Laboratory 9

Title of the Laboratory Exercise: Graph algorithms

1. Introduction and Purpose of Experiment

   Graph algorithms such Depth First Search (DFS) and Breadth First Search (BFS) are very important in many graph applications which are used to traverse the graphs, check for graph connectivity and to find spanning trees etc. This experiment introduces the applications DFS and BFS.

2. Aim and Objectives

   Aim

   - To apply DFS and BFS algorithm for graph applications

   Objectives

   At the end of this lab, the student will be able to

   - Apply DFS and BFS for graph applications such as graph connectivity check
   - Compare the efficiency of both

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C/java language
   iv. Compile the C/java program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Design and develop algorithms to check whether given graph is connected or not using DFS and BFS. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

1. BFS Algorithm

```
Step 1 : Start

Step 2 : Start by putting any one of the graph's vertices at the back of a queue.

Step 3 : Take the front item of the queue and add it to the visited list.

Step 4 : Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

Step 5 : Keep repeating steps 2 and 3 until the queue is empty.

Step 6 : Stop
```

2. DFS Algorithm

```
Step 1 : Start

Step 2 : Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.

Step 3 : If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)

Step 4 : Repeat Rule 1 and Rule 2 until the stack is empty.

Step 5 : Stop
```

Implementation :

```c
#include <stdio.h>

#define MAX 10

int G[MAX][MAX]={}, visited[MAX]={}, n, queue[MAX]={}, front = 0, rear = -1;

void menu();
void reset_visited();
void read_adj_matrix();

// BFS and DFS
void DFS(int);
void BFS(int);

int main() {
    menu();

    return 0;
}
```

```c
void menu() {
    while (1) {
        int choice;
        printf("\nBFS and DFS on Graph"
               "\n1.\tEnter Adjacency Matrix"
               "\n2.\tDFS of the Graph"
               "\n3.\tBFS of the Graph"
               "\n4.\tExit"
               "\nYour Choice : ");
        scanf("%d", &choice);
        switch (choice) {
            case 1 :
                read_adj_matrix();
                break;
            case 2 :
                reset_visited();
                printf("\n");
                DFS(0);
                break;
            case 3 :
                reset_visited();
                BFS(0);
                printf("\nThe nodes which are reachable are : \n");
                for (int i = 0 ; i < n ; i++) {
                    if (visited[i]) {
                        printf(" %d", i);
                    } else {
                        printf("\n Not all nodes are reachable, BFS not possible.\n");
                        break;
                    }
                }
                break;
            case 4 :
                return;
            default :
                printf("\nWrong Choice !");
        }
    }
}

void reset_visited() {
    front = 0;
    rear = -1;
    for (int i = 0 ; i < MAX ; i++)
        visited[i] = 0;
}

void read_adj_matrix() {
    printf("\nEnter the number of vertices : ");
    scanf("%d", &n);

    printf("\nEnter the adjacency matrix of the graph : ");

    for (int i = 0 ; i < n ; i++) {
        for (int j = 0 ; j < n ; j++) {
            scanf("%d", &G[i][j]);
        }
    }
}

void DFS(int i) {
    printf(" %d", i);
    visited[i] = 1;

    for (int j = 0 ; j < n ; j++) {
        if (!visited[j] && G[i][j] == 1) {
            DFS(j);
        }
    }
}

void BFS(int v) {
    for (int i = 0 ; i < n ; i++) {
        if (!visited[i] && G[v][i]) {
            queue[++rear] = i;
        }
    }

    if (front <= rear) {
        visited[queue[front]] = 1;
        BFS(queue[front++]);
    }
}
```

6. Presentation of Results

```
BFS and DFS on Graph

1.    Enter Adjacency Matrix

2.    DFS of the Graph

3.    BFS of the Graph

4.    Exit

Your Choice : 1

Enter the number of vertices : 8

Enter the adjacency matrix of the graph :


0 1 1 1 1 0 0 0

1 0 0 0 0 1 0 0

1 0 0 0 0 1 0 0

1 0 0 0 0 0 1 0

1 0 0 0 0 0 1 0

0 1 1 0 0 0 0 1

0 0 0 1 1 0 0 1
```

```
0 0 0 0 0 1 1 0

BFS and DFS on Graph
1.    Enter Adjacency Matrix
2.    DFS of the Graph
3.    BFS of the Graph
4.    Exit
Your Choice : 2
 0 1 5 2 7 6 3 4

BFS and DFS on Graph
1.    Enter Adjacency Matrix
2.    DFS of the Graph
3.    BFS of the Graph
4.    Exit
Your Choice : 3
The nodes which are reachable are :
 0 1 2 3 4 5

BFS and DFS on Graph
1.    Enter Adjacency Matrix
2.    DFS of the Graph
3.    BFS of the Graph
4.    Exit
Your Choice : 4

Process finished with exit code 0
```

7. Analysis and Discussions

Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes to deeper and deeper until we find the goal node or the node which has no children. The algorithm, then backtracks from the dead end towards the most recent node that is yet to be completely unexplored.

The data structure which is being used in DFS is stack. The process is similar to BFS algorithm. In DFS, the edges that leads to an unvisited node are called discovery edges while the edges that leads to an already visited node are called block edges.

Breadth first search is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes. Then, it selects the nearest node and explore all the unexplored nodes. The algorithm follows the same process for each of the nearest node until it finds the goal.

The algorithm of breadth first search is given below. The algorithm starts with examining the node A and all of its neighbours. In the next step, the neighbours of the nearest node of A are explored and process continues in the further steps. The algorithm explores all neighbours of all the nodes and ensures that each node is visited exactly once and no node is visited twice.

8. Conclusions

Graph traversal means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once. The order in which the vertices are visited are important and may depend upon the algorithm or question that you are solving.

During a traversal, it is important that you track which vertices have been visited. The most common way of tracking vertices is to mark them.