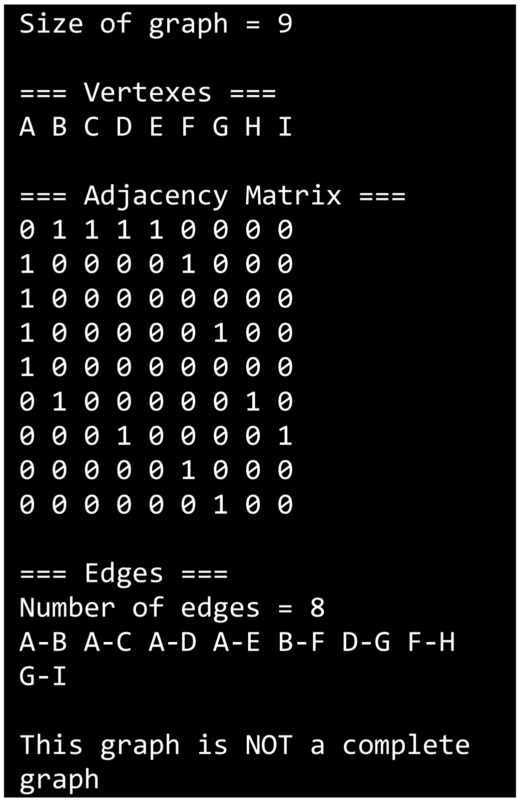
Data Structures and Algorithms Laboratory		
Laboratory 11: Graph	**School of Information Technology**	
Name: Pattarapon Bunchuai	**ID:64321503044**	**Section:3**
Date:	**Due date: on the LMS**	

Objective

- To implement the graph structure with adjacency matrix.
- To create a weighted digraph and its adjacency matrix

<u>Exercise 1 (In-Class):</u> Given the incomplete code below, fill in the code add vertex and edge in a graph and get the following result.

Expected Result:



```
Size of graph = 9

=== Vertexes ===
A B C D E F G H I

=== Adjacency Matrix ===
0 1 1 1 1 0 0 0 0
1 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0
0 0 0 1 0 0 0 0 1
0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0

=== Edges ===
Number of edges = 8
A-B A-C A-D A-E B-F D-G F-H
G-I

This graph is NOT a complete
graph
```

