National Yang Ming Chiao Tung University
Department of Electrical Engineering
Data Structures, Fall 2021

December 1, 2021
Instructor: Mark P.-H. Lin
TA: I-Ping Yeh

## Programming Assignment #5

# Graphs

## 1 Problem Description

The graph data structure is the ultimate abstraction for many real world problems, such as the social graphs in Facebook's social network, knowledge graphs in Google's search engine, circuit connectivity graphs in chip design optimization and verification, and other types of graphs in maps, communication networks, and many other applications.

In this programming assignment, you are asked to implement the shortest path algorithm, or the Bellman–Ford algorithm, to search for the shortest paths in a weighted digraph from a source vertex to all the other vertices. **Your algorithm must be capable of handling negative edge weights and detecting negative cycles in the digraph.**

## 2 Input Format

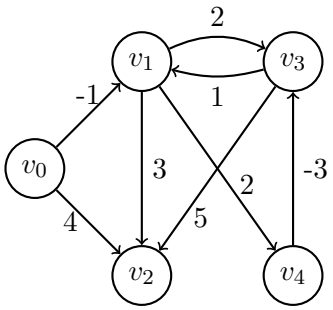
The input file gives the adjacency matrix, $A$, of a graph, $G = (V, E)$, with $n$ vertices, where $1 < n \leq 100$, and the index of each vertex ranges from 0 to $n - 1$. Each element, $a_{ij}$, in the matrix corresponds to the edge weight, $w_{ij}$, between two vertices, $v_i$ and $v_j$, which is defined as follows:

$$\mathbf{A} = \begin{pmatrix} a_{00} & a_{01} & \cdots \\ a_{10} & a_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}, \text{ where } a_{ij} = \begin{cases} 0 & \text{if } i = j \text{ or } (i,j) \notin E \\ w_{ij} & \text{otherwise} \end{cases}$$

Two example input files, "sample1.in" and "sample2.in", and the corresponding digraphs are given below. We assume that the source vertex is always $v_0$, while all the other vertices, $v_1$, $v_2$, …, $v_{n-1}$, are destinations.

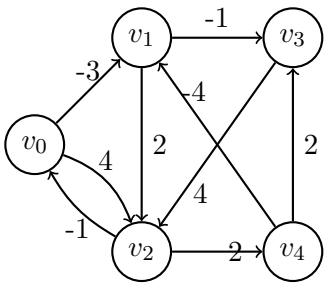
sample1.in

```
0  −1  4  0  0
0  0  3  2  2
0  0  0  0  0
0  1  5  0  0
0  0  0  −3  0
```




sample2.in

```
0  −3  4  0  0
0  0  2  -1  0
-1  0  0  0  2
0  0  4  0  0
0  −4  0  2  0
```



# 3 Output Format

After executing the shortest path algorithm on the input file, "sample.in", you will need to generate the corresponding output file, "sample.out". If the graph in the sample input has no negative cycle, the output file must contains the shortest paths and their path length between $v_0$ and each of the other vertices. Different paths should be output in the ascending order of the indices of the destination vertices. If the graph in the sample input contains one or multiple negative cycles, your program only needs to output either one of the negative cycles, instead of the shortest paths.

The sample output files, "sample1.out" and "sample2.out", resulting from "sample1.in" and "sample2.in" are given below, respectively.

| sample1.out |
| --- |
| 0-1 −1 |
| 0-1-2 2 |
| 0-1-4-3 −2 |
| 0-1-4 1 |

| sample2.out |
| --- |
| 2-0-1-2 |

As the graph in "sample1.in" has no negative cycle, "sample1.out" lists all the shortest paths from $v_0$ to $v_1$, $v_2$, $v_3$, and $v_4$, respectively, together with their path lengths at the end of each path. However, since the graph in "sample2.in" contains some negative cycles, "sample2.out" simply gives one detected negative cycle in the graph, which starts and ends at the same vertex. Any vertex in the negative cycle can be the starting and ending vertex. The source vertex may or may not be in the negative cycle.

# 4 Command-line Parameter

In order to correctly test your program, you are asked to add the following command-line parameters to your program.

**[executable file name] [input file name] [output file name]**

(e.g., StudentID.exe sample.in sample.out)

# 5 Submission Information

Your program must be written in the C/C++ language, and can be compiled on the Linux platform. The source files of your program must be named with "[your student ID].h" and "[your student ID].cpp". The executable file name of your program must be "[your student ID].exe". To submit your program, please archive both executable and source files of your program into a single zip file, named "[your student ID].zip", and upload to E3.

# 6 Due Date

The zip file must be submitted through E3 before 23:59, December 12, 2021.

# 7 Grading Policy

The programming assignment will be graded based on the following rules:

- Pass sample inputs with compilable source code (50%)
- Pass five hidden test cases (50%)

**The submitted source codes, which are copied from or copied by others, will NOT be graded. There will be 25% penalty per day for late submission.**