Lab#12:

"Graphs"

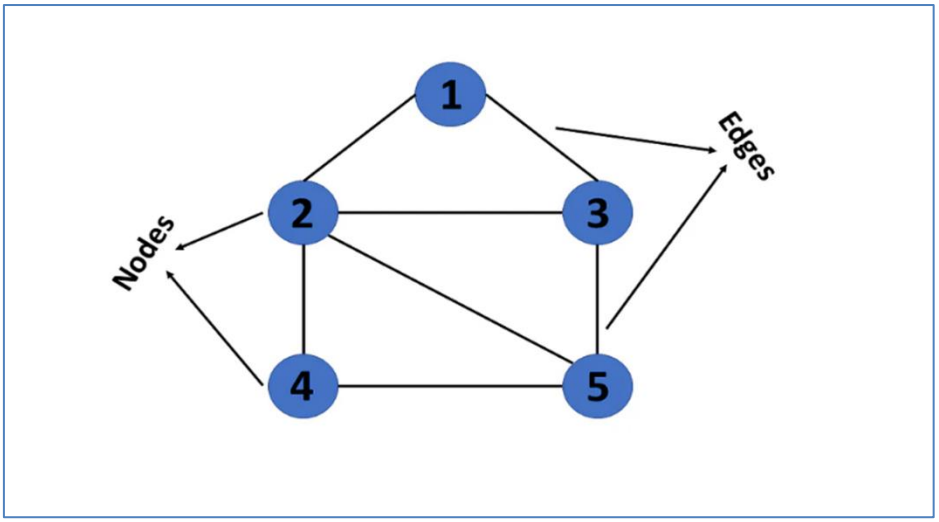
Objective:

The objective of this lab are as follows

- Introduction to graph theory
- Components of graphs
- Useful terminologies
- Representation of graphs
- Introduction to STL
- Implementation of graph representation in c++

Introduction:

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.



Types of graphs:

There are two types of graphs with respect to the direction of the edges

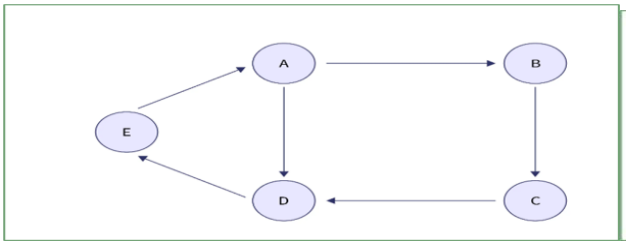
1) Directed Graphs
2) Undirected Graphs

Directed Graphs:

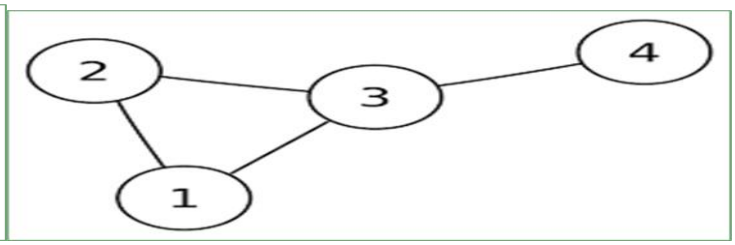
One vertex is directed towards another vertex through an edge between them

Undirected or Bi directed Graphs:

One vertex is connected with another vertex through an edge between them but no direction is specified



Directed Graph



Undirected Graph

Furthermore the directed and undirected graphs can be weighted and unweighted graphs.

Weighted Graphs:

These are the graph data structures in which the edges are given some weight or value based on the type of graph we are representing.

Unweighted Graphs:

Unweighted graphs are the graph data structure that are not associated with any weight or value.

