Exercise 6 Graph Traversals

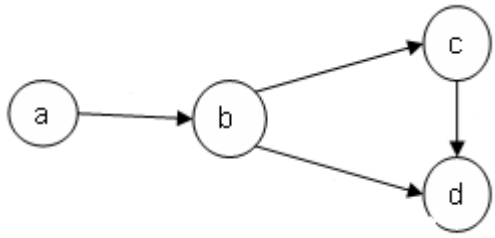
6. Write a C++ Program for implementation of graph using following Traversals on it
a) BFS (Breadth-first Search) b) DFS (Depth-first Search)

Objective: The objective of this exercise is to enable you to perform traversing graph using BFS and DFS algorithms.

Procedure and description:
A graph data structure consists mainly of a set of entities called nodes (or vertices or points) and a set of edges (or arcs or links) connecting the vertices.

A graph structure G is defined as G = (V, E) where V is a set of vertices is a set of edges, and each edge is formed from pair of distinct vertices in V

Representation of Graph



Definition of Graph:
Class vertices
Start
Int maxnodes
End
Vertices graph vertices [maxnodes] // to store the vertices
Double mat [maxnodes][maxnodes] // to store the edges in the graph

a) BFS (Breadth-first Search):
In graph theory, BFS is a strategy for searching in a graph when search is limited to essentially two operations: (a) visit and inspect a node of a graph; (b) gain access to visit the neighbor nodes of the currently visited node. The BFS begins at a root node and inspect all the neighboring nodes. Then for

each of those neighbor nodes in turn, it inspects their neighbor nodes which were unvisited, and so on.

BFS is a uniformed search method that aims to expand and examine all nodes of a graph or combination of sequences by systematically searching through every solution. In other wors, it exhaustively searches the entire graph or sequence without considering the goal until it finds it.From the standpoint of the algorithm, all child nodes obtained by expanding a node are added to a FIFO (i.e., First In, First Out) queue. In typical implementations, nodes that have not yet been examined for their neighbors are placed in some container (such as a queue or linked list) called "open" and then once examined are placed in the container "closed".

Algorithm:

Step 1: Select a vertex as a starting point (arbitrary vertex) from where to traverse a graph.
Step 2: traverse all the adjacent vertices of arbitrary vertex.
Step 3: Now travers unvisited vertices which are adjacent to the Visited vertex.
Step 4: repeat step 3 until all the vertices in graph are visited.
Step 5: Stop.

b) DFS (Depth-first Search)

DFS is an algorithm for traversing or searching a graph. One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking.

Formally, DFS is an uninformed search that progresses by expanding the first child node of the search graph that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hasn't finished exploring. In a non-recursive implementation, all freshly expanded nodes are added to a stack for exploration.

Algorithm:

Step 1: Select a vertex as a starting point (arbitrary vertex) from where to traverse a graph.
Step 2: select any one of adjacency vertices of arbitrary vertex from adjacency list and traverse it.

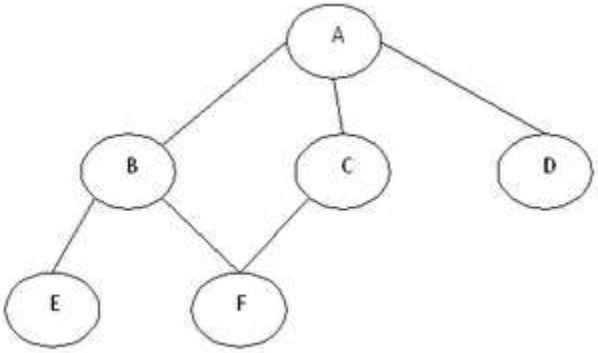
Step 3: now traverse the vertices that are adjacent to the selected vertex.

Step 4: Repeat step 3 until all the vertices in the graph are visited.

Step 5: Stop.

Expected Output:

After execution of the program. Enter input graph vertices. For better understanding see pictorial representation of the input Graph and output.



The Breadth-first search traversal of above graph will be:

A B C D E F

The depth-first search traversal of above graph will be:

A B E F C D