# Assignment No - 6 (Group C)

## Problem statement:

There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight take to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Check whether the graph is connected or not. Justify the storage representation used

## Pre-requisite

- Knowledge of C++ programming
- Knowledge of Graph data structure

## Objective

- To understand concept of Graph data structure
- To understand concept of representation of graph

## Input

- Number of cities.
- Time required to travel from one city to another.

## Output

- Create Adjacency matrix to represent path between various cities

## Software and Hardware requirements:-

1. **Operating system:** Linux- Ubuntu 16.04 to 17.10, or Windows 7 to 10,
2. **RAM**- 2GB RAM (4GB preferable)
3. C++ / gcc compiler- / 64 bit Fedora, eclipse IDE

## Theory-

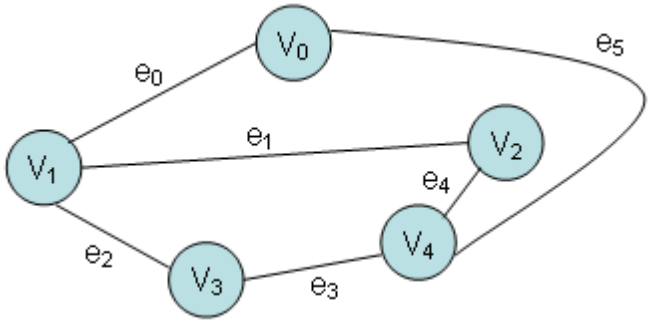
Graphs are the most general data structure. They are also commonly used data structures.

**Graph definitions:**

A non-linear data structure consisting of nodes and links between nodes.

## Undirected graph definition:

- An undirected graph is a set of nodes and a set of links between the nodes.
- Each node is called a **vertex**, each link is called an **edge**, and each edge connects two vertices.
- The order of the two connected vertices is unimportant.
- An undirected graph is a finite set of vertices together with a finite set of edges. Both sets might be empty, which is called the empty graph.




## Graph Implementation:

Different kinds of graphs require different kinds of implementations, but the fundamental concepts of all graph implementations are similar. We'll look at several representations for one particular kind of graph: directed graphs in which loops are allowed.

## Representing Graphs with an Adjacency Matrix

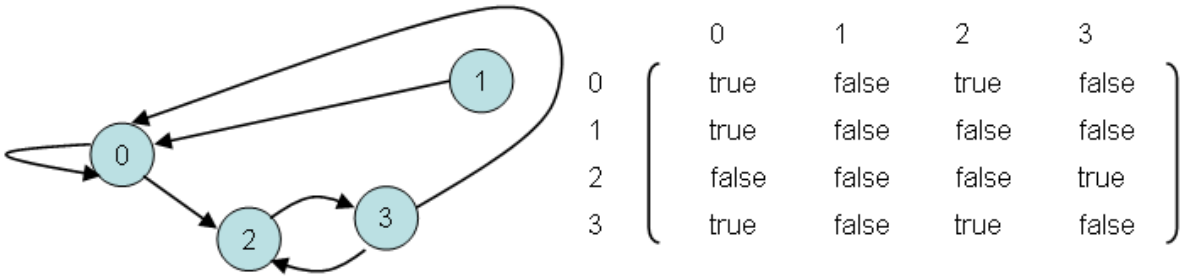



Fig: Graph and adjacency matrix

**Definition**:

- An adjacency matrix is a square grid of true/false values that represent the edges of a graph.
- If the graph contains *n* vertices, then the grid contains *n* rows and *n* columns.
- For two vertex numbers *i* and *j*, the component at row *i* and column *j* is true if there is an edge from vertex *i* to vertex *j*; otherwise, the component is false.

  We can use a two-dimensional array to store an adjacency matrix:

  **boolean[][] adjacent = new boolean[4][4];**

Once the adjacency matrix has been set, an application can examine locations of the matrix to determine which edges are present and which are missing.

## Representing Graphs with Edge Lists

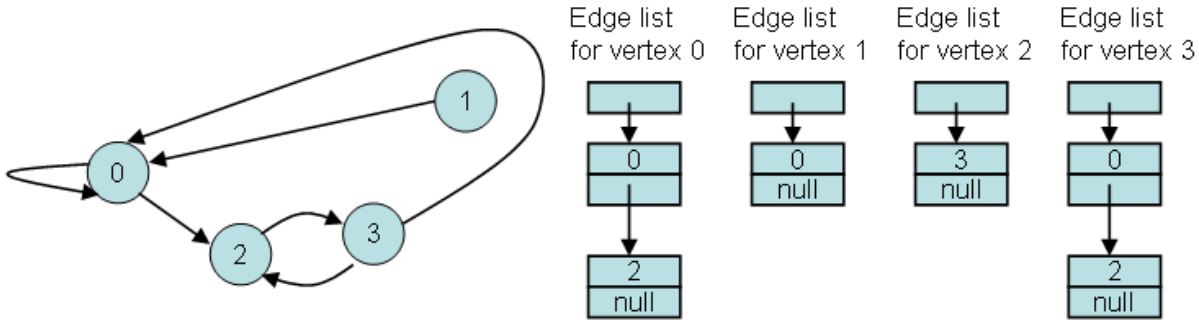



Fig: Graph and adjacency list for each node

**Definition**:

- A directed graph with *n* vertices can be represented by *n* different linked lists.
- List number *i* provides the connections for vertex *i*.
- For each entry *j* in list number *i*, there is an edge from *i* to *j*.

Loops and multiple edges could be allowed.

**Representing Graphs with Edge Sets**

To represent a graph with *n* vertices, we can declare an array of *n* sets of integers. For example:

**IntSet[] connections = new IntSet[10]; // 10 vertices**

A set such as connections[*i*] contains the vertex numbers of all the vertices to which vertex *i* is connected.

**Conclusion:**

This program gives us the knowledge of adjacency matrix graph.

**Sample Oral Questions :**

1. What are different ways to represent the graph? Give suitable example.
2. What is time complexity of function to create adjacency matrix?