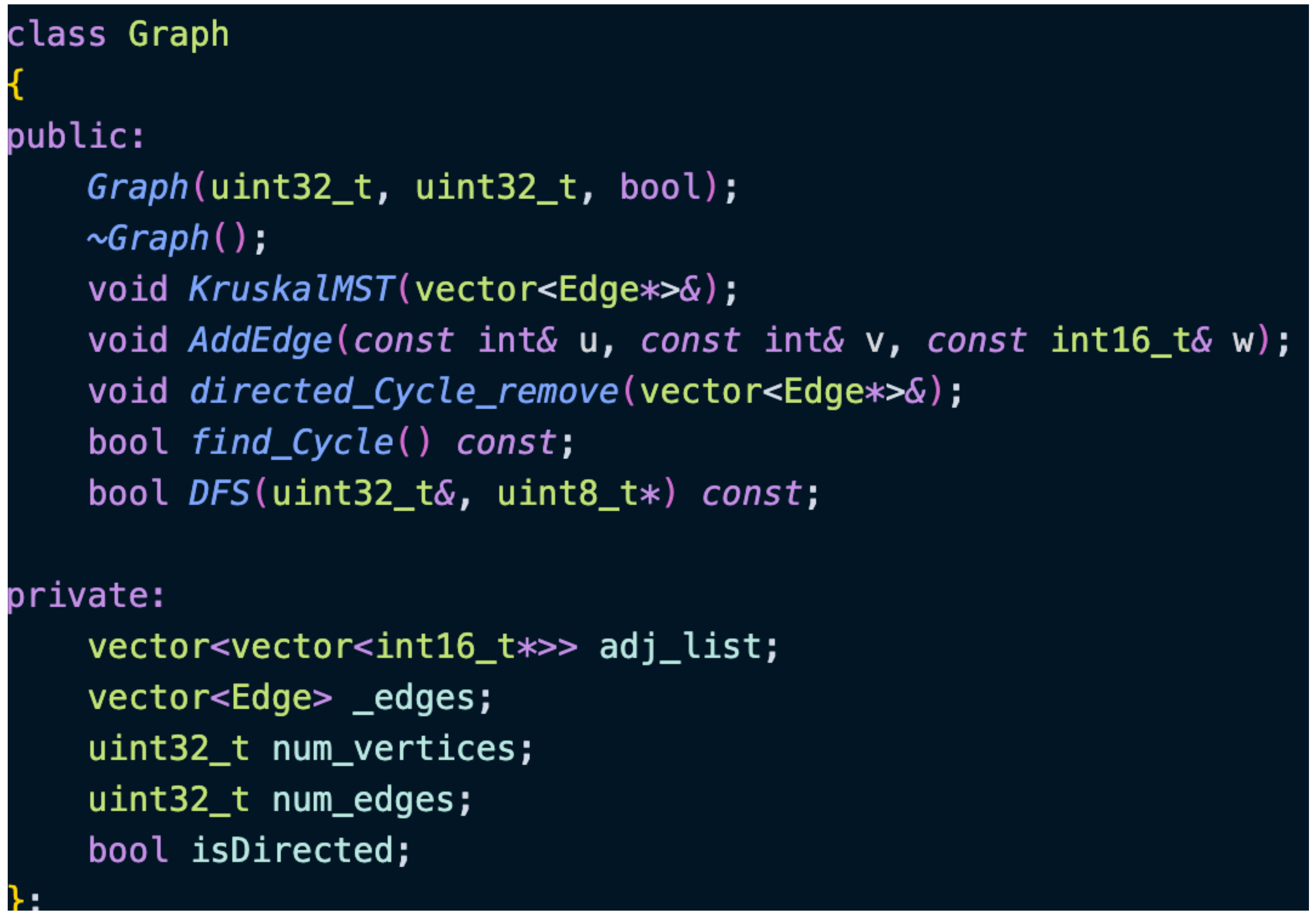
PA3 report

B09901104翁瑋杉

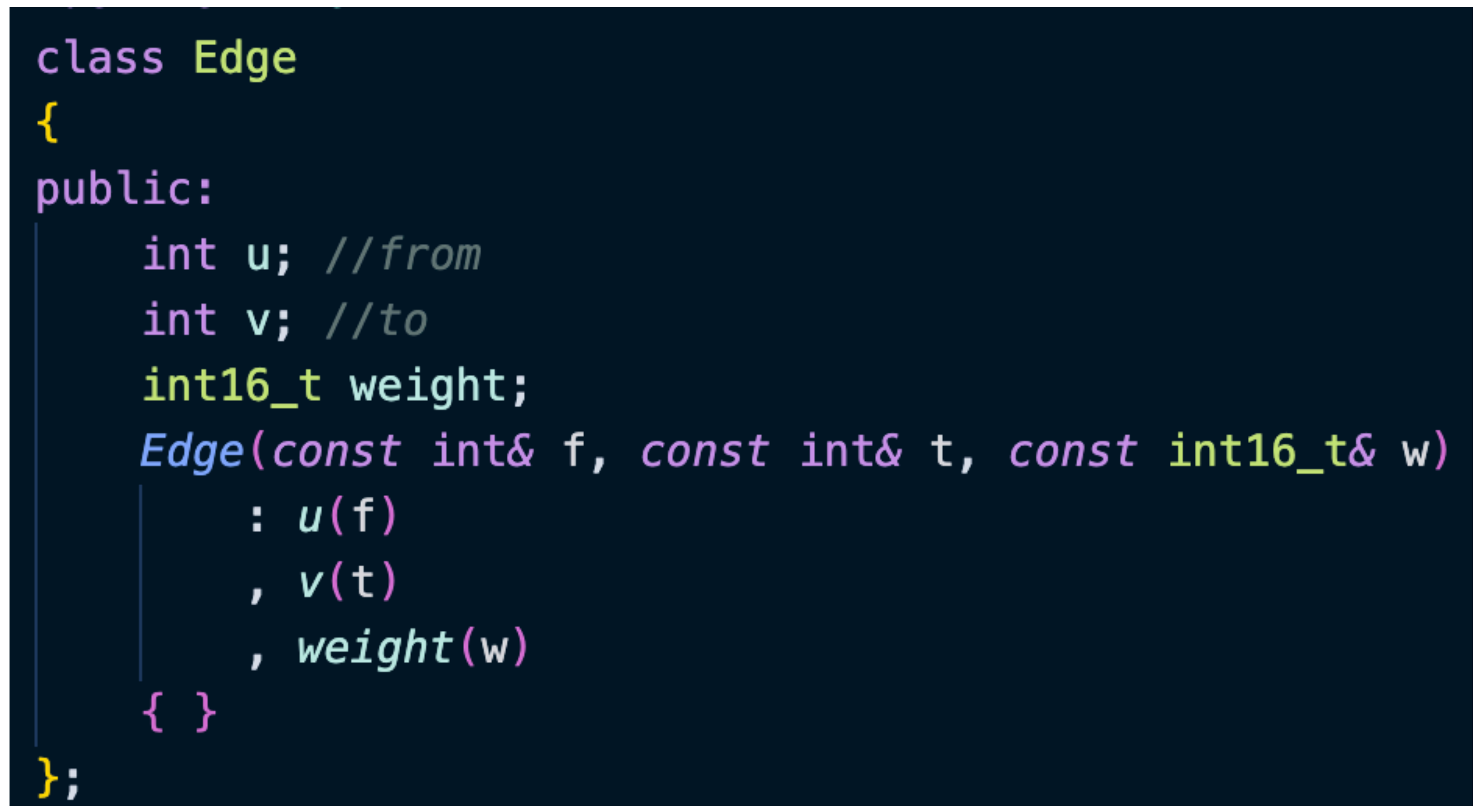
Data Structure used

- The graph class is shown at left
- KruskalMST is in reverse order to get the maximum spanning tree, because we want the weights sum of removed edges to be as small as possible.
- I have _edges to record undirected edges, while adj_list is an adjacency matrix to record directed edges. I make adj_list a matrix instead of list because I want to access the weight as fast as possible.
- For case 1 and 2 , I simply apply modified KruskalMST to get the removal edges with min cost.
- For case3, I treat directed edges as directed first and apply modified KruskalMST, so the directed graph is at least weakly connected. However, removing all these edges will be too costly, so I add edge by edge back to the graph and see if there exist a cycle. If yes, that means I have to remove the edge, while no means I can place the edge back in graph and save removal cost. In addition, if edge.weight <= 0 , I choose not to add the edge back to the graph in order to get min removal cost.
- Intuitively, the problem is NPC for directed graph because adding an edge back to the graph could block other edges. Therefore, I decide to add edge back to the graph in descending order to get suboptimal solution. In practice, this approach does yield better solution then adding edge back in random order.



```
class Graph
{
public:
    Graph(uint32_t, uint32_t, bool);
    ~Graph();
    void KruskalMST(vector<Edge*>&);
