British museum search

```
#include <stdio.h>

#define MAX_VERTICES 7

int graph[MAX_VERTICES][MAX_VERTICES];
int visited[MAX_VERTICES];
int path[MAX_VERTICES];
int pathIndex = 0;

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

void addEdge(int v, int w) {
    graph[v][w] = 1;  // Directed edge
}

void printPath(int start, int end) {
    printf("Path: ");
    for (int i = start; i <= end; ++i) {
        printf("%d ", path[i]);
    }
    printf("\n");
}

Void bms(int current, int goal, int numVertices) {
    visited[current] = 1;
    path[pathIndex++] = current;

    if (current == goal) {
        // Print the current path
        printPath(0, pathIndex - 1);
    } else {
        for (int i = 0; i < numVertices; ++i) {
            if (graph[current][i] && !visited[i]) {
                bms(i, goal, numVertices);
            }
        }
    }
    visited[current] = 0;  // Reset visited for backtracking
    pathIndex--;
}

int main() {
    int numVertices = 7;

    initGraph(numVertices);

    // Adding edges for a directed graph
    addEdge(0, 1);
    addEdge(0, 2);
    addEdge(1, 2);
    addEdge(1, 4);
    addEdge(2, 1);
    addEdge(2, 3);
    addEdge(3, 5);
    addEdge(4, 6);

    int source = 0;
    int goal = 6;

    printf("All paths from vertex %d to %d:\n", source, goal);
    bms(source, goal, numVertices);

    return 0;
}
```

Output:

```
All paths from vertex 0 to 6:
Path: 0 1 4 6
Path: 0 2 1 4 6
```

DFS

```
#include <stdio.h>

#define MAX_VERTICES 7

int graph[MAX_VERTICES][MAX_VERTICES];
int visited[MAX_VERTICES];
int path[MAX_VERTICES];
int pathIndex = 0;
```

```
void initGraph(int numVertices) {
    for (int i = 0; i < numVertices; ++i) {
        visited[i] = 0;
        for (int j = 0; j < numVertices; ++j) {
            graph[i][j] = 0;
        }
    }
}

void addEdge(int v, int w) {
    graph[v][w] = 1;  // Directed edge
}

void printPath(int start, int end) {
    printf("Path: ");
    for (int i = start; i <= end; ++i) {
        printf("%d ", path[i]);
    }
    printf("\n");
}

void dfs(int vertex, int goal, int numVertices) {
    visited[vertex] = 1;
    path[pathIndex++] = vertex;

    if (vertex == goal) {
        // Print the current path
        printPath(0, pathIndex - 1);
        return;  // Stop recursion when the goal is reached
    }

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

    visited[vertex] = 0;  // Reset visited for backtracking
    pathIndex--;
}

int main() {
    int numVertices = 7;

    initGraph(numVertices);

    // Adding edges for a directed graph
    addEdge(0, 1);
    addEdge(0, 2);
    addEdge(1, 2);
    addEdge(1, 4);
    addEdge(2, 1);
    addEdge(2, 3);
    addEdge(3, 5);
    addEdge(4, 6);

    int source = 0;
    int goal = 6;

    printf("DFS traversal from vertex %d to %d:\n", source, goal);
    dfs(source, goal, numVertices);

    return 0;
}
```

Output:

```
DFS traversal from vertex 0 to 6:
Path: 0 1 4 6
```

BFS

```
#include <stdio.h>

#define MAX_VERTICES 7
#define MAX_QUEUE_SIZE 100

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

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

void addEdge(int v, int w) {
    graph[v][w] = 1;  // Directed edge
```

```
}

void bfs(int source, int goal, int numVertices) {
    visited[source] = 1;
    queue[++rear] = source;

    int level = 0;  // Track the level

    while (front != rear) {
        int levelSize = rear - front;

        printf("Level %d: ", level++);
        for (int i = 0; i < levelSize; ++i) {
            int current = queue[++front];
            printf("%d ", current);

            if (current == goal) {
                printf("\nBFS traversal from vertex %d to %d:\n", source, goal);
                return;  // Stop BFS when the goal is reached
            }

            for (int j = 0; j < numVertices; ++j) {
                if (graph[current][j] && !visited[j]) {
                    visited[j] = 1;
                    queue[++rear] = j;
                }
                //visited[j] = 0;
            }
        }
        printf("\n");
    }

    printf("\nGoal %d not reachable from source %d\n", goal, source);
}

int main() {
    int numVertices = 7;

    initGraph(numVertices);

    // Adding edges for a directed graph
    addEdge(0, 1);
    addEdge(0, 2);
    addEdge(1, 2);
    addEdge(1, 4);
    addEdge(2, 3);
    addEdge(2, 1);
    addEdge(3, 5);
    addEdge(4, 6);

    int source = 0;
    int goal = 6;

    bfs(source, goal, numVertices);

    return 0;
}
```

Output 👍

```
Level 0: 0
Level 1: 1 2
Level 2: 4 3
Level 3: 6
BFS traversal from vertex 0 to 6:
```