Name : Sanchit Kripalani
Batch : M1
Roll no : 31145

Date of Performing : 10/1/22
Date of submission : 21/1/22

Page No. 1
Date

Group A [Artificial Intelligence]
Assignment 1

- **Problem statement :**

  Implement depth first search and breadth first search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

- **objectives :**

  1. Create a custom data structure Graph, which will be used to store the vertices of graph, as well as edges of the graph.
  2. Use adjancency list representation of graphs.
  3. Implement DFS and BFS traversal algorithms.
  4. Implement a menu-driven, user-friendly code.

- **Software and Hardware Requirements :**

  Windows 10 (64 bit, 8 GB Ram), vscode, GCC compiler.

- **Theory :**

  Graph -

  A graph is a non-linear data structure consisting of nodes and edges. Nodes are also called vertices and edges connected any 2 nodes in the graph.

  Types of Graph -

  1. Directed Graph :

     The Edges of graph have directions. The direction indicates a one-way relationship, meaning the edge can be traversed in a single direction.

  2. Undirected Graph :

     The Edges of graph has no direction. This means that the edge can be traversed in both direction.

Example -

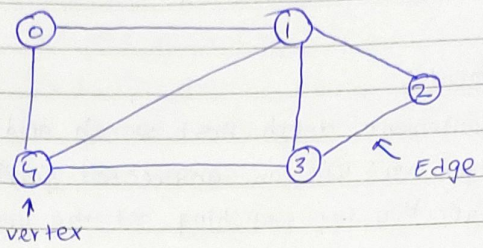


Fig. Undirected Graph

Representation -

Commonly used represention of graphs include Adjancy Matrix, Adjacency list.

Adjancency Matrix -

This is a 2D matrix of size V x V where V is no. of vertices in a graph. Matrix[i][j] represent whether there is a graph between vertex i and j

Adjacency List -

This is a linked list representation of a graph An entry array[i] represents the list of vertices adjacent to the ith vertex.

Adjacency list example -

(For graph shown above)

0 → 1 → 4

1 → 0 → 2 → 3 → 4

2 → 1 → 3

3 → 1 → 2 → 4

4 → 0 → 1 → 3

Traversal Algorithms -

These Algorithms are used to visit vertices of graphs through edge connections. There are 2 types of traversal algorithms.

(1) Breadth First Search (BFS) -

This algo is used to visit all of the nodes of a given graph. In this algorithm, one node is selected and then all of the adjacent nodes are visited one by one. After completing all of the adjacent vertices, it moves further to check another vertex and it adjacent vertex again.

Algorithm - [Use Queue Data structure]

1. Set status of all nodes as not visited
2. Take input for which vertex should traversal start with.
3. Set status of this vertex as visited
4. Push this vertex in the queue.
5. while queue is not empty
   - 5a. Print the front element of queue and then remove it
   - 5b. Now move on to vertices adjacent to current vertex.
   - 5c. If an adjacent vertex is not visited, then add it to queue.
   - 5d. Then set the status of the current vertex as visited.
6. End Algorithm.

(2) Depth First Search (DFS) -

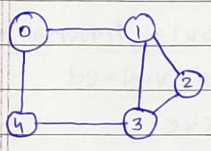
This algo works by using a starting given vertex. when an adjacent vertex is found, we move to the adjacent vertex first and then try to traverse in the same manner.

We use a stack/recursive approach to achieve this task.

Algorithm - [Recursive]

1. Take starting node is input.
2. Set the status of current verter as visited.
3. Print the current vertex
4. For all the elements in adjancency list corresponding to the current vertex
   4a. IF is the vertex is not visited, recursively call DFS function with this vertex.
5. End algorithm

- Test Case :

Input Graph	Expected O/P	Actual O/P	Result
	BFS : (1st vertex=0) 0 1 2 3 4	BFS : 0 1 2 3 4	Pass
	DFS : 0 1 2 3 4	DFS : 0 1 2 3 4	

- Conclusion :

Thus we have successfully created the Graph data structure and implemented Depth First and Breadth First Traversal Algorithms for graph.