Algorithm PA3 Report

B07303024 經濟三 李品樺

- Data structures used in the program: Vector, list
- Findings:

● Store Input Data

First of all, we know that if the given graph is an undirected graph, we need to reverse the start point and the end point. However, we can only remove edges that are in the original input in the directed graph. Therefore, we need to save two edges: one is for graph implementation, another is for checking whether it is original input. So, we use adjacency list to store the original input data, and use vector of vector to save original input. Notice that in the adjacency list, a pointer is pointed to each node, and the node is pointed to the next node.

For example, if the input data is:

Start point	0	0	1	1	2	3
End point	1	2	3	4	5	4
Weight	3	5	10	8	9	5

Then, we first create a pointer pointed to these 5 nodes. Then, we use adjacency list to save the edges. In order to print out the graph, we can use codes like this:

```
void Graph::printGraph()
{
    std::cout << "========== Graph ==========" << endl;
    // print adjacency list representation of graph
    for (int i = 0; i < nodeNum; ++i){
        std::cout << "head" << i << ":" << endl;
        Node* a = head[i];
        while(a != nullptr){
            std::cout << "("<< i << "," <<a->nodeKey << "," << a->cost  << ")"<< endl;
            a = a ->next;
        }
        std::cout << endl;
    }
}
```

● Remove Edges

- Undirected Graph

In the case of undirected graph, we create a maximum spanning tree(MST) and remove all the edges that are not in the MST. Initially, we append two way of the edge in the adjacency list. Therefore, we use a two-dimensional matrix map to store the original input. When we want to remove the edge, we need to check whether this is the original input.For example, if the original is: 0 2 5, which means

the edge is from node 0 to node 2 with weight 5, we will append 2 to the adjacency list of 0, append 0 to the adjacency list of 2, and set the value of map[0][2] = true.

- Directed Graph

  First, we treat the directed graph as the undirected one. Then we have a set of candidate edges to remove from. However, we do not want to remove all the edges that are not in the MST. Therefore, we add back the edge one by one and check whether the new graph will create a cycle. We implement DFS in order to check if the new graph contains cycle. If not, we add it back to the MST, and the edge is removed from the candidate set; otherwise, the edge remains in the candidate set.
  Finally, we sum up all the edges and return back the main program.
  Notice that in the directed edges, we can only remove those edges that are in the original input.

- Saving Memory Space

  Since the largest case needs to create more than 50000000 nodes, so we modify the loop from”for(int i = 0; i < n; i++)” to “for(size_t i = 0; i < n; i++)”.