EECS2040 Data Structure Hw #5 (Chapter 6 Graph) **by 108061217, 鍾永桓 due date 6/13/2021(Part 2)**

Part 2 Coding

You should submit:

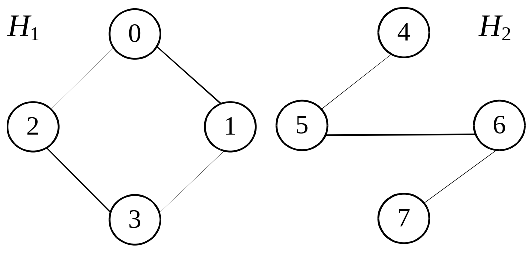
(a) All your source codes (C++ file).

(b) Show the execution trace of your program.

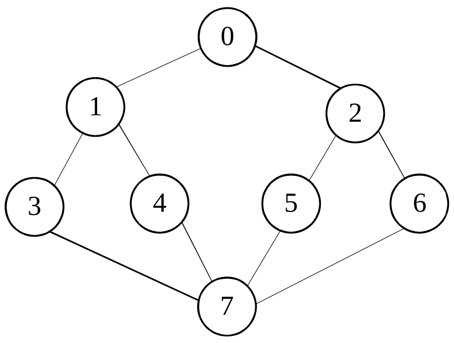
1. (50%) Graph(linked adjacency list), BFS, DFS, connected components, Computing dfn and low: Write a C++ program to perform the following basic graph functions:
   1. BFS(v) (Prog. 6.2) (v: starting vertex. You need to output the vertices visited in BFS order)
   2. DFS(v) (Prog. 6.1) (v: starting vertex. 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.

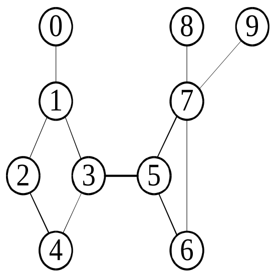
```
    Graph g1(8),g2(8),g3(10);
g1.Setup1();
//BFS
g1.BFS(0);
g1.BFS(3);
g1.BFS(7);
//DFS
g1.DFS(0);
g1.DFS(3);
g1.DFS(7);
//Components & DfnLow
g1.Components();
g1.DfnLow(3);
```



G_1



G_2



G_3

Ans:

BFS(int v):Breadth-First Search.

DFS(int v):Depth-First Search.

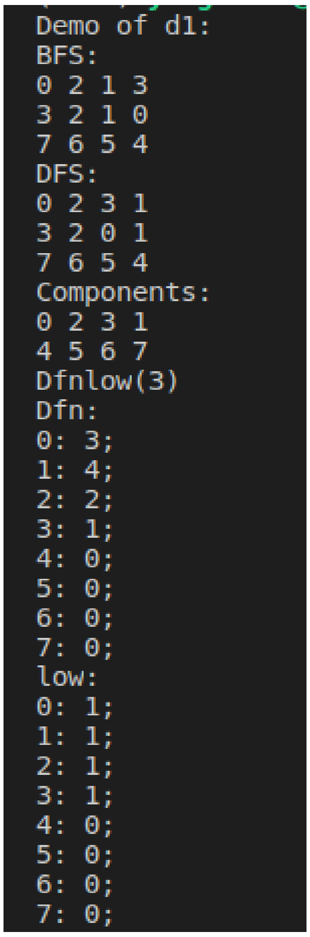
v is the vertex that the search start from.

Components():output the components of the graph.

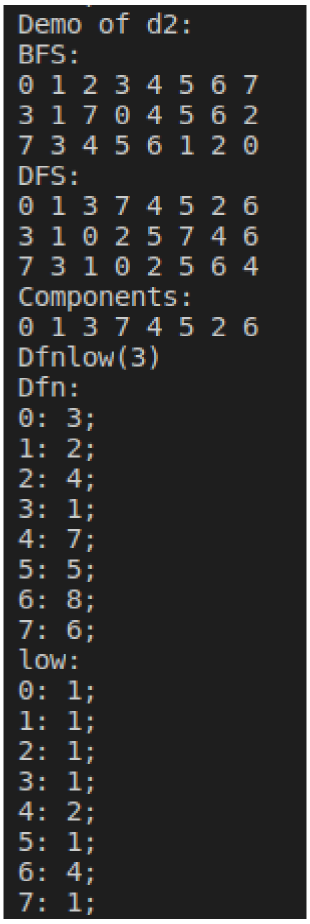
DfnLow(int x):output the Dfn and Low of the graph.