Graph.java

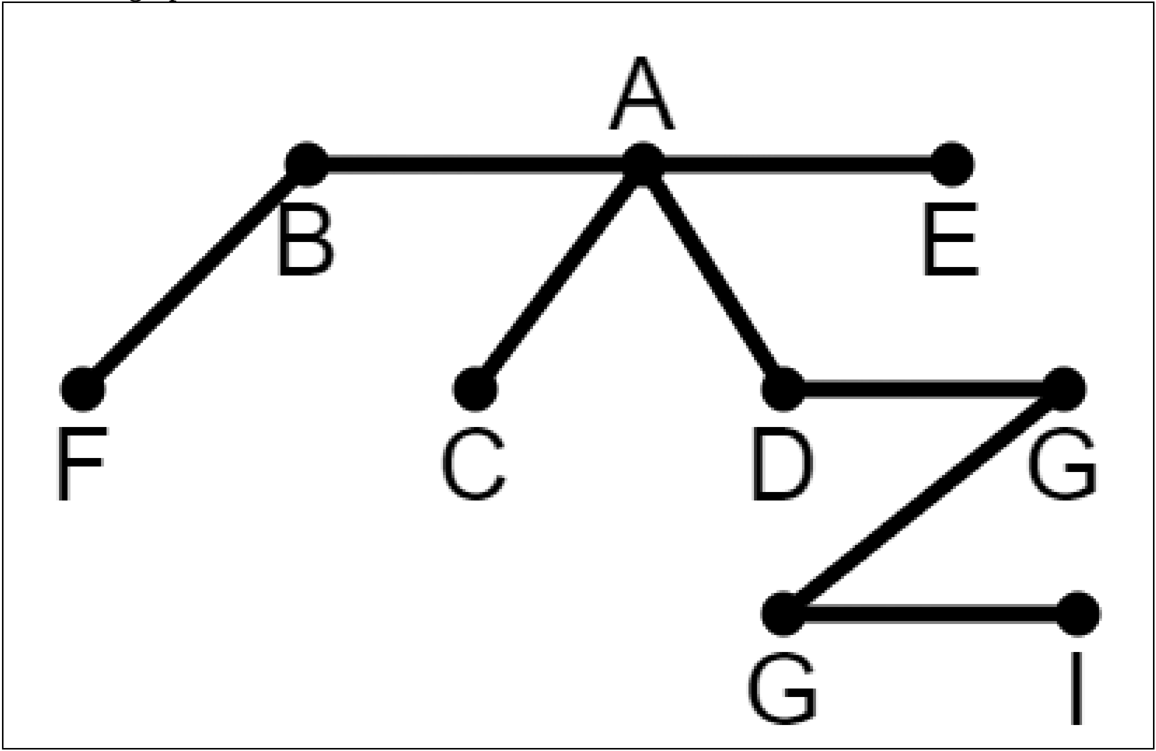
```
package algorithm;

public class Graph {
    private static final int MAX_VERTICES = 20;     // allow only 20
vertices
    protected char[] vertex;         // list of vertices
    protected int[][] adjMatrix;      // adjacency matrix
    protected int numVertices;           //number of vertices
    protected int numEdges;

    //default constructor
    public Graph(){
        //create vertex objects
        vertex = new char[MAX_VERTICES];
        //create adjacency matrix objects
        adjMatrix = new int[MAX_VERTICES][MAX_VERTICES];
        numVertices = 0;
        numEdges = 0;

        //set all elements of adjacency matrix to be zero (no edges)
        for(int i=0; i< MAX_VERTICES; i++) {
            for(int j=0; j< MAX_VERTICES; j++) {
                adjMatrix[i][j] = 0;
            }
        }
    }

    //add new vertex with title
    public void addVertex(char title) {
        vertex[numVertices++] = title;
    }

    //add edge between two vertices
    public void addEdge(int start, int end) {
        // set value in adjacency matrix
        adjMatrix[start][end] = 1;
        adjMatrix[end][start] = 1;
        numEdges++;
    }

    //graph's size
    public void showSize(){
        System.out.println("Size of graph = " + numVertices);
        System.out.println();
    }

    // display each vertex's title
    public void showVertex() {
        System.out.println("=== Vertexes ===");
        for (int i = 0; i < numVertices; i++) {
            System.out.print(vertex[i] + " ");
        }
        System.out.println("\n");
    }

    //display adjacency matrix
    public void showAdjacency() {
        System.out.println("=== Adjacency Matrix ===");
        for (int i = 0; i < numVertices; i++) {
            for (int j = 0; j < numVertices; j++) {
                System.out.print(adjMatrix[i][j] + " ");
            }
            System.out.println();
        }
        System.out.println();
    }

    //display all edges
    public void showEdge() {
        System.out.println("=== Edges ===");
        System.out.println("Number of edge = " + numEdges);
        for (int i = 0; i < numVertices; i++) {
            for (int j = 0; j < numVertices; j++) {
                if (adjMatrix[i][j] == 1){
                    System.out.print(vertex[i] + "-" + vertex[j] + " 
");
                }
            }
        }
        System.out.println("\n");
    }

    //check if the graph complete?
    public void checkComplete() {
        int complete = vertex.length * (vertex.length-1) / 2;
        System.out.println("This graph is " + (numVertices == complete 
? "" : "NOT") + " a complete graph");
    }

}
```

