Faculty of Computing



Data Structure and Algorithm

Lab Manual

**Lab Manual Development Team**

|  |  |
| --- | --- |
|  | Supervision and Coordination |

**Ms. Anum Aleem**

Lecturer

Faculty of Computing

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
|  | Lab Designers | | |

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Fareeha Ashraf**  Teaching Fellow  Faculty of Computing | |  | |  | |

ss

# Lab 14: Introduction to Graphs

1. **Activity Time boxing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Activity Name** | **Activity time** | **Total Time** |
| 1 |  |  |  |
| 2 | Lecture | 30 min |  |
| 3 | Walkthrough Tasks | 10 min |  |
| 4 | Practice tasks | 90 min |  |
| 5 | Evaluation | 20 min | 170 |

1. **Concept Map**

* Graphs
* Types of graphs
* Components of graphs
* Adjacency Matrix
* Adjacency List
* BFS
* DFS

# Introduction

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 a directed graph (di-graph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

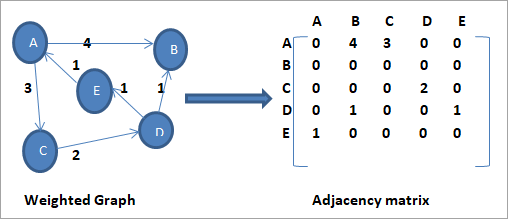
Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like LinkedIn, Facebook. For example, in Facebook, each person is represented with a vertex (or node). Each node is a structure and contains information like person id, name, gender and locale. See [this](http://en.wikipedia.org/wiki/Graph_theory#Applications)for more applications of graph.

Following is an example of an undirected graph with 5 vertices.

https://media.geeksforgeeks.org/wp-content/uploads/undirectedgraph.png

There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation is situation specific. It totally depends on the type of operations to be performed and ease of use.

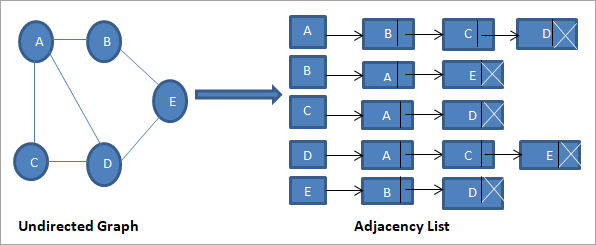
3.1. Adjacency Matrix:  
Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2019/08/6.weighted-graph-and-its-adjacency-matrix.png)

## 4.2. Adjacency List:

We use the adjacency list for the linked representation of the graph. The adjacency list representation maintains each node of the graph and a link to the nodes that are adjacent to this node. When we traverse all the adjacent nodes, we set the next pointer to null at the end of the list.

**Let us first consider an undirected graph and its adjacency list.**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2019/08/7.undirected-graph-and-its-adjacency-list.png)

As shown above, we have a linked list (adjacency list) for each node. From vertex A, we have edges to vertices B, C and D. Thus these nodes are linked to node A in the corresponding adjacency list.

## 4.3. Basic Operations For Graphs

Following are the basic operations that we can perform on the graph data structure**:**

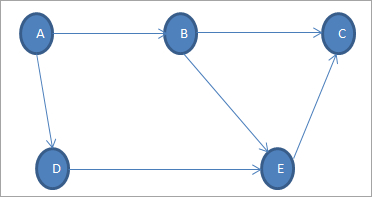
* **Add a vertex:** Adds vertex to the graph.
* **Add an edge:** Adds an edge between the two vertices of a graph.
* **Display the graph vertices:** Display the vertices of a graph.

## 4.5. Types of Graphs – Directed And Undirected Graph

A graph in which the edges do not have directions is called the Undirected graph. The graph shown above is an undirected graph.

A graph in which the edges have directions associated with them is called a Directed graph.

**Given below is an example of a directed graph.**

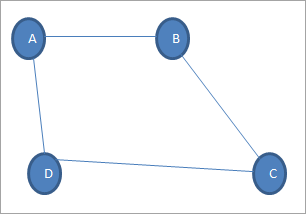
[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2019/08/2.directed-graph-example.png)

In the directed graph shown above, edges form an ordered pair wherein each edge represents a specific path from one vertex to another vertex. The vertex from which the path initiates is called “**Initial Node**” while the vertex into which the path terminates is called the “**Terminal Node**”.

Thus in above graph, the set of vertices is {A, B, C, D, E} and the set of edges is {(A,B),(A,D),(B,C),(B,E),(D,E)(E,C)}.

We will discuss the graph terminology or the common terms used in relation to the graph below.

## 4.6. Graph Terminology

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2019/08/3.Graph-Terminology.png)

1. **Vertex:** Each node of the graph is called a vertex. In the above graph, A, B, C, and D are the vertices of the graph.
2. **Edge:** The link or path between two vertices is called an edge. It connects two or more vertices. The different edges in the above graph are AB, BC, AD, and DC.
3. **Adjacent node:** In a graph, if two nodes are connected by an edge then they are called adjacent nodes or neighbours. In the above graph, vertices A and B are connected by edge AB. Thus A and B are adjacent nodes.
4. **Degree of the node:** The number of edges that are connected to a particular node is called the degree of the node. In the above graph, node A has a degree 2.
5. **Path:** The sequence of nodes that we need to follow when we have to travel from one vertex to another in a graph is called the path. In our example graph, if we need to go from node A to C, then the path would be A->B->C.
6. **Closed path:** If the initial node is the same as a terminal node, then that path is termed as the closed path.
7. **Cycle:** A path in which there are no repeated edges or vertices and the first and last vertices are the same is called a cycle. In the above graph, A->B->C->D->A is a cycle.
8. **Connected Graph:** A connected graph is the one in which there is a path between each of the vertices. This means that there is not a single vertex which is isolated or without a connecting edge. The graph shown above is a connected graph.
9. **Weighted graph:** A positive value assigned to each edge indicating its length (distance between the vertices connected by an edge) is called weight. The graph containing weighted edges is called a weighted graph. The weight of an edge e is denoted by **w(e)** and it indicates the cost of traversing an edge.
10. **Diagraph:** A digraph is a graph in which every edge is associated with a specific direction and the traversal can be done in specified direction only.