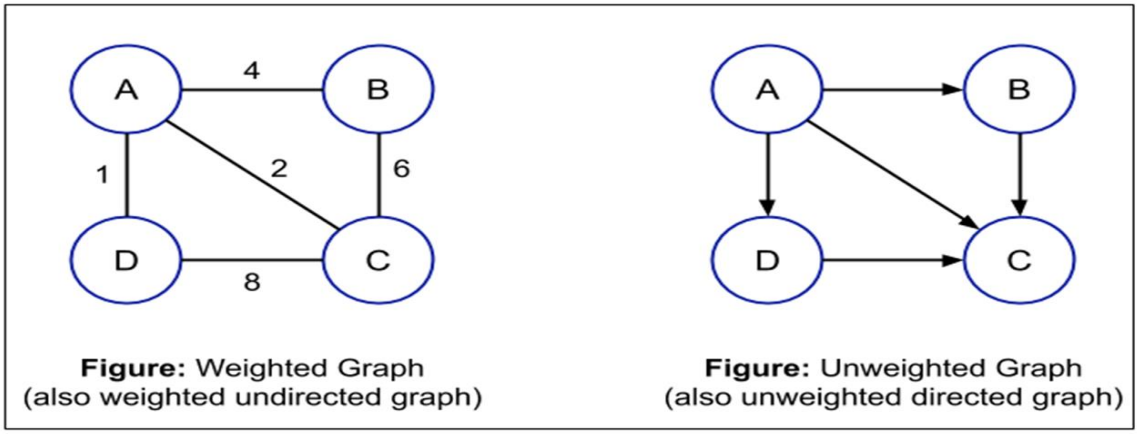
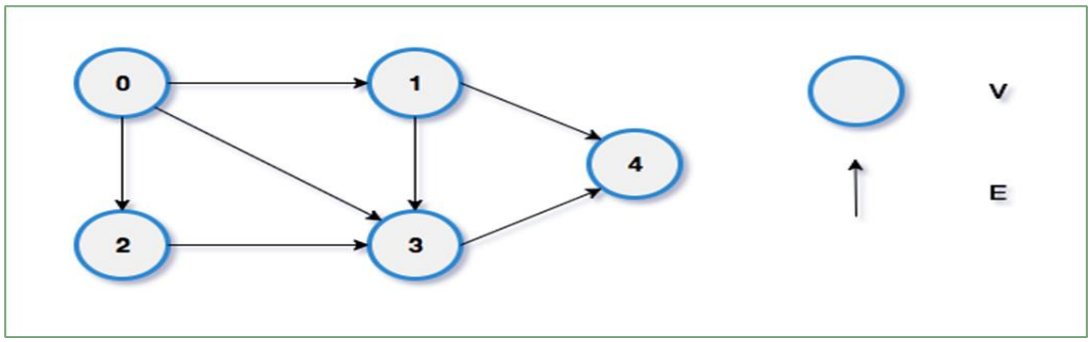


Figure: Weighted Graph
(also weighted undirected graph)

Figure: Unweighted Graph
(also unweighted directed graph)

Path:

- In graph theory the path is any route along the edges of a graph. A path may follow a single edge directly between two vertices or It may follow multiple edges through multiple vertices. Node cant be repeated.
- Paths from 0 to 4{0->1->4,0->3->4,0->2->3->4}



Applications of Graph:

Many applications are built on the basis of graph theory,some of the well known examples are as follows;

- **Facebook;** Facebook uses the graph theory in its algorithm
- **Google Maps;** Google maps use the concept of graph theory in its implementation
- **WWW;** World Wide Web uses the concept of graph theory as well

Representation of Graphs:

It can be represented in two ways

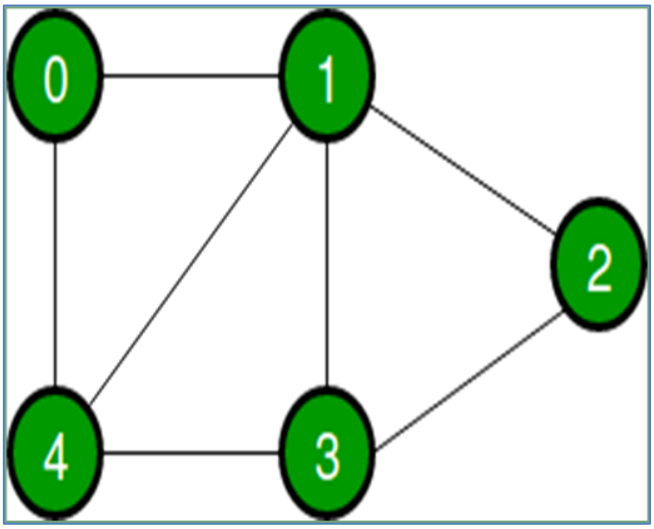
1. Adjacency Matrix
2. Adjacency List

Adjacency Matrix:

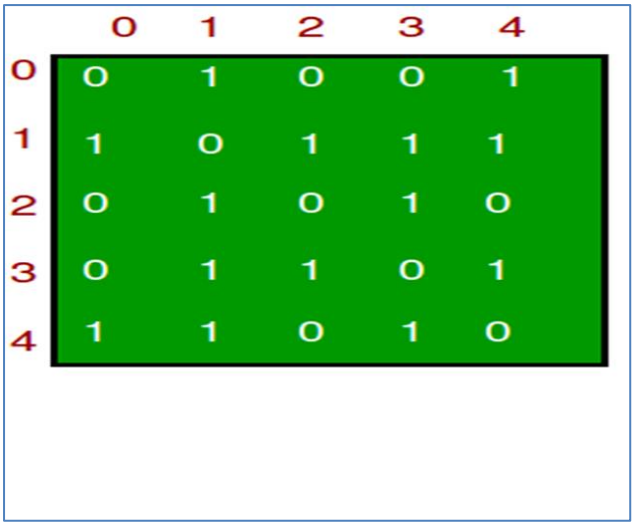
Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph

