```
#include <stdio.h>
#include <stdlib.h>

#define MAX_VERTICES 100

int graph[MAX_VERTICES][MAX_VERTICES];
int visited[MAX_VERTICES];
int queue[MAX_VERTICES];
int front = -1, rear = -1;

void initializeGraph(int vertices) {
    for (int i = 0; i < vertices; i++) {
        visited[i] = 0;
        for (int j = 0; j < vertices; j++) {
            graph[i][j] = 0;
        }
    }
}

void addEdge(int start, int end) {
    graph[start][end] = 1;
    graph[end][start] = 1;
}

void dfs(int vertex, int vertices) {
    printf("%d ", vertex);
    visited[vertex] = 1;

    for (int i = 0; i < vertices; i++) {
        if (graph[vertex][i] == 1 && !visited[i]) {
            dfs(i, vertices);
        }
    }
}

void bfs(int start, int vertices) {
    printf("%d ", start);
    visited[start] = 1;
    enqueue(start);
    while (!isEmpty()) {
        int current = dequeue();
        for (int i = 0; i < vertices; i++) {
            if (graph[current][i] == 1 && !visited[i]) {
                printf("%d ", i);
                visited[i] = 1;
                enqueue(i);
            }
        }
    }
}

void enqueue(int vertex) {
    if (rear == MAX_VERTICES - 1) {
        printf("Queue is full\n");
    } else {
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = vertex;
    }
}

int dequeue() {
    int vertex;
    if (front == -1 || front > rear) {
        printf("Queue is empty\n");
        return -1;
    } else {
        vertex = queue[front];
        front++;
        return vertex;
    }
}

int isEmpty() {
    return front == -1 || front > rear;
}

int main() {
    int vertices, edges;
    printf("Enter the number of vertices and edges: ");
    scanf("%d %d", &vertices, &edges);

    initializeGraph(vertices);

    printf("Enter the edges (format: start end):\n");
    for (int i = 0; i < edges; i++) {
        int start, end;
        scanf("%d %d", &start, &end);
        addEdge(start, end);
    }

    printf("DFS traversal: ");
    dfs(0, vertices);

    initializeGraph(vertices);  // Reset visited array

    printf("\nBFS traversal: ");
    bfs(0, vertices);

    return 0;
}
```

Output

```
Enter the number of vertices and edges: 5 6
Enter the edges (format: start end):
0 1
0 2
1 3
1 4
2 4
3 4

DFS traversal: 0 1 3 4 2
BFS traversal: 0 1 2 3 4
```