## 4.7. Applications Of Graphs

* Let us discuss some of the applications of graphs.
* Graphs are used extensively in computer science to depict network graphs, or semantic graphs or even to depict the flow of computation.
* Graphs are widely used in Compilers to depict allocation of resources to processes or to indicate data flow analysis, etc.
* Graphs are also used for query optimization in database languages in some specialized compilers.
* In social networking sites, graphs are main the structures to depict the network of people.
* Graphs are extensively used to build the transportation system especially the road network. A popular example is Google maps that extensively uses graphs to indicate directions all over the world.

### **Breadth First Search in Graph:**

for a graph is similar to Breadth First Traversal of a tree .The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex. For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don’t mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, 1.

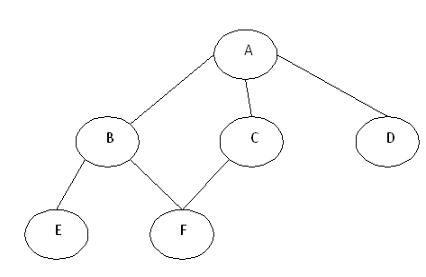
[](https://media.geeksforgeeks.org/wp-content/uploads/bfs-5.png)

**Algorithmic Steps**

1. **Step 1**: Push the root node in the Queue.
2. **Step 2**: Loop until the queue is empty.
3. **Step 3**:Remove the node from the Queue.
4. **Step 4**: If the removed node has unvisited child nodes, mark them as visited and insert the unvisited children in the queue.

### **Depth First Search (DFS)**

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search. Let’s see how depth first search works with respect to the following graph:



As stated before, in DFS, nodes are visited by going through the depth of the tree from the starting node. If we do the depth first traversal of the above graph and print the visited node, it will be “A B E F C D”. DFS visits the root node and then its children nodes until it reaches the end node, i.e. E and F nodes, then moves up to the parent nodes.

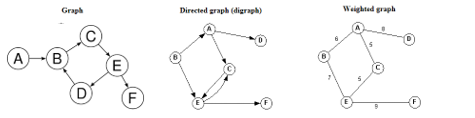
#### **Algorithmic Steps**

1. **Step 1**: Push the root node in the Stack.
2. **Step 2**: Loop until stack is empty.
3. **Step 3**: Peek the node of the stack.
4. **Step 4**: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack.
5. **Step 5**: If the node does not have any unvisited child nodes, pop the node from the stack.

# Practice Tasks

This section will provide practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, put these tasks into your lab designated Folder announced by lab instructor.

### **Lab Task 1:**



* + Make Adjacency Matrix of above graphs.
  + Make Adjacency list of above graphs.
  + Perform BFS.
  + Perform DFS.

**Lab Task 2:**

Draw a graph modes, starting the type of graph used, to represent airline routs where every day there are four flights from Boston to Newark, two flights from Newark to Boston, three flights from Newark to Miami, two flights from Miami to Newark, one flight from Newark to Detroit, two flights from Detroit to Newark, three flights from Newark to Washington, two flights from Washington to Newark, and one flight from Washington to Miami, with

(1) an edge between vertices representing cities that have a flight between them (in either direction)

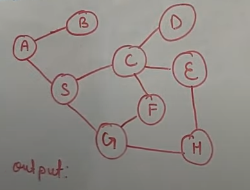
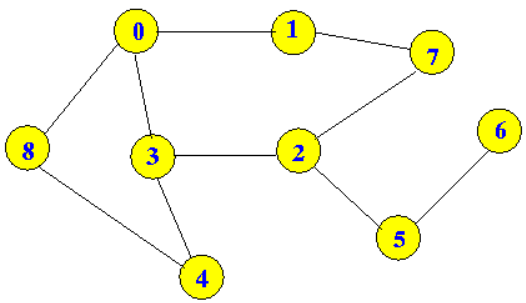
(2) an edge between vertices representing cities for each flight that operates between them) in either direction)

(3) an edge between vertices representing cities for each flight that operates between them (in either direction) plus a loop for a special sightseeing trip that takes off and lands in Miami

**Lab Task 3**

Describe a graph model that represents whether each person at a class knows the name of each other person at the class. Should the edges be directed or undirected? Should multiple edges be allowed? Should loops be allowed?

**Lab Task 4:**



* + Perform BFS.
  + Perform DFS.

**Evaluation criteria**

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Task No** | **Description** | **Marks** |
| 1 | Task 1 | Lab Task | 5 |
| 2 | Task 2 | Lab Task | 5 |
| 3 | Task 3 | Lab Task | 5 |
|  | Task 4 | Lab Task | 5 |

**Further Reading**

**Books**

Uploaded on moellim