Adjacency List:

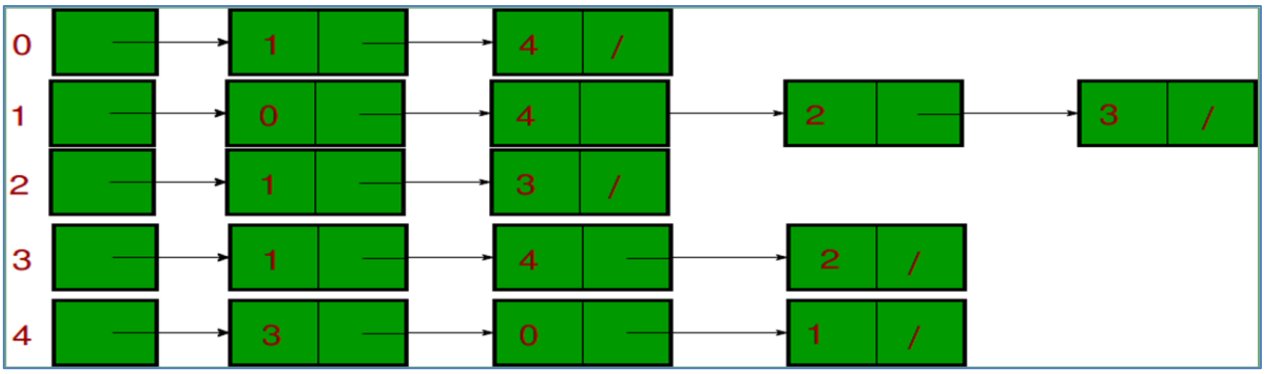
An array of lists is used. The size of the array is equal to the number of vertices



Graph



Adjacency Matrix



Adajacency List

Introduction To STL:

C++ STL (standard template library) is a software library for the C++ language . It provides a collection of templates representing containers, iterators, algorithms, and function objects. These can be very handful to implement data structures like linked list, arrays, stacks, etc,These can also be handful to apply sorting and searching algorithms.

Tasks:

Task#01:

Statement: C++ Code for adjacency Matrix

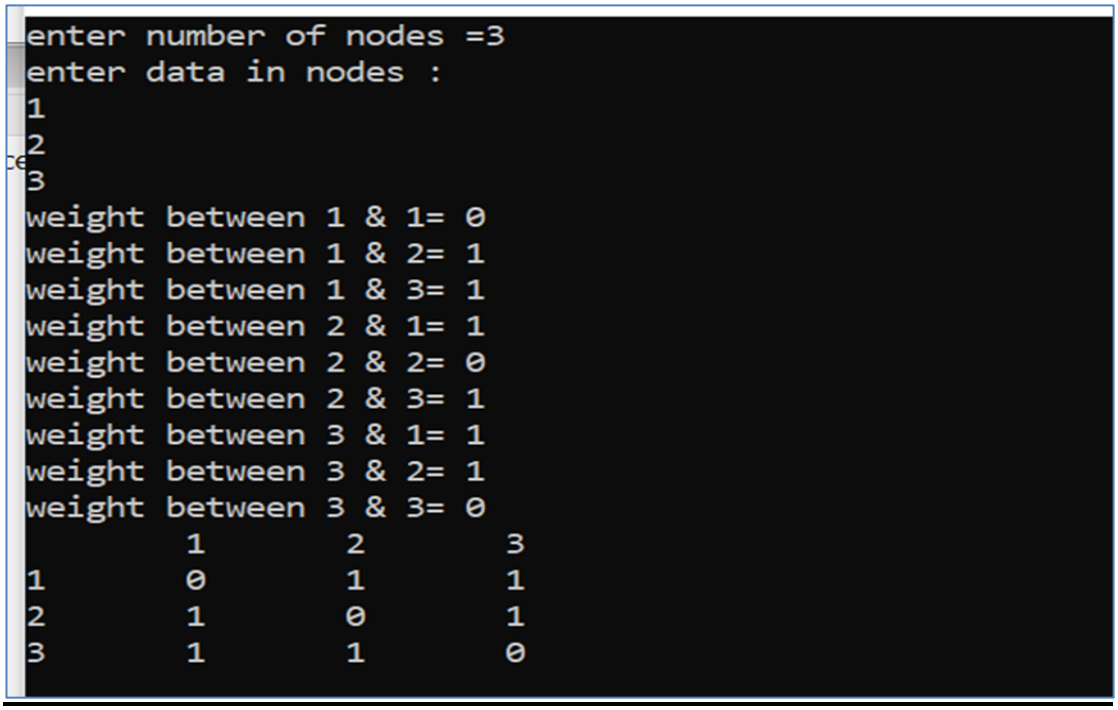
Code:

```
#include <iostream>

using namespace std;

int main()
```

```
{
int n,nn;
cout<<"enter number of nodes =";
cin>>n;
char arr[n];
cout<<"enter data in nodes :"<<endl;
for(int i=0;i<n;i++)
{
     cin>>arr[i];
}
nn=n;
int m[n][nn];
for(int i=0;i<n;i++)
{
     for(int j=0;j<nn;j++)
     {
          cout<<"weight between "<<arr[i]<<" & "<<arr[j]<<"= ";
          cin>>m[i][j];
     }
}
cout<<"\t";
for(int i=0;i<n;i++)
{
     cout<<arr[i]<<"\t";//for clear view purpose
}
cout<<endl;
for(int i=0;i<n;i++)//printing the matrix
{
     cout<<arr[i]<<"\t";
     for(int j=0;j<nn;j++)
     {
          cout<<m[i][j]<<"\t";
     }
     cout<<endl;
}
```

```
}
```