InsertEdge(int u , int v): insert the edge between u and v.

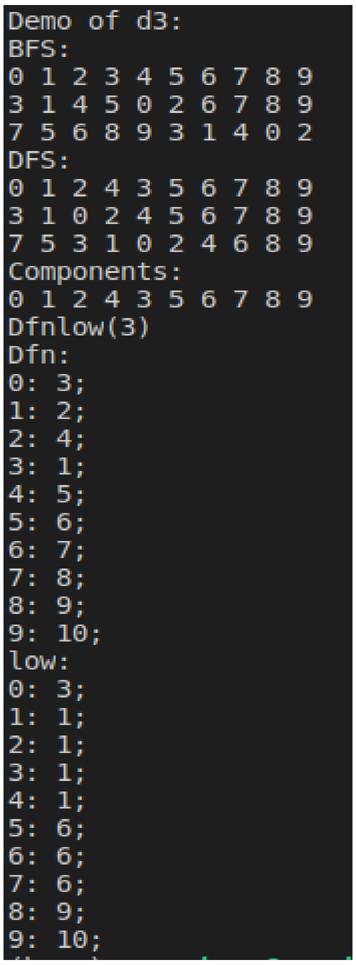
G1 G2 G3



```
Demo of d1:
BFS:
0 2 1 3
3 2 1 0
7 6 5 4
DFS:
0 2 3 1
3 2 0 1
7 6 5 4
Components:
0 2 3 1
4 5 6 7
Dfnlow(3)
Dfn:
0: 3;
1: 4;
2: 2;
3: 1;
4: 0;
5: 0;
6: 0;
7: 0;
low:
0: 1;
1: 1;
2: 1;
3: 1;
4: 0;
5: 0;
6: 0;
7: 0;
```



```
Demo of d2:
BFS:
0 1 2 3 4 5 6 7
3 1 7 0 4 5 6 2
7 3 4 5 6 1 2 0
DFS:
0 1 3 7 4 5 2 6
3 1 0 2 5 7 4 6
7 3 1 0 2 5 6 4
Components:
0 1 3 7 4 5 2 6
Dfnlow(3)
Dfn:
0: 3;
1: 2;
2: 4;
3: 1;
4: 7;
5: 5;
6: 8;
7: 6;
low:
0: 1;
1: 1;
2: 1;
3: 1;
4: 2;
5: 1;
6: 4;
7: 1;
```



```
Demo of d3:
BFS:
0 1 2 3 4 5 6 7 8 9
3 1 4 5 0 2 6 7 8 9
7 5 6 8 9 3 1 4 0 2
DFS:
0 1 2 4 3 5 6 7 8 9
3 1 0 2 4 5 6 7 8 9
7 5 3 1 0 2 4 6 8 9
Components:
0 1 2 4 3 5 6 7 8 9
Dfnlow(3)
Dfn:
0: 3;
1: 2;
2: 4;
3: 1;
4: 5;
5: 6;
6: 7;
7: 8;
8: 9;
9: 10;
low:
0: 3;
1: 1;
2: 1;
3: 1;
4: 1;
5: 6;
6: 6;
7: 6;
8: 9;
9: 10;
```

2. (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:

(a) Single source/all destination nonnegative weights (Dijkstra) (Prog.6.8)

(b) Single source/all destination negative weights DAG (Bellman-Ford) (Prog. 6.9)

(c) 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.

Ans:

The function ShortestPath(const in n ,const int v) is for (a) .

