Batch: A1 **Experiment Number: 2**

Roll Number: 16010422012 **Name: Ayaan Bhoje**

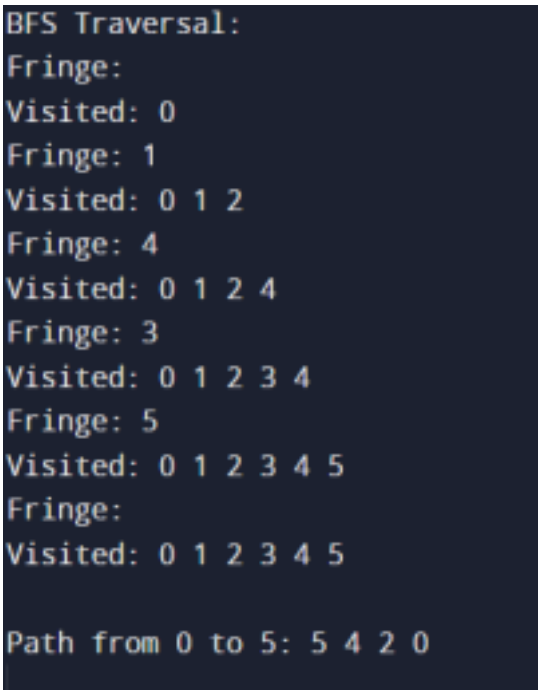
Aim of the Experiment: Graph Traversal by BFS

Program/ Steps:

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

#define MAX_VERTICES 100

// Structure to represent a node in the adjacency list
struct Node {
   int vertex;
   struct Node* next;
};

// Structure to represent the adjacency list
struct Graph {
   struct Node* adjacencyList[MAX_VERTICES];
   int visited[MAX_VERTICES];
};

// Function to create a new node
struct Node* createNode(int v) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->vertex = v;
   newNode->next = NULL;
   return newNode;
}

// Function to create a graph
struct Graph* createGraph() {
   struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
   for (int i = 0; i < MAX_VERTICES; ++i) {
      graph->adjacencyList[i] = NULL;
      graph->visited[i] = 0;
   }
   return graph;
}

// Function to add an edge to the graph
void addEdge(struct Graph* graph, int src, int dest) {
   // Add edge from src to dest
   struct Node* newNode = createNode(dest);
   newNode->next = graph->adjacencyList[src];
   graph->adjacencyList[src] = newNode;

   // Add edge from dest to src (since it's an undirected graph)
   newNode = createNode(src);
   newNode->next = graph->adjacencyList[dest];
   graph->adjacencyList[dest] = newNode;
}

// Function to perform BFS traversal
void BFS(struct Graph* graph, int startVertex) {
   // Create a queue for BFS
```

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

    // Enqueue the start vertex and mark it as visited
    queue[++rear] = startVertex;
    graph->visited[startVertex] = 1;

    // Loop until the queue is empty
    while (front != rear) {
      // Dequeue a vertex from the queue
      int currentVertex = queue[++front];
      printf("%d ", currentVertex);

      // Traverse all the adjacent vertices of the current vertex
      struct Node* temp = graph->adjacencyList[currentVertex];
      while (temp) {
        int adjVertex = temp->vertex;
        if (!graph->visited[adjVertex]) {
          // Mark the adjacent vertex as visited and enqueue it
          queue[++rear] = adjVertex;
          graph->visited[adjVertex] = 1;
        }
        temp = temp->next;
      }
    }
  }

  int main() {
    // Create a graph
    struct Graph* graph = createGraph();

    // Add edges to the graph
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 1, 4);
    addEdge(graph, 2, 5);

    printf("BFS Traversal starting from vertex 0:\n");
    BFS(graph, 0);

    return 0;
  }
```

Output/Result:



Outcomes: Analyse and formalize the problem

and select the best suited algorithm.

Conclusion:

The provided C code implements Breadth-First Search (BFS) for graph traversal, displaying the contents of the Fringeand Visited nodes at each iteration and finally printing the traversed path. It utilizes a simple graph representation through adjacency lists and employs a queue data structure

for BFS traversal

1. **References:** Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach,Second Edition, Pearson Publication

2. Luger, George F. Artificial Intelligence : Structures and strategies for complex problemsolving , 2009 ,6th Edition, Pearson Education