##### **Experiment No. 06**

**Aim:** To illustrate the concept of graph.

**Problem Statement :**

Represent a given graph using adjacency matrix/list to perform DFS and using adjacency list to perform BFS. Use the map of the area around the college as the graph. Identify the prominent land marks as nodes and perform DFS and BFS on that.

Write a function to get the number of vertices in an undirected graph and its edges. You may assume that no edge is input twice.

1. Use adjacency list representation of the graph and find runtime of the function
2. Use adjacency matrix representation of the graph and find runtime of the function

**Learning Objectives:**

* To understand directed and undirected graph.
* To implement program to represent graph using adjacency matrix and list.

**Learning Outcome:**

* Student able to implement program for graph representation.
* Apply and analyze non linear data structures to solve real world complex problems

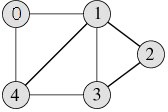
**Theory:**

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. See this for more applications of graph.

Following is an example undirected graph with 5 vertices.



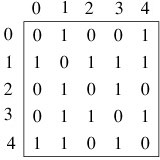
Following two are the most commonly used representations of graph.

1. Adjacency Matrix
2. Adjacency List

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.

##### **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.

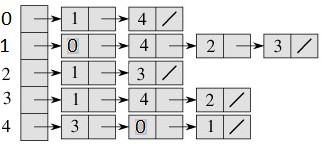
The adjacency matrix for the above example graph is: 

**Pros**: Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O(1).

**Cons**: Consumes more space O(V^2). Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is O(V^2) time.

##### **Adjacency List:**

An array of linked lists is used. Size of the array is equal to number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ith vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists. Following is adjacency list representation of the above graph.



**Pros**: Saves space O(|V|+|E|) . In the worst case, there can be C(V, 2) number of edges in a graph thus consuming O(V^2) space. Adding a vertex is easier.

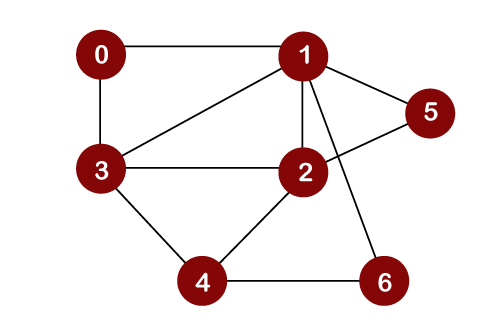
**Cons**: Queries like whether there is an edge from vertex u to vertex v are not efficient and can be done O(V).

Graph Traversal methods:

1. BFS
2. DFS

[**BFS:**](https://www.javatpoint.com/breadth-first-search-algorithm) **stands for *Breadth First Search***. It is also known as **level order traversal**. The Queue data structure is used for the Breadth First Search traversal. When we use the BFS algorithm for the traversal in a graph, we can consider any node as a root node.

Let's consider the below graph for the breadth first search traversal.

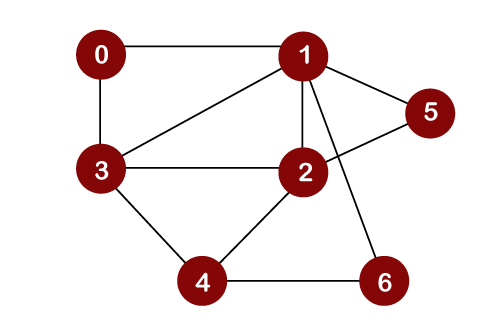


Suppose we consider node 0 as a root node. Therefore, the traversing would be started from node 0.

**So BFS Traversal Will be 0, 1, 3, 2, 5, 6**

**DFS: Stands for Depth First Search.** In DFS traversal, the stack data structure is used, which works on the LIFO (Last In First Out) principle. In DFS, traversing can be started from any node, or we can say that any node can be considered as a root node until the root node is not mentioned in the problem.

In the case of BFS, the element which is deleted from the Queue, the adjacent nodes of the deleted node are added to the Queue. In contrast, in DFS, the element which is removed from the stack, then only one adjacent node of a deleted node is added in the stack.



Suppose we consider node 0 as a root node. Therefore, the traversing would be started from node 0.

**So BFS Traversal Will be 6, 4, 2, 5, 3, 1, 0**

**Conclusion:** Student implemented program for graph presentation in adjacency matrix and list.

**Questions**

1. An undirected graph having n edges, then find out no. Of vertices that graph have?
2. Define data structure to represent graph.
3. What are the methods to display graph.
4. Where you apply directed and undirected graph?
5. What is complexity of your graph to represent it in adjacency matrix and list?