Experiment No-07: Graph Representation and Traversal.

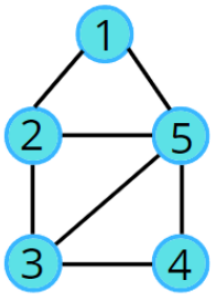
Objectives

- Represent a graph in C++.
- Traverse a graph using the breadth-first search technique.

Example 1: Graph Representation using Adjacency List.

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

/* Inputting Format

4 4      nodes  edges
0 1
0 2
0 3
1 2
*/

int main(){

vector<int>graph[5];    // initialize a vector of array
int nodes, edge, u, v;

cout<<"Enter Number of Nodes: ";
cin>>nodes;
cout<<"Enter Number of Edges: ";
cin>>edge;

for (int i = 0;i<edge;i++){
    cin>>u>>v;           // take input the edges connection
    graph[u].push_back(v);
    graph[v].push_back(u);
}

cout<<"Adjacency List of the Graph: "<<endl;

for (int j = 0; j<nodes; j++){
    cout<<j<<" --> ";
    for(auto it: graph[j]){
        cout<<it<<" ";
    }
    cout<<endl;
}

}
```

Example 2: Breadth First Search (BFS) Traversal.

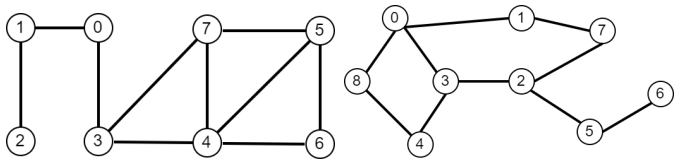


```
#include<bits/stdc++.h>
using namespace std;

vector<int> adj[100];
int visited[100]; // create an array with all zero values

// BFS function
vector<int>Bfs(int source) {
    vector<int>bfs;
    queue<int> q; // declare a empty queue
    visited[source] = 1;
    q.push(source); // push source node into queue

    while (!q.empty()) {
        int node = q.front(); // front element of the queue
        q.pop(); // pop the node
        bfs.push_back(node);
        for (auto it: adj[node]) {
            int nxt_node = it; // the neighbour node
            // if the neighbour has previously not been visited,
            if (visited[nxt_node]) continue;
            visited[nxt_node] = 1;
            q.push(nxt_node); // push into the queue
        }
    }
    return bfs;
}

int main() {

    int i, j, k;
    int n, e;
    vector<int>bfs;
    cout<< "No.of Nodes: ";
    cin >> n;
    cout<< "No.of Edges: ";
    cin >> e;
    cout<<"Enter the edge connections: "<<endl;
```

```
    // adjacency list
    for (i = 0; i < e; ++i) {
        int u, v;         // edge inputs
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int source;
    cout<<"Enter the Source Node: "<<endl;
    cin >> source;
    // call the BFS method
    bfs = Bfs(source);
    // print the values
    for (auto it: bfs){
        cout<<it<<" ";
    }

}
```

Practice Exercise

1. Write a C++ program to Represent the following graphs using an adjacency matrix (Figure 1).
2. Write a C++ program to Represent the following graphs using an adjacency List (Figure 1).
3. Write a C++ program to find the traversal of the following graphs (Figure 1). [Choose a random node as a source]



(a) (b)

Figure 1

Resources (Link)

Try to solve similar problems at an online Judge.

1. Graph Representation
2. BFS Traversal