演算法 PA3

B10901176 蔡弘祥

140.112.48.76

1. Data Structure

   The input file is composed of several lines. The first line is a number which is the number of points, let's call it V. The second line is a number representing the number of edges, let's call it E. In both undirected and directed case, we store edges using dynamic array with size E.

2. Method

   i. Undirected case

      This method is similar to Kruskal MST Algorithm to some extent. But what we have modified is that we asked it to sort decreasingly and add back the edges in the decreasing order. In this manner, we can construct a new graph with the lowest cost of removing the edges.

   ii. Directed case

      The directed case is however a NP-Hard problem, which means that we cannot always get the optimal solution in a limited time. I have come up with two method that can give us an acceptable solution. The first one is just the Undirected case algorithm but modified the way to detect cycle. The second one is a also a modified version of the Undirected case. We first assume the graph is undirected and execute the Undirected case algorithm. After that, we will get two set of edges, one contains those we removed, while the other contains those we did not removed. Moreover, we attempt to add back the "positive" removed edges in the decreasing order. If adding back would create cycle, then we cannot add it back. If adding back would not create cycle, then we should add it back to lower the cost.

3. Other Findings

   One important finding is that path compression can decrease running time tremendously. Originally, I just use recursive function to find out what root is. Although the running time is acceptable, but I expect it can do better, thus I perform the path compression that we taught in class. It actually runs three times better than the original method.