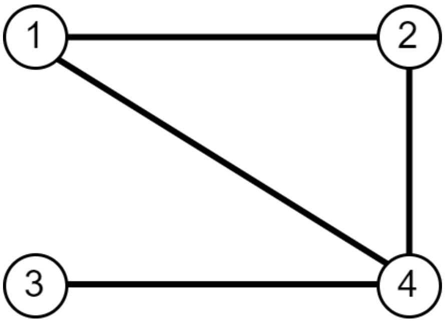
Experiment No-08: Shortest Path Finder using BFS Algorithm.

Objectives

- Find the shortest distances using BFS.
- Find the shortest path from a source to a destination node.



Example 1: Finding the shortest distance from the source to all the nodes.

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

vector<int> adj[100];
int dis[100], visited[100];

// BFS function
void Bfs(int source) {
    queue<int> q; // declare a empty queue
    dis[source] = 0;
    visited[source] = 1;
    q.push(source); // push source node into queue
    while (!q.empty()) {
        int node = q.front(); // front element of the queue
        for (auto it: adj[node]) {
            int nxt_node = it;
            // already visited then skip
            if (visited[nxt_node]) continue;
            dis[nxt_node] = 1 + dis[node];
            visited[nxt_node] = 1;
            q.push(nxt_node); // push into the queue
        }
        // pop the node
        q.pop();
    }
}

int main() {
    int i, j, k;
    int n, m;
    cout<< "No.of Nodes: "<<endl;
    cin >> n;
    cout<< "No.of Edges: "<<endl;
    cin >> m;
    cout<<"Enter the edge connections: "<<endl;
    for (i = 0; i < m; ++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(source);
    for (i = 1; i <= n; ++i) {
        cout << "Distance " << source << " to " << i << " = " << dis[i]
            << endl;
    }
}
```

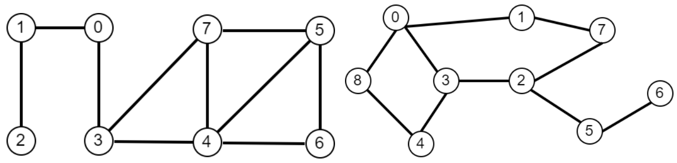
Example 2: Finding the shortest path from a source to the destination node.

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

vector<int> adj[100];
vector<int> path;
int parent[100], dis[100], visited[100];

// Function for finding the shortest path
void shortest_path(int d){
    if (d!=-1){
        int p = parent[d];
        path.push_back(d); // push the paths into a vector
        shortest_path(p); // recursively called
    }
}

// BFS function for finding the shortest distance
void Bfs(int source) {
    queue<int> q; // declare a empty queue
    dis[source] = 0;
    visited[source] = 1;
    parent[source] = -1;
    q.push(source); // push source node into queue
    while (!q.empty()) {
        int node = q.front(); // front element of the queue
        for (auto it: adj[node]) {
            int nxt_node = it;
            // already visited then skip
            if (visited[nxt_node]) continue;
            dis[nxt_node] = 1 + dis[node];
            visited[nxt_node] = 1;
            parent[nxt_node] = node;
            q.push(nxt_node); // push into the queue
        }
        // pop the node
        q.pop();
    }
}

int main() {
    int i, j, k;
    int n, m;
    cout<< "No.of Nodes: "<<endl;
    cin >> n;
    cout<< "No.of Edges: "<<endl;
    cin >> m;
    cout<<"Enter the edge connections: "<<endl;

    for (i = 0; i < m; ++i) {
        int u, v; // edge inputs
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    int source,dest;
    cout<<"Enter the Source Node: "<<endl;
    cin >> source;
    cout<<"Enter the Destination Node:"<<endl;
    cin>> dest;
    // call the BFS method
    Bfs(source);
    cout<<"Shortest Distance from "<<source<<" to "<<dest<<" =
        "<<dis[dest]<<endl;
    cout<<"Shortest Path is: ";
    shortest_path(dest); // call the shortest path function
    // Reverse the path vector
    reverse(path.begin(), path.end());
    // print the path
    for (auto it: path){
        cout<<it<<" ";
    }
}
```

Practice Exercise

Write C++ programs for the following graphs to -

1. Find the shortest distance from an arbitrary source to all the nodes.
2. Find the shortest distance and path from an arbitrary source to an arbitrary destination node.



(a) (b)

Figure 1