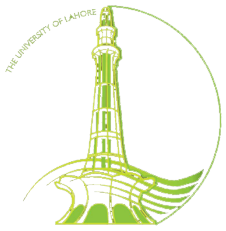


THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Data Structure and Algorithms

Lab Report

Name:	Muhammad Arslan Ishaq
Registration #:	CSU-F16-112
Lab Report #:	07
Dated:	21-04-2018
Submitted To:	Mr. Usman Ahmed

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

Experiment # 7
Data Structure - Graph

Objective
The objectives of this lab session are to understand the basic of Graphs in Data Structure.

1 Theory

A set of items connected by edges. Each item is called a vertex or node. Formally, a graph is a set of vertices and a binary relation between vertices, adjacency.

Graph
Graph is a non linear data structure, it contains a set of points known as nodes (or vertices) and set of linkes known as edges (or Arcs) which connets the vertices. A graph is defined as follows... Graph is a collection of vertices and arcs which connects vertices in the graph Graph is a collection of nodes and edges which connects nodes in the graph Generally, a graph G is represented as G = (V , E), where V is set of vertices and E is set of edges.

Example
The following is a graph with 5 vertices and 6 edges. This graph G can be defined as G = (V , E) Where V = A,B,C,D,E and E = (A,B),(A,C)(A,D),(B,D),(C,D),(B,E),(E,D).

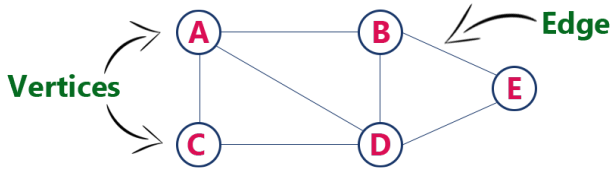


Figure : 1 Vertices and Edge

Graph Terminology:-
We use the following terms in graph data structure...

Vertex: -
A individual data element of a graph is called as Vertex. Vertex is also known as node. In above example graph, A, B, C, D and E are known as vertices

Edge: -
An edge is a connecting link between two vertices. Edge is also known as Arc. An edge is represented as (startingVertex, endingVertex). For example, in above graph, the link between vertices A and B is represented as (A,B). In above example graph, there are 7 edges (i.e., (A,B), (A,C), (A,D), (B,D), (B,E), (C,D), (D,E)).

Origin:-
If an edge is directed, its first endpoint is said to be origin of it.

Destination:-
If an edge is directed, its first endpoint is said to be origin of it and the other endpoint is said to be the destination of the edge.

Undirected Graph:-
A graph with only undirected edges is said to be undirected graph.

Directed Graph:-
A graph with only directed edges is said to be directed graph.

Parallel edges or Multiple edges:-
If there are two undirected edges to have the same end vertices, and for two directed edges to have the same origin and the same destination. Such edges

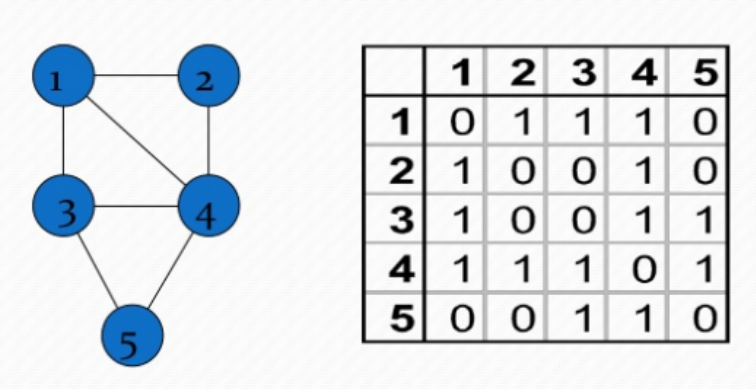
are called parallel edges or multiple edges.

Adjacent:-
If there is an edge between vertices A and B then both A and B are said to be adjacent. In other words, Two vertices A and B are said to be adjacent if there is an edge whose end vertices are A and B.

Path:-
A path is a sequence of alternating vertices and edges that starts at a vertex and ends at a vertex such that each edge is incident to its predecessor and successor vertex.

Tress and Graphs in Data Structures:-
A tree data structure, like a graph, is a collection of nodes. There is a root node. The node can then have children nodes. The children nodes can have their own children nodes called grandchildren nodes Graphs evolved from the field of mathematics. They are primarily used to describe a model that shows the route from one location to another location.

Adjacent Nodes:-
In this representation, graph can be represented using a matrix of size total number of vertices by total number of vertices. . In this matrix, rows and columns both represents vertices. This matrix is filled with either 1 or 0. Here, 1 represents there is an edge from row vertex to column vertex and 0 represents there is no edge from row vertex to column vertex.



	1	2	3	4	5
1	0	1	1	1	0
2	1	0	0	1	0
3	1	0	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

Figure : 1 Output Adjacent Nodes

2 Lab Task

Write a C++ program to implement all the above described algorithms and display the following menu and ask the user for the desired operation.
The program should have the option for reusing it after you have completed the desired task.

2.1 Program

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

using namespace std;


void addEdge(vector<int> adj[], int u, int v )
{
        adj[u].push_back(v);
        adj[v].push_back(u);

}
 void printGraph(vector<int> adj[], int V)
 {
```

```
        for (int v=0; v<V; ++v)
        {
                cout<<"\n Adjacency List of vertex"<<v<<"\n";
                for (auto x: adj [v])
                    cout<<" "<< x;
                printf("\n");
                        }
}

int main()
{
int v=5;
        cout<<"Arslan";
        vector<int> adj[v];
        addEdge(adj, 0, 1);
        addEdge(adj, 0, 4);
        addEdge(adj, 1, 2);
        addEdge(adj, 1, 4);
        addEdge(adj, 2, 3);
        addEdge(adj, 3, 4);
        printGraph(adj, v);
}
```

3 Conclusion

In this Lab we have learn how to implement Data Structures Graph using programing in C++. Saving and traversing using graphs is very help full in storing and accessing data. The Speed of data accessing is quite faster and Binary search can be done very effectively using tags in Graphs.