**EECS2040 Data Structure Hw #5 (Chapter 6 Graph)**

**due date 6/13/2021**

**By 107061123, 孫元駿**

**Part 2 Coding**

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program.

1. (50%) Graph(linked adjacency list), BFS, DFS, connected components, Computing dfn and low:

Write a C++ program to perform the following basic graph functions:

1. BFS(v) (Prog. 6.2) (v: starting vertex. You need to output the vertices visited in BFS order)
2. DFS(v) (Prog. 6.1) (v: starting vertex. You need to output the vertices visited in DFS order)
3. Component() (Prog. 6.3 where OutputNewComponent() can be simplified to just output the vertices of the component)
4. DfnLow() (Prog. 6.4) (Display the computed dfn[i] and low[i] of the graph)

on a linked adjacency list based graph. Add whatever you think necessary to your class Graph to implement the required functions, e.g., setup functions for setting up various graphs required.

Show your results using the following three graphs in your program. The main() would contain similar codes segment shown below. BFS and DFS should start from 3 vertices: 0, 3, 7, respectively as shown in the code segment.

Graph g1(8),g2(8),g3(10);

g1.Setup1();

//BFS

g1.BFS(0);

g1.BFS(3);

g1.BFS(7);

//DFS

g1.DFS(0);

g1.DFS(3);

g1.DFS(7);

//Components & DfnLow

g1.Components();

g1.DfnLow(3);



How to use my code:

All the input file are in input.txt.

First, input file will input the number of edge and set the graph up for G1, then it will find the BFS from 0, 3, 7. Then find DFS from 0, 3, 7. Then show the components and DFN and Low.

Second, input file will input the number of edge and set the graph up for G2, then it will find the BFS from 0, 3, 7. Then find DFS from 0, 3, 7. Then show the components and DFN and Low.

Last, input file will input the number of edge and set the graph up for G3, then it will find the BFS from 0, 3, 7. Then find DFS from 0, 3, 7. Then show the components and DFN and Low.

How to compile and run my code (on MacOS):

$ g++ -Wall -std=c++17 main\_1.cpp

$ ./a.out < input\_1.txt



1. (50%) Shortest paths: single source/all destination nonnegative weights (Dijkstra), single source/all destination negative weights DAG (Bellman-Ford), All pairs shortest paths (Floyd)

Write a C++ program to perform some basic graph functions:

1. Single source/all destination nonnegative weights (Dijkstra) (Prog.6.8)
2. Single source/all destination negative weights DAG (Bellman-Ford) (Prog. 6.9)
3. All pairs DAG shortest paths (Floyd) (Prog. 6.10)

Assume the graph is represented using weighted adjacency matrix. Add whatever you think necessary to your class Graph to implement the required functions, such as setups for setting up various graphs required and display corresponding adjacency matrix of the graph.

You should demonstrate your code by applying these three functions to graphs given below.

For (a), modify Prog. 6.8 to generate results like Fig. 6.28 in textbook (shown below) and output the computed “paths”.

You need to demonstrate your code of (a) by processing: G1, G1’, and G1” (in Part 1. Problem 6.) shown below.









G1”

For (b), modify Prog. 6.9 to display results like Fig. 6.31(b) shown below.

You need to demonstrate your code of (b) by processing: G2 and G2’ shown below.



Figure 6.31



For (c), modify Prog. 6.10 to display results like Fig. 6.32 shown below. You need to demonstrate your code of (c) by processing G3 (below) and G2 (above).

|  |  |
| --- | --- |
| (a) Digraph G3 |  |

Figure 6.32

How to use my code:

All the input file are in input.txt.

First, input file will input the number of edge and set the graph up for G1, G1’, and G1”, then it will find the ShortestPath, and show the path and the distance.

Second, input file will input the number of edge and set the graph up for G2 and G2’, then it will find the distance by Bellman–Ford algorithm and show the distance by processing.

Last, input file will input the number of edge and set the graph up for G3 and G2, then it will find the distance by Floyd-Warshall algorithm and show the matrix by processing.

