**COMPUTER ENGINEERING**

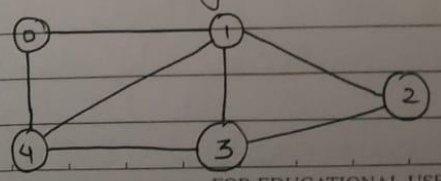
**DS ODD SEM 2021-22/EXPERIMENT 8**

**NAME:- GAURAV AMARNANI (D7A, 67)**

# Experiment - 8

AIM: Implement Graph Traversal Techniques
a) Depth first Search
b) Breadth first Search

Theory:
Graph is a data structure that consists of following two components:
1) A finite set of vertices also called as nodes
2) A finite set of ordered pairs of the form (U,V) called as edge. The pair is ordered because (U,V) is not as same as (V,U) in case of directed graph (di-graph). The pair of form (U,V) indicates that there is an edge form vertex U to vertex V. The edges may contain weight / value / cost.
Graphs are used to represent many real life applications. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are used in social networks like linked in facebooks for example in facebook each person is represented with a vertex (or node). Each node is a structure and contains information like person id, name, gender and location



## Graph Traversal:

Traversal means visiting all that nodes of a graph. There are two types of graph traversal
1) Depth first Search
2) Breadth first Search

## Depth first Search:

Depth first Search is a recursive algorithm for searching all the vertices of a graph of tree data structure. The algorithm starts at the root node and explores as far as possible along each branch before backtracking

## Breadth First Search:

Breadth first Search is a recursive algorithm for searching all the vertices of a graph or tree data structure. The only catch here is unlike trees, graphs may contain cycles, so we may come to the some node again. To avoid processing a node more than once, we were use a boolean visited array. for simplicity it is assumed that all vertices are reachable from the starting vertex.

Algorithm for Death First Search

Step 1 : SET STATE = 1 for each node in G

Step 2 : Push the Starting node A on the stack and set its STATUS = 2 (waiting State).

Step 3: Repeat step 4 and 5 until STACK is empty.

Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)

Step 5: Push on the stack all the neighbours of N that are in the ready state (whose status and set their STATUS = 2

[END OF LOOP]

Step 6: EXIT

Algorithm for Breadth First Search

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Enque the starting node A and set its STATUS = 2 (waiting state)

Step 3: Repeat step 4 and 5 until Queue is empty.

Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state)

Step 5: Enqueue all the neighbours of N that are in ready state and set their STATUS = 2

[End of LOOP]

Step 6: EXIT

Conclusion:

Hence by performing this experiment we learned about the concept of Graph data Structure and technique to implement Graph Traversal. We also learned to implement the techniques into code.

# Depth First Search:

```
#include<stdio.h>
#include<conio.h>
int source, V, E, time, visited[20],G[20][20];void DFS(int i){
int j; visited[i] = 1;
printf("%d->",i+1);
for(j=0;j<V;j++) {
        if(G[i][j]==1 && visited[j]==0)DFS(j);
} }
void main(){
int i,j,v1,v2; printf("\t\t\tGraphs\n");
printf("Enter the no of edges:");
scanf("%d",&E);
printf("Enter the no of vertices:");scanf("%d",&V);
for(i=0; i<V; i++) {
        for(j=0; j<V; j++)G[i][j]
            = 0;
}
for(i=0;i<E;i++) {
        printf("Enter the edges(format: V1 V2):");
        scanf("%d%d",&v1,&v2);
        G[v1-1][v2-1] = 1;
}
for(i=0;i<V;i++) {
        for(j=0; j<V; j++)
            printf("%d",G[i][j]);
        printf("\n");
}
printf("Enter the source: ");
    scanf("%d",&source);
        DFS(source-1);
}
```

```
                        Graphs
Enter the no of edges:5
Enter the no of vertices: 5
Enter the edges(format: V1 V2):1 2
Enter the edges(format: V1 V2):2 3
Enter the edges(format: V1 V2):3 4
Enter the edges(format: V1 V2):4 5
Enter the edges(format: V1 V2):5 6
01000
00100
00010
00001
00000
Enter the source: 5
5->                     Graphs
```

# Breadth First Search:

```
#include<stdio.h>
#include<conio.h>
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;void bfs(int v) {
	for(i = 1; i <= n; i++)
		if(a[v][i] && !visited[i])q[++r] =
			i;
		if(f <= r) {
			visited[q[f]] = 1;
			bfs(q[f++]);
		}
}

void main() {
	clrscr();
	int v;
	printf("Enter the number of vertices: ");scanf("%d",&n);
	for(i=1; i <= n; i++) {q[i] = 0;
		visited[i] = 0;
	}
	printf("\nEnter graph data in matrix form:\n");for(i=1; i<=n; i++) {
		for(j=1;j<=n;j++) {
			scanf("%d", &a[i][j]);
		}
	}
	printf("Enter the starting vertex: ");scanf("%d", &v);
	bfs(v);
	printf("\nThe node which are reachable are:");for(i=1; i <= n; i++) {
		if(visited[i])
			printf(" %d", i);
		else {
			printf("\nBFS is not possible. All nodes are not reachable!");break;

		}
	} getch(); }
```

```
Enter the number of vertices: 3

Enter graph data in matrix form:
2
4
5
2
3
4
1
7
8
Enter the starting vertex: 2

The node which are reachable are: 1 2 3_
```