The function BellmanFord(const in n ,const int v) is for (b)

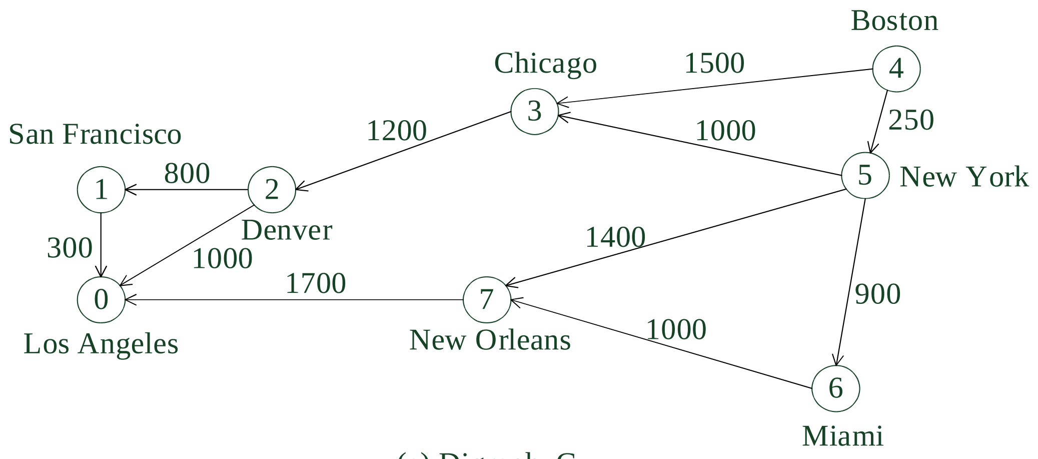
The function AllLength(const in n) is for (c)

n is the number of vertex , and v is the vertex that we start from.

The function InsertEdge(int u , int v , int w) is to insert the edge.(w is the weight of edge)

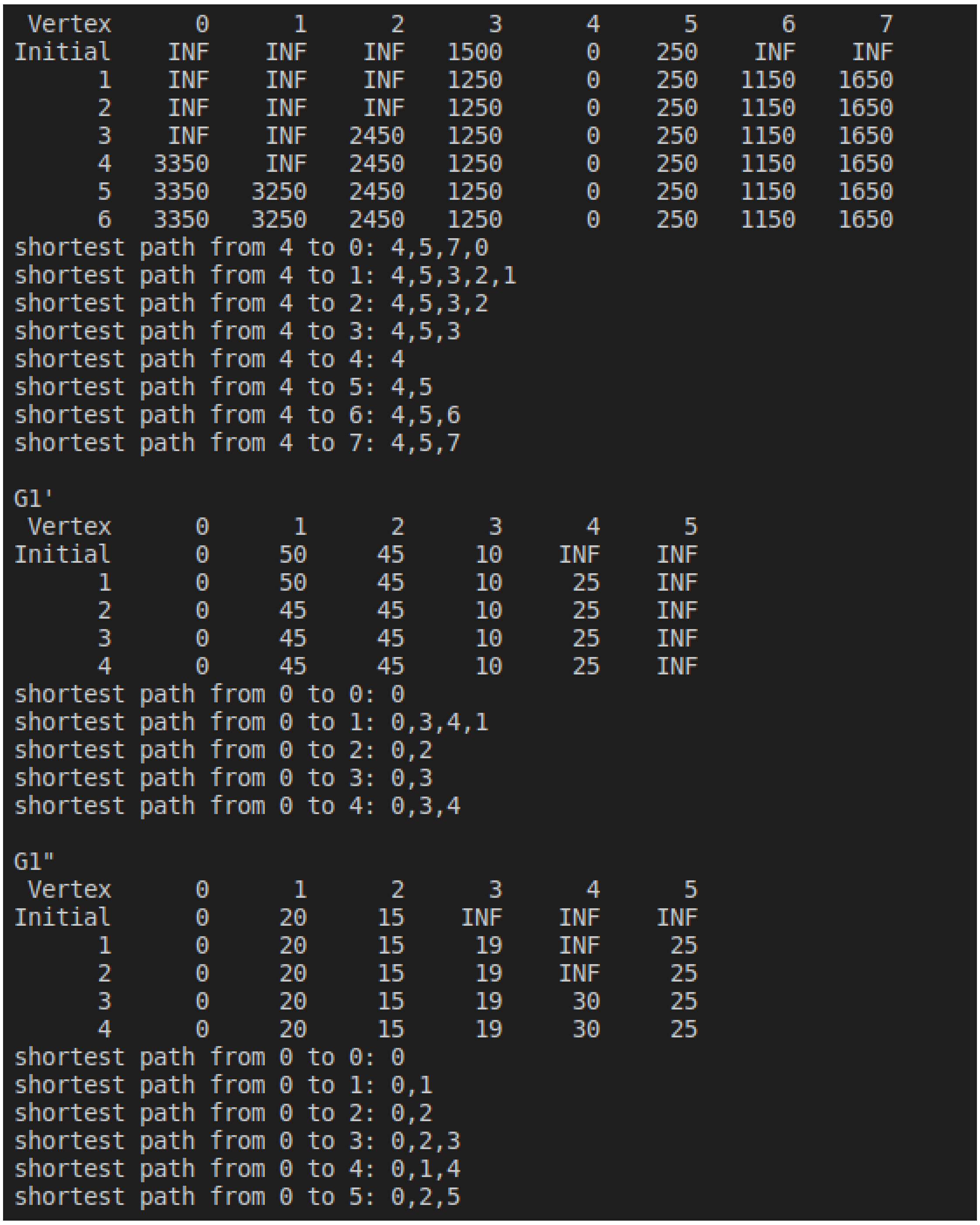
When declare the graph, I use the constructor to assign the number of vertex,the v in all function cannot large than (nubmer of vertex -1) .

