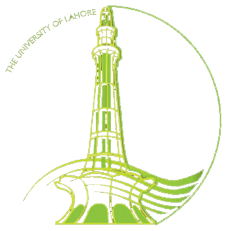


THE UNIVERSITY
OF LAHORE
ISLAMABAD
CAMPUS

Data Structures & Algorithms
(CS09203)

Lab Report

Name:	Muhammad Umer
Registration #:	CSU-F16-104
Lab Report #:	08
Dated:	27-05-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 1
Create a C++ program to implement the weighted and directed graph and add edges in the adjacency list and display added edges.

Objective
To understand and implement the weighted and directed graph with basic edges insertion in the adjacency list.

Software Tool
1. Sublime Text Editor
2. Dev C++
3. Window 7 (32 Bit)

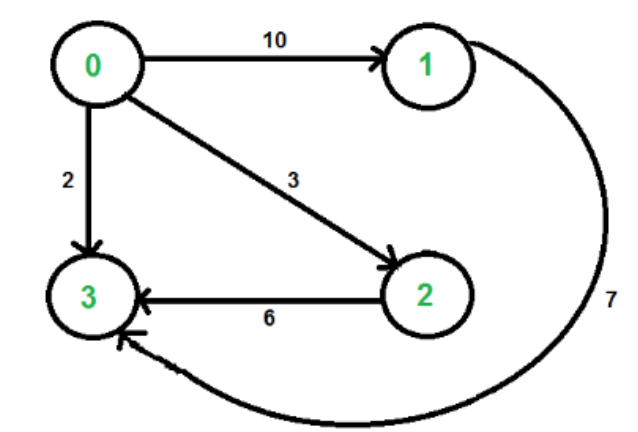
1 Theory

Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.
2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook. For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale.

Following is an example of weighted and directed graph with 4 vertices.



2 Task

2.1 Procedure: Task 1 Weighted Undirected Graph

2.1.1 Procedure: Task 1.1 Inserting Edges

The following code is the representation of the above graph using STL

```
void addEdge(vector <pair<int, int> > adj[], int u, int v, int wt)
{
        adj[u].push_back(make_pair(v, wt));
        adj[v].push_back(make_pair(u, wt));
}
```

2.1.2 Procedure: Task 1.2 Displaying the Edges

```
void printGraph(vector<pair<int,int> > adj[], int V)
{
        int v, w;
        for (int u = 0; u < V; u++)
        {
                cout << "Node " << u << " makes an edge with \n";
                for (auto it = adj[u].begin(); it!=adj[u].end(); it++)
```

```
                {
                        v = it->first;
                        w = it->second;
                        cout << "\tNode " << v << " with edge weight ="
                                << w << "\n";
                }
                cout << "\n";
        }
}
```

2.1.3 Procedure: Task 1.3 Main Function

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
        int V = 4;
        vector<pair<int, int> > adj[V];
        addEdge(adj, 0, 1, 10);
        addEdge(adj, 0, 3, 2);
        addEdge(adj, 0, 2, 3);
        addEdge(adj, 1, 3, 7);
        addEdge(adj, 2, 3, 6);
        printGraph(adj, V);
        return 0;
}
```

Output:

Consider the Figure 1 (in the end of this document) for the output of the above graph.

Note: My IDE (Dev C++) doesn't support C++11, as the above code is written in C++11, I run this code in the online IDE "https://ide.geeksforgeeks.org/index.php", which give me the output showed in Figure 1

2.2 Procedure: Task 2 Weighted directed Graph

For the Weighted Directed graph we just only remove the line that is use for bidirectional (undirected) graph in the addEdge();.
The following function is the representation of the above weighted and directed graph using STL.

Note: The follow funtion can be use instead of addEdge(); in the weighted undirected graph.

```
void addEdge(vector <pair<int, int> > adj[], int u, int v, int wt)
{
        adj[u].push_back(make_pair(v, wt));
}
```

Output:

Consider the Figure 2 (in the end of this document) for the output of the above graph.

Source Code
https://goo.gl/ccBvqK

```
Node 0 makes an edge with
        Node 1 with edge weight =10
        Node 3 with edge weight =2
        Node 2 with edge weight =3

Node 1 makes an edge with
        Node 0 with edge weight =10
        Node 3 with edge weight =7

Node 2 makes an edge with
        Node 0 with edge weight =3
        Node 3 with edge weight =6

Node 3 makes an edge with
        Node 0 with edge weight =2
        Node 1 with edge weight =7
        Node 2 with edge weight =6
```

Figure 1: Adjacency list representation of the weighted undirected graph

3 Conclusion

Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook. For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale.

(Concerned Teacher/Lab Engineer)

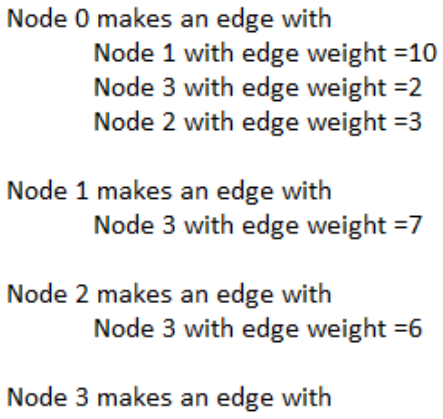


Figure 2: Adjacency list representation of the weighted directed graph