Graph Algorithms

5A. Breadth First Search Implement BFS and show the adjacency matrix of the spanning tree.

Algorithm:

Step 1: Create a queue Q to store the vertices.
Step 2: Push the source vertex S in the queue Q.
Step 3: Mark **S** as visited.
Step 4: While the queue Q is not empty
Step 5: Remove vertex U from the front of the queue.
Step 6: For every vertex V adjacent to the vertex U**,** If the vertex V is not visited Then Explore the vertex V and mark V as visited. Push the vertex V in the queue Q.

Program: [In next page]

```cpp
// Exp5A: Breadth First Search
// Author: Pranjal Timsina
#include <iostream>
#include <vector>

using namespace std;

class Graph {

    int v, e; // vertices, edges
    int** adj; // Adjacency matrix

public:
    Graph(int v, int e); // constructor

    void addEdge(int start, int e); // insert new edge

    void BFS(int start); // BFS traversal
    void printMatrix() {
        for (int row = 0; row < v; row++) {
            cout << "\n";
            for (int c= 0;c< v;c++) {
                cout << adj[row][c] << " ";
            }
        }
    }
};
Graph::Graph(int v, int e) {
    // fill adjacency matrix
    this->v = v;
    this->e = e;
    adj = new int*[v];
    for (int row = 0; row < v; row++) {
        adj[row] = new int[v];
        for (int column = 0; column < v; column++) {
            adj[row][column] = 0;
        }
    }
}
```

```
void Graph::addEdge(int start, int e) {
    // add an edge to the graph
    adj[start][e] = 1;
    adj[e][start] = 1;
}

void Graph::BFS(int start) {
    vector<bool> visited(v, false);
    vector<int> q;
    q.push_back(start);
    visited[start] = true;

    int vis;
    while (!q.empty()) {
        vis = q[0];

        // Print the current node
        cout << vis << " ";
        q.erase(q.begin());

        // For every adjacent vertex to the current vertex
        for (int i = 0; i < v; i++) {
            if (adj[vis][i] == 1 && (!visited[i])) {

                // Push the adjacent node to the queue
                q.push_back(i);
                visited[i] = true;
            }
        }
    }
}
```

```
int main() {
    int v = 5, e = 4;
    cout << "Enter vertices: ";
    cin >> v;
    cout <<  "Enter edges: ";
    cin >> e;

    Graph G(v, e);
    for (int i = 0; i < e; i++) {
        int a, b;
        cout << "Enter pair (0 1): ";
        cin >> a >> b;
        G.addEdge(a, b);

    }
    G.BFS(0);
    G.printMatrix();
}
```

Output:

```
(base) PS D:\College\Data Structures and Algorithms\Lab\DSA-Assignments\Assignment 5\src> g++ .\BFS2.cpp
(base) PS D:\College\Data Structures and Algorithms\Lab\DSA-Assignments\Assignment 5\src> .\a.exe
Enter vertices: 5
Enter edges: 4
Enter pair (0 1): 0 1
Enter pair (0 1): 1 2
Enter pair (0 1): 1 3
Enter pair (0 1): 3 4
0 1 2 3 4
0 1 0 0 0
1 0 1 1 0
0 1 0 0 0
0 1 0 0 1
0 0 0 1 0
```

Results:

Thus, the breadth first search algorithm has been implemented