21. Graph Traversal using Breadth First Search (BFS)

Aim:

To traverse a graph using BFS.

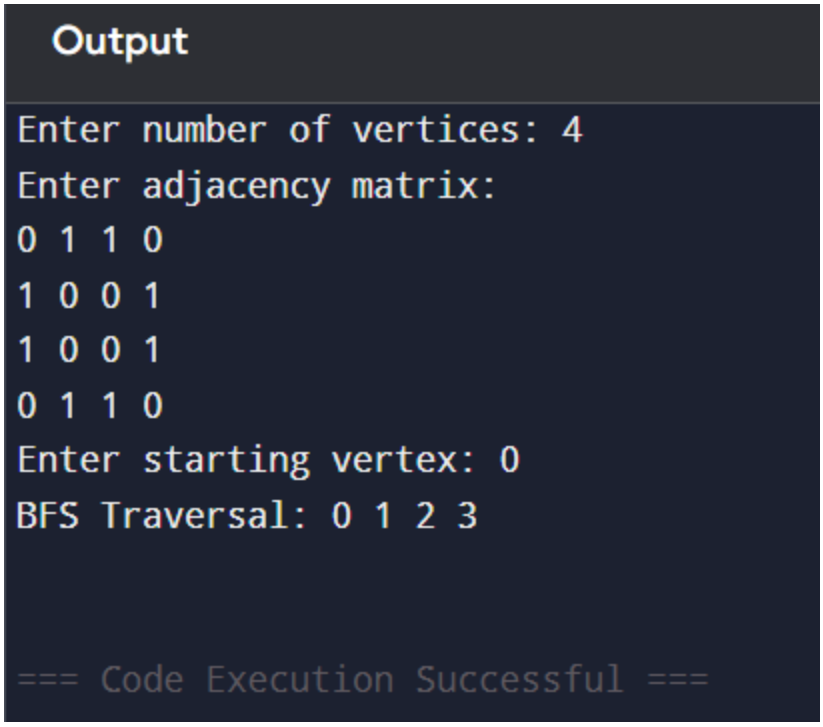
Algorithm:

1. Use a queue to explore nodes level by level.
2. Start from the source vertex, mark it visited, and enqueue it.
3. While queue is not empty:
    - Dequeue a vertex.
    - Visit all unvisited adjacent vertices, mark them visited, and enqueue them.

CODE:

```
#include <stdio.h>
#define MAX 20

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

void enqueue(int v) {
   if (rear == MAX - 1) return;
   if (front == -1) front = 0;
   queue[++rear] = v;
}

int dequeue() {
   if (front == -1 || front > rear) return -1;
   return queue[front++];
}

void BFS(int adj[MAX][MAX], int n, int start) {
   for (int i = 0; i < n; i++) visited[i] = 0;
   enqueue(start);
```

```
    visited[start] = 1;

    printf("BFS Traversal: ");
    while (front <= rear) {
        int v = dequeue();
        printf("%d ", v);
        for (int i = 0; i < n; i++) {
            if (adj[v][i] && !visited[i]) {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
    printf("\n");
}

int main() {
    int n, adj[MAX][MAX], start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &adj[i][j]);
    printf("Enter starting vertex: ");
    scanf("%d", &start);

    BFS(adj, n, start);
    return 0;
}
```



RESULT:

The program successfully executed and displayed the graph traversal using bfs.