    void AddEdge(const int& u, const int& v, const int16_t& w);
    void directed_Cycle_remove(vector<Edge*>&);
    bool find_Cycle() const;
    bool DFS(uint32_t&, uint8_t*) const;

private:
    vector<vector<int16_t*>> adj_list;
    vector<Edge> _edges;
    uint32_t num_vertices;
    uint32_t num_edges;
    bool isDirected;
};
```



```
class Edge
{
public:
    int u; //from
    int v; //to
    int16_t weight;
    Edge(const int& f, const int& t, const int16_t& w)
        : u(f)
        , v(t)
        , weight(w)
    { }
};
```

Finding in this assignment

- Since I need to include different source header files, I have encountered several errors in makefile. However, after asking some seniors for best files structure and makefile, I successfully format my .o .h .cpp files in a cleaner manner. Now, all .o files are stored in /build , .h in /inc , .cpp in /src and binary files in /bin.
- Cycle breaking problem is originally NPC, however, I got to learn how we can approach suboptimal solutions in polynomial time through this PA.
- Since this PA requires data structures such as disjoint set, adjacent list, I really pay attention in deciding whether it is important to store the variables, what is the most efficient way to store, access and modify the variables. With the help of chatGPT, I got to dive deeper in C++ and understood some details like memory allocation, memory leak.
- In this PA, I think the algorithm itself is rather easy, and the coding part is what really time-consuming, but I do feel more confident and comfortable to build a C++ project.