TestGraph.java

```
public class TestGraph {
      public static void main(String[] args) {
          Graph graph = new Graph();

          //create vertices
          graph.addVertex('A'); //0
          graph.addVertex('B');//1
          graph.addVertex('C');//2
          graph.addVertex('D'); //3
          graph.addVertex('E'); //4
          graph.addVertex('F'); //5
          graph.addVertex('G'); //6
          graph.addVertex('H');//7
          graph.addVertex('I');//8

            //connect vertices
          graph.addEdge(0,1);  //AB
```

```
            graph.addEdge(0,2);  //AC
            graph.addEdge(0,3);  //AD
            graph.addEdge(0,4); //AE
            graph.addEdge(1,5); //BF
            graph.addEdge(5,7);  //FH
            graph.addEdge(3,6); //DG
            graph.addEdge(6,8);  //GI

            //size
            graph.showSize();

            //show vertexes
            graph.showVertex();

            //show adjacency matrix
            graph.showAdjacency();

            //show edges
            graph.showEdge();

            //a complete graph?
            graph.checkComplete();
        }
}
```

Draw the graph created in the exercise 1.



Exercise 2 (Homework): Modify the exercise 1 to support the creation of a weighted digraph and get the following result.

Expected result:

```
Size of digraph = 5

=== Vertexes ===
A B C D E

=== Adjacency Matrix ===
0 3 4 1 0
0 0 0 0 2
0 0 0 3 0
0 2 3 0 0
0 0 4 0 0

=== Edges ===
Number of edges = 8
A-B(3) A-C(4) A-D(1) B-E(2) C-D(3) D-B(2) D-C(3) E-C(4)

This digraph is NOT a complete digraph

The path length ABECD = 12
```