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: G_1, G_1’, and G_1” (in Part 1. Problem 6.) shown below.



(a) Digraph G_1

Demo(a):

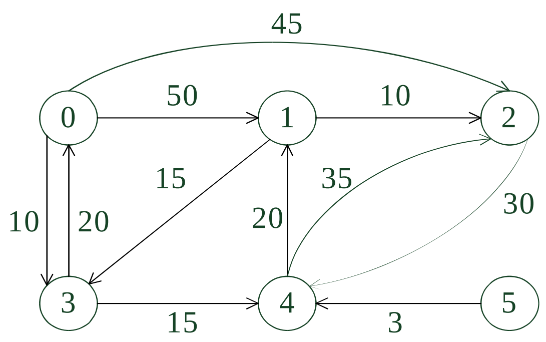


```
 Vertex      0      1      2      3      4      5      6      7
Initial    INF    INF    INF   1500      0    250    INF    INF
      1    INF    INF    INF   1250      0    250   1150   1650
      2    INF    INF    INF   1250      0    250   1150   1650
      3    INF    INF   2450   1250      0    250   1150   1650
      4   3350    INF   2450   1250      0    250   1150   1650
      5   3350   3250   2450   1250      0    250   1150   1650
      6   3350   3250   2450   1250      0    250   1150   1650
shortest path from 4 to 0: 4,5,7,0
shortest path from 4 to 1: 4,5,3,2,1
shortest path from 4 to 2: 4,5,3,2
shortest path from 4 to 3: 4,5,3
shortest path from 4 to 4: 4
shortest path from 4 to 5: 4,5
shortest path from 4 to 6: 4,5,6
shortest path from 4 to 7: 4,5,7

G1'
 Vertex      0      1      2      3      4      5
Initial      0     50     45     10    INF    INF
      1      0     50     45     10     25    INF
      2      0     45     45     10     25    INF
      3      0     45     45     10     25    INF
      4      0     45     45     10     25    INF
shortest path from 0 to 0: 0
shortest path from 0 to 1: 0,3,4,1
shortest path from 0 to 2: 0,2
shortest path from 0 to 3: 0,3
shortest path from 0 to 4: 0,3,4

G1"
 Vertex      0      1      2      3      4      5
Initial      0     20     15    INF    INF    INF
      1      0     20     15     19    INF     25
      2      0     20     15     19    INF     25
      3      0     20     15     19     30     25
      4      0     20     15     19     30     25
shortest path from 0 to 0: 0
shortest path from 0 to 1: 0,1
shortest path from 0 to 2: 0,2
shortest path from 0 to 3: 0,2,3
shortest path from 0 to 4: 0,1,4
shortest path from 0 to 5: 0,2,5
```