X means infinite.

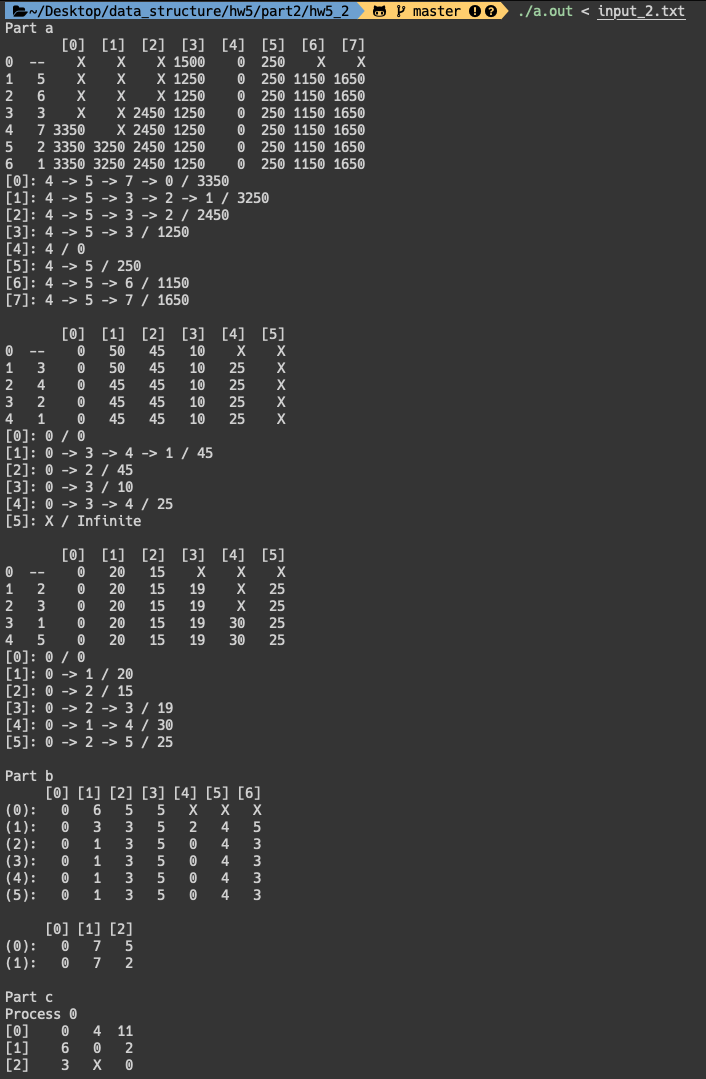
How to compile and run my code (on MacOS):