Output:



Task#02;

Statement: Code in C++ for Adjacency List.

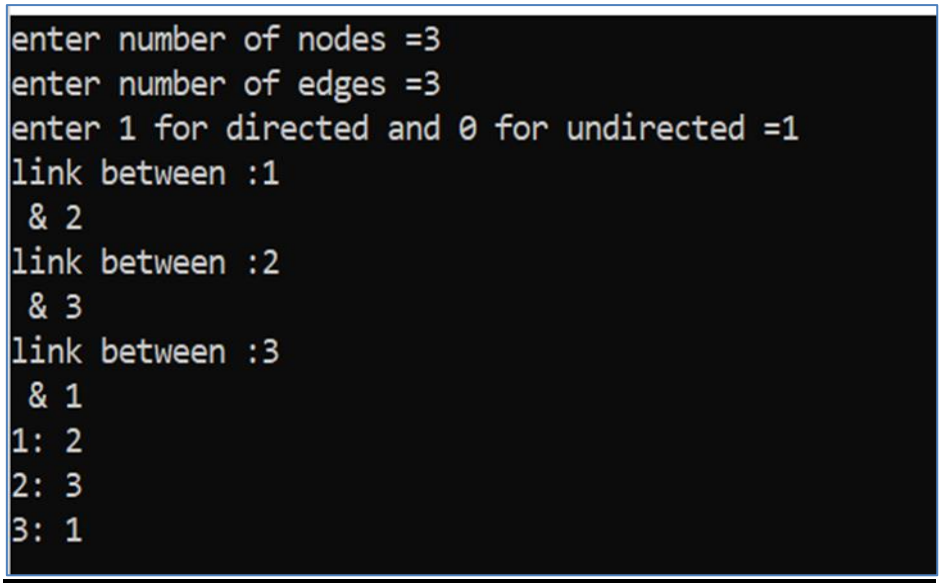
Code:

```
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

// This class represents a directed graph using adjacency list
representation
class Graph {
  int V; // Number of vertices
  unordered_map<int, vector<int>> adj; // Adjacency list

 public:
  Graph(int V) {
    this->V = V;
  }
```

```
  void addEdge(int v, int w) {
    adj[v].push_back(w); // Add w to v's list
  }

  void printGraph() {
    // Iterate through all the vertices
    for (int v = 1; v <=V; v++) {
      cout << v << ": ";

      // Iterate through all adjacent vertices of v
      for (int i : adj[v]) {
        cout << i << " ";
      }

      cout << endl;
    }
  }
};

int main() {
    int n,m,c;
    cout<<"enter number of nodes =";
    cin>>n;
  Graph g(n);
  cout<<"enter number of edges =";
  cin>>m;
  cout<<"enter 1 for directed and 0 for undirected =";
  cin>>c;
  if(c==0)
    m=2*n;
  for(int i=0;i<m;i++)
     {
          int u,v;
          cout<<"link between :";
          cin>>u;
```

```
            cout<<" & ";
            cin>>v;
            //cin>>u>>v;
            //creating an undirected graph
            g.addEdge(u,v);
      }
g.printGraph();

  return 0;
}
```

Output:



Conclusion:

- Graph is a non linear data structure
- It is the combination of nodes and edges
- It can be directed and undirected
- The edges can be weighted and unweighted
- Graph are very useful in making networks, maps and social media platforms
- Graphs can be represented by two methods ,Adjacency matrix and Adjacency list
- STL(standard template library) is very handful library of containers and algorithms