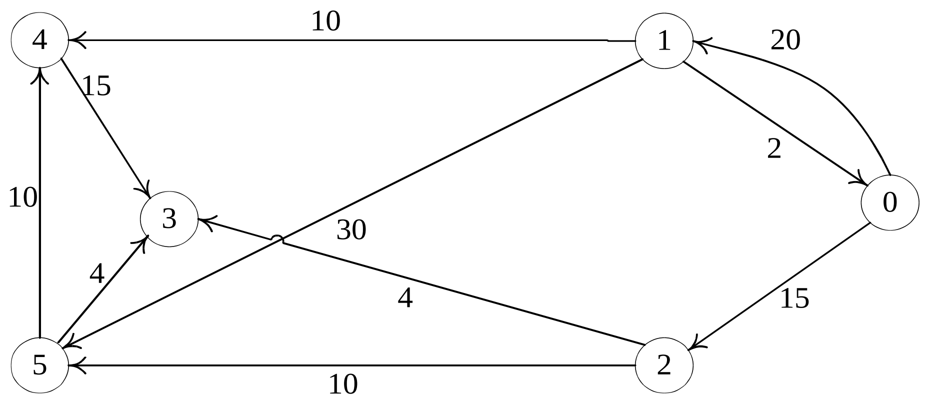
Iteration	Vertex selected	Distance							
		LA	SF	DEN	CHI	BOST	NY	MIA	NO
		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Initial	----	ɱ	ɱ	ɱ	1500	0	250	ɱ	ɱ
1	5	ɱ	ɱ	ɱ	1250	0	250	1150	1650
2	6	ɱ	ɱ	ɱ	1250	0	250	1150	1650
3	3	ɱ	ɱ	2450	1250	0	250	1150	1650
4	7	3350	ɱ	2450	1250	0	250	1150	1650
5	2	3350	3350	2450	1250	0	250	1150	1650
6	1	3350	3350	2450	1250	0	250	1150	1650



路徑	長度
1) 0, 3	10
2) 0, 3, 4	25
3) 0, 3, 4, 1	45
4) 0, 2	45

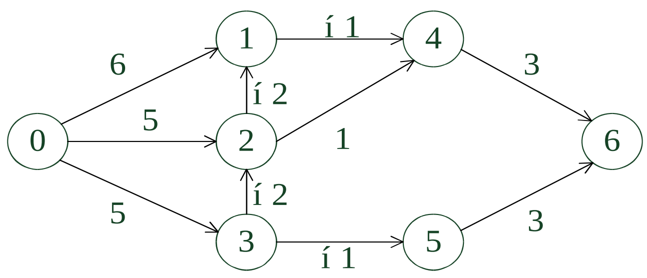
(a) Digraph G_1'

(b) 從 0 出發的最短路徑



G_1”

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: G_2 and G_2’ shown below.

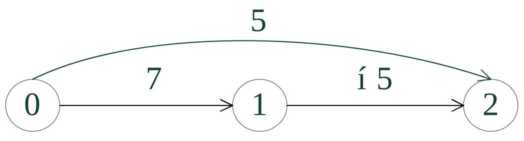


(a) Digraph G_2

k	$dist^k[7]$						
	0	1	2	3	4	5	6
1	0	6	5	5	ɱ	ɱ	ɱ
2	0	3	3	5	5	4	ɱ
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

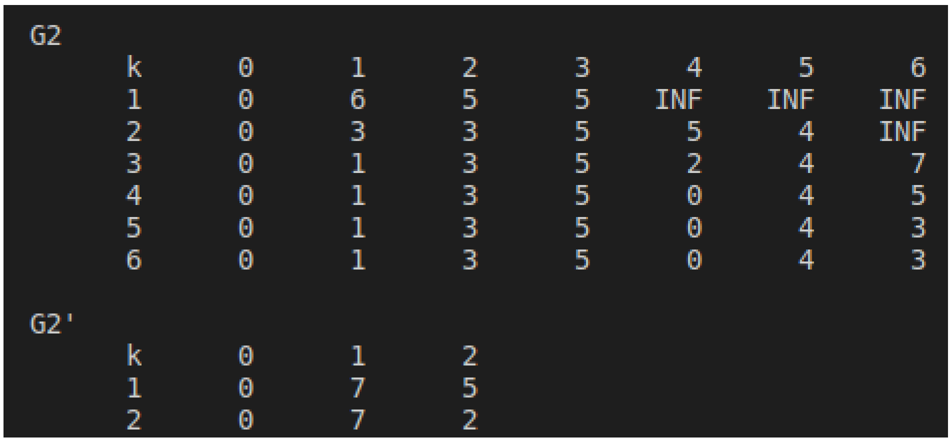
(b) $dist^k$

Figure 6.31



G_2'

