A ***graph*** *G is* a set *V* of ***vertices*** and a collection *E* of pairs of vertices from *V*, called ***edges***. The aim of this worksheet is to make **two implementations** of the Graph ADT one based on the ***adjacency matrix*** and other on the ***adjacency map*** representation. Figure 1 illustrates a graph with both representations.

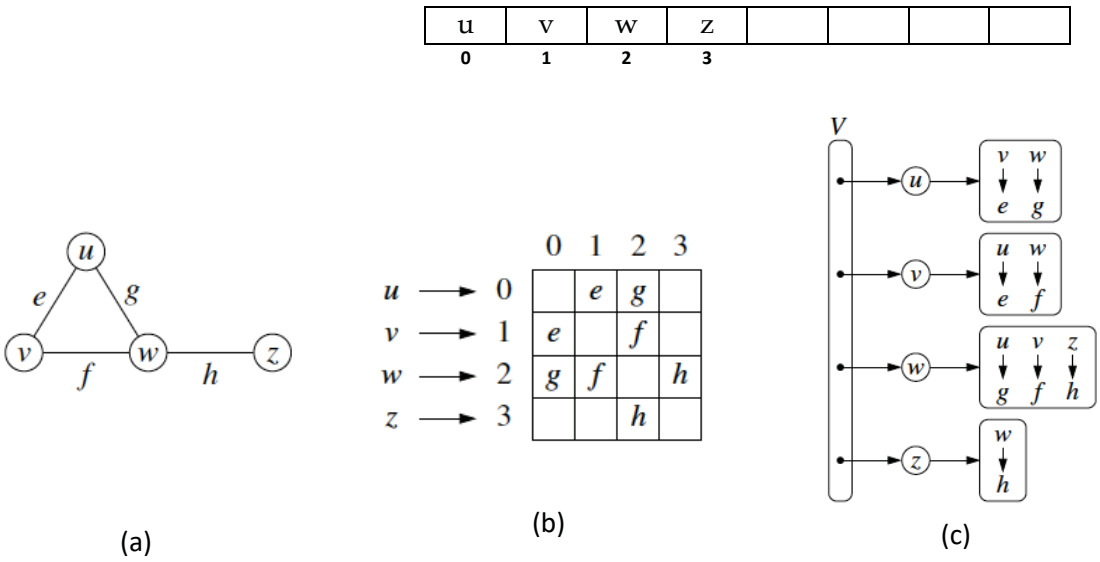

Figure1 - (a) An undirected graph G; (b) a schematic representation of the adjacency matrix
(c) a schematic representation of the adjacency map structure

The Edge class stores the weight of the edge and its both endpoints vertices. The Vertex class, in the adjacency map representation, stores the information associated with the vertex and a map with its outgoing edges.
A graph instance maintains the number of vertices and edges of the graph, a boolean variable that designates whether the graph is directed and a list with all its vertices.

1. Complete the generic class MatrixGraph<V,E> implementing the following methods:
   a. `public Collection<Edge<V, E>> edges()`
   b. `public Collection<Edge<V, E>> outgoingEdges(V vert)`
   c. Test the methods.

2. Complete the generic class MapGraph<V, E> implementing the methods:
   a. `public Collection<Edge<V, E>> incomingEdges(V vert)`
   b. `public Collection<V> adjVertices(V vert)`
   c. Test the methods.

3. Complete the Algorithms class developing the following methods and testing them:
   a. **Breadth-first search** of a graph starting in a vertex with a given information
   b. **DepthFirstSearch** of a graph starting in a vertex with a given information
   c. Minimum distance graph using **Floyd-Warshall** algorithm
   d. Create unit test for the **Floyd-Warshall** algorithm for both representations
   e. **Shortest-path** from a source vertex to a destination vertex of the graph, using Dijkstra's algorithm
   f. **Shortest-paths** from a source vertex and all other vertices of the graph, using Dijkstra's algorithm

4. Implement the **HighwayMap class**, which is intended to represent a map of cities and the interconnected highways. Each city has a name. Highways have name, distance (double) and cost (double).
   a. Declare the classes, and their attributes, using an adjacency map graph to represent the map (suggestion: create an external public class to represent a highway)
   b. Implement methods to insert cities and highways in the map.
   c. Implement methods to return the set of highways departing from a particular city and the set of cities connected through just one highway (one hop).
   d. Implement a method to check if two cities are reachable through highways.
   e. Implement a method to return a path between two cities.
   f. Implement a method to return the path between two cities using the minimum number of highways.
   g. Implement a method to return the path between two cities with the minimum distance.
   h. Implement a method to return the path between two cities with the minimum cost.
   i. Create unit tests for methods developed in 1.e to 1.h.