$ g++ -Wall -std=c++17 main\_2.cpp

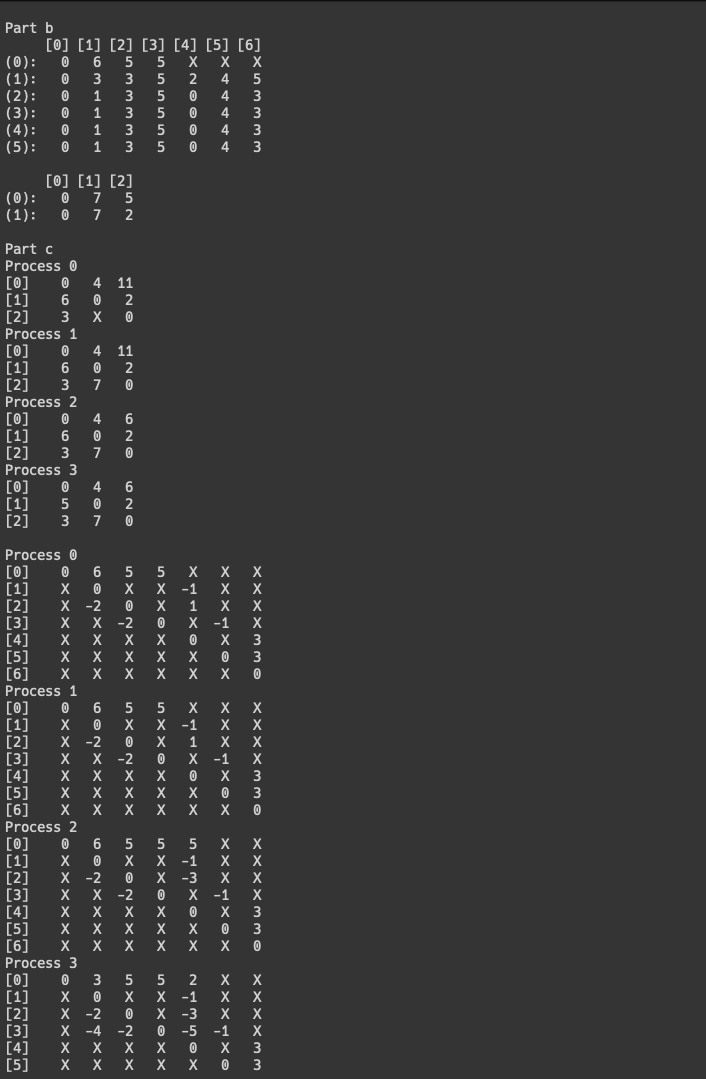
$ ./a.out < input\_2.txt

因為印出 process 太多了，所以我分頁截圖

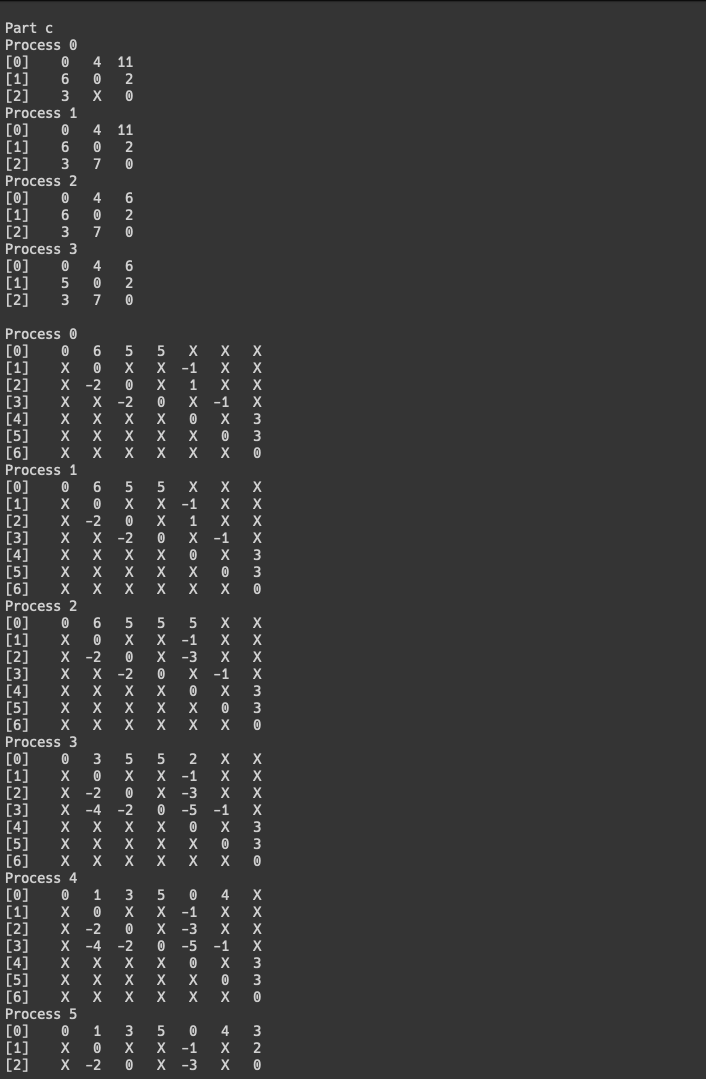
Part a



Part b



Part c\_1



Part c\_2