DiGraph.java

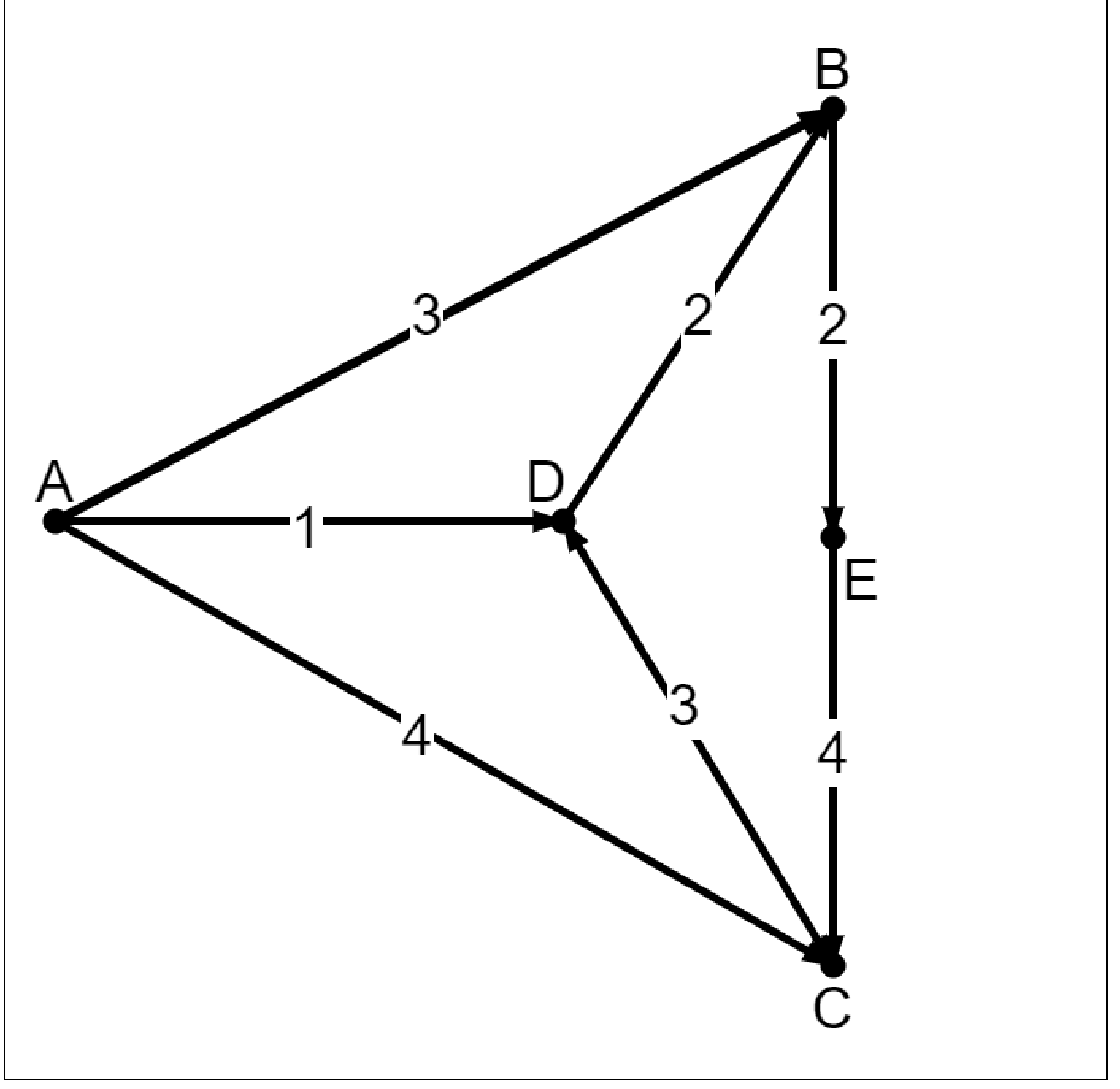
```
package algorithm;

public class DiGraph extends Graph {

    @Override
    public void addEdge(int start, int end) {/* Do nothing*/}

    public void addEdge(int start, int end, int weight){
        adjMatrix[start][end] = weight;
        numEdges++;
    }

    @Override
    public void showEdge() {
        System.out.println("=== Edges ===");
        System.out.println("Number of edge = " + numEdges);
        for (int i = 0; i < numVertices; i++) {
            for (int j = 0; j < numVertices; j++) {
                int weight = adjMatrix[i][j];
                if (weight > 0){
                    System.out.print(vertex[i] + "-" + vertex[j] + "("
+ weight + ") ");
                }
            }
        }
        System.out.println("\n");
    }

    //compute path length
    public int findPathLength(int[] path) {
        int length = 0;
        for (int i = 0; i < path.length - 1; i++){
            // add the weight of the edge to the length which is 0 or
more
            length += adjMatrix[path[i]][path[i+1]];
        }
        return length;
    }
}
```

TestDiGraph.java

```
public class TestDiGraph {
	public static void main(String[] args) {
		DiGraph graph = new DiGraph();

		//create vertices
		graph.addVertex('A'); //0
		graph.addVertex('B');//1
		graph.addVertex('C');//2
		graph.addVertex('D'); //3
		graph.addVertex('E'); //4

		  //connect vertices
		graph.addEdge(0,1,3);  //AB weight = 3
		graph.addEdge(0,2,4);  //AC weight = 4
		graph.addEdge(0,3,1);  //AD weight = 1
		graph.addEdge(1,4,2);  //BE weight = 2
		graph.addEdge(2,3,3);  //CD weight = 3
		graph.addEdge(3,1,2);  //DB weight = 2
		graph.addEdge(3,2,3);  //DC weight = 3
		graph.addEdge(4,2,4);  //EC weight = 4

		//size
		graph.showSize();

		//show vertexes
		graph.showVertex();

		//show adjacency matrix
		graph.showAdjacency();

		//show edges
		graph.showEdge();

		//a complete graph?
		graph.checkComplete();

		//weighted path length
		//ABECD
```

```
            int[] path = {0,1,4,2,3};
            System.out.println("The path length ABECD = " +
graph.findPathLength(path));
       }
}
```

Draw the graph created in the exercise 2.


B
A
D
E
C
3
2
2
1
4
3
4