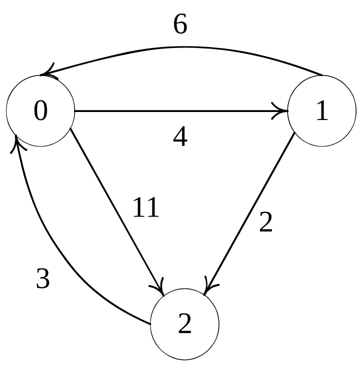
Demo(b)



```
G2
        k       0       1       2       3       4       5       6
        1       0       6       5       5     INF     INF     INF
        2       0       3       3       5       5       4     INF
        3       0       1       3       5       2       4       7
        4       0       1       3       5       0       4       5
        5       0       1       3       5       0       4       3
        6       0       1       3       5       0       4       3

G2'
        k       0       1       2
        1       0       7       5
        2       0       7       2
```

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 G_3 (below) and G_2 (above).



(a) Digraph G_3

A^{-1}	0	1	2
0	0	4	11
1	6	0	2
2	3	∞	0

(b) A^{-1}

A^{0}	0	1	2
0	0	4	11
1	6	0	2
2	3	7	0

(c) A^{0}

A^{1}	0	1	2
0	0	4	6
1	6	0	2
2	3	7	0

(d) A^{1}

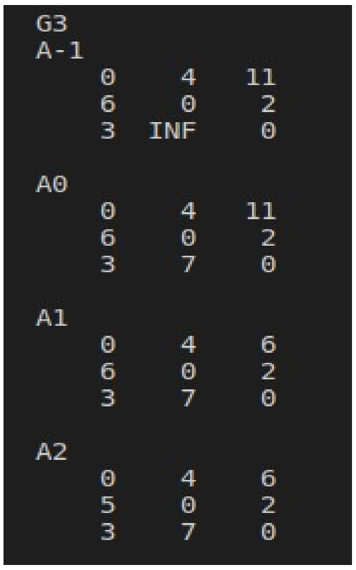
A^{2}	0	1	2
0	0	4	6
1	5	0	2
2	3	7	0

(e) A^{2}

Figure

Demo(c):

