**Experiment 10**

Aim : Implement Graph Traversal Techniques:

1. Depth First Search b) Breadth First Search

**Code:-**

a) Depth First Search:

#include <stdio.h>

#define MAX 5

**void** depth\_first\_search(**int** adj[][MAX],**int** visited[],**int** start)

{

**int** stack[MAX];

**int** top = -1, i;

printf("%c->",start+65);

visited[start] = 1;

stack[++top] = start;

**while**(top != -1)

{

start = stack[top];

**for**(i = 0; i < MAX; i++)

{

**if**(adj[start][i] && visited[i] == 0)

{

stack[++top] = i;

printf("%c->", i+65);

visited[i] = 1;

**break**;

}

}

**if**(i == MAX)

top--;

}

}

**int** main()

{

**int** adj[MAX][MAX];

**int** visited[MAX] = {0}, i, j;

printf("\n Enter the adjacency matrix: ");

**for**(i = 0; i < MAX; i++)

**for**(j = 0; j < MAX; j++)

scanf("%d", &adj[i][j]);

printf("DFS Traversal: ");

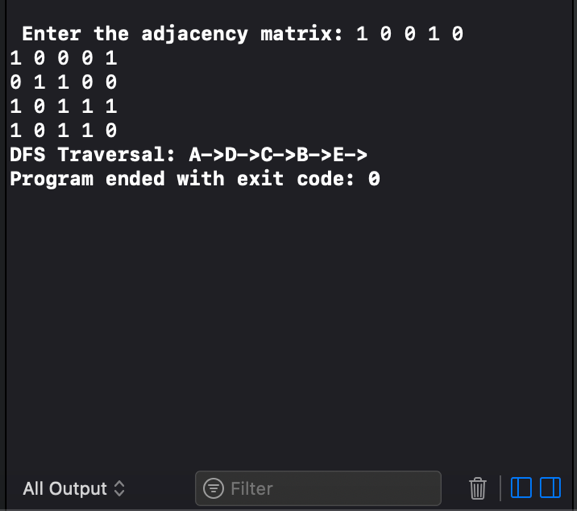
depth\_first\_search(adj,visited,0);

printf("\n");

**return** 0;

}

**Output:-**

****

b) Breadth First Search:

#include <stdio.h>

#define MAX 5

**void** breadth\_first\_search(**int** adj[][MAX],**int** visited[],**int** start)

{

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

queue[++rear] = start;

visited[start] = 1;

**while**(rear != front)

{

start = queue[++front];

**if**(start == 4)

printf("E->");

**else**

printf("%c->",start + 65);

**for**(i = 0; i < MAX; i++)

{

**if**(adj[start][i] == 1 && visited[i] == 0)

{

queue[++rear] = i;

visited[i] = 1;

}

}

}

}

**int** main()

{

**int** visited[MAX] = {0};

**int** adj[MAX][MAX], i, j;

printf("\n Enter the adjacency matrix: ");

**for**(i = 0; i < MAX; i++)

**for**(j = 0; j < MAX; j++)

scanf("%d", &adj[i][j]);

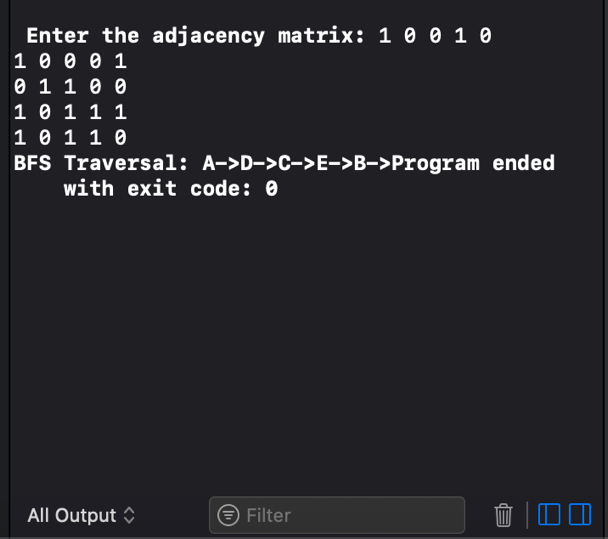
printf("BFS Traversal: ");

breadth\_first\_search(adj,visited,0);

**return** 0;

}

**Output:-**

****

**Conclusion:-** We thus have implemented graph Traversal techniques