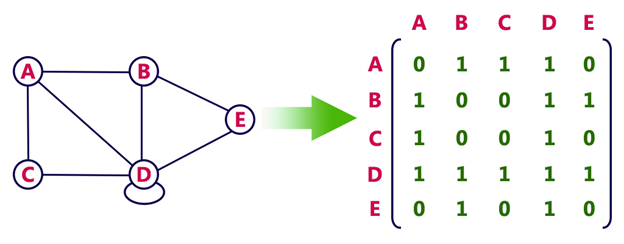
***Graph Representations***

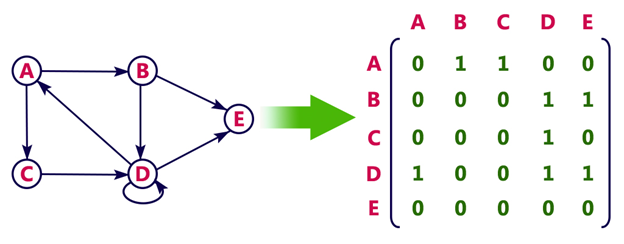
## **Adjacency Matrix**

* It is used to represent which nodes are adjacent to each other. i.e. is there any edge connecting nodes to a graph.
* In this representation, we have to construct a nXn matrix A. If there is any edge from a vertex i to vertex j, then the corresponding element of A, ai,j = 1, otherwise ai,j= 0.
* If there is any weighted graph then instead of 1s and 0s, we can store the weight of the edge.

**Undirected graph representation**



**Directed graph representation**



## **Incidence Matrix**

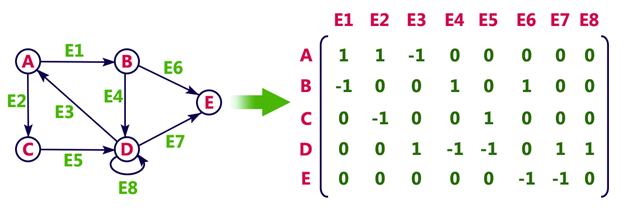
In **Incidence matrix representation**, graph can be represented using a matrix of size:

Total number of vertices by total number of edges.

It means if a graph has 4 vertices and 6 edges, then it can be represented using a matrix of 4X6 class. In this matrix, columns represent edges and rows represent vertices.

This matrix is filled with either **0 or 1** or -1. Where,

* 0 is used to represent row edge which is not connected to column vertex.
* 1 is used to represent row edge which is connected as outgoing edge to column vertex.
* -1 is used to represent row edge which is connected as incoming edge to column vertex.



## **Adjacency List**

* Adjacency list is a linked representation.
* In this representation, for each vertex in the graph, we maintain the list of its neighbors. It means, every vertex of the graph contains list of its adjacent vertices.
* We have an array of vertices which is indexed by the vertex number and for each vertex v, the corresponding array element points to a **singly linked list** of neighbors of v.