G3



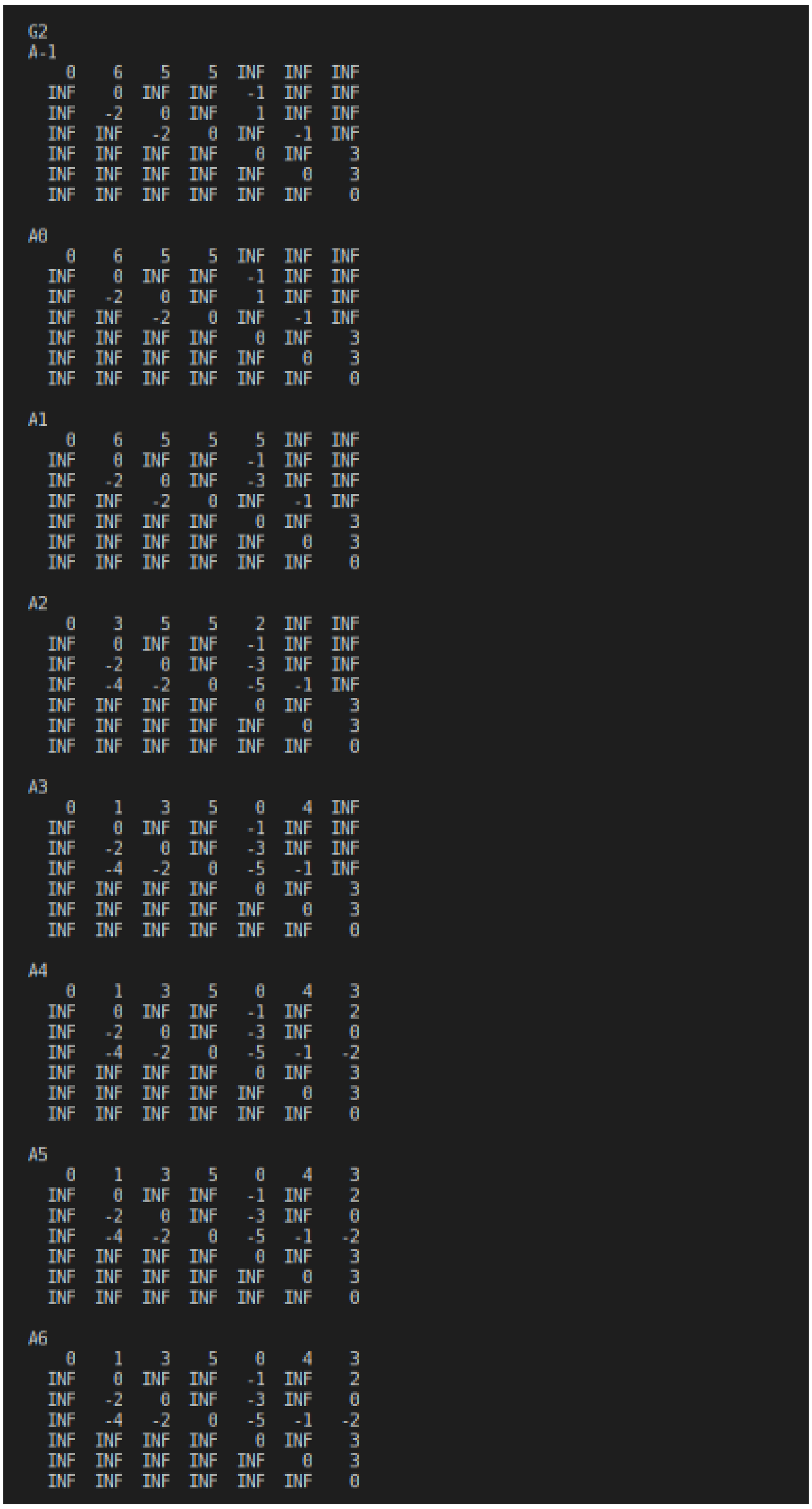
```
G3
A-1
        0     4    11
        6     0     2
        3   INF     0

A0
        0     4    11
        6     0     2
        3     7     0

A1
        0     4     6
        6     0     2
        3     7     0

A2
        0     4     6
        5     0     2
        3     7     0
```

G2



```
G2
A-1
    0    6    5    5  INF  INF  INF
  INF    0  INF  INF   -1  INF  INF
  INF   -2    0  INF    1  INF  INF
  INF  INF   -2    0  INF   -1  INF
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A0
    0    6    5    5  INF  INF  INF
  INF    0  INF  INF   -1  INF  INF
  INF   -2    0  INF    1  INF  INF
  INF  INF   -2    0  INF   -1  INF
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A1
    0    6    5    5    5  INF  INF
  INF    0  INF  INF   -1  INF  INF
  INF   -2    0  INF   -3  INF  INF
  INF  INF   -2    0  INF   -1  INF
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A2
    0    3    5    5    2  INF  INF
  INF    0  INF  INF   -1  INF  INF
  INF   -2    0  INF   -3  INF  INF
  INF   -4   -2    0   -5   -1  INF
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A3
    0    1    3    5    0    4  INF
  INF    0  INF  INF   -1  INF  INF
  INF   -2    0  INF   -3  INF  INF
  INF   -4   -2    0   -5   -1  INF
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A4
    0    1    3    5    0    4    3
  INF    0  INF  INF   -1  INF    2
  INF   -2    0  INF   -3  INF    0
  INF   -4   -2    0   -5   -1   -2
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A5
    0    1    3    5    0    4    3
  INF    0  INF  INF   -1  INF    2
  INF   -2    0  INF   -3  INF    0
  INF   -4   -2    0   -5   -1   -2
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0

A6
    0    1    3    5    0    4    3
  INF    0  INF  INF   -1  INF    2
  INF   -2    0  INF   -3  INF    0
  INF   -4   -2    0   -5   -1   -2
  INF  INF  INF  INF    0  INF    3
  INF  INF  INF  INF  INF    0    3
  INF  INF  INF  INF  INF  INF    0
```