EE450
Summer 2019
Lab 1

Lab 1 Report

Code Summary

In this lab, an implementation of Bellman-Ford algorithm is required to achieve by us.

Given to the characteristic of this algorithm, I divide this project in to three parts:

1.Graph: Graph.h and Graph.cpp
In this section, I implement the preprocessing of input data. For example: in lack of rows or column, the initialization of the matrix based on the input data, the format change to meet the requirements of the rest parts of the code.

2.PathFinder: PathFinder.h and PathFinder.cpp
In this section, is the main part of the implementation of bellman-ford algorithm. I come with a method for boundary checking, the takestep() method is to search for all of the next node based on the current node, update the minCost and the path when a cheaper one emerged. Use the corresponding format to return the result by tuple.

3.BellmanFord: BellmanFord.h and BellmanFord.cpp
In this section, I mainly handle the print format in the requirements and the file open and write parts are also included in this section.

Key Aspects of Implementation

Beside the part of dueling with incoming data, to pre-process and the part of output the result, the core implementation is in the PathFinder section.
The basic idea is to find the shortest path from 0 to all other node, therefore, we started from 0, traverse all the nodes, find out if it is reachable and keep watching the min cost of the path, update it when it became cheaper. The time complexity is O(n^2) due to the fact that we need to traverse all the nodes, which is O(n) time, and for each node we need to check all other node which is a O(n) operation as well.

Experimental Result

Result shows as required.

Output File Format

Result shows as required.