving, I defined the following member functions or classes:

(1) Class `DSU` (Disjoint Set Union): For applying Kruskal's algorithm.

(2) Function `get_adj_list()`: To get the adjacency list of the graph from edges.

(3) Function `pre_cyc_tool` and `detect_cycle`: To determine whether the graph has a cycle.

2. Algorithms: My solution highly depends on Kruskal's algorithm

(1) Undirected Graph: Since we want the total weight of removal to be minimized, we must keep the edges of maximal weights in the graph. Note that a Maximum Spanning Tree is connected and doesn't contain any cycle. Therefore, after inputting the graph $G = (V, E)$, we sort the edges in E by decreasing weights (this is realized by operator overloading in C++) and perform Kruskal's algorithm to find a Maximum Spanning Tree $G_M = (V, E_M)$ of G. The set $E \backslash E_M$ contains the edges of minimum weights that must be removed. We simply calculate the sum of edge weights and output each edge of $E \backslash E_M$.

(2) Directed Graph: The procedure is similar to the case in Directed Graph. Nevertheless, since the graph is directed, we need to determine whether the edges $E \backslash E_M$ are contained in a cycle. We first sort the edges in $E \backslash E_M$ in decreasing order. Next, for each edge $e \in E \backslash E_M$, we check whether the augmented graph $G_M' = (V, (E \backslash E_M) \cup \{e\})$ contains any cycles using DFS. If there is a cycle, then e is removed, otherwise we keep e and continue the loop. When the process stops, we have found a maximal set $E_k =$

$\{e_1, e_2, \ldots, e_k\} \subseteq E_M$ such that $G''_M = (V, (E \backslash E_M) \cup E_k)$ contains no cycles for some $k \geq 0$. Hence, the set $E_M \backslash E_k$ contains the edges that must be removed.

3. README: Please refer to the README file in the uploaded file.
4. References:

(1) Disjoint Set Union: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

(2) Cycle Detection of a graph: https://www.geeksforgeeks.org/detect-cycle